1С ПАБЛИШИНГ

Е.Ю. Хрусталева

Знакомство с разработкой мобильных приложений

на платформе «1С:Предприятие 8»

Требуется доступ к Интернету

Издание 3

БИБЛИОТЕКА РАЗРАБОТЧИКА

Хрусталева Е. Ю.

Знакомство с разработкой мобильных приложений на платформе «1С:Предприятие 8». 3-е издание

Электронная книга в формате pdf; ISBN 978-5-9677-3191-4.

Электронный аналог печатного издания «Знакомство с разработкой мобильных приложений на платформе "1С:Предприятие 8"». 3-е издание (ISBN 978-5-9677-3181-5) М.: ООО «1С-Паблишинг», 2022; артикул печатной книги по прайс-листу фирмы «1С»: 4601546146441;
по вопросам приобретения печатных изданий издательства «1С-Паблишинг» обращайтесь к партнеру «1С», обслуживающему вашу организацию, или к другим партнерам фирмы «1С», в магазины «1С Интерес», а также в книжные и интернет-магазины).

Книга адресована разработчикам прикладных решений в системе «1С:Предприятие 8», которые хотят научиться создавать приложения на платформе для мобильных устройств. Такие приложения могут работать на смартфонах и планшетных ПК под управлением операционных систем Android, iOS или Windows.

На примере создания нескольких реальных мобильных приложений показаны особенности разработки: подготовка мобильного устройства, установка платформы разработчика, взаимодействие с устройством в процессе разработки, сборка готового мобильного приложения для ОС Android.

Третье издание книги адаптировано к версии платформы 8.3.20 и демонстрирует ряд ее новых возможностей:

- разработку приложения мобильного клиента;
- разработку приложения мобильного клиента с автономным режимом;
- адаптацию интерфейса настольного приложения для работы на мобильном устройстве;
- использование облачного сервиса сборки мобильных приложений.

Книга содержит большое количество рисунков и примеров кода на встроенном языке, снабженных подробными комментариями. Для создания демонстрационных примеров использованы следующие версии платформ:

- «1С:Предприятие 8.3», учебная версия (8.3.20.1479);
- мобильная платформа «1С:Предприятия» (8.3.19.51).

Дополнительные материалы (базы данных) вы можете скачать по адресу https://its.1c.ru/bmk/mobile83.

Книга выпущена под редакцией Максима Радченко.

Оглавление

Введение

Организация бизнеса в современном мире все чаще требует наличия удаленного доступа пользователей к данным и функционалу систем по управлению хозяйственной деятельностью предприятия.

Например, руководителям и различным «управленцам» необходимо, не находясь в офисе, быстро просмотреть какие-то важные отчеты и документы на смартфоне по дороге на совещание.

Многим пользователям важно иметь полный доступ ко всему функционалу «офисного» прикладного решения на своем мобильном устройстве. Они должны иметь возможность оперативно внести в офисную базу результаты своей деятельности, чтобы в офисе не дожидались их возвращения. При отсутствии надежного интернет-соединения функциональность этих мобильных приложений уменьшается, а затем, при появлении соединения, восстанавливается.

С другой стороны, все больше рядовых сотрудников работают «на выезде» у клиентов: курьеры интернет-магазинов, торговые агенты и т. д. Этим пользователям не нужен доступ к полнофункциональной версии офисного приложения. Им достаточно иметь при себе мобильное устройство (смартфон или планшет) с установленным на нем приложением для решения ограниченного круга задач.

В этой книге пошагово и подробно рассмотрен процесс разработки трех вариантов мобильных приложений: приложения мобильного клиента, приложения мобильного клиента с автономным режимом и приложения мобильной платформы, каждое из которых предназначено для решения определенного круга задач.

Что такое платформа для мобильных устройств

Платформа для мобильных устройств содержит компоненты для отладки и сборки мобильных приложений, работающих на устройствах с операционными системами Android, iOS или Windows.

Существует три варианта платформы для мобильных устройств. Их не следует путать. Эти варианты различаются прежде всего способом взаимодействия с информационными базами, а также объемом функциональности, доступной мобильным приложениям, собранным на их основе:

- *Мобильный клиент* – позволяет взаимодействовать с информационными базами онлайн, аналогично тому, как это делают клиентские приложения платформы для «настольных» компьютеров. При этом на мобильном устройстве будет доступна вся функциональность «офисного» прикладного решения.
- *Мобильный клиент с автономным режимом* – в зависимости от наличия соединения позволяет либо взаимодействовать с информационными базами онлайн, либо использовать для работы автономную информационную базу на мобильном устройстве. При этом на мобильном устройстве может быть доступна либо вся функциональность «офисного» прикладного решения, либо только его автономная часть (в зависимости от качества соединения или по выбору пользователя).

- *Мобильная платформа* – использует для работы только автономную информационную базу на мобильном устройстве. При этом мобильное приложение будет иметь собственную функциональность, которая, вообще говоря, не зависит от функциональности «офисного» прикладного решения. Она определяется только конфигурацией самого мобильного приложения. Работа в таких приложениях ведется в режиме офлайн, а при появлении связи или по возвращении в офис выполняется обмен данными с офисным приложением.

Таким образом, в зависимости от потребностей пользователя и требующейся ему функциональности приложения, а также от его возможностей соединения и работы с информационной базой можно выбрать тот вариант мобильного приложения, который ему больше подходит.

Прежде чем двигаться дальше, важно понимать особенности этих трех видов платформы для мобильных устройств. Ниже о каждом из вариантов будет рассказано подробнее, чтобы вы могли решить, какой из них лучше применить в том или ином случае.

Мобильный клиент

С помощью этого варианта платформы можно собрать мобильное приложение, используя которое, «удаленный» пользователь на своем мобильном устройстве сможет работать с информационной базой так же, как если бы он работал в офисе за стационарным компьютером. При этом он будет вносить данные в ту же информационную базу, подключаться к которой сможет только онлайн.

Если проводить аналогию с платформой для настольных компьютеров, то такое мобильное приложение является аналогом тонкого клиента, работающего с информационной базой, опубликованной на веб-сервере.

Архитектуру мобильного клиента можно представить следующим образом (рис. 1.1).

Таким образом, с помощью этого варианта платформы можно достаточно легко и быстро, с минимальными усилиями разработчика собрать мобильное приложение, имеющее всю функциональность офисного приложения и работающее с информационной базой в режиме онлайн.

Нужно только следить за тем, чтобы интерфейс мобильного приложения обеспечивал комфортную работу на любых мобильных устройствах с любым размером и расположением экрана.

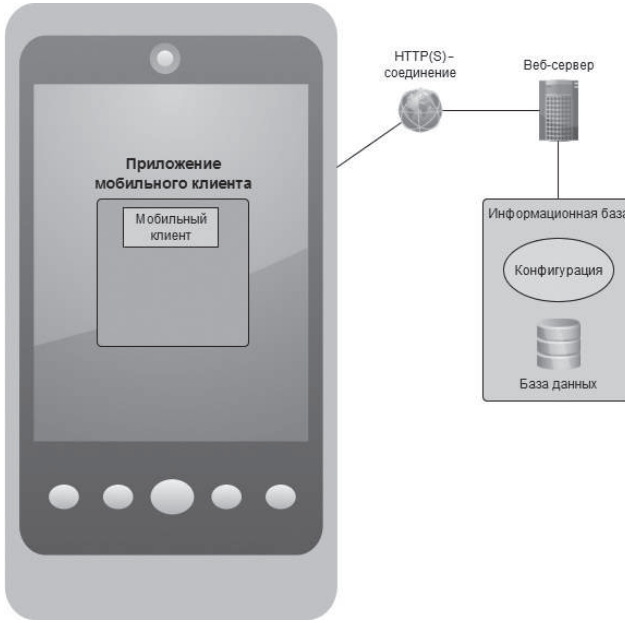


Рис. 1.1. Архитектура мобильного клиента

Для этого, особенно при создании мобильных клиентов для крупных прикладных решений, вам необходимо предпринимать дополнительные усилия. О разработке мобильного клиента будет рассказано во второй главе книги – «Приложение мобильного клиента».

Мобильный клиент с автономным режимом

Мобильное приложение, собранное с помощью данного варианта платформы, это более комфортный мобильный клиент, с помощью которого пользователь может:

- Работать с офисной информационной базой онлайн и с полной функциональностью прикладного решения при хорошем HTTP-соединении. Таким образом реализуется сценарий работы приложения мобильного клиента, описанный выше.
- Работать офлайн, когда нет связи с офисной базой. При этом используются автономная конфигурация и данные, которые хранятся

в автономной информационной базе на мобильном устройстве. А при восстановлении соединения данные между мобильным устройством и серверной информационной базой синхронизируются.

- В случае плохой связи пользователь может выбрать режим работы (онлайн или офлайн).

Архитектуру мобильного клиента с автономным режимом можно представить следующим образом (рис. 1.2).

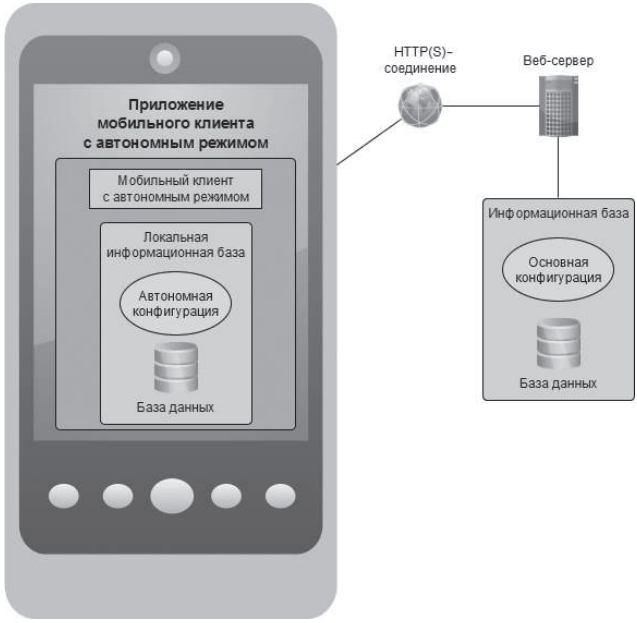


Рис. 1.2. Архитектура мобильного клиента с автономным режимом

Таким образом, создание мобильных приложений с помощью этого варианта платформы потребует больших усилий по сравнению с предыдущим вариантом, так как придется выделять часть функциональности в автономную конфигурацию и реализовывать блок обмена данными с офисным приложением. Но все же это значительно проще, чем разрабатывать конфигурацию для мобильного приложения с нуля. О создании мобильного клиента с автономным режимом будет рассказано в третьей главе книги – «Приложение мобильного клиента с автономным режимом».

Мобильная платформа

Мобильное приложение, собранное с помощью этого варианта платформы, может работать автономно на мобильном устройстве. При этом для функционирования мобильного приложения специально разрабатывается отдельная конфигурация, совершенно не связанная с конфигурацией «офисного» прикладного решения.

Если проводить аналогию с платформой для настольных компьютеров, то такое мобильное приложение является аналогом тонкого клиента, работающего с файловой информационной базой, которая находится на том же компьютере, что и тонкий клиент.

Архитектуру мобильной платформы можно представить следующим образом (рис. 1.3).

Рис. 1.3. Архитектура мобильной платформы

Работа в приложениях мобильной платформы ведется в режиме офлайн, а при появлении связи или по возвращении в офис выполняется обмен данными с офисной базой средствами платформы (рис. 1.4).

Рис. 1.4. Обмен данными между офисным приложением и приложением мобильной платформы

Таким образом, помимо разработки конфигурации для мобильного приложения на вас ложится еще и разработка обмена данными как в мобильном приложении, так и в офисном.

В результате создание мобильных приложений с помощью этого варианта платформы – самый трудоемкий вариант, требующий значительных ресурсов. Его имеет смысл использовать в тех случаях, когда работа в основном ведется автономно и функциональность мобильного приложения сильно отличается от функциональности офисного.

Если нет задачи использовать в мобильном приложении какую-то функциональность помимо той, что есть в офисном приложении, то проще использовать мобильный клиент с автономным режимом. При этом пользователь также сможет работать офлайн с выделенной ему автономной частью конфигурации и вносить данные в автономную базу на мобильном устройстве, а при возможности будет произведен обмен данными с офисной базой.

О разработке приложения мобильной платформы будет рассказано в четвертой главе книги – «Приложение мобильной платформы».

Разработка, сборка и публикация мобильных приложений

При разработке конфигураций для мобильных устройств, так же как и при разработке «настольных» приложений, используется конфигуратор. С одной стороны, функциональность платформы для мобильных устройств меньше, чем функциональность «настольной» платформы. Эту особенность нужно учитывать при разработке. Текущий состав ограничений описан в документации (https://its.1c.ru/db/v83doc#bookmark:dev:TI000001284).

С другой стороны, платформа для мобильных устройств поддерживает функции, доступные только на мобильных устройствах: доступ к фотокамере, геопозиционированию, PUSH-уведомлениям и т. д.

Когда разработка конфигурации для любого из трех видов (описанных выше) мобильных приложений завершена, вы можете собрать дистрибутивы мобильного приложения для операционных систем Android, iOS и Windows и опубликовать их в магазинах приложений. Для этого используется облачный сервис сборки и публикации мобильных приложений (доступен начиная с версии платформы «1С:Предприятие» 8.3.20).

После этого пользователи, которым раздали установочный apk-файл, могут установить мобильное приложение на свое устройство или же скачать его из App Store, Google Play или Windows Phone Store.

ПОДРОБНЕЕ

Документация 1С:Предприятие 8.3.20. Руководство разработчика. Глава 29. Разработка для мобильных устройств.

Постановка задачи

В данной книге будет описана реализация взаимодействия между курьером интернет-магазина, работающим на мобильном устройстве (телефоне или планшете), и офисом самого интернет-магазина, где работает офисное приложение на стационарном компьютере.

Офисное приложение будет одно, хотя вариантов мобильных приложений, которые могут с ним взаимодействовать, будет три: приложение мобильного клиента, приложение мобильного клиента с автономным режимом и приложение мобильной платформы. Создание этих приложений будет описано соответственно во второй, третьей и четвертой главах книги.

Одно общее офисное приложение, во-первых, упростит задачу, а во-вторых, позволит вам легче разобраться в видах разрабатываемых мобильных приложений, понять их назначение и отличие друг от друга.

Функциональность офисного приложения

Разработка офисного приложения не входит в задачу этой книги. Для простоты будет использована готовая конфигурация, которая лишь схематично и упрощенно отражает основные бизнес-процессы интернет-магазина по продаже одежды, которые необходимы для совместной работы курьера интернет-магазина и офиса.

ПРИМЕЧАНИЕ

Эта конфигурация находится в архиве, который вы можете скачать по адресу https://its.1c.ru/bmk/mobile83. Распакуйте архив, создайте пустую информационную базу «1С:Предприятия» и в режиме Конфигуратор загрузите в нее информационную базу из файла «Интернет магазин.dt» (Администрирование – Загрузить информационную базу).

Сначала интернет-магазин получает товары на склад. Затем в офисе создаются заказы клиентов. После выполнения заказов курьером на основании заказов создаются документы продаж товаров.

Затем документы проводятся и производят движения в регистрах накопления, на основе которых формируются отчеты об остатках товаров, о причинах отказа от товаров, о продажах клиентам и т. п.

Для учета поступлений, заказов товаров и их продаж клиентам используются справочники товаров, складов, клиентов. Товары приходуются и расходуются в разрезе цветов и размеров, которые хранятся в соответствующих справочниках. Также ведется справочник для учета причин отказа от покупки товаров. Список курьеров, обслуживающих заказы, хранится в справочнике пользователей.

Итак, в офисном приложении реализованы следующие функции:

- При поступлении товаров на склад менеджер интернет-магазина создает приходные накладные. Суммы по товару автоматически пересчитываются при изменении его цены и количества. Поступившие товары учитываются в разрезе складов, размеров и цветов.
- Полученные товары заносятся в справочник товаров. Для каждого товара можно выбрать картинку, которая находится в справочнике хранимых файлов.
- Менеджер устанавливает цены товаров на определенную дату, по которым они будут продаваться. Эти цены хранятся в периодическом регистре сведений.

- При создании заказа менеджер вводит данные клиента, дату доставки, отмечает важность и статус заказа (Открыт). Адрес доставки заполняется автоматически из данных клиента. При заполнении заказа актуальные цены товаров подставляются в заказ. Суммы по строкам табличной части автоматически пересчитываются при изменении товара, его цены и количества.
- Менеджер интернет-магазина создает заказ со статусом Открыт, затем указывает курьера, который будет обслуживать заказ, и устанавливает статус заказа В работе. После выполнения заказа курьером менеджер проверяет заказ со статусом Выполнен и присваивает ему статус Закрыт. Заказанные товары учитываются в разрезе складов, размеров и цветов.
- Редактирование ранее созданных заказов возможно только в том случае, если заказ имеет статус Открыт или В работе. Заказ со статусом Выполнен можно только закрыть, а заказ со статусом Закрыт можно только просматривать.
- Список заказов отображается с условным оформлением в зависимости от статуса заказа и его важности.
- В офисном приложении реализована работа с хранимыми файлами: выбор файла из локальной файловой системы, запись в справочник хранимых файлов, чтение файла из справочника и его открытие или запись в файловую систему пользователя. В основном справочник хранимых файлов используется для хранения и выбора картинок товаров.
- На основе закрытых заказов создаются расходные накладные со списком товаров, которые клиент купил. Кроме того, менеджер может ввести расходную накладную вручную. Суммы по товару автоматически пересчитываются при изменении товара, его цены и количества. Проданные товары учитываются в разрезе клиентов и курьеров.
- Для анализа работы интернет-магазина в офисе строятся различные отчеты: об остатках товаров на складах в разрезе цветов и размеров, о причинах отказа от товаров и о продажах товаров клиентам.

Функциональность мобильного приложения

Курьер с помощью мобильного приложения должен выполнять следующую последовательность действий:

- Курьер получает из интернет-магазина список заказов в порядке их обслуживания (см. раздел 4-й главы «Обмен данными через Web-сервис»). Затем он забирает заказанные товары со склада.

- Курьер открывает первый заказ из этого списка и определяет местоположение клиента, сделавшего заказ, на карте (см. раздел 4-й главы «Геопозиционирование»).
- Звонит клиенту и сообщает о том, что он к нему едет. Возможно, он не дозванивается или задерживается, тогда курьер может послать клиенту SMS (см. раздел 4-й главы «Связь с клиентом»).
- Приезжает к клиенту. После знакомства с товарами клиент какие-то позиции заказа берет, а какие-то – нет.
- Курьер отмечает факт отказа от товаров в заказе. При этом у возвращенных товаров указывается причина отказа (например, не подошел размер, цвет и т. п.).
- В случае несоответствия товара по цвету или качеству курьер может сделать его фото- и видеосъемку, а также записать аудиоотзыв клиента о заказе (см. раздел 2-й главы «Работа с мультимедиа» и раздел 4-й главы «Работа со средствами мультимедиа»).
- Курьер может добавить новый заказ на этого же или нового клиента, если возникла такая необходимость (например, нужно перезаказать тот же товар, но другого размера).
- В случае необходимости курьер может создавать для себя напоминания (например, напоминание о звонке клиенту), которые будут появляться в указанное время или повторяться периодически (см. раздел 4-й главы «Напоминание о звонке»). Кроме того, курьер может отправить клиенту письмо по электронной почте (см. раздел 4-й главы «Электронная почта»).
- После окончания обслуживания заказа курьер ставит отметку о его выполнении (присваивает заказу статус Выполнен).
- При необходимости курьер может сформировать и проанализировать данные в отчетах. Причем эти отчеты могут быть построены как на данных мобильного приложения (см. раздел 4-й главы «Отчет по данным мобильного приложения»), так и удаленно в офисном приложении и переданы курьеру через веб-сервис (см. раздел 4-й главы «Формирование удаленного отчета по данным офисного приложения»).
- Интернет-магазин может посылать курьеру сообщения – например, если в офисе изменилась важная для курьера информация (см. раздел второй главы «Работа с Push-уведомлениями»).
- Затем курьер находит следующий по порядку заказ в списке обслуживания, открывает данные клиента, сделавшего заказ, и т. д.

В этой книге для упрощения примера функциональность, специфичная для мобильного устройства (геопозиционирование, звонки, СМС, работа с мультимедиа и т. п.), будет показана в основном в четвертой главе при разработке приложения мобильной платформы и частично во второй главе, где рассказывается про возможности мобильных устройств в мобильном клиенте. Чтобы этой функциональностью можно было пользоваться и в приложениях мобильного клиента, и мобильного клиента с автономным режимом, все то же самое нужно реализовать в офисной конфигурации. Но так как офисная конфигурация работает и на настольных компьютерах, то все «мобильные» фрагменты кода нужно обернуть в инструкцию препроцессора #Если МобильныйКлиент.

Глава 1. Подготовка планшета и компьютера

Для разработки мобильных приложений, показанных в книге, использовался планшет Samsung Galaxy Tab A (2018, 10,5) SM-T595, работающий под управлением операционной системы Android 10.

При изучении этой книги каждый из видов мобильного приложения вы будете разрабатывать в конфигураторе «1С:Предприятия» на стационарном компьютере. В процессе разработки вы будете запускать приложение в пользовательском режиме на планшете и смотреть, как оно выглядит и реализует свои функции.

Чтобы были возможны запуск и отладка приложения, на планшете должна быть установлена мобильная платформа разработчика (каждого из трех видов). Поэтому, прежде чем начинать разрабатывать мобильное приложение, нужно подготовить стационарный компьютер, установить платформы разработчика на планшет и настроить связь между компьютером и планшетом. Эти действия необходимы для разработки любого мобильного приложения и никак не связаны с прикладной спецификой рассматриваемых примеров.

Подготовка стационарного компьютера

Платформа «1С:Предприятие»

На стационарном компьютере должна быть установлена «боевая» или учебная версия платформы «1С:Предприятие», содержащая компонент «Модули расширения веб-сервера». Этот компонент необходим, чтобы опубликовать мобильное приложение на веб-сервере.

Если у вас нет платформы, можно бесплатно скачать учебную версию платформы «1С:Предприятие» по адресу https://online.1c.ru/catalog/free/learning.php#platform. Там же можно скачать и платформу для мобильных устройств.

При написании данной книги использовалась платформа «1С:Предприятие» версии 8.3.20.1479 и платформа для мобильных устройств версии 8.3.19.51.

Если платформа «1С:Предприятие» уже имеется на компьютере, вы можете проверить, какие компоненты у нее установлены. Для этого откройте список установленных на компьютере программ (Панель управления > Программы > Программы и компоненты), выделите используемую версию «1С:Предприятия» и нажмите Изменить. Появится диалог обслуживания программ платформы с помеченной секцией Изменить (рис. 1.5).

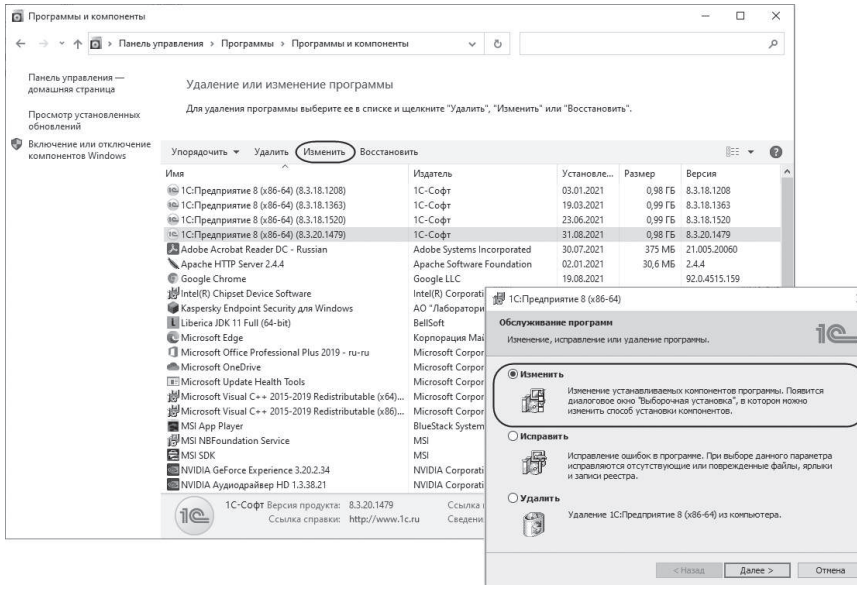


Рис. 1.5. Диалог обслуживания программ «1С:Предприятия»

В этом диалоге нажмите Далее. В появившемся списке компонентов платформы убедитесь, что компонент Модули расширения веб-сервера установлен. В этом случае можно нажать Отмена и выйти из диалога выборочной установки. В противном случае сделайте компонент доступным, нажмите Далее и продолжите установку платформы, как обычно (рис. 1.6).

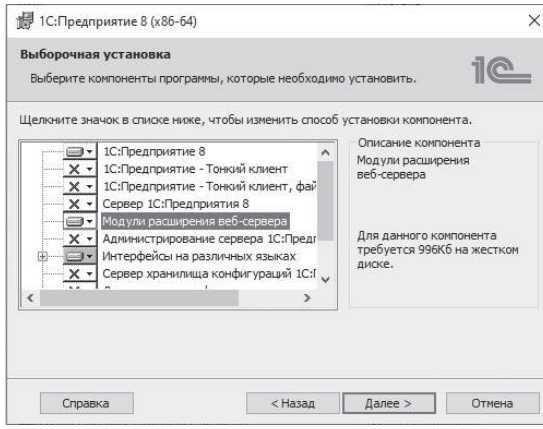


Рис. 1.6. Выборочная установка компонентов платформы «1С:Предприятия»

Установка веб-сервера

В процессе запуска и отладки мобильного приложения его взаимодействие со стационарным компьютером будет осуществляться через веб-сервер. При написании данной книги использовался веб-сервер Apache.

Если на вашем компьютере уже установлен локальный веб-сервер Internet Information Services (IIS), то можно использовать его.

Если вы хотите, как в книге, использовать Apache, то он может конфликтовать с работающим IIS. В этом случае лучше остановить или удалить IIS из состава компонентов Windows (Панель управления > Программы > Программы и компоненты > Включение или отключение компонентов Windows) – рис. 1.7.

Скачать установочный пакет MSI Installer для веб-сервера Apache можно по адресу: http://archive.apache.org/dist/httpd/binaries/win32/.

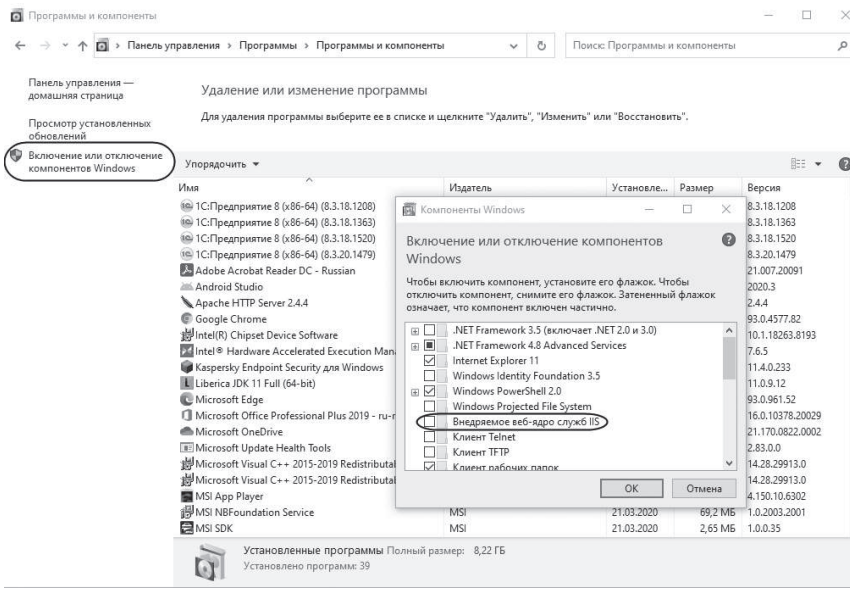


Рис. 1.7. Удаление IIS из состава компонентов Windows

На открывшейся странице находятся ссылки на дистрибутивы Apache для Windows. Чтобы быстро найти наиболее свежую версию дистрибутива, можно щелкнуть два раза по заголовку колонки Last modified, затем выбрать и скачать дистрибутив httpd-2.2.25-win32-x86-no_ssl (рис. 1.8).

Затем нужно сохранить установочный пакет, запустить его и установить веб-сервер Apache на компьютер. После установки значок веб-сервера появится в нижнем правом углу монитора, в списке загруженных программ в панели задач.

Name	Last modified	Size	Description
Parent Directory		-	
symbols/	2020-07-06 14:16	-	
patches_applied/	2020-07-06 14:16	-	
LEGACY.html	2020-07-06 14:17	5.6K	
TROUBLESHOOTING.html	2020-07-06 14:16	2.7K	
httpd-2.2.25-win32-x86-openssl-0.9.8y.msi.sha1	2013-07-10 08:06	83	
httpd-2.2.25-win32-x86-openssl-0.9.8y.msi.md5	2013-07-10 08:06	76	
httpd-2.2.25-win32-x86-openssl-0.9.8y.msi.asc	2013-07-10 08:06	835	
httpd-2.2.25-win32-x86-openssl-0.9.8y.msi	2013-07-10 08:06	6.1M	
httpd-2.2.25-win32-x86-no_ssl.msi.sha1	2013-07-10 08:06	75	
httpd-2.2.25-win32-x86-no_ssl.msi.md5	2013-07-10 08:06	68	
httpd-2.2.25-win32-x86-no_ssl.msi.asc	2013-07-10 08:06	835	
httpd-2.2.25-win32-x86-no_ssl.msi	2013-07-10 08:06	5.5M	
httpd-2.0.65-win32-x86-openssl-0.9.8y.msi.sha1	2013-07-10 08:06	83	
httpd-2.0.65-win32-x86-openssl-0.9.8y.msi.md5	2013-07-10 08:06	76	
httpd-2.0.65-win32-x86-openssl-0.9.8y.msi.asc	2013-07-10 08:06	835	
httpd-2.0.65-win32-x86-openssl-0.9.8y.msi	2013-07-10 08:06	5.5M	
httpd-2.0.65-win32-x86-no_ssl.msi.sha1	2013-07-10 08:06	75	
httpd-2.0.65-win32-x86-no_ssl.msi.md5	2013-07-10 08:06	68	
httpd-2.0.65-win32-x86-no_ssl.msi.asc	2013-07-10 08:06	835	
httpd-2.0.65-win32-x86-no_ssl.msi	2013-07-10 08:06	4.8M	
httpd-2.2.22-win32-x86-openssl-0.9.8t.msi.sha1	2012-01-30 22:06	90	
httpd-2.2.22-win32-x86-openssl-0.9.8t.msi.md5	2012-01-30 22:06	76	
httpd-2.2.22-win32-x86-openssl-0.9.8t.msi.asc	2012-01-30 22:06	833	
httpd-2.2.22-win32-x86-openssl-0.9.8t.msi	2012-01-30 22:06	6.1M	
httpd-2.2.22-win32-x86-no_ssl.msi.sha1	2012-01-30 22:06	82	
httpd-2.2.22-win32-x86-no_ssl.msi.md5	2012-01-30 22:06	68	
httpd-2.2.22-win32-x86-no_ssl.msi.asc	2012-01-30 22:06	833	
httpd-2.2.22-win32-x86-no_ssl.msi	2012-01-30 22:06	5.4M	

Рис. 1.8. Установочный пакет веб-сервера Apache для Windows

Android SDK

Планшет, который используется для примера в этой книге, работает под управлением ОС Android. Поэтому для разработки мобильных приложений на стационарный компьютер нужно установить еще среду разработки Android Studio и настроить в ней нужную версию Android SDK.

Для этого перейдите по ссылке http://developer.android.com/studio/index.html, на открывшейся странице нажмите кнопку Download Android Studio, согласитесь с правилами и условиями (установите флажок I have read and agree…) и скачайте файл android-studio-2020.3.1.23-windows.exe (рис. 1.9).

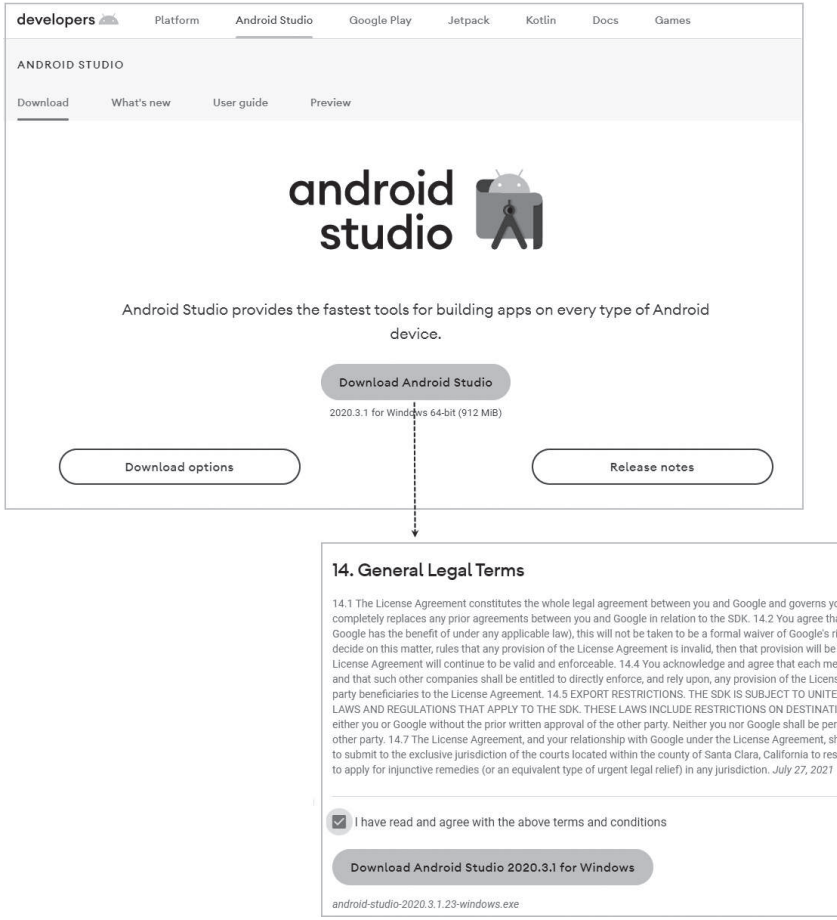


Рис. 1.9. Установка Android Studio

Затем запустите полученный файл, установите на компьютере Android Studio и откройте его. В стартовом окне выделите в списке меню строку Customize и нажмите Configure… (рис. 1.10).

После этого раскройте пункт меню Appearance & Behavior, а затем пункты System Settings и Android SDK. На закладке SDK Platforms установите флажок Show Package Details и отметьте все версии SDK Platforms, которые вы хотите установить. Необходимо, чтобы на компьютере был установлен Android SDK Platform версии не ниже 26 (рис. 1.11).

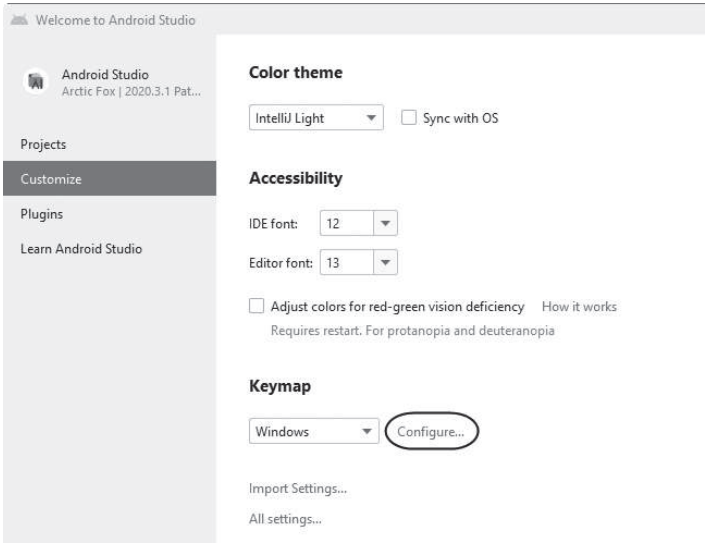


Рис. 1.10. Конфигурирование Android Studio

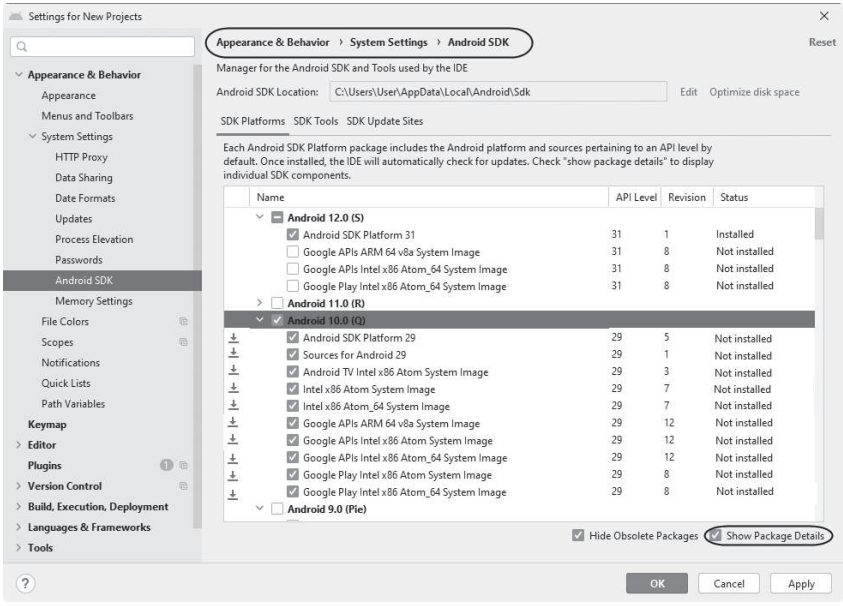


Рис. 1.11. Выборочная установка компонентов SDK Platforms

На закладке SDK Tools убедитесь, что номера помеченных версий Android SDK Platforms-Tools и Android SDK Build-Tools согласованы друг с другом и что к установке не помечены временные и beta-версии. В данном случае все можно оставить без изменений (рис. 1.12).

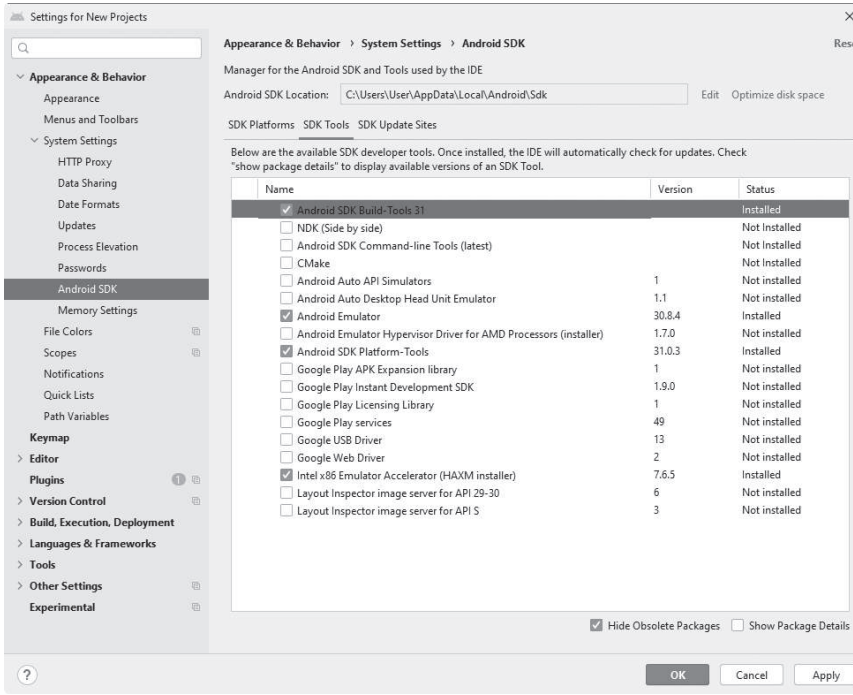


Рис. 1.12. Выборочная установка компонентов SDK Tools

Нажмите OK и еще раз подтвердите список выбранных для установки компонентов Android SDK (рис. 1.13).

Затем выделите каждый из трех видов лицензий и отметьте опцию Accept, примите все лицензионные соглашения и нажмите кнопку Next (рис. 1.14).

Начнется процесс установки выбранных вами компонентов Android SDK, который может занять длительное время.

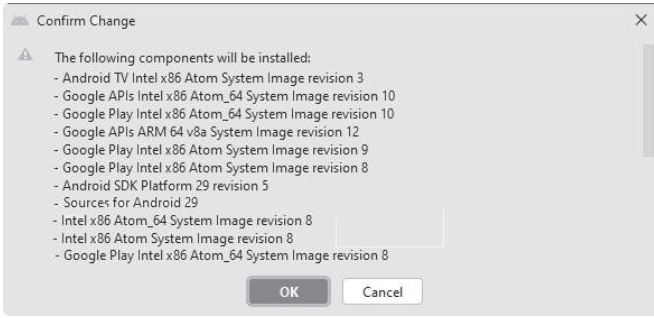


Рис. 1.13. Выборочная установка компонентов Android SDK

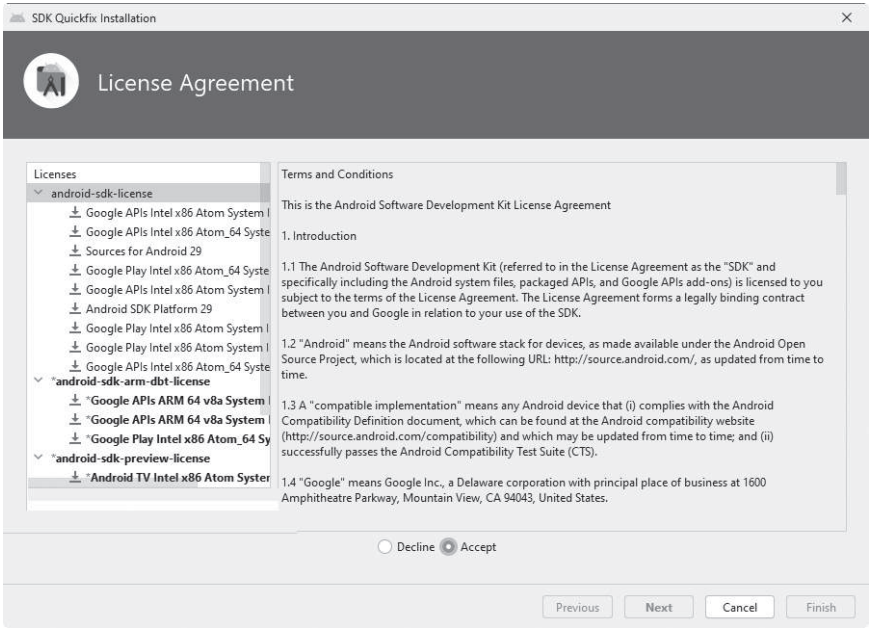


Рис. 1.14. Выборочная установка компонентов Android SDK

Подготовка планшета

Настройки планшета

Чтобы планшет мог взаимодействовать с настольным компьютером, нужно изменить некоторые его настройки. Откройте список настроек, вызвав приложение Настройки в списке приложений планшета.

ПРИМЕЧАНИЕ

Приведенная далее последовательность действий может быть неприменима к другим Android-устройствам. Это зависит и от версии Android, и от производителя устройства. Поэтому в каждом конкретном случае нужно искать в Интернете, каким способом устанавливаются перечисленные настройки.

Для установки на планшете платформ разработчика вам нужно разрешить установку приложений из неизвестных источников. Поскольку apk-файлы будут передаваться на планшет с компьютера по USB-кабелю, то этот источник – приложение Мои файлы.

Для этого раскройте группу настроек Биометрия и безопасность, затем откройте настройку Установка неизвестных приложений и включите разрешение для источника Мои файлы (рис. 1.15, 1.16, 1.17).

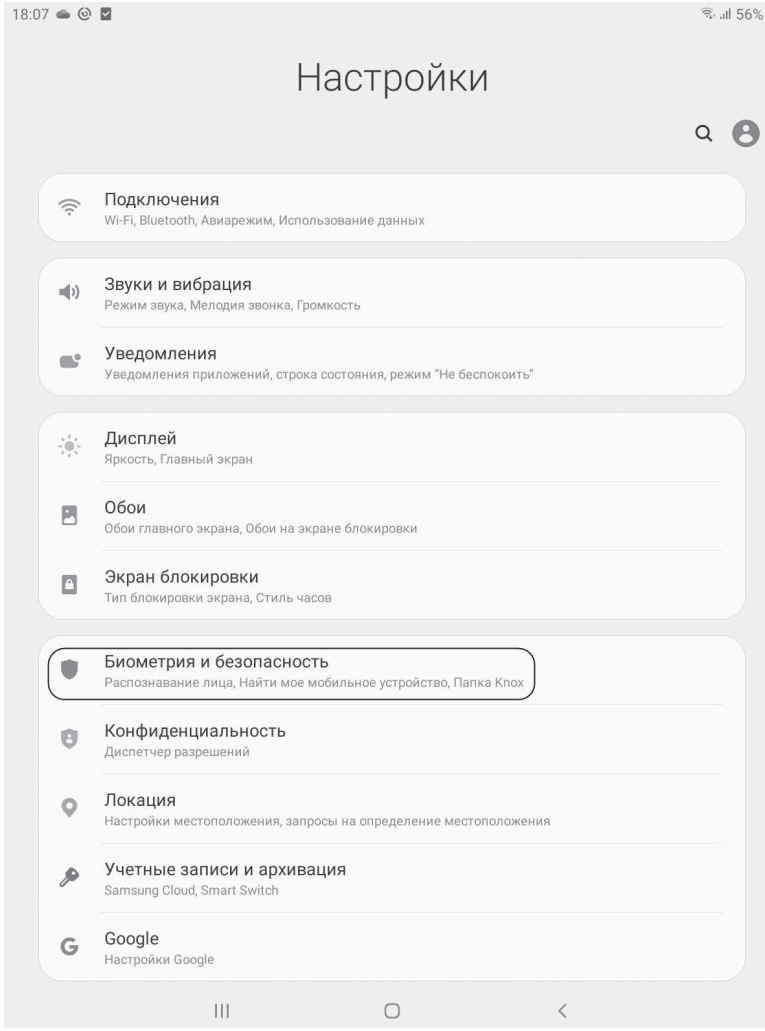


Рис. 1.15. Настройка установки приложений из неизвестных источников

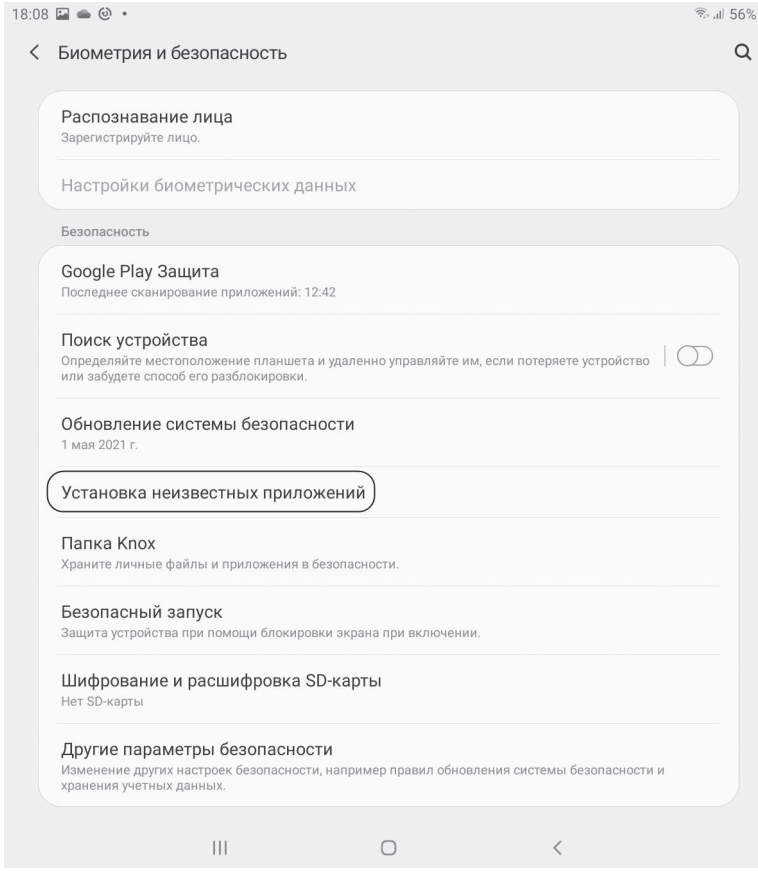


Рис. 1.16. Настройка установки приложений из неизвестных источников

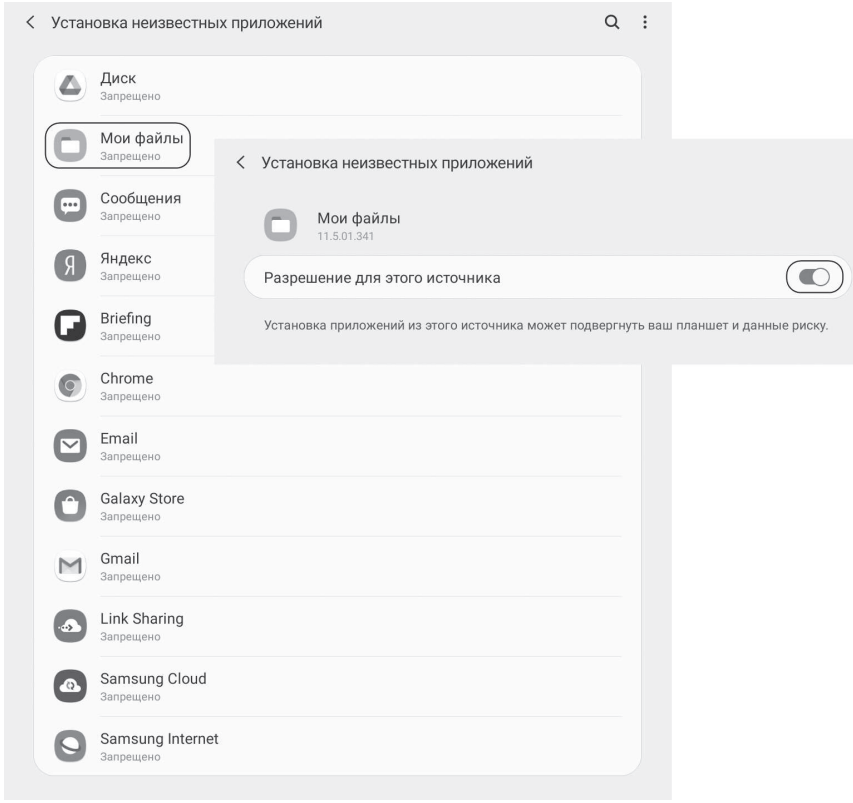


Рис. 1.17. Настройка установки приложений из неизвестных источников

Теперь вам надо включить настройку Отладка по USB.

Для этого раскройте группу настроек Сведения о планшете, затем откройте настройку Сведения о ПО и после этого семь раз нажмите на Номер сборки (рис. 1.18, 1.19, 1.20).

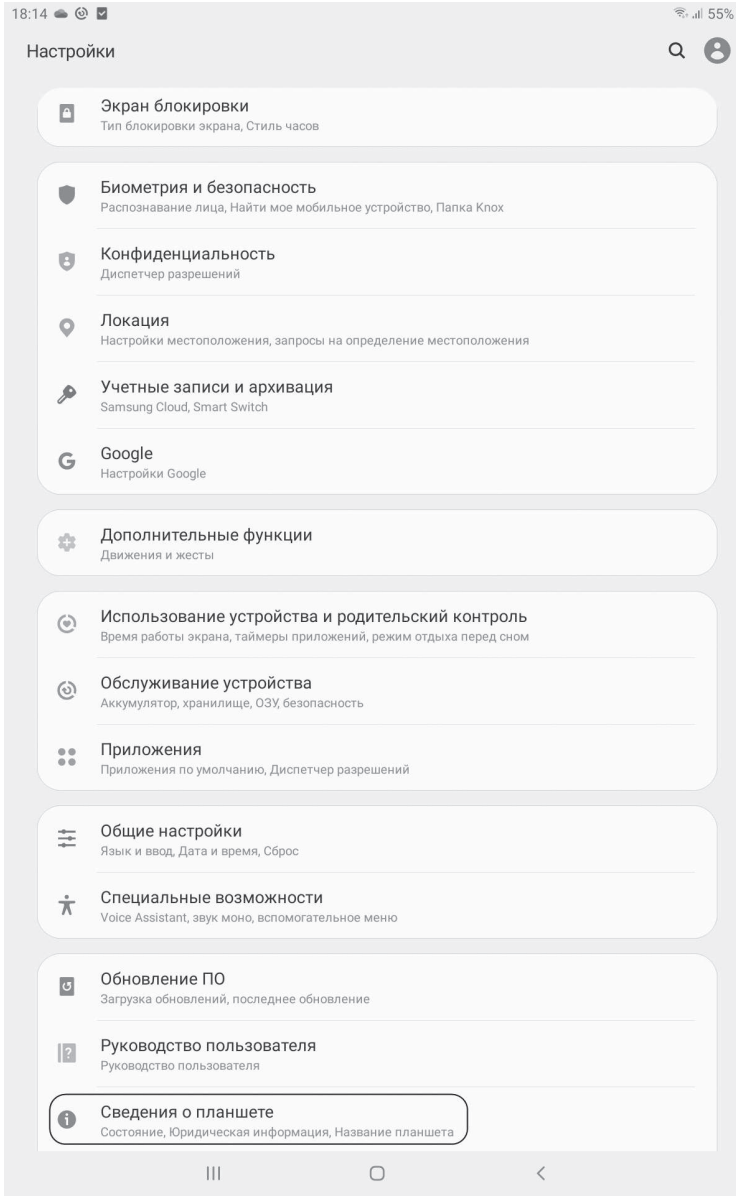


Рис. 1.18. Настройка возможности отладки по USB

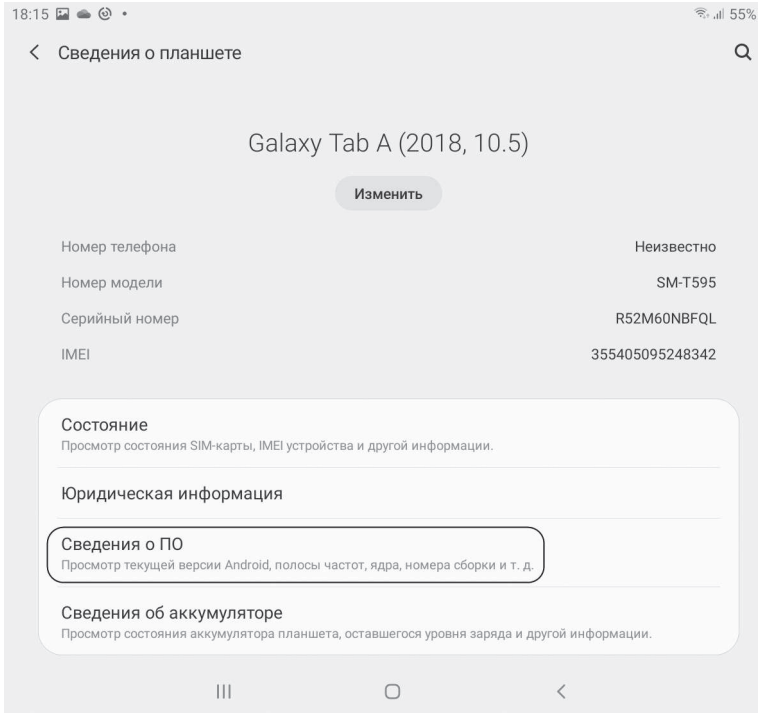


Рис. 1.19. Настройка возможности отладки по USB

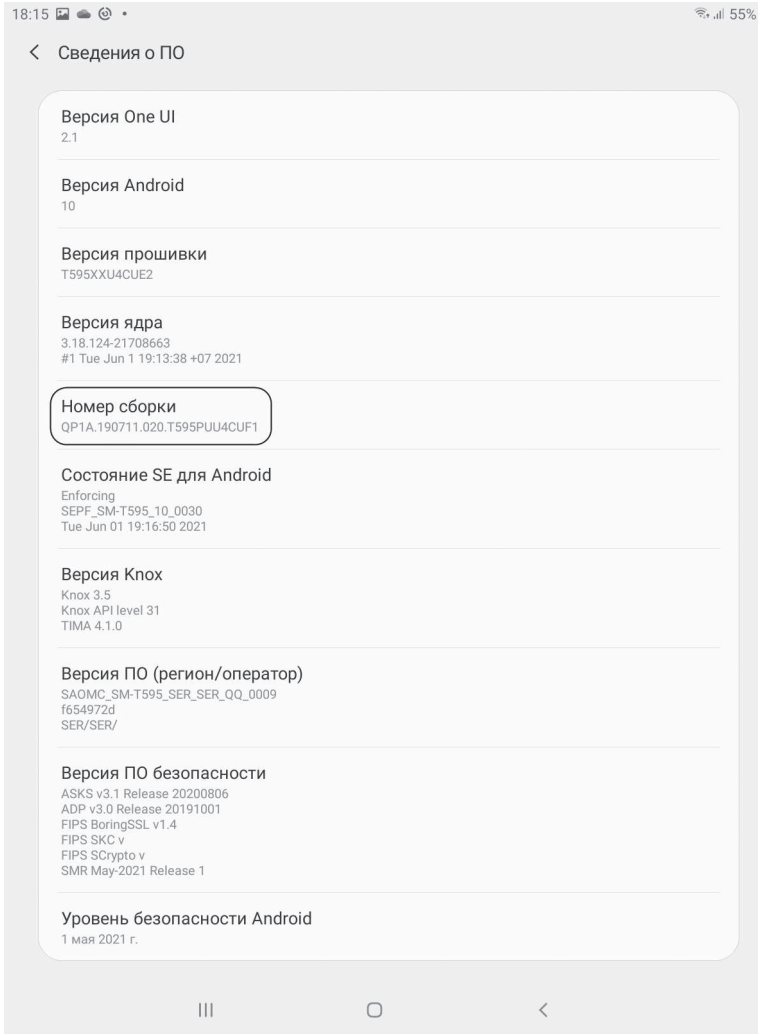


Рис. 1.20. Настройка возможности отладки по USB

После этого в самом низу списка настроек появится группа настроек Параметры разработчика. Откройте ее, включите настройку Отладка по USB и подтвердите разрешение на такую отладку (рис. 1.21, 1.22).

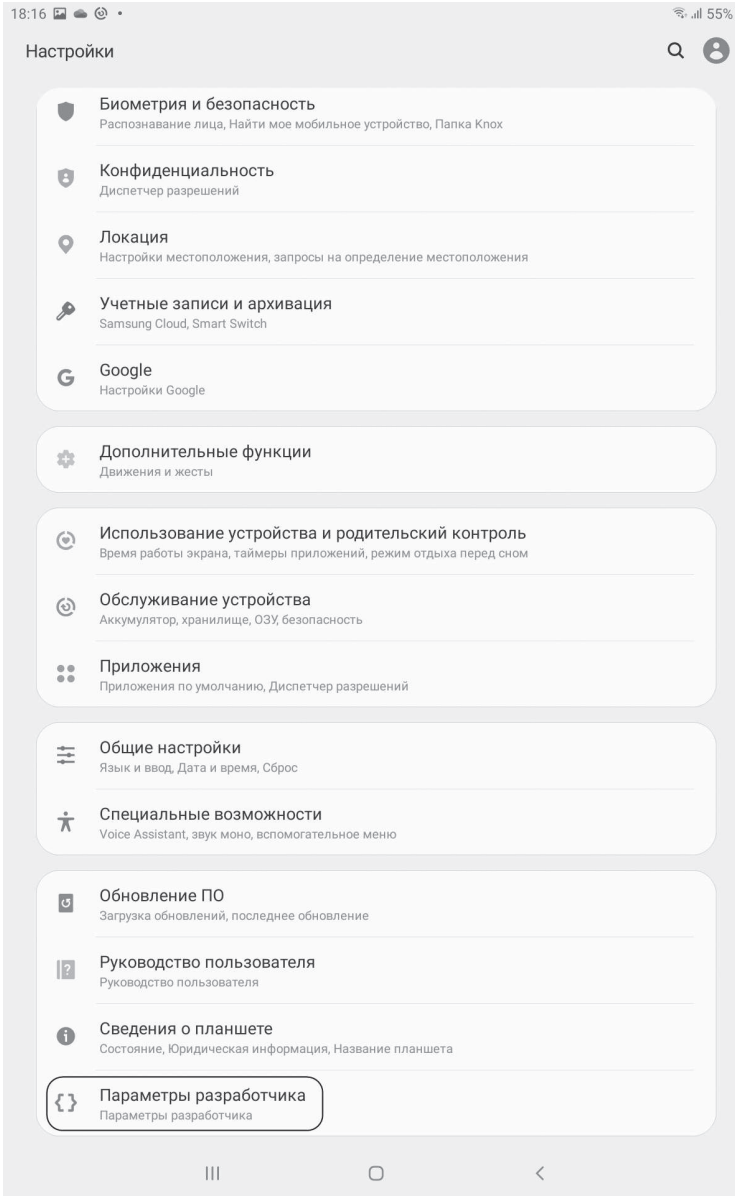


Рис. 1.21. Настройка возможности отладки по USB

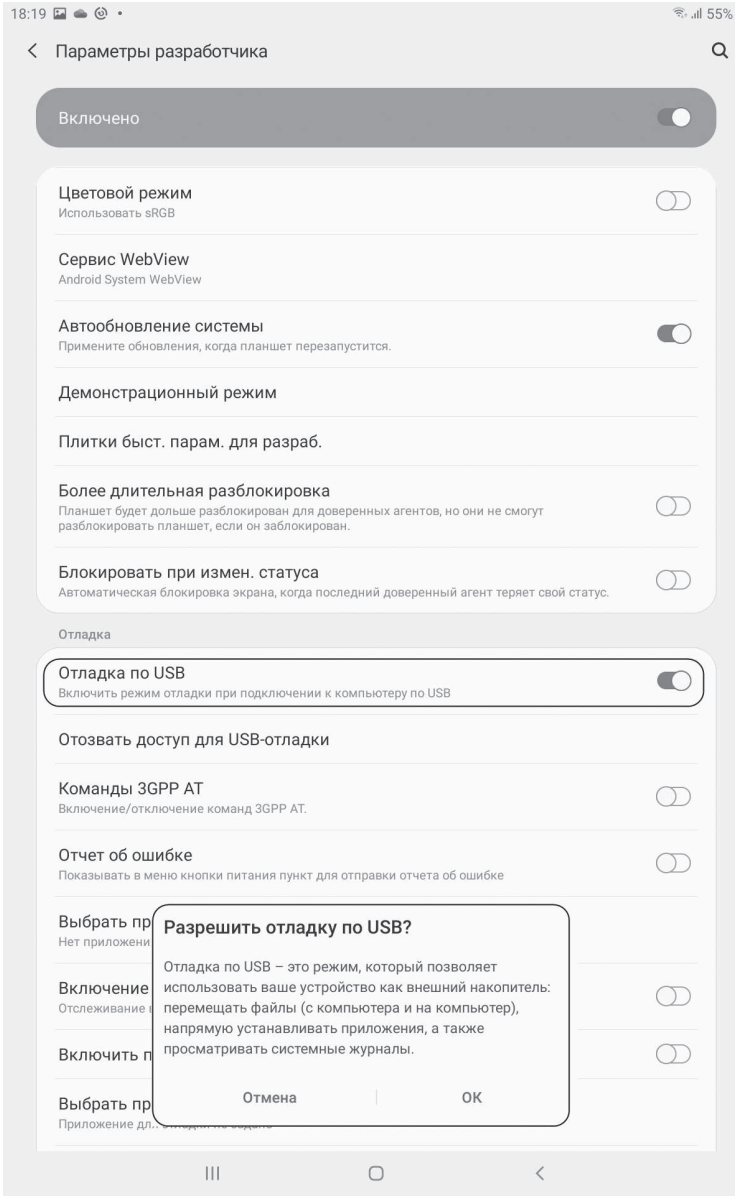


Рис. 1.22. Настройка возможности отладки по USB

Мобильная платформа «1С:Предприятия»

На мобильном устройстве должна быть установлена платформа разработчика того вида, который соответствует разрабатываемому приложению:

- мобильный клиент разработчика,
- мобильный клиент разработчика с автономным режимом,
- мобильная платформа разработчика.

Поскольку в книге будут разрабатываться все три приложения, понадобятся все три файла. Но в реальной жизни нужен только один из этих файлов.

Сначала скачайте платформу для мобильных устройств на стационарный компьютер.

Если вы являетесь зарегистрированным пользователем «1С:Предприятия», скачайте ее с сайта поддержки пользователей по адресу https://releases.1c.ru/project/mobile.

Если вы не являетесь зарегистрированным пользователем, скачайте ее из комплекта учебной версии по адресу https://online.1c.ru/catalog/free/learning.php#platform.

При написании данной книги использовалась версия платформы 8.3.19.51.

Сохраните полученный архив mobile.zip и запомните расположение этого архива, так как в будущем он понадобится вам для сборки уже готового мобильного приложения.

Кроме того, распакуйте архив mobile.zip в какую-нибудь папку. В результате в этой папке окажутся папки с файлами: Android, iOS, Windows, MobileAppMaker и служебные файлы.

Исходя из операционной системы и архитектуры своего мобильного устройства, на котором вы будете вести разработку, найдите нужные файлы платформы разработчика. Например, для планшета Samsung Galaxy Tab A (2018, 10,5) SM-T595, который использовался при написании этой книги, понадобятся следующие файлы из папки Android:

- 1cem-client-arm.apk – мобильный клиент разработчика;
- 1cem-standalone-arm.apk – мобильный клиент разработчика с автономным режимом;
- 1cem-arm.apk – мобильная платформа разработчика.

ПРИМЕЧАНИЕ

Подробнее об именах файлов мобильной платформы можно прочитать в документации: https://its.1c.ru/db/v83doc#bookmark:dev:TI000000916.

Установка мобильной платформы разработчика для мобильных устройств

Чтобы установить платформу разработчика на планшет, подключите его к стационарному компьютеру через USB-порт и затем с помощью интерпретатора командной строки ОС Windows выполните следующую команду:

```
"C:\Users\User\AppData\Local\Android\Sdk\platform-tools\adb.exe" install -r "C:\mobile_8_3_19_51\Android\1cem-client-arm.apk"
```

Здесь:

- C:\Users\User\AppData\Local\Android\Sdk – каталог, куда вы установили комплект средств разработки Android SDK.
- C:\mobile_8_3_19_51 – каталог, куда вы распаковали мобильную платформу.
- 1cem-client-arm.apk – файл платформы разработчика (в данном примере – мобильный клиент разработчика).

ВНИМАНИЕ!

> В приведенном выше примере показана установка мобильного клиента разработчика. В дальнейшем описанную выше команду надо повторить для каждого из файлов платформы (мобильный клиент разработчика, мобильный клиент разработчика с автономным режимом, мобильная платформа разработчика), которые соответствуют вашему мобильному приложению. Чтобы не запутаться, так как ярлыки платформ довольно похожи друг на друга, лучше устанавливать платформы поочередно, по мере изучения второй, третьей и четвертой главы книги.

Перед установкой платформы подтвердите, что вы разрешаете отладку по USB со своего компьютера (рис. 1.23).

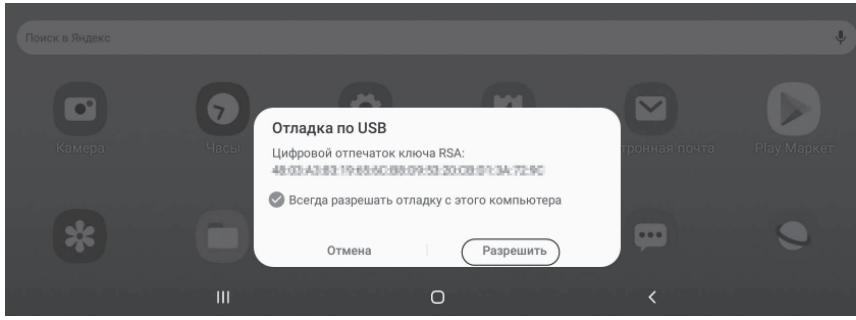


Рис. 1.23. Установка мобильной платформы разработчика

И после этого еще раз выполните описанную выше команду установки мобильной платформы с помощью интерпретатора командной строки.

В результате мобильная платформа для разработки приложения мобильного клиента будет установлена на планшет и появится в списке приложений (рис. 1.24).

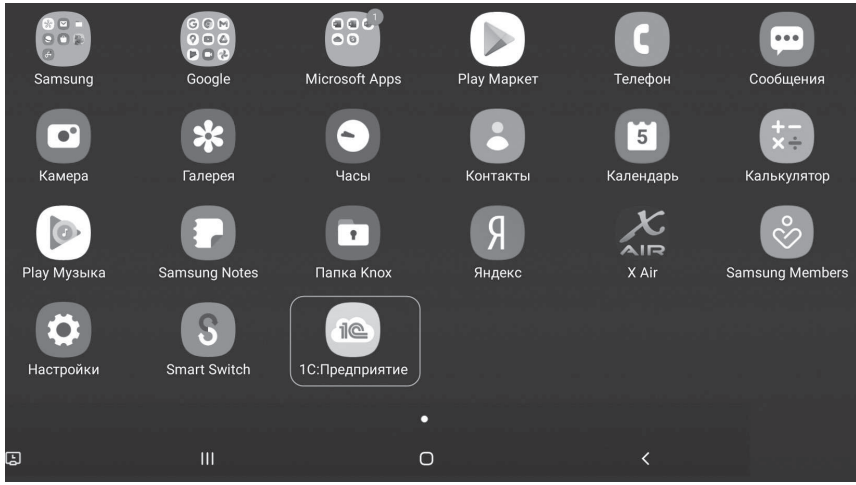


Рис. 1.24. Мобильная платформа для разработки приложения мобильного клиента

ПРИМЕЧАНИЕ

Процесс установки платформы разработчика на мобильные iOS- и Windows-устройства описан в документации по следующим ссылкам:

- iOS https://its.1c.ru/db/v83doc#bookmark:dev:TI000000917
- Windows https://its.1c.ru/db/v83doc#bookmark:dev:TI000001736

Глава 2. Приложение мобильного клиента

В этой главе вы разработаете приложение мобильного клиента, с помощью которого курьер интернет-магазина сможет выполнять свои функции, описанные в разделе «Функциональность мобильного приложения». При этом курьеру будет доступен весь объем задач, как и при работе в офисе, но для этого у него на планшете должно быть постоянное хорошее интернет-соединение.

Для тестирования и отладки приложения мобильного клиента на планшете нужно, чтобы на нем был установлен мобильный клиент разработчика.

ВНИМАНИЕ!

> Прежде чем выполнять то, что написано дальше, убедитесь, что у вас установлены все компоненты, перечисленные в первой главе книги «Подготовка планшета и компьютера».

Чтобы разработать приложение мобильного клиента, вам нужно открыть в конфигураторе офисную конфигурацию Интернет-магазин, изменить пару свойств и опубликовать информационную базу на веб-сервере.

ПРИМЕЧАНИЕ

Эта конфигурация находится в архиве, который вы можете скачать по адресу https://its.1c.ru/bmk/mobile83. Распакуйте архив, создайте пустую информационную базу «1С:Предприятия» и в режиме Конфигуратор загрузите в нее информационную базу из файла «Интернет магазин.dt» (Администрирование – Загрузить информационную базу).

Затем нужно запустить приложение мобильного клиента на планшете, подключиться к этой информационной базе и оценить, как оно выглядит и насколько удобно с ним работать. Если необходимо, можно усовершенствовать некоторые интерфейсные моменты с точки зрения адаптации конфигурации под мобильный интерфейс. И все! Давайте начнем.

Начало разработки

Сначала сделайте копию офисной конфигурации. Это нужно потому, что в процессе изучения данной книги вы разработаете друг за другом (во 2, 3 и 4-й главе) три разных мобильных приложения, и для разработки каждого из них вам понадобится «базовое» офисное приложение (без доработок).

Измените имя конфигурации на МобильныйКлиент, затем отметьте у свойства Назначение использования оба значения: Приложение для платформы и Приложение для мобильной платформы, а также установите свойство Режим совместимости в значение 8.3.19 (рис. 2.1).

Это значит, что конфигурация будет работать и на стационарном компьютере, и на планшете. Важно понимать, что и там и там будет использоваться одна и та же конфигурация, которую вы разработаете на базе офисной конфигурации ИнтернетМагазин.

Режим совместимости нужен потому, что версия платформы «1С:Предприятия» – 8.3.20, а версия мобильной платформы – 8.3.19.

Обновите конфигурацию базы данных (F7).

Затем войдите в конфигуратор с правами администратора и опубликуйте информационную базу на веб-сервере с помощью команды главного меню Администрирование > Публикация на веб-сервере… (рис. 2.2).

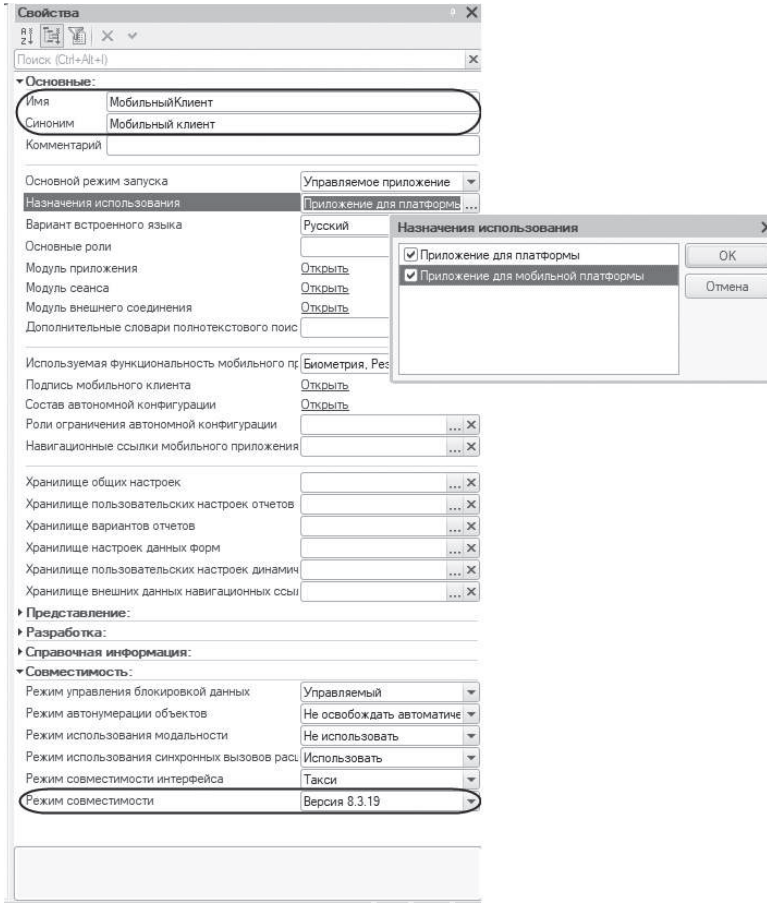


Рис. 2.1. Свойства конфигурации

В появившемся диалоге в поле Имя нужно задать имя виртуального каталога на веб-сервере, в который будет выполнена публикация информационной базы (например, MC).

В поле Каталог нужно указать физический каталог компьютера, в котором будет находиться файл публикации информационной базы (например, C:\Work\MC\).

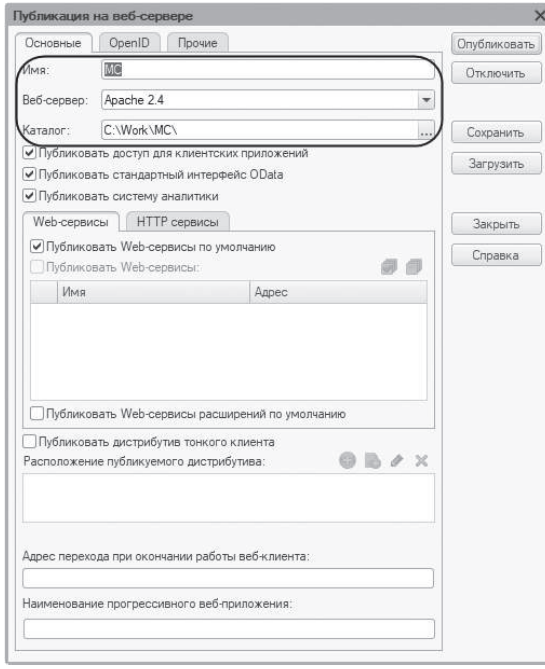


Рис. 2.2. Публикация информационной базы на веб-сервере

Добавление приложения на планшет

Для тестирования и отладки приложения мобильного клиента на планшете нужно, чтобы на нем была установлена мобильная платформа для разработки соответствующего типа приложений. О том, как это сделать, рассказывалось в первой главе в разделе «Установка мобильной платформы разработчика для мобильных устройств».

Найдите в списке приложений планшета мобильную платформу для разработки мобильного клиента и запустите ее. Платформа откроет список своих приложений, который пока пуст.

Добавьте новое мобильное приложение, нажав на значок «+» в правом верхнем углу экрана в строке Приложения. В появившемся окне нужно задать наименование приложения, которое будет отображаться в списке приложений мобильной платформы разработчика, затем в поле Веб-сервер необходимо указать URL веб-сервера, на котором опубликована информационная база, и нажать Готово в правом верхнем углу экрана (рис. 2.3).

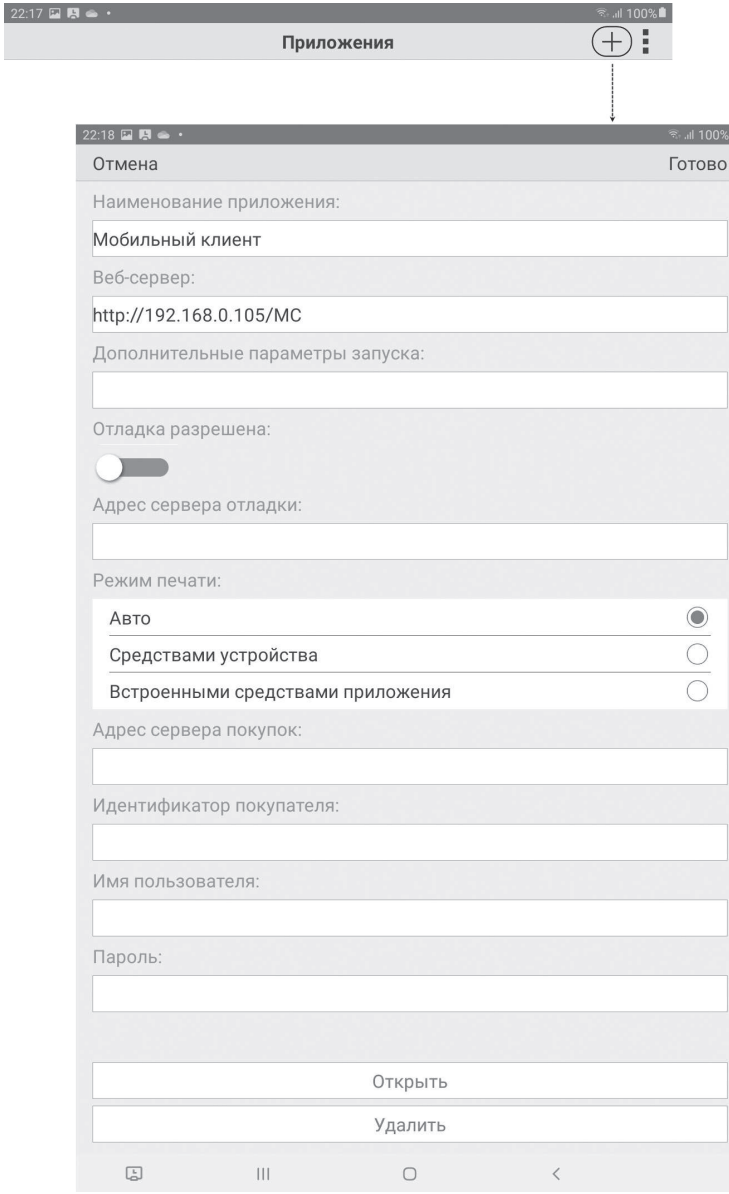


Рис. 2.3. Создание приложения мобильного клиента

ПРИМЕЧАНИЕ

Если ваш компьютер не подключен к сети и не имеет сетевого имени, то, чтобы к нему обратиться, нужно настроить роутер так, чтобы ваш компьютер имел в беспроводной сети статический IP-адрес.

В приведенном примере на рис. 2.3 в поле Веб-сервер указано: http://<IP-адрес компьютера в беспроводной сети>/<Каталог веб-сервера, на котором опубликована информационная база>.

После этого в списке мобильных приложений платформы разработчика появится созданное вами приложение, которое можно запустить кратковременным нажатием на строку Мобильный клиент. При длинном нажатии на эту строку появится контекстное меню, из которого можно изменить свойства мобильного клиента, выбрав пункт Изменить (рис. 2.4).

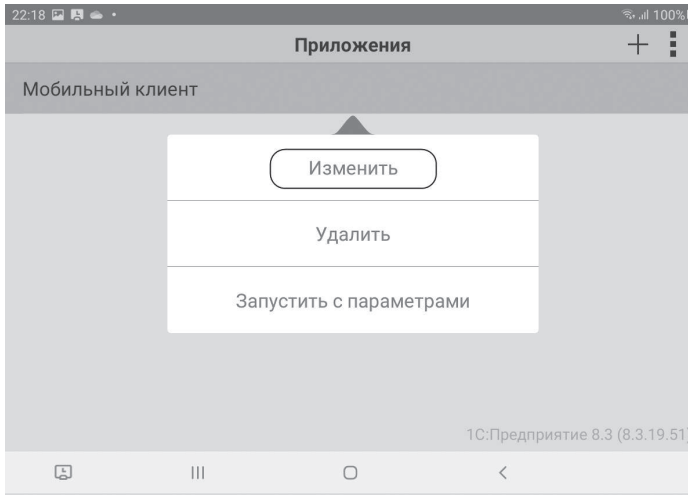


Рис. 2.4. Изменение свойств мобильного клиента

Запустите мобильный клиент. На начальной странице приложения вы увидите команды для перехода к документам, справочникам, отчетам.

Команды отображаются в виде матрицы кнопок, сгруппированных по страницам, которые можно пролистывать. Но в вертикальной ориентации планшета все команды уместились на одну страницу. Картинка на кнопке соответствует связанной с ней команде. В данном случае у кнопок отображается стандартная картинка, так как у соответствующих команд картинка не настроена (рис. 2.5).

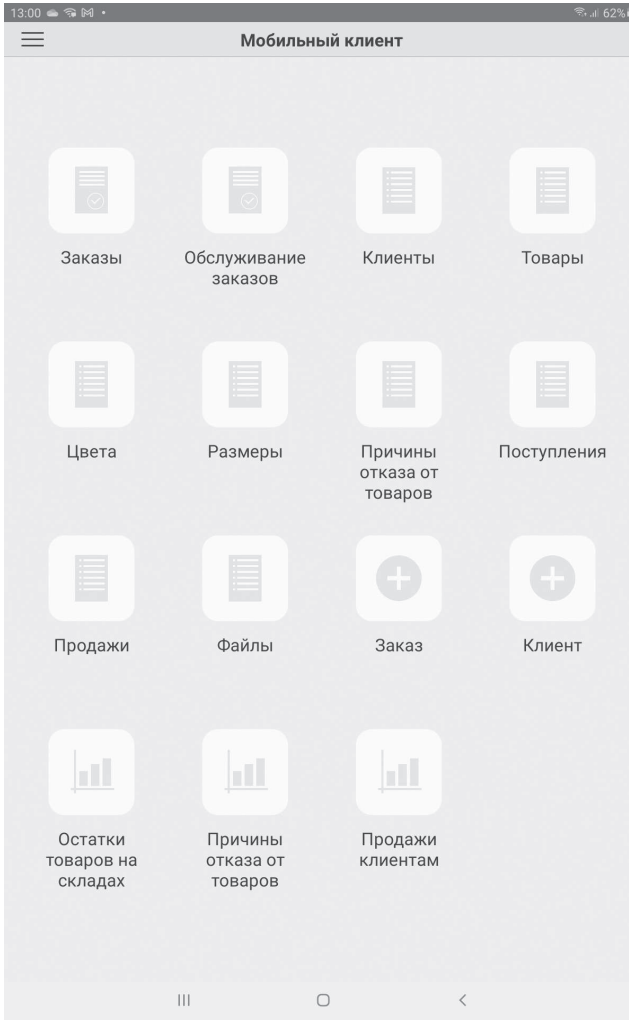


Рис. 2.5. Начальная страница мобильного клиента

В мобильном клиенте доступен практически весь командный интерфейс основного раздела «настольного» приложения, кроме группы «См. также». Чтобы увидеть все команды, можно нажать на значок меню приложения (три горизонтальные черты в левом верхнем углу экрана) или пролистнуть экран у левого края планшета и выбрать раздел Главное (рис. 2.6).

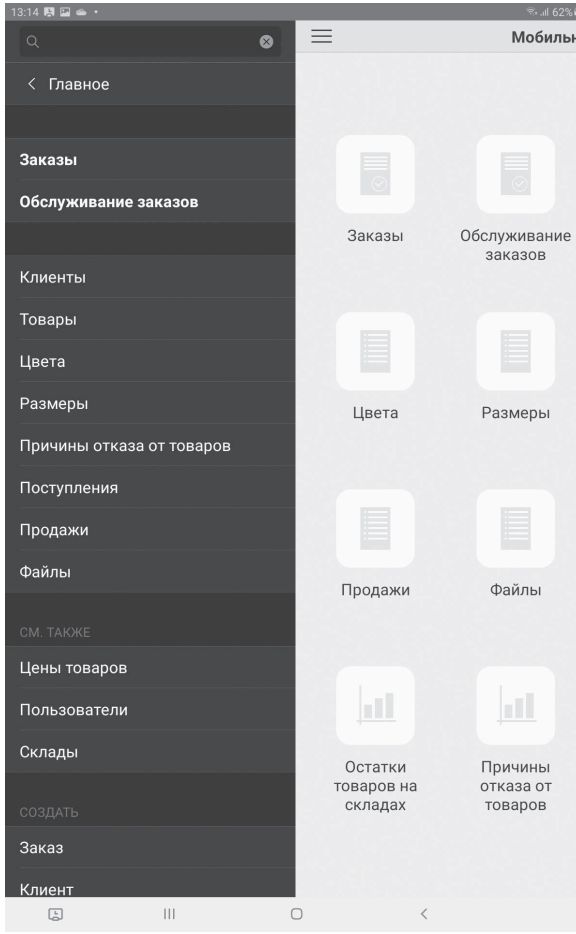


Рис. 2.6. Список команд мобильного клиента

Все данные, введенные ранее в интернет-магазине, показываются и на планшете, потому что это не какая-то другая, а та же самая информационная база, к которой мобильный клиент подключился через веб-сервер. И теперь курьер на планшете в мобильном клиенте может работать с этими данными и изменять их так же, как и другие пользователи в офисе.

Например, нажав на кнопку Заказы, курьер увидит список заказов, созданных в интернет-магазине, сможет открыть и просмотреть любой из них и изменить заказы, находящиеся у него в работе (рис. 2.7).

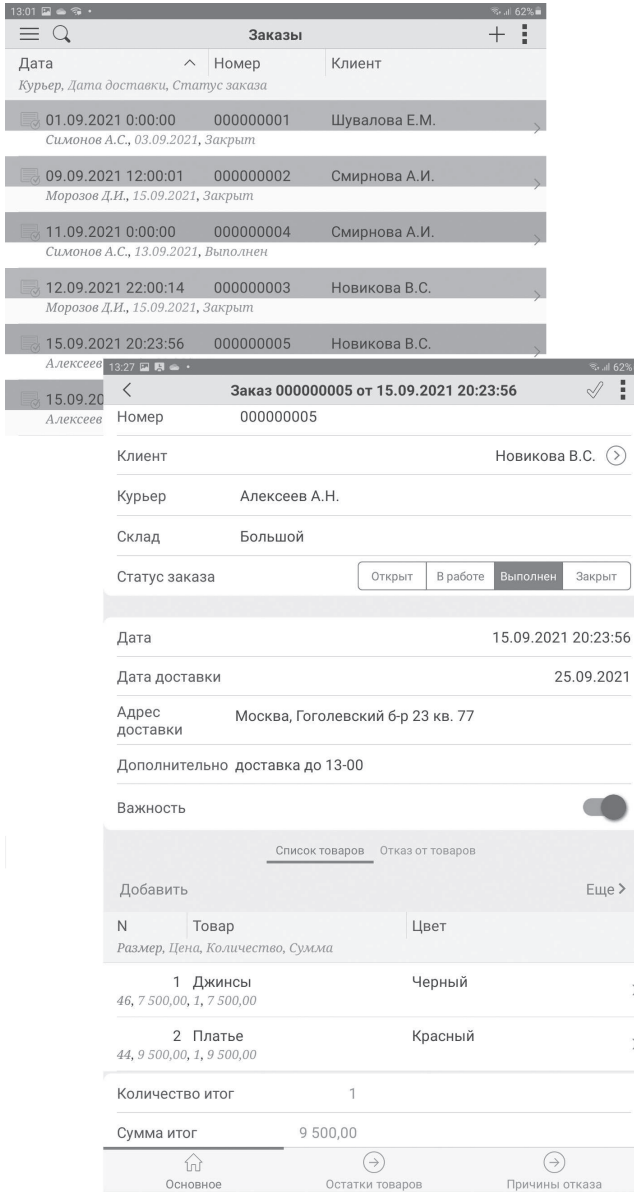


Рис. 2.7. Работа с заказами

Нажав на кнопку Товары, курьер может открыть список товаров и изменить информацию о любом товаре (рис. 2.8).

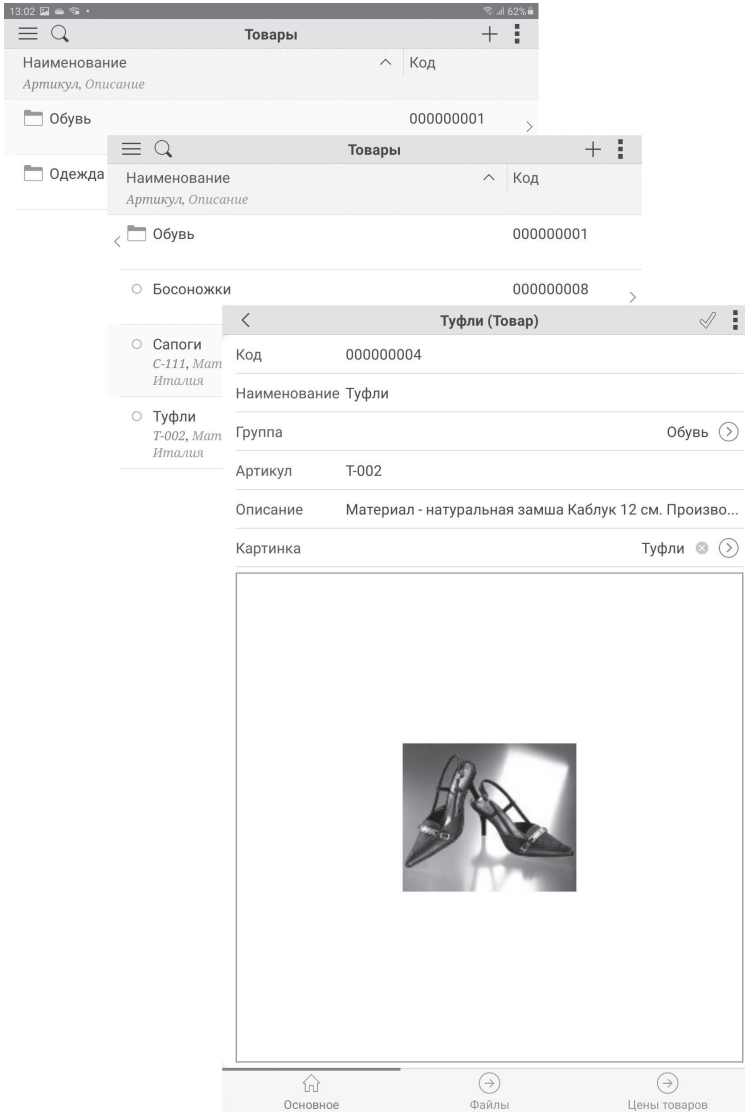


Рис. 2.8. Информация о товарах

Или же курьер может сформировать любой из отчетов и проанализировать содержащуюся в нем информацию (рис. 2.9).

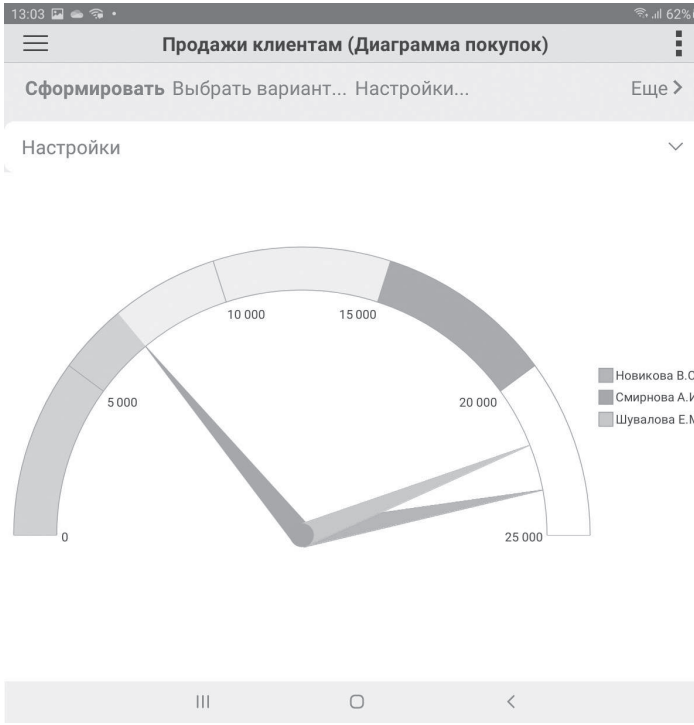


Рис. 2.9. Отчет «Продажи клиентам»

Ну, и так далее. В общем-то, уже сейчас курьер может обслуживать заказы и вносить данные непосредственно в базу данных интернет-магазина.

Однако стандартный мобильный интерфейс, формируемый платформой, может оказаться недостаточно компактным и удобным в сложных и специфических формах. Кроме этого офисное приложение не содержит никаких функций, специфичных для мобильных устройств. В следующем разделе будут показаны способы настройки интерфейса и реализация некоторых мобильных функций приложения.

Доработка интерфейса мобильного клиента

Мобильный клиент формирует для приложения мобильный интерфейс. Мобильный интерфейс отличается от интерфейса, который формирует платформа для настольных компьютеров. Мобильные устройства обладают маленьким экраном, который может быстро менять ориентацию (вертикальная/горизонтальная). Поэтому мобильный интерфейс более компактен и имеет другое расположение компонентов на экране, в отличие от «настольного» интерфейса.

Как же получается этот «новый интерфейс»? Оказывается, платформа львиную долю работы делает сама. Используя то формализованное описание форм, которое имеется в конфигураторе, она компонует формы таким образом, чтобы обеспечить удобную работу с ними на маленьких экранах мобильных устройств.

Но, конечно, платформа не может полностью обеспечить нужный внешний вид форм, поэтому ей нужны «подсказки» от разработчика, чтобы она лучше понимала, как показывать формы на мобильных устройствах. То есть конфигурацию необходимо адаптировать к работе в мобильном клиенте.

Все эти доработки можно разделить на два основных направления.

Первое – это избавиться от особенных и специфических интерфейсных решений, больше полагаясь на автоматическую компоновку форм, выполняемую платформой на основе информации о типах данных. Такими специфическими решениями могут быть фиксированные размеры полей, жестко установленная горизонтальная группировка элементов и тому подобное.

Другое направление – это подсказать мобильному клиенту дополнительную информацию об элементах формы. Это потребуется в первую очередь для нестандартных или больших форм, где автоматической адаптации и расстановки элементов будет недостаточно. В этих случаях полезно вручную указать, какие из элементов являются более и менее важными, как отображать списки и т. п.

ПОДРОБНЕЕ

О новых свойствах форм и их элементов для адаптации к работе в мобильном клиенте можно прочитать в документации «1С:Предприятие 8.3.20. Руководство разработчика»:

- глава 29 «Разработка для мобильных устройств»,
- раздел 29.3.4.3,
- 29.3.4.4.8,
- 29.5.4,
- а также глава 7 «Формы», раздел 7.7.13.9.

При разработке интерфейса мобильного клиента прежде всего надо понимать, что при работе на мобильном устройстве (на планшете и в особенности на смартфоне) существует проблема недостатка рабочего места. Поэтому размеры элементов форм должны автоматически подстраиваться под размер мобильного устройства. По расположению элементов формы должны быть вытянуты в длину и ограничены по ширине, так как вертикальная прокрутка форм вполне ожидаема и привычна, в то время как горизонтальная прокрутка на мобильных устройствах не поддерживается.

Как уже говорилось, чтобы минимизировать усилия разработчика, в платформе реализован ряд механизмов для самостоятельной адаптации форм к работе на маленьких экранах мобильных устройств.

Свойства форм и их элементов (которые будут подробно рассмотрены ниже), отвечающие за адаптацию к работе в мобильном клиенте, пока установлены в значение Авто. Конфигурация ИнтернетМагазин, которую вы адаптируете для работы в мобильном клиенте, достаточно небольшая и не имеет перегруженных форм, но даже на ней вы можете увидеть, как изменение этих свойств повлияет на отображение форм на мобильных устройствах.

Поведение таблиц при сжатии по горизонтали

Сначала, ничего не меняя, посмотрите, как выглядят формы списков в случае полной автоматической адаптации к размеру планшета.

Запустите мобильный клиент, нажмите на кнопку Заказы, откройте список заказов клиентов и оцените вид списка заказов при вертикальной ориентации планшета (рис. 2.10).

Вид списка заказов (как и любых других списков) изменился по сравнению с экраном стационарного компьютера. Таблицы списка имеют только вертикальную прокрутку и сжимаются по горизонтали. При этом колонки, не умещающиеся в одну строку, не отображаются в этой же строке, а их значения переносятся на следующую строку и отображаются мелким курсивом разного цвета через запятую.

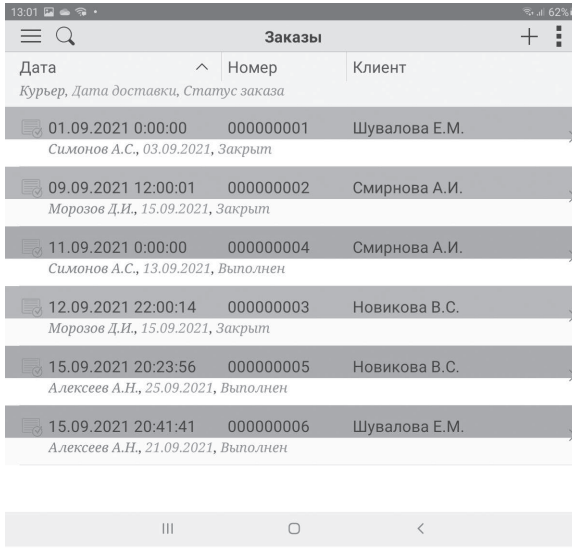


Рис. 2.10. Список заказов клиентов

Ключевым свойством, помогающим формам (самостоятельно или с помощью разработчика) адаптироваться к экранам мобильных устройств, является свойство ВажностьПриОтображении. Это свойство есть у полей формы, групп, страниц, таблиц и других элементов формы. Оно принимает значения перечисления ВажностьПриОтображении:

- Авто. Элементам формы, связанным с основным реквизитом формы, устанавливается важность ОченьВысокая. Важность Высокая присваивается командным панелям, источником которых выступает сама форма и т. п.
- Высокая. Важность высокая.
- Низкая. Важность низкая.
- Обычная. Важность обычная.
- ОченьВысокая. Важность очень высокая.
- ОченьНизкая. Важность очень низкая.

ПОДРОБНЕЕ

Подробнее об этом можно прочитать в документации «1С:Предприятие 8.3.20. Руководство разработчика»:

- глава 29 «Разработка для мобильных устройств», раздел 29.3.4.3;
- а также глава 7 «Формы», раздел 7.7.13.9.

При адаптации списков к ширине экранов мобильных устройств поведение таблиц формы, отображающих данные списков, определяется свойством ПоведениеПриСжатииПоГоризонтали. Оно принимает значения перечисления ПоведениеТаблицыПриСжатииПоГоризонтали:

- Авто:
  - В мобильной платформе используется значение СкрыватьЭлементыПоВажности;
  - В мобильном клиенте – ПереноситьЭлементыПоВажности.
- СкрыватьЭлементыПоВажности. При адаптации к ширине экрана колонки с наименьшей важностью скрываются. В пределах одного уровня важности скрываются элементы, последовательно расположенные справа налево.
- ПереноситьЭлементыПоВажности. При адаптации к ширине экрана колонки с наименьшей важностью скрываются, а их значения отображаются в отдельной строке мелким шрифтом с курсивным начертанием с использованием нескольких цветов. В пределах одного уровня важности переносятся элементы, расположенные справа налево. Элементы в строке располагаются в порядке убывания важности.

Таким образом, хотя у таблицы формы списка документов Заказ свойство ПоведениеПриСжатииПоГоризонтали стандартно установлено в значение Авто, на мобильном клиенте оно трактуется как ПереноситьЭлементыПоВажности.

Стандартно у всех колонок таблицы списка свойство ВажностьПриОтображении принимает значение Авто. На мобильном клиенте это трактуется как ОченьВысокая важность, так как колонки отображают данные основного реквизита формы – динамического списка. Поскольку важность у всех колонок пока одинаковая, то переносятся не поместившиеся по ширине элементы, начиная справа – налево. А порядок их следования друг за другом определяется порядком колонок в таблице списка, заданным при конфигурировании формы.

Теперь откройте в конфигураторе форму списка документа Заказ, измените важность колонок и посмотрите, как изменится вид списка в мобильном клиенте.

Установите свойство ВажностьПриОтображении у колонок таблицы формы Список в следующие значения:

- Дата – Авто;
- Номер – Авто;
- Клиент – Обычная;

- Курьер – Низкая;
- ДатаДоставки – Очень высокая;
- СтатусЗаказа – Высокая.

Обновите конфигурацию базы данных (F7), снова запустите мобильный клиент на планшете и откройте список заказов клиентов. Как вы видите, вид списка изменился.

Поскольку колонкам динамического списка со значением Авто присваивается ОченьВысокая важность, при вертикальной ориентации планшета по мере убывания важности колонки будут следовать друг за другом в следующем порядке: Дата, Номер, Дата доставки, Статус заказа, Клиент, Курьер. При этом последние не поместившиеся по ширине колонки будут переноситься на следующую строку (рис. 2.11).

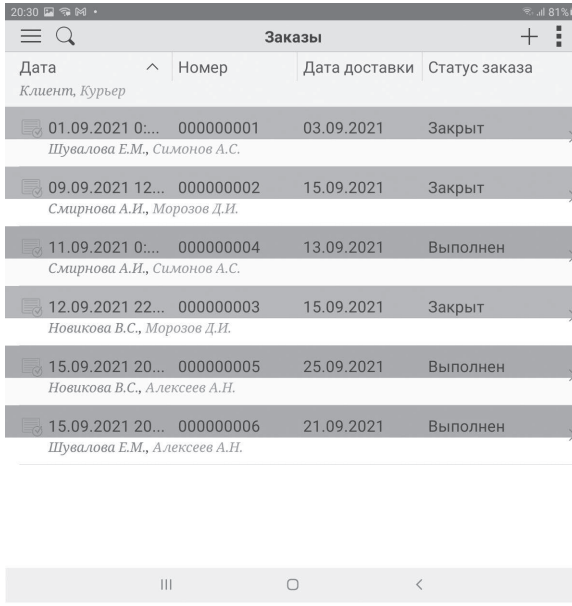


Рис. 2.11. Список заказов клиентов при вертикальной ориентации планшета

Причем колонки переносятся по важности только в том случае, если они не помещаются по ширине экрана. Так, при горизонтальной ориентации планшета никакого переноса не происходит, так как все колонки списка заказов помещаются на экране (рис. 2.12).

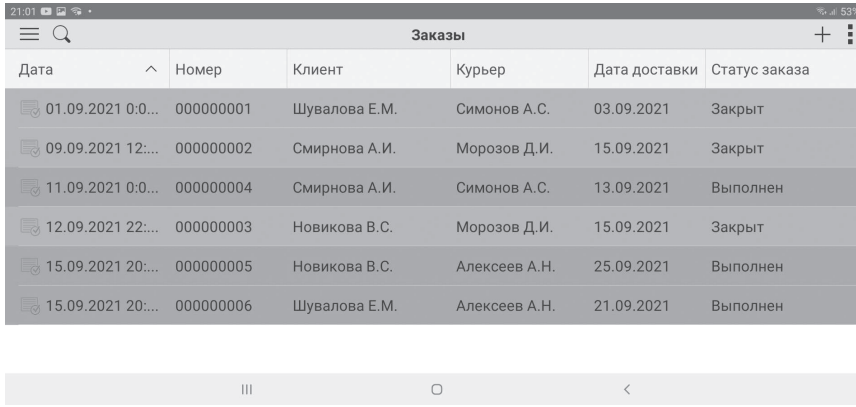


Рис. 2.12. Список заказов клиентов в горизонтальной ориентации планшета

Таким образом, если вы хотите изменить порядок следования колонок в списке, то должны соответствующим образом установить свойство ВажностьПриОтображении у колонок таблицы формы, отображающей данные списка. А свойство ПоведениеПриСжатииПоГоризонтали можно не изменять, так как оно стандартно установлено в значение Авто, что на мобильном клиенте трактуется как ПереноситьЭлементыПоВажности.

Сворачивание элементов форм по важности

Теперь посмотрите, как адаптируются к размерам мобильных устройств формы документов, элементов справочников и т. п.

Прежде всего форма пытается подстроиться к ширине экрана на мобильном устройстве. Для тех элементов формы, ширина которых жестко задана в конфигурации и превышает фактическую ширину экрана мобильного устройства, ширина уменьшается принудительно таким образом, чтобы элемент поместился (по ширине) в текущей ориентации мобильного устройства без горизонтальной прокрутки.

Поэтому при разработке форм фиксированную ширину элементов лучше не использовать (свойство Ширина должно быть равно нулю, а свойство АвтоМаксимальнаяШирина – установлено).

А также для групп формы, элементы которых на стационарном компьютере требуется отображать горизонтально, свойство Группировка должно быть установлено в значение Горизонтальная если возможно. Потому что если свойство Группировка установлено в значение Горизонтальная всегда, то

на узких экранах мобильных устройств элементы таких групп могут быть не видны, так как горизонтальная прокрутка форм на мобильном устройстве не используется.

Например, в документе Заказ реквизиты документа помещены в две группы с вертикальной группировкой ЛеваяКолонка и ПраваяКолонка. В свою очередь, эти две группы объединены родительской группой Шапка. Тип группировки у этой группы не определен, что трактуется как значение по умолчанию – Горизонтальная если возможно (см. рис. 2.13).

В окне предварительного просмотра вы видите, что на экране стационарного компьютера реквизиты документа (при достаточной ширине и разрешении экрана) будут выводиться горизонтально, в две вертикальные колонки (рис. 2.13).

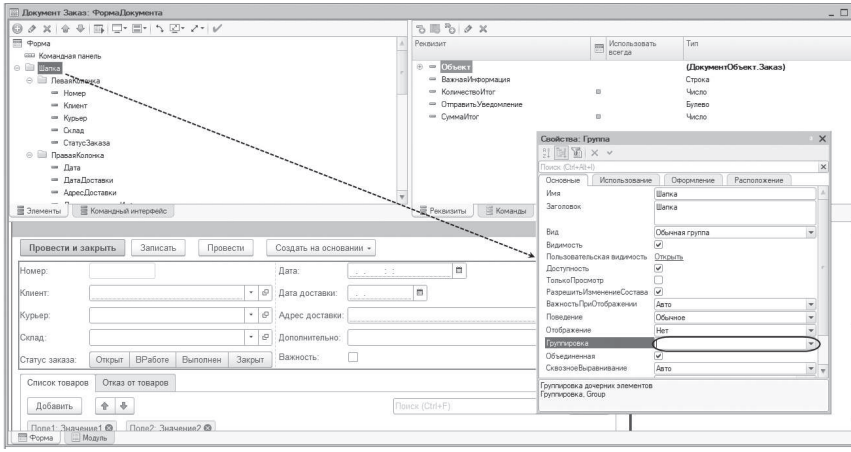


Рис. 2.13. Форма документа «Заказ» в редакторе формы

Но, поскольку мобильные устройства значительно ограничены по ширине, на них левая и правая колонка реквизитов будут располагаться вертикально друг под другом. То есть, поскольку горизонтальная группировка элементов невозможна, она превращается в вертикальную.

Это можно увидеть непосредственно в конфигураторе. Для этого в командной панели редактора формы нужно нажать кнопку для выбора вида платформы и указать Мобильное устройство (мобильный клиент). А чтобы изменить расположение формы с вертикального на горизонтальное, нужно нажать соседнюю кнопку Повернуть устройство (рис. 2.14).

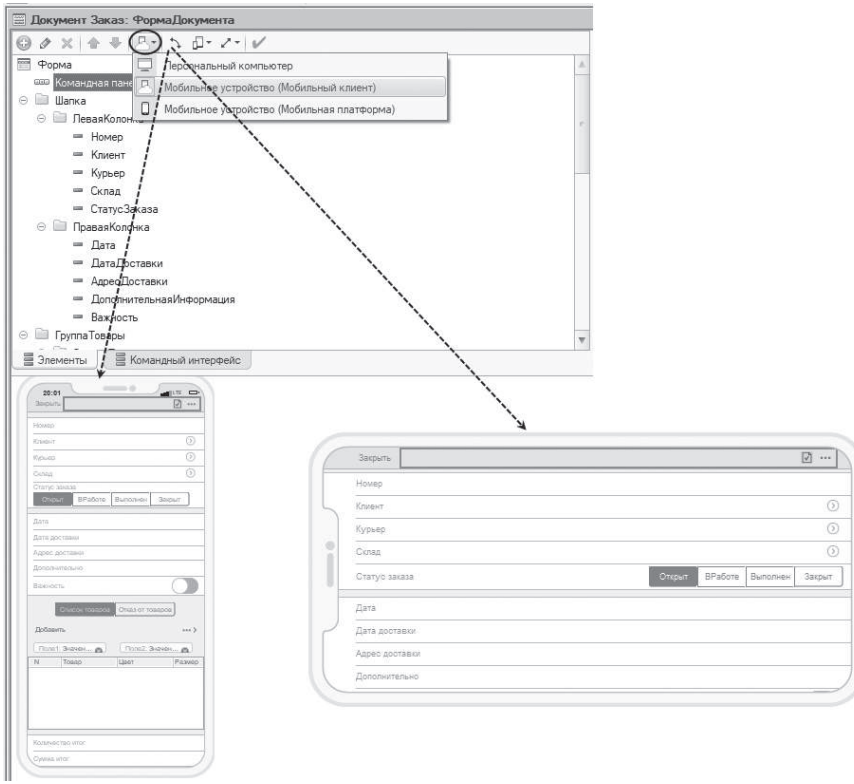


Рис. 2.14. Форма заказа в мобильном клиенте

Вертикальное расположение групп друг под другом может быть неудобно, особенно в случае документа с большой табличной частью. Но вы можете оптимизировать форму документа, чтобы не приходилось ее все время прокручивать по вертикали.

В процессе адаптации к размерам мобильных устройств платформа пытается уменьшить количество отображаемых элементов формы, чтобы улучшить ее внешний вид, сделать форму более лаконичной и акцентировать внимание пользователя на самой важной информации. Для управления возможностью такой перестройки предназначено свойство формы СворачиваниеЭлементовПоВажности. Свойство принимает значения перечисления СворачиваниеЭлементовФормыПоВажности:

- Авто. Интерпретируется как Использовать.

- Использовать. Сворачивание элементов формы по важности используется.
- НеИспользовать. Сворачивание элементов формы по важности не используется.

Для определения того, какие элементы формы отображать полностью, а какие сворачивать, используется свойство ВажностьПриОтображении, о котором было рассказано выше. Изменяя значение этого свойства у элементов формы, вы можете добиться нужного вида формы на экране устройства. Значение этого свойства мобильный клиент обрабатывает исходя из принципа, что более важным элементам отводится больше места в форме:

- Если более важный элемент расположен под менее важными и эти менее важные элементы занимают более трех строк формы, то менее важные элементы объединяются и помещаются в сворачиваемую группу.
- Если более важный элемент расположен внутри иерархии групп или страниц и при этом в разных местах иерархии над ним расположены менее важные элементы, занимающие более трех строк, в каждом уровне иерархии формируется сворачиваемая группа.
- Если имеются несколько элементов более высокой важности, то менее важные элементы, находящиеся между ними, помещаются в сворачиваемую группу, если менее важные элементы занимают более трех строк, и т. д.

ПОДРОБНЕЕ

Подробнее об этом можно прочитать в документации «1С:Предприятие 8.3.20. Руководство разработчика»:

- глава 29 «Разработка для мобильных устройств», раздел 29.3.4.3;
- а также глава 7 «Формы», раздел 7.7.13.9.

Попробуйте изменить эти свойства и посмотрите, как это скажется на отображении формы заказа в мобильном клиенте.

У группы формы ПраваяКолонка установите свойство ВажностьПриОтображении в значение Низкая. И удалите заголовок группы, если он задан.

У всех остальных элементов формы ВажностьПриОтображении стандартно установлена в Авто. У самой формы свойство СворачиваниеЭлементовПоВажности также принимает значение Авто. Это значит, что сворачивание элементов формы по важности используется.

В конфигураторе (как на рис. 2.14) эти изменения увидеть не получится. Поэтому обновите конфигурацию базы данных (F7), запустите мобильный клиент на планшете и откройте форму любого заказа.

Как вы видите, вся группа реквизитов, относящихся к группе ПраваяКолонка (Дата, Дата доставки, Адрес доставки, Дополнительно, Важность, см. рис. 2.15) показана в свернутом виде. Заголовок этой группы формируется перечислением через запятую заголовков всех элементов, входящих в свертываемую группу. Вы можете раскрыть ее на отдельном экране планшета, нажав на заголовок группы или на специальный значок справа от заголовка (рис. 2.15).

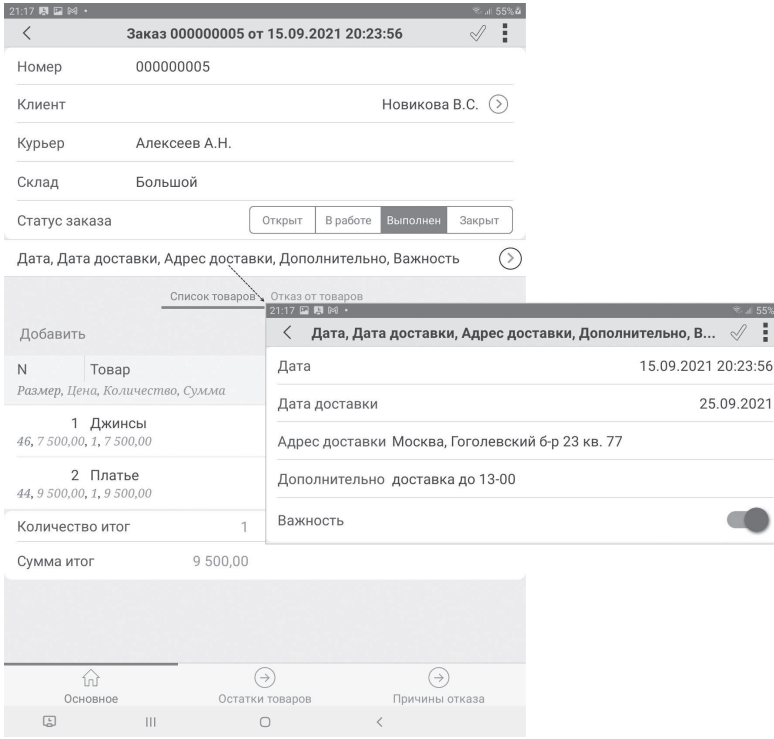


Рис. 2.15. Форма заказа

В данном случае, поскольку группа ПраваяКолонка с низкой важностью находится между более важными элементами формы и эта группа занимает более трех строк, она сворачивается.

Заголовок свертываемой группы формируется по следующему алгоритму:

- Если свертывается уже существующая группа, то используется заголовок этой группы.

- Если в свертываемую группу попадает один элемент с заданным заголовком, то используется заголовок этого элемента.
- Если не подходит ни один из вышеперечисленных пунктов – заголовок формируется соединением через «,» всех заголовков элементов, входящих в свертываемую группу.

В данном случае, показанном на рис. 2.15, используется последний вариант.

Однако перечисление заголовков реквизитов, входящих в свертываемую группу, не всегда удобно, особенно когда таких реквизитов много. Поэтому дайте понятный короткий заголовок группе ПраваяКолонка, например: Информация о доставке. В результате этот заголовок и станет заголовком свертываемой группы (рис. 2.16).

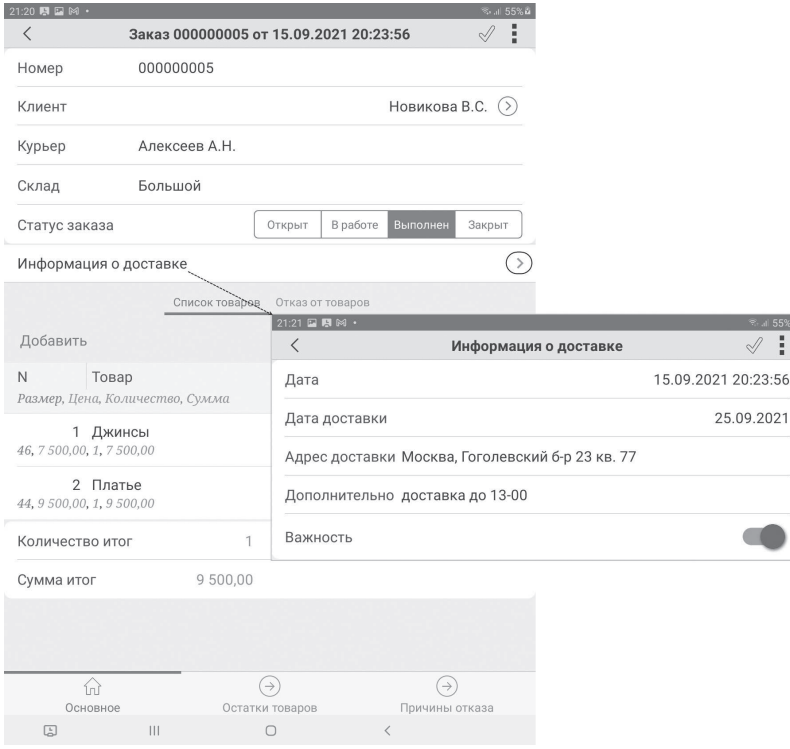


Рис. 2.16. Форма заказа

Использование текущей строки таблицы

Нажатие на строку таблицы в мобильном клиенте имеет свои особенности, которые нужно знать и учитывать. Поведение текущей строки таблицы при этом определяется свойством ИспользованиеТекущейСтроки. Оно принимает значения перечисления ИспользованиеТекущейСтрокиТаблицы:

- Авто. Данное значение трактуется как значение Выбор и на мобильной платформе, и в мобильном клиенте.
- Выбор. В этом случае свойства ТекущаяСтрока, ТекущийЭлемент, ТекущиеДанные определены только во время выполнения:
  - обработчика контекстной команды;
  - событий активизации строки или ячейки;
  - событий редактирования строки.
- В остальное время эти свойства таблицы имеют значение Неопределено. Для команды формы, которой необходимы данные текущей строки, следует явно указать такую необходимость в свойстве команды ИспользованиеТекущейСтроки. Дополнительно необходимо указать таблицу, данные из которой будет использовать команда, с помощью свойства ИспользуемаяТаблица.
- В списке текущая строка визуально определяется кратковременно, во время нажатия на строку. При нажатии на строку вызываются события ПриАктивизацииПоля, ПриАктивизацииСтроки, ПриАктивизацииЯчейки, Выбор. Данное значение рекомендуется использовать для таблиц, логически не связанных с какими-то другими данными (в том числе и табличными).
- ОтображениеВыделения. В этом случае текущая строка существует всегда, если таблица содержит хотя бы одну строку. В списке текущая строка визуально определяется всегда. При нажатии на строку вызывается событие ПриАктивизацииПоля. Событие Выбор не вызывается. Данное значение рекомендуется использовать для таблиц, которые логически связаны с какими-то другими данными (в том числе и табличными).
- ОтображениеВыделенияИВыбор. В этом случае текущая строка существует всегда, если таблица содержит хотя бы одну строку. В списке текущая строка визуально определяется всегда. Также в правой части строки имеется кнопка, при нажатии которой вызываются события ПриАктивизацииПоля и Выбор.

Например, вы хотите сделать так, чтобы в мобильном клиенте, когда курьер открывает список документов Обслуживание заказов, он мог бы сразу открыть форму заказа, выделенного в табличной части документа. Для этого нужно, чтобы строка таблицы со списком заказов находилась не в режиме выбора и редактирования (что является стандартным поведением), а подсвечивалась бы с возможностью открыть выделенный в табличной части заказ.

Чтобы реализовать эту задачу, откройте в конфигураторе форму документа ОбслуживаниеЗаказов и установите свойство ИспользованиеТекущейСтроки таблицы Заказы в значение ОтображениеВыделенияИВыбор (рис. 2.17).

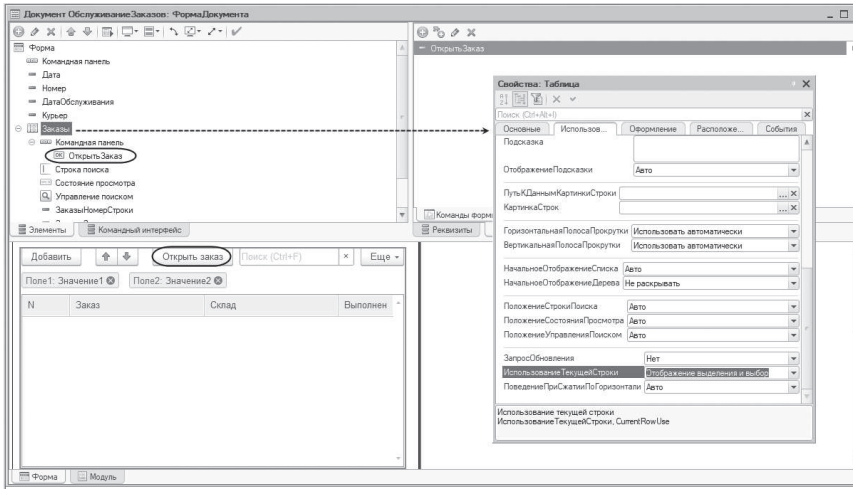


Рис. 2.17. Форма документа «Обслуживание заказов»

Затем добавьте в форму команду ОткрытьЗаказ и перетащите ее в командную панель таблицы Заказы (см. рис. 2.17). Обработчик команды заполните следующим образом (листинг 2.1).

Листинг 2.1. Обработчик команды «ОткрытьЗаказ»

```
&НаКлиенте
Процедура ОткрытьЗаказ(Команда)
        ПоказатьЗначение(, Элементы.Заказы.ТекущиеДанные.Заказ);
КонецПроцедуры
```

В этом обработчике, используя свойство ТекущиеДанные таблицы Заказы, вы получаете объект, содержащий данные, находящиеся в текущей строке таблицы. Обращаясь через точку к полю этого объекта Заказ, вы получаете

ссылку на конкретный заказ, содержащуюся в этом поле. И затем передаете эту ссылку в метод глобального контекста ПоказатьЗначение(), который открывает форму выбранного заказа.

Обновите конфигурацию базы данных (F7), запустите мобильный клиент на планшете и откройте форму документа Обслуживание заказов. Затем выделите в табличной части нужный заказ и нажмите Открыть заказ над списком заказов (рис. 2.18).

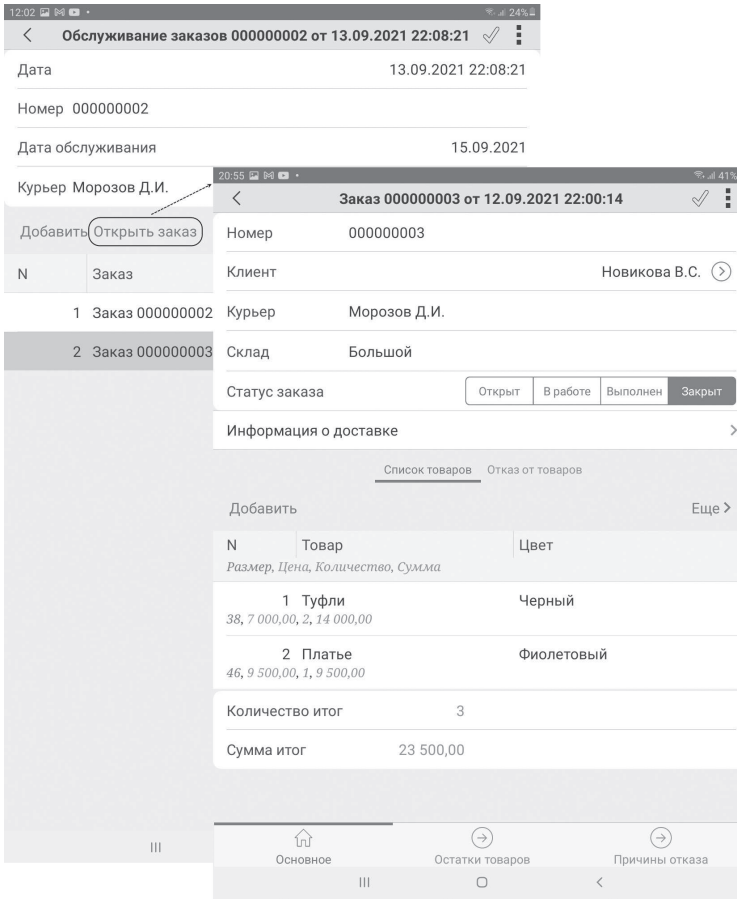


Рис. 2.18. Просмотр заказа из формы документа «Обслуживание заказов»

Все работает так, как вы и хотели, хотя такого же результата можно добиться за счет изменения свойств самой команды, а не таблицы. Такое поведение является более логичным и привычным для мобильного клиента. При этом команда, связанная со строкой таблицы, показывается в контекстном меню этой строки. Об этом будет рассказано в следующем разделе.

Использование текущей строки командой формы

Типичный пример использования командой формы текущей строки таблицы – когда из формы списка справочника вызывается команда, связанная с текущим элементом этого списка.

Например, вам нужно открыть список заказов, сделанных конкретным клиентом. Для отображения списка клиентов в форме списка справочника используется таблица Список, связанная с основным реквизитом Список (типа ДинамическийСписок), получающим данные из справочника Клиенты.

Добавьте в форму списка справочника команду Заказы и перетащите ее в командную панель формы. Установите свойство команды ИспользованиеТекущейСтроки в значение Использует, а в свойстве ИспользуемаяТаблица выберите таблицу Список (рис. 2.19).

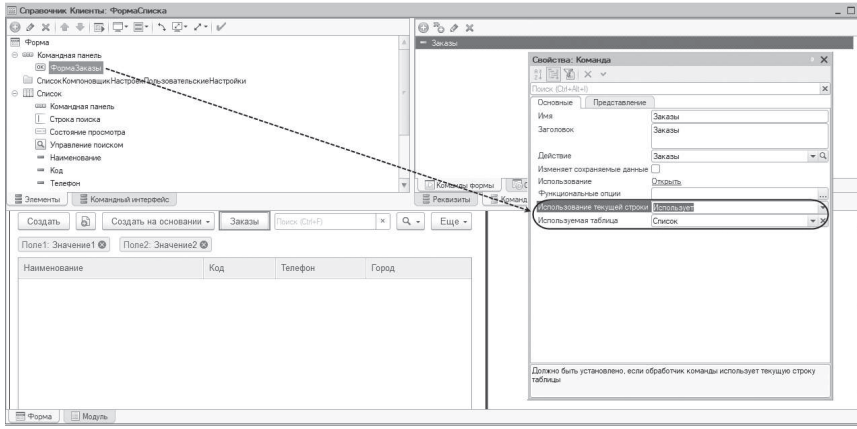


Рис. 2.19. Форма списка справочника «Клиенты» в конфигураторе

Таким образом вы задали, что команда Заказы будет использовать данные текущей строки таблицы Список. Создайте обработчик команды и заполните его следующим образом (листинг 2.2).

Листинг 2.2. Обработчик команды «Заказы»

```
&НаКлиенте
Процедура Заказы(Команда)

    ПараметрыФормы = Новый Структура("Отбор", Новый Структура(
                                        "Клиент", Элементы.Список.ТекущаяСтрока));
    ОткрытьФорму("Документ.Заказ.ФормаСписка", ПараметрыФормы);

КонецПроцедуры
```

В этом обработчике, используя свойство ТекущаяСтрока таблицы Список, вы получаете ссылку на текущий элемент списка клиентов. Эту ссылку вы устанавливаете как значение отбора по полю Клиент в структуре параметров формы и затем открываете форму списка документа Заказ с отбором по данному клиенту.

Обновите конфигурацию базы данных (F7), запустите мобильный клиент на планшете и откройте список клиентов.

Стандартно у таблицы списка свойство ИспользованиеТекущейСтроки установлено в значение Авто, поэтому она работает в режиме выбора конкретного клиента и показывает выделенную строку лишь кратковременно. Но, поскольку команда Заказы использует текущую строку таблицы, она будет доступна из контекстного меню, которое можно вызвать жестом пролистывания с правой стороны экрана нужной строки списка клиентов или же длительным нажатием на эту строку.

Вызовите контекстное меню у какого-либо клиента и выберите в нем команду Заказы. В результате в новом окне планшета откроется список заказов, сделанных выбранным клиентом (рис. 2.20).

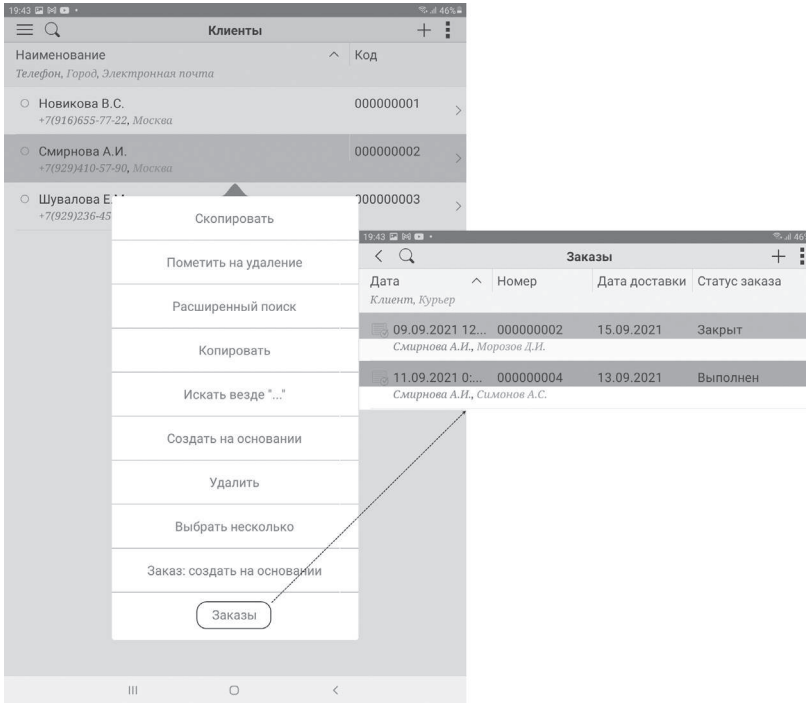


Рис. 2.20. Список заказов клиента

Использование текущей строки группой формы

Если в форме есть данные, связанные с текущей строкой таблицы, но они носят второстепенный, справочный характер, то такие данные лучше объединять в группы и указывать для групп свойства ИспользованиеТекущейСтроки и ИспользуемаяТаблица по аналогии с командами формы (см. рис. 2.19).

Для каждой используемой таблицы должна быть своя группа данных, так как одна группа может отображать данные только одной таблицы.

В мобильном клиенте в соответствии с концепцией лаконичности интерфейса группы, использующие текущую строку таблицы, отображаться не будут. Чтобы их увидеть, нужно вызвать контекстное меню какой-либо строки таблицы и выполнить команду Связанные данные. В результате откроется отдельное окно, в котором будут показаны все группы с данными, которые используют выбранную строку таблицы.

Например, в списке клиентов вы хотите отображать адресные данные клиента опционально, по специальной команде контекстного меню, вызванного у конкретной строки списка.

Для решения этой задачи в редакторе формы списка справочника Клиенты перетащите поля Улица, Дом и Квартира из основного реквизита Список в отдельную группу Дополнительно, которая будет использовать текущую строку этой таблицы (рис. 2.21).

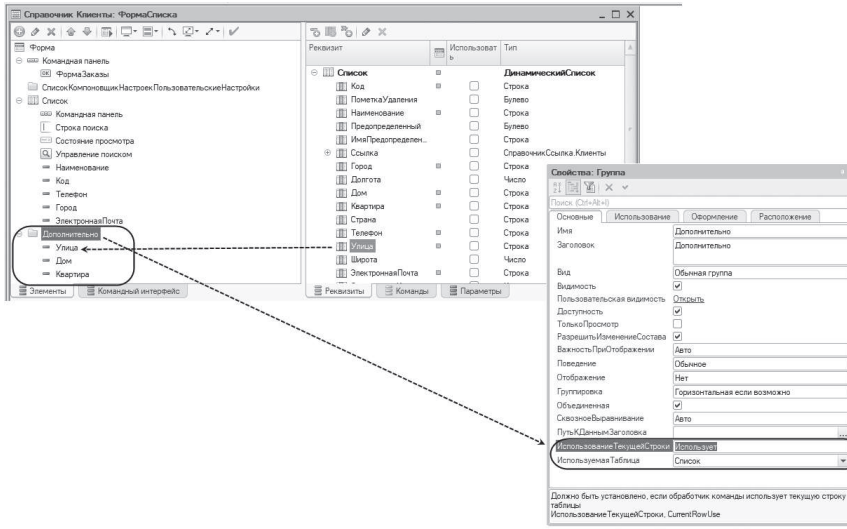


Рис. 2.21. Форма списка справочника «Клиенты» в конфигураторе

В результате в списке клиентов на планшете вы не увидите группы Дополнительно, так как она использует текущую строку списка. Все адресные данные, помещенные в эту группу, вы можете увидеть в специальном окне. Для этого вызовите контекстное меню с помощью жеста пролистывания справа у конкретной строки списка и выполните команду Связанные данные (рис. 2.22).

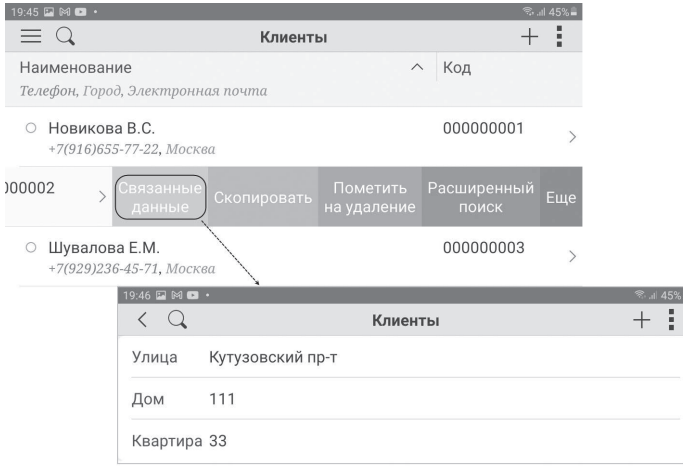


Рис. 2.22. Адресные данные клиента

Начальная страница

В мобильном клиенте отображаются все доступные формы начальной страницы (указанные в настройках рабочей области начальной страницы) за исключением форм, которые недоступны в соответствии с правами доступа, а также форм, отключенных функциональными опциями.

Отображаемые формы в мобильном клиенте располагаются в виде закладок, у которых переключатель закладок расположен сверху. При запуске мобильного клиента на первой закладке отображаются самая первая видимая форма и закладка в виде трех вертикальных точек, при нажатии на которую загружаются все остальные формы начальной страницы. После этого каждая форма будет доступна на отдельной закладке.

Например, в конфигураторе вы поместили в левую часть начальной страницы формы списка заказов и клиентов, а в правую – форму списка товаров (рис. 2.23).

При запуске мобильного клиента вы увидите закладку со списком заказов и три вертикальные точки справа от нее. При нажатии на них появятся отдельные закладки со списком клиентов и списком товаров (рис. 2.24).

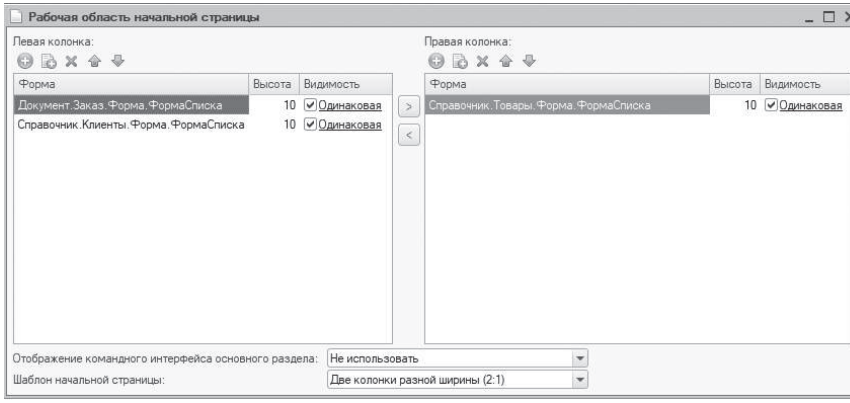


Рис. 2.23. Рабочая область начальной страницы

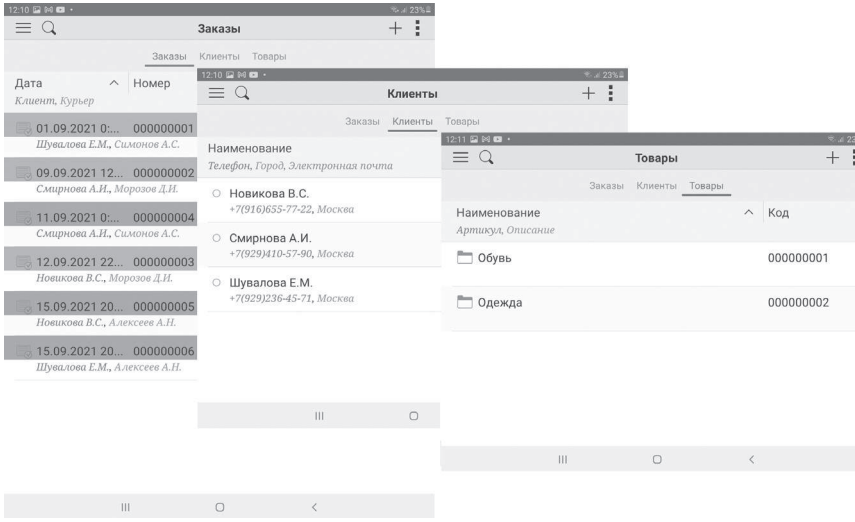


Рис. 2.24. Начальная страница приложения мобильного клиента

Возможности мобильных устройств

Как уже говорилось, функциональность мобильного клиента практически полностью соответствует функциональности «настольного» приложения, но при этом она может быть расширена за счет возможностей, специфичных для мобильных устройств.

Примеры реализации таких возможностей в основном рассматриваются в четвертой главе книги «Приложение мобильной платформы». Все перечисленные там функции можно реализовать и в мобильном клиенте, но, поскольку приложение мобильного клиента должно работать и на стационарном компьютере тоже, фрагменты кода, которые не будут работать в «настольном» приложении, в конфигурации нужно закрыть инструкцией #Если МобильныйКлиент. А функциональность, недоступная на мобильном клиенте, должна быть закрыта инструкцией препроцессора #Если Не МобильныйКлиент.

Работа с мультимедиа

Для примера автоматизируйте в мобильном клиенте задачу, когда курьер, находясь у клиента, должен сделать фото/видеосъемку товаров, записать аудиоотзыв клиента и т. п.

В вашей конфигурации существует справочник Хранимые файлы, в форме которого можно выбрать файл из локальной файловой системы и сохранить его содержимое в справочнике хранимых файлов, а также открыть содержимое хранимого файла соответствующим приложением и сохранить его в локальную файловую систему.

Теперь вам нужно реализовать в этой форме мультимедийные возможности, при этом файловое взаимодействие, описанное выше, также должно быть доступно в мобильном клиенте.

На примере разработки формы хранимого файла вы увидите, как создать универсальный код, работающий как на стационарном компьютере, так и на мобильном устройстве.

Итак, в форме элемента справочника Хранимые файлы существуют команды ВыбратьФайлСДискаИЗаписать, ПрочитатьФайлИСохранитьНаДиск, которые должны работать и в «настольном приложении», и в мобильном клиенте. Эти команды представлены в форме в виде обычных кнопок.

Добавьте в форму команды для работы со средствами мультимедиа: СделатьАудиозапись, СделатьВидеозапись, СделатьФотоснимок, которые будут работать только в мобильном клиенте. Перетащите их в форму и задайте представление соответствующих им кнопок в виде гиперссылок (рис. 2.25).

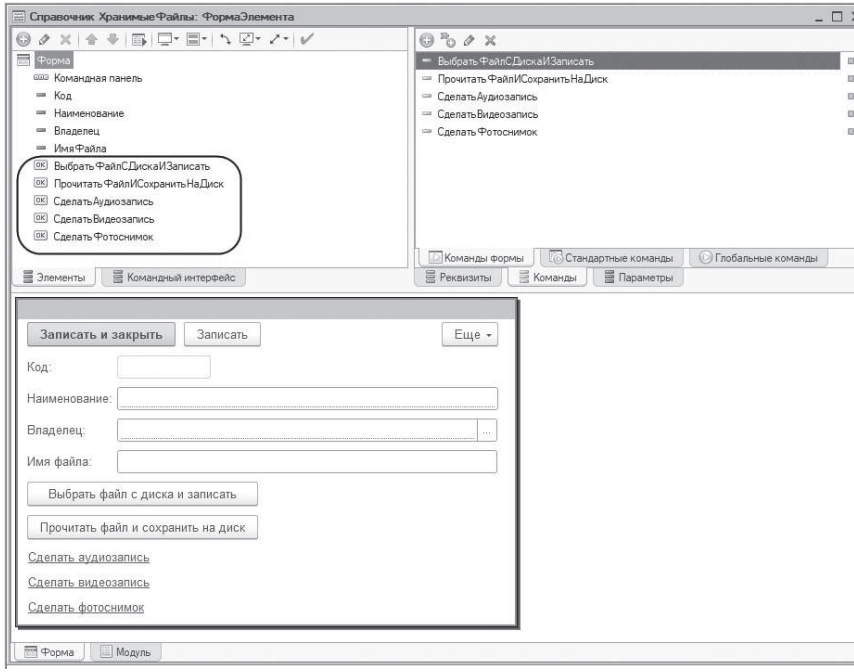


Рис. 2.25. Форма элемента справочника «Хранимые файлы» в конфигураторе

Создайте и заполните обработчик события формы ПриОткрытии(), в котором задайте видимость и доступность кнопок, связанных с этими командами (листинг 2.3).

Листинг 2.3. Обработчик события формы «ПриОткрытии»

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)

#Если НЕ МобильныйКлиент Тогда

        Элементы.СделатьАудиозапись.Видимость = Ложь;
        Элементы.СделатьВидеозапись.Видимость = Ложь;
        Элементы.СделатьФотоснимок.Видимость = Ложь;

#Иначе

        Если ТолькоПросмотр = Ложь Тогда
                Элементы.СделатьАудиозапись.Доступность
                                        = СредстваМультимедиа.ПоддерживаетсяАудиозапись();
                Элементы.СделатьВидеозапись.Доступность
                                        = СредстваМультимедиа.ПоддерживаетсяВидеозапись();
```

```
            Элементы.СделатьФотоснимок.Доступность
                                    = СредстваМультимедиа.ПоддерживаетсяФотоснимок();
        Иначе
            Элементы.СделатьАудиозапись.Доступность = Ложь;
            Элементы.СделатьВидеозапись.Доступность = Ложь;
            Элементы.СделатьФотоснимок.Доступность = Ложь;
        КонецЕсли;

#КонецЕсли

КонецПроцедуры
```

В этом обработчике устанавливается, что команды для работы со средствами мультимедиа будут видимы только при работе в мобильном клиенте (#Если МобильныйКлиент) и доступны только в том случае, если данные в форме хранимого файла можно редактировать и на мобильном устройстве поддерживаются мультимедийные возможности.

Таким образом, в форме хранимого файла, открытой на стационарном компьютере, пользователь не увидит команд для работы со средствами мультимедиа, как будто их и нет вообще (рис. 2.26).

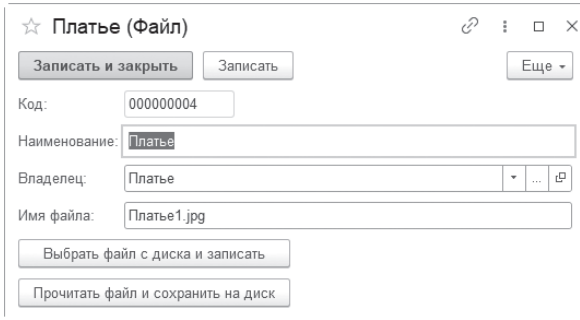


Рис. 2.26. Форма хранимого файла на стационарном компьютере

С помощью команд ВыбратьФайлСДискаИЗаписать и ПрочитатьФайлИСохранитьНаДиск пользователь (как «настольного» приложения, так и мобильного клиента) может выбрать файл из локальной файловой системы и сохранить его содержимое в справочнике хранимых файлов, а также сохранить его в локальную файловую систему.

ПОДРОБНЕЕ

О файловом взаимодействии подробно рассказывалось в книге «Технологии интеграции "1С:Предприятия 8.3"».

Подробное объяснение и описание работы с файловой системой не входит в задачу этой книги. Поэтому обработчики команд, выполняющих файловое взаимодействие, подробно рассматриваться не будут. Нужно лишь заметить, что в них используются асинхронные методы ПоместитьФайлНаСерверАсинх() и ПолучитьФайлССервераАсинх(), листинги 2.4–2.7.

Листинг 2.4. Обработчик команды «ВыбратьФайлСДискаИЗаписать»

```
&НаКлиенте
Процедура ВыбратьФайлСДискаИЗаписать(Команда)
    ЗапуститьВыборФайлаСДискаИЗапись();
КонецПроцедуры
```

Листинг 2.5. Процедура «ЗапуститьВыборФайлаСДискаИЗапись()»

```
&НаКлиенте
Асинх Процедура ЗапуститьВыборФайлаСДискаИЗапись()

    Если Не ЗначениеЗаполнено(Объект.Владелец) Тогда
        Ждать ПредупреждениеАсинх("Не заполнен владелец хранимого файла");
        Возврат;
    КонецЕсли;

    Попытка
        ОписаниеПомещенногоФайла = Ждать ПоместитьФайлНаСерверАсинх(
                                                    ,,,, УникальныйИдентификатор);
    Исключение
        Ждать ПредупреждениеАсинх("Ошибка помещения файла на сервер");
        Возврат;
    КонецПопытки;

    Если Не ОписаниеПомещенногоФайла = Неопределено Тогда
        НовыйОбъект = Объект.Ссылка.Пустая();
        Объект.ИмяФайла = ОписаниеПомещенногоФайла.ССылкаНаФайл.Имя;

        Если Не ЗначениеЗаполнено(Объект.Наименование) Тогда
#Если МобильныйКлиент Тогда
            Объект.Наименование = ОписаниеПомещенногоФайла.ССылкаНаФайл.Файл.
                                ПолучитьПредставлениеФайлаБиблиотекиМобильногоУстройства();
#Иначе
            ОписаниеФайла = Новый Файл(ОписаниеПомещенногоФайла.ССылкаНаФайл.Имя);
            Объект.Наименование = ОписаниеФайла.Имя;
#КонецЕсли
        КонецЕсли;

        ПоместитьФайлОбъекта(ОписаниеПомещенногоФайла.Адрес);

        Если НовыйОбъект Тогда
            ОтобразитьИзменениеДанных(Объект.Ссылка, ВидИзмененияДанных.Добавление);
        Иначе
            ОтобразитьИзменениеДанных(Объект.Ссылка, ВидИзмененияДанных.Изменение);
        КонецЕсли;
    КонецЕсли;

КонецПроцедуры
```

Листинг 2.6. Обработчик команды «ПрочитатьФайлИСохранитьНаДиск»

```
&НаКлиенте
Процедура ПрочитатьФайлИСохранитьНаДиск(Команда)
    ВыполнитьПрочитатьФайлИСохранитьНаДиск();
КонецПроцедуры
```

Листинг 2.7. Процедура «ВыполнитьПрочитатьФайлИСохранитьНаДиск()»

```
&НаКлиенте
Асинх Процедура ВыполнитьПрочитатьФайлИСохранитьНаДиск()

    Если Объект.Ссылка.Пустая() Тогда
        Ждать ПредупреждениеАсинх("Данные не записаны");
        Возврат;
    КонецЕсли;

    Если ПустаяСтрока(Объект.ИмяФайла) Тогда
        Ждать ПредупреждениеАсинх("Имя не задано");
        Возврат;
    КонецЕсли;

    Адрес = ПолучитьНавигационнуюССылку(Объект.Ссылка, "ДанныеФайла");
    ПутьКФайлу = СокрЛП(КаталогВременныхФайлов());
    Если Не СтрЗаканчиваетсяНа(ПутьКФайлу, ПолучитьРазделительПутиКлиента()) Тогда
        ПутьКФайлу = ПутьКФайлу + ПолучитьРазделительПутиКлиента();
    КонецЕсли;
    ПолученныеФайлы = Неопределено;

    Попытка
        ПолученныеФайлы = Ждать ПолучитьФайлССервераАсинх(Адрес, ПутьКФайлу + Объект.
                                                ИмяФайла, Новый ПараметрыДиалогаПолученияФайлов());
    Исключение
        Ждать ПредупреждениеАсинх("Ошибка получения файла с сервера");
        Возврат;
    КонецПопытки;

КонецПроцедуры
```

Содержимое файла в виде двоичных данных из временного хранилища помещается в элемент справочника хранимых файлов с помощью метода ПоместитьФайлОбъекта(), листинг 2.8.

Листинг 2.8. Процедура «ПоместитьФайлОбъекта()»

```
&НаСервере
Процедура ПоместитьФайлОбъекта(АдресВременногоХранилища)

    ЭлементСправочника = РеквизитФормыВЗначение("Объект");
    ДвоичныеДанные = ПолучитьИзВременногоХранилища(АдресВременногоХранилища);
    ЭлементСправочника.ДанныеФайла = Новый ХранилищеЗначения(
                                            ДвоичныеДанные, Новый СжатиеДанных());

    Файл = Новый Файл(ЭлементСправочника.ИмяФайла);
    ЭлементСправочника.ИмяФайла = Файл.Имя;
```

```
    ЭлементСправочника.Записать();

    Модифицированность = Ложь;
    УдалитьИзВременногоХранилища(АдресВременногоХранилища);
    ЗначениеВРеквизитФормы(ЭлементСправочника, "Объект");

КонецПроцедуры
```

Теперь вам нужно добавить в модуль формы обработчики команд для работы со средствами мультимедиа и процедуру ПоместитьМультимедиа(), с помощью которой мультимедийное содержимое будет сохраняться в элементе справочника хранимых файлов (листинг 2.9). Весь блок этих процедур нужно обрамить инструкциями препроцессора #Если МобильныйКлиент … #КонецЕсли, так как они будут выполняться только в мобильном клиенте и, значит, в «настольном» приложении их незачем компилировать.

Листинг 2.9. Блок процедур для работы со средствами мультимедиа

```
#Если МобильныйКлиент Тогда

&НаКлиенте
Процедура СделатьАудиозапись(Команда)

    ДанныеМультимедиа = СредстваМультимедиа.СделатьАудиозапись();
    ПоместитьМультимедиа(ДанныеМультимедиа);

КонецПроцедуры

&НаКлиенте
Процедура СделатьВидеозапись(Команда)

    ДанныеМультимедиа = СредстваМультимедиа.СделатьВидеозапись();
    ПоместитьМультимедиа(ДанныеМультимедиа);

КонецПроцедуры

&НаКлиенте
Процедура СделатьФотоснимок(Команда)

    Отметка = Новый ОтметкаНаФотоснимке(Истина, "ДФ='dd.ММ.уууу ЧЧ:мм'", "");
    ДанныеМультимедиа = СредстваМультимедиа.СделатьФотоснимок(,,,,,Отметка);
    ПоместитьМультимедиа(ДанныеМультимедиа);

КонецПроцедуры

&НаКлиенте
Процедура ПоместитьМультимедиа(ДанныеМультимедиа)

    Если ДанныеМультимедиа <> Неопределено Тогда
        НовыйОбъект = Объект.Ссылка.Пустая();
        АдресВременногоХранилища = ПоместитьВоВременноеХранилище(
                ДанныеМультимедиа.ПолучитьДвоичныеДанные(), УникальныйИдентификатор);
```

```
            ТипСодержимого = ДанныеМультимедиа.ТипСодержимого;
            Номер = Найти(ТипСодержимого, "/");
            Если Номер > 0 Тогда
                  ТипСодержимого = Лев(ТипСодержимого, Номер - 1);
            КонецЕсли;
            Объект.Наименование = ТипСодержимого + " " + Строка(ТекущаяДата());
            Объект.ИмяФайла = СтрЗаменить(Строка(ТекущаяДата()), ":", "_") + "."
                        + ДанныеМультимедиа.РасширениеФайла;

            ПоместитьФайлОбъекта(АдресВременногоХранилища);

            Если НовыйОбъект Тогда
                  ОтобразитьИзменениеДанных(Объект.Ссылка, ВидИзмененияДанных.Добавление);
            Иначе
                  ОтобразитьИзменениеДанных(Объект.Ссылка, ВидИзмененияДанных.Изменение);
            КонецЕсли;
      КонецЕсли;

КонецПроцедуры

#КонецЕсли
```

В обработчиках команд для работы с мультимедиа вызывается соответствующий метод (СделатьФотоснимок(), СделатьВидеозапись(), СделатьАудиозапись()) средств мультимедиа, для доступа к которым используется свойство глобального контекста СредстваМультимедиа.

В результате выполнения этих методов возвращается объект ДанныеМультимедиа, который передается в процедуру ПоместитьМультимедиа().

В этой процедуре из параметра ДанныеМультимедиа получается содержимое данных мультимедиа в виде двоичных данных и помещается во временное хранилище методом глобального контекста ПоместитьВоВременноеХранилище().

Затем заполняется наименование объекта (хранимого файла) на основе типа содержимого данных мультимедиа (image / video / audio) и текущей даты. Реквизиту объекта ИмяФайла присваивается строка, состоящая из текущей даты и расширения файла (jpg / mp4 / 3gp), полученного из параметра ДанныеМультимедиа.

Затем вызывается процедура ПоместитьФайлОбъекта() и в нее передается адрес временного хранилища, по которому было помещено содержимое мультимедиа (листинг 2.8). В результате данные мультимедиа сохраняются в справочнике Хранимые файлы.

Для того чтобы использовать в мобильном приложении функциональность, специфичную для мобильных устройств, вам нужно предоставить необходимые разрешения в свойстве конфигурации Используемая функцио-

нальность мобильного приложения. Установите сразу не только разрешения на работу со средствами мультимедиа, но и возможности работы с телефонией, контактами, уведомлениями и т. д. (рис. 2.27).

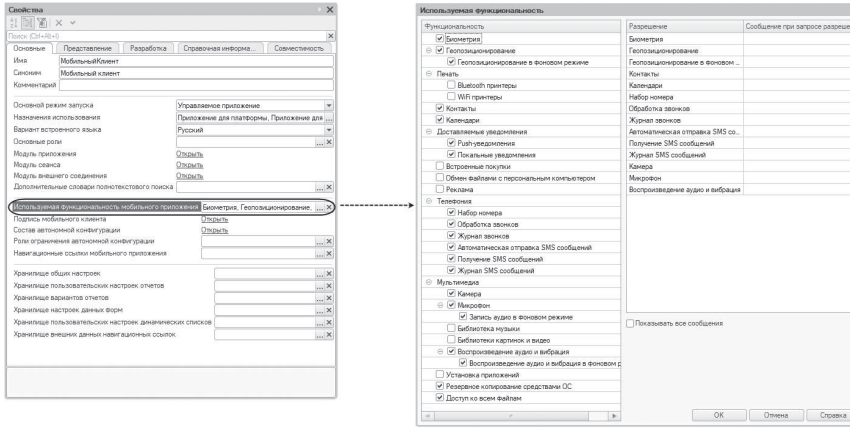


Рис. 2.27. Используемая функциональность мобильного приложения

В мобильном клиенте курьеру будут доступны все команды для работы с хранимыми файлами (рис. 2.28).

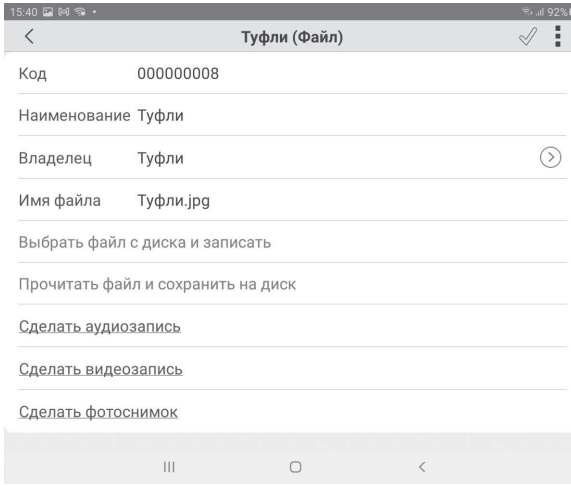


Рис. 2.28. Форма хранимого файла на планшете

Например, возможна такая ситуация. Если курьер доставляет клиенту товар с браком, то открыв карточку товара, на закладке Файлы курьер может добавить фото товара, на котором будет зафиксирован этот брак (рис. 2.29).

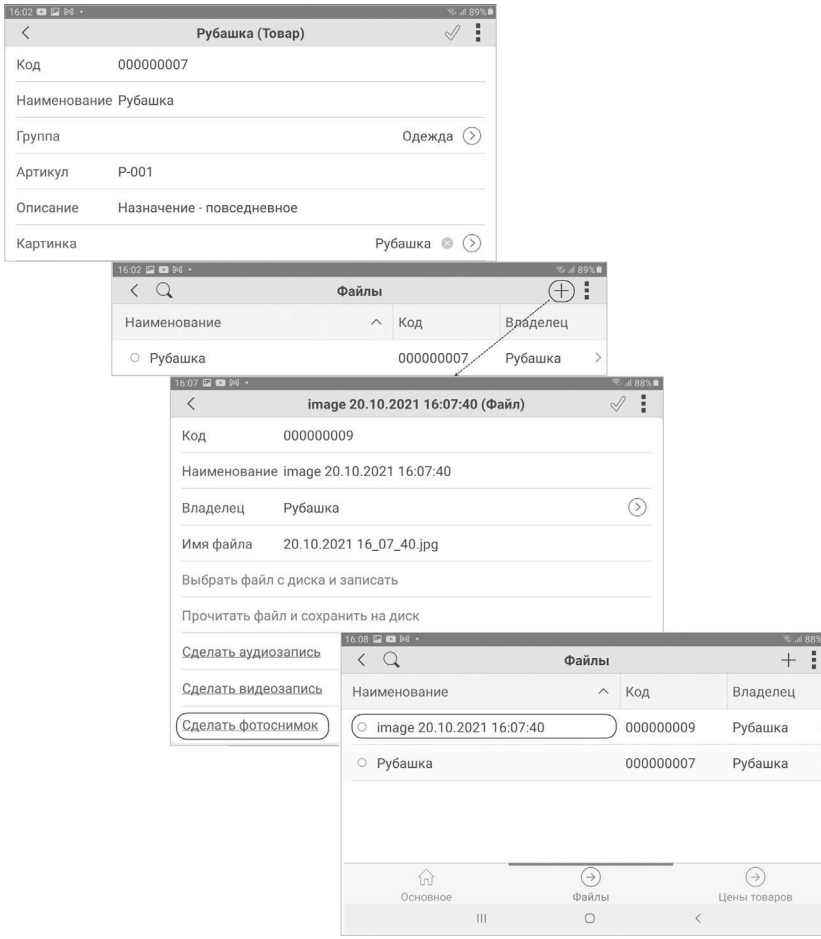


Рис. 2.29. Добавление фотоснимка товара

В конфигурации МобильныйКлиент в списке хранимых файлов реализована возможность открытия содержимого текущего файла соответствующим приложением.

Для этого в форму списка справочника Хранимые файлы добавлена команда ОткрытьФайл. Связанная с ней кнопка находится в командной панели формы. Обработчик команды выглядит следующим образом (листинг 2.10).

Листинг 2.10. Обработчик команды «ОткрытьФайл»

```
&НаКлиенте
Процедура ОткрытьФайл(Команда)

    ХранимыйФайл = Элементы.Список.ТекущиеДанные;
    ВыполнитьОткрытьФайл(ХранимыйФайл);

КонецПроцедуры

&НаКлиенте
Асинх Процедура ВыполнитьОткрытьФайл(ХранимыйФайл);

    Файл = Новый Файл(ХранимыйФайл.ИмяФайла);
    ИмяВременногоФайла = ПолучитьИмяВременногоФайла(Файл.Расширение);
    Адрес = ПолучитьНавигационнуюСсылку(ХранимыйФайл.Ссылка, "ДанныеФайла");

    Попытка
        ПолученныеФайлы = Ждать ПолучитьФайлССервераАсинх(Адрес, ИмяВременногоФайла);
    Исключение
        Ждать ПредупреждениеАсинх("Ошибка получения файла "
                                    + ИмяВременногоФайла + "с сервера");
        Возврат;
    КонецПопытки;

    ЗапуститьПриложениеАсинх(ИмяВременногоФайла);

КонецПроцедуры
```

С помощью свойства ТекущиеДанные таблицы Список данные текущего хранимого файла передаются в асинхронную процедуру, в которой происходит открытие файла ассоциированным с ним приложением с помощью асинхронного метода ЗапуститьПриложениеАсинх().

Чтобы эта команда работала в мобильном клиенте, вам нужно только установить свойство команды ИспользованиеТекущейСтроки в значение Использует, а в свойстве ИспользуемаяТаблица выбрать таблицу Список, как было показано в разделе «Использование текущей строки командой формы» (см. рис. 2.19).

В результате в списке хранимых файлов на планшете можно открыть его содержимое, нажав в контекстном меню Открыть файл (рис. 2.30).

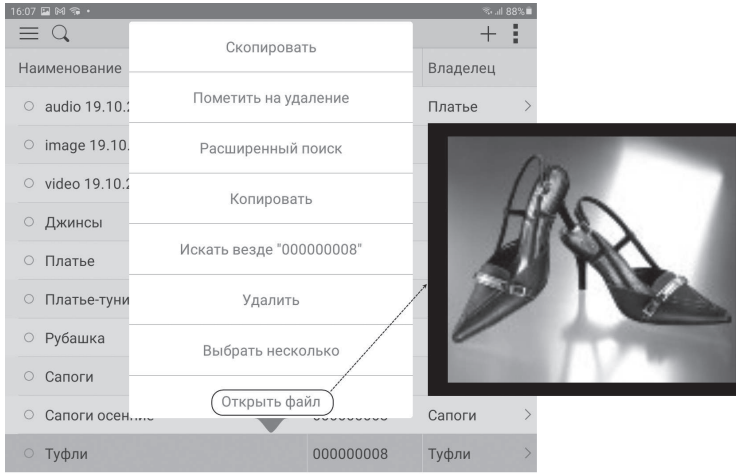


Рис. 2.30. Открытие файла из списка хранимых файлов

Работа с Push-уведомлениями

В процессе работы интернет-магазина может возникнуть необходимость срочно проинформировать курьера, работающего удаленно на мобильном устройстве, о наступлении каких-то событий в офисе. Например, о том, что изменилась важная для курьера информация.

Такие уведомления, которые отправляются из центральной базы мобильным устройствам, называются Push-уведомлениями.

Общая информация

Прежде всего, надо понимать, что в получении и рассылке Push-уведомлений (далее просто «уведомлений») участвуют три стороны:

- Собственно мобильное приложение, которое получает уведомления и реализует реакцию на них со стороны мобильного устройства.
- Прикладное решение (созданное на платформе «1С:Предприятие»), которое занимается рассылкой уведомлений. Оно выступает в роли отправителя уведомлений.
- Сервис доставки уведомлений. Это могут быть сервисы APNs (Apple Push Notification Service), FCM (Firebase Cloud Messaging), WNS (Windows Push Notification Services).

- Кроме того, фирмой «1С» разработан сервис «1С:Центр уведомлений» для отправки уведомлений, который и будет использоваться в показанном ниже примере. Этот сервис предназначен для облегчения реализации отправки уведомлений из тиражных прикладных решений «1С:Предприятия». Сервис позволяет изолировать получателей уведомлений одного отправителя от других отправителей, которые работают с тем же мобильным приложением. Кроме того, сервис позволяет избежать публикации конфиденциальной информации. Сервис расположен по адресу https://pushnotifications.1c.com.

В общем случае отправка уведомления выглядит следующим образом:

- Отправитель формирует уведомление.
- Отправитель определяет список получателей уведомления.
- Отправитель подключается к сервису уведомлений.
- Отправитель передает сервису уведомление и список получателей.
- Сервис обеспечивает доставку уведомлений на мобильные устройства.
- Мобильное приложение на устройстве обеспечивает обработку уведомлений.

Чтобы получение и рассылка Push-уведомлений стали возможными, отправитель должен знать идентификаторы получателей:

- Отправитель уведомления должен получить идентификатор получателя уведомлений от мобильного приложения.
- Получатель уведомления (мобильное приложение) должен передать идентификатор получателя уведомлений офисному приложению.

Надо заранее пояснить, что в мобильном клиенте это делается с помощью справочника мобильных устройств.

Чтобы использовать уведомления, следует установить разрешение Push-уведомления, о которых говорилось в предыдущем разделе (см. рис. 2.27).

ПОДРОБНЕЕ

Документация 1С:Предприятие 8.3.20. Руководство разработчика. Глава 29. Разработка для мобильных устройств > Мобильная версия «1С:Предприятия» > Специальные возможности мобильного устройства > Работа с уведомлениями > Push-уведомления.

Пример работы с Push-уведомлениями

Предположим, в заказах, находящихся в работе у курьера, изменилась какая-то важная информация. Например, клиент позвонил в офис и изменил адрес или время доставки. После записи изменений заказа офисное приложение должно послать курьеру, обслуживающему данный заказ, сообщение с информацией об этих изменениях.

Для реализации этой задачи откройте в конфигураторе форму объекта документа Заказ и добавьте в нее два новых реквизита: ВажнаяИнформация типа Строка (в нем будет храниться информация, передаваемая курьеру) и ОтправитьУведомление типа Булево (в нем будет храниться признак необходимости отправки уведомления).

В обработчике события формы ПриСозданииНаСервере признак отправки уведомлений ОтправитьУведомление установите в значение Ложь (листинг 2.11).

Листинг 2.11. Обработчик события формы «ПриСозданииНаСервере»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    ОтправитьУведомление = Ложь;
    ПересчитатьИтоговуюСумму();

КонецПроцедуры
```

Затем создайте обработчики событий ПриИзменении полей формы АдресДоставки и ДополнительнаяИнформация и заполните их следующим образом (листинг 2.12, 2.13).

Листинг 2.12. Обработчик события при изменении поля формы «АдресДоставки»

```
&НаКлиенте
Процедура АдресДоставкиПриИзменении(Элемент)

    Если Объект.СтатусЗаказа =
                    ПредопределенноеЗначение("Перечисление.СтатусыЗаказов.ВРаботе") Тогда
        ОтправитьУведомление = Истина;
        Если ВажнаяИнформация <> "" Тогда
            ВажнаяИнформация = ВажнаяИнформация + Символы.ПС +
                "Адрес доставки: " + СокрЛП(Объект.АдресДоставки);
        Иначе
            ВажнаяИнформация = "Адрес доставки: " + СокрЛП(Объект.АдресДоставки);
        КонецЕсли;
    КонецЕсли;

КонецПроцедуры
```

Листинг 2.13. Обработчик события при изменении поля формы «ДополнительнаяИнформация»

```
&НаКлиенте
Процедура ДополнительнаяИнформацияПриИзменении(Элемент)

    Если Объект.Важность И Объект.СтатусЗаказа =
                    ПредопределенноеЗначение("Перечисление.СтатусыЗаказов.ВРаботе") Тогда
        ОтправитьУведомление = Истина;
        Если ВажнаяИнформация <> "" Тогда
            ВажнаяИнформация = ВажнаяИнформация + Символы.ПС +
                "Дополнительно: " + СокрЛП(Объект.ДополнительнаяИнформация);
        Иначе
            ВажнаяИнформация = "Дополнительно: " +
                                СокрЛП(Объект.ДополнительнаяИнформация);
        КонецЕсли;
    КонецЕсли;

КонецПроцедуры
```

В этих обработчиках, если заказы находятся в работе у курьера (статус заказа – В работе) и если заказ содержит важную информацию, признак отправки уведомления ОтправитьУведомление устанавливается в значение Истина и заполняется строковый реквизит ВажнаяИнформация, который после записи заказа будет передан курьеру в тексте уведомления.

Чтобы реализовать формирование и отправку уведомления, создайте обработчик события формы заказа ПослеЗаписиНаСервере и заполните его следующим образом (листинг 2.14).

Листинг 2.14. Обработчик события формы «ПослеЗаписиНаСервере»

```
&НаСервере
Процедура ПослеЗаписиНаСервере(ТекущийОбъект, ПараметрыЗаписи)

    Если ОтправитьУведомление = Истина Тогда
        Уведомление = Новый ДоставляемоеУведомление();
        Уведомление.Текст = "Заказ " + ТекущийОбъект.Номер +
                " - Изменилась важная информация." + Символы.ПС + ВажнаяИнформация;
        УведомленияСервер.ОтправитьУведомление(Уведомление,
                ТекущийОбъект.Курьер);
    КонецЕсли;

КонецПроцедуры
```

В этом обработчике, если признак отправки уведомления установлен, вы создаете объект ДоставляемоеУведомление, заполняете его свойство Текст так, чтобы проинформировать курьера, что в заказе изменилась важная информации (что именно изменилось – содержится в реквизите ВажнаяИнформация).

Затем вы вызываете процедуру ОтправитьУведомление() общего модуля УведомленияСервер, в которую передаете само уведомление и ссылку на курьера, которому его надо доставить. В этой процедуре (она будет рассмотрена позже, см. листинг 2.17) и происходят определение списка получателей уведомления и отправка уведомления через сервис «1С:Центр уведомлений».

Чтобы подключиться к нему, нужно перед рассылкой уведомлений зарегистрировать отправителя уведомлений на этом сервисе по адресу https://pushnotifications.1c.com и получить для него ключ доступа, который будет храниться в константе КлючДоступаОтправителя.

Добавьте эту константу в вашу конфигурацию и установите ее свойства: Синоним – Ключ доступа, Тип – Строка, Длина – 54, Использовать стандартные команды – Нет.

А также добавьте еще две константы типа Булево: ИспользоватьPushУведомления (для хранения признака использования рассылки Push-уведомлений) и ВоспроизводитьТекстУведомления (для хранения признака звукового воспроизведения текста уведомлений). Свойство этих констант Использовать стандартные команды установите в значение Нет.

Теперь создайте общую форму констант с именем Настройки. Чтобы при настройке рассылки Push-уведомлений администратор мог сгенерировать ключ отправителя на сервисе «1С:Центр уведомлений», добавьте в форму команду ПолучитьКлюч, перетащите ее в форму и объедините ее в группу с константой КлючДоступаОтправителя. Обработчик этой команды заполните следующим образом (листинг 2.15).

Листинг 2.15. Обработчик команды «ПолучитьКлюч»

```
&НаКлиенте
Процедура ПолучитьКлюч(Команда)

    ПерейтиПоНавигационнойСсылке("https://pushnotifications.1c.com/push/publishers/new");

КонецПроцедуры
```

Таким образом, нажав на кнопку Получить ключ в форме Настройки, специалист по внедрению сможет перейти на сайт сервиса «1С», зарегистрироваться там (если это еще не было сделано) и зарегистрировать приложение-отправитель уведомлений. Полученный ключ нужно будет скопировать и вставить в форму настроек в поле Ключ доступа.

Чтобы сделать интерфейс более дружественным, можно добавить расширенную подсказку для поля КлючДоступаОтправителя.

Для этого вызовите из контекстного меню этого поля пункт Показать расширенную подсказку и заполните свойство Заголовок этой подсказки поясняющим текстом: «*Чтобы получить ключ доступа для рассылки уведомлений на сервисе «1С:Центр уведомлений», нажмите кнопку "Получить ключ" и зарегистрируйтесь на сервисе. Затем зарегистрируйте отправителя уведомлений, скопируйте сгенерированный для него ключ доступа и впишите его в поле "Ключ доступа"*» (рис. 2.31).

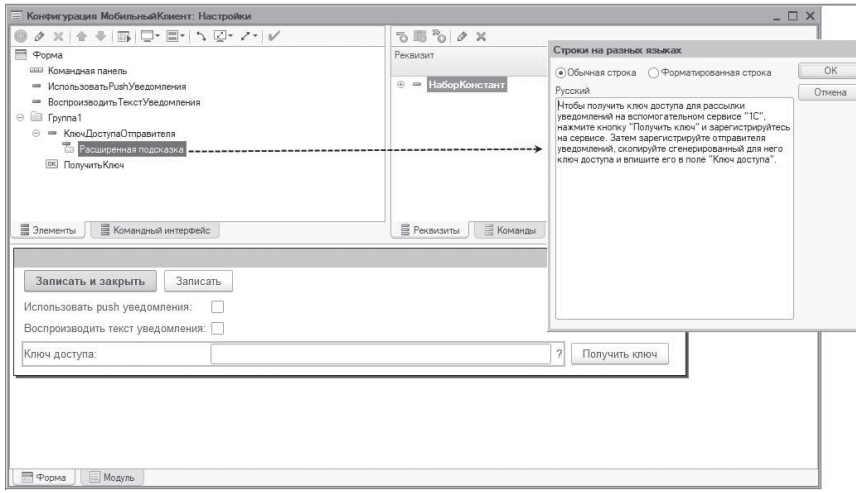


Рис. 2.31. Форма «Настройки» в конфигураторе

Затем у поля КлючДоступаОтправителя установите свойство ОтображениеПодсказки в значение Кнопка.

В результате в режиме «1С:Предприятие» справа от поля Ключ доступа в форме настроек появится кнопка в виде вопросительного знака, нажав на которую, можно увидеть расширенную подсказку (рис. 2.32).

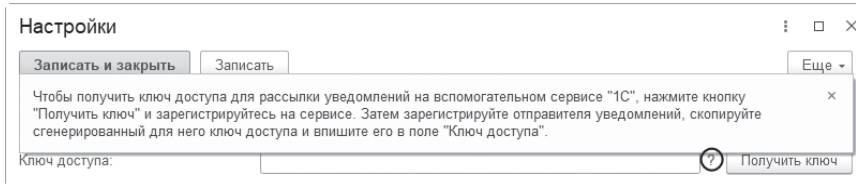


Рис. 2.32. Форма «Настройки» в режиме «1С:Предприятие»

Для хранения идентификатора получателей уведомлений вам понадобится справочник мобильных устройств. При запуске приложения-получателя (мобильного клиента) на мобильном устройстве оно должно обновлять (так как идентификатор устройства может измениться) и сохранять в этом справочнике свой идентификатор, а при отправке уведомлений приложение-отправитель должно получить его из этого справочника.

Кроме того, вам понадобится справочник пользователей, которому будет подчинен справочник мобильных устройств. Это нужно для того, чтобы приложение-отправитель уведомлений могло идентифицировать получателя по текущему пользователю мобильного клиента. В рассматриваемом примере – чтобы уведомление получал только тот курьер, которому оно предназначено.

В конфигурации МобильныйКлиент уже существует справочник Пользователи. При тестировании отправки и получения Push-уведомлений вам нужно будет только немного изменить его, чтобы коды пользователей соответствовали пользователям информационной базы (см. рис. 2.34).

Добавьте справочник МобильныеУстройства. Затем добавьте реквизиты справочника: ИдентификаторПодписчикаДоставляемыхУведомлений типа ХранилищеЗначения (в нем будет храниться идентификатор получателя уведомлений, который определяет конкретную информационную базу в определенном приложении на конкретном устройстве) и Подписчик типа Строка (36) фиксированной длины (в нем будет храниться идентификатор клиента для данного пользователя конкретного мобильного устройства).

После этого откройте модуль менеджера справочника МобильныеУстройства и добавьте туда процедуру для записи идентификатора получателя уведомлений в реквизит справочника ИдентификаторПодписчикаДоставляемыхУведомлений (листинг 2.16).

Листинг 2.16. Процедура «НовыйИдентификаторПодписчикаУведомлений()»

```
Процедура НовыйИдентификаторПодписчикаУведомлений(Подписчик, Идентификатор) Экспорт

    Запрос = Новый Запрос;
    Запрос.Текст = "ВЫБРАТЬ
                   |    МобильныеУстройства.Ссылка КАК Ссылка,
                   |    МобильныеУстройства.ПометкаУдаления КАК ПометкаУдаления
                   |ИЗ
                   |    Справочник.МобильныеУстройства КАК МобильныеУстройства
                   |ГДЕ
                   |    МобильныеУстройства.Подписчик = &Подписчик";
    Запрос.УстановитьПараметр("Подписчик", Подписчик);
    Выборка = Запрос.Выполнить().Выбрать();
    Устройство = 0;
    Пока Выборка.Следующий() Цикл
```

```
            Если Выборка.ПометкаУдаления = Ложь Тогда
                Устройство = Выборка.Ссылка.ПолучитьОбъект();
                Прервать;
            КонецЕсли;
        КонецЦикла;

        Если Устройство = 0 Тогда
        Устройство = СоздатьЭлемент();
            Пользователь = ПользователиИнформационнойБазы.ТекущийПользователь();
            Устройство.Владелец = Справочники.Пользователи.НайтиПоКоду(Пользователь);
            Устройство.Подписчик = Подписчик;
        КонецЕсли;

        Устройство.ИдентификаторПодписчикаДоставляемыхУведомлений =
                                                    Новый ХранилищеЗначения(Идентификатор);
        Устройство.Записать();

КонецПроцедуры
```

В эту процедуру в качестве параметров Подписчик и Идентификатор передаются уникальный идентификатор клиента конкретного мобильного устройства и идентификатор получателя уведомлений конкретного мобильного приложения, полученные при запуске мобильного клиента. Вызов этой процедуры будет рассмотрен ниже в листинге 2.21.

Из справочника МобильныеУстройства выбираются все записи для данного устройства и пользователя ОС, содержащегося в параметре Подписчик.

Если такого подписчика еще нет, то в справочнике создается новый элемент. Его поле Владелец заполняется ссылкой на текущего пользователя мобильного клиента, а поле Подписчик – значением параметра Подписчик.

И у нового или уже существующего объекта поле ИдентификаторПодписчикаДоставляемыхУведомлений заполняется значением параметра Идентификатор, помещенным в хранилище значения. Затем объект записывается.

Таким образом, с помощью процедуры НовыйИдентификаторПодписчикаУведомлений(), определенной в модуле менеджера справочника МобильныеУстройства, для каждого подписчика будет получен и сохранен идентификатор получателя уведомлений конкретного мобильного приложения. Позже вы сделаете так, чтобы этот идентификатор регулярно обновлялся.

Теперь, наконец, пришло время заняться процедурой отправки уведомлений. Создайте в конфигурации общий неглобальный серверный модуль УведомленияСервер (с признаком Вызов сервера) и напишите в нем процедуру ОтправитьУведомление(). Первым параметром в нее передается само уведомление, во втором параметре содержится либо ссылка на пользователя мобильного приложения (курьера, которому надо передать уведомление), либо Неопределено (листинг 2.17).

Листинг 2.17. Процедура «ОтправитьУведомление()»

```
Процедура ОтправитьУведомление(Уведомление, Пользователь) Экспорт

	ИспользоватьPushУведомления = Константы.ИспользоватьPushУведомления.Получить();
	Если НЕ ИспользоватьPushУведомления Тогда
		Возврат;
	КонецЕсли;

	Выборка = Справочники.МобильныеУстройства.Выбрать();
	Пока Выборка.Следующий() Цикл
		Если Выборка.ИдентификаторПодписчикаДоставляемыхУведомлений <> Неопределено Тогда
			Если Пользователь = Неопределено ИЛИ Пользователь = Выборка.Владелец Тогда
				Идентификатор
				= Выборка.ИдентификаторПодписчикаДоставляемыхУведомлений.Получить();
				Если Идентификатор <> Неопределено Тогда
					Уведомление.Получатели.Добавить(Идентификатор);
				КонецЕсли;
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;

	Если Уведомление.Получатели.Количество() > 0 Тогда
		ДанныеАутентификации = "";
		ДанныеАутентификации = СокрЛП(Константы.КлючДоступаОтправителя.Получить());

		УдаленныеТокены = Новый Массив;
		ОтправкаДоставляемыхУведомлений.Отправить(
				Уведомление, ДанныеАутентификации, УдаленныеТокены, Истина);
		НеИспользоватьИдентификаторы(УдаленныеТокены);
	КонецЕсли;

КонецПроцедуры
```

Эта процедура выполняется в том случае, если возможность отправки Push-уведомлений включена.

Сначала вы определяете массив получателей уведомления. Для этого вы обходите выборку из справочника МобильныеУстройства и для каждого элемента выборки анализируете значение реквизита ИдентификаторПодписчикаДоставляемыхУведомлений. Если он определен, проверяется: если значение параметра Пользователь, переданного в процедуру, не определено или если в реквизите Владелец содержится ссылка на этого пользователя, то этому подписчику нужно отправить уведомление. В этом случае вы получаете идентификатор подписчика доставляемых уведомлений из соответствующего реквизита элемента справочника и добавляете в массив получателей уведомления (Уведомление.Получатели.Добавить(Идентификатор)).

Если массив получателей уведомления содержит хотя бы один элемент, то в переменной ДанныеАутентификации вы присваиваете значение константы КлючДоступаОтправителя, в которой позже будет сохранен ключ доступа, полученный для отправителя уведомлений на сервисе «1С:Центр уведомлений» при настройке Push-уведомлений. При вводе (копировании) ключа доступа с сервиса можно допустить ошибку, поэтому на всякий случай незначащие символы в начале и в конце ключа лучше программно удалить.

Затем с помощью свойства глобального контекста ОтправкаДоставляемыхУведомлений вы обращаетесь к менеджеру отправки доставляемых уведомлений и методом Отправить() отправляете уведомление всем получателям, определенным выше. В первом параметре вы передаете само Уведомление, во втором – ДанныеАутентификации, которые необходимы для подключения к сервису отправки уведомлений (в данном случае это ключ доступа к сервису «1С:Центр уведомлений»).

Третьим параметром вы передаете в метод Отправить() пустой массив УдаленныеТокены, в который возвращается массив строк с идентификаторами устройств, которые не подходят для отправки и, соответственно, должны быть исключены из будущих рассылок. Это справедливо только для устройств под управлением Android, на устройствах под iOS этот параметр игнорируется. И четвертым параметром вы передаете Истина, так как для отправки уведомлений используется сервис «1С:Центр уведомлений».

В случае если массив УдаленныеТокены заполнен, надо исключить устройства, до которых не удалось доставить уведомления (например, пользователь удалил мобильное приложение), из будущих рассылок. Для этого вызывается процедура НеИспользоватьИдентификаторы() (расположенная в том же общем модуле УведомленияСервер), в которую передается массив строк с идентификаторами таких устройств (листинг 2.18).

Листинг 2.18. Процедура «НеИспользоватьИдентификаторы()»

```
Процедура НеИспользоватьИдентификаторы(Токены)

    Если Токены.Количество() > 0 Тогда
        Выборка = Справочники.МобильныеУстройства.Выбрать();
        Пока Выборка.Следующий() Цикл
            Если Выборка.ИдентификаторПодписчикаДоставляемыхУведомлений <> Неопределено
                Тогда
                Идентификатор =
                    Выборка.ИдентификаторПодписчикаДоставляемыхУведомлений.Получить();
                Если Идентификатор <> Неопределено И Токены.Найти(
                        Идентификатор.ИдентификаторУстройства) <> Неопределено Тогда
                    Узел = Выборка.ПолучитьОбъект();
                    Узел.ИдентификаторПодписчикаДоставляемыхУведомлений = Неопределено;
```

```
                        Узел.Записать();
                    КонецЕсли;
                КонецЕсли;
            КонецЦикла;
        КонецЕсли;

КонецПроцедуры
```

Если массив УдаленныеТокены заполнен, выборка из справочника МобильныеУстройства обходится в цикле, и для каждого элемента выборки анализируется значение реквизита ИдентификаторПодписчикаДоставляемыхУведомлений.

Если он определен, значение идентификатора извлекается из хранилища значения методом Получить(). Если в полученном идентификаторе устройства содержится такая же строка, как и в массиве, переданном в процедуру в качестве параметра, то для текущего элемента выборки получается объект, его реквизит ИдентификаторПодписчикаДоставляемыхУведомлений устанавливается в значение Неопределено и объект записывается.

Таким образом, пока приложение мобильного клиента, работающее на конкретном мобильном устройстве, снова не запишет свой идентификатор подписчика доставляемых уведомлений в справочник мобильных устройств, оно не попадет в список рассылки.

Итак, вы реализовали ту часть функциональности, которая будет работать на стороне отправителя Push-уведомлений. Теперь вам осталось реализовать другую часть, которая будет исполняться в мобильном клиенте на стороне получателя уведомлений. Важно понимать, что для разрабатываемой вами конфигурации мобильного клиента это деление условно, так как все это делается в одной и той же конфигурации.

Прежде всего мобильный клиент должен иметь возможность регулярно получать и сохранять в справочнике мобильных устройств свой идентификатор получателя уведомлений для конкретного мобильного устройства. Кроме того, вам нужно реализовать реакцию мобильного клиента на получение Push-уведомлений.

Нужно учитывать, что идентификатор подписчика уведомлений может измениться, поэтому его рекомендуется получать и обновлять регулярно. Поэтому разместите процедуру обновления идентификатора подписчика уведомления в модуле приложения в обработчике события ПриНачалеРаботыСистемы.

Откройте модуль приложения и поместите в него следующий код (листинг 2.19).

Листинг 2.19. Обработчик события «ПриНачалеРаботыСистемы» в модуле приложения

```
Процедура ПриНачалеРаботыСистемы()

#Если МобильныйКлиент Тогда
    // идентификатор подписчика надо получать регулярно, он может измениться
    УведомленияКлиент.ОбновитьИдентификаторПодписчикаУведомлений();

    ...
#КонецЕсли

КонецПроцедуры
```

Затем создайте общий неглобальный клиентский модуль УведомленияКлиент и поместите в нем процедуру ОбновитьИдентификаторПодписчикаУведомлений(), листинг 2.20.

Листинг 2.20. Процедура «ОбновитьИдентификаторПодписчикаУведомлений()»

```
Процедура ОбновитьИдентификаторПодписчикаУведомлений() Экспорт

#Если МобильныйКлиент Тогда

    Попытка
        ИдентификаторПодписчикаУведомлений =
                ДоставляемыеУведомления.ПолучитьИдентификаторПодписчикаУведомлений();
        ТекстОшибки = "";

        УведомленияСервер.ПередатьИдентификаторПодписчикаУведомлений(
                ИдентификаторПодписчикаУведомлений, ТекстОшибки);
        Если ТекстОшибки <> "" Тогда
            Сообщить(ТекстОшибки);
        КонецЕсли
    Исключение
        Инфо = ИнформацияОбОшибке();
        ПоказатьИнформациюОбОшибке(Инфо);
    КонецПопытки;

#КонецЕсли

КонецПроцедуры
```

В процедуре с помощью свойства глобального контекста ДоставляемыеУведомления вы обращаетесь к менеджеру доставляемых уведомлений и методом ПолучитьИдентификаторПодписчикаУведомлений() получаете уникальный идентификатор подписчика уведомлений для конкретного мобильного устройства, на котором работает мобильное приложение.

Затем с помощью процедуры ПередатьИдентификаторПодписчикаУведомлений() модуля УведомленияСервер вы сохраняете этот идентификатор в справочнике МобильныеУстройства.

Откройте этот общий модуль и поместите в нем указанную процедуру. В качестве параметров в нее передаются полученный идентификатор подписчика уведомлений и пустая строка для заполнения текста ошибки (листинг 2.21).

Листинг 2.21. Процедура «ПередатьИдентификаторПодписчикаУведомлений()»

```
Процедура ПередатьИдентификаторПодписчикаУведомлений(ИдентификаторПодписчикаУведомлений
                                                    , ТекстОшибки) Экспорт

    СисИнфо = Новый СистемнаяИнформация;
    Справочники.МобильныеУстройства.НовыйИдентификаторПодписчикаУведомлений(Строка(
                                СисИнфо.ИдентификаторКлиента), ИдентификаторПодписчикаУведомлений);

КонецПроцедуры
```

В этой процедуре с помощью объекта СистемнаяИнформация вы получаете уникальный идентификатор клиента для конкретного мобильного устройства и конкретного пользователя операционной системы.

Затем вы вызываете процедуру НовыйИдентификаторПодписчикаУведомлений() менеджера справочника МобильныеУстройства, в которую передаются уникальный идентификатор клиента и идентификатор подписчика уведомлений. Эта процедура была подробно описана и объяснена выше, в листинге 2.16.

Теперь вам осталось сделать так, чтобы мобильное приложение реагировало на получение Push-уведомлений. Для этого нужно подключить обработчик уведомлений в модуле приложения в обработчике события ПриНачалеРаботыСистемы.

Откройте этот обработчик и добавьте в него следующий код (листинг 2.22).

Листинг 2.22. Обработчик события «ПриНачалеРаботыСистемы» в модуле приложения

```
Процедура ПриНачалеРаботыСистемы()

#Если МобильныйКлиент Тогда
    // идентификатор подписчика надо получать регулярно, он может измениться
    УведомленияКлиент.ОбновитьИдентификаторПодписчикаУведомлений();

    // Подключение обработчика Push-уведомлений
    ОписаниеОповещения = Новый ОписаниеОповещения("ОбработкаУведомлений", УведомленияКлиент);
    ДоставляемыеУведомления.ПодключитьОбработчикУведомлений(ОписаниеОповещения);

#КонецЕсли

КонецПроцедуры
```

В этом обработчике методом ПодключитьОбработчикУведомлений менеджера доставляемых уведомлений в качестве обработчика уведомлений вы подключаете процедуру ОбработкаУведомлений(), находящуюся в общем модуле УведомленияКлиент и описанную в объекте ОписаниеОповещения.

Поместите эту процедуру в общем модуле УведомленияКлиент. Обратите внимание, что процедура асинхронная (описана с модификатором Асинх) и выполняется только на мобильном клиенте, так как обрамлена инструкцией препроцессора #Если МобильныйКлиент … #КонецЕсли (листинг 2.23).

Листинг 2.23. Процедура «ОбработкаУведомлений()»

```
#Если МобильныйКлиент Тогда

Асинх Процедура ОбработкаУведомлений(Уведомление, Локальное, Показано, Параметры) Экспорт

    Если НЕ Показано Тогда
        // Иначе пользователь оповещен об уведомлении системными средствами.
        СредстваМультимедиа.ВоспроизвестиЗвуковоеОповещение();
    КонецЕсли;
    Если УведомленияСервер.ВоспроизводитьТекстУведомления() Тогда
        СредстваМультимедиа.ВоспроизвестиТекст(Уведомление.Текст);
    КонецЕсли;
    Если Локальное Тогда
        …
    Иначе
        Если НЕ Показано Тогда
            Ждать ПредупреждениеАсинх(Уведомление.Текст);
        КонецЕсли;
    КонецЕсли;

КонецПроцедуры

#КонецЕсли
```

Когда в офисном приложении в заказе меняется какая-то важная для курьера информация, то ему приходит Push-уведомление с текстом сообщения. В случае если приложение не запущено или работает в фоновом режиме, оно показывается средствами операционной системы (в списке уведомлений планшета).

Если в форме настроек признак (константа) ВоспроизводитьТекстУведомления включен, то текст уведомления воспроизводится средствами мультимедиа методом ВоспроизвестиТекст().

Если приложение запущено и активно, то уведомление сразу передается в мобильное приложение и не показывается пользователю (параметр Показано = Ложь). В этом случае (в последней секции оператора условия) для привлечения внимания курьера оно показывается с помощью асинхронного метода ПредупреждениеАсинх().

Обработка получения Push-уведомлений мало чем отличается от обработки локальных уведомлений. Поэтому в ветке условия Если Локальное ... вы можете в дальнейшем добавить обработку локальных уведомлений на планшете – например, напоминание о звонке клиенту и т. п.

Тестирование отправки и получения Push-уведомлений

Прежде всего в конфигураторе добавьте пользователей информационной базы Администратор, Курьер, Менеджер (рис. 2.33).

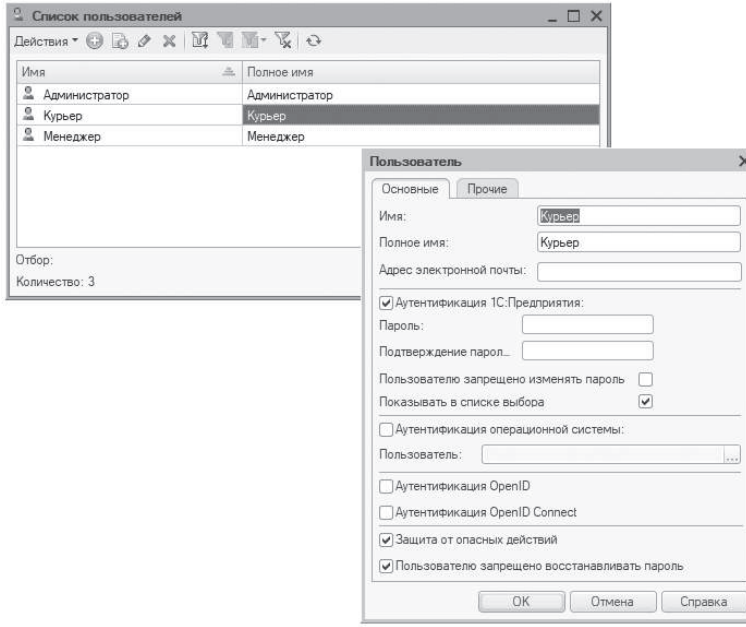


Рис. 2.33. Список пользователей информационной базы в конфигураторе

Затем запустите конфигурацию в режиме исполнения на стационарном компьютере, откройте справочник Пользователи и внесите туда пользователей с соответствующими кодами (рис. 2.34).

Рис. 2.34. Список пользователей информационной базы в режиме «1С:Предприятие»

Это нужно потому, что курьер при входе в мобильный клиент на планшете будет обновлять и сохранять свой идентификатор подписчика доставляемых уведомлений в справочнике Мобильные устройства. Этот пользователь (курьер) будет являться владельцем данного подписчика уведомлений, и по этому владельцу при отправке уведомления определяется, нужно ли посылать этому подписчику уведомление.

Теперь настройте в приложении-отправителе возможность рассылки Push-уведомлений через сервис «1С:Центр уведомлений». Для этого в группе Сервис выполните команду Настройки и откройте форму настроек приложения (см. рис. 2.32).

Включите признак Использовать Push-уведомления. Затем нажмите Получить ключ и зарегистрируйтесь на открывшемся сайте сервиса «1С:Центр уведомлений» (если это еще не было сделано).

После этого в разделе Доставляемые уведомления > Отправители уведомлений зарегистрируйте ваше приложение в качестве нового отправителя уведомлений – придумайте и заполните поля Код отправителя и Наименование отправителя и нажмите Сохранить. В списке отправителей уведомлений нажмите Редактировать, скопируйте сгенерированный для вашего приложения ключ из поля Ключ доступа отправителя и вставьте его в форму настроек в поле Ключ доступа. Нажмите Записать и закрыть (рис. 2.35).

Теперь зайдите в приложение мобильного клиента на планшете от имени пользователя Курьер (рис. 2.36).

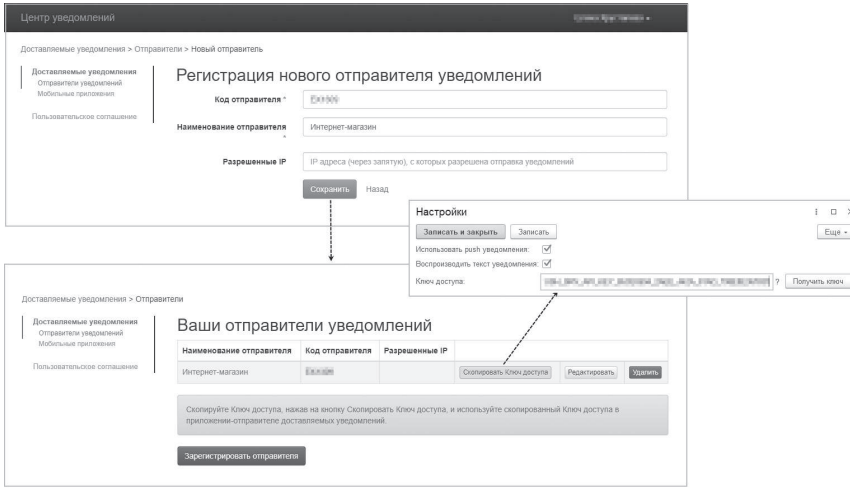


Рис. 2.35. Настройка Push-уведомлений в приложении-отправителе уведомлений

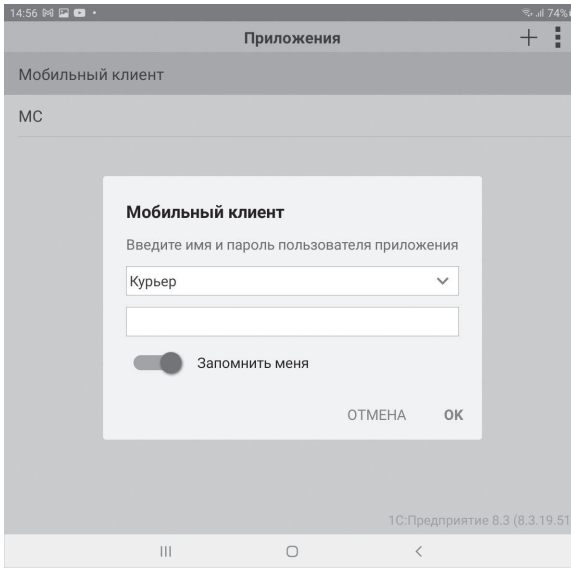


Рис. 2.36. Авторизация курьера в мобильном клиенте

При запуске мобильного клиента в первый раз в справочник Мобильные устройства добавляется элемент, владельцем которого является авторизовавшийся в нем пользователь, в данном случае курьер Морозов. Поле Подписчик заполняется уникальным идентификатором клиента на данном планшете, а в поле ИдентификаторПодпискиДоставляемыхУведомлений (которого вы не видите, т. к. оно имеет тип ХранилищеЗначения) сохраняется идентификатор подписчика уведомлений (рис. 2.37).

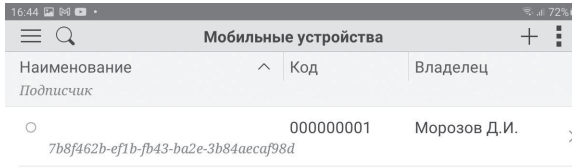


Рис. 2.37. Данные справочника «Мобильные устройства»

При последующих входах в мобильный клиент этого же курьера данные в справочнике будут обновляться.

Перед рассылкой уведомления приложение-отправитель получает идентификатор доставляемых уведомлений для конкретного подписчика (мобильного устройства и пользователя) из справочника мобильных устройств и по его владельцу определяет, нужно ли добавлять этого пользователя в список получателя уведомления.

«Поработайте» немного с мобильным клиентом, чтобы приложение было активно.

Затем откройте в «настольном» приложении какой-нибудь заказ для курьера Морозов с установленным флажком Важность, измените информацию в поле Дополнительно и нажмите Провести и Закрыть. В мобильном приложении тут же (поскольку оно активно), без значка в списке уведомлений планшета и звукового сигнала, появится следующее уведомление (рис. 2.38).

Затем запустите на планшете какое-то другое приложение (чтобы мобильный клиент работал в фоновом режиме). Откройте в «настольном» приложении тот же заказ, измените информацию в поле Адрес доставки и в поле Дополнительно и проведите его.

На планшете прозвучит стандартный сигнал, установленный в системе для уведомлений, и в списке уведомлений планшета появится текст сообщения, который всплывет на несколько секунд (рис. 2.39). При нажатии на уведомление приложение мобильного клиента будет активизировано (это справедливо для мобильных устройств, работающих под управлением ОС Android).

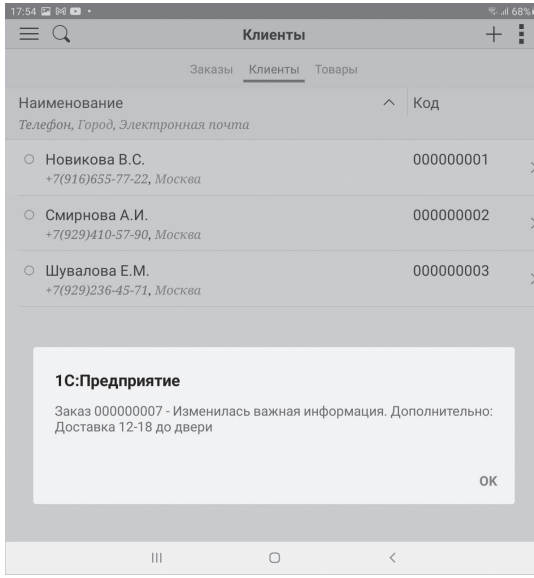


Рис. 2.38. Push-уведомление, полученное от «настольного» приложения для курьера «Морозов»

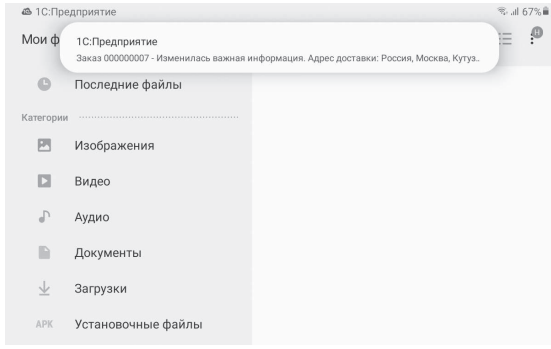


Рис. 2.39. Push-уведомление, полученное от «настольного» приложения для курьера «Морозов»

Выйдите из приложения. В результате вы окажетесь в списке приложений мобильного клиента. Откройте в «настольном» приложении тот же заказ, измените информацию в поле Адрес доставки и проведите его. На планшете

появится текст сообщения, и будет предложено перейти к соответствующему приложению (рис. 2.40).

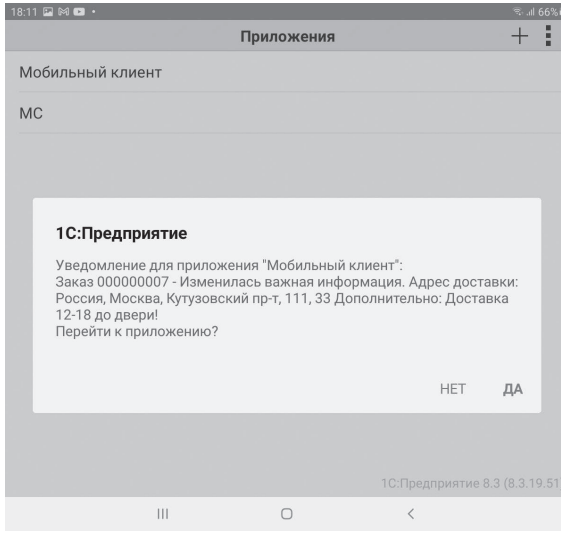


Рис. 2.40. Push-уведомление, полученное от «настольного» приложения для курьера «Морозов»

Если мобильное приложение не запущено, то текст уведомления будет показан только в списке уведомлений. При нажатии на него будет открыто мобильное приложение, но текст уведомления больше не появится.

Итак, вы убедились, что при изменении важной информации в заказе из офиса на планшет тому курьеру, который обслуживает заказ, приходят уведомления об этом. Все работает так, как вы и хотели!

Система взаимодействия

Общение пользователей, находящихся в офисе, и «удаленных» пользователей, работающих на планшете, можно организовать с помощью системы взаимодействия. Достаточно просто зарегистрировать приложение на сервере взаимодействия, и можно вообще ничего не программировать. Или же можно так же, как и в предыдущем примере, написать программный код – например, при изменении каких-то полей в заказе программно создать предметное обсуждение, добавить туда пользователя (курьера) и отправить ему сообщение.

ПОДРОБНЕЕ

О работе с системой взаимодействия подробно рассказывалось в книге Система взаимодействия. Коммуникация в бизнес-приложениях. Разработка в системе «1С:Предприятие 8.3».

Ниже будет показан самый простой вариант интерактивного взаимодействия, когда при изменении информации в заказе менеджер из офиса отправляет сообщение курьеру на планшет.

А также вы увидите, как пользователи офисного приложения и пользователи мобильного клиента на планшете могут продемонстрировать друг другу свой экран при видеосвязи через систему взаимодействия.

Итак, запустите приложение на стационарном компьютере от имени пользователя Администратор с полными правами. В системном меню вызовите Функции для технического специалиста и откройте стандартную обработку Управление системой взаимодействия. Введите адрес электронной почты абонента, наименование приложения, получите код регистрации с сервера «1С:Диалог» и зарегистрируйте приложение в системе взаимодействия (рис. 2.41).

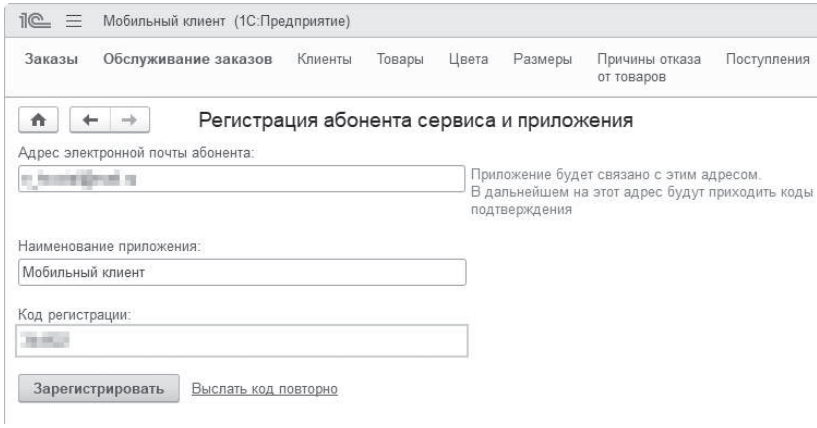


Рис. 2.41. Регистрация приложения в системе взаимодействия

Затем, нажав на ссылку Пользователи, зарегистрируйте остальных пользователей (Менеджер, Курьер) в системе взаимодействия (рис. 2.42). Или же это будет сделано автоматически при первом входе пользователя в систему.

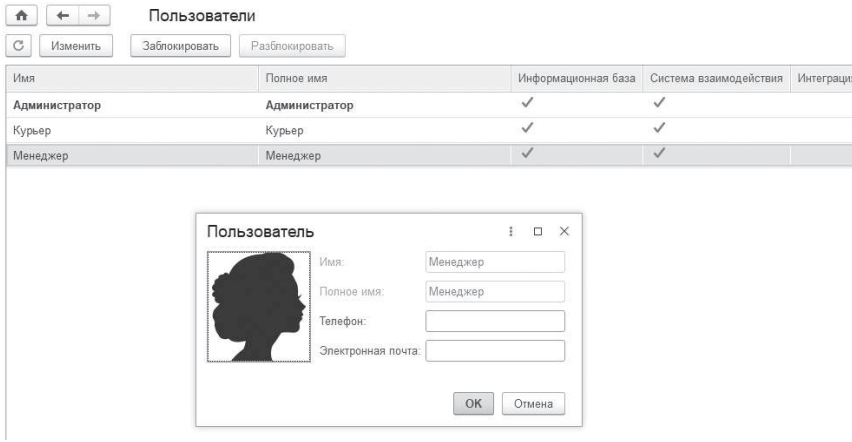


Рис. 2.42. Регистрация пользователей в системе взаимодействия

Предметное обсуждение

После регистрации пользователей в системе взаимодействия они могут общаться между собой, в том числе в предметных обсуждениях.

Например, менеджер офисного приложения изменил информацию в заказе и хочет сообщить об этом курьеру. Для этого он открывает в форме этого заказа панель предметного обсуждения (с помощью кнопки Обсуждение), указывает, что надо оповестить курьера, и отправляет ему сообщение (рис. 2.43).

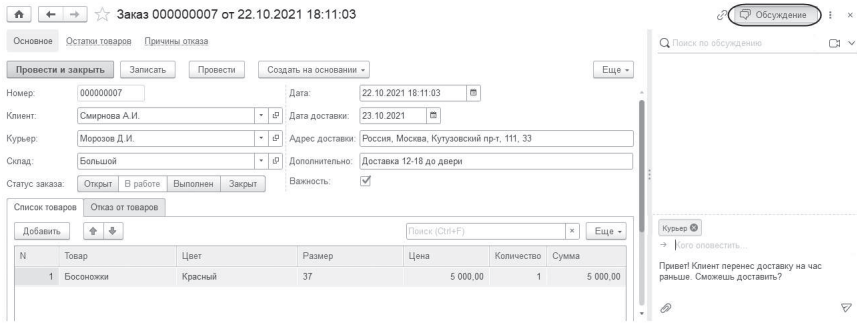


Рис. 2.43. Предметное обсуждение менеджера с курьером в форме заказа на стационарном компьютере

У курьера на планшете звучит звуковой сигнал и всплывает текст сообщения, нажав на который, он может открыть обсуждение по данному заказу и ответить менеджеру (рис. 2.44–2.46).

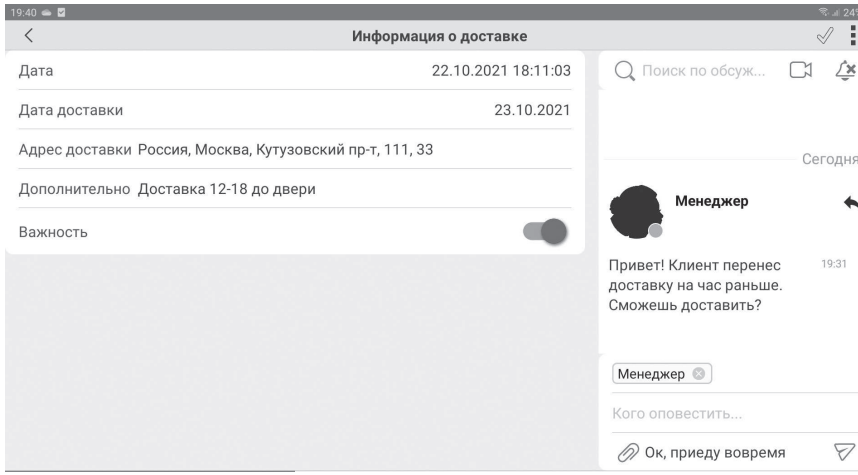


Рис. 2.44. Предметное обсуждение менеджера с курьером в форме заказа на планшете

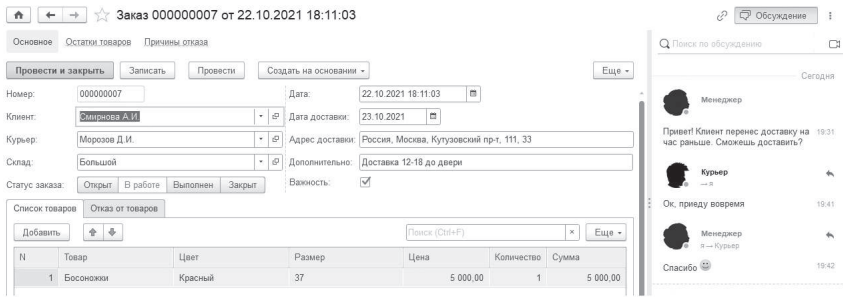


Рис. 2.45. Предметное обсуждение менеджера с курьером в форме заказа на стационарном компьютере

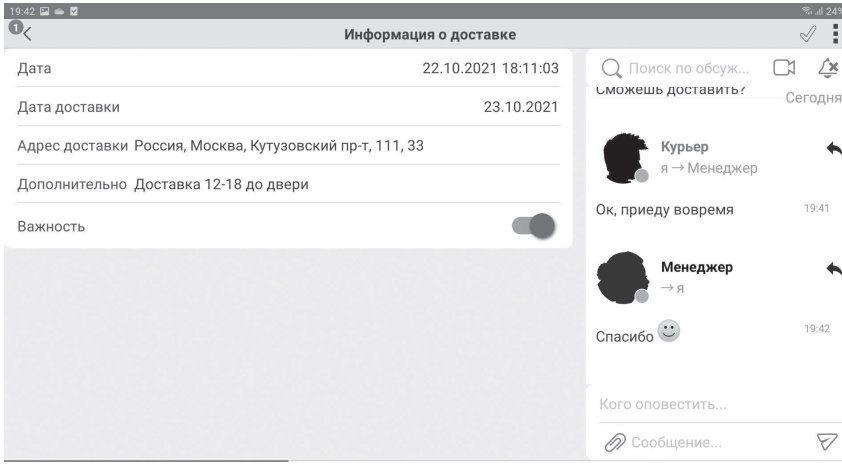


Рис. 2.46. Предметное обсуждение менеджера с курьером в форме заказа на планшете

Если курьер не может сразу открыть обсуждение (занят, нет на месте, в дороге и т. п.), то он может увидеть его, нажав на строку Оповещения в меню приложения (рис. 2.47).

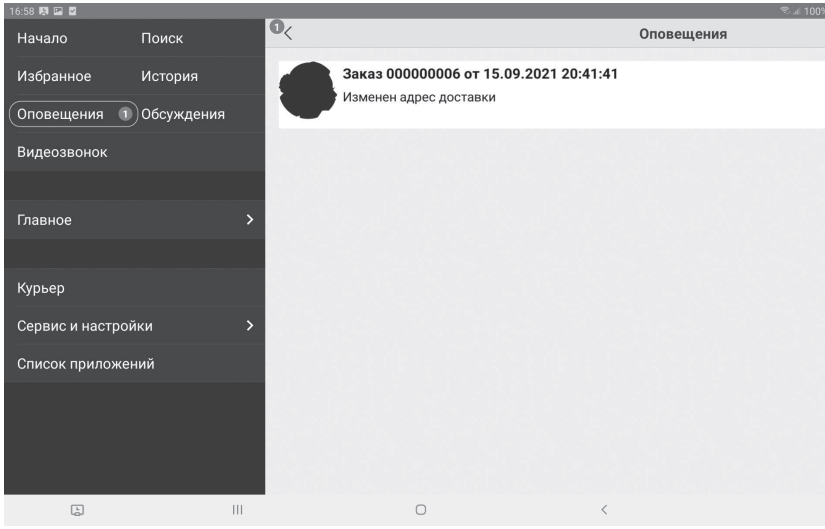


Рис. 2.47. Список оповещений курьера в мобильном клиенте

Демонстрация экрана

При общении по видеосвязи менеджер в офисе и курьер на планшете могут продемонстрировать друг другу свой экран, чтобы сразу на месте решить спорные вопросы. Для этого в меню видеозвонка нужно выбрать команду Начать показ экрана (рис. 2.48, 2.50).

Например, курьер может сделать видеозвонок менеджеру и показать ему картинку товара, которая, по его мнению, не соответствует товару (рис. 2.48, 2.49), или фотографию товара с браком и т. п.

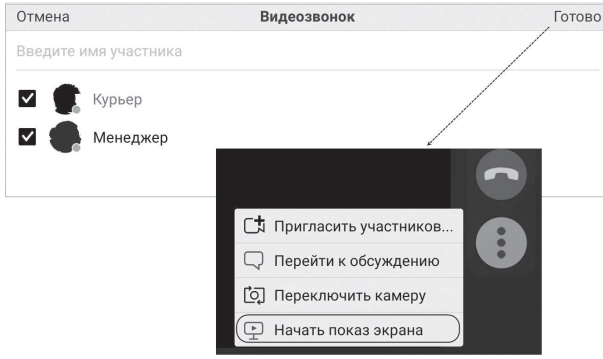


Рис. 2.48. Включение демонстрации экрана курьером

Менеджер в офисе также может показать курьеру какие-то важные данные – например, отчет, сформированный в офисе, на который у курьера нет прав (рис. 2.50, 2.51).

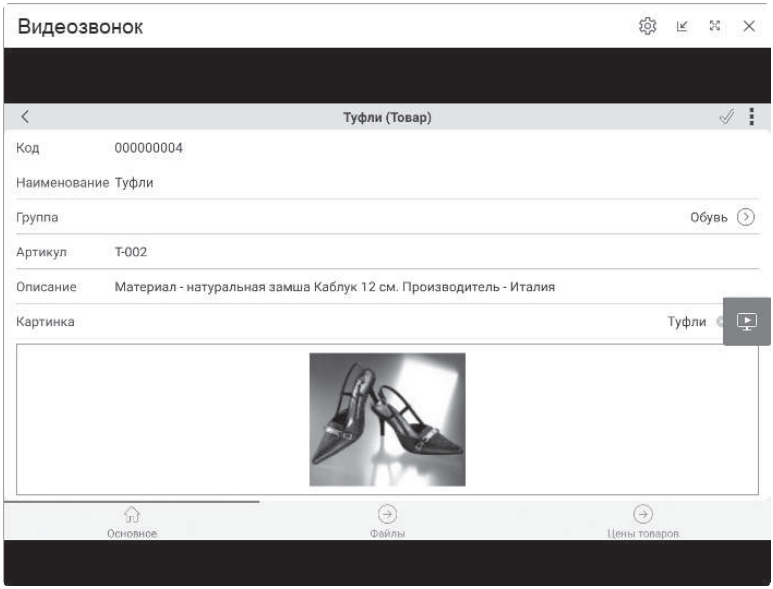


Рис. 2.49. Экран курьера, который видит менеджер

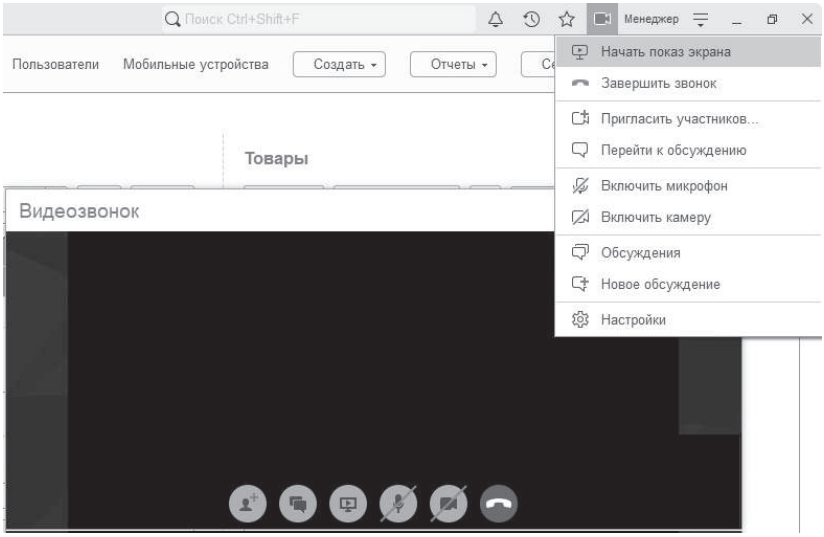


Рис. 2.50. Включение демонстрации экрана менеджером

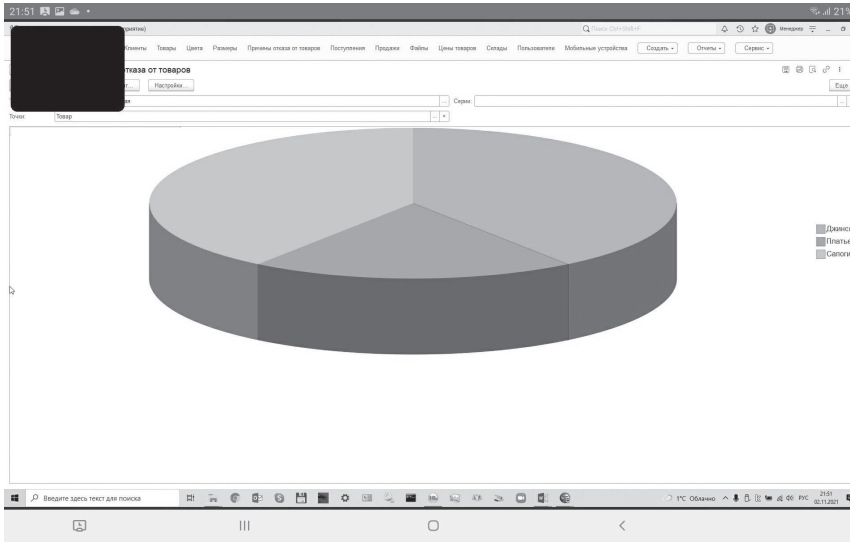


Рис. 2.51. Экран менеджера, который видит курьер

Глава 3. Приложение мобильного клиента с автономным режимом

В этой главе вы разработаете приложение мобильного клиента с автономным режимом. Это приложение совмещает в себе две технологии: работа в режиме «обычного» мобильного клиента (см. предыдущую главу книги «Приложение мобильного клиента») и работа в автономном режиме, когда между мобильным устройством и офисной базой нет связи (см. следующую главу книги «Приложение мобильной платформы»).

Прежде чем приступать к разработке мобильного клиента с автономным режимом, нужно разобраться с основными понятиями, которые будут использоваться далее в этой главе:

- Под *основным* сервером понимается кластер серверов прикладного решения на стационарном компьютере, в данном случае в офисе.
- *Основная* конфигурация – конфигурация, исполняющаяся на основном сервере.
- Под *автономным сервером* понимается серверная часть приложения мобильного клиента с автономным режимом, которая работает на мобильном устройстве.
- *Обычный режим* – это режим работы с подключением к основной информационной базе через Интернет. В этом режиме мобильное приложение может использовать как основной, так и автономный сервер.

- *Автономный режим* – это режим работы без подключения к основной информационной базе, при котором мобильное приложение может использовать только автономный сервер.
- *Автономная конфигурация* – часть основной конфигурации, которая предназначена для работы на мобильном устройстве в автономном режиме.

Особенностью мобильного клиента с автономным режимом является то, что помимо обычного мобильного клиента он содержит автономный сервер с файловой базой данных. Поэтому в обычном режиме работы он может использовать сразу две серверные части прикладного решения: автономный сервер и основной сервер.

При работе в автономном режиме используются только те данные (и та конфигурация), которые определены во время настройки состава автономной конфигурации. Во время этой настройки указывается не только список объектов и форм, доступных в автономном режиме, но и то, какие данные (реквизиты и табличные части) будут использоваться в автономных объектах.

Для начального заполнения базы данных мобильного клиента с автономным режимом нужно использовать стандартные механизмы обмена «1С:Предприятия» (см. раздел «Обмен данными»).

ПОДРОБНЕЕ

Документация 1С:Предприятие 8.3.20. Руководство разработчика. Глава 29. Разработка для мобильных устройств > Мобильный клиент с автономным режимом.

Начало разработки

Мобильный клиент с автономным режимом, так же как и «простой» мобильный клиент, подключается к информационной базе и предоставляет ее функциональность в мобильном интерфейсе на устройстве. Поэтому к нему применимы все те доработки основной конфигурации, которые вы выполняли в предыдущей главе.

Отличие мобильного клиента с автономным режимом – это автономная конфигурация и файловая база данных, в которой хранятся данные этой конфигурации.

В данной главе вы узнаете, какие усилия нужно предпринять, чтобы приложение мобильного клиента с автономным режимом работало офлайн с автономной информационной базой и затем данные, введенные на мобильном

устройстве, синхронизировались бы с основным приложением. А также увидите, как может изменяться функциональность приложения при работе в автономном режиме.

Итак, загрузите сделанную ранее копию офисной конфигурации и измените имя конфигурации на МобильныйКлиентАвтономныйРежим. Затем отметьте у свойства Назначение использования оба значения: Приложение для платформы и Приложение для мобильной платформы. А также свойство Режим совместимости установите в значение 8.3.19.

Это значит, что конфигурация будет работать и на стационарном компьютере, и на планшете. Режим совместимости нужен потому, что версия платформы «1С:Предприятия» – 8.3.20, а версия мобильной платформы – 8.3.19.

Теперь вам нужно отметить автономную конфигурацию – часть объектов, которые будут доступны на мобильном устройстве всегда, даже в случае потери соединения с офисной базой. В этом случае приложение будет продолжать работать в автономном режиме с теми объектами и той функциональностью, которые будут входить в состав автономной конфигурации.

Основной задачей курьера как пользователя мобильного приложения является, в первую очередь, обслуживание заказов клиентов. Исходя из этого добавьте в состав автономной конфигурации документ Заказ и те объекты, которые необходимы для работы с заказами: все справочники, перечисление СтатусыЗаказов и регистр сведений ЦеныТоваров.

Для этого в контекстном меню конфигурации нажмите Открыть состав автономной конфигурации и выберите нужные объекты в верхней части окна (рис. 3.1).

Состав автономной конфигурации можно задавать вплоть до реквизитов объектов. Например, в каком-то справочнике в определенном реквизите может находиться большой объем данных, который нецелесообразно скачивать на мобильное устройство. Но в данном примере эта возможность рассматриваться не будет.

Обновите конфигурацию базы данных (F7).

Затем опубликуйте информационную базу на веб-сервере с помощью команды главного меню Администрирование > Публикация на веб-сервере…, как это было показано в главе про мобильный клиент «Начало разработки».

В появившемся диалоге в поле Имя нужно задать имя виртуального каталога на веб-сервере, в который будет выполнена публикация информационной базы. В поле Каталог нужно указать физический каталог компьютера, в котором будет находиться файл публикации информационной базы.

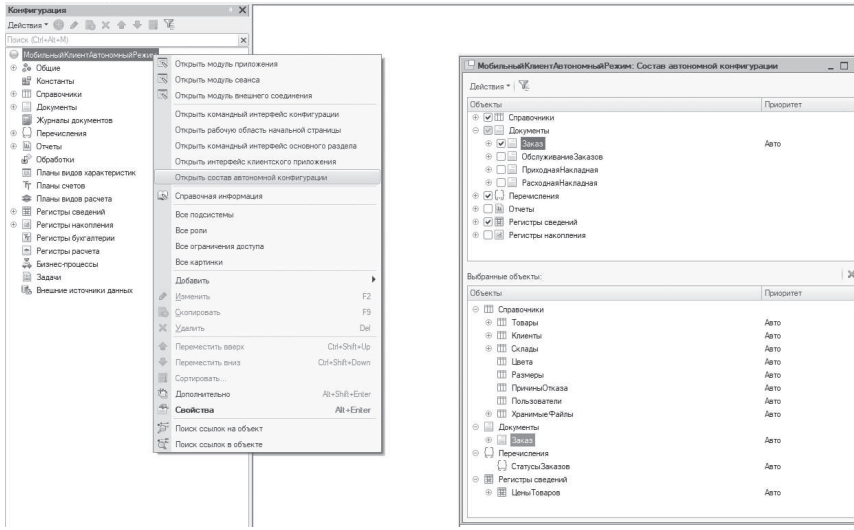


Рис. 3.1. Состав автономной конфигурации

Добавление приложения на планшет

Для тестирования и отладки приложения мобильного клиента автономным режимом на планшете нужно, чтобы на нем была установлена мобильная платформа для разработки соответствующего типа приложений. О том, как это сделать, рассказывалось в первой главе в разделе «Установка мобильной платформы разработчика для мобильных устройств».

Найдите в списке приложений планшета мобильную платформу для разработки мобильного клиента с автономным режимом и запустите ее. Платформа откроет список своих приложений, который пока пуст.

Добавьте новое мобильное приложение, нажав на значок «+» в правом верхнем углу экрана в строке Приложения. В появившемся окне задайте наименование приложения, в поле Веб-сервер укажите URL веб-сервера, на котором опубликована информационная база, и нажмите Готово в правом верхнем углу экрана, как это было показано в главе про мобильный клиент «Добавление приложения на планшет».

После этого в списке мобильных приложений платформы разработчика появится созданное вами приложение, которое можно запустить кратковременным нажатием на его наименование. При длинном нажатии на эту

строку появится контекстное меню, из которого можно изменить свойства мобильного приложения, выбрав пункт Изменить.

Запустите мобильное приложение на планшете в условиях, когда на планшете есть интернет-соединение. После небольшого ожидания, пока автономная конфигурация загрузится на планшет, на начальной странице мобильного приложения появятся команды для доступа ко всем объектам конфигурации.

То есть при наличии соединения вы сможете работать точно так же, как и в приложении «обычного» мобильного клиента, с полным доступом к информационной базе и функциональности офисного приложения (рис. 3.2).

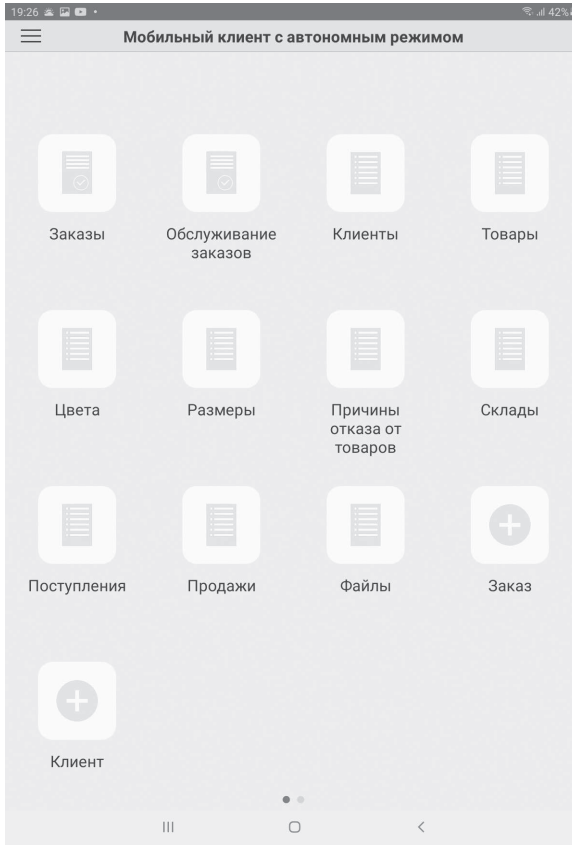


Рис. 3.2. Начальная страница мобильного приложения при наличии соединения

Теперь сымитируйте ситуацию, когда интернет-соединение отсутствует. Например, выйдите из приложения, включите на планшете режим полета и зайдите снова. Как вы видите, теперь часть объектов, которые вы не добавили в состав автономной конфигурации, стали недоступны. Это документы Обслуживание заказов, Поступления (приходные накладные), Продажи (расходные накладные) и т. д. (рис. 3.3).

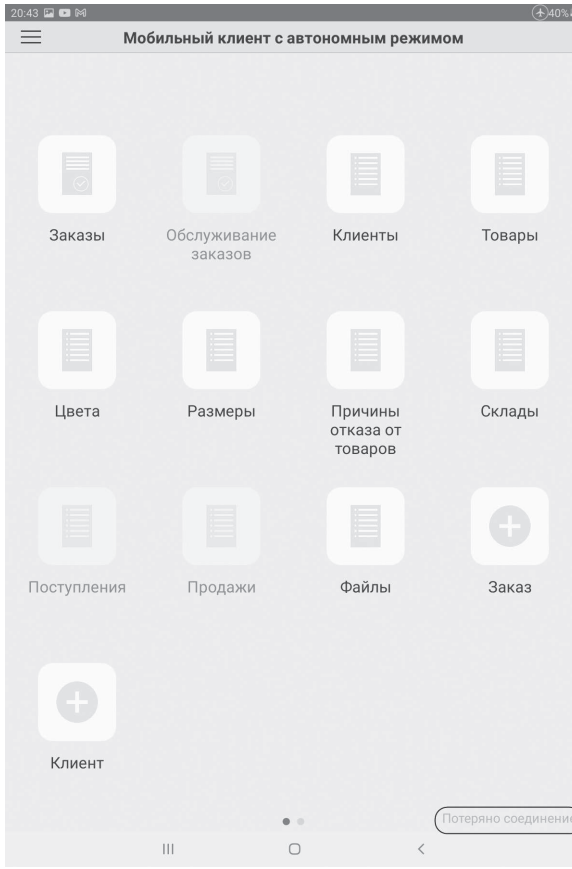


Рис. 3.3. Начальная страница мобильного приложения при потере соединения

Объекты, входящие в состав автономной конфигурации, доступны, но в них пока нет никаких данных. Так произошло потому, что в случае отсутствия соединения мобильное приложение переходит в автономный режим работы,

при котором используется не основная, а автономная информационная база. Эта база пока пуста.

Чтобы заполнить автономную базу данными, нужно реализовать в конфигурации блок обмена данными, с помощью которого произойдет первоначальный перенос данных на мобильное устройство и затем будет выполняться периодическая синхронизация между автономной и основной базами данных.

Обмен данными

Чтобы синхронизировать данные между основной информационной базой и автономной базой на мобильном устройстве, вы будете использовать стандартный механизм обмена данными системы «1С:Предприятие». Основная и автономная части прикладного решения при этом рассматриваются как разные узлы плана обмена, который служит для регистрации изменений в данных, а данные передаются с помощью стандартной инфраструктуры сообщений.

При разработке обмена данными вы должны обеспечить, чтобы все объекты, которые отмечены в составе автономной конфигурации, были включены в состав плана обмена. Также для каждой информационной базы автономного мобильного приложения в основной информационной базе должен существовать соответствующий элемент плана обмена.

С точки зрения обмена данными общую схему работы можно представить следующим образом:

- Первый запуск мобильного клиента с автономным режимом всегда осуществляется при наличии соединения с основным сервером.
- При первом запуске мобильного приложения с основного сервера на мобильное устройство загружается автономная конфигурация и выполняется инициализация автономной информационной базы.
- В процессе первого запуска мобильный клиент с автономным режимом должен обратиться к основному серверу и выполнить следующие действия:
  - В основной информационной базе создать «свой» узел обмена.
  - В автономной информационной базе создать узел обмена, относящийся к основной информационной базе.
  - Выполнить первоначальное заполнение автономной информационной базы данными из основной информационной базы.
- Периодически нужно синхронизировать данные между автономной и основной информационными базами.

Реализация обмена данными

Чтобы решить поставленные задачи, вам понадобятся: план обмена для хранения основного и мобильного узлов обмена (а также для регистрации изменений данных для этих узлов), константа, в которой будет храниться порядковый номер нового узла обмена, и справочник пользователей информационной базы для идентификации текущего пользователя мобильного приложения. Итак, давайте начнем.

Добавьте в конфигурацию план обмена Мобильные. В состав плана обмена включите те объекты, которые вы ранее включили в состав автономной конфигурации (рис. 3.4).

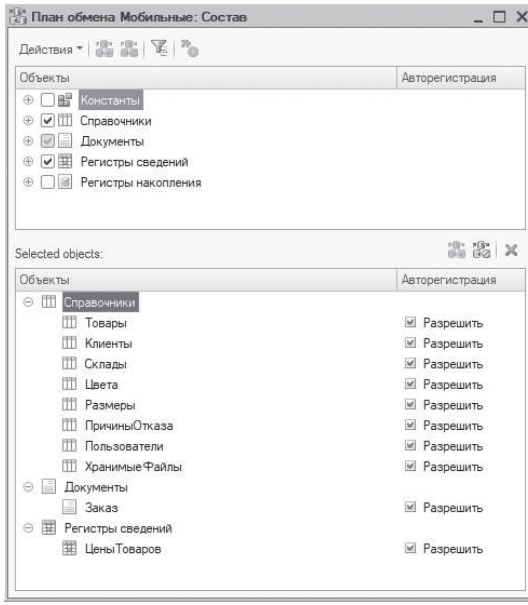


Рис. 3.4. Состав плана обмена «Мобильные»

В конфигурации МобильныйКлиентАвтономныйРежим уже существует справочник Пользователи. При тестировании обмена данными вам нужно будет только немного изменить его, чтобы коды пользователей соответствовали пользователям информационной базы (см. рис. 3.7).

Затем добавьте в конфигурацию константу КодНовогоУзлаПланаОбмена со свойствами: Тип – Число, Длина – 10, Неотрицательное – Да, Использовать стандартные команды – Нет.

Теперь вам нужно сделать так, чтобы при начале работы мобильного приложения автоматически запускался бы процесс обмена данными между автономной базой на мобильным устройстве и офисной базой, реализующий описанную выше функциональность (инициализацию узлов обмена, первоначальное заполнение данными автономной базы и последующую синхронизацию).

Для этого откройте модуль приложения, создайте в нем процедуру ПриНачалеРаботыСистемы() и заполните ее следующим образом (листинг 3.1).

Листинг 3.1. Процедура «ПриНачалеРаботыСистемы()»

```
Процедура ПриНачалеРаботыСистемы()

#Если МобильныйКлиент Тогда
        Если ОсновнойСерверДоступен() = Истина Тогда
                ОбменМобильныеАвтономныйКлиент.НачатьОбмен();
        КонецЕсли;
#КонецЕсли

КонецПроцедуры
```

Эта процедура всегда вызывается на клиенте при старте как «настольного», так и мобильного приложения. Поскольку обмен должен начинаться на мобильном клиенте с автономным режимом, поместите вызов процедуры, инициирующей обмен, в ветку условия #Если МобильныйКлиент.

Сначала вы проверяете доступность основного сервера с мобильного клиента с помощью метода глобального контекста ОсновнойСерверДоступен(). Если соединение с основным сервером есть, то вы вызываете процедуру НачатьОбмен() общего модуля ОбменМобильныеАвтономныйКлиент, о которой будет рассказано чуть ниже (см. листинг 3.3).

А также вам нужно добавить в модуль приложения процедуру ПриИзмененииДоступностиОсновногоСервера() с точно таким же кодом, чтобы обмен данными начинался также и при восстановлении соединения между мобильным приложением и основным сервером (листинг 3.2).

Листинг 3.2. Процедура «ПриИзмененииДоступностиОсновногоСервера()»

```
Процедура ПриИзмененииДоступностиОсновногоСервера(НачалоСеансаОсновногоСервера)

#Если МобильныйКлиент Тогда
        Если ОсновнойСерверДоступен() = Истина Тогда
                ОбменМобильныеАвтономныйКлиент.НачатьОбмен();
        КонецЕсли;
#КонецЕсли

КонецПроцедуры
```

Затем создайте общий клиентский модуль ОбменМобильныеАвтономныйКлиент и поместите в него процедуру НачатьОбмен(), листинг 3.3.

Листинг 3.3. Процедура «НачатьОбмен()»

```
Процедура НачатьОбмен() Экспорт

#Если МобильныйКлиент Тогда
    Если ОсновнойСерверДоступен() = Истина Тогда
        ОбменМобильныеАвтономныйСервер.ВыполнитьОбменДанными();
        Оповестить("ОбменЗакончен");
    Иначе
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Обмен невозможен. Недоступен основной сервер.";
        Сообщение.Сообщить();
    КонецЕсли;
#КонецЕсли

КонецПроцедуры
```

Эта процедура исполняется только на мобильном клиенте (#Если МобильныйКлиент). В случае если основной сервер доступен, вы вызываете процедуру ВыполнитьОбменДанными() общего модуля ОбменМобильныеАвтономныйСервер, в которой и происходит, собственно, обмен данными между основной и автономной базами.

После окончания обмена данными вы оповещаете об этом все созданные формы с помощью метода Оповестить(). Если соединения с основным сервером нет, выводите сообщение о невозможности обмена.

Создайте общий модуль ОбменМобильныеАвтономныйСервер с установленными флажками Сервер и Вызов сервера и поместите в него процедуру ВыполнитьОбменДанными(), листинг 3.4.

Листинг 3.4. Процедура «ВыполнитьОбменДанными()»

```
#Если МобильныйАвтономныйСервер Тогда
Процедура ВыполнитьОбменДанными() Экспорт

    НаименованиеУзла = ПользователиИнформационнойБазы.ТекущийПользователь()
                                                        .Имя + " - Автономный узел";

    ЦентральныйУзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду("001");
    Если ЦентральныйУзелОбмена.Пустая() Тогда
        НовыйУзел = ПланыОбмена.Мобильные.СоздатьУзел();
        НовыйУзел.Код="001";
        НовыйУзел.Наименование="Центральный";
        НовыйУзел.Записать();
        ЦентральныйУзелОбмена = НовыйУзел.Ссылка;
    КонецЕсли;

    Узел = ПланыОбмена.Мобильные.ЭтотУзел();
```

```
    // Выполнить инициализацию обмена, проверить, есть ли нужный автономный узел в плане обмена.
    НовыйКод = ОсновнойСервер. ОбменМобильныеВызовПК.НачатьОбмен(
                                Узел.Код, НаименованиеУзла, ЦентральныйУзелОбмена.НомерПринятого,
                                ЦентральныйУзелОбмена.НомерОтправленного);

    Если  Узел.Код <> НовыйКод Тогда
        ОбъектУзла = Узел.ПолучитьОбъект();
        ОбъектУзла.Код = НовыйКод;
        ОбъектУзла.Наименование = НаименованиеУзла;
        ОбъектУзла.Записать();
    КонецЕсли;

    ДанныеОбмена = ОбменМобильныеСервер.СформироватьПакетОбмена(ЦентральныйУзелОбмена);
    ДанныеОбмена = ОсновнойСервер.ОбменМобильныеВызовПК.ВыполнитьОбменДанными(
                                Узел.Код, ДанныеОбмена);
    ОбменМобильныеСервер.ПринятьПакетОбмена(ЦентральныйУзелОбмена, ДанныеОбмена);

КонецПроцедуры
#КонецЕсли
```

Обратите внимание, что вся процедура обрамлена инструкцией #Если МобильныйАвтономныйСервер … #КонецЕсли. Это значит, что процедура будет скомпилирована и будет исполняться только на мобильном автономном сервере.

Сначала в этой процедуре вы задаете наименование узла обмена на мобильном устройстве (далее – автономного узла) на основе имени текущего пользователя мобильного приложения.

Затем проверяете, есть ли в автономной базе в списке узлов плана обмена Мобильные узел, соответствующий основной базе, с кодом «001». Если нет (обмен еще ни разу не выполнялся), создаете его.

После этого с помощью функции ЭтотУзел() получаете ссылку на предопределенный узел обмена на мобильном устройстве (автономный узел) и передаете код этого узла в функцию НачатьОбмен() общего модуля основного сервера ОбменМобильныеВызовПК (см. листинг 3.5). для инициализации обмена в основной базе.

Здесь нужно сделать отступление. Дело в том, что в процессе работы мобильный клиент с автономным режимом может использовать одновременно и автономный, и основной сервер (если с ним есть связь). Если код мобильного приложения исполняется на автономном сервере, то из него можно вызвать основной сервер, но не наоборот. То есть вызов возможен только в одну сторону (автономный сервер à основной сервер).

Таким образом, при нахождении на автономном сервере можно перенести исполнение на основной сервер, вызвав его метод через точку от свойства глобального контекста ОсновнойСервер. С помощью данного свойства

на автономном сервере доступны серверные общие модули основного сервера.

В данном случае из кода, который выполняется на стороне автономного сервера (в процедуре ВыполнитьОбменДанными() общего модуля ОбменМобильныеАвтономныйСервер), вам нужно инициализировать обмен в основной информационной базе на стороне основного сервера. А именно – проверить, есть ли нужный автономный узел в плане обмена в основной базе, и если нет, то создать его и зарегистрировать для него изменения в данных и т. п.

Чтобы вызвать функцию НачатьОбмен() общего модуля основного сервера ОбменМобильныеВызовПК со стороны автономного сервера, используется конструкция: ОсновнойСервер.ОбменМобильныеВызовПК.НачатьОбмен(). Сама функция будет подробно рассмотрена ниже (см. листинг 3.5).

После этого исполнение возвращается на автономный сервер, и вы вызываете функцию СформироватьПакетОбмена() для формирования в мобильном приложении пакета изменений, предназначенных для центрального узла обмена (предопределенного узла обмена основной базы). Эту функцию вы поместите в общий модуль ОбменМобильныеСервер. Сама функция будет подробно рассмотрена ниже (см. листинг 3.8).

Затем вы передаете данные обмена, помещенные в хранилище значения и возвращенные этой функцией, и код автономного узла обмена в функцию ВыполнитьОбменДанными() общего модуля основного сервера ОбменМобильныеВызовПК. При этом исполнение снова переходит на сторону основного сервера.

В этой функции сначала изменения, принятые от автономного узла обмена, записываются в центральную базу данных, а затем в этой базе формируется пакет изменений, предназначенных для автономного узла обмена. Сама функция будет подробно рассмотрена ниже (см. листинг 3.7).

В результате данные обмена, помещенные в хранилище значения, возвращаются в мобильное приложение, и исполнение возвращается на автономный сервер.

После этого вы вызываете процедуру ПринятьПакетОбмена() для записи изменений, принятых от центрального узла обмена, в базу мобильного приложения (автономную базу). Эту процедуру вы поместите в общий модуль ОбменМобильныеСервер. Сама процедура будет подробно рассмотрена ниже (см. листинг 3.9).

Теперь пришло время создать и подробно рассмотреть все описанные выше процедуры и функции.

Создайте общий модуль ОбменМобильныеВызовПК с установленными флажками Сервер и Вызов сервера и поместите в него функцию НачатьОбмен(). В эту функцию передаются код и наименование автономного узла обмена и номера сообщений, принятых/отправленных центральным узлом обмена (листинг 3.5).

Листинг 3.5. Функция «НачатьОбмен()»

```
Функция НачатьОбмен(КодУзла, НаименованиеМобильногоКомпьютера, НомерОтправленного
                                                              , НомерПринятого) Экспорт

    УстановитьПривилегированныйРежим(Истина);

    УзелОбмена = ПланыОбмена.Мобильные.ЭтотУзел().ПолучитьОбъект();
    Если НЕ ЗначениеЗаполнено(УзелОбмена.Код) Тогда
        УзелОбмена.Код="001";
        УзелОбмена.Наименование="Центральный";
        УзелОбмена.Записать();
    КонецЕсли;

    УзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду(КодУзла);
    Если УзелОбмена.Пустая() Тогда

        НовыйУзел = ПланыОбмена.Мобильные.СоздатьУзел();

        НачатьТранзакцию();

        Блокировка = Новый БлокировкаДанных;
        ЭлементБлокировки = Блокировка.Добавить("Константа.КодНовогоУзлаПланаОбмена");
        ЭлементБлокировки.Режим = РежимБлокировкиДанных.Исключительный;
        Блокировка.Заблокировать();

        КодНовогоУзла = Константы.КодНовогоУзлаПланаОбмена.Получить();
        Если КодНовогоУзла = 0 Тогда
            КодНовогоУзла = 2;
        КонецЕсли;
        Константы.КодНовогоУзлаПланаОбмена.Установить(КодНовогоУзла + 1);

        ЗафиксироватьТранзакцию();

        Если СтрДлина(КодНовогоУзла) < 3 Тогда
            НовыйУзел.Код = Формат(КодНовогоУзла, "ЧЦ=3; ЧВН=");
        Иначе
            НовыйУзел.Код = КодНовогоУзла;
        КонецЕсли;

        НовыйУзел.Наименование = НаименованиеМобильногоКомпьютера;
        НовыйУзел.НомерОтправленного = НомерОтправленного;
        НовыйУзел.НомерПринятого = НомерПринятого;
        НовыйУзел.Записать();
        ОбменМобильныеСервер.ЗарегистрироватьИзмененияДанных(НовыйУзел.Ссылка);
        УзелОбмена = НовыйУзел.Ссылка;
```

```
    Иначе

        Если УзелОбмена.ПометкаУдаления ИЛИ УзелОбмена.Наименование <>
                НаименованиеМобильногоКомпьютера Тогда
            Узел = УзелОбмена.ПолучитьОбъект();
            Узел.ПометкаУдаления = Ложь;
            Узел.Наименование = НаименованиеМобильногоКомпьютера;
            Узел.Записать();
        КонецЕсли;

        Если УзелОбмена.НомерОтправленного <>
                НомерОтправленного ИЛИ УзелОбмена.НомерПринятого <> НомерПринятого Тогда
            Узел = УзелОбмена.ПолучитьОбъект();
            Узел.НомерОтправленного = НомерОтправленного;
            Узел.НомерПринятого = НомерПринятого;
            Узел.Записать();
            ОбменМобильныеСервер.ЗарегистрироватьИзмененияДанных(УзелОбмена);
        КонецЕсли;

    КонецЕсли;

    Возврат УзелОбмена.Код;

КонецФункции
```

В отличие от одноименной функции, выполняющейся на мобильном клиенте в модуле ОбменМобильныеАвтономныйКлиент (см. листинг 3.3), эта функция выполняется на стороне основного сервера и манипулирует данными основной базы.

Сначала вы проверяете, заполнен ли код предопределенного узла обмена в плане обмена Мобильные в основной базе, если нет – заполняете код и наименование значениями «001» и «Центральный» (это стандартные значения для этого узла).

Затем проверяете, есть ли автономный узел с кодом, который передан в параметре КодУзла, в списке узлов плана обмена. Если нет – создаете новый узел, блокируете константу КодНовогоУзлаПланаОбмена, присваиваете значение этой константы новому узлу обмена и затем увеличиваете это значение на единицу.

После этого инициализируете этот узел наименованием и номерами сообщений, принятых/отправленных центральным узлом обмена, и с помощью процедуры ЗарегистрироватьИзмененияДанных() регистрируете изменения центральной базы для нового узла обмена. Эту процедуру вы поместите в общий модуль ОбменМобильныеСервер. Сама процедура будет подробно рассмотрена ниже (см. листинг 3.6).

Если автономный узел обмена, код которого передан в функцию, уже существует, вы также синхронизируете номера принятых/отправленных сообщений в случае их несоответствия в центральном и автономном узлах обмена и регистрируете изменения центральной базы для автономного узла.

В результате функция возвращает код нового или измененного автономного узла в процедуру ВыполнитьОбменДанными() общего модуля ОбменМобильныеАвтономныйСервер.

Теперь создайте общий модуль ОбменМобильныеСервер с установленными флажками Сервер и Вызов сервера и поместите в него процедуру ЗарегистрироватьИзмененияДанных(), листинг 3.6.

Листинг 3.6. Процедура «ЗарегистрироватьИзмененияДанных()»

```
Процедура ЗарегистрироватьИзмененияДанных(УзелОбмена) Экспорт

    СоставПланаОбмена = УзелОбмена.Метаданные().Состав;
    Для Каждого ЭлементСоставаПланаОбмена Из СоставПланаОбмена Цикл
        ПланыОбмена.ЗарегистрироватьИзменения(
                                    УзелОбмена, ЭлементСоставаПланаОбмена.Метаданные);
    КонецЦикла;

КонецПроцедуры
```

В данной процедуре используется метод менеджера планов обмена ЗарегистрироватьИзменения(), с помощью которого в цикле для каждого объекта из состава плана обмена регистрируются изменения данных для узла обмена, переданного в процедуру.

Затем поместите в модуль ОбменМобильныеВызовПК функцию ВыполнитьОбменДанными(). Эта функция снова переносит исполнение на основной сервер, так как вызывается через точку от свойства глобального контекста ОсновнойСервер (листинг 3.7).

Листинг 3.7. Функция «ВыполнитьОбменДанными()»

```
Функция ВыполнитьОбменДанными(КодУзла, ДанныеМобильногоПриложения) Экспорт

    УзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду(КодУзла);

    Если УзелОбмена.Пустая() Тогда
        ВызватьИсключение("Неизвестное устройство - " + КодУзла);
    КонецЕсли;

    ОбменМобильныеСервер.ПринятьПакетОбмена(УзелОбмена, ДанныеМобильногоПриложения);

    Возврат ОбменМобильныеСервер.СформироватьПакетОбмена(УзелОбмена);

КонецФункции
```

В этой функции по коду, переданному в параметре КодУзла, в плане обмена находится автономный узел. И затем данные обмена, полученные от этого узла в параметре ДанныеМобильногоПриложения, записываются в центральную базу данных с помощью процедуры ПринятьПакетОбмена(). Затем в этой базе формируется пакет изменений, предназначенных для автономного узла обмена, с помощью функции СформироватьПакетОбмена(). Эта функция возвращает данные обмена, помещенные в хранилище значения, в мобильное приложение.

Теперь поместите в модуль ОбменМобильныеСервер функцию СформироватьПакетОбмена(), листинг 3.8.

Листинг 3.8. Функция «СформироватьПакетОбмена()»

```
Функция СформироватьПакетОбмена(УзелОбмена) Экспорт

    УстановитьПривилегированныйРежим(Истина);
    ЗаписьXML = Новый ЗаписьXML;

    ЗаписьXML.УстановитьСтроку("UTF-8");
    ЗаписьXML.ЗаписатьОбъявлениеXML();

    ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
    ЗаписьСообщения.НачатьЗапись(ЗаписьXML, УзелОбмена);

    ЗаписьXML.ЗаписатьСоответствиеПространстваИмен(
                        "xsi", "http://www.w3.org/2001/XMLSchema-instance");
    ЗаписьXML.ЗаписатьСоответствиеПространстваИмен("v8",  "http://v8.1c.ru/data");

    ВыборкаИзменений = ПланыОбмена.ВыбратьИзменения(
                        УзелОбмена, ЗаписьСообщения.НомерСообщения);
    Пока ВыборкаИзменений.Следующий() Цикл

        Данные = ВыборкаИзменений.Получить();

        // Если перенос данных не нужен, то записать удаление данных
        Если Не НуженПереносДанных(Данные, УзелОбмена) Тогда
            УдалениеДанных(Данные);
        КонецЕсли;

        ЗаписатьXML(ЗаписьXML, Данные);

    КонецЦикла;

    ЗаписьСообщения.ЗакончитьЗапись();

    Возврат Новый ХранилищеЗначения(ЗаписьXML.Закрыть(), Новый СжатиеДанных(9));

КонецФункции
```

В этой функции формируется пакет данных обмена для узла, переданного в параметре УзелОбмена. По завершении работы функция возвращает этому узлу данные обмена, помещенные в хранилище значения.

Обратите внимание, что если функция исполняется на стороне автономного сервера, то пакет изменений формируется для центрального узла обмена. И наоборот: если функция исполняется на стороне основного сервера, то данные обмена формируются для автономного узла.

Это стандартная функция, формирующая пакет изменений для узла обмена, поэтому если она вам уже хорошо знакома, то дальнейшее объяснение можете пропустить.

Сначала создается новый объект – ЗаписьXML. Затем с помощью метода этого объекта УстановитьСтроку() задается, что запись будет идти в XML-строку, и в начало этой строки записывается объявление XML.

Затем создается новый объект ЗаписьСообщенияОбмена, метод которого НачатьЗапись() позволяет, кроме всего прочего, создать очередной номер сообщения и записать заголовок сообщения в XML.

Для сокращения размера XML-строки определяется соответствие используемым пространствам имен.

После этого получается выборка из записей регистрации изменений, предназначенных узлу-получателю (переданному в параметре УзелОбмена). При формировании выборки методом менеджера планов обмена ВыбратьИзменения() вторым параметром в нее передается номер сообщения.

В цикле перебора измененных данных эти данные анализируются на необходимость выгрузки в узел обмена с помощью вызова функции НуженПереносДанных(), см. листинг 3.10. В случае если выгрузка не должна производиться (функция возвращает Ложь), вызывается процедура УдалениеДанных(), см. листинг 3.11.

Данные, которые прошли проверку, сериализуются в XML методом глобального контекста ЗаписатьXML().

После выхода из цикла запись заканчивается методом ЗакончитьЗапись() объекта ЗаписьСообщенияОбмена. XML-строка, полученная после закрытия объекта ЗаписьXML, помещается в хранилище значения, и пакет изменений возвращается функцией тому узлу обмена, для которого он был предназначен.

Теперь поместите в модуль ОбменМобильныеСервер процедуру ПринятьПакетОбмена(). В процедуру передаются узел обмена, от которого принимаются изменения, и пакет обмена (ДанныеОбмена), полученный от этого узла и помещенный в ХранилищеЗначения (листинг 3.9).

Листинг 3.9. Процедура «ПринятьПакетОбмена()»

```
Процедура ПринятьПакетОбмена(УзелОбмена, ДанныеОбмена) Экспорт

    УстановитьПривилегированныйРежим(Истина);
    ЧтениеXML = Новый ЧтениеXML;
    ЧтениеXML.УстановитьСтроку(ДанныеОбмена.Получить());
    ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
    ЧтениеСообщения.НачатьЧтение(ЧтениеXML);
    ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель
                                    , ЧтениеСообщения.НомерПринятого);

    НачатьТранзакцию();
    Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

        Данные = ПрочитатьXML(ЧтениеXML);

        Если НЕ Данные = Неопределено Тогда

            Если Не ПринятьИзменения(ЧтениеСообщения.Отправитель, Данные) Тогда
                Продолжить;
            КонецЕсли;

            Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
            Данные.ОбменДанными.Загрузка = Истина;

            Данные.Записать();
        КонецЕсли;

    КонецЦикла;
    ЗафиксироватьТранзакцию();

    ЧтениеСообщения.ЗакончитьЧтение();
    ЧтениеXML.Закрыть();

КонецПроцедуры
```

В зависимости от контекста исполнения эта процедура, так же как и предыдущая функция, принимает данные обмена или от автономного, или от центрального узла обмена. Это стандартная процедура для чтения и записи данных, принятых от узла обмена, поэтому если она вам уже хорошо знакома, то дальнейшее объяснение можете пропустить.

Сначала создается новый объект – ЧтениеXML. Затем с помощью метода этого объекта УстановитьСтроку() данные обмена получаются из хранилища значения.

Затем создается объект ЧтениеСообщенияОбмена. С помощью метода этого объекта НачатьЧтение() считывается заголовок XML-сообщения, в котором содержится, в том числе, информация об отправителе сообщения.

Перед тем как начать чтение данных, с помощью метода менеджера планов обмена УдалитьРегистрациюИзменений() удаляются все записи регистрации изменений, которые были сделаны для этого узла и соответствовали номерам сообщений, меньшим или равным принятому, указанному в обрабатываемом сообщении. Это делается для того, чтобы исключить дублирование данных, которые уже были ранее посланы этому узлу и им обработаны.

Затем в цикле выполняется чтение данных, содержащихся в сообщении обмена. При этом для каждого элемента в функции ВозможностьЧтенияXML() получается очередной тип данных XML и определяется, имеется ли соответствующий тип «1С:Предприятия».

Для чтения данных используется метод глобального контекста ПрочитатьXML(). В результате переменная Данные будет содержать объект «1С:Предприятия», полученный из сообщения обмена.

Затем, чтобы разрешить коллизию, когда один и тот же объект изменялся сразу в обоих узлах обмена, вызывается функция ПринятьИзменения() (см. листинг 3.12), в которую передаются полученный объект обмена (Данные) и ссылка на узел обмена, для которого отправлено сообщение (СообщениеОбмена.Отправитель).

После этого полученные данные записываются методом Записать(). Перед записью полученного объекта в параметрах обмена данными у него устанавливается узел отправителя, чтобы система при записи в базу данных не формировала записи регистрации изменений этого объекта для того узла, от которого они были только что получены.

Кроме того, в параметрах обмена данными устанавливается свойство Загрузка, информирующее систему о том, что запись объекта будет происходить в режиме обновления данных, полученных в результате обмена. Такое указание позволяет системе упростить процедуру записи объекта, отказавшись от ряда стандартных проверок и исключив изменения связанных данных, которые выполняются при обычной записи.

После выхода из цикла чтения вызывается метод ЗакончитьЧтение() объекта ЧтениеСообщенияОбмена и заканчивается чтение данных из XML-строки.

Теперь поместите в модуль ОбменМобильныеСервер вспомогательные функции, о которых говорилось выше: НуженПереносДанных() – листинг 3.10, УдалениеДанных() – листинг 3.11, ПринятьИзменения() – листинг 3.12.

Функция НуженПереносДанных() реализует стратегию распространения данных из основного приложения на мобильные устройства. По этой стратегии для планшета выгружаются только те заказы, которые относятся к пользователю (курьеру), вошедшему в мобильное приложение.

Листинг 3.10. Функция «НуженПереносДанных()»

```
Функция НуженПереносДанных(Данные, УзелОбмена) Экспорт

	Перенос = Истина;

#Если НЕ МобильныйАвтономныйСервер Тогда
	Если ТипЗнч(Данные) = Тип("ДокументОбъект.Заказ") Тогда

		// Проверяем, что курьер документа – это текущий пользователь
		Пользователь = ПользователиИнформационнойБазы.ТекущийПользователь();
		Владелец = Справочники.Пользователи.НайтиПоКоду(Пользователь);
		Если Данные.Курьер <> Владелец Тогда
			Перенос = Ложь;
		КонецЕсли;

	КонецЕсли;
#КонецЕсли

	Возврат Перенос;

КонецФункции
```

В параметре Данные в функцию передается полученный объект обмена.

Если тип объекта – ДокументОбъект.Заказ, то проверяется следующее условие переноса данных. Если реквизит Курьер документа Заказ не равен ссылке на элемент справочника Пользователи, соответствующий текущему пользователю мобильного приложения, то функция возвращает Ложь. В остальных случаях возвращается Истина.

Обратите внимание, что условие переноса проверяется только на основном сервере (#Если НЕ МобильныйАвтономныйСервер), то есть при переносе данных из основного приложения в автономную базу. В обратную сторону (из автономной базы в основную базу) будут приниматься все данные.

В случае если данные, полученные при выборке изменений, не подлежат выгрузке в узел обмена, производится вызов процедуры УдалениеДанных(). В этой процедуре для данных, имеющих объектную природу, создается объект УдалениеОбъекта. Этим достигается удаление в узле-приемнике

«ранее неправильно отосланных данных». Наборы записей регистров очищаются (листинг 3.11).

Листинг 3.11. Процедура «УдалениеДанных()»

```
Процедура УдалениеДанных(Данные) Экспорт

	ОбъектМетаданных = ?(ТипЗнч(Данные) = Тип("УдалениеОбъекта")
	                                    , Данные.Ссылка.Метаданные(), Данные.Метаданные());
	Если Метаданные.Справочники.Содержит(ОбъектМетаданных)
		ИЛИ Метаданные.Документы.Содержит(ОбъектМетаданных) Тогда

		Данные = Новый УдалениеОбъекта(Данные.Ссылка);

	ИначеЕсли Метаданные.РегистрыСведений.Содержит(ОбъектМетаданных) Тогда

		Данные.Очистить();

	КонецЕсли;

КонецПроцедуры
```

В функции ПринятьИзменения() разрешается коллизия при приеме данных обмена, когда один и тот же объект изменялся сразу в обоих узлах (листинг 3.12).

Листинг 3.12. Функция «ПринятьИзменения()»

```
Функция ПринятьИзменения(Отправитель, Данные) Экспорт

	Прием = Истина;

	Если ПланыОбмена.ИзменениеЗарегистрировано(Отправитель, Данные) Тогда

#Если НЕ МобильныйАвтономныйСервер Тогда
		Если ТипЗнч(Данные) = Тип("СправочникОбъект.Клиенты") ИЛИ
				ТипЗнч(Данные) = Тип("СправочникОбъект.ПричиныОтказа") Тогда
			Прием = Ложь;
		КонецЕсли;
#Иначе
		Если ТипЗнч(Данные) = Тип("ДокументОбъект.Заказ") Тогда
			Прием = Ложь;
		КонецЕсли;
#КонецЕсли

	КонецЕсли;

	Возврат Прием;
КонецФункции
```

В этой функции проверяется, зарегистрировано ли изменение объекта в узле обмена, для которого отправлены данные. Если да, то стратегия приема данных определяется в зависимости от контекста исполнения.

Если функция исполняется на основном сервере (#Если НЕ МобильныйАвтономныйСервер), то, в случае если тип объекта – СправочникОбъект.Клиенты или СправочникОбъект.ПричиныОтказа, возвращается Ложь.

Если функция исполняется на автономном сервере (#Если МобильныйАвтономныйСервер), то, в случае если тип объекта – ДокументОбъект.Заказ, возвращается Ложь.

Во всех остальных случаях возвращается Истина.

Таким образом, если данные обмена принимаются в основной базе, то изменения справочников Клиенты и ПричиныОтказа, полученные от планшета, будут отклонены, если эти объекты уже изменялись в офисе. То есть изменения нормативно-справочной информации в офисе имеют приоритет.

И, наоборот, если данные обмена принимаются в автономной базе, то изменения документов Заказ, полученные из офиса, будут отклонены, если эти объекты уже изменялись на планшете. То есть изменения заказов курьером имеют приоритет.

Во всех остальных случаях принимаются любые изменения.

После того как вы создали процедуры обмена, желательно сделать так, чтобы пользователь мобильного устройства (курьер) мог произвести обмен данными с основной базой в любой момент по своему желанию. Для этого добавьте в конфигурацию общую команду ОбменСЦентральнойБазой. Задайте синоним команды – Синхронизировать данные и укажите, что она будет отображаться в группе Панель действий.Сервис. Обработчик команды заполните следующим образом (листинг 3.13).

Листинг 3.13. Обработчик команды «ОбменСЦентральнойБазой»

```
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

    ОбменМобильныеАвтономныйКлиент.НачатьОбмен();

КонецПроцедуры
```

При выполнении этой команды обмен данными будет начинаться только на мобильном клиенте и при наличии связи с основным сервером. Эти условия проверяются в процедуре НачатьОбмен() модуля ОбменМобильныеАвтономныйКлиент (см. листинг 3.3).

Теперь в модуле приложения в процедуре ПриНачалеРаботыСистемы(), которую вы уже создавали ранее (см. листинг 3.1), подключите обработчик оповещения, который будет срабатывать при оповещении об окончании обмена методом Оповестить(), листинг 3.14.

Листинг 3.14. Процедура «ПриНачалеРаботыСистемы()»

```
Процедура ПриНачалеРаботыСистемы()

#Если МобильныйКлиент Тогда
    Если ОсновнойСерверДоступен() = Истина Тогда
        ОбменМобильныеАвтономныйКлиент.НачатьОбмен();
    КонецЕсли;

    ПодключитьОбработчикОповещения("ОбработкаОповещений");
#КонецЕсли

КонецПроцедуры
```

Саму процедуру обработчика оповещения ОбработкаОповещений() поместите также в модуле приложения и заполните следующим образом (листинг 3.15).

Листинг 3.15. Процедура «ОбработкаОповещений()»

```
#Если МобильныйКлиент Тогда
Процедура ОбработкаОповещений(ИмяСобытия, Параметр, Источник) Экспорт

    Если ИмяСобытия = "ОбменЗакончен" Тогда
        ОповеститьОбИзменении(Тип("ДокументСсылка.Заказ"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.Склады"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.Цвета"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.Размеры"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.ПричиныОтказа"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.ХранимыеФайлы"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.Клиенты"));
        ОповеститьОбИзменении(Тип("СправочникСсылка.Товары"));
        ОбновитьИнтерфейс();
        ПоказатьОповещениеПользователя("Выполнена синхронизация данных");
    КонецЕсли

КонецПроцедуры
#КонецЕсли
```

Эта процедура выполняется и компилируется только на мобильном клиенте. Когда из процедуры НачатьОбмен() модуля ОбменМобильныеАвтономныйКлиент (см. листинг 3.3) приходит оповещение методом Оповестить(«ОбменЗакончен»), то динамические списки заказов и списки всех справочников оповещаются об изменении и обновляются. А также пользователю показывается сообщение об окончании синхронизации данных.

Ну и, наконец, вы должны предусмотреть, чтобы заказы со статусом Выполнен, принятые в основную базу при обмене данными, формировали бы движения в регистрах накопления – такие же, как и при их проведении. Для этого поместите в модуль документа Заказ процедуру ПередЗаписью() и заполните ее следующим образом (листинг 3.16).

Листинг 3.16. Процедура «ПередЗаписью()»

```
Процедура ПередЗаписью(РежимЗаписи, РежимПроведения)

#Если НЕ МобильныйАвтономныйСервер Тогда
    Если ОбменДанными.Загрузка И СтатусЗаказа = Перечисления.СтатусыЗаказов.Выполнен Тогда

        Движения.ПричиныОтказа.Записывать = Истина;
        Движения.ОстаткиТоваров.Записывать = Истина;

        Для Каждого ТекСтрокаТовары Из Товары Цикл

            // регистр ОстаткиТоваров Приход
            Движение = Движения.ОстаткиТоваров.Добавить();
            Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
            Движение.Период = Дата;
            Движение.Товар = ТекСтрокаТовары.Товар;
            Движение.Цвет = ТекСтрокаТовары.Цвет;
            Движение.Размер = ТекСтрокаТовары.Размер;
            Движение.Склад = Склад;
            Движение.Заказано = ТекСтрокаТовары.Количество;

            Если ТекСтрокаТовары.Отказ Тогда
                // регистр ПричиныОтказа
                Движение = Движения.ПричиныОтказа.Добавить();
                Движение.Период = ДатаДоставки;
                Движение.Товар = ТекСтрокаТовары.Товар;
                Движение.ПричинаОтказа = ТекСтрокаТовары.ПричинаОтказа;
                Движение.Отказ = 1;

                // регистр ОстаткиТоваров Расход
                Движение = Движения.ОстаткиТоваров.ДобавитьРасход();
                Движение.Период = ДатаДоставки;
                Движение.Товар = ТекСтрокаТовары.Товар;
                Движение.Цвет = ТекСтрокаТовары.Цвет;
                Движение.Размер = ТекСтрокаТовары.Размер;
                Движение.Склад = Склад;
                Движение.Заказано = ТекСтрокаТовары.Количество;
            КонецЕсли;

        КонецЦикла;

    КонецЕсли;
#КонецЕсли

КонецПроцедуры
```

Обратите внимание, что формирование движений обрамляется инструкцией #Если НЕ МобильныйАвтономныйСервер … #КонецЕсли, так как регистры накопления недоступны в автономной базе.

По тем же причинам такие же условия нужно добавить в процедуру обработки проведения документа Заказ, иначе при изменении и проведении заказа в режиме автономной работы будет получена ошибка (листинг 3.17).

Листинг 3.17. Обработка проведения документа «Заказ»

```
Процедура ОбработкаПроведения(Отказ, Режим)

#Если НЕ МобильныйАвтономныйСервер Тогда
        Если СтатусЗаказа = Перечисления.СтатусыЗаказов.ВРаботе Тогда

                ...

        КонецЕсли;
#КонецЕсли

КонецПроцедуры
```

В заключение вы должны обеспечить уникальность номеров и кодов при создании новых объектов, которые могут добавляться и в мобильном, и в основном приложении. Дело в том, что курьер может в автономном режиме вводить новые заказы, новых клиентов и причины отказа от товаров.

Поэтому сделайте так, чтобы при вводе этих объектов на мобильном устройстве их номер или код начинался бы с префикса – кода узла обмена.

Для этого в модуле документа Заказ создайте процедуру – обработчик события ПриУстановкеНовогоНомера() и заполните следующим образом (листинг 3.18).

Листинг 3.18. Процедура «ПриУстановкеНовогоНомера()»

```
Процедура ПриУстановкеНовогоНомера(СтандартнаяОбработка, Префикс)

#Если МобильныйАвтономныйСервер Тогда
        Префикс = ПланыОбмена.Мобильные.ЭтотУзел().Код + "-";
#КонецЕсли

КонецПроцедуры
```

Событие ПриУстановкеНовогоНомера возникает в момент, когда выполняется установка нового номера документа. Вторым параметром вызова обработчика передается префикс, который будет заполнен в данной процедуре и использован системой для генерации кода. Параметру Префикс присваивается строковый код предопределенного узла автономной базы.

По аналогии в модули объектов справочника Клиенты и справочника ПричиныОтказа добавьте обработчик события ПриУстановкеНовогоКода(), листинг 3.19.

Листинг 3.19. Процедура «ПриУстановкеНовогоКода()»

```
Процедура ПриУстановкеНовогоКода(СтандартнаяОбработка, Префикс)

#Если МобильныйАвтономныйСервер Тогда
        Префикс = ПланыОбмена.Мобильные.ЭтотУзел().Код + "-";
#КонецЕсли

КонецПроцедуры
```

Таким образом, в офисной базе будут создаваться все объекты обычным образом, без префиксов номеров или кодов. А заказы, клиенты и причины отказа, созданные в автономной базе, будут иметь номера/коды с префиксом «002-», «003-» и т. д.

Теперь вам осталось создать пользователей информационной базы.

Сначала нужно создать роли Администратор, Менеджер и Курьер, которые этим пользователям будут назначены. Как уже говорилось, основной задачей курьера является обслуживание заказов, а также он может добавлять и редактировать клиентов и причины отказа от товаров. Исходя из этого для роли Курьер задайте права Чтение, Просмотр и Ввод по строке для всех объектов конфигурации и все права, кроме права интерактивного удаления, на документ Заказ и справочники Клиенты и ПричиныОтказа.

Для плана обмена Мобильные установите права Чтение, Добавление, Изменение, Удаление и Просмотр (так как при обмене данными возможно программное изменение плана обмена). А для общей команды ОбменСЦентральнойБазой установите право Просмотр.

Обновите информационную базу (F7).

Затем добавьте пользователей информационной базы Администратор, Курьер1, Курьер2 с соответствующими ролями (рис. 3.5).

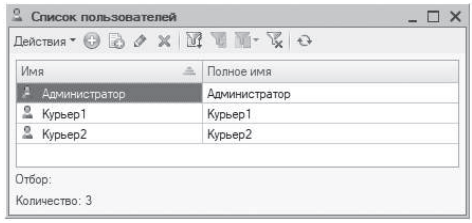


Рис. 3.5. Список пользователей информационной базы в конфигураторе

В заключение вам нужно включить в состав автономной конфигурации все новые объекты, которые вы добавили в ходе реализации обмена (рис. 3.6):

- План обмена Мобильные.

- Роли:
  - Администратор;
  - Менеджер;
  - Курьер;
- Общая команда ОбменСЦентральнойБазой.
- Общие модули:
  - ОбменМобильныеАвтономныйКлиент;
  - ОбменМобильныеАвтономныйСервер;
  - ОбменМобильныеСервер.

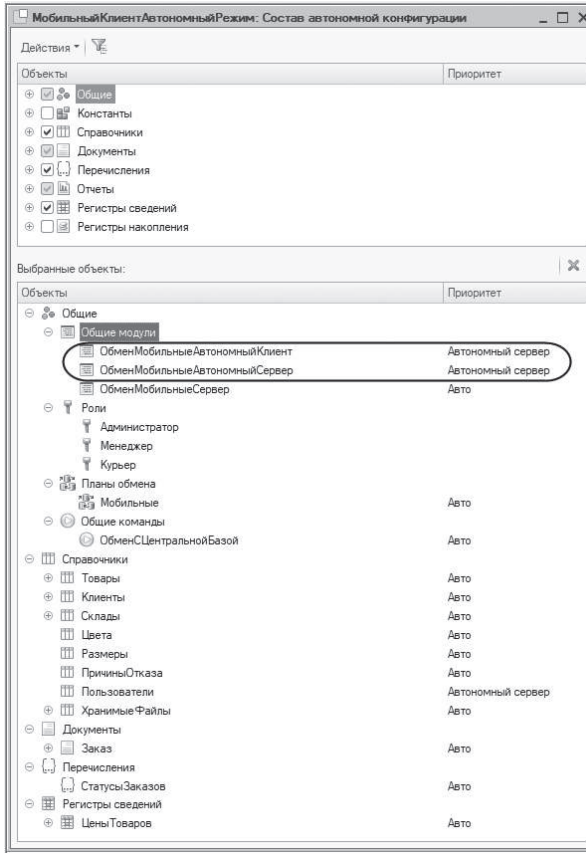


Рис. 3.6. Состав автономной конфигурации

Таким образом вся функциональность обмена данными, реализованная в общих модулях, станет доступна в автономной конфигурации на мобильном устройстве.

Обратите внимание, что в состав автономной конфигурации не включены общий модуль ОбменМобильныеВызовПК и константа КодНовогоУзлаПланаОбмена, так как они используются только на стороне основного сервера.

А также заметьте, что в колонке Приоритет может задаваться приоритет использования объекта автономной конфигурации. Он может принимать следующие значения:

- Основной сервер – в этом случае система «1С:Предприятие» старается использовать для доступа к объекту основной сервер системы.
- Автономный сервер – в этом случае система «1С:Предприятие» старается использовать для доступа к объекту конфигурации автономный сервер.
- Авто – это значение по умолчанию. Данное значение трактуется как Основной сервер для типообразующих объектов конфигурации (например, справочник), а для подчиненных им объектов (например, форма справочника) – трактуется как значение типообразующего объекта.

У модулей ОбменМобильныеАвтономныйКлиент и ОбменМобильныеАвтономныйСервер и справочника Пользователи установите приоритет использования Автономный сервер. Дело в том, что при наличии соединения в обычном режиме работы по умолчанию будет использоваться основной сервер. Но, так как обмен данными начинается всегда на автономном сервере, вам понадобится доступ к объектам обмена в автономной базе.

Тестирование обмена данными

Запустите конфигурацию в режиме исполнения на стационарном компьютере, откройте справочник Пользователи и внесите туда пользователей с кодами Администратор, Курьер1, Курьер2, заданными в списке пользователей информационной базы (рис. 3.7).

Справочник пользователей нужен для определения текущего пользователя, от имени которого запущено мобильное приложение. Если курьер в заказе соответствует этому пользователю, то при обмене данными заказы из основного приложения выгружаются на мобильное устройство.

Вообще-то мобильный клиент с автономным режимом не поддерживает работу с пользователями. В то же время, при запуске мобильного приложения выполняется запрос пользователя и пароля. Однако тот пользователь, который указан при первом входе в приложение, должен использоваться и при всех следующих запусках. При попытке запуска мобильного приложения под другим пользователем будет выдано сообщение об ошибке.

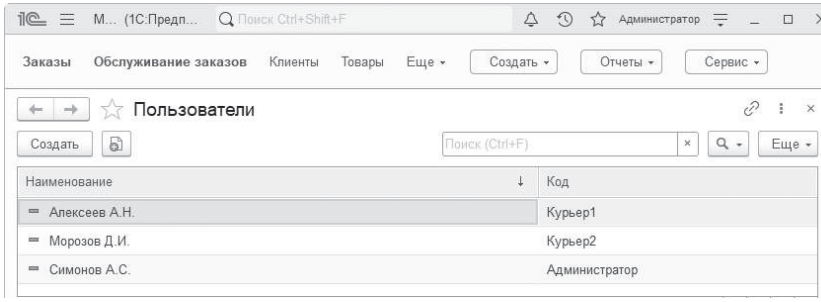


Рис. 3.7. Справочник пользователей информационной базы в режиме «1С:Предприятие»

Таким образом, если на одном мобильном устройстве с одной основной информационной базой работают несколько пользователей – для каждого пользователя необходимо создать собственное приложение в списке приложений мобильной платформы.

Итак, убедитесь, что на планшете есть интернет-соединение, и запустите мобильный клиент с автономным режимом от имени пользователя Курьер1 (рис. 3.8). Этому пользователю в справочнике Пользователи соответствует курьер Алексеев А.Н. (см. рис. 3.7).

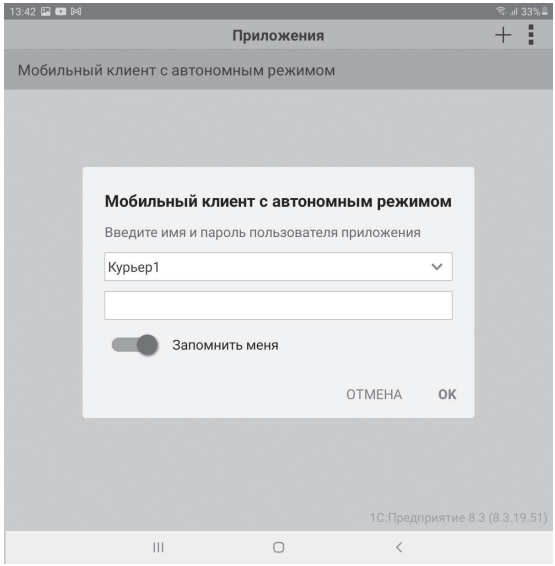


Рис. 3.8. Диалог авторизации при первом входе в мобильное приложение

Обратите внимание, что в диалоге авторизации стандартно включен переключатель Запомнить меня. Это значит, что, пока пользователь специально не нажмет Завершить работу в диалоге свойств пользователя, больше диалог авторизации ему показываться не будет.

После этого при начале работы приложения будет выполнен обмен данными с основным приложением. В результате инициализации обмена и в основной, и в автономной базе данных на мобильном устройстве в списке узлов плана обмена Мобильные появятся соответствующие записи (рис. 3.9).

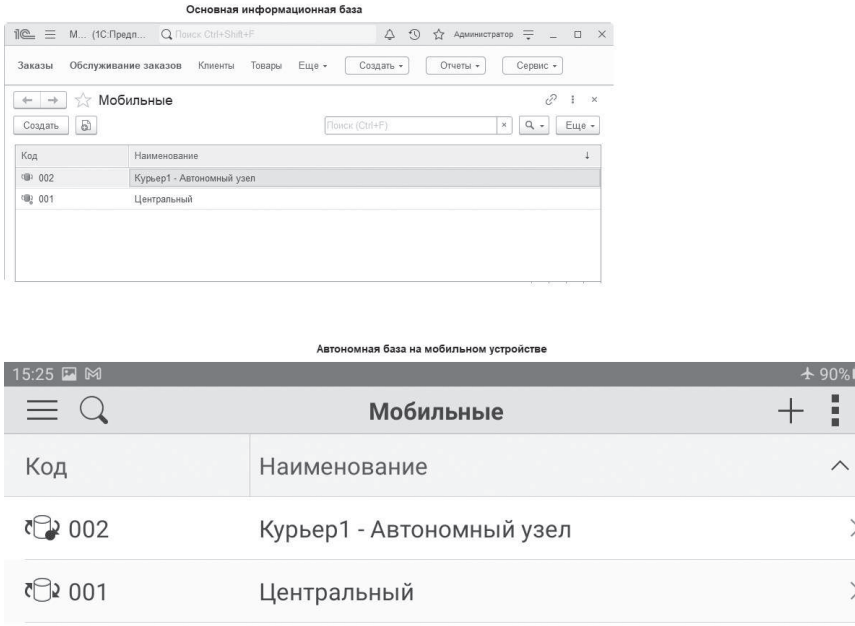


Рис. 3.9. Список узлов обмена в основной и в автономной базах данных

ВНИМАНИЕ!

Нужно понимать, что при работе в основном режиме на мобильном устройстве вы подключаетесь к основной базе и видите ее содержимое. Поэтому при наличии соединения с основным сервером в мобильном приложении вы увидите такой же список узлов обмена, как и на рис. 3.9 сверху. Чтобы увидеть список узлов обмена в автономной базе, показанный на рис. 3.9 снизу, нужно перейти в автономный режим работы (например, включить на устройстве режим полета).

Чтобы протестировать результат обмена данными на стороне автономной базы, включите режим полета на планшете – в результате мобильное приложение перейдет в автономный режим работы. Команды для доступа к объектам, отсутствующим в автономной конфигурации, станут недоступны. Вы можете убедиться, что теперь объекты, включенные в состав автономной конфигурации, заполнены данными. То есть при старте приложения произошел первоначальный перенос данных из основной базы на мобильное устройство.

Внесите теперь какие-то изменения в автономную базу. Например, откройте список клиентов и измените адрес одного из них. При следующей синхронизации данных ваши изменения будут перенесены в основную базу.

Вернитесь на начальную страницу приложения и пролистните вправо список команд. Как вы видите, команда Синхронизировать данные доступна, так как вы включили ее в состав автономной конфигурации, но при выполнении этой команды будет выдано сообщение о невозможности обмена данными по причине недоступности основного сервера (рис. 3.10).

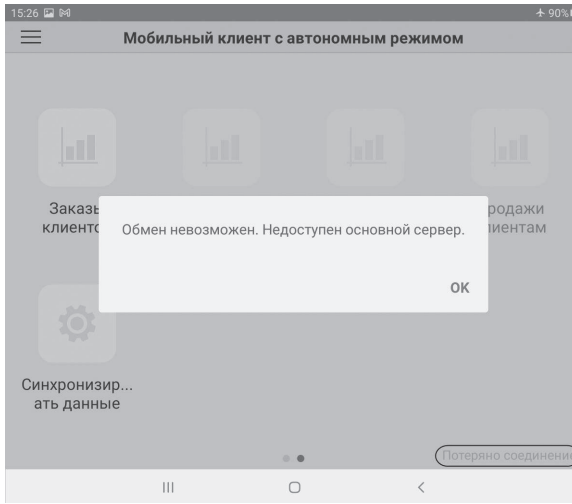


Рис. 3.10. Команда «Синхронизировать данные» в автономном режиме

Отключите режим полета. При появлении связи будет написано, что соединение восстановлено, в модуле приложения сработает событие ПриИзмененииДоступностиОсновногоСервера, и будет выполнен обмен данными с основным приложением. После этого вы увидите сообщение об окончании синхронизации данных (рис. 3.11).

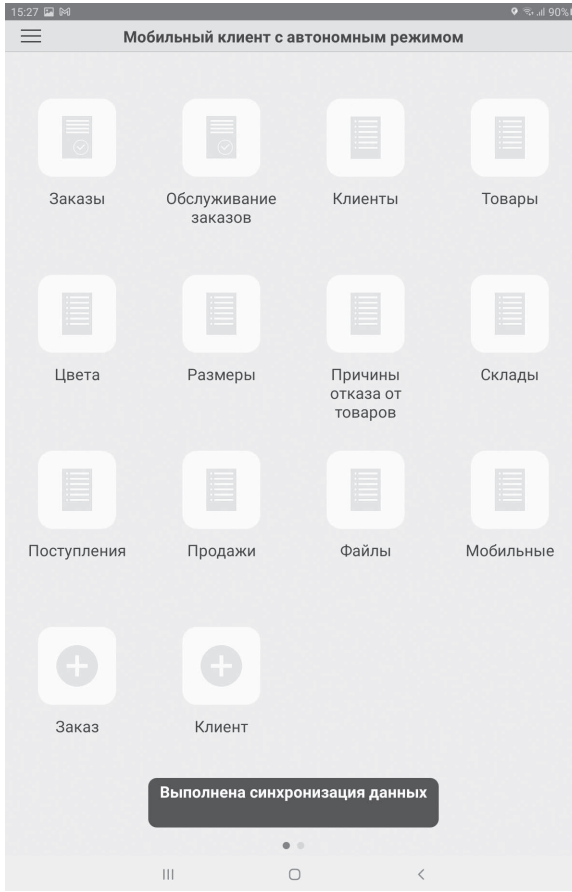


Рис. 3.11. Синхронизация данных при появлении связи с основным сервером

В результате в справочнике клиентов в основной базе появится ваше изменение, сделанное в автономном режиме работы.

То есть все работает ровно так, как вы и запрограммировали при разработке процедур обмена!

Теперь проверьте, правильно ли переносятся заказы из основного приложения. Вы реализовали стратегию распространения данных, согласно которой при обмене данными из основной базы на мобильное устройство выгружаются только те заказы, которые соответствуют пользователю (курьеру) мобильного приложения.

Откройте список заказов в автономном режиме работы. Как вы видите, у всех заказов указан курьер Алексеев А.Н. Код этого курьера в справочнике Пользователи – Курьер1, что соответствует имени пользователя, под которым запущено мобильное приложение (рис. 3.12).

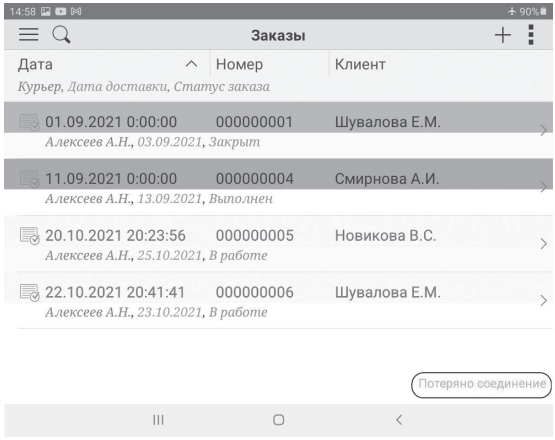


Рис. 3.12. Список заказов в автономной базе

При наличии соединения с основным сервером на мобильном устройстве доступен полный список заказов, содержащихся в основной базе (рис. 3.13).

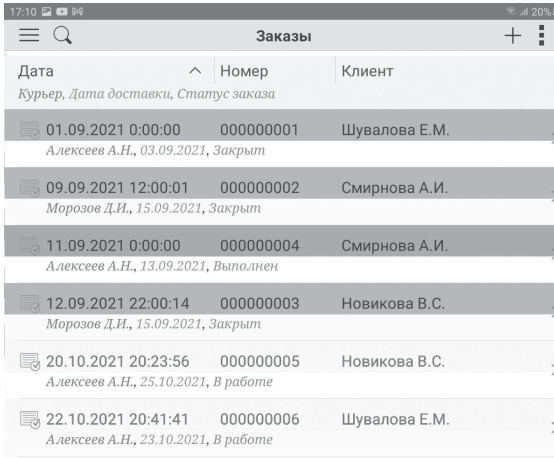


Рис. 3.13. Список заказов в основной базе

Если курьер не хочет видеть заказы других курьеров в своем приложении (а при наличии соединения он их увидит), то он может настроить отбор по курьеру в списке заказов с помощью системы компоновки данных – точно так же, как это делается в основном приложении (рис. 3.14, 3.15).

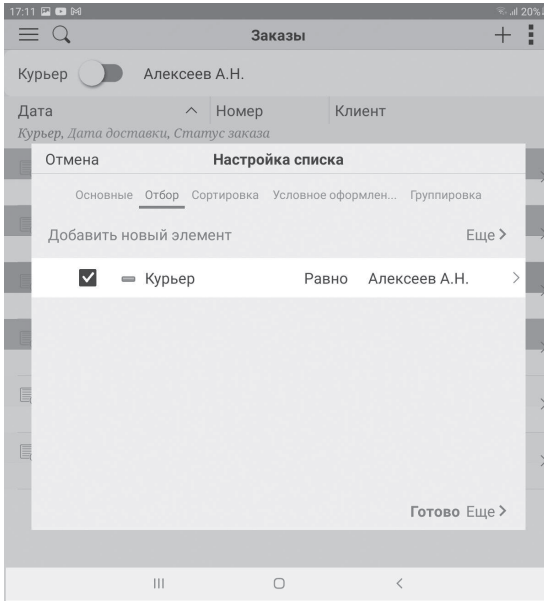


Рис. 3.14. Настройка отбора по курьеру в списке заказов

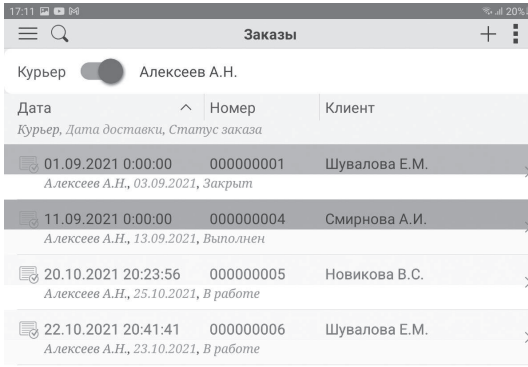


Рис. 3.15. Отбор по курьеру в списке заказов

В результате эта настройка будет видна только в основном приложении либо в основном режиме работы мобильного приложения. Поскольку это пользовательская настройка, она не будет никому мешать и будет доступна только тому курьеру, который настроил отбор.

Ну, и в завершение проверьте, как происходит обмен новыми данными и данными, измененными сразу в обеих базах.

Перейдите в автономный режим работы на мобильном устройстве, введите новый заказ и новую причину отказа. При восстановлении соединения будет выполнен обмен данными, и в основной базе появятся заказ с номером «002-00001» и причина отказа с кодом «002-00001».

Теперь измените какой-нибудь заказ, а также данные какого-нибудь клиента в основной базе. На мобильном устройстве в автономном режиме работы также измените этот же заказ и этого же клиента. После синхронизации данных в автономной базе данных останутся изменения заказа и появятся изменения клиента, сделанные в основном приложении. В основной же базе, наоборот, останутся изменения клиента и будут приняты изменения заказа, сделанные в мобильном приложении в автономном режиме работы.

Таким образом, коллизия при приеме данных тоже решается правильно – изменения клиентов и причин отказа в основном приложении имеют приоритет над мобильным устройством, а изменения заказа на мобильном устройстве имеет приоритет над основным приложением.

Режимы работы

Обычный режим

В условиях соединения с основным сервером мобильный клиент с автономным режимом работает в обычном режиме работы. Работа мобильного приложения в этом случае ничем не отличается от работы обычного мобильного клиента, о котором рассказывалось в предыдущей главе книги «Приложение мобильного клиента». Функциональность пользователя мобильного приложения при этом может быть ограничена только его правами доступа к данным.

Обновление конфигурации

Как уже говорилось, для первого запуска мобильного клиента с автономным режимом всегда требуется устойчивый канал связи с основной информационной базой, так как при первом запуске выполняется загрузка автономной конфигурации.

Далее при каждом запуске мобильного приложения, а также при создании/восстановлении соединения между мобильным приложением и основным сервером выполняется проверка наличия обновления автономной конфигурации.

При старте мобильного приложения в обычном режиме работы автономная конфигурация и информационная база на мобильном устройстве обновляется автоматически, практически незаметно для пользователя.

Если же приложение работало в автономном режиме и если при восстановлении соединения найдена новая версия конфигурации, пользователю будет показан запрос об обновлении версии приложения на мобильном устройстве (рис. 3.16).

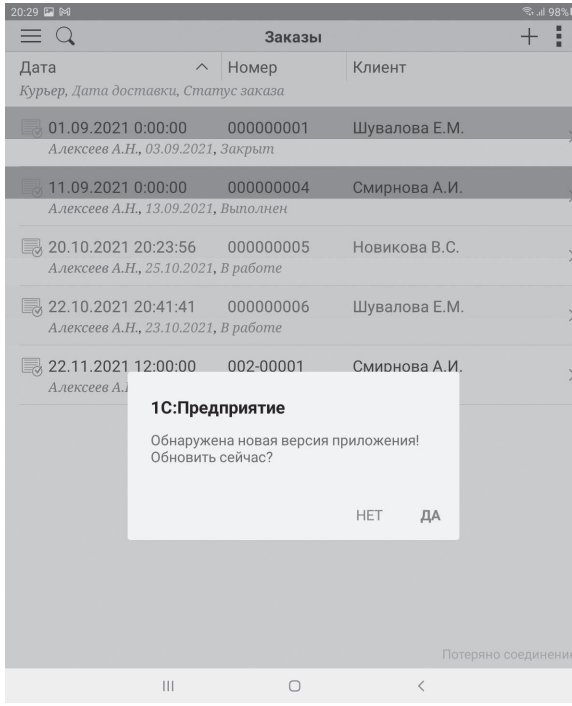


Рис. 3.16. Запрос об обновлении автономной конфигурации

Использование основного и автономного серверов

В обычном режиме работы для серверных вызовов может использоваться как основной, так и автономный сервер. При вызове система анализирует приоритет использования того или иного объекта, а также заданные режимы работы клиентского приложения и на основании анализа принимает решение о том, какой сервер будет использоваться для вызова.

Однако могут возникнуть ситуации, когда необходимо изменить сервер, который будет использоваться для вызова:

- Например, если нужно вызвать серверный метод основного сервера из метода, который выполняется на стороне автономного сервера. Для этого можно использовать свойство глобального контекста ОсновнойСервер. С помощью данного свойства на автономном сервере доступны серверные общие модули основного сервера. В этом случае вызов будет выглядеть следующим образом: ОсновнойСервер.<ИмяСереверногоОбщегоМодуля>.<ИмяМетодаОбщегоМодуля()>. Такой пример был рассмотрен выше, при реализации процедур обмена.
- Или, если нужно на стороне клиентского приложения выполнить серверный вызов, который будет использовать строго основной или строго автономный сервер вне зависимости от текущих параметров. Для этого можно использовать методы глобального контекста УстановитьИспользуемыйСервер() или Установить ПреимущественноеИспользованиеОсновногоСервера(). В эти методы передаются значения системного перечисления ИспользуемыйСервер: Основной или Автономный.

Объекты автономной конфигурации в обычном режиме работы, как правило, используют основной сервер, кроме тех объектов, у которых в приоритете использования явно указан автономный сервер. Приоритет использования того или иного сервера задается при настройке автономной конфигурации. Пример показывался в конце раздела про реализацию обмена.

Интерактивное переоткрытие формы

Если вы работали с формой в обычном режиме, то при наличии интернет-соединения форма была открыта с основного сервера. При прерывании канала связи с информационной базой эта форма становится недоступной для работы, ее можно только закрыть. Но если у формы есть аналог в автономной конфигурации, то ее можно переоткрыть с автономного сервера с помощью кнопки Открыть автономно, которая появляется в правом верхнем углу приложения (рис. 3.17).

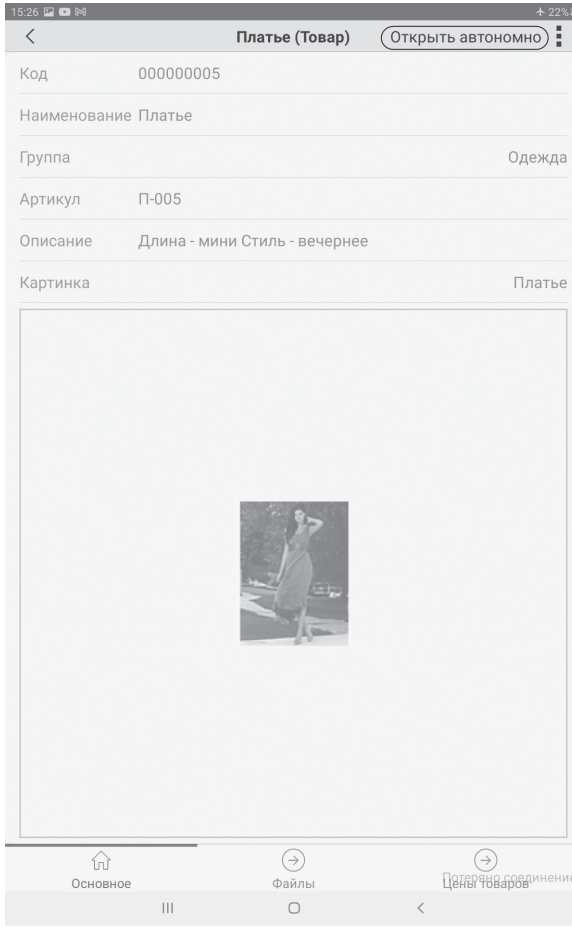


Рис. 3.17. Форма товара в случае потери интернет-соединения

При работе в обычном режиме мобильное приложение автоматически пытается восстановить соединение с основной информационной базой.

Если соединение было восстановлено, когда форма была открыта автономно (с автономного сервера), то вам будет предложено переоткрыть форму онлайн и продолжить с ней работать в обычном режиме (рис. 3.18).

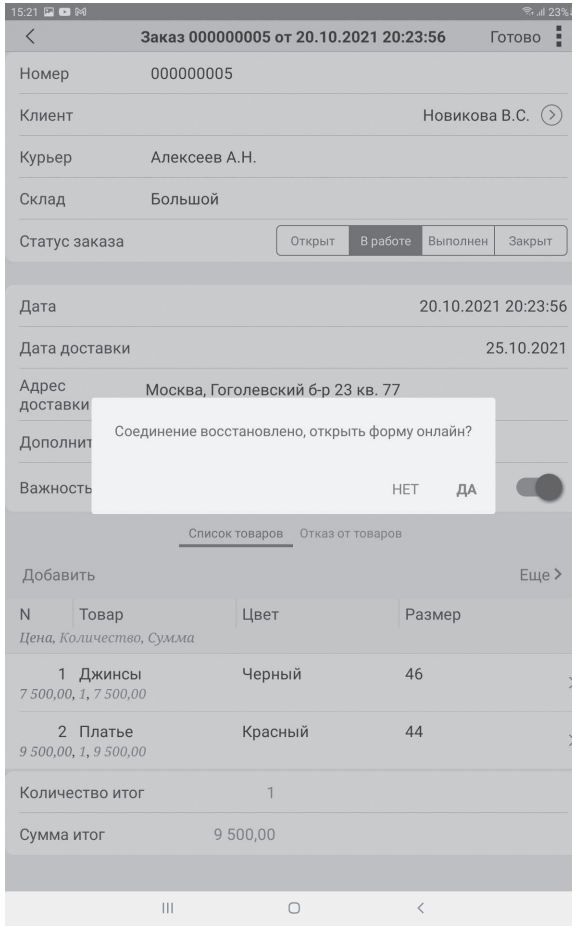


Рис. 3.18. Форма заказа при восстановлении соединения

Кроме того, находясь в обычном режиме работы, можно принудительно переоткрыть форму автономно, а затем снова переоткрыть ее онлайн. Для этого нужно перетащить соответствующие команды ФормаОткрытьСОсновногоСервера и ФормаОткрытьСАвтономногоСервера из списка стандартных команд в командную панель формы (рис. 3.19).

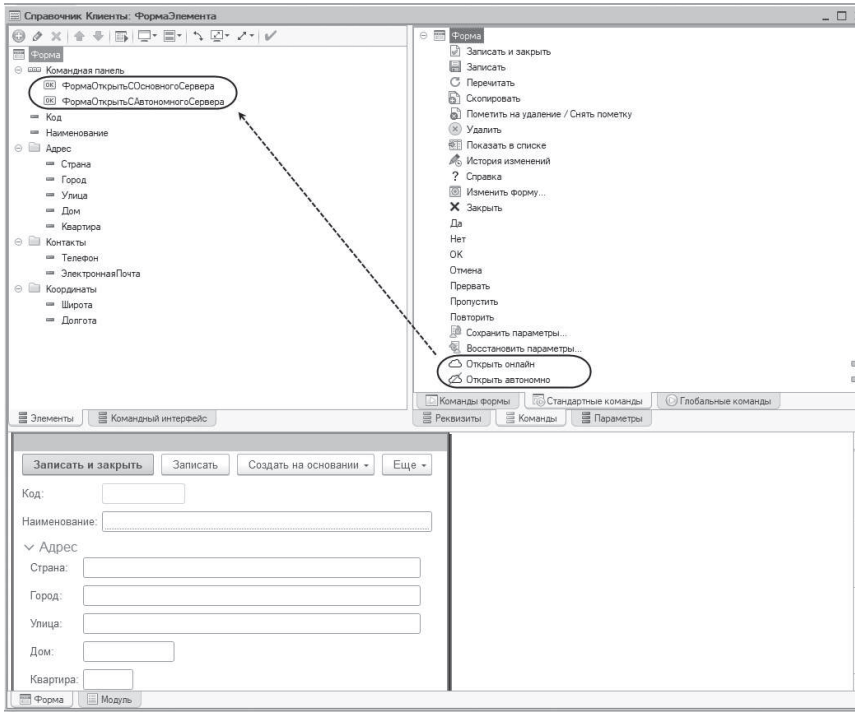


Рис. 3.19. Форма элемента справочника «Клиенты» в конфигураторе

После этого в меню действий, открывающемся при нажатии на три вертикальные точки в правом верхнем углу формы, появятся команды Открыть автономно и Открыть онлайн (рис. 3.20).

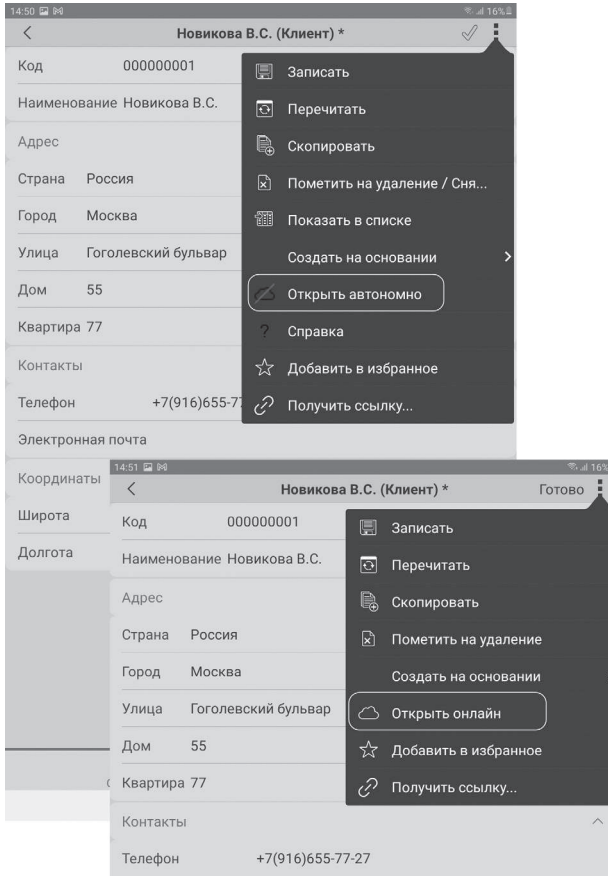


Рис. 3.20. Принудительное переоткрытие формы в мобильном приложении

Автономный режим

В автономном режиме работы используется автономная информационная база, состав которой определяется во время настройки состава автономной конфигурации, а объем данных – параметрами плана обмена с точностью до последней синхронизации.

Важно понимать, что если вы работали в обычном режиме, то в случае потери соединения с основной информационной базой мобильное приложение переходит в автономный режим работы, но при этом пытается автоматически восстановить потерянное соединение.

Кроме того, можно перейти в автономный режим принудительно. Это делается либо перед стартом мобильного приложения в диалоге свойств информационной базы (рис. 3.21), либо в любой момент в диалоге свойств пользователя (рис. 3.22). В этом случае приложение не пытается автоматически восстановить соединение с основной информационной базой.

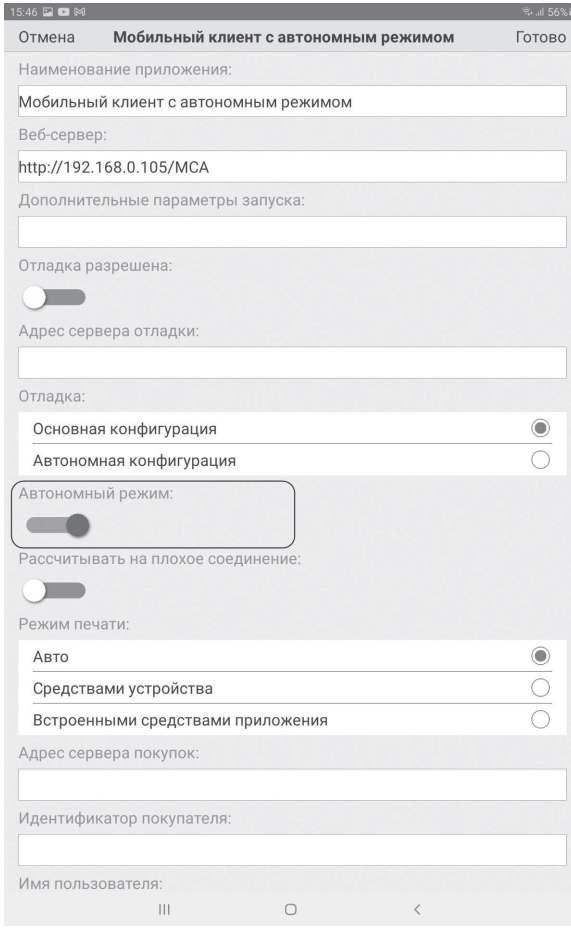


Рис. 3.21. Принудительная установка автономного режима

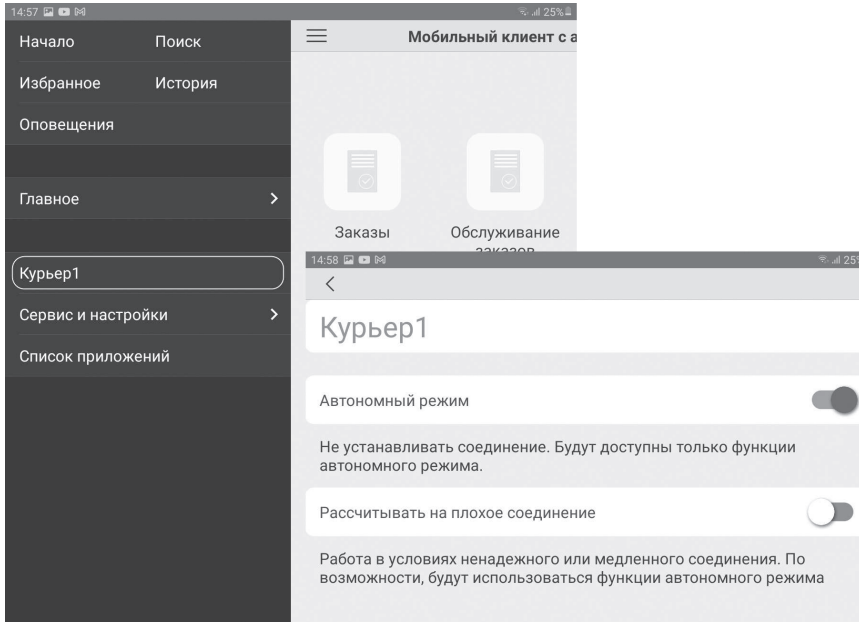


Рис. 3.22. Принудительная установка автономного режима

Объекты, включенные в состав автономной конфигурации, используют только автономный сервер, а остальные объекты становятся недоступны.

Плохое соединение с основной информационной базой

Таким же образом можно принудительно включить режим плохого соединения с основной информационной базой. Это делается в диалоге свойств информационной базы мобильного приложения или в диалоге свойств пользователя (рис. 3.23).

Этот режим работы обычно включается в условиях ненадежного или медленного соединения. Он похож на обычный режим работы, но объекты, включенные в состав автономной конфигурации, будут использовать автономный сервер (вне зависимости от заданного приоритета). Остальные объекты будут доступны только после восстановления соединения.

Таким образом, для объектов, доступных в автономной конфигурации, в режиме плохого соединения всегда будут использоваться автономный сервер и данные автономной базы на мобильном устройстве – даже при наличии устойчивого интернет-соединения.

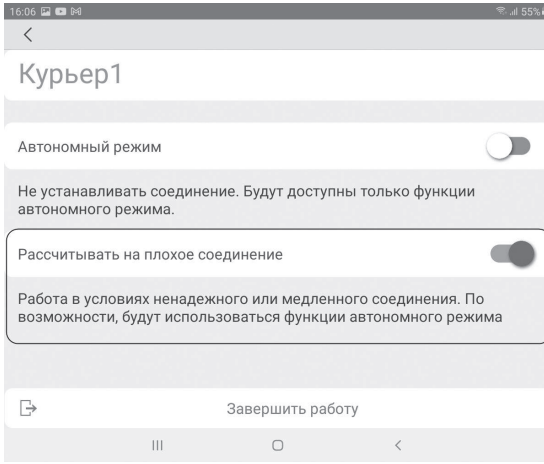


Рис. 3.23. Принудительная установка режима плохого соединения

Данные формы при переоткрытии с другого сервера

Формы объектов, входящие в состав автономной конфигурации, могут реагировать на изменение доступности основного сервера с помощью события формы ПриИзмененииДоступностиОсновногоСервера. В обработчике этого события можно программно переоткрыть форму с помощью метода ПереоткрытьСДругогоСервера(). Или же можно сделать это интерактивно, как было показано на рис. 3.17–3.20.

В случае, когда форма, открытая онлайн (с основного сервера), переоткрывается автономно (с автономного сервера), а также, наоборот, форма, открытая автономно, переоткрывается онлайн, происходит ряд событий, в обработчиках которых можно адаптировать форму на мобильном устройстве под изменяющиеся условия: доступность или недоступность основного сервера.

Сначала у текущей формы происходит событие ПередПереоткрытиемСДругогоСервера. В обработчик события передаются все параметры, которые использовались при открытии текущей формы. В обработчике этого события можно модифицировать список параметров, с которыми будет открыта новая форма. Если в обработчике установить параметр Отказ в значение Истина, то форма не будет переоткрыта.

Затем создается новая (переоткрываемая) форма. При этом срабатывают все серверные события создаваемой формы.

У новой (только что созданной) формы происходит событие ПриПереоткрытииСДругогоСервера. В обработчик события передается переоткрываемая форма и параметры заполнения. По умолчанию параметр СтандартнаяОбработка обработчика события установлен в значение Истина. Это значит, что платформа автоматически перенесет все данные из «старой» в новую форму, опираясь на значение параметра ПараметрыЗаполнения. Изменяя значения этого параметра, можно повлиять на формирование списка копируемых реквизитов из одной формы в другую. Например, можно добавить в список исключаемых те реквизиты, которые не могут быть или не должны быть скопированы.

При переносе данных следует учитывать, что если реквизиты имеют сложные типы (например, табличный документ или диаграмма), то скопировать их не получится (потому что, в частности, состояние табличного документа на клиенте доступно не полностью). А также данные табличных частей, списков значений и т. п. будут перенесены в том случае, если эти данные полностью загружены в оперативную память.

Если параметр СтандартнаяОбработка установлен в значение Ложь, это означает, что вы должны самостоятельно выполнить перенос данных из «старой» формы в новую.

Чтобы попробовать описанные выше возможности, выполните следующий пример. При переоткрытии формы (программном или интерактивном) должно быть выполнено следующее условие: если форма клиента была открыта онлайн, то при ее переоткрытии в автономную форму переносятся все данные, кроме реквизита Долгота. А из автономной формы в форму, открываемую онлайн, переносятся все данные, кроме реквизита Широта.

Для этого в форме справочника Клиенты создайте обработчик события ПриПереоткрытииСДругогоСервера() и заполните его следующим образом (листинг 3.20).

Листинг 3.20. Процедура «ПриПереоткрытииСДругогоСервера()»

```
&НаКлиенте
Процедура ПриПереоткрытииСДругогоСервера(ИсходнаяФорма
                                        , ПараметрыЗаполнения, СтандартнаяОбработка)

    Если ИсходнаяФорма.ИспользуемыйСерверФормы = ИспользуемыйСервер.Основной Тогда
        // Не переносить в автономную форму из основной формы
        ПараметрыЗаполнения.ИсключитьРеквизиты.Добавить("Объект.Долгота");
    КонецЕсли;

    Если ИсходнаяФорма.ИспользуемыйСерверФормы = ИспользуемыйСервер.Автономный Тогда
        // Не переносить в основную форму из автономной формы
        ПараметрыЗаполнения.ИсключитьРеквизиты.Добавить("Объект.Широта");
    КонецЕсли;

КонецПроцедуры
```

В этом обработчике с помощью метода ИсключитьРеквизиты() объекта ПараметрыЗаполненияПриПереоткрытииФормы, содержащегося в параметре ПараметрыЗаполнения, в массив путей к исключаемым реквизитам добавляется путь к реквизиту Долгота (в случае если форма была открыта с основного сервера) или путь к реквизиту Широта (в случае если форма была открыта с автономного сервера).

Сервер, с которого загружена форма, определяется с помощью свойства ИспользуемыйСерверФормы, которое возвращает значения системного перечисления ИспользуемыйСервер: Основной или Автономный.

Проверьте, как это работает. Откройте форму клиента в условиях соединения с основным сервером, измените значения реквизитов Широта (например, введите значение 99) и Долгота (33).

Теперь переоткройте форму автономно с помощью команды Открыть автономно в меню действий или сымитируйте потерю соединения, включив режим полета, и нажмите Открыть автономно в правом верхнем углу формы. В результате в автономную форму клиента перенесется значение реквизита Широта (99), а значение реквизита Долгота останется таким, каким оно было до изменения (0) в основной базе.

Теперь измените значения реквизитов Широта (44) и Долгота (22). Затем переоткройте форму онлайн с помощью команды Открыть онлайн в меню действий или отключите режим полета и нажмите Открыть онлайн в появившемся запросе при восстановлении соединения. В результате в форму клиента, открытую онлайн, перенесется значение реквизита Долгота (22), а значение реквизита Широта останется таким, каким оно было до изменения (0) в автономной базе.

Таким образом, введенные, но еще не сохраненные данные между формой, открытой онлайн, и формой, открытой автономно, переносятся так, как вы и задали в обработчике события ПриПереоткрытииСДругогоСервера().

Функциональность формы при наличии/отсутствии соединения

Обычно в формах есть такая функциональность, которая имеет смысл только при наличии соединения с основной информационной базой, так как используемых в ней данных нет в автономной базе. Тогда при наличии соединения эта функциональность будет доступна, а при отсутствии – скрыта.

Например, в форме товара рассчитывается его остаток, полученный из данных регистра накопления, которые недоступны автономно. Поэтому нужно рассчитывать остаток товара только в том случае, когда исполнение находится на основном сервере (в ветке условия #Если НЕ МобильныйАвтономныйСервер).

Чтобы увидеть на примере, как меняется функциональность формы в зависимости от наличия соединения с основным сервером, выполните следующий пример.

Откройте в конфигураторе форму элемента справочника Товары и добавьте в нее группу без отображения (горизонтальную, если возможно). Затем добавьте в форму реквизит Остаток числового типа и команду ПолучитьОстатокТовара и перетащите их в группу ГруппаОстаток (рис. 3.24).

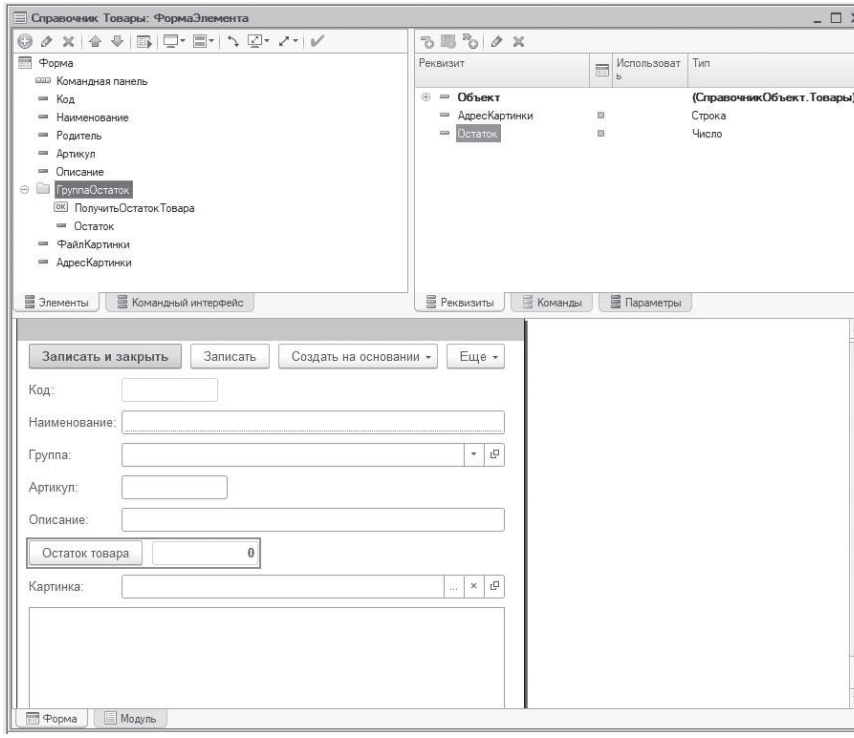


Рис. 3.24. Форма товара в конфигураторе

Создайте обработчик команды ПолучитьОстатокТовара и заполните его следующим образом (листинг 3.21).

Листинг 3.21. Обработчик команды «ПолучитьОстатокТовара»

```
&НаКлиенте
Процедура ПолучитьОстатокТовара(Команда)
        Остаток = ДанныеРегистровНакопления.ПолучитьОстатокТовара(Объект.Ссылка);
КонецПроцедуры
```

В этом обработчике вызывается функция ПолучитьОстатокТовара() общего модуля ДанныеРегистровНакопления, которая возвращает остаток товара из регистра накопления ОстаткиТоваров.

Затем создайте общий модуль ДанныеРегистровНакопления со свойствами Сервер и Вызов сервера и поместите в нем функцию для получения остатка товара по переданной в нее ссылке на товар (листинг 3.22).

Листинг 3.22. Функция «ПолучитьОстатокТовара()»

```
Функция ПолучитьОстатокТовара(ТоварСсылка) Экспорт

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       ОстаткиТоваров.КоличествоОстаток
        |ИЗ
        |       РегистрНакопления.ОстаткиТоваров.Остатки( , Товар = &Товар) КАК ОстаткиТоваров";

        Запрос.УстановитьПараметр("Товар", ТоварСсылка);

        Результат = Запрос.Выполнить();
        Если Результат.Пустой() Тогда
                Возврат 0;
        КонецЕсли;

        Остатки = Результат.Выбрать();
        Остатки.Следующий();
        Остаток = Остатки.КоличествоОстаток;

        Возврат Остаток;

КонецФункции
```

Теперь вам нужно сделать так, чтобы в случае, когда форма открыта онлайн, с помощью функции ПолучитьОстатокТовара() запрашивался бы остаток товара, а когда форма открыта автономно, поле остатка и команда для его получения скрывались бы от пользователя.

Поэтому добавьте в обработчик события формы ПриСозданииНаСервере следующий фрагмент (листинг 3.23).

Листинг 3.23. Процедура «ПриСозданииНаСервере()»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

	ФайлКартинки = Объект.ФайлКартинки;
	Если НЕ ФайлКартинки.Пустая() Тогда
		АдресКартинки = ПолучитьНавигационнуюССылку(ФайлКартинки, “ДанныеФайла”)
	Конецесли;

#Если НЕ МобильныйАвтономныйСервер Тогда
	Остаток = ПолучитьОстатокТовараНаСервере(Объект.Ссылка);
#Иначе
	ЭтотОбъект.Элементы.ГруппаОстаток.Видимость = Ложь;
#КонецЕсли

КонецПроцедуры
```

С помощью выделенного фрагмента проверяется, находится ли исполнение на основном сервере (#Если НЕ МобильныйАвтономныйСервер). Если нет, то группа ГруппаОстаток скрывается.

И в заключение вам нужно обеспечить, чтобы при переоткрытии формы онлайн автоматически отображался бы актуальный остаток товара. Для этого вам нужно создать обработчик события формы ПриПереоткрытииСДругогоСервера и отменить его стандартную обработку (листинг 3.24).

Листинг 3.24. Процедура «ПриПереоткрытииСДругогоСервера()»

```
&НаКлиенте
Процедура ПриПереоткрытииСДругогоСервера(ИсходнаяФорма
                                         , ПараметрыЗаполнения, СтандартнаяОбработка)
	СтандартнаяОбработка = Ложь;
КонецПроцедуры
```

Дело в том, что, как уже говорилось, перед событием ПриПереоткрытииСДругогоСервера у формы возникает событие ПриСозданииНаСервере, в котором получается остаток товара в том случае, если форма переоткрывается с основного сервера. Однако при стандартной обработке события ПриПереоткрытииСДругогоСервера это значение заменяется предыдущим полученным остатком, который мог измениться за это время.

В результате в форме товара, открытой онлайн, будут отображаться остаток товара и кнопка Остаток товара для обновления остатка (рис. 3.25).

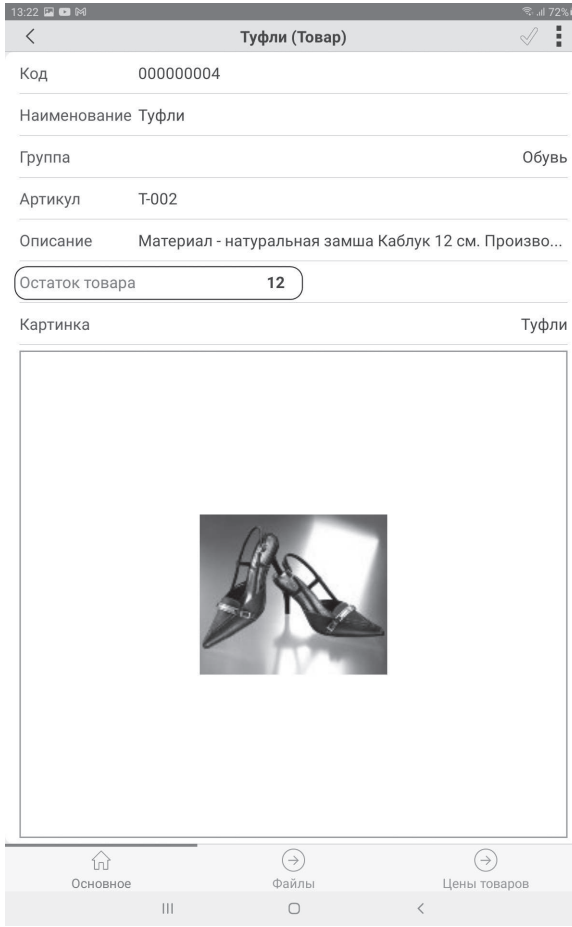


Рис. 3.25. Форма товара, открытая онлайн

Таким образом, если остаток товара изменился за время работы с формой, то при нажатии на кнопку Остаток товара можно актуализировать его при наличии соединения с основной информационной базой.

В случае потери соединения и переоткрытии формы автономно группа остатка будет скрыта от пользователя (рис. 3.26).

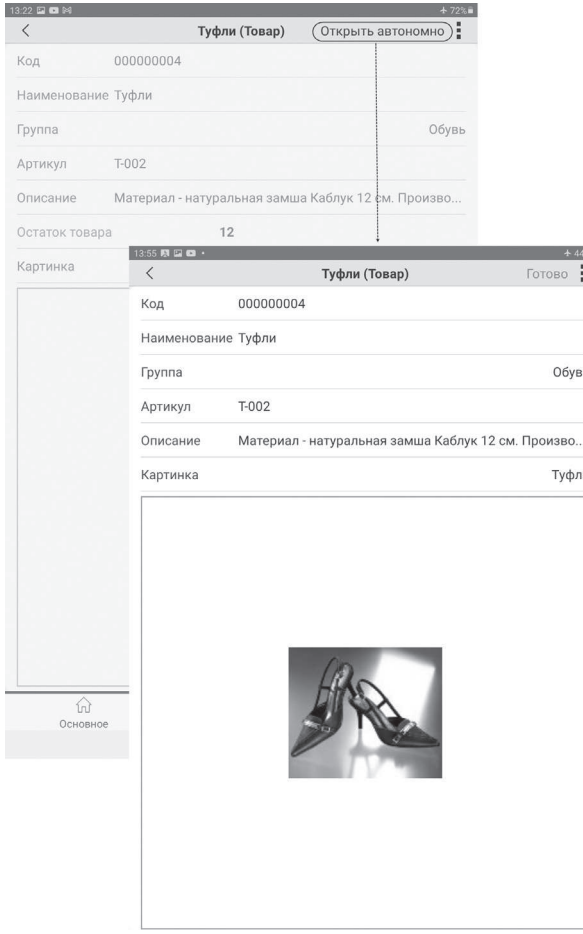


Рис. 3.26. Переоткрытие формы товара автономно

А при появлении соединения и переоткрытии формы товара онлайн снова будет показан актуальный остаток товара.

Формы начальной страницы

При включении объекта в состав автономной конфигурации стандартно в нее включаются и все его реквизиты и формы. Однако при определении состава автономной конфигурации можно убрать отметку с любого реквизита или формы объекта.

Например, вы не хотите включать в автономную конфигурацию форму списка справочника Клиенты (рис. 3.27).

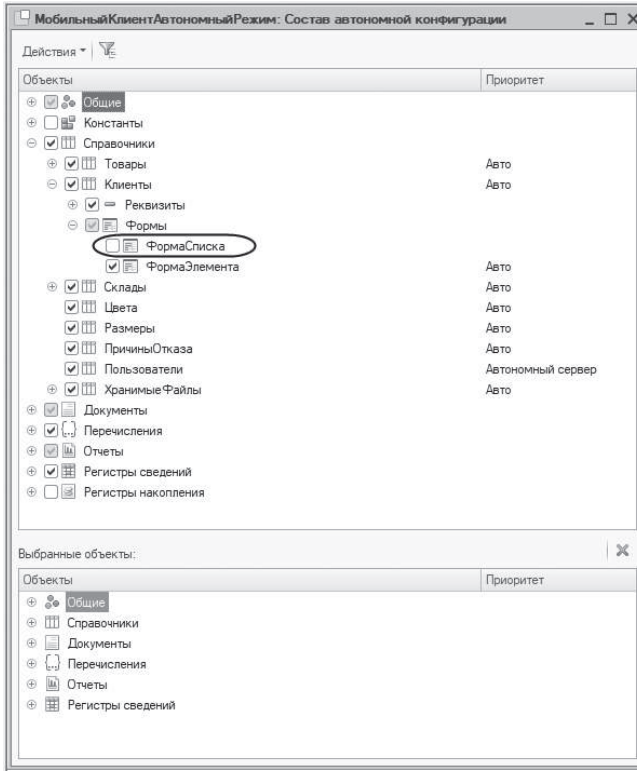


Рис. 3.27. Состав автономной конфигурации

В результате при наличии соединения с основной информационной базой форма списка клиентов на мобильном устройстве будет такой же, как и в офисном приложении, поскольку мобильное устройство подключено к офисной базе (рис. 3.28 вверху).

А в условиях потери соединения мобильное устройство будет подключено к автономной базе, в которой у справочника Клиенты нет формы. Поэтому форма списка клиентов будет автоматически сгенерирована платформой (то есть она будет содержать все реквизиты справочника), рис. 3.28 внизу.

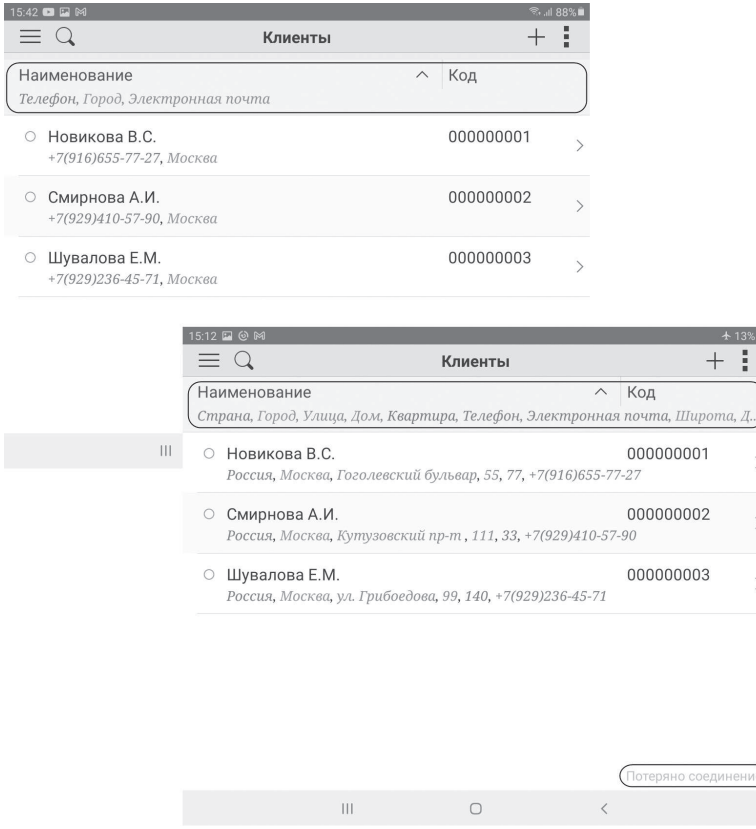


Рис. 3.28. Форма списка клиентов в условиях наличия/потери соединения

В случае если форма содержится на начальной странице приложения (рис. 3.29), внешний вид начальной страницы мобильного приложения будет зависеть от наличия интернет-соединения.

При наличии соединения с основным сервером форма списка клиентов будет располагаться на начальной странице мобильного приложения на отдельной закладке (рис. 3.30).

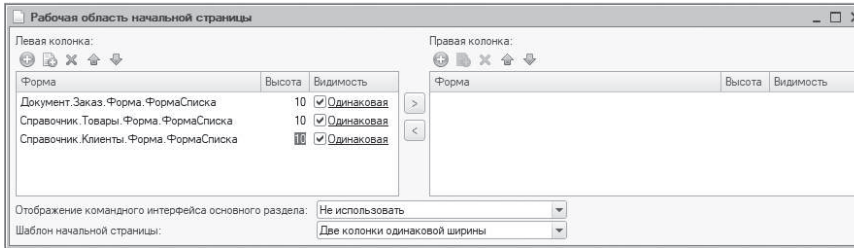


Рис. 3.29. Рабочая область начальной страницы в конфигураторе

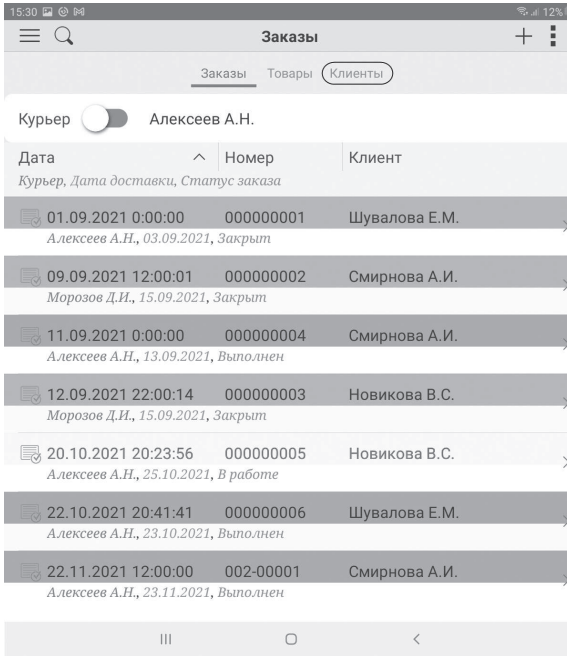


Рис. 3.30. Начальная страница мобильного приложения при наличии соединения

При потере соединения на начальной странице мобильного приложения формы списка клиентов не будет, так как она отсутствует в автономной конфигурации (рис. 3.31).

Поэтому рекомендуется первой формой на начальную страницу помещать ту форму, которая точно есть в автономной конфигурации и на которую у пользователя есть права.

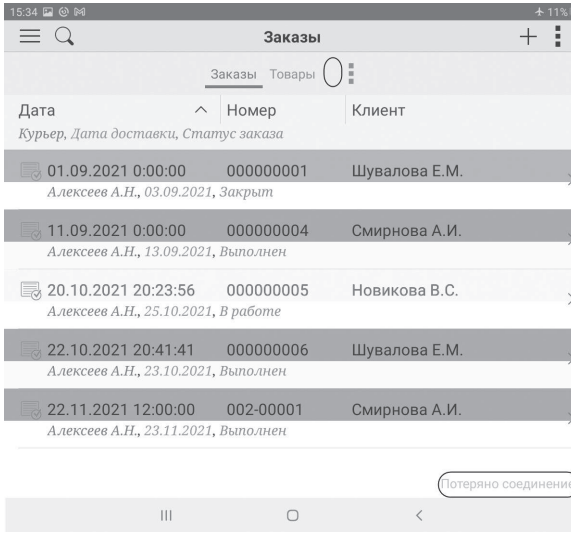


Рис. 3.31. Начальная страница мобильного приложения при потере соединения

Отчет на основе данных автономной конфигурации

Теперь посмотрите, как работают отчеты, построенные на основе данных, которые есть в автономной конфигурации.

Разработайте в конфигурации простой отчет ЗаказыКлиентов, получающий данные из документов Заказ на основе следующего запроса (листинг 3.25).

Листинг 3.25. Запрос для получения данных для отчета «ЗаказыКлиентов»

```
ВЫБРАТЬ
	ЗаказТовары.Товар КАК Товар,
	ЗаказТовары.Количество КАК Количество,
	ЗаказТовары.Сумма КАК Сумма,
	ЗаказТовары.Ссылка.Клиент КАК Клиент,
	ЗаказТовары.Ссылка.ДатаДоставки КАК ДатаДоставки,
	ЗаказТовары.Ссылка.Курьер КАК Курьер
ИЗ
	Документ.Заказ.Товары КАК ЗаказТовары
ГДЕ
	(ЗаказТовары.Ссылка.СтатусЗаказа = ЗНАЧЕНИЕ(Перечисление.СтатусыЗаказов.Выполнен) ИЛИ
		ЗаказТовары.Ссылка.СтатусЗаказа = ЗНАЧЕНИЕ(Перечисление.СтатусыЗаказов.Закрыт))
		И ЗаказТовары.Отказ = ЛОЖЬ
```

В ресурсы отчета добавьте поля Количество и Сумма.

В настройки отчета добавьте группировку по полю Клиент и вложенную в нее группировку по полю Товар, а в выбранные поля отчета добавьте поля ДатаДоставки, Количество и Сумма (рис. 3.32).

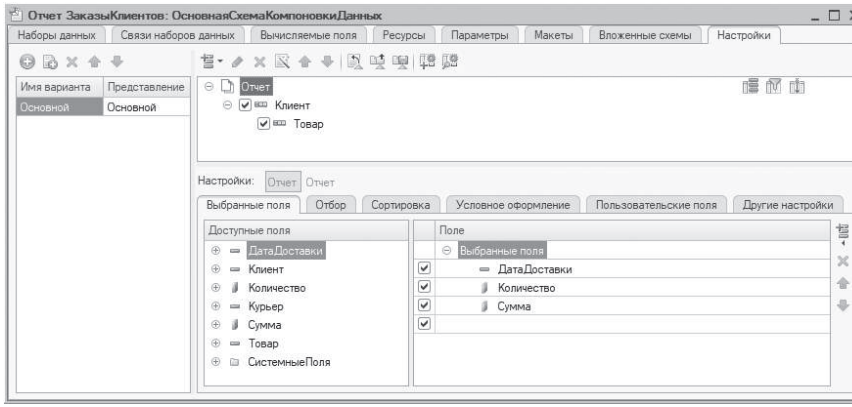


Рис. 3.32. Настройки отчета «ЗаказыКлиентов»

Создайте в отчете отбор по полю Курьер и добавьте эту настройку в состав быстрых пользовательских настроек (рис. 3.33).

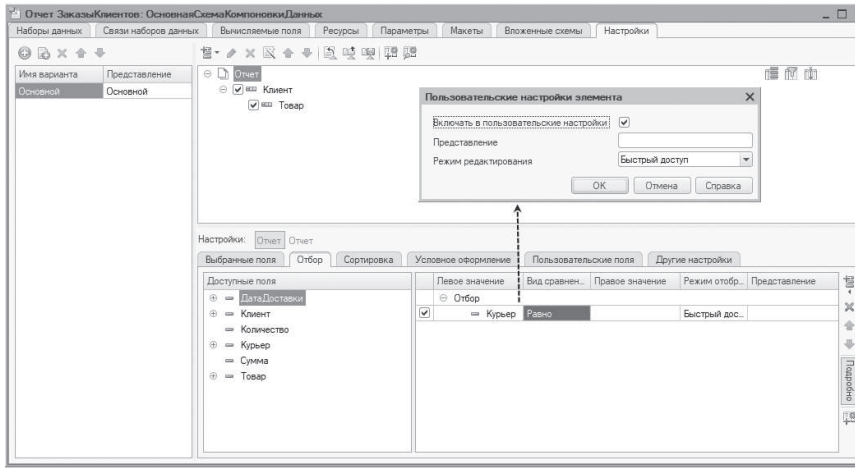


Рис. 3.33. Настройки отчета «ЗаказыКлиентов»

Затем включите отчет ЗаказыКлиентов в состав автономной конфигурации, так как он основан на данных документов Заказ, которые доступны в автономном режиме работы. Остальные отчеты недоступны автономно, так как они основаны на данных регистров накопления, которых нет в автономной конфигурации.

Найдите и запустите отчет из меню Главное мобильного приложения при отсутствии соединения с основным сервером (рис. 3.34).

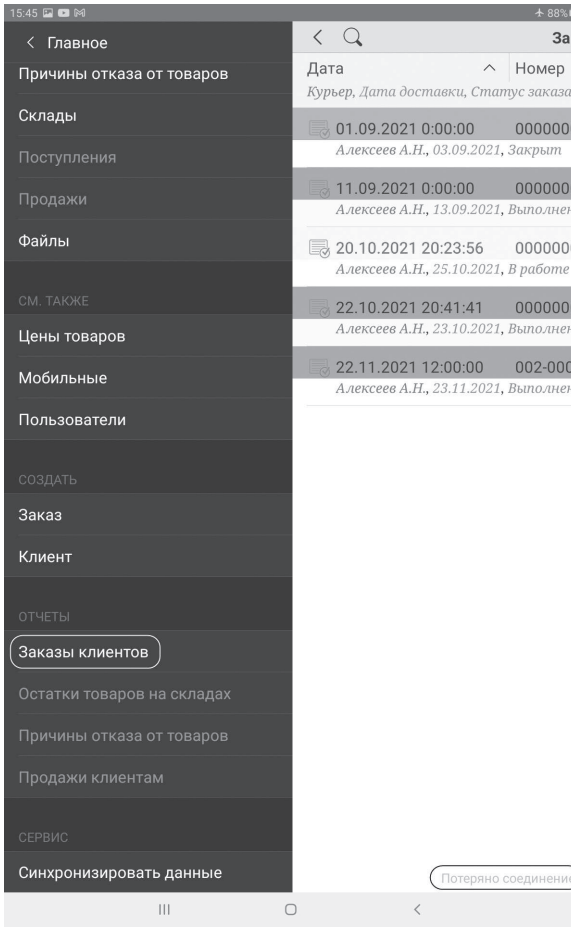


Рис. 3.34. Запуск отчета «Заказы клиентов» при потере соединения

Установите отбор по курьеру Алексеев А.Н. и сформируйте отчет. Затем отключите эту настройку и снова выполните отчет. Результат не изменится (рис. 3.35).

Так произошло потому, что в условиях потери соединения с основным сервером отчет получает данные из заказов, которые находятся в автономной информационной базе на мобильном устройстве. А при обмене данными из основной базы на планшет передаются только заказы для текущего курьера Алексеев А.Н. Поэтому других заказов в автономной базе просто нет.

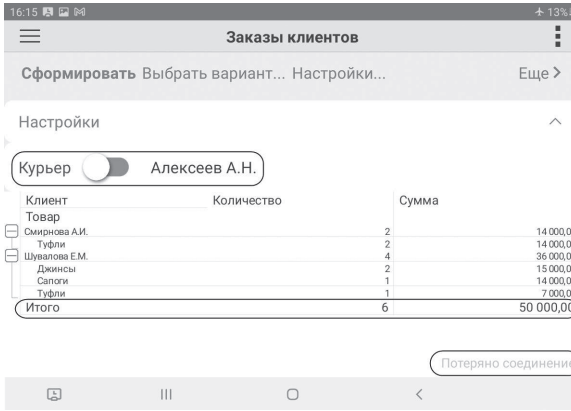


Рис. 3.35. Результат отчета «Заказы клиентов» при потере соединения

При появлении соединения вам будет предложено переоткрыть форму отчета онлайн. Теперь отчет будет содержать данные всех заказов, содержащихся в основной информационной базе (рис. 3.36 вверху), а при включении настройки отбора результат будет таким же, как если бы отчет выполнялся в автономной базе (рис. 3.36 внизу).

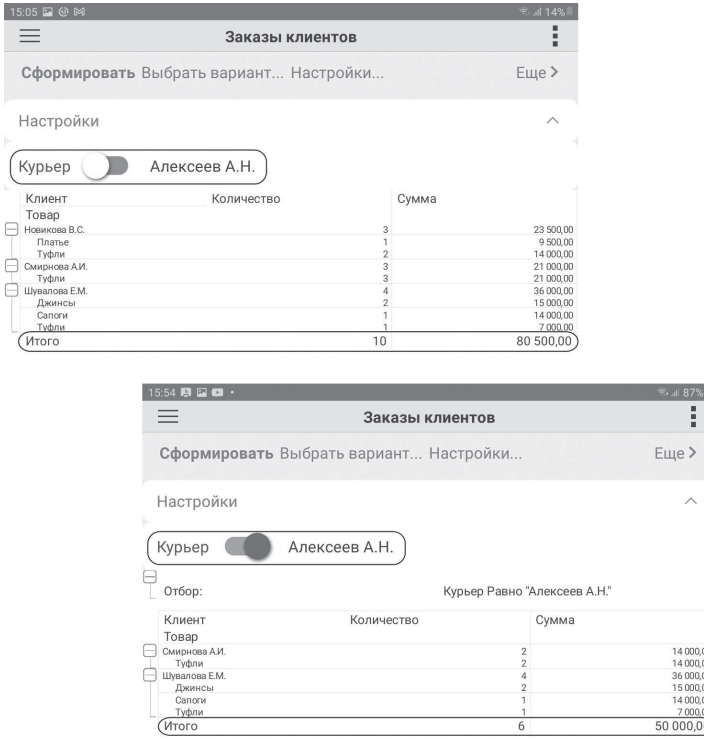


Рис. 3.36. Результат отчета «Заказы клиентов» при наличии соединения

Глава 4. Приложение мобильной платформы

В данной главе вы разработаете мобильное приложение, которое будет являться удаленным рабочим местом курьера интернет-магазина, работающего на планшете. Это приложение должно позволять курьеру решать все возникающие перед ним задачи, описанные в разделе «Функциональность мобильного приложения».

Все данные мобильного приложения будут храниться локально, на мобильном устройстве, но при этом приложение будет обмениваться данными с офисным приложением интернет-магазина (работающим на стационарном компьютере), то есть получать оттуда данные, изменять их и отсылать обратно.

ПОДРОБНЕЕ

Документация 1С:Предприятие 8.3.20. Руководство разработчика. Глава 29. Разработка для мобильных устройств > Приложение на мобильной платформе.

Начало разработки

Загрузите сделанную ранее копию офисной конфигурации и измените имя конфигурации на КурьерИнтернетМагазина. Затем установите свойство Назначение использования в значение Приложение для мобильной платформы. А также свойство Режим совместимости установите в значение 8.3.19 (рис. 4.1).

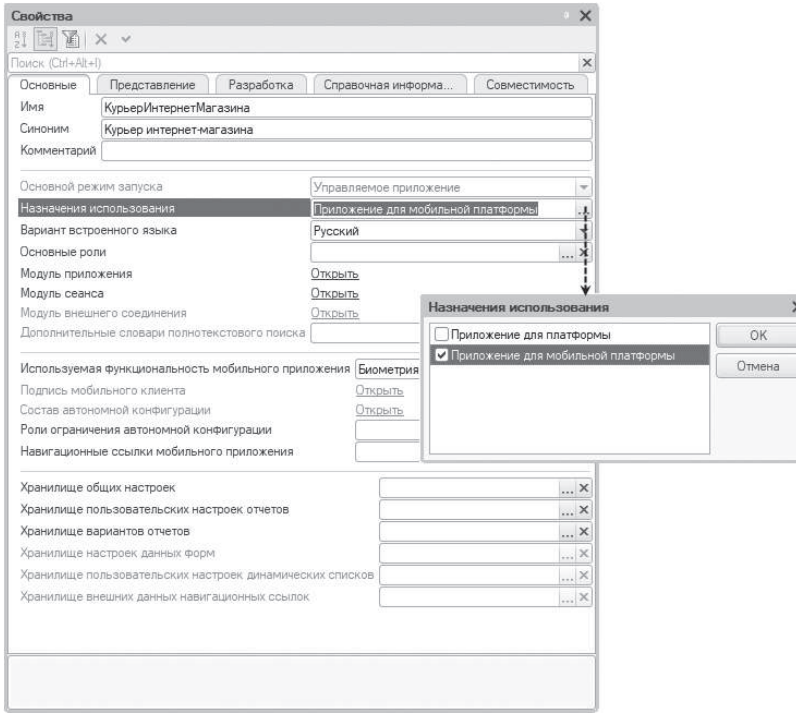


Рис. 4.1. Свойства конфигурации

Это значит, что конфигурация будет работать только на мобильном устройстве (в данном случае на планшете). Режим совместимости нужен потому, что версия платформы «1С:Предприятия» – 8.3.20, а версия мобильной платформы – 8.3.19.

Вообще говоря, конфигурация мобильного приложения, скорее всего, будет разрабатываться полностью с нуля, и она может быть вообще никак не связана с офисной конфигурацией. Но в данной книге для упрощения используется единая офисная конфигурация ИнтернетМагазин, и на ее основе разрабатываются все три вида приложений, рассматриваемых в книге.

Удалите из конфигурации КурьерИнтернетМагазина те объекты, которые не понадобятся курьеру для выполнения его задач:

- Документы:
  - ПриходнаяНакладная;
  - РасходнаяНакладная.
- Отчеты:
  - ОстаткиТоваровНаСкладах;
  - ПричиныОтказаОтТоваров;
  - ПродажиКлиентам.
- Регистры накопления:
  - ОстаткиТоваров;
  - Продажи;
  - ПричиныОтказа.

Затем войдите в конфигуратор с правами администратора и опубликуйте мобильное приложение на веб-сервере.

Для этого обновите конфигурацию базы данных (F7) и выполните команду конфигуратора Конфигурация > Мобильное приложение > Публиковать ...

В появившемся диалоге в поле Имя задайте имя виртуального каталога на веб-сервере, в который будет выполнена публикация мобильного приложения (например, MP).

В поле Каталог укажите физический каталог компьютера, в котором будет находиться файл публикации мобильного приложения (например, C:\Work\MP\).

Затем нажмите кнопку Опубликовать и подтвердите, что нужно обновить мобильное приложение сейчас (рис. 4.2).

Обратите внимание, что флажок Обновлять мобильное приложение при обновлении конфигурации базы данных стандартно установлен. Это значит, что при обновлении конфигурации базы данных будет автоматически обновляться публикация мобильного приложения на веб-сервере.

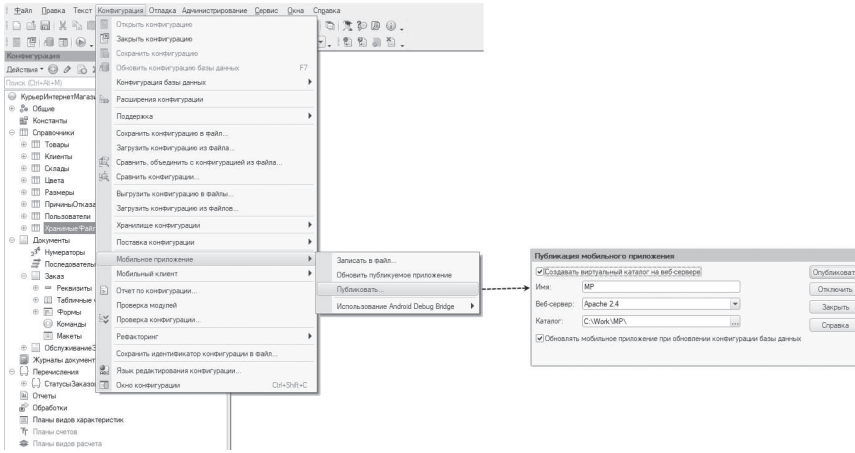


Рис. 4.2. Публикация мобильного приложения на веб-сервере

Добавление приложения на планшет

Для тестирования и отладки приложения мобильной платформы на планшете нужно, чтобы на нем была установлена мобильная платформа для разработки соответствующего типа приложений. О том, как это сделать, рассказывалось в первой главе в разделе «Установка мобильной платформы разработчика для мобильных устройств».

Найдите в списке приложений планшета мобильную платформу разработчика и запустите ее. Платформа откроет список своих приложений, который пока пуст.

Добавьте новое мобильное приложение, нажав на значок «+» в правом верхнем углу экрана в строке Приложения. В появившемся окне, в поле Адрес, укажите URL веб-сервера, на котором опубликовано мобильное приложение, и нажмите Загрузить в правом верхнем углу экрана (рис. 4.3).

После этого в качестве наименования приложения по умолчанию подставится синоним конфигурации. Это наименование можно потом изменить. Оно будет отображаться в списке приложений мобильной платформы разработчика. Включите переключатель Перезапуск из конфигуратора и нажмите ОК в правом верхнем углу экрана (рис. 4.4).

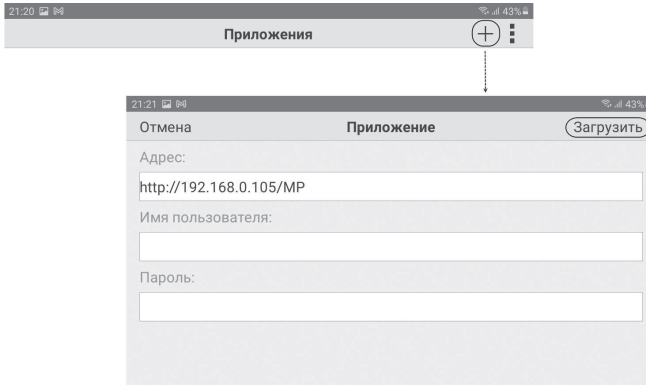


Рис. 4.3. Создание нового мобильного приложения

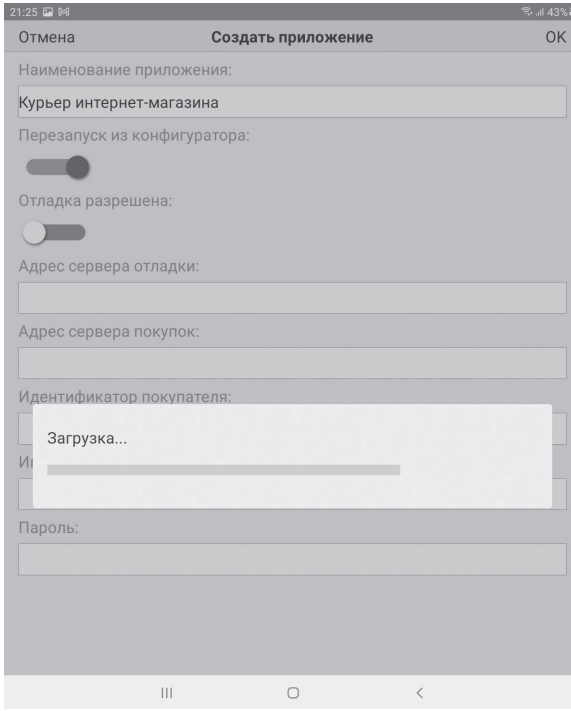


Рис. 4.4. Создание нового мобильного приложения

В результате мобильное приложение будет загружено с веб-сервера. При этом следует обратить внимание на состояние переключателя Перезапуск из конфигуратора. Если он установлен, то при открытии мобильного приложения платформа разработчика будет выполнять поиск новой версии мобильного приложения на веб-сервере и приложение на планшете будет обновлено. Стандартно этот переключатель не установлен.

После этого в списке мобильных приложений платформы разработчика появится созданное вами приложение, которое можно запустить кратковременным нажатием на его наименование. При длинном нажатии на эту строку появится контекстное меню, из которого можно изменить свойства мобильного приложения, выбрав пункт Изменить.

Запустите мобильное приложение Курьер интернет-магазина. При открытии мобильное приложение будет обновлено, так как переключатель Перезапуск из конфигуратора установлен.

На начальной странице приложения вы увидите команды для перехода к тем документам и справочникам, которые вы оставили в конфигурации мобильного приложения и которые будут нужны курьеру интернет-магазина в его повседневной работе (рис. 4.5).

Но если вы откроете любой справочник или документ, то увидите, что никаких данных там нет. Чтобы заполнить мобильное приложение данными из офисной базы, а также периодически синхронизировать данные между офисным и мобильным приложением, нужно реализовать в мобильном приложении блок обмена данными. В этом примере данные будут синхронизироваться через Web-сервис, опубликованный офисным приложением.

С помощью обмена в начале каждого рабочего дня курьер сможет получать все необходимые ему данные из интернет-магазина на свой планшет. Затем, при наличии соединения, он может синхронизировать измененные данные с офисным приложением.

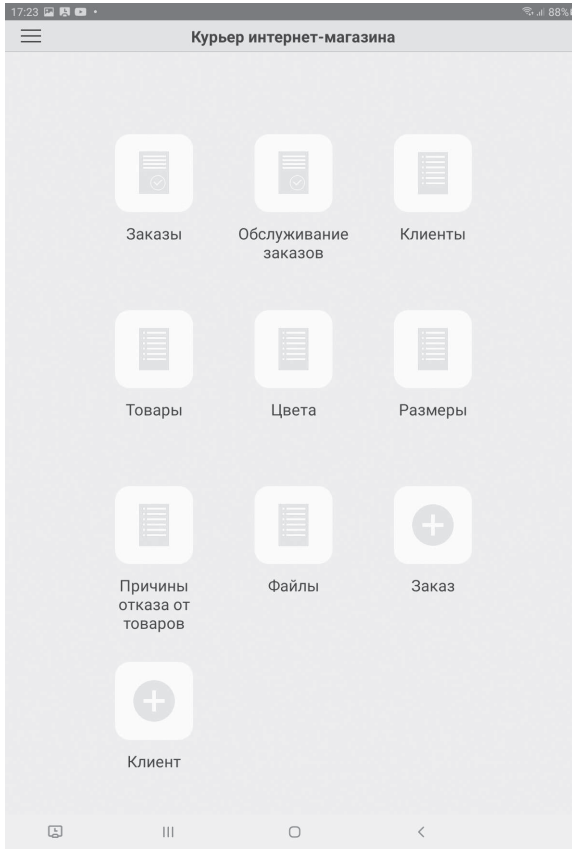


Рис. 4.5. Запуск мобильного приложения «Курьер интернет-магазина»

Обмен данными через Web-сервис

Для реализации обмена данными в данной главе будут использоваться возможности сервис-ориентированной архитектуры, с помощью которой система «1С:Предприятие» может экспортировать свою функциональность через Web-сервисы.

Синхронизация данных будет запускаться из мобильного приложения. При этом будет произведено обращение к Web-сервису, предоставляемому офисным приложением, и будут вызваны его соответствующие операции,

которые формируют/принимают данные обмена. По сути, это стандартные процедуры обмена, подробно рассмотренные в третьей главе при реализации обмена в мобильном клиенте с автономным режимом.

Итак, в офисной конфигурации существует WebСервис с именем MAExchange. Для обращения к сервису из мобильного приложения вам понадобится URI пространства имен, которому он принадлежит: http://localhost/wsExchange. А также имя файла публикации Web-сервиса на веб-сервере – wsExchange.1cws.

У Web-сервиса определены следующие операции для обмена данными:

- НачатьОбмен. Эта функция используется для инициализации обмена в офисной базе. Проверяется, есть ли нужный мобильный узел в плане обмена в офисной базе, если нет – то этот узел создается, для него регистрируются изменения в данных и т. п. Процедура инициализации обмена подробно рассматривалась в третьей главе в листинге 3.5.
- ПолучитьДанные. Эта функция используется для формирования пакета изменений, предназначенных для узла обмена, код которого передается в функцию. По завершении работы функция возвращает этому узлу данные обмена, помещенные в хранилище значения. Данные обмена формируются с помощью стандартной процедуры СформироватьПакетОбмена(), которая подробно рассматривалась в третьей главе в листинге 3.8.
- ЗаписатьДанные. Эта функция используется для записи пакета изменений, принятых от узла. В функцию передаются код узла обмена, от которого принимаются изменения, и пакет обмена, полученный от этого узла и помещенный в хранилище значения. Данные обмена записываются с помощью стандартной процедуры ПринятьПакетОбмена(), которая подробно рассматривалась в третьей главе в листинге 3.9.

ПОДРОБНЕЕ

Если вы хотите более подробно ознакомиться с устройством Web-сервиса MAExchange, можете скачать офисную конфигурацию ИнтернетМагазин с блоком обмена по адресу: https://its.1c.ru/bmk/mobile83. Распакуйте архив, создайте пустую информационную базу «1С:Предприятия» и в режиме Конфигуратор загрузите в нее информационную базу из файла «Интернет магазин с обменом.dt» (Администрирование – Загрузить информационную базу).

Реализация обмена данными

Для реализации обмена данными в мобильном приложении вам понадобится план обмена для хранения офисного и мобильного узлов обмена, а также для регистрации изменений данных в этих узлах. Затем для доступа

к Web-сервису вам потребуется константа, в которой будет храниться имя виртуального каталога на веб-сервере, в который выполнена публикация Web-сервиса, и команда, инициирующая обмен данными между мобильным и офисным приложением.

Итак, давайте начнем.

Добавьте в конфигурацию КурьерИнтернетМагазина план обмена Мобильные с синонимом Мобильные устройства и реквизитом Курьер, ссылающимся на справочник Пользователи.

В состав плана обмена включите все справочники, документы Заказ и ОбслуживаниеЗаказов и регистр сведений ЦеныТоваров (рис. 4.6).

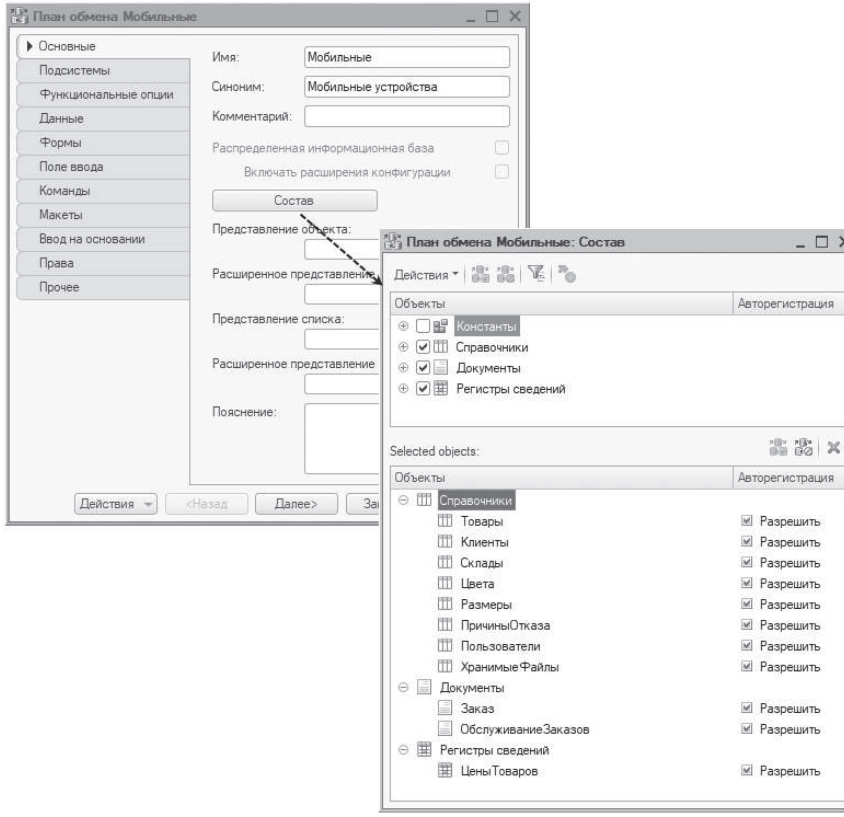


Рис. 4.6. Состав плана обмена «Мобильные»

Затем добавьте в конфигурацию константу АдресЦентральнойБазы типа Строка длиной 256 символов и снимите флажок у свойства константы Использовать стандартные команды.

Для редактирования и записи значения этой константы создайте общую форму констант с именем Настройки. В дальнейшем в эту форму будут добавляться и другие настройки, хранящиеся в константах. Обратите внимание: так как вы указали, что конфигурация будет использоваться только на мобильном устройстве, форма в окне предварительного просмотра имеет соответствующий вид (рис. 4.7).

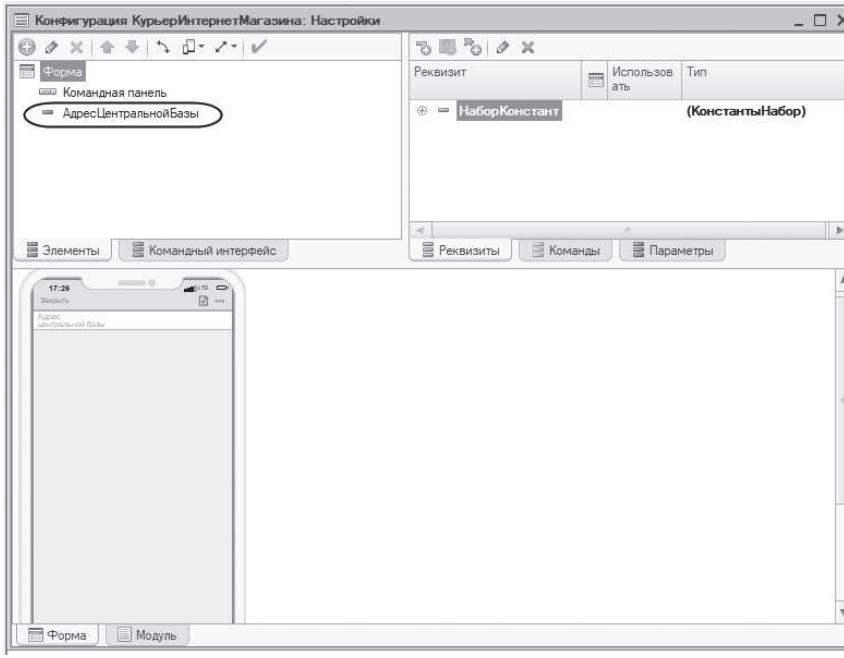


Рис. 4.7. Форма констант «Настройки»

Затем добавьте в конфигурацию общую команду ОбменСЦентральнойБазой с синонимом Синхронизировать данные. Укажите группу для отображения команды в интерфейсе приложения – Панель действий.Сервис. Таким образом, команда для выполнения синхронизации данных будет находиться в группе Сервис главного меню мобильного приложения.

Обработчик команды заполните следующим образом (листинг 4.1).

Листинг 4.1. Обработчик команды «ОбменСЦентральнойБазой»

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

	ТекстОшибки = "";
	Если Обмен.ВыполнитьОбменДанными(ТекстОшибки) Тогда
		ОповеститьОбИзменении(Тип("ДокументСсылка.Заказ"));
		ОповеститьОбИзменении(Тип("ДокументСсылка.ОбслуживаниеЗаказов"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.Склады"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.Цвета"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.Размеры"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.ПричиныОтказа"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.ХранимыеФайлы"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.Клиенты"));
		ОповеститьОбИзменении(Тип("СправочникСсылка.Товары"));
		ОбновитьИнтерфейс();
		ПоказатьОповещениеПользователя("Выполнена синхронизация данных");
	Иначе
		ПоказатьПредупреждение(, ТекстОшибки);
	КонецЕсли;

КонецПроцедуры
```

В обработчике команды для выполнения синхронизации данных с основным приложением вызывается функция ВыполнитьОбменДанными(), которую вы поместите в общем модуле Обмен (см. листинг 4.2).

Если эта функция возвращает значение Истина, то динамические списки документов и списки всех справочников в мобильном приложении оповещаются об изменении и обновляются. Пользователю показывается сообщение об окончании синхронизации данных. Если функция возвращает Ложь, то пользователю выводится текст ошибки, произошедшей при обмене данными.

Теперь создайте общий модуль Обмен с установленными флажками Сервер и Вызов сервера и поместите в него функцию ВыполнитьОбменДанными(), листинг 4.2.

Листинг 4.2. Функция «ВыполнитьОбменДанными()»

```
Функция ВыполнитьОбменДанными(ТекстОшибки) Экспорт

	Прокси = ПолучитьПрокси(ТекстОшибки);
	Если Прокси = Неопределено Тогда
		Возврат Ложь;
	КонецЕсли;

	НаименованиеУстройства = "Мобильный автономный узел";

	ЦентральныйУзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду("001");
	Если ЦентральныйУзелОбмена.Пустая() Тогда
		НовыйУзел = ПланыОбмена.Мобильные.СоздатьУзел();
```

```
            НовыйУзел.Код="001";
            НовыйУзел.Наименование="Центральный";
            НовыйУзел.Записать();
            ЦентральныйУзелОбмена = НовыйУзел.Ссылка;
        КонецЕсли;

        Узел = ПланыОбмена.Мобильные.ЭтотУзел();

        // Инициализируем обмен, проверяем, есть ли нужный узел в плане обмена.
        НовыйКод = Прокси.НачатьОбмен(Узел.Код, НаименованиеУстройства, ЦентральныйУзелОбмена.
                                        НомерПринятого, ЦентральныйУзелОбмена.НомерОтправленного);

        Если  Узел.Код <> НовыйКод Тогда
            ОбъектУзла = Узел.ПолучитьОбъект();
            ОбъектУзла.Код = НовыйКод;
            ОбъектУзла.Наименование = НаименованиеУстройства;
            ОбъектУзла.Записать();
        КонецЕсли;

        // Отправляем данные
        ДанныеОбмена = Обмен.СформироватьПакетОбмена(ЦентральныйУзелОбмена);
        Прокси.ЗаписатьДанные(Узел.Код, ДанныеОбмена);

        // Принимаем данные
        ДанныеОбмена = Прокси.ПолучитьДанные(Узел.Код);
        Обмен.ПринятьПакетОбмена(ЦентральныйУзелОбмена, ДанныеОбмена);

        Возврат Истина;

КонецФункции
```

Прежде всего вы вызываете функцию ПолучитьПрокси(), в которой получается описание Web-сервиса на основе динамической ссылки и происходит подключение к Web-сервису (см. листинги 4.3, 4.4).

В случае успешного подключения к Web-сервису с помощью объекта Прокси вы получаете доступ к операциям Web-сервиса, определенным в офисном приложении.

Сначала вы задаете наименование узла обмена на мобильном устройстве (далее – «мобильного узла обмена»).

Затем проверяете, есть ли в списке узлов плана обмена на мобильном устройстве узел, соответствующий офисной базе, с кодом «001». Если нет (обмен еще ни разу не выполнялся) – создаете его.

После этого с помощью функции ЭтотУзел() получаете ссылку на предопределенный узел обмена на мобильном устройстве и передаете код этого узла в операцию Web-сервиса НачатьОбмен для инициализации обмена в офисной базе.

Кроме того, в операцию НачатьОбмен передаются наименование мобильного узла обмена и номера сообщений, принятых/отправленных центральным узлом обмена. При выполнении этой операции инициализируется обмен

между основным и мобильным приложением, в случае необходимости синхронизируются номера принятых и отправленных сообщений в узлах обмена и регистрируются изменения центральной базы для мобильного узла обмена.

Затем вы выполняете обмен данными с мобильным приложением в обе стороны.

Сначала вы вызываете функцию СформироватьПакетОбмена() для формирования в мобильном приложении пакета изменений, предназначенных для центрального узла обмена. Эту функцию вы поместите в общий модуль Обмен (см. листинг 4.5).

Данные обмена, помещенные в хранилище значения и возвращенные этой функцией, и код мобильного узла обмена вы передаете в операцию Web-сервиса ЗаписатьДанные. При выполнении этой операции изменения, принятые от мобильного узла обмена, записываются в центральную базу данных.

Таким образом вы передаете данные от мобильного приложения в центральную базу.

Затем вы вызываете операцию Web-сервиса ПолучитьДанные и передаете в нее код мобильного узла обмена. При выполнении этой операции в центральной базе формируется пакет изменений, предназначенных для мобильного узла обмена. В результате в мобильное приложение возвращаются данные обмена, помещенные в хранилище значения.

После этого вы вызываете процедуру ПринятьПакетОбмена() для записи изменений, принятых от центрального узла обмена, в базу мобильного приложения. Эту функцию вы поместите в общий модуль Обмен (см. листинг 4.6).

Таким образом вы передаете данные от центральной базы в мобильное приложение.

Теперь пришло время создать в модуле Обмен и подробно рассмотреть все описанные выше процедуры и функции.

Листинг 4.3. Функция «ПолучитьПрокси()»

```
Функция ПолучитьПрокси(ТекстОшибки) Экспорт

    ТекстОшибки = "";

    Адрес = Константы.АдресЦентральнойБазы.Получить();
    Адрес = Адрес + "/ws/wsExchange.1cws?wsdl";

    Попытка
        Определения = Новый WSОпределения(Адрес);
```

```
    Исключение
        // Сообщим пользователю о том, что не получилось получить определение сервиса
        ТекстОшибки = "Не удалось установить соединение с сервером. Повторите попытку позже.";
        Возврат Неопределено;
    КонецПопытки;

    URI = URIПространстваИменСервиса();
    Прокси = Новый WSПрокси(Определения, URI, "MAExchange", "MAExchangeSoap");

    Возврат Прокси;

КонецФункции
```

В данной функции создается объект WSОпределения, который получает определение Web-сервисов из WSDL-файла. Местоположение WSDL-файла складывается из каталога веб-сервера, на котором опубликован Web-сервис (значение константы АдресЦентральнойБазы), и имени файла публикации.

На основе определения Web-сервиса и его свойства URI пространства имен создается объект WSПрокси и связывает его с точкой подключения Web-сервиса.

Листинг 4.4. Функция для получения URI пространства имен Web-сервиса

```
Функция URIПространстваИменСервиса()

    Возврат "http://localhost/wsExchange";

КонецФункции
```

В случае успешного подключения к Web-сервису функция ПолучитьПрокси() возвращает объект WSПрокси, иначе – функция возвращает значение Неопределено и заполняет параметр ТекстОшибки.

Функция СформироватьПакетОбмена() из общего модуля Обмен (листинг 4.5) – это стандартная процедура обмена. Она подробно рассматривалась в третьей главе в листинге 3.8. Отличие состоит в том, что в мобильном приложении не используется привилегированный режим. Поэтому в функции отсутствует строка УстановитьПривилегированныйРежим (Истина).

Листинг 4.5. Функция «СформироватьПакетОбмена()»

```
Функция СформироватьПакетОбмена(УзелОбмена) Экспорт

    ЗаписьXML = Новый ЗаписьXML;

    ЗаписьXML.УстановитьСтроку("UTF-8");
    ЗаписьXML.ЗаписатьОбъявлениеXML();
```

```
    ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
    ЗаписьСообщения.НачатьЗапись(ЗаписьXML, УзелОбмена);

    ЗаписьXML.ЗаписатьСоответствиеПространстваИмен(
                                        "xsi", "http://www.w3.org/2001/XMLSchema-instance");
    ЗаписьXML.ЗаписатьСоответствиеПространстваИмен("v8", "http://v8.1c.ru/data");

    ВыборкаИзменений = ПланыОбмена.ВыбратьИзменения(
                                        УзелОбмена, ЗаписьСообщения.НомерСообщения);
    Пока ВыборкаИзменений.Следующий() Цикл

        Данные = ВыборкаИзменений.Получить();

        // Записываем данные в сообщение
        ЗаписатьXML(ЗаписьXML, Данные);

    КонецЦикла;

    ЗаписьСообщения.ЗакончитьЗапись();

    Возврат Новый ХранилищеЗначения(ЗаписьXML.Закрыть(), Новый СжатиеДанных(9));

КонецФункции
```

Эта функция практически аналогична одноименной функции офисного приложения, которая не приводится в книге. Единственное отличие состоит в том, что в мобильном приложении данные не анализируются на предмет переноса. В пакет изменений для центральной базы записываются все изменения, полученные от планшета, тогда как из центральной базы будут переноситься только те документы, у которых курьер совпадает с реквизитом узла обмена (если этот реквизит заполнен).

Процедура ПринятьПакетОбмена() из общего модуля Обмен (листинг 4.6) – это стандартная процедура обмена. Она подробно рассматривалась в третьей главе в листинге 3.9. Отличие состоит в том, что в мобильном приложении не используется привилегированный режим. Поэтому в процедуре отсутствует строка УстановитьПривилегированныйРежим(Истина).

Листинг 4.6. Процедура «ПринятьПакетОбмена()»

```
Процедура ПринятьПакетОбмена(УзелОбмена, ДанныеОбмена) Экспорт

    ЧтениеXML = Новый ЧтениеXML;
    ЧтениеXML.УстановитьСтроку(ДанныеОбмена.Получить());
    ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
    ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

    ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
                                                ЧтениеСообщения.НомерПринятого);

    НачатьТранзакцию();
```

```
    Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

        Данные = ПрочитатьXML(ЧтениеXML);

        Если Не Данные = Неопределено Тогда
            // Не переносим изменение, полученное из офиса, если есть регистрация изменения на
            // планшете
            Если Не ПринятьИзменения(ЧтениеСообщения.Отправитель, Данные) Тогда
                Продолжить;
            КонецЕсли;

            Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
            Данные.ОбменДанными.Загрузка = Истина;

            Данные.Записать();
        КонецЕсли;

    КонецЦикла;
    ЗафиксироватьТранзакцию();

    ЧтениеСообщения.ЗакончитьЧтение();
    ЧтениеXML.Закрыть();

КонецПроцедуры
```

Эта процедура практически аналогична одноименной процедуре офисного приложения, которая не приводится в книге. Единственное отличие состоит в вопросе разрешения коллизий при приеме изменений. В мобильном приложении приоритет имеют изменения заказов (см. функцию ПринятьИзменения() в листинге 4.7), в то время как в офисе более приоритетными являются изменения справочников.

Листинг 4.7. Функция для разрешения коллизии при обмене данными

```
Функция ПринятьИзменения(Отправитель, Данные) Экспорт

    Прием = Истина;
    Если ПланыОбмена.ИзменениеЗарегистрировано(Отправитель, Данные) Тогда

        Если ТипЗнч(Данные) = Тип("ДокументОбъект.Заказ") Тогда
            Прием = Ложь;
        КонецЕсли;

    КонецЕсли;

    Возврат Прием;

КонецФункции
```

В этой функции проверяется, зарегистрировано ли изменение объекта в узле обмена (на планшете), для которого отправлены данные. Если да и если тип объекта ДокументОбъект.Заказ, функция возвращает Ложь, во всех остальных случаях возвращается Истина.

Таким образом, в функции ПринятьПакетОбмена() отклоняются изменения документа Заказ, полученные из офиса, если этот объект уже изменялся на планшете. Во всех остальных случаях принимаются все изменения.

В заключение вы должны обеспечить уникальность номеров и кодов при создании новых объектов, которые могут добавляться и в мобильном, и в офисном приложении. Дело в том, что курьер может в процессе работы на планшете вводить новые заказы, новых клиентов и причины отказа от товаров.

Поэтому сделайте так, чтобы при вводе этих объектов на мобильном устройстве их номер или код начинался бы с префикса – кода узла обмена.

Для этого в модуле документа Заказ создайте процедуру – обработчик события ПриУстановкеНовогоНомера() и заполните следующим образом (листинг 4.8).

Листинг 4.8. Процедура «ПриУстановкеНовогоНомера()»

```
Процедура ПриУстановкеНовогоНомера(СтандартнаяОбработка, Префикс)

    Префикс = ПланыОбмена.Мобильные.ЭтотУзел().Код + "-";

КонецПроцедуры
```

Событие ПриУстановкеНовогоНомера возникает в момент, когда выполняется установка нового номера документа. Вторым параметром вызова обработчика передается префикс, который будет заполнен в данной процедуре и использован системой для генерации кода. Параметру Префикс присваивается строковый код предопределенного узла базы на мобильном устройстве.

По аналогии в модули объектов справочника Клиенты, справочника ПричиныОтказа и справочника ХранимыеФайлы добавьте обработчик события ПриУстановкеНовогоКода(), листинг 4.9.

Листинг 4.9. Процедура «ПриУстановкеНовогоКода()»

```
Процедура ПриУстановкеНовогоКода(СтандартнаяОбработка, Префикс)

    Префикс = ПланыОбмена.Мобильные.ЭтотУзел().Код + "-";

КонецПроцедуры
```

Таким образом, в офисном приложении будут создаваться все объекты обычным образом, без префиксов номеров или кодов. А заказы, клиенты, причины отказа и хранимые файлы, созданные в мобильном приложении, будут иметь номера/коды с префиксом «002-», «003-» и т. д.

Тестирование обмена данными

Проверьте, как работает обмен данными между офисным и мобильным приложениями. Обновите конфигурацию базы данных (F7), при этом автоматически обновится версия мобильного приложения на веб-сервере.

Запустите мобильное приложение на планшете и нажмите Настройки. В форме настроек приложения вам нужно задать адрес для доступа к Web-сервису, опубликованному офисным приложением. Установите значение константы АдресЦентральнойБазы – http://<IP-адрес компьютера в беспроводной сети>/<Каталог веб-сервера, на котором опубликован Web-сервис> (в данном случае – …/wsExchange) и нажмите Готово в правом верхнем углу формы настроек (рис. 4.8).

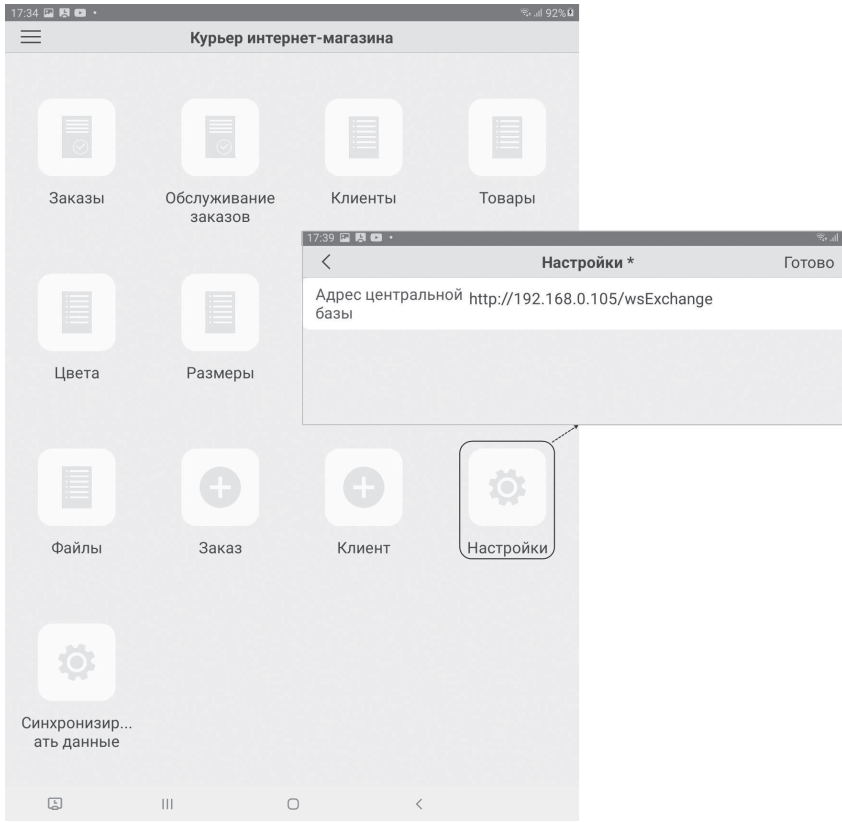


Рис. 4.8. Форма констант «Настройки»

Убедитесь, что на планшете есть интернет-соединение, и выполните команду Синхронизировать данные (ее можно вызвать из списка команд на рабочем столе или из главного меню приложения), рис. 4.9.

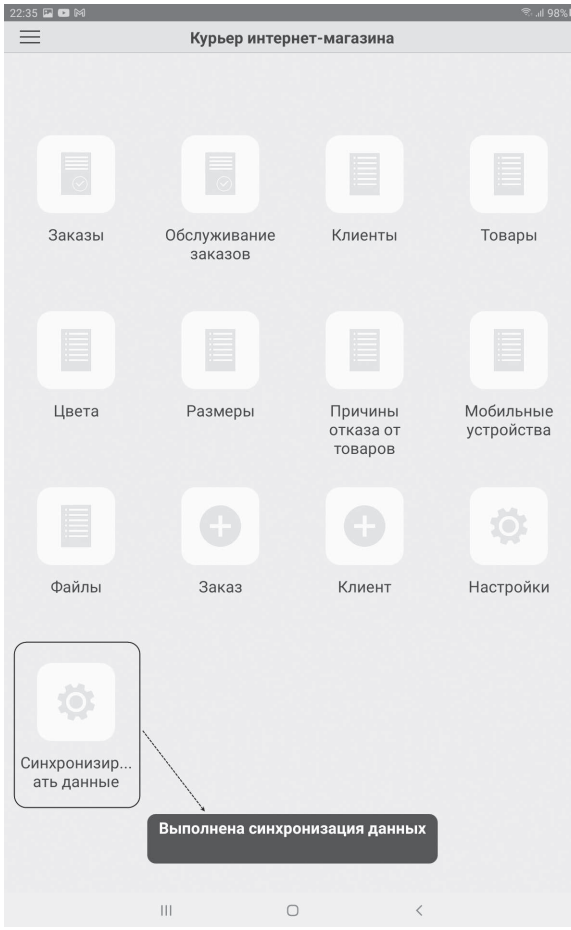


Рис. 4.9. Вызов команды «Синхронизировать данные»

В результате вы увидите сообщение о завершении синхронизации данных. При этом все динамические списки (списки заказов, товаров, клиентов и т. п.) автоматически обновятся с учетом полученных данных.

В случае ошибки при соединении с веб-сервером будет выведено сообщение с ошибкой.

После этого можно открыть любой документ или справочник мобильного приложения и увидеть, что все данные перенесены из офисного приложения (рис. 4.10).

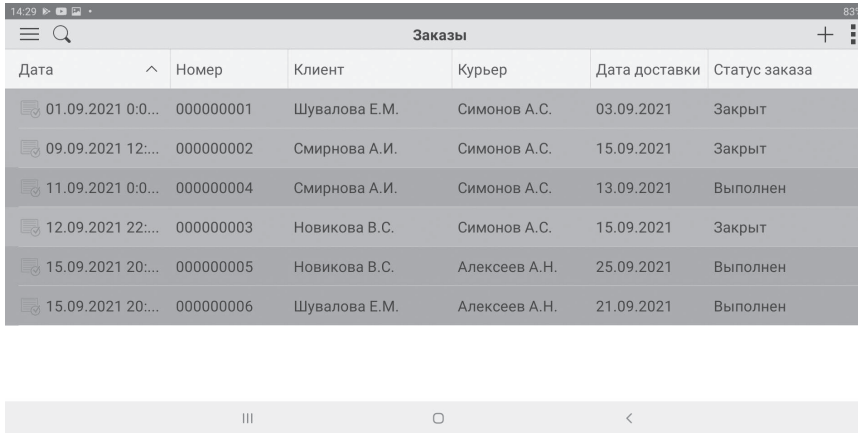


Рис. 4.10. Список заказов, заполненный из офисного приложения

Но, как вы видите, при первоначальном заполнении базы мобильного приложения переносятся все заказы, а также списки обслуживания заказов, которые есть в офисном приложении. Однако каждый курьер, работающий с приложением, должен видеть и редактировать только свои заказы.

Для этого при формировании пакета изменений для мобильного узла обмена в офисной базе реализована соответствующая стратегия распространения данных. Чтобы она работала, нужно в списке узлов плана обмена офисного приложения, в реквизите Курьер, указать, кто из пользователей работает с мобильным приложением (рис. 4.11).

После этого нужно перепровести все документы Заказ и Обслуживание заказов, чтобы для них снова зарегистрировались изменения в плане обмена, и выполнить обмен данными еще раз. В результате на планшет попадут только заказы и списки обслуживания заказов, относящиеся к указанному выше курьеру (Симонов А.С.), рис. 4.12.

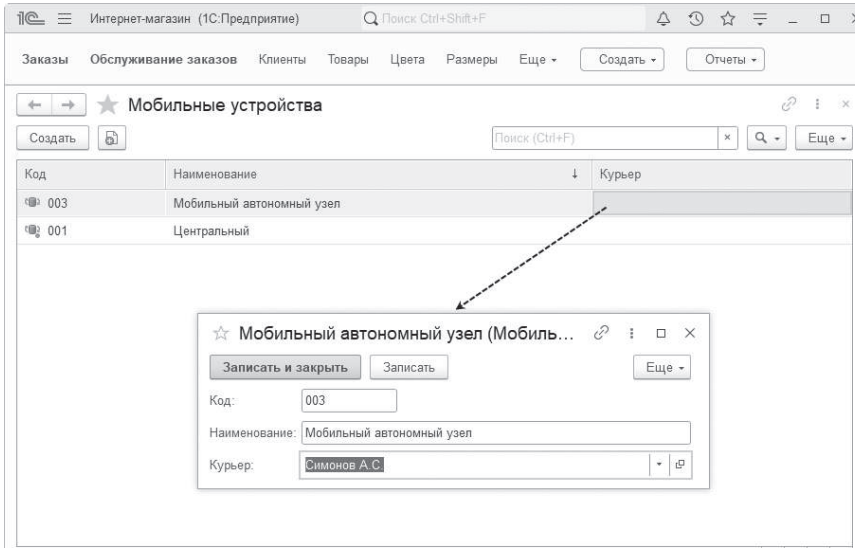


Рис. 4.11. Установка курьера для мобильного узла плана обмена в офисном приложении

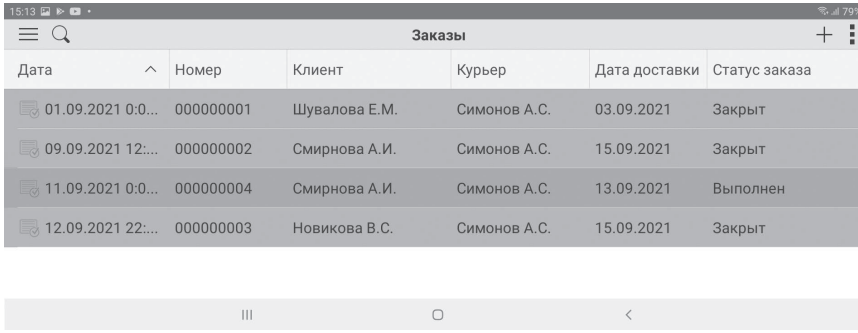


Рис. 4.12. Список заказов, заполненный из офисного приложения

ПРИМЕЧАНИЕ

Приложение мобильной платформы не поддерживает работу с пользователями. Поэтому, если на одном мобильном устройстве с одной и той же офисной информационной базой работают несколько пользователей, для каждого пользователя необходимо создать собственное приложение в списке приложений мобильной платформы.

Внесите теперь какие-то изменения в базу мобильного приложения. Например, откройте список клиентов и измените адрес одного из них. При следующей синхронизации данных ваши изменения будут перенесены в офисную базу.

Теперь проверьте, как происходит обмен новыми данными и данными, измененными сразу в обеих базах данных.

Введите на мобильном устройстве новый заказ и новую причину отказа. Затем выполните обмен данными – в офисной базе появятся заказ с номером «003-00001» и причина отказа с кодом «003-00001».

Теперь измените какой-нибудь заказ, а также данные какого-нибудь клиента в офисной базе. На мобильном устройстве также измените этот же заказ и этого же клиента. После синхронизации данных в мобильном приложении останутся изменения заказа и появятся изменения клиента, сделанные в офисном приложении. В офисной же базе, наоборот, останутся изменения клиента и будут приняты изменения заказа, сделанные в мобильном приложении.

Таким образом, коллизия при приеме данных решается правильно – изменения клиентов, причин отказа и хранимых файлов в офисном приложении имеют приоритет над мобильным устройством, а изменения заказов на мобильном устройстве имеет приоритет над офисным приложением.

Функциональность мобильного приложения

Доработка командного интерфейса

Прежде всего сделайте интерфейс вашего мобильного приложения более лаконичным и интуитивно понятным.

Ограничьте состав командного интерфейса, чтобы пользователь мог лучше и быстрее ориентироваться в главном меню приложения. Как правило, ему не нужны команды для открытия списков всех объектов, которые присутствуют в мобильном приложении.

В процессе своей работы курьер может изменять и создавать новые заказы, редактировать информацию о клиентах, создавать новых клиентов, изменять и создавать новые причины отказа от товаров и медиафайлы, относящиеся к клиентам. Также курьер должен иметь возможность просматривать информацию о товарах и список заказов для обслуживания без возможности их изменения.

То есть курьеру нужно предоставить возможность работать со следующими объектами мобильного приложения:

- справочники: Клиенты, Товары (только просмотр), Причины отказа от товаров;
- документы: Заказы, Обслуживание заказов (только просмотр).

Остальные справочники (Склады, Пользователи, Цвета, Размеры) нужны только для выбора из них значения в соответствующих ссылочных полях.

А справочник Хранимые файлы он может редактировать и просматривать только как подчиненный объект из справочника клиентов и товаров.

Откройте конфигурацию КурьерИнтернетМагазина и снимите флажок Использовать стандартные команды в свойствах справочников Склады, Пользователи, Цвета, Размеры. Таким образом, никакие команды, связанные с этими объектами, не попадут в соответствующие фрагменты командного интерфейса.

Затем откройте окно настроек Командный интерфейс основного раздела и перенесите справочник Хранимые файлы и план обмена Мобильные в группу команд Панель навигации.См. также (рис. 4.13).

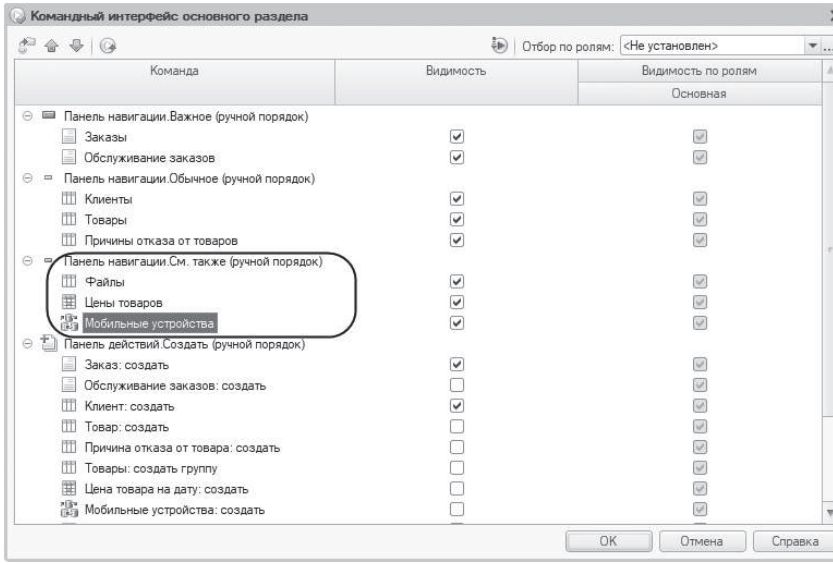


Рис. 4.13. Настройка командного интерфейса мобильного приложения

Таким образом, команды для открытия справочника хранимых файлов и списка узлов плана обмена будут доступны только из главного меню приложения и не видны на начальной странице (см. рис. 4.15).

Обновите конфигурацию базы данных (F7) и запустите мобильное приложение на планшете. На начальной странице и в главном меню приложения вы увидите теперь ограниченный набор команд интерфейса, в котором курьеру будет проще ориентироваться (рис. 4.14, 4.15).

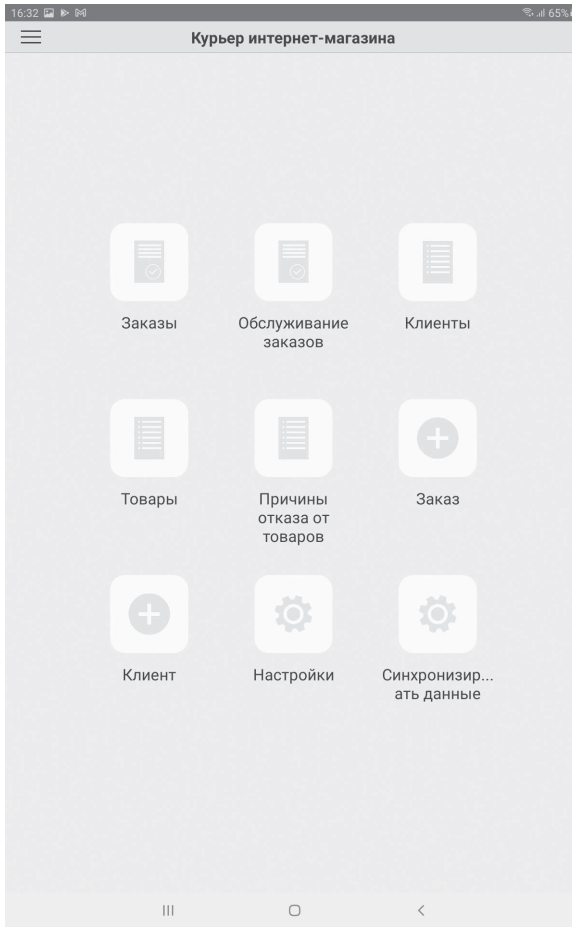


Рис. 4.14. Начальная страница мобильного приложения

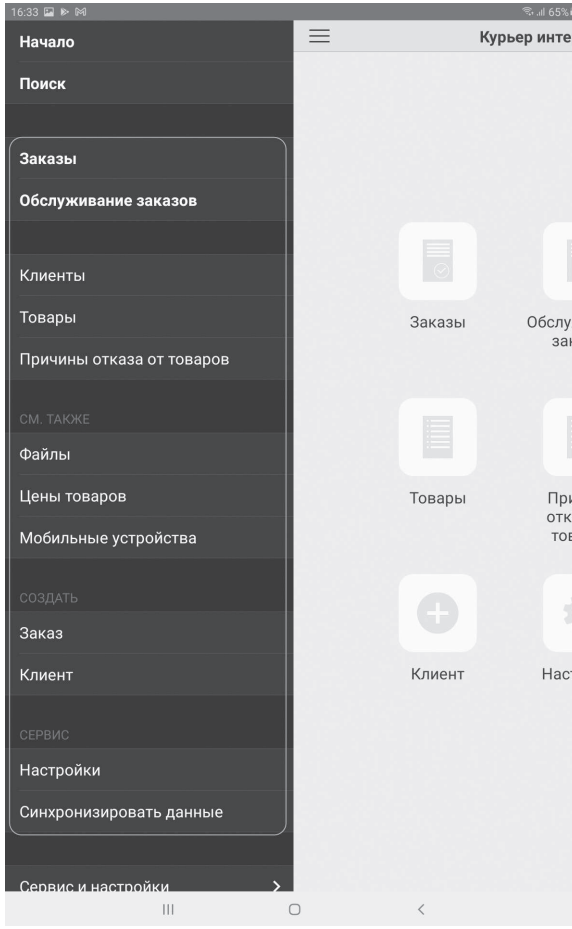


Рис. 4.15. Главное меню мобильного приложения

Помимо команд открытия различных списков вы видите здесь также команды для создания новых заказов и клиентов. Эти команды унаследованы из офисной конфигурации, и они понадобятся курьеру в его работе.

Ограничения прав доступа

Для того чтобы ограничить возможность изменения данных в соответствии с вышеописанными задачами, вам нужно в свойстве мобильного приложения Основные роли указать роль, которая будет использоваться для ограничения доступа к данным, когда пользователь в мобильном приложении отсутствует (как в данном примере).

Но сначала нужно создать эту роль и установить для нее нужные права. Для этого добавьте в конфигурацию роль с именем Основная. Для корня приложения установите все права и установите флажок Устанавливать права для новых объектов (рис. 4.16).

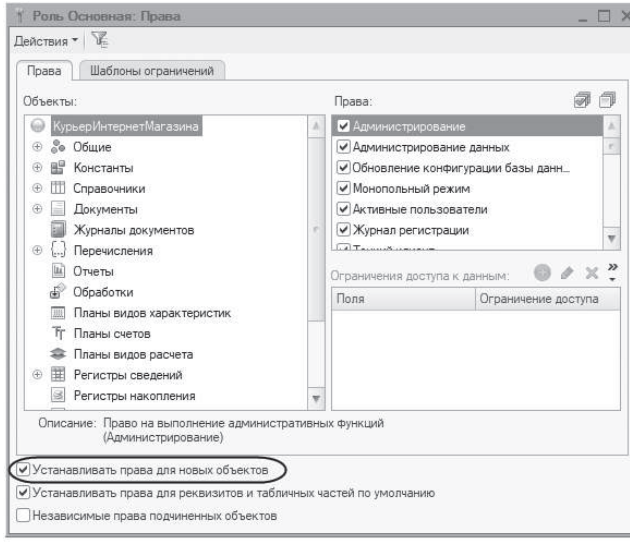


Рис. 4.16. Установка прав для роли «Основная»

Затем выделите ветвь объектов Справочники и отметьте права Чтение, Добавление, Изменение, Удаление, Просмотр и Ввод по строке (рис. 4.17).

Эти права будут даны сразу всем справочникам. Для справочников Клиенты, Причины отказа и Хранимые файлы, которые курьер может редактировать, установите все права, кроме права Интерактивное удаление (рис. 4.18).

Для документа Заказ также установите все права, кроме права Интерактивное удаление, а для документа Обслуживание заказов – только права Ввод по строке, Чтение и Просмотр.

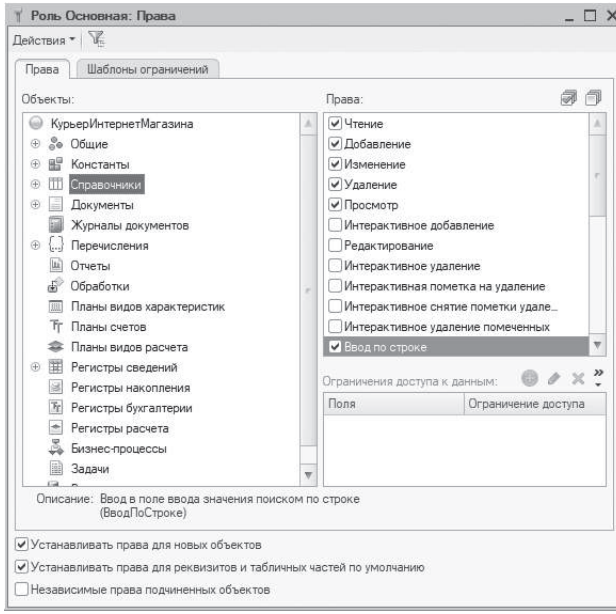


Рис. 4.17. Установка прав для роли «Основная»

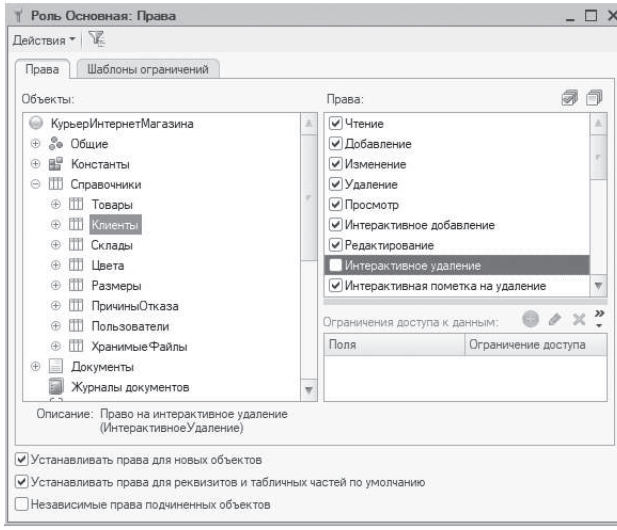


Рис. 4.18. Установка прав для роли «Основная»

Для регистра сведений Цены товаров установите права Чтение и Просмотр.

Для плана обмена Мобильные установите права Чтение, Добавление, Изменение, Удаление и Просмотр.

Для общей команды ОбменСЦентральнойБазой и общей формы Настройки установите право Просмотр. А для константы АдресЦентральнойБазы установите все права.

После этого выберите эту роль в списке свойства Основные роли вашей конфигурации (рис. 4.19).

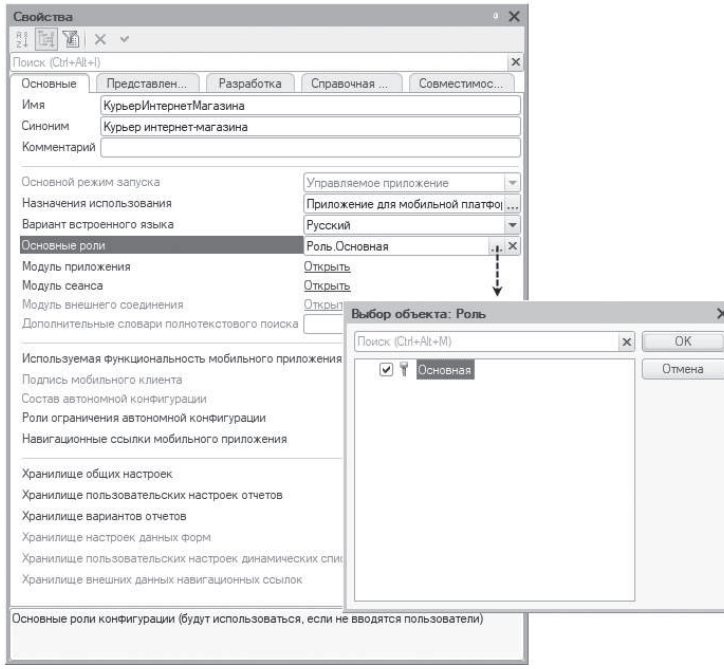


Рис. 4.19. Установка свойства мобильного приложения «Основные роли»

В результате при формировании командного интерфейса приложения мобильная платформа автоматически скроет от пользователя все недоступные команды в соответствии с правами, установленными для роли Основная.

Теперь можно уже заняться «программной начинкой» конфигурации, с помощью которой курьер сможет решать все возникающие перед ним задачи, описанные в разделе «Функциональность мобильного приложения».

Заказы

Доработка форм заказа

Сначала доработайте форму заказа и форму списка заказов так, чтобы курьеру было удобно и просто с ними работать.

Форма списка заказов унаследована из офисной конфигурации и имеет довольно много полей. На мобильной платформе это неприемлемо, особенно при вертикальной ориентации планшета. В то же время не хотелось бы сокращать количество полей в списке (например, оставлять только дату и номер), иначе список заказов будет выглядеть неинформативно.

Откройте форму списка документа Заказ в конфигурации КурьерИнтернетМагазина.

Самое простое – повлиять на отображение списка заказов с помощью свойства ПоведениеПриСжатииПоГоризонтали таблицы Список, отображающей данные динамического списка. Стандартно это свойство установлено в значение Авто, что на мобильной платформе трактуется как Скрывать элементы по важности. При этом колонки, не умещающиеся в одну строку, скрываются по мере убывания важности (или в порядке следования друг за другом в случае одинаковой важности).

Таким образом, если ничего не предпринимать, список заказов при вертикальной ориентации планшета будет выглядеть следующим образом (рис. 4.20).

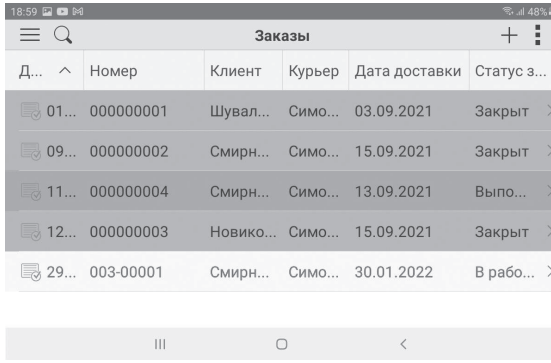


Рис. 4.20. Список заказов при вертикальной ориентации планшета

Получается не очень хорошо.

Попробуйте установить это свойство в значение Переносить элементы по важности (рис. 4.21 слева). При этом колонки, не умещающиеся

в одну строку, переносятся на следующую строку и отображаются мелким курсивом разного цвета через запятую. Порядок переноса не уместившихся колонок определяется свойством ВажностьПриОтображении. Такое поведение свойственно мобильному клиенту и рассматривалось во второй главе этой книги в разделе «Поведение таблиц при сжатии по горизонтали».

Установите свойство ВажностьПриОтображении у колонок таблицы формы Список в следующие значения (рис. 4.21 справа):

- Дата – Низкая,
- Номер – Низкая,
- Клиент – Высокая,
- Курьер – Очень низкая,
- ДатаДоставки – Очень высокая,
- СтатусЗаказа – Обычная.

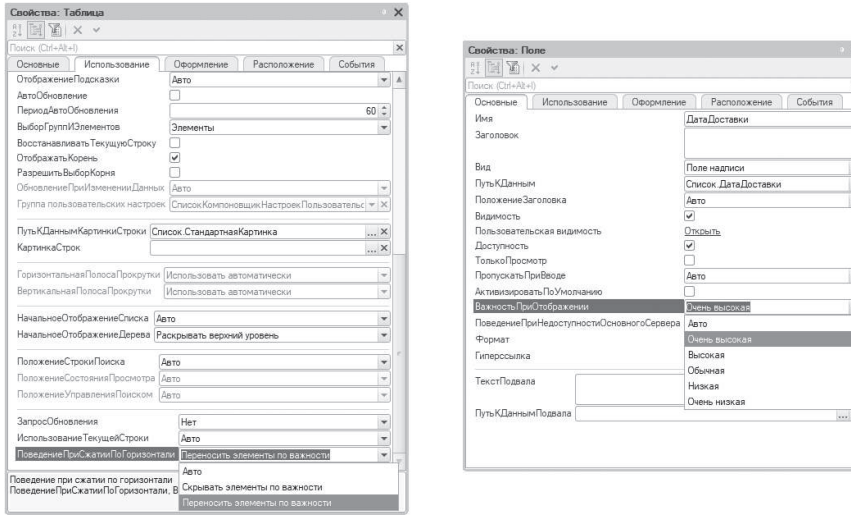


Рис. 4.21. Установка свойств, отвечающих за отображение списка заказов

В результате список заказов при вертикальной ориентации планшета будет выглядеть следующим образом (рис. 4.22).

Как вы видите, список выглядит так же, как и в приложении мобильного клиента. Кажется, что в таком виде список заказов выглядит эстетично, современно и информативно.

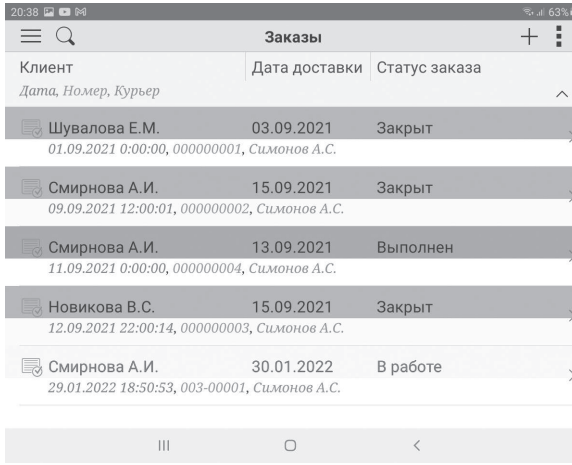


Рис. 4.22. Список заказов с переносом колонок по важности

Кстати, можно вообще не устанавливать у колонок списка важность при отображении, а просто поменять порядок их следования друг за другом. Ведь ваша конфигурация разрабатывается отдельно, и это никак не затронет офисное приложение. Такой способ будет рассмотрен дальше при работе со списком клиентов.

Конечно, это дело вкуса, но можно использовать и «старый», «проверенный» способ с группировкой колонок и их вложением друг в друга.

Например, на следующем рисунке 4.22 в дереве элементов формы в таблицу динамического списка заказов Список вложены две группы колонок с вертикальной группировкой. Группа1 содержит поля динамического списка Дата и Номер, а Группа2 – поля Клиент и Курьер, а также в нее вложена группа колонок Группа3 с горизонтальной группировкой, включающая поля ДатаДоставки и СтатусЗаказа.

Поскольку ранее вы отметили, что конфигурация будет использоваться только на мобильной платформе, то в нижнем окне редактора формы можно сразу увидеть, как выглядит форма на мобильном устройстве (рис. 4.23).

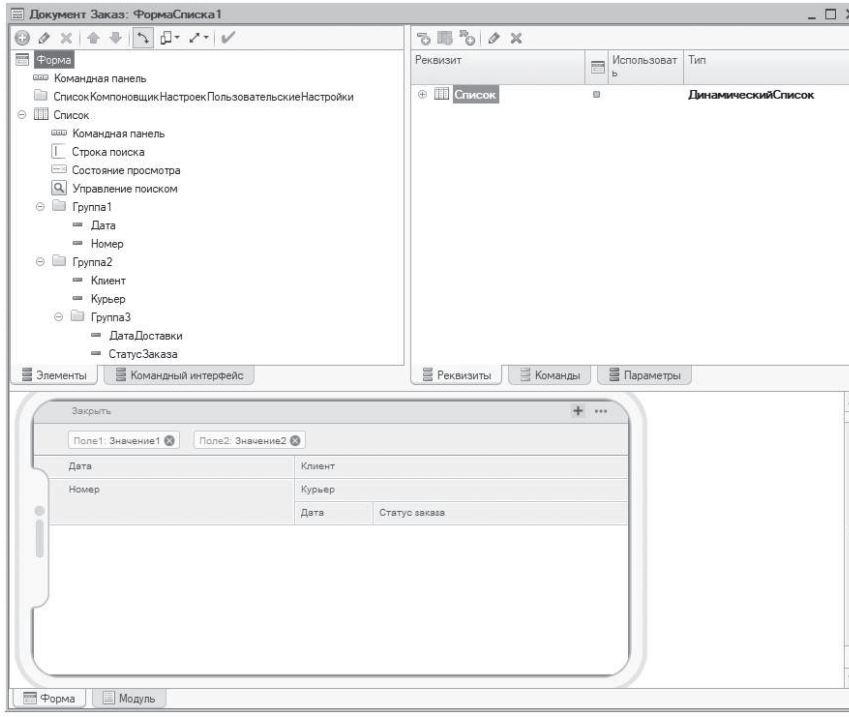


Рис. 4.23. Форма списка заказов с группировкой колонок в редакторе

Однако на планшете список заказов выглядит довольно громоздко, и, что особенно странно, вид списка никак не подстраивается под ориентацию планшета (рис. 4.24).

К тому же эти изменения сделать сложнее и дольше, чем просто переключить одно свойство, как в показанном выше варианте (см. рис. 4.21, 4.22). Одним словом – выбор за вами.

Теперь займитесь формой документа Заказ, которая также унаследована из офисной конфигурации. В окне предварительного просмотра редактора формы в офисной конфигурации показано, как выглядит форма на персональном компьютере. Переключив вид платформы, можно увидеть, как выглядит форма на мобильном устройстве (рис. 4.25).

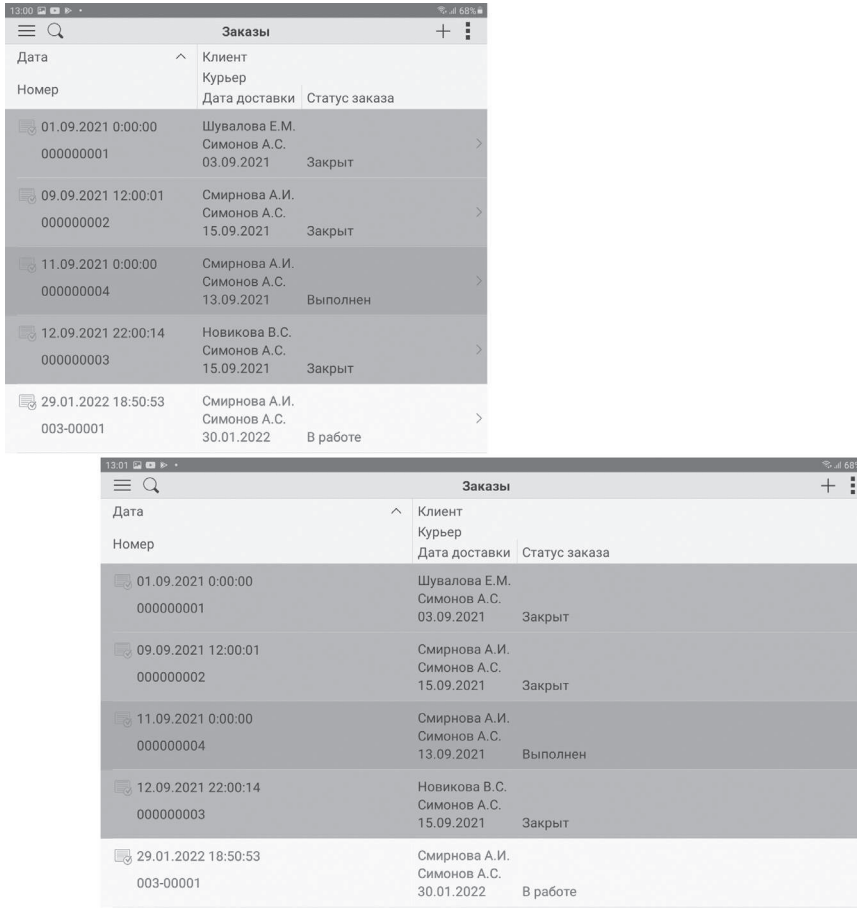


Рис. 4.24. Список заказов при вертикальной/горизонтальной ориентации планшета

Откройте форму заказа в конфигурации КурьерИнтернетМагазина. Вы видите, что группы ЛеваяКолонка и ПраваяКолонка на мобильном устройстве расположены вертикально друг под другом.

Дайте этим группам короткие понятные заголовки (соответственно Основное и Доставка), установите их свойство Поведение в значение Свертываемая и Отображение – в Слабое выделение, а также установите у группы Доставка флажок Свернута. Вынесите эти группы на верхний уровень и удалите ненужную теперь группу Шапка (рис. 4.26).

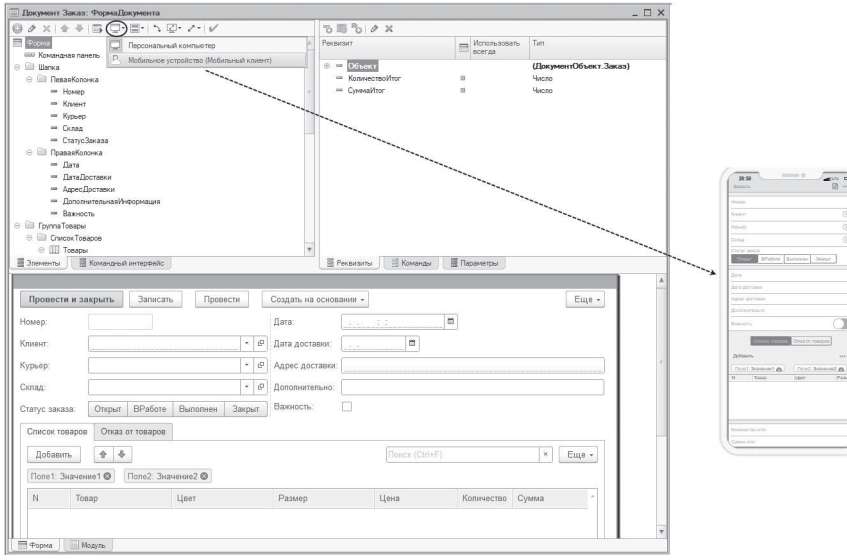


Рис. 4.25. Вид формы документа «Заказ» в офисной конфигурации

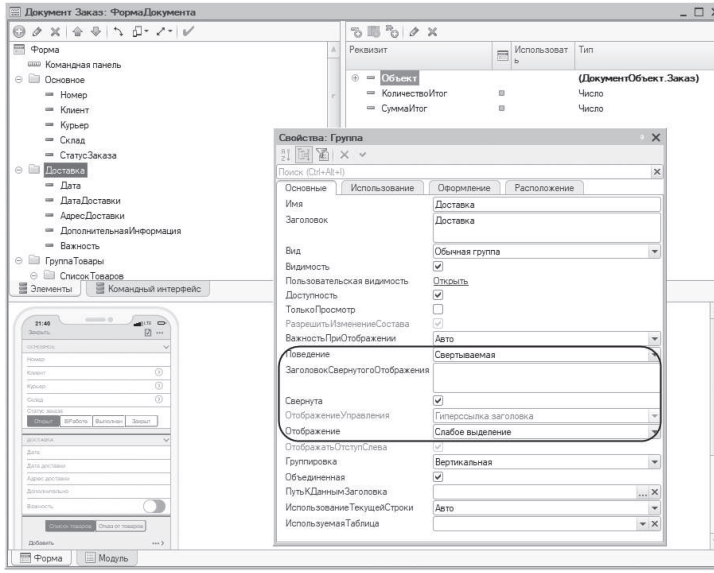


Рис. 4.26. Форма заказа в конфигурации «КурьерИнтернетМагазина»

А свойство ПоведениеПриСжатииПоГоризонтали таблицы Товары, отражающей данные табличной части документа, установите в значение Переносить элементы по важности.

В результате в мобильном приложении форма заказа будет выглядеть компактно и читабельно. Группы полей Основное и Доставка можно развернуть/свернуть, нажав на соответствующий значок справа от их названия. Вторая группа будет изначально свернута, а не поместившиеся по ширине колонки табличной части будут перенесены на следующую строку (рис. 4.27).

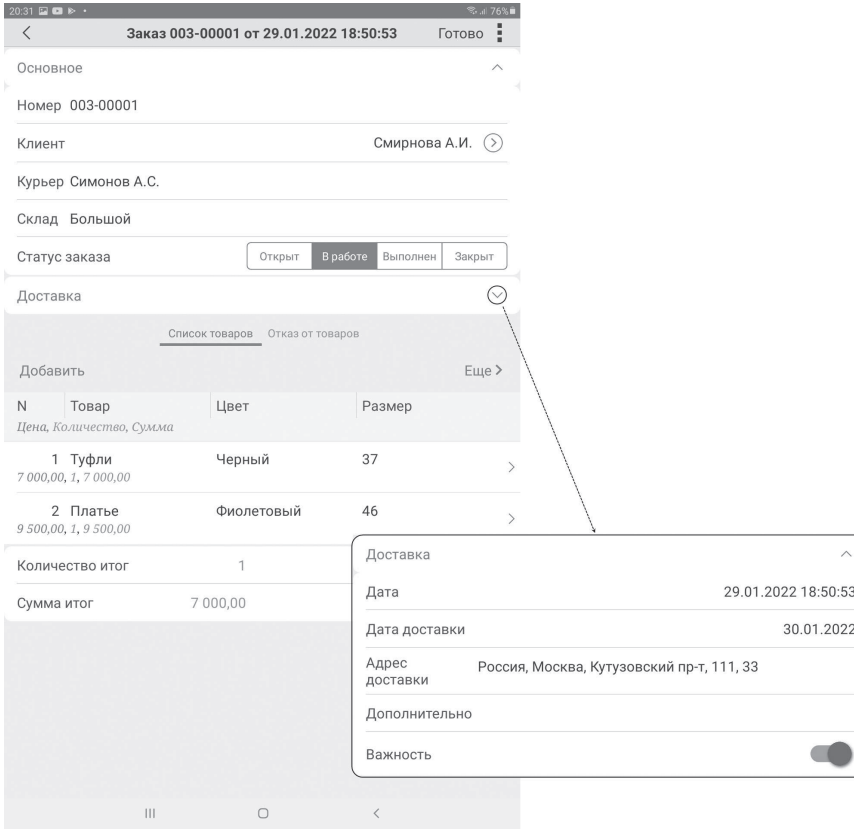


Рис. 4.27. Форма заказа в мобильном приложении

Сделайте еще несколько полезных усовершенствований в форме заказа.

Как вы видите, для поля Клиент стандартно доступна кнопка выбора (значок «>»), при нажатии на которую открывается форма выбора клиента. Кнопка открытия формы самого клиента по умолчанию не видна. Установите у поля Клиент свойство КнопкаОткрытия в значение Да. Теперь прямо из формы заказа вы можете просмотреть или изменить информацию о клиенте, нажав на значок с информацией («i») в этом поле (рис. 4.28).

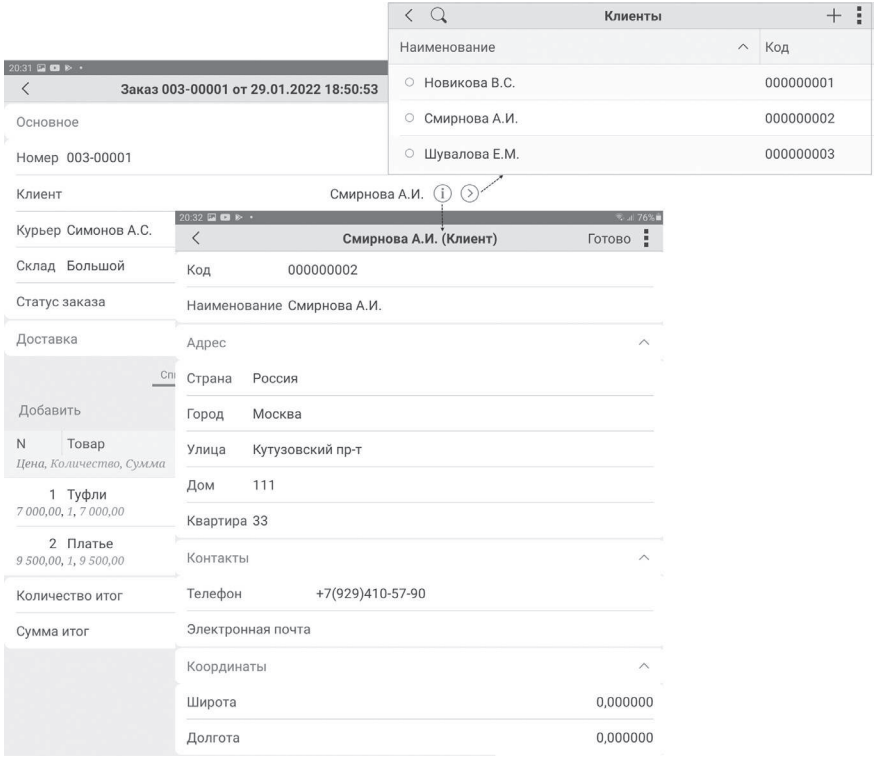


Рис. 4.28. Вызов из заказа формы клиента и формы выбора клиентов

ПРИМЕЧАНИЕ

Все вспомогательные формы в мобильном приложении открываются в отдельных окнах, а не всплывают, как показано на рис. 4.28. Это сделано просто для компактности рисунка.

При нажатии на строку таблицы со списком заказанных товаров или на значок «>» в конце каждой строки табличной части открывается специ-

альная системная форма для редактирования информации из таблицы. Установите у поля ТоварыКоличество свойство КнопкаРегулирования в значение Да, чтобы было удобнее увеличивать/уменьшать количество заказанных товаров. В результате форма редактирования примет следующий вид (рис. 4.29).

Рис. 4.29. Форма редактирования строки табличной части

Отказ от товаров

Как вы видите, информация о заказанных товарах и информация об отказе от товаров находятся в разных таблицах, расположенных на двух страницах (Список товаров и Отказ от товаров) формы заказа (см. рис. 4.27).

Кроме того, в таблице, содержащей информацию об отказе от товаров, при установке флажка Отказ сразу же вызывается форма выбора причины отказа от товара, и выбранное значение устанавливается в поле Причина отказа. А при снятии флажка Отказ причина отказа также очищается.

Это все уже реализовано в офисной конфигурации, и вам этого делать не придется.

Вам осталось сделать так, чтобы в мобильном приложении курьер не смог бы ни добавлять, ни удалять информацию в этой таблице, а также изменять список заказанных товаров.

Чтобы обеспечить такое поведение, установите у таблицы Товары1 свойство ПоложениеКомандногоПанели в значение Нет и отключите свойство ИзменятьСоставСтрок. А у поля этой таблицы Товары1Товар установите свойство ТолькоПросмотр.

Поведение этой таблицы (Товар1) при сжатии по горизонтали можно не менять. Информация в таблице смотрится неплохо, так как в ней немного полей (рис. 4.30).

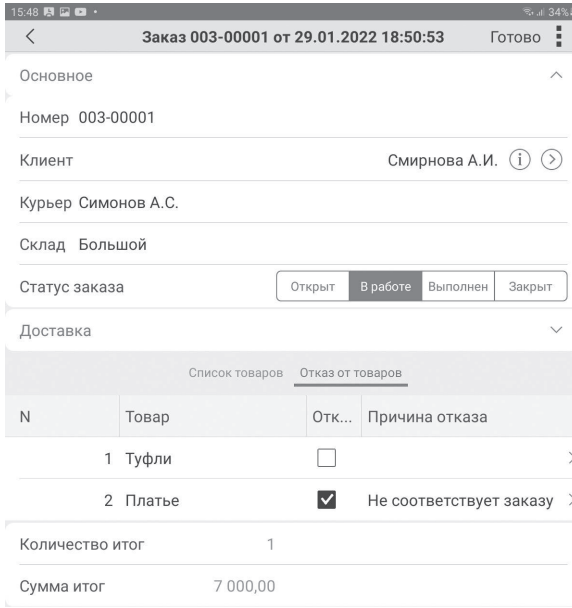


Рис. 4.30. Информация об отказе от товаров в форме заказа

Редактирование уже существующего заказа

Сделайте еще одно небольшое, но важное изменение в форме заказа – ограничьте возможность изменения информации в заказе, который находится в работе у курьера, так чтобы курьер мог вводить только информацию об отказе от товаров и изменять статус заказа. А если заказ уже закрыт или выполнен, нужно, чтобы курьер мог только просматривать такой заказ.

Для этого измените в модуле формы обработчик события ПриЧтенииНаСервере, который выполняется только для уже существующих в информационной базе объектов, следующим образом (листинг 4.10).

Листинг 4.10. Обработчик события формы документа «ПриЧтенииНаСервере»

```
&НаСервере
Процедура ПриЧтенииНаСервере(ТекущийОбъект)

    Если ТекущийОбъект.СтатусЗаказа = Перечисления.СтатусыЗаказов.Закрыт ИЛИ ТекущийОбъект.
                                        СтатусЗаказа = Перечисления.СтатусыЗаказов.Выполнен Тогда
        ЭтаФорма.ТолькоПросмотр = Истина;
    ИначеЕсли ТекущийОбъект.СтатусЗаказа = Перечисления.СтатусыЗаказов.ВРаботе Тогда
        Элементы.Номер.ТолькоПросмотр = Истина;
```

```
            Элементы.Дата.ТолькоПросмотр = Истина;
            Элементы.ДатаДоставки.ТолькоПросмотр = Истина;
            Элементы.АдресДоставки.ТолькоПросмотр = Истина;
            Элементы.Клиент.ТолькоПросмотр = Истина;
            Элементы.Курьер.ТолькоПросмотр = Истина;
            Элементы.Склад.ТолькоПросмотр = Истина;
            Элементы.ДополнительнаяИнформация.ТолькоПросмотр = Истина;
            Элементы.Важность.ТолькоПросмотр = Истина;
            Элементы.Товары.ТолькоПросмотр = Истина;
        КонецЕсли;

КонецПроцедуры
```

При создании новых заказов данный обработчик вызываться не будет, и новые заказы курьер сможет редактировать, как и ранее созданные заказы со статусом Открыт.

Работа с клиентами

Теперь реализуйте функции, необходимые для работы с клиентами непосредственно из формы заказа. Было бы удобно, чтобы, не закрывая заказа, курьер мог добавить нового клиента, для которого создается заказ, позвонить или отправить SMS клиенту.

Что касается добавления клиента, то это стандартная функциональность мобильной платформы, и здесь вам ничего делать не придется. Также в офисной конфигурации уже реализована удобная возможность – при добавлении нового клиента в заказ (или изменении уже существующего) сразу же заполняется адрес доставки, исходя из адресных реквизитов клиента. Таким образом, курьер может быстро и легко добавить нового клиента, не закрывая текущего заказа.

Чтобы позвонить клиенту или написать ему сообщение, используются возможности сотовой связи, которые поддерживаются не всеми мобильными устройствами. В частности, у планшета, который используется в книге, такой возможности нет, однако соответствующие функции предусмотрены в мобильном приложении.

Добавьте в форму заказа команды Позвонить и ОтправитьСМС и перетащите их в командную панель формы.

Чтобы сделать команду Позвонить более «симпатичной», задайте ее отображение в виде картинки со значком телефонной трубки. Поскольку подходящей картинки в библиотеке стандартных картинок платформы нет, добавьте соответствующую картинку в общие картинки вашего мобильного приложения (рис. 4.31).

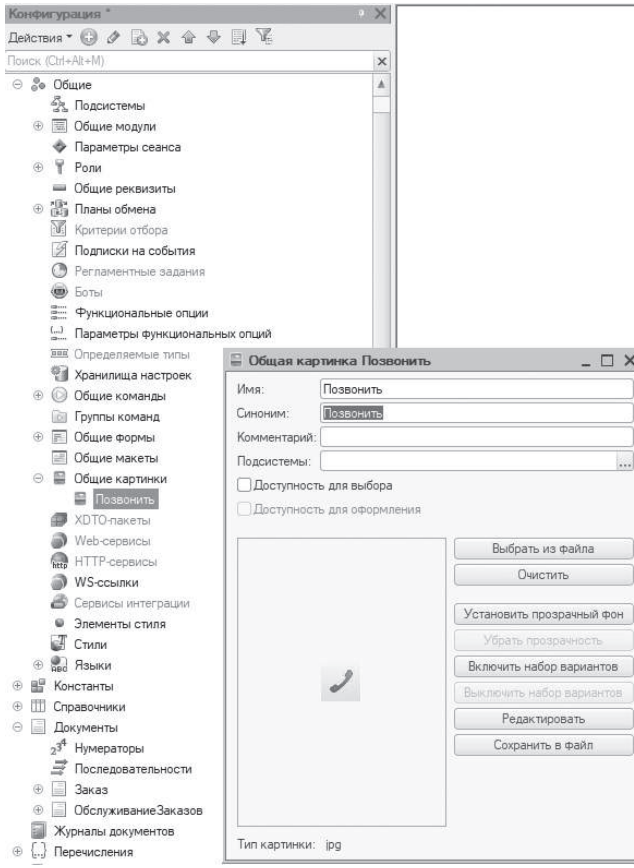


Рис. 4.31. Добавление картинки «Позвонить» в общие картинки конфигурации

После этого установите эту картинку в свойстве Картинка для команды Позвонить (рис. 4.32).

В результате в мобильном приложении эти команды (как и все остальные команды) будут доступны в меню действий, вызываемом при нажатии на три вертикальные точки в правом углу экрана (рис. 4.33).

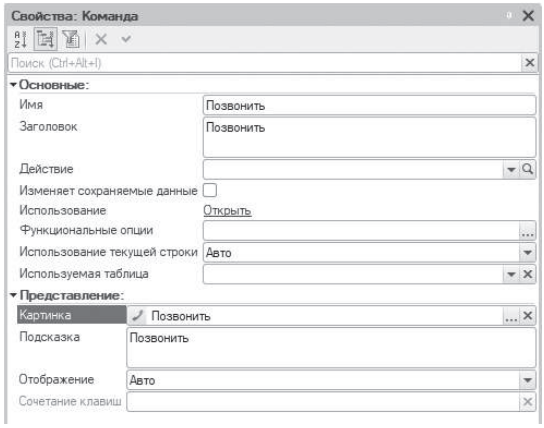


Рис. 4.32. Палитра свойств команды формы «Позвонить»

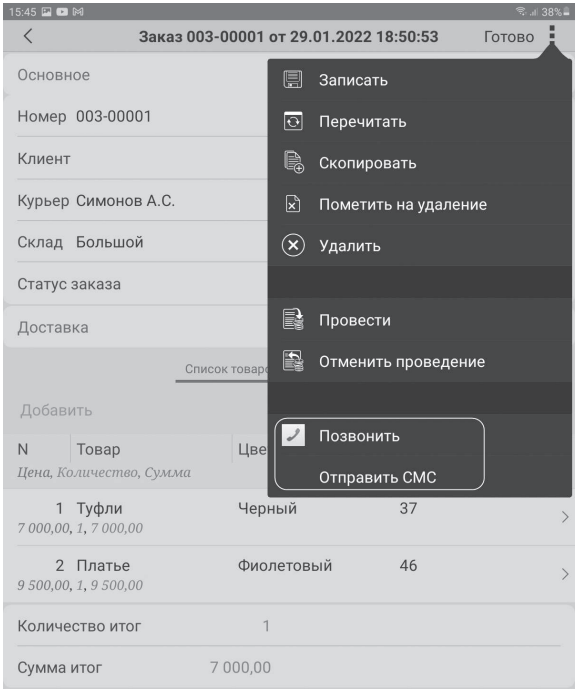


Рис. 4.33. Форма заказа в мобильном приложении

Затем создайте клиентские обработчики для команд Позвонить и ОтправитьСМС и заполните их следующим образом (листинг 4.11, листинг 4.12).

Листинг 4.11. Обработчик команды «Позвонить»

```
&НаКлиенте
Процедура Позвонить(Команда)

	Если ЗначениеЗаполнено(Объект.Клиент) Тогда
		Телефон = ПолучитьТелефон(Объект.Клиент);

		Если ЗначениеЗаполнено(Телефон) Тогда
			СредстваТелефонии.НабратьНомер(Телефон, Ложь);
		Иначе
			Сообщение = Новый СообщениеПользователю();
			Сообщение.Текст = "Не указан телефон клиента!";
			Сообщение.Поле  = "Объект.Клиент";
			Сообщение.Сообщить();
		КонецЕсли

	Иначе
		Сообщение = Новый СообщениеПользователю();
		Сообщение.Текст = "Клиент не выбран!";
		Сообщение.Поле  = "Объект.Клиент";
		Сообщение.Сообщить();
	КонецЕсли

КонецПроцедуры
```

Листинг 4.12. Обработчик команды «ОтправитьСМС»

```
&НаКлиенте
Процедура ОтправитьСМС(Команда)

	Если ЗначениеЗаполнено(Объект.Клиент) Тогда
		Телефон = ПолучитьТелефон(Объект.Клиент);

		Если ЗначениеЗаполнено(Телефон) Тогда
			Сообщение = Новый SMSСообщение();
			Сообщение.Получатели.Добавить(Телефон);
			СредстваТелефонии.ПослатьSMS(Сообщение, Истина);
		Иначе
			Сообщение = Новый СообщениеПользователю();
			Сообщение.Текст = "Не указан телефон клиента";
			Сообщение.Поле  = "Объект.Клиент";
			Сообщение.Сообщить();
		КонецЕсли

	Иначе
		Сообщение = Новый СообщениеПользователю();
		Сообщение.Текст = "Клиент не выбран!";
		Сообщение.Поле  = "Объект.Клиент";
		Сообщение.Сообщить();
	КонецЕсли

КонецПроцедуры
```

В обработчиках обеих команд, в случае если поле заказа Клиент заполнено, вызывается серверная внеконтекстная функция ПолучитьТелефон() для получения телефона клиента (листинг 4.13).

Листинг 4.13. Функция для получения телефона клиента

```
&НаСервереБезКонтекста
Функция ПолучитьТелефон(Клиент)

    Возврат Клиент.Телефон;

КонецФункции
```

Если телефон клиента заполнен, то выполняется звонок или отправка сообщения клиенту с помощью объекта глобального контекста СредстваТелефонии, который предоставляет доступ к средствам телефонии на мобильном устройстве.

Для звонка клиенту в процедуре Позвонить()используется метод средств телефонии НабратьНомер(), в который передается телефон клиента. Поскольку вы передаете во втором параметре метода Ложь, при выполнении метода будет открыто приложение работы со звонками, в котором уже установлен заданный номер. Курьеру нужно будет только нажать кнопку начала вызова.

Для отправки СМС клиенту в процедуре ОтправитьСМС() используется метод средств телефонии ПослатьSMS(). Сначала создается объект SMSСообщение, затем в свойство Получатели этого объекта добавляется телефон клиента. После этого вызывается метод средств телефонии ПослатьSMS(), в который первым параметром передается объект SMSСообщение, а вторым Истина. При этом будет запущено системное приложение для отправки SMS-сообщений. Курьеру остается ввести текст сообщения и нажать кнопку отправки.

Следует иметь в виду, что объекты СредстваТелефонии и SMSСообщение доступны только в мобильном приложении. Поэтому при синтаксической проверке текста модуля формы могут возникнуть ошибки. Но, поскольку вы указали в свойстве конфигурации Назначение использования значение Приложение для мобильной платформы, параметры проверки модулей (Сервис > Параметры > Модули > Проверка) уже стандартно должны быть настроены только для мобильного приложения. Если нет – оставьте отметку только у флажков Мобильное приложение – клиент и Мобильное приложение – сервер (рис. 4.34).

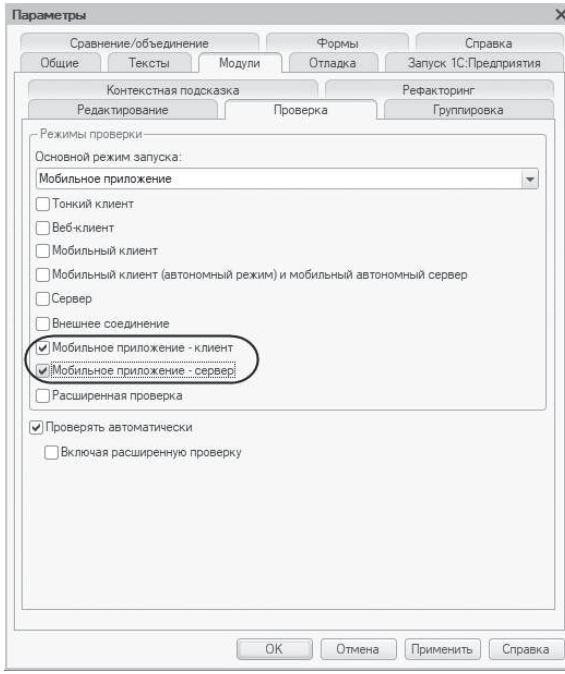


Рис. 4.34. Параметры проверки текстов модулей

При этом надо учитывать, что вы сможете без ошибок сохранять конфигурацию базы данных и обновлять мобильное приложение, но работать это будет только на планшете, так как работа со средствами телефонии (а также с данными мультимедиа, средствами геопозиционирования и средствами почты, которые будут показаны дальше) возможна только на мобильном устройстве.

Если же вам хочется, чтобы мобильное приложение работало и на стационарном компьютере, то все части кода, где происходит обращение к средствам телефонии и т. п., нужно обрамлять с помощью инструкций препроцессора, указывающих, что выполнять компиляцию этих частей кода нужно только на мобильном клиенте.

Например:

Листинг 4.14. Фрагмент обработчика команды «Позвонить»

```
&НаКлиенте
Процедура Позвонить(Команда)

    ...
      Если ЗначениеЗаполнено(Телефон) Тогда
#Если МобильноеПриложениеКлиент Тогда
          СредстваТелефонии.НабратьНомер(Телефон, Ложь);
#КонецЕсли
    Иначе
      ...

КонецПроцедуры
```

Как уже говорилось, возможности средств телефонии поддерживаются не на всех мобильных устройствах. Поэтому при открытии формы заказа установите доступность команд для работы со средствами телефонии только в случае, если мобильное устройство поддерживает такие возможности. Для этого добавьте в обработчик события формы ПриОткрытии следующий фрагмент кода (листинг 4.15).

Листинг 4.15. Обработчик события формы «ПриОткрытии»

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)

    ...

    Элементы.ФормаПозвонить.Доступность = СредстваТелефонии
                                          .ПоддерживаетсяНаборНомера();
    Элементы.ФормаОтправитьСМС.Доступность = СредстваТелефонии
                                          .ПоддерживаетсяОтправкаSMS(Истина);

КонецПроцедуры
```

В заключение установите в свойстве конфигурации Используемая функциональность мобильного приложения необходимые разрешения для работы со средствами телефонии, а также средствами мультимедиа, геопозиционирования и локальными уведомлениями (которые понадобятся вам позже) на мобильном устройстве (рис. 4.35).

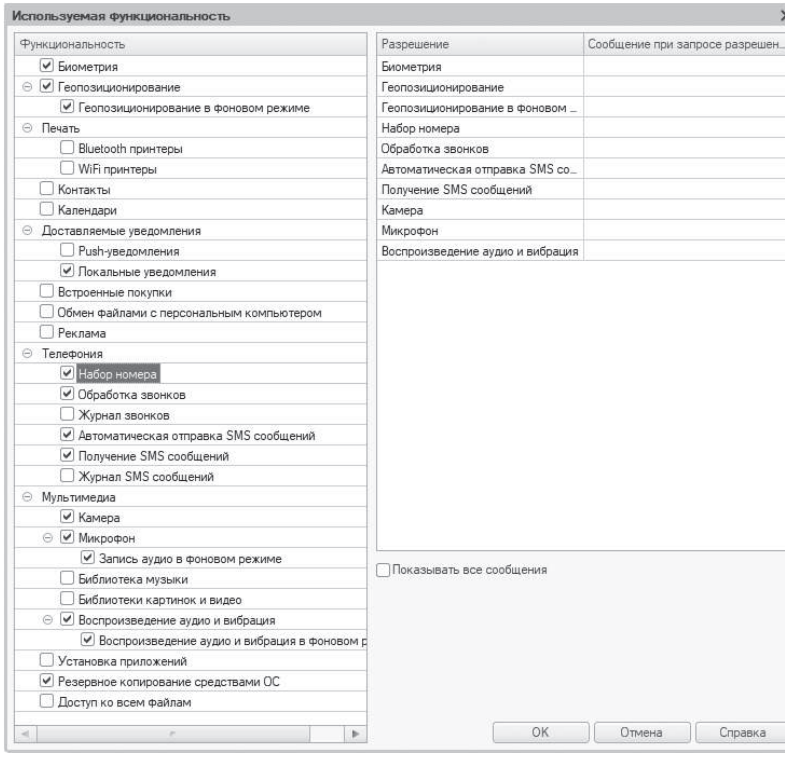


Рис. 4.35. Используемая функциональность мобильного приложения

Работа со средствами мультимедиа

Поскольку мобильные устройства обладают различными мультимедийными возможностями, было бы удобно, чтобы курьер мог воспользоваться этими возможностями прямо в заказе. Например, мог бы сделать фотоснимки товаров, не понравившихся клиенту, или сделать видео-/аудиозапись с отзывом клиента о заказе.

Файлы мультимедиа будут храниться в справочнике ХранимыеФайлы, унаследованном из офисной конфигурации. Кроме того, в этом справочнике хранятся также картинки товаров, которые выбираются в офисном приложении и в процессе обмена данными передаются в мобильное приложение.

Чтобы реализовать мультимедийные возможности, добавьте в форму заказа команды СделатьФотоснимок, СделатьВидеозапись и СделатьАудиоЗапись и перетащите их в командную панель формы.

Затем создайте клиентские обработчики для этих команд и заполните их следующим образом (листинги 4.16–4.18).

Листинг 4.16. Обработчик команды «СделатьФотоснимок»

```
&НаКлиенте
Процедура СделатьФотоснимок(Команда)

	Если Объект.Ссылка.Пустая() Тогда
		ПоказатьПредупреждение(, "Данные не записаны!");
		Возврат;
	КонецЕсли;

	Отметка = Новый ОтметкаНаФотоснимке(Истина, "ДФ='dd.MM.yyyy ЧЧ:мм'", "");
	ДанныеМультимедиа = СредстваМультимедиа.СделатьФотоснимок(,,,,,Отметка);
	Если ДанныеМультимедиа <> Неопределено Тогда
		СоздатьНовыйФайл(ДанныеМультимедиа.ПолучитьДвоичныеДанные(), ДанныеМультимедиа
				.РасширениеФайла, ДанныеМультимедиа.ТипСодержимого);
	КонецЕсли;

КонецПроцедуры
```

Листинг 4.17. Обработчик команды «СделатьВидеозапись»

```
&НаКлиенте
Процедура СделатьВидеозапись(Команда)

	Если Объект.Ссылка.Пустая() Тогда
		ПоказатьПредупреждение(, "Данные не записаны!");
		Возврат;
	КонецЕсли;

	ДанныеМультимедиа = СредстваМультимедиа.СделатьВидеозапись();
	Если ДанныеМультимедиа <> Неопределено Тогда
		СоздатьНовыйФайл(ДанныеМультимедиа.ПолучитьДвоичныеДанные(), ДанныеМультимедиа
				.РасширениеФайла, ДанныеМультимедиа.ТипСодержимого);
	КонецЕсли;

КонецПроцедуры
```

Листинг 4.18. Обработчик команды «СделатьАудиозапись»

```
&НаКлиенте
Процедура СделатьАудиозапись(Команда)

	Если Объект.Ссылка.Пустая() Тогда
		ПоказатьПредупреждение(, "Данные не записаны!");
		Возврат;
	КонецЕсли;

	ДанныеМультимедиа = СредстваМультимедиа.СделатьАудиозапись();
	Если ДанныеМультимедиа <> Неопределено Тогда
		СоздатьНовыйФайл(ДанныеМультимедиа.ПолучитьДвоичныеДанные(), ДанныеМультимедиа
				.РасширениеФайла, ДанныеМультимедиа.ТипСодержимого);
	КонецЕсли;

КонецПроцедуры
```

В этих обработчиках сначала проверяется, что данные заказа записаны, так как владельцем данных мультимедиа будет клиент, указанный в заказе.

Затем выполняется требуемый метод (СделатьФотоснимок(), СделатьВидеозапись(), СделатьАудиозапись()) средств мультимедиа, для доступа к которым используется свойство глобального контекста СредстваМультимедиа.

В результате выполнения этих методов возвращается объект ДанныеМультимедиа. Содержимое данных мультимедиа в виде двоичных данных, расширение при сохранении этих данных в файл и тип содержимого мультимедиа передаются в серверную процедуру СоздатьНовыйФайл() для сохранения данных мультимедиа в справочнике ХранимыеФайлы (листинг 4.19).

Листинг 4.19. Процедура для сохранения данных мультимедиа в справочнике «ХранимыеФайлы»

```
&НаСервере
Процедура СоздатьНовыйФайл(Данные, Расширение, Тип)

	ТипСодержимого = Тип;
	Номер = Найти(ТипСодержимого, "/");
	Если Номер > 0 Тогда
		ТипСодержимого = Лев(ТипСодержимого, Номер - 1);
	КонецЕсли;
	Файл = Новый Файл(СтрЗаменить(Строка(ТекущаяДата()), ":", "_") + "." + Расширение);

	ХранимыйФайл = Справочники.ХранимыеФайлы.СоздатьЭлемент();
	ХранимыйФайл.Владелец = Объект.Клиент;
	ХранимыйФайл.Наименование = "Заказ № " + Объект.Номер + " " + ТипСодержимого
	                                                          + " " + Строка(ТекущаяДата());
	ХранимыйФайл.ИмяФайла = Файл.Имя;
	ХранимыйФайл.ДанныеФайла = Новый ХранилищеЗначения(Данные, Новый СжатиеДанных());
	ХранимыйФайл.Записать();

КонецПроцедуры
```

В данной процедуре создается объект Файл с именем, образованным из текущей даты и расширения файла (jpg / mp4 / 3gp), переданного в процедуру. Затем добавляется новая запись в справочник ХранимыеФайлы. Имя файла, записывается в соответствующий реквизит справочника ИмяФайла, а данные мультимедиа помещаются в хранилище значения и сохраняются в реквизите ДанныеФайла.

Стандартному реквизиту справочника Владелец присваивается значение ссылки на клиента (Объект.Клиент). Таким образом, конкретные файлы мультимедиа будут относиться к клиенту, указанному в заказе. Реквизиту Наименование присваивается строка, состоящая из номера заказа, типа содержимого мультимедиа (image / video / audio) и текущей даты.

После этого созданный элемент справочника ХранимыеФайлы записывается. При этом надо понимать, что сам файл физически на планшете не создается, но данные мультимедиа в двоичном виде и имя, связанное с этими данными, сохраняются в справочнике хранимых файлов.

При открытии формы заказа установите видимость команд для работы со средствами мультимедиа только в случае, если мобильное устройство поддерживает такие возможности. Для этого добавьте в обработчик события формы ПриОткрытии следующий фрагмент кода (листинг 4.20).

Листинг 4.20. Фрагмент обработчика события формы «ПриОткрытии»

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
    ...

    Элементы.ФормаПозвонить.Доступность = СредстваТелефонии.ПоддерживаетсяНаборНомера();
    Элементы.ФормаОтправитьСМС.Доступность = СредстваТелефонии
                                            .ПоддерживаетсяОтправкаSMS(Истина);
    Элементы.ФормаСделатьАудиозапись.Доступность = СредстваМультимедиа
                                            .ПоддерживаетсяАудиозапись();
    Элементы.ФормаСделатьВидеозапись.Доступность = СредстваМультимедиа
                                            .ПоддерживаетсяВидеозапись();
    Элементы.ФормаСделатьФотоснимок.Доступность = СредстваМультимедиа
                                            .ПоддерживаетсяФотоснимок();

КонецПроцедуры
```

Проверьте работу со средствами мультимедиа на планшете. Выполните из меню действий заказа команды СделатьФотоснимок, СделатьВидеозапись и СделатьАудиоЗапись (рис. 4.36).

Справочник ХранимыеФайлы подчинен справочникам Товары и Клиенты. При создании файлов мультимедиа в форме заказа вы задали в качестве владельца этих файлов клиента, указанного в заказе (см. листинг 4.19).

Таким образом, открыв форму клиента и активизировав закладку Хранимые файлы в панели навигации внизу формы, курьер может просмотреть файлы, связанные с этим клиентом (рис. 4.37).

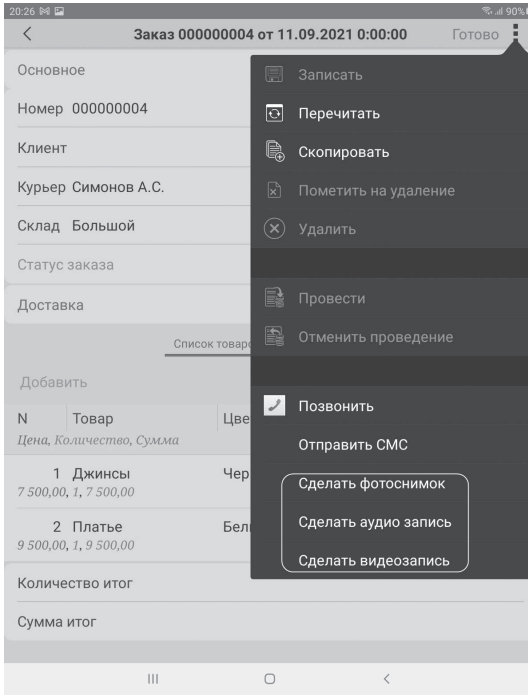


Рис. 4.36. Фото, аудио- и видеозапись из формы заказа

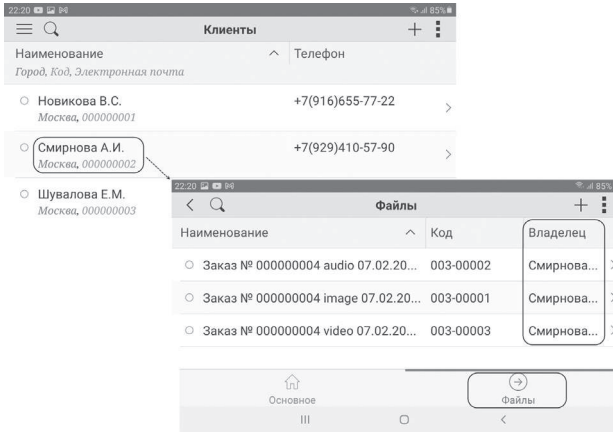


Рис. 4.37. Список хранимых файлов, связанных с конкретным клиентом

Список хранимых файлов

В форме списка справочника Хранимые файлы, унаследованной из офисной конфигурации, реализована возможность открытия содержимого текущего файла соответствующим приложением. Для этого в форму списка добавлена команда ОткрытьФайл. Связанная с ней кнопка находится в командной панели формы.

Чтобы эта команда работала в мобильном клиенте, вам нужно только установить свойство команды ИспользованиеТекущейСтроки в значение Использует, а в свойстве ИспользуемаяТаблица выбрать таблицу Список так же, как было показано в главе про мобильный клиент в разделе «Использование текущей строки командой формы» (рис. 4.38).

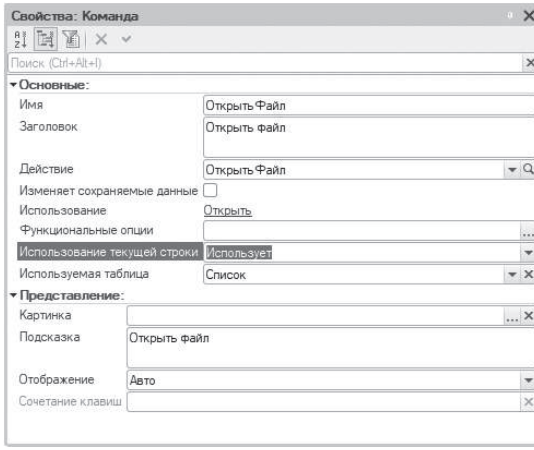


Рис. 4.38. Свойства команды «ОткрытьФайл»

Дело в том, что данные текущей строки таблицы в мобильном приложении доступны только во время обработчика контекстной команды, событий активизации строки или ячейки или событий редактирования строки. А эти данные нужны для получения ссылки на хранимый файл, который требуется открыть.

Обработчик команды ОткрытьФайл выглядит следующим образом (листинг 4.21).

Листинг 4.21. Обработчик команды «ОткрытьФайл»

```
&НаКлиенте
Процедура ОткрытьФайл(Команда)

    ХранимыйФайл = Элементы.Список.ТекущиеДанные;
    ВыполнитьОткрытьФайл(ХранимыйФайл);

КонецПроцедуры
```

В этом обработчике вы обращаетесь к таблице формы Список, которая отображает данные динамического списка, полученного на основе справочника ХранимыеФайлы. Используя свойство ТекущиеДанные этой таблицы, в переменной ХранимыйФайл вы получаете объект, содержащий данные, находящиеся в текущей строке таблицы. Этот объект вы передаете в клиентскую процедуру ВыполнитьОткрытьФайл(), где, собственно, и происходит открытие файла мультимедиа соответствующим ему приложением.

Измените эту процедуру следующим образом (листинг 4.22).

Листинг 4.22. Процедура для открытия файла мультимедиа

```
&НаКлиенте
Процедура ВыполнитьОткрытьФайл(ХранимыйФайл);

    Файл = Новый Файл(ХранимыйФайл.ИмяФайла);
    ИмяВременногоФайла = ПолучитьИмяВременногоФайла(Файл.Расширение);
    Адрес = ПолучитьНавигационнуюСсылку(ХранимыйФайл.Ссылка, "ДанныеФайла");
    ПолучитьФайл(Адрес, ИмяВременногоФайла, Ложь);
    ЗапуститьПриложение(ИмяВременногоФайла);

КонецПроцедуры
```

В данной процедуре сначала вы создаете объект Файл на основе значения реквизита ИмяФайла элемента справочника хранимых файлов, переданного в процедуру в параметре ХранимыйФайл.

Затем, для того чтобы записать данные мультимедиа во временный файл, методом глобального контекста ПолучитьИмяВременногоФайла() вы получаете уникальное имя временного файла с желаемым расширением, полученным из созданного объекта Файл.

Затем с помощью метода ПолучитьНавигационнуюСсылку() переменной Адрес вы присваиваете навигационную ссылку на реквизит ДанныеФайла объекта, переданного в процедуру в параметре ХранимыйФайл. В первом параметре метода ПолучитьНавигационнуюСсылку() передается ссылка на объект – ХранимыйФайл.Ссылка, а во втором – имя реквизита.

Далее с помощью метода ПолучитьФайл() на основе навигационной ссылки, сохраненной в переменной Адрес, вы получаете файл с данными мультимедиа и сохраняете его на планшете под именем временного файла.

И после этого с помощью метода глобального контекста ЗапуститьПриложение() выполняется открытие этого временного файла с помощью ассоциированного с ним приложения (например, при открытии видео запускается видеоплеер и т. п.).

В результате в списке хранимых файлов в мобильном приложении можно открыть его содержимое, нажав в контекстном меню Открыть файл (рис. 4.39).

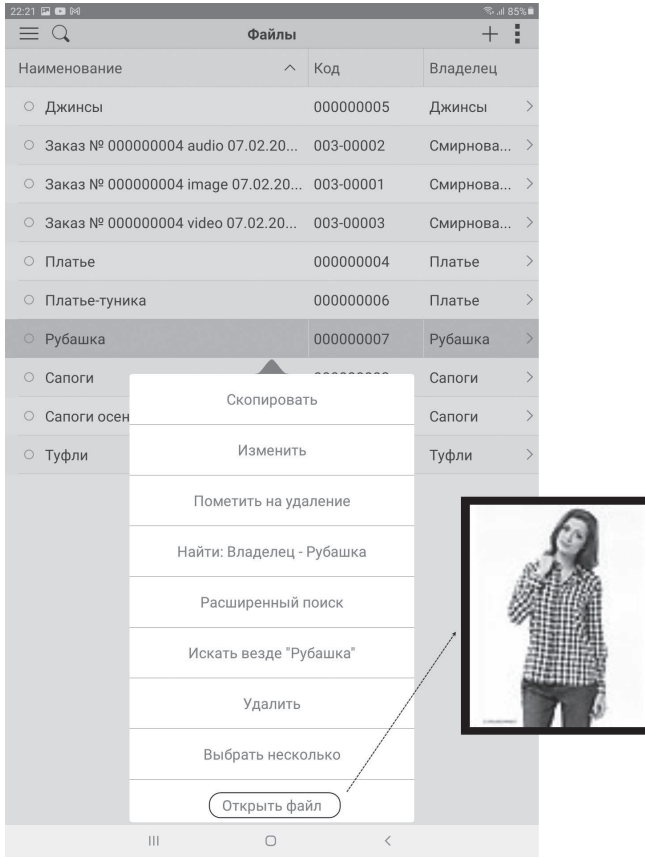


Рис. 4.39. Открытие файла из списка хранимых файлов

Теперь сделайте так, чтобы при выборе строки списка открывалась бы не форма элемента справочника хранимых файлов, как происходит по умолчанию, а соответствующий файл мультимедиа.

Для этого создайте обработчик события Выбор для таблицы формы Список и заполните его следующим образом (листинг 4.23).

Листинг 4.23. Обработчик события «Выбор» для таблицы «Список»

```
&НаКлиенте
Процедура СписокВыбор(Элемент, ВыбраннаяСтрока, Поле, СтандартнаяОбработка)

    СтандартнаяОбработка = Ложь;
    ВыполнитьОткрытьФайл(Элемент.ТекущиеДанные);

КонецПроцедуры
```

В этом обработчике вы отменяете стандартную обработку события, при котором открывается форма элемента справочника, выбранного в списке.

В параметре обработчика Элемент вам доступна таблица формы Список, которая отображает данные динамического списка, полученного на основе справочника ХранимыеФайлы. Используя свойство ТекущиеДанные этой таблицы, вы получаете объект, содержащий данные, находящиеся в текущей строке таблицы. Этот объект вы передаете в клиентскую процедуру ВыполнитьОткрытьФайл(), где, собственно, и происходит открытие файла мультимедиа соответствующим ему приложением (см. листинг 4.22).

В результате при нажатии на строку списка открывается файл мультимедиа, а не форма элемента справочника (как происходит по умолчанию). Теперь, чтобы открыть форму хранимого файла, нужно нажать Изменить в контекстном меню, вызываемом с помощью жеста пролистывания с правой стороны экрана у конкретной строки списка хранимых файлов.

Но основная форма справочника ХранимыеФайлы, как правило, нужна только в офисном приложении для чтения/открытия и записи в локальную файловую систему пользователя. В справочнике в основном хранятся файлы картинок, которые выбираются и устанавливаются в офисе для конкретных товаров.

А создание файлов мультимедиа для клиентов лучше делать непосредственно из формы заказа, как показано в предыдущем разделе «Работа со средствами мультимедиа». Реализовывать этот же функционал в форме хранимого файла (как это было сделано в главе про мобильный клиент «Работа с мультимедиа») нет особенного смысла. Поэтому уберите признак основной формы объекта у справочника ХранимыеФайлы и удалите эту форму из вашей конфигурации.

Клиенты

Сначала доработайте форму списка справочника Клиенты, чтобы список клиентов был более компактным и читабельным и чтобы в начале списка отображались бы наиболее важные данные клиентов.

Для этого свойство ПоведениеПриСжатииПоГоризонтали таблицы формы Список, отражающей данные динамического списка элементов справочника, установите в значение Переносить элементы по важности и поменяйте порядок следования полей в этой таблице так, как вы считаете нужным. Это равносильно тому, как если бы вы (не меняя порядок) установили у этих полей свойство ВажностьПриОтображении. Но, поскольку ваша конфигурация разрабатывается отдельно и ваши изменения никак не затронут офисное приложение, можно сделать и так – или совсем удалить некоторые ненужные поля из таблицы формы.

Теперь наполните форму элемента справочника Клиенты той функциональностью, которая может понадобиться курьеру при доставке заказов клиенту.

Прежде всего курьер должен определить местоположение клиента на карте по его адресу, чтобы понять, как до него добраться. Для этого используются возможности геопозиционирования, позволяющие определять географические координаты по адресу. Возможна и обратная операция – определение адреса по географическим координатам исходя из текущего местоположения мобильного устройства.

Кроме того, предоставьте курьеру удобную возможность проложить маршрут до клиента, используя сервис карт Google Maps.

Также курьер должен иметь возможность связаться с клиентом (позвонить или отправить СМС), чтобы уточнить время или адрес доставки. Для этого используются возможности сотовой связи, которые поддерживаются не всеми мобильными устройствами. В частности, у планшета, который используется в книге, такой возможности нет, однако соответствующие функции предусмотрены в мобильном приложении.

Кроме того, предоставьте курьеру удобную возможность создать для себя напоминание, например, о том, что ему нужно перезвонить клиенту в определенное время. Это локальное напоминание должно сработать в указанные дату и время – в результате должна автоматически появиться форма звонка с заполненным телефоном клиента и текстом напоминания.

А также курьер должен иметь возможность послать клиенту письмо по электронной почте, используя его электронный адрес.

Команды звонка и отправки СМС клиенту уже реализованы вами в форме заказа. Здесь их нужно просто продублировать, а откуда их вызывать – дело вкуса и привычки конкретного пользователя мобильного приложения.

Команды для определения местоположения клиента на карте также можно было бы поместить в форму заказа, но вообще-то не стоит перенасыщать заказ командами, иначе в них будет трудно потом разобраться.

Итак, откройте форму элемента справочника Клиенты и добавьте в нее команды ПоказатьНаКарте, ИспользоватьТекущееМестоположение, ПроложитьМаршрут, Позвонить, ОтправитьСМС, НапомнитьОЗвонке и ОтправитьПисьмо.

Для команды Позвонить в свойстве Картинка выберите картинку Позвонить из общих картинок вашей конфигурации.

Затем добавьте в командную панель формы две группы вида Подменю с заголовками Телефония и Геопозиционирование и перетащите в них соответствующие команды. Команду ОтправитьПисьмо перетащите не в подменю, а просто в командную панель формы.

В результате форма элемента справочника в конфигураторе примет следующий вид (рис. 4.40).

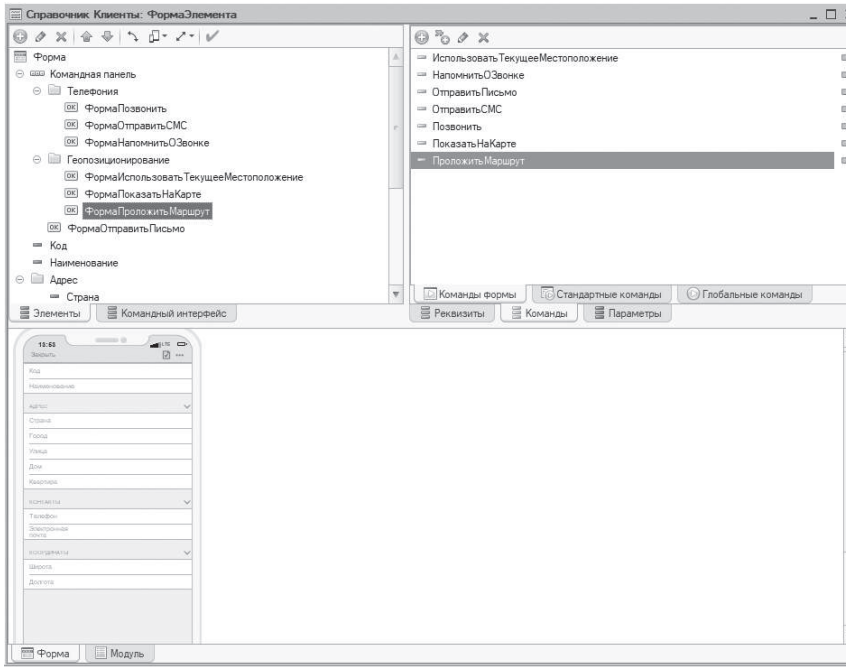


Рис. 4.40. Форма клиента в редакторе формы

Геопозиционирование

Сначала реализуйте в форме клиента команды, связанные с геопозиционированием. Чтобы показать на карте местоположение клиента по его адресу, создайте обработчик команды ПоказатьНаКарте и заполните его следующим образом (листинг 4.24).

Листинг 4.24. Обработчик команды «ПоказатьНаКарте»

```
&НаКлиенте
Процедура ПоказатьКарту(Команда)

    Координаты = ПолучитьКоординатыКлиента();

    Если Координаты <> Неопределено Тогда
        ПоказатьНаКарте(Координаты);
    Иначе
        // Сообщим пользователю о том, что информация не консистентна
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Не заполнены поля, описывающие адрес клиента!";
        Сообщение.Поле  = "Объект.Страна";
        Сообщение.Сообщить();
    КонецЕсли;

КонецПроцедуры
```

В этом обработчике вы сначала вызываете функцию ПолучитьКоординатыКлиента() для получения географических координат местоположения клиента (см. листинг 4.25).

После этого вы выполняете метод ПоказатьНаКарте(), в который передаете полученные координаты. Для отображения местоположения клиента по его адресу используются мобильные карты Google Maps. С картами Google Maps мобильная платформа работает «напрямую», установка их в виде отдельного приложения не требуется.

Нужно учитывать, что во время разработки вы запускаете мобильное приложение через платформу разработчика, поэтому для работы с геопозиционированием ничего настраивать не нужно. А вот при сборке вашего мобильного приложения нужно получить специальный ключ для работы с Google Maps и сделать необходимые настройки в сборщике приложений.

Если координаты не определены, значит, адресные реквизиты клиента не заполнены. В этом случае выводится сообщение об ошибке.

Листинг 4.25. Функция «ПолучитьКоординатыКлиента()»

```
&НаКлиенте
Функция ПолучитьКоординатыКлиента()

    Координаты = Неопределено;
    Если Объект.Широта <> 0 ИЛИ Объект.Долгота <> 0 Тогда
```

```
        Координаты = Новый ГеографическиеКоординаты(Объект.Широта, Объект.Долгота);
    Иначе
        СтруктураДанныхАдреса = Новый Структура();
        СтруктураДанныхАдреса.Вставить("Страна",Объект.Страна);
        СтруктураДанныхАдреса.Вставить("Город",Объект.Город);
        СтруктураДанныхАдреса.Вставить("Улица",Объект.Улица);
        СтруктураДанныхАдреса.Вставить("Дом",Объект.Дом);
        ДанныеАдреса = Новый ДанныеАдреса(СтруктураДанныхАдреса);
        Координаты = ПолучитьМестоположениеПоАдресу(ДанныеАдреса);
    КонецЕсли;
    Возврат Координаты;

КонецФункции
```

В функции для получения координат клиента, если реквизиты клиента Широта и Долгота заполнены, то на их основе вы создаете объект ГеографическиеКоординаты для описания координат местоположения.

Если координаты не заполнены, то вы создаете структуру с полями Страна, Город, Улица, Дом и заполняете ее значениями соответствующих строковых реквизитов адреса клиента. Затем на основе этой структуры создаете объект ДанныеАдреса для описания адреса клиента. Для более точного описания адреса можно добавить в структуру поля Регион и Индекс, но в данном примере у клиента этих реквизитов нет. Затем с помощью метода ПолучитьМестоположениеПоАдресу() вы получаете географические координаты клиента по его адресу, описанному в объекте ДанныеАдреса.

Проверьте, как это работает на планшете. Прежде всего проверьте настройки геолокации на вашем устройстве. Для выполнения команд геопозиционирования нужно, чтобы в разделе Локация были включены соответствующие настройки и установлены разрешения для мобильной платформы «1С:Предприятия» (рис. 4.41).

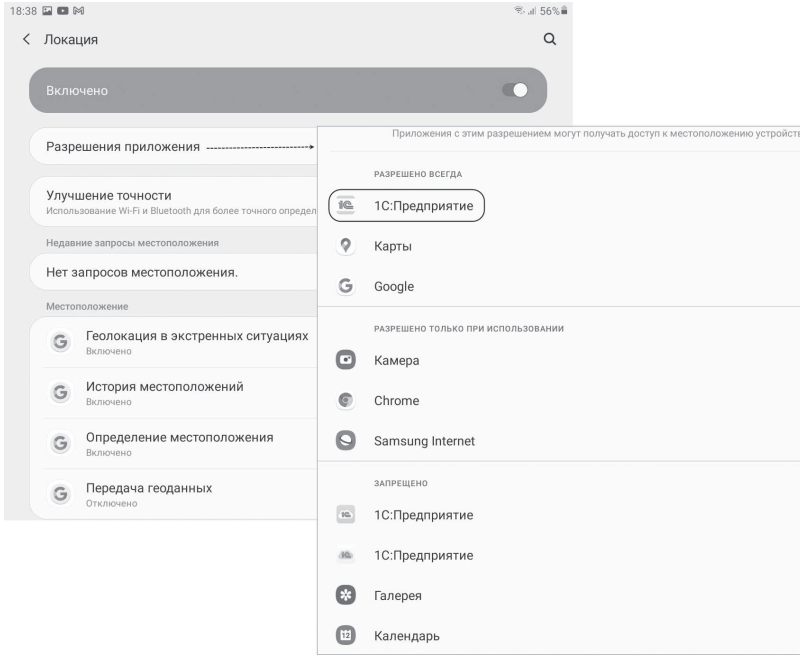


Рис. 4.41. Настройки геолокации

После этого откройте форму клиента, в меню действий раскройте группу команд Геопозиционирование и нажмите Показать на карте. В результате местоположение клиента, соответствующее его адресу, показывается на карте (рис. 4.42).

В приведенном выше примере у клиента заполнены адресные реквизиты, но не заполнены координаты. Если координаты клиента заполнены, то отображаться на карте будут именно они, а не координаты, полученные по адресу.

При выполнении команды Показать на карте может появиться сообщение, предлагающее обновить приложение Google Play. Для этого нужно добавить существующий аккаунт Google или создать новый.

СОВЕТ

Если вам не удалось получить координаты местоположения клиента по его адресу или координаты адреса по местоположению, хотя у клиента заполнены все адресные данные, то просто перезагрузите ваше мобильное устройство – и все должно заработать.

Рис. 4.42. Отображение адреса клиента на карте

Теперь реализуйте команду для определения адреса клиента по текущему местоположению мобильного устройства. Для этого создайте обработчик команды ИспользоватьТекущееМестоположение и заполните его следующим образом (листинг 4.26).

Листинг 4.26. Обработчик команды «ИспользоватьТекущееМестоположение»

```
&НаКлиенте
Процедура ИспользоватьТекущееМестоположение(Команда)

    Провайдер = СредстваГеопозиционирования.ПолучитьСамогоЭнергоЭкономичногоПровайдера();

    Если СредстваГеопозиционирования.ОбновитьМестоположение(Провайдер.Имя, 60) Тогда
        ДанныеМестоположения = СредстваГеопозиционирования
                                .ПолучитьПоследнееМестоположение(Провайдер.Имя);
        ДанныеАдреса = ПолучитьАдресПоМестоположению(ДанныеМестоположения.Координаты);

        Если ДанныеАдреса <> Неопределено Тогда
            Объект.Страна = ДанныеАдреса.Страна;
            Объект.Город = ДанныеАдреса.Город;
            Объект.Улица = ДанныеАдреса.Улица;
            Объект.Дом = ДанныеАдреса.Дом;
            Объект.Широта = ДанныеМестоположения.Координаты.Широта;
            Объект.Долгота = ДанныеМестоположения.Координаты.Долгота;
            Модифицированность = Истина;
        Иначе
            Сообщение = Новый СообщениеПользователю();
            Сообщение.Текст = "Не удалось установить адрес по местоположению!";
            Сообщение.Сообщить();
        КонецЕсли;

    Иначе
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Не удалось получить данные от провайдера геопозиционирования!";
        Сообщение.Сообщить();
    КонецЕсли;

КонецПроцедуры
```

В этом обработчике с помощью свойства глобального контекста СредстваГеопозиционирования вы получаете доступ к средствам геопозиционирования, которые определены только на мобильном клиенте. Для определения текущего местоположения мобильного устройства предназначен провайдер геопозиционирования.

С помощью метода средств геопозиционирования ПолучитьСамогоЭнергоЭкономичногоПровайдера() вы получаете информацию о провайдере, который обеспечивает минимальное потребление энергии.

Затем вы выполняете метод средств геопозиционирования ОбновитьМестоположение() и передаете туда имя полученного провайдера геопозиционирования (Провайдер.Имя), для того чтобы провайдер определил актуальные данные о местоположении.

Во втором параметре метода вы назначаете тайм-аут на определение местоположения – 60 секунд. В случае если метод завершен по тайм-ауту, возвращается Ложь и пользователю выводится сообщение об ошибке.

Если метод ОбновитьМестоположение() завершен успешно, возвращается Истина. В этом случае вы выполняете метод средств геопозиционирования ПолучитьПоследнееМестоположение(). Этот метод возвращает последние данные о местоположении, полученные переданным провайдером геопозиционирования, описанные в объекте ДанныеМестоположения. В свойстве этого объекта Координаты содержатся географические координаты текущего местоположения.

На основе этих координат с помощью метода ПолучитьАдресПоМестоположению() вы получаете адрес в виде объекта ДанныеАдреса, который описывает адрес в виде структуры полей Страна, Город, Улица, Дом, Регион, Индекс.

Затем вы заполняете адресные реквизита клиента значениями соответствующих полей структуры. Реквизиты Широта и Долгота заполняете аналогичными данными объекта ДанныеМестоположения. А также свойству Модифицированность формы присваиваете значение Истина, так как в форме изменились данные адреса клиента.

Если структура ДанныеАдреса не определена, то выводите сообщение об ошибке.

В результате выполнения команды Использовать текущее местоположение адресные реквизиты и географические координаты клиента будут заполнены данными о местоположении мобильного устройства.

Теперь реализуйте команду, которая с помощью сервиса карт Google Maps позволяет проложить маршрут от текущего местоположения курьера (то есть мобильного устройства) до местонахождения клиента. Для этого создайте обработчик команды ПроложитьМаршрут и заполните его следующим образом (листинг 4.27).

Листинг 4.27. Обработчик команды «ПроложитьМаршрут»

```
&НаКлиенте
Процедура ПроложитьМаршрут(Команда)

    ДанныеМестоположения = Неопределено;
    Провайдер = СредстваГеопозиционирования.ПолучитьСамогоЭнергоЭкономичногоПровайдера();

    Если СредстваГеопозиционирования.ОбновитьМестоположение(Провайдер.Имя, 60) Тогда
        ДанныеМестоположения = СредстваГеопозиционирования
                                    .ПолучитьПоследнееМестоположение(Провайдер.Имя);
    КонецЕсли;

    Если ДанныеМестоположения = Неопределено Тогда
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Не удалось установить текущее местоположение!";
        Сообщение.Сообщить();
        Возврат;
```

```
    КонецЕсли;

    КоординатыКлиента = ПолучитьКоординатыКлиента();
    Если КоординатыКлиента = Неопределено Тогда
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Не удалось установить расположение клиента!";
        Сообщение.Сообщить();
        Возврат;
    КонецЕсли;

    Запуск = Новый ЗапускПриложенияМобильногоУстройства("android.intent.action.VIEW",
            "http://maps.google.com/maps?saddr="
            + XMLСтрока(ДанныеМестоположения.Координаты.Широта) + "," +
            XMLСтрока(ДанныеМестоположения.Координаты.Долгота)
            + "&daddr="
            + XMLСтрока(КоординатыКлиента.Широта) + "," +
            XMLСтрока(КоординатыКлиента.Долгота));
    Запуск.Запустить(Ложь);

КонецПроцедуры
```

В этом обработчике вы, так же как и в предыдущем примере (см. листинг 4.26), сначала получаете самого энергоэкономичного провайдера геопозиционирования. Затем вы выполняете метод средств геопозиционирования ОбновитьМестоположение() и передаете туда имя полученного провайдера геопозиционирования (Провайдер.Имя), для того чтобы провайдер определил актуальные данные о местоположении.

Если данные о местоположении обновлены успешно, вы выполняете метод средств геопозиционирования ПолучитьПоследнееМестоположение(). Этот метод в объекте ДанныеМестоположения возвращает последние данные о местоположении, полученные провайдером геопозиционирования. В свойстве этого объекта Координаты содержатся географические координаты текущего местоположения.

Затем вы вызываете функцию ПолучитьКоординатыКлиента для получения географических координат местоположения клиента (см. листинг 4.25). Если координаты текущего местоположения или координаты клиента не определены, то выводите сообщение об ошибке.

После этого вы запускаете сервис карт Google Maps, в который передаются координаты текущего местоположения мобильного устройства и координаты местонахождения клиента для построения маршрута движения между ними.

Проверьте, как это работает на планшете. Откройте форму клиента, в меню действий раскройте группу команд Геопозиционирование и нажмите Проложить маршрут. В результате на карте прокладывается маршрут движения от местоположения курьера до клиента (рис. 4.43).

Рис. 4.43. Построение маршрута движения от курьера до клиента

Следует иметь в виду, что для работы методов преобразования координат в адрес и обратно на мобильном устройстве должен быть доступ в Интернет.

В качестве общей рекомендации по решению задач геопозиционирования можно посоветовать выбирать того провайдера, который максимально подходит для реализации поставленной задачи. Для этого следует проанализировать свойства используемого провайдера геопозиционирования: является ли он платным, использует ли сотовую сеть, использует ли спутник и т. п. В общем случае стоит выбирать провайдера, который обеспечивает минимальное энергопотребление и максимальную точность определения координат.

ПОДРОБНЕЕ

Документация 1С:Предприятие 8.3. Руководство разработчика. Глава 29. Разработка для мобильных устройств > Мобильная версия «1С:Предприятия» > Специальные возможности мобильного устройства > Средства геопозиционирования.

Телефония

Связь с клиентом

Теперь реализуйте команды для связи с клиентом. Для этого создайте обработчики команд Позвонить и ОтправитьСМС и заполните их следующим образом (листинги 4.28, 4.29).

Листинг 4.28. Обработчик команды «Позвонить»

```
&НаКлиенте
Процедура Позвонить(Команда)

    Если ЗначениеЗаполнено(Объект.Телефон) Тогда
        СредстваТелефонии.НабратьНомер(Объект.Телефон, Ложь);
    Иначе
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Не указан телефон клиента!";
        Сообщение.Поле  = "Объект.Телефон";
        Сообщение.Сообщить();
    КонецЕсли

КонецПроцедуры
```

Листинг 4.29. Обработчик команды «ОтправитьСМС»

```
&НаКлиенте
Процедура ОтправитьСМС(Команда)

    Если ЗначениеЗаполнено(Объект.Телефон) Тогда
        Сообщение = Новый SMSСообщение();
        Сообщение.Получатели.Добавить(Объект.Телефон);
        СредстваТелефонии.ПослатьSMS(Сообщение, Истина);
    Иначе
```

```
            Сообщение = Новый СообщениеПользователю();
            Сообщение.Текст = "Не указан телефон клиента";
            Сообщение.Поле  = "Объект.Телефон";
            Сообщение.Сообщить();
        КонецЕсли

КонецПроцедуры
```

В обработчиках обеих команд, в случае если телефон клиента заполнен, выполняется звонок или отправка сообщения клиенту с помощью объекта глобального контекста СредстваТелефонии, который предоставляет доступ к средствам телефонии на мобильном устройстве.

Для звонка клиенту в процедуре Позвонить() используется метод средств телефонии НабратьНомер(), в который передается телефон клиента. Поскольку вы передаете во втором параметре метода Ложь, при выполнении метода будет открыто приложение работы со звонками, в котором уже установлен заданный номер. Курьеру нужно будет только нажать кнопку начала вызова.

Для отправки СМС клиенту в процедуре ОтправитьСМС() используется метод средств телефонии ПослатьSMS(). Сначала создается объект SMSСообщение, затем в свойство Получатели этого объекта добавляется телефон клиента. После этого вызывается метод средств телефонии ПослатьSMS(), в который первым параметром передается объект SMSСообщение, а вторым Истина. При этом будет запущено системное приложение для отправки SMS-сообщений. Курьеру остается ввести текст сообщения и нажать кнопку отправки.

Напоминание о звонке (локальное уведомление)

Теперь реализуйте очень удобную возможность для курьера – создавать для себя напоминание о том, что ему надо позвонить клиенту в определенное время. Это напоминание появится в системе в указанные дату и время. При нажатии на это напоминание должна автоматически открыться форма звонка с переданными в нее номером телефона и текстом напоминания (см. рис. 4.46).

Для реализации описанной задачи вы должны сначала создать в системе локальное уведомление, информирующее пользователя о событии, произошедшем на конкретном мобильном устройстве. А затем соответствующим образом обработать это локальное уведомление в вашем мобильном приложении.

Для этого сначала создайте обработчик команды НапомнитьОЗвонке и заполните его следующим образом (листинг 4.30).

Листинг 4.30. Обработчик команды «НапомнитьОЗвонке»

```
&НаКлиенте
Асинх Процедура НапомнитьОЗвонке(Команда)

   Если ЗначениеЗаполнено(Объект.Телефон) Тогда
            Дата = ТекущаяДата();
            ДатаНапоминания = Ждать ВвестиДатуАсинх(Дата, "Введите время напоминания");
            Если НЕ ДатаНапоминания = Неопределено Тогда
                  Уведомление = Новый ДоставляемоеУведомление();
                  Уведомление.Текст = "Перезвоните клиенту: " +  Объект.Наименование;
                  Уведомление.Данные = Объект.Телефон;
                  Уведомление.ДатаПоявленияУниверсальноеВремя
                                                = УниверсальноеВремя(ДатаНапоминания);
                  ДоставляемыеУведомления.ДобавитьЛокальноеУведомление(Уведомление);
            КонецЕсли;
      Иначе
            // Сообщим пользователю о том, что информация не консистентна
            Сообщение = Новый СообщениеПользователю();
            Сообщение.Текст = "Телефон клиента не указан!";
            Сообщение.Поле  = "Объект.Телефон";
            Сообщение.Сообщить();
      КонецЕсли

КонецПроцедуры
```

В этом обработчике, если телефон клиента заполнен, вы вызываете диалог для ввода пользователем даты и времени напоминания. Для этого вы используете асинхронный метод глобального контекста ВвестиДатуАсинх(), который позволяет ввести дату, но не ожидает завершения ввода. В переменной ДатаНапоминания сохраняется выбранная пользователем дата или Неопределено, если он отказался от выбора.

Если дата выбрана, вы создаете объект ДоставляемоеУведомление, заполняете его свойство Текст как произвольный текст напоминания, Данные – как телефон клиента. В свойстве ДатаПоявленияУниверсальноеВремя устанавливаете дату напоминания, введенную пользователем, в формате универсального времени (UTC).

Затем с помощью свойства глобального контекста ДоставляемыеУведомления вы обращаетесь к менеджеру доставляемых уведомлений и методом ДобавитьЛокальноеУведомление() добавляете созданное вами локальное уведомление в систему.

Проверьте, как это работает на планшете. Откройте форму клиента, в меню действий раскройте группу команд Телефония и нажмите Напомнить о звонке. В результате появится диалог для ввода даты и времени напоминания (рис. 4.44).

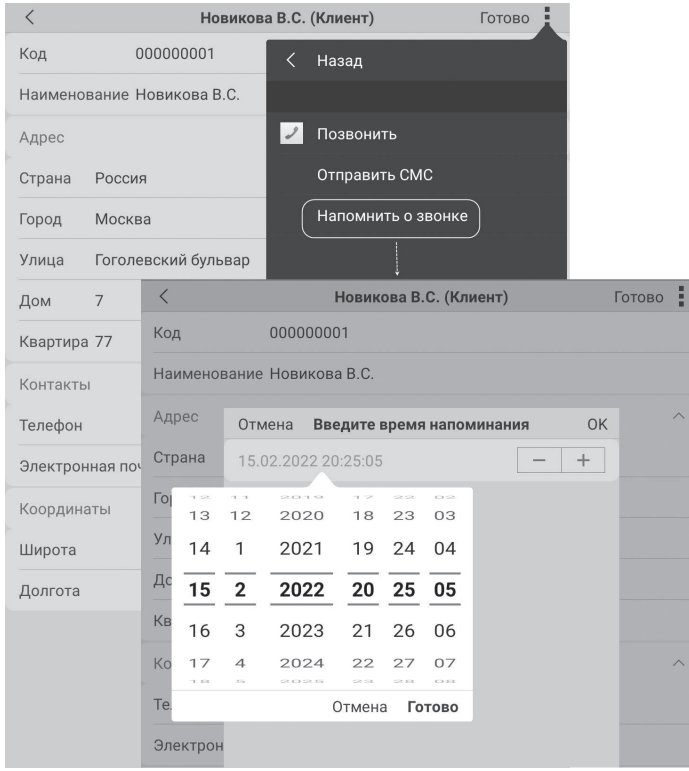


Рис. 4.44. Создание напоминания о звонке клиенту

После нажатия ОК в вашем мобильном приложении создается локальное уведомление, в него передается текст напоминания и телефон клиента – в результате это уведомление появится на мобильном устройстве в указанное курьером время.

После того как пользователь мобильного приложения нажмет на уведомление, будет активизировано мобильное приложение, создавшее уведомление, и в этом приложении будет вызван обработчик уведомления. В рамках этого обработчика вы и должны реализовать необходимую обработку уведомления, а именно – появление формы для звонка клиенту.

Для этого нужно подключить обработчик уведомления в модуле управляемого приложения в обработчике события ПриНачалеРаботыСистемы (листинг 4.31).

Листинг 4.31. Обработчик события «ПриНачалеРаботыСистемы»

```
Процедура ПриНачалеРаботыСистемы()

    // Подключение обработчика локальных и Push-уведомлений
    ОписаниеОповещения = Новый ОписаниеОповещения("ОбработкаУведомлений", УведомленияКлиент);
    ДоставляемыеУведомления.ПодключитьОбработчикУведомлений(ОписаниеОповещения);

КонецПроцедуры
```

В этом обработчике методом ПодключитьОбработчикУведомлений() менеджера доставляемых уведомлений в качестве обработчика уведомлений подключается процедура ОбработкаУведомлений(), находящаяся в общем модуле УведомленияКлиент и описанная в объекте ОписаниеОповещения.

Создайте клиентский общий модуль УведомленияКлиент, поместите в него процедуру ОбработкаУведомлений() и заполните ее следующим образом (листинг 4.32).

Листинг 4.32. Процедура «ОбработкаУведомлений()»

```
Процедура ОбработкаУведомлений(Уведомление, Локальное, Показано, Параметры) Экспорт

    Если НЕ Показано Тогда
        // Иначе пользователь оповещен об уведомлении системными средствами.
        СредстваМультимедиа.ВоспроизвестиЗвуковоеОповещение();
    КонецЕсли;

    Если Локальное Тогда
        ПараметрыФормы = Новый Структура("Текст, Данные", Уведомление.Текст, Уведомление.Данные);
        ОткрытьФорму("ОбщаяФорма.Звонок", ПараметрыФормы);
    КонецЕсли;

КонецПроцедуры
```

В этом обработчике сначала проверяется, что полученное уведомление локальное (в параметре Локальное передано Истина). Это условие нужно для того, чтобы в этом же обработчике можно было обрабатывать также

и Push-уведомления. Но в данной главе работа с Push-уведомлениями рассматриваться не будет (этот вопрос рассматривался в главе про мобильный клиент, в разделе «Работа с Push-уведомлениями»).

Если условие истинно, то создается структура параметров формы с полями Текст и Данные, которые заполняются соответственно текстом и данными полученного уведомления. Затем открывается общая форма Звонок и в нее передается структура параметров. Из этой формы и будет инициирован звонок клиенту.

Создайте в вашем мобильном приложении общую форму Звонок, добавьте в нее команду Позвонить и строковые реквизиты Текст и Данные для хранения соответствующих параметров формы. Для команды Позвонить в свойстве Картинка выберите картинку Позвонить из общих картинок вашей конфигурации и свойство команды Отображение установите как Картинка.

Реквизит Текст и команду Позвонить перетащите в окно элементов формы. В результате в конфигураторе форма звонка будет выглядеть следующим образом (рис. 4.45).

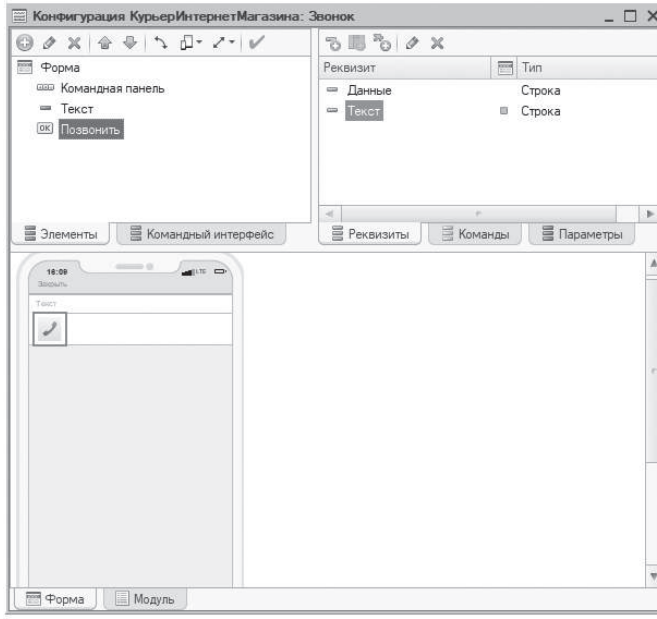


Рис. 4.45. Общая форма «Звонок» в редакторе формы

Чтобы при создании формы заполнить реквизиты формы соответствующими полями структуры параметров, переданной в форму, создайте обработчик события формы ПриСозданииНаСервере и заполните его следующим образом (листинг 4.33).

Листинг 4.33. Обработчик события формы «ПриСозданииНаСервере»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    Данные = Параметры.Данные;
    Текст = Параметры.Текст;

КонецПроцедуры
```

Таким образом, при создании формы в реквизитах Текст и Данные будут сохранены текст напоминания и телефон клиента, переданные в уведомлении.

Текст напоминания (содержимое реквизита Текст) будет просто отображаться в форме. Для команды Позвонить создайте обработчик и заполните его следующим образом (листинг 4.34).

Листинг 4.34. Обработчик команды «Позвонить»

```
&НаКлиенте
Процедура Позвонить(Команда)

    СредстваТелефонии.НабратьНомер(Данные, Ложь);

КонецПроцедуры
```

Для звонка клиенту в процедуре Позвонить()используется метод средств телефонии НабратьНомер(), в который передается телефон клиента в реквизите формы Данные.

Если приложение активно и в этот момент наступает время напоминания, то обработчик уведомлений будет вызван сразу же, без значка уведомлений и звукового сигнала. Поэтому, чтобы форма звонка не затерялась среди других форм (т. к. она немодальная), установите свойство формы РежимОткрытияОкна в значение Блокировать весь интерфейс.

В результате на экране поверх всех окон, блокируя весь остальной интерфейс, откроется форма звонка клиенту с установленным текстом напоминания. При нажатии на кнопку звонка будет открыто приложение работы со звонками, в котором уже установлен заданный номер. Курьеру останется только нажать кнопку начала вызова (рис. 4.46).

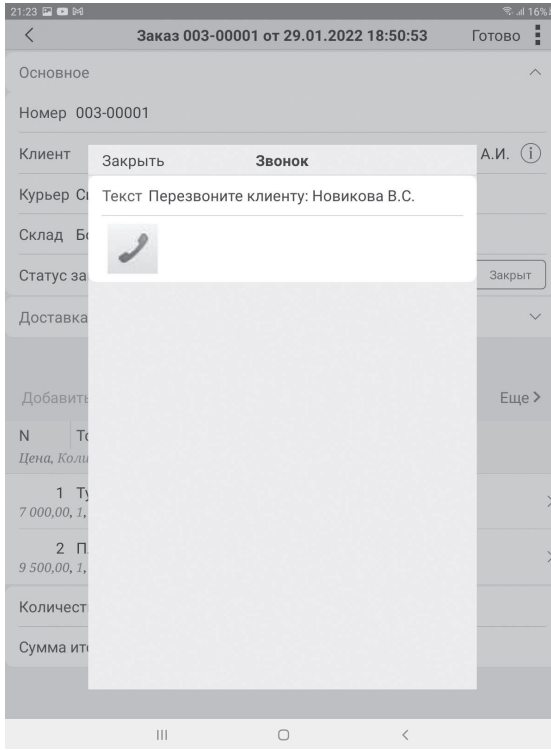


Рис. 4.46. Появление формы звонка клиенту

Если приложение неактивно, на планшете прозвучит сигнал, установленный пользователем (или же стандартный звуковой сигнал) для уведомлений, и в списке уведомлений планшета появится текст сообщения, который всплывет на несколько секунд (рис. 4.47). При нажатии на уведомление мобильное приложение будет активизировано.

Если вы вышли из приложения и находитесь в списке приложений мобильной платформы, то поверх него появится текст уведомления и будет предложено перейти к соответствующему приложению (рис. 4.48).

ПОДРОБНЕЕ

Документация 1С:Предприятие 8.3. Руководство разработчика. Глава 29. Разработка для мобильных устройств > Мобильная версия «1С:Предприятия» > Специальные возможности мобильного устройства > Работа с уведомлениями > Локальные уведомления.

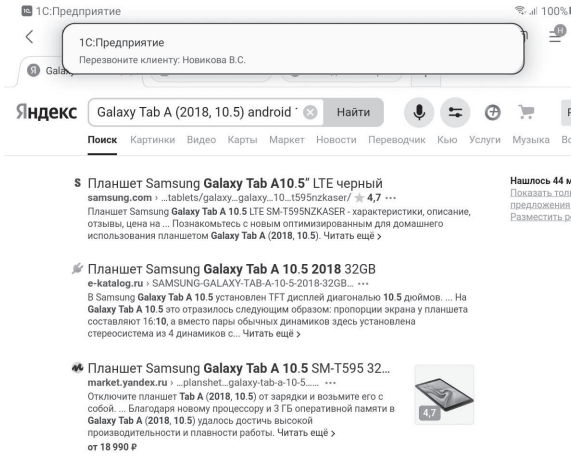


Рис. 4.47. Появление уведомления о звонке клиенту

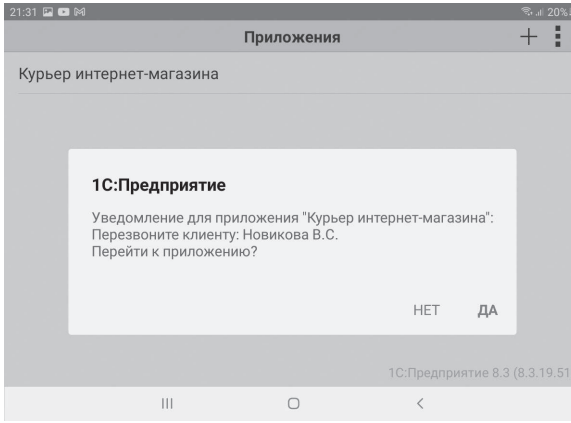


Рис. 4.48. Появление уведомления о звонке клиенту

В заключение при открытии формы клиента установите видимость команд для работы с клиентами только в том случае, если мобильное устройство поддерживает работу с телефонией и отправку почты. Для этого создайте обработчик события формы ПриОткрытии и заполните его следующим образом (листинг 4.35).

Листинг 4.35. Обработчик события формы «ПриОткрытии»

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)

    Элементы.ФормаПозвонить.Доступность = СредстваТелефонии.ПоддерживаетсяНаборНомера();
    Элементы.ФормаОтправитьСМС.Доступность = СредстваТелефонии
                                                .ПоддерживаетсяОтправкаSMS(Истина);
    Элементы.ФормаОтправитьПисьмо.Доступность = СредстваПочты.ПоддерживаетсяОтправка();
    Элементы.ФормаПоказатьНаКарте.Доступность = ПоддерживаетсяОтображениеКарты();

КонецПроцедуры
```

И аналогичную проверку на возможность звонка нужно добавить в форму Звонок в обработчик события формы ПриОткрытии (листинг 4.36).

Листинг 4.36. Обработчик события формы «ПриОткрытии» формы «Звонок»

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)

    Элементы.Позвонить.Доступность = СредстваТелефонии.ПоддерживаетсяНаборНомера();

КонецПроцедуры
```

Электронная почта

Если телефон клиента по каким-то причинам недоступен, то курьер должен иметь возможность отправить клиенту письмо по электронной почте. Чтобы реализовать такую возможность, создайте обработчик команды ОтправитьПисьмо и заполните его следующим образом (листинг 4.37).

Листинг 4.37. Обработчик команды «ОтправитьПисьмо»

```
&НаКлиенте
Процедура ОтправитьПисьмо(Команда)

    Если ЗначениеЗаполнено(Объект.ЭлектроннаяПочта) Тогда
        Сообщение = Новый ИнтернетПочтовоеСообщение;
        Сообщение.Тема = "Сообщение о заказе";
        Сообщение.Получатели.Добавить(СокрЛП(Объект.ЭлектроннаяПочта));
        Текст = Сообщение.Тексты.Добавить("Уважаемый клиент, " + СокрЛП(Объект.Наименование) + "!");
        Текст.ТипТекста = ТипТекстаПочтовогоСообщения.ПростойТекст;
        СредстваПочты.Послать(Сообщение);
    Иначе
        // Сообщим пользователю о том, что информация не консистентна
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "Электронный адрес не указан!";
        Сообщение.Поле = "Объект.ЭлектроннаяПочта";
        Сообщение.Сообщить();
    КонецЕсли

КонецПроцедуры
```

В этом обработчике, если адрес электронной почты клиента заполнен, вы создаете объект ИнтернетПочтовоеСообщение, заполняете его свойства Тема и Тексты (текст не должен содержать элементы форматирования!). В свойство Получатели почтового сообщения добавляете адрес электронной почты клиента (Объект.ЭлектроннаяПочта).

Затем с помощью свойства глобального контекста СредстваПочты вы получаете доступ к менеджеру средств встроенной почты мобильной платформы. Для отправки созданного вами почтового сообщения вы используете метод менеджера Послать() и передаете в него сообщение в качестве параметра.

В результате при нажатии Отправить письмо в меню действий в форме клиента запускается почтовое приложение для отправки электронной почты (E-mail, Gmail и т. п.), в котором уже заполнены адрес получателя, тема и частично текст (рис. 4.49).

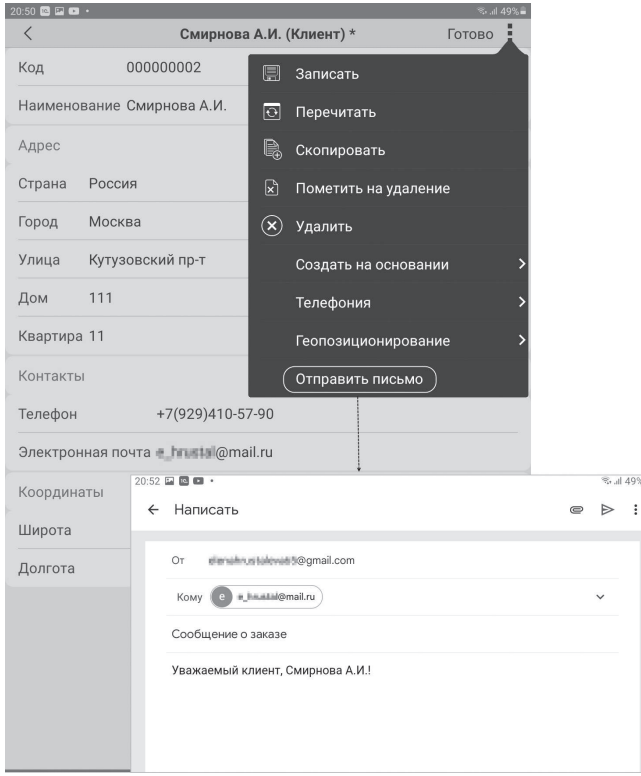


Рис. 4.49. Отправка электронной почты клиенту

Дальше курьер может работать с этим почтовым приложением, как обычно (вложить файлы, дополнить текст сообщения и т. д.), и отправить письмо кнопкой Отправить.

Отбор в списке заказов

Ранее вы уже дорабатывали форму списка заказов, но занимались только ее внешним видом в разделе «Доработка форм заказа». Теперь сделайте этот список более удобным в работе и информативным для курьера за счет отбора заказов с установленным признаком важности.

Причем отбор будет выполняться программным образом, так как у пользователя в мобильном приложении нет возможности выполнять интерактивную настройку динамического списка.

Поместите в форму флажок, при установке которого будет выполняться программный отбор важных заказов. Для этого создайте реквизит формы Важные типа Число и перетащите его в окно элементов формы над таблицей списка. У элемента формы Важные установите свойства: Вид – Поле флажка и Заголовок – Показывать только важные. В результате форма списка заказов в конфигураторе будет выглядеть следующим образом (рис. 4.50).

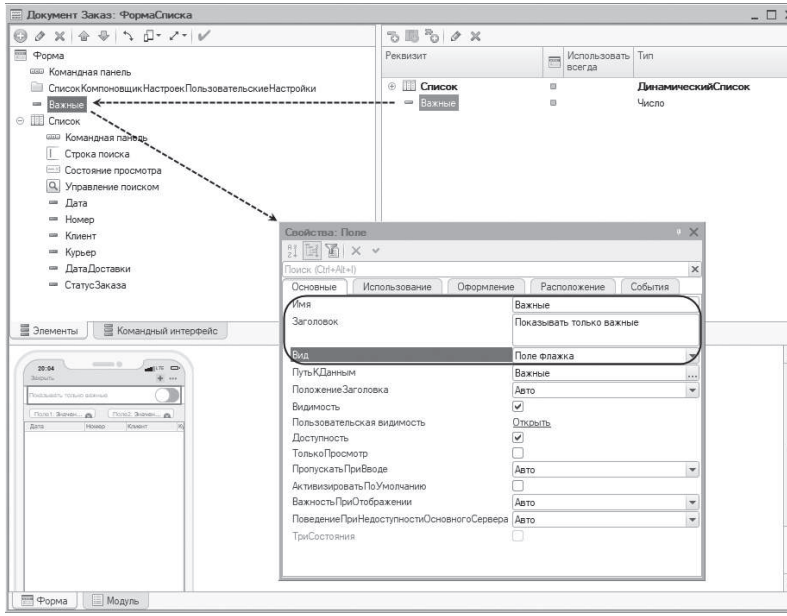


Рис. 4.50. Форма списка заказов в редакторе формы

Затем для поля флажка Важные создайте обработчик события ПриИзменении и заполните его следующим образом (листинг 4.38).

Листинг 4.38. Обработчик события «ПриИзменении» для поля формы «Важные»

```
&НаКлиенте
Процедура ВажныеПриИзменении(Элемент)

    Если Важные = 1 Тогда
        УстановитьФильтрПоВажнымЗаказам();
    Иначе
        УдалитьФильтрПоВажнымЗаказам();
    КонецЕсли;

КонецПроцедуры
```

В этом обработчике при изменении состояния флажка вызываются процедуры УстановитьФильтрПоВажнымЗаказам() (листинг 4.39) или УдалитьФильтрПоВажнымЗаказам(), листинг 4.40.

Листинг 4.39. Процедура «УстановитьФильтрПоВажнымЗаказам()»

```
&НаКлиенте
Процедура УстановитьФильтрПоВажнымЗаказам()

    Настройки = Список.КомпоновщикНастроек.Настройки;

    ЭлементОтбора = Настройки.Отбор.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
    ЭлементОтбора.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("Важность");
    ЭлементОтбора.ВидСравнения = ВидСравненияКомпоновкиДанных.Равно;
    ЭлементОтбора.ПравоеЗначение = Истина;

    Список.КомпоновщикНастроек.ЗагрузитьНастройки(Настройки);

КонецПроцедуры
```

В процедуре установки фильтра вы сначала получаете настройки динамического списка с помощью его свойства КомпоновщикНастроек. Затем в коллекцию настроек добавляете элемент отбора по полю Важность (левое значение) и сравниваете ее с булевым значением Истина (правое значение). И загружаете новые настройки в компоновщик настроек динамического списка методом ЗагрузитьНастройки().

Листинг 4.40. Процедура «УдалитьФильтрПоВажнымЗаказам()»

```
&НаКлиенте
Процедура УдалитьФильтрПоВажнымЗаказам()

    Настройки = Список.КомпоновщикНастроек.Настройки;
    Для Каждого ЭлементНастроек Из Настройки.Отбор.Элементы Цикл
        Если ЭлементНастроек.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("Важность") Тогда
```

```
                Настройки.Отбор.Элементы.Удалить(ЭлементНастроек);
            КонецЕсли;
        КонецЦикла;
        Список.КомпоновщикНастроек.ЗагрузитьНастройки(Настройки);

КонецПроцедуры
```

В процедуре удаления фильтра вы сначала получаете настройки динамического списка с помощью его свойства КомпоновщикНастроек. Затем обходите в цикле коллекцию элементов отбора этих настроек. Если в коллекции присутствует элемент отбора, в котором ЛевоеЗначение – поле компоновки данных Важность, то этот элемент удаляется. Затем измененные настройки загружаются в компоновщик настроек.

Таким образом, открыв список заказов и установив флажок Показывать только важные, курьер может отобрать только важные заказы из общего списка заказов (рис. 4.51).

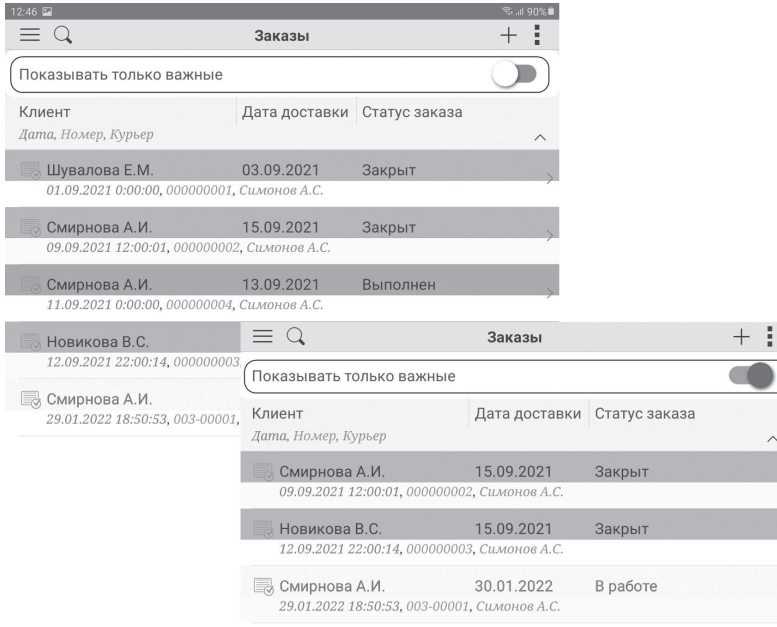


Рис. 4.51. Отбор по важности в списке заказов

Обслуживание заказов

Как уже говорилось, курьер не может редактировать документ ОбслуживаниеЗаказов, поскольку этот документ формируется в самом интернет-магазине и определяет порядок обслуживания заказов. Поэтому форма документа открывается только в режиме просмотра, и из табличной части документа Обслуживание заказов нельзя открыть заказ.

Однако для удобства работы курьера надо предусмотреть такую возможность. Например, вероятна следующая последовательность действий (рис. 4.52):

- курьер открывает список обслуживания заказов;
- выбирает из списка конкретный документ и открывает его;
- в табличной части документа Обслуживание заказов курьер вызывает контекстное меню конкретного заказа, выполняет команду Открыть заказ и открывает форму выбранного заказа, в которой может работать с этим заказом.

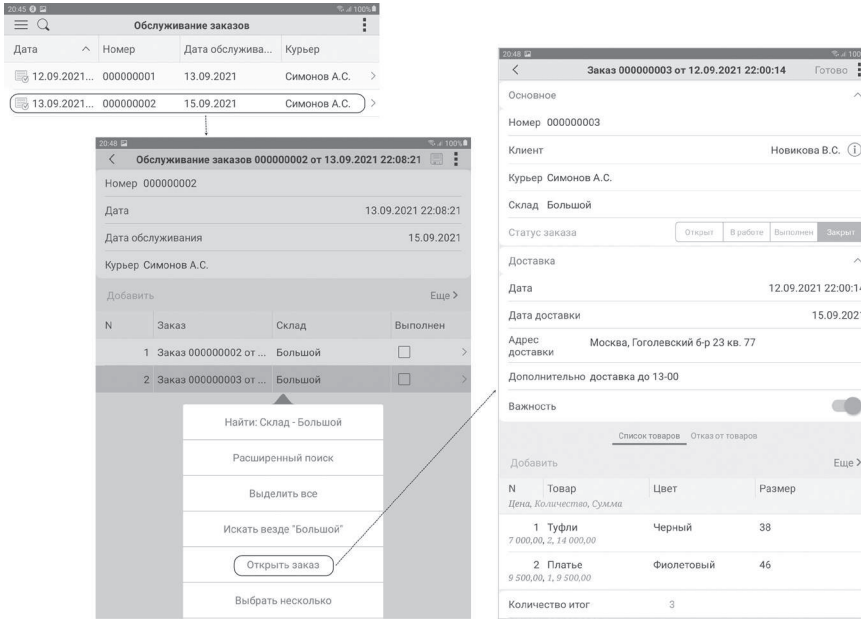


Рис. 4.52. Открытие заказа из документа «Обслуживание заказов»

Чтобы предусмотреть такую возможность, создайте форму документа ОбслуживаниеЗаказов, добавьте в нее команду ОткрытьЗаказ и перетащите ее в командную панель формы. Установите свойство этой команды Использование текущей строки в значение Использует и в свойстве Используемая таблица выберите таблицу с данными табличной части документа Заказы.

С помощью этих свойств вы устанавливаете, что команда будет использовать данные текущей строки таблицы Заказы. Эти данные нужны для получения ссылки на заказ, который требуется открыть.

Затем создайте обработчик этой команды и заполните его следующим образом (листинг 4.41).

Листинг 4.41. Обработчик команды «ОткрытьЗаказ»

```
&НаКлиенте
Процедура ОткрытьЗаказ(Команда)

    ПоказатьЗначение(, Элементы.Заказы.ТекущиеДанные.Заказ);

КонецПроцедуры
```

В этом обработчике, используя свойство ТекущиеДанные таблицы Заказы, вы получаете объект, содержащий данные, находящиеся в текущей строке таблицы. Обращаясь через точку к полю этого объекта Заказ, вы получаете ссылку на конкретный заказ, содержащуюся в этом поле. И затем передаете эту ссылку в метод глобального контекста ПоказатьЗначение(), который открывает форму выбранного заказа.

Отчеты

При необходимости курьер может сформировать и проанализировать информацию в отчетах. Причем эти отчеты могут быть построены как на данных мобильного приложения, так и удаленно в офисном приложении и переданы курьеру через веб-сервис.

На момент написания книги система компоновки данных доступна в мобильной платформе, но без интерактивных компонентов. Поэтому вы не можете в своем мобильном приложении создавать и настраивать отчет так же, как в офисной конфигурации. Однако вы можете использовать созданный в конфигураторе макет – схему компоновки данных – и формировать отчеты на ее основе. Этот вариант будет показан в разделе «Отчет по данным мобильного приложения».

Если же в мобильном приложении нет данных для формирования нужного курьеру отчета, то с помощью веб-сервиса он может удаленно сформировать этот отчет в офисном приложении и получить готовый отчет на свое мобильное устройство. При условии, конечно, что у него есть права на этот отчет. Этот вариант будет показан в разделе «Формирование удаленного отчета по данным офисного приложения».

Отчет по данным мобильного приложения

Предположим, что курьер хочет проанализировать данные заказов, уже находящиеся в его мобильном приложении. Например, сформировать диаграмму (в виде гистограммы), построенную на основе данных о заказанных товарах в разрезе их цветов. Таким образом можно, например, легко проанализировать, какой товар и какого цвета пользуется наибольшим спросом.

Для реализации поставленной задачи добавьте в вашу конфигурацию обработку ЗаказыТоваров. Затем создайте основную форму обработки, добавьте в нее команду ВыполнитьОтчет (с заголовком Сформировать) и перетащите ее в командную панель формы.

А также добавьте в форму реквизит Содержимое типа ТабличныйДокумент, перетащите его в дерево элементов формы и установите у получившегося поля формы свойство ПоложениеЗаголовка в значение Нет.

После этого создайте у обработки ЗаказыТоваров макет типа Схема компоновки данных, получающий данные из документов Заказ на основе следующего запроса (листинг 4.42).

Листинг 4.42. Запрос для получения данных для отчета «ЗаказыТоваров»

```
ВЫБРАТЬ
	ЗаказТовары.Товар КАК Товар,
	ЗаказТовары.Цвет КАК Цвет,
	ЗаказТовары.Размер КАК Размер,
	ЗаказТовары.Количество КАК Количество
ИЗ
	Документ.Заказ.Товары КАК ЗаказТовары
ГДЕ
	(ЗаказТовары.Ссылка.СтатусЗаказа = ЗНАЧЕНИЕ(Перечисление.СтатусыЗаказов.Выполнен)
		ИЛИ ЗаказТовары.Ссылка.СтатусЗаказа = ЗНАЧЕНИЕ(Перечисление.СтатусыЗаказов.Закрыт))
```

В этом запросе из табличной части Товары документов Заказ выбираются поля Товар, Цвет, Размер, Количество. В выборку отбираются только заказы, уже обслуженные курьером (со статусом Выполнен или Закрыт).

В ресурсы отчета добавьте поле Количество.

В настройки отчета добавьте диаграмму. В точки диаграммы добавьте группировку по полю Товар, в серии – группировку по полю Цвет, а в выбранные поля отчета добавьте поле Количество (рис. 4.53).

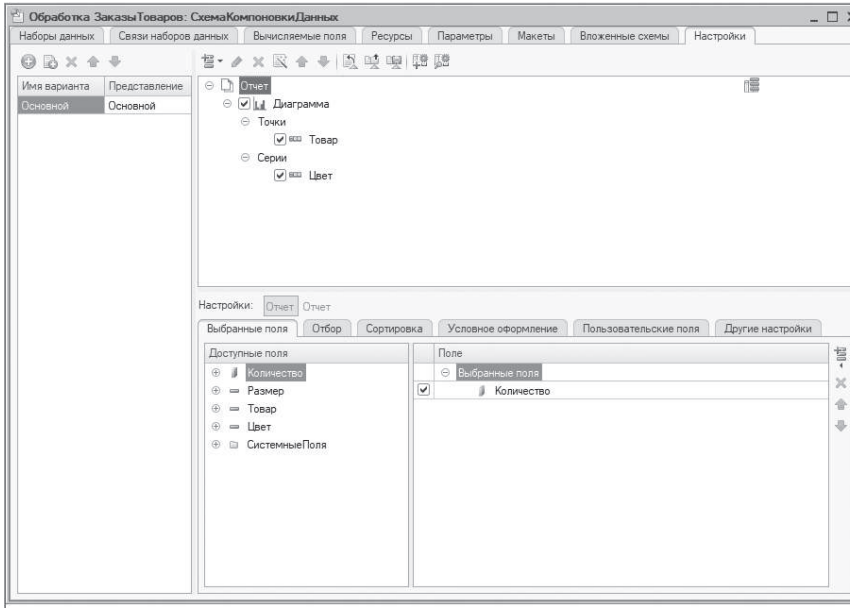


Рис. 4.53. Настройки отчета «ЗаказыТоваров»

Затем создайте обработчик команды ВыполнитьОтчет и заполните его следующим образом (листинг 4.43).

Листинг 4.43. Обработчик команды «ВыполнитьОтчет»

```
&НаКлиенте
Процедура ВыполнитьОтчет(Команда)

    ВыполнитьОтчетНаСервере();

КонецПроцедуры
```

В этом обработчике вызывается серверная процедура ВыполнитьОтчетНаСервере(), в которой, собственно, и формируется сам отчет (листинг 4.44).

Листинг 4.44. Процедура «ВыполнитьОтчетНаСервере()»

```
&НаСервере
Процедура ВыполнитьОтчетНаСервере()

	Перем ДанныеРасшифровкиОбъект;

	Содержимое.Очистить();

	СхемаКомпоновкиДанных = Обработки.ЗаказыТоваров.ПолучитьМакет("СхемаКомпоновкиДанных");
	НастройкиКомпоновкиДанных = СхемаКомпоновкиДанных.НастройкиПоУмолчанию;

	КомпоновщикМакета = Новый КомпоновщикМакетаКомпоновкиДанных;
	МакетКомпоновкиДанных = КомпоновщикМакета.Выполнить(СхемаКомпоновкиДанных
		, НастройкиКомпоновкиДанных, ДанныеРасшифровкиОбъект);

	ПроцессорКомпоновкиДанных = Новый ПроцессорКомпоновкиДанных;
	ПроцессорКомпоновкиДанных.Инициализировать(МакетКомпоновкиДанных
		, , ДанныеРасшифровкиОбъект);

	ПроцессорВывода = Новый ПроцессорВыводаРезультатаКомпоновкиДанныхВТабличныйДокумент;
	ПроцессорВывода.УстановитьДокумент(Содержимое);
	ПроцессорВывода.Вывести(ПроцессорКомпоновкиДанных);

КонецПроцедуры
```

В этой процедуре реализуются программное формирование отчета и вывод результата в табличный документ (элемент формы обработки Содержимое).

Сначала содержимое табличного документа очищается. Затем методом глобального контекста ПолучитьМакет() получается макет СхемаКомпоновкиДанных обработки ЗаказыТоваров, и для этой схемы компоновки данных получаются стандартные настройки по умолчанию.

После этого создается объект КомпоновщикМакетаКомпоновкиДанных и с помощью его метода Выполнить() производится компоновка макета компоновки данных на основе ранее полученных схемы компоновки данных и стандартных настроек. Третьим параметром в этот метод передается переменная ДанныеРасшифровкиОбъект, в которую будет помещено значение расшифровки.

Затем создается объект ПроцессорКомпоновкиДанных и с помощью метода Инициализировать() инициализируется макетом компоновки данных и переменной, в которую надо поместить данные расшифровки.

После этого создается объект ПроцессорВыводаРезультатаКомпоновкиДанныхВТабличныйДокумент и с помощью его метода УстановитьДокумент() устанавливается табличный документ (элемент формы обработки Содержимое), куда нужно выводить результат компоновки данных. Для этого используется метод Вывести() процессора вывода, в качестве параметра которому передается процессор компоновки данных.

В завершение создайте обработчик события формы ПриСозданииНаСервере и вставьте туда вызов процедуры ВыполнитьОтчетНаСервере(), чтобы отчет формировался сразу при открытии формы обработки, без нажатия кнопки Сформировать (листинг 4.45).

Листинг 4.45. Обработчик события формы «ПриСозданииНаСервере()»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    ВыполнитьОтчетНаСервере();

КонецПроцедуры
```

Проверьте, как работает ваш отчет на планшете. Обновите конфигурацию базы данных, запустите мобильное приложение и нажмите Заказы товаров. В результате заказанные товары в разрезе цветов будут показаны в виде гистограммы (рис. 4.54).

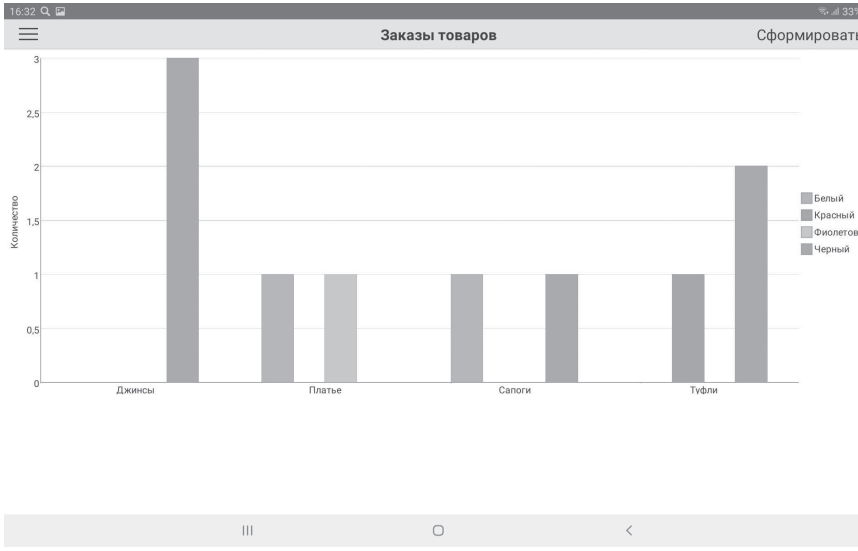


Рис. 4.54. Обработка «ЗаказыТоваров»

Формирование удаленного отчета по данным офисного приложения

В офисном приложении заказы, полученные от курьера, проводятся и формируют движения в регистрах накопления. Эти регистры служат источниками данных для различных отчетов. Хотя исходной информации у курьера на планшете нет, он может выполнить нужный ему отчет удаленно в офисном приложении и получить его через веб-сервис себе на планшет.

Например, курьеру необходимо получить отчет об остатках товаров на складах, построенный на данных регистра накопления ОстаткиТоваров, которого нет в мобильном приложении.

Для этого можно использовать операцию ПолучитьОтчет веб-сервиса, существующего в офисном приложении, с помощью которого оно экспортирует свою функциональность. В эту функцию передаются даты начала и окончания отчетного периода и строка, куда будет записана информация расшифровки. В этой функции по переданным параметрам строится отчет ОстаткиТоваровНаСкладах и в виде объекта XDTO возвращается обратно, а также заполняется информация расшифровки отчета.

Для получения отчета в мобильном приложении добавьте в конфигурацию КурьерИнтернетМагазина обработку ОстаткиТоваров, в которой будет отображаться ваш отчет. В форме обработки добавьте команду Сформировать и перетащите ее в командную панель формы.

А также добавьте реквизиты формы ДатаНачала и ДатаОкончания, в которых будут храниться даты отчетного периода. Добавьте в форму группу без отображения с типом группировки Горизонтальная если возможно и перетащите туда эти реквизиты.

Затем добавьте реквизит Содержимое типа ТабличныйДокумент, который будет содержать данные отчета. Перетащите его в форму и установите у соответствующего элемента формы свойство Положение заголовка в значение Нет.

Кроме того, добавьте реквизит ИнформацияРасшифровки типа Строка – он будет содержать адрес во временном хранилище, по которому будут храниться данные о расшифровке отчета.

В результате форма обработки в конфигураторе примет следующий вид (рис. 4.55).

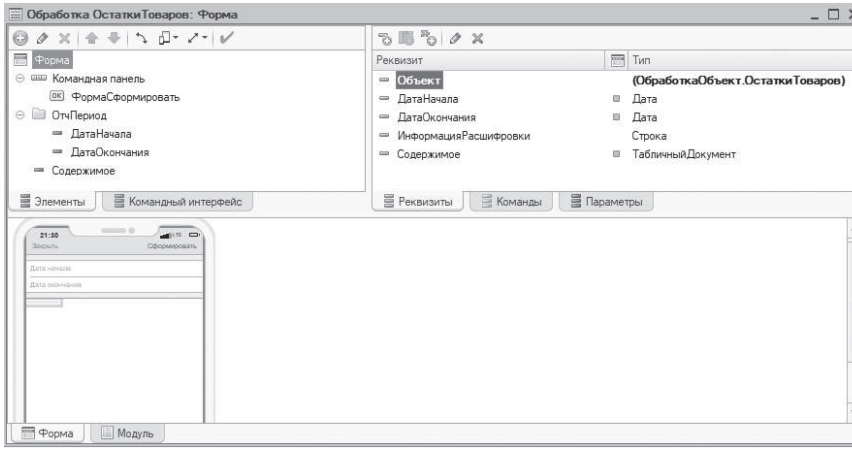


Рис. 4.55. Форма обработки «ОстаткиТоваров» в конфигураторе

Затем создайте обработчик команды Сформировать и заполните его следующим образом (листинг 4.46).

Листинг 4.46. Обработчик команды «Сформировать»

```
&НаКлиенте
Процедура Сформировать(Команда)

	ТекстОшибки = ПолучитьОтчет();
	Если ТекстОшибки <> "" Тогда
		ПоказатьПредупреждение(, ТекстОшибки);
	КонецЕсли;

КонецПроцедуры
```

В этом обработчике вызывается серверная функция ПолучитьОтчет(), которая получает отчет и возвращает текст ошибки (листинг 4.47). Если текст ошибки заполнен, то он выводится пользователю.

Листинг 4.47. Функция для получения отчета

```
&НаСервере
Функция ПолучитьОтчет()

	Если ИнформацияРасшифровки <> "" Тогда
		УдалитьИзВременногоХранилища(ИнформацияРасшифровки);
	КонецЕсли;

	ДанныеРасшифровки = Неопределено;
	ТекстОшибки = "";
```

```
    Прокси = Обмен.ПолучитьПрокси(ТекстОшибки);
    Если Прокси = Неопределено Тогда
        Возврат ТекстОшибки;
    КонецЕсли;

    СериализаторXDTO = Новый СериализаторXDTO(Прокси.ФабрикаXDTO);
    ИнформацияРасшифровкиXDTO = Неопределено;
    ДокументXDTO = Прокси.ПолучитьОтчет(ДатаНачала, ДатаОкончания, ИнформацияРасшифровкиXDTO);
    ДанныеРасшифровки = Сериализатор.ПрочитатьXDTO(ИнформацияРасшифровкиXDTO);

    Содержимое = Сериализатор.ПрочитатьXDTO(ДокументXDTO);

    Если ДанныеРасшифровки = Неопределено Тогда
        ИнформацияРасшифровки = "";
    Иначе
        ИнформацияРасшифровки = ПоместитьВоВременноеХранилище(
                                        ДанныеРасшифровки, УникальныйИдентификатор);
    КонецЕсли;

    Возврат ТекстОшибки;

КонецФункции
```

В этой функции вы сначала удаляете из временного хранилища данные расшифровки отчета, если они уже были раньше заполнены.

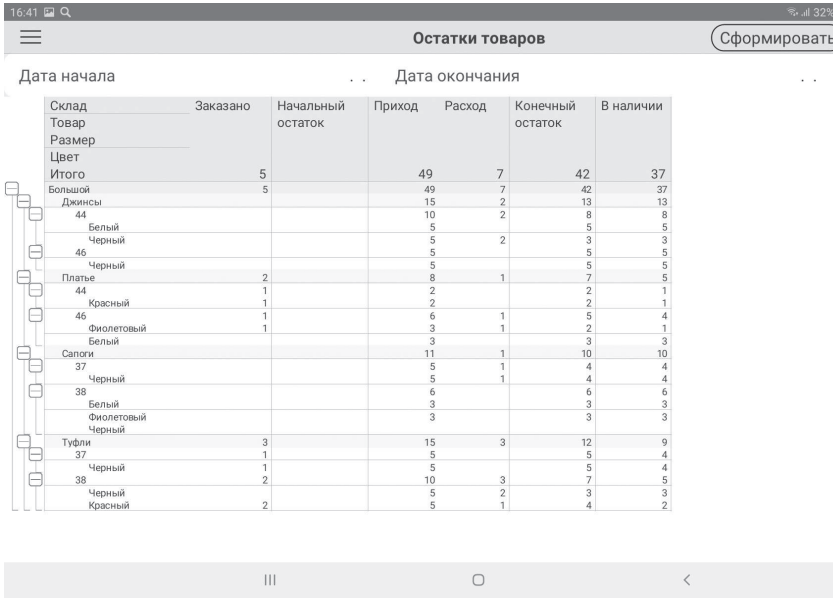
Затем вы подключаетесь к Web-сервису с помощью функции ПолучитьПрокси() общего модуля Обмен, как было подробно объяснено в разделе про обмен данными через веб-сервис (см. листинг 4.3).

После этого вы создаете объект СериализаторXDTO на основе фабрики XDTO, содержащей определения всех типов XDTO, доступных этому Web-сервису. Этот объект нужен для чтения данных XDTO, возвращенных при выполнении отчета, и последующей сериализации данных XML в значения платформы «1С:Предприятие».

Затем выполняется операция Web-сервиса ПолучитьОтчет, в которой в офисном приложении по переданным параметрам строится отчет ОстаткиТоваровНаСкладах и в виде объекта XDTO возвращается обратно. А также выходной параметр ИнформацияРасшифровкиXDTO операции Web-сервиса ПолучитьОтчет заполняется данными расшифровки отчета.

После этого данные XDTO читаются с помощью метода ПрочитатьXDTO() объекта Сериализатор. В итоге реквизит формы Содержимое будет содержать результат отчета в виде табличного документа, а переменная ДанныеРасшифровки будет содержать информацию о расшифровке отчета – ее вы поместите во временное хранилище, адрес которого будет сохранен в реквизите формы ИнформацияРасшифровки.

Проверьте, как работает ваш отчет. На начальной странице мобильного приложения вызовите обработку Остатки товаров, заполните даты отчетного периода (их можно оставить пустыми) и нажмите Сформировать. В результате отчет примет следующий вид (рис. 4.56).



Склад / Товар / Размер / Цвет	Заказано	Начальный остаток	Приход	Расход	Конечный остаток	В наличии
Итого	5		49	7	42	37
Большой	5		49	7	42	37
Джинсы			15	2	13	13
44			10	2	8	8
Белый			5		5	5
Черный			5	2	3	3
46			5		5	5
Черный			5		5	5
Платье	2		8	1	7	5
44	1		2		2	1
Красный	1		2		2	1
46	1		6	1	5	4
Фиолетовый	1		3	1	2	1
Белый			3		3	3
Сапоги			11	1	10	10
37			5	1	4	4
Черный			5	1	4	4
38			6		6	6
Белый			3		3	3
Фиолетовый			3		3	3
Черный						
Туфли	3		15	3	12	9
37	1		5		5	4
Черный	1		5		5	4
38	2		10	3	7	5
Черный			5	2	3	3
Красный	2		5	1	4	2

Рис. 4.56. Результат отчета «Остатки товаров»

Теперь вам осталось реализовать стандартное поведение результата отчета, при котором, выбрав поле табличного документа быстрым касанием, можно открыть его значение.

Для этого создайте для элемента формы Содержимое обработчик события ОбработкаРасшифровки и заполните его следующим образом (листинг 4.48).

Листинг 4.48. Обработчик события «ОбработкаРасшифровки» для поля формы «Содержимое»

```
&НаКлиенте
Процедура СодержимоеОбработкаРасшифровки(Элемент, Расшифровка, СтандартнаяОбработка)

    СтандартнаяОбработка = Ложь;
    Если ИнформацияРасшифровки <> "" Тогда
        ДанныеРасшифровки = ПолучитьИзВременногоХранилища(ИнформацияРасшифровки);
```

```
            Если ДанныеРасшифровки <> Неопределено Тогда
                Значение = ДанныеРасшифровки.Получить(Расшифровка);
                Если Значение <> Неопределено И Значение <> Null Тогда
                    ПоказатьЗначение(, Значение);
                КонецЕсли;
            КонецЕсли;
        КонецЕсли;

КонецПроцедуры
```

В этом обработчике вы сначала отменяете стандартную обработку расшифровки табличного документа, так как это не отчетная форма, а форма обработки.

Затем вы получаете данные расшифровки в виде соответствия из временного хранилища по адресу, сохраненному в реквизите формы ИнформацияРасшифровки.

После этого значение расшифровки получается из соответствия ДанныеРасшифровки по ключу, который передается в параметре события Расшифровка, и открывается немодальным методом глобального контекста ПоказатьЗначение().

Итак, выполните отчет, быстрым касанием выберите одну из ячеек (например, с названием товара) и нажмите Расшифровать в контекстном меню выделенной ячейки. В результате будет открыта форма товара, название которого отображено в выбранной ячейке отчета (рис. 4.57).

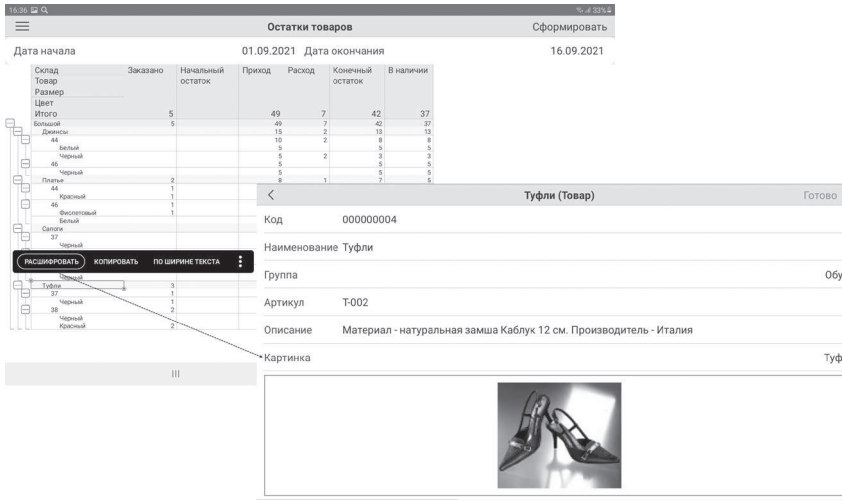


Рис. 4.57. Расшифровка ячейки отчета

Глава 5. Сервис сборки и публикации мобильных приложений

Завершающим этапом разработки является сборка разработанного мобильного приложения в установочный файл для загрузки на мобильное устройство или публикация его в магазине приложений.

Для облегчения этой задачи фирмой «1С» разработан облачный сервис сборки и публикации мобильных приложений, в котором настроено и поддерживается в актуальном виде все необходимое программное обеспечение.

Этот сервис доступен начиная с версии платформы «1С:Предприятие» 8.3.20. В настоящее время использование сервиса является бесплатным для разработчиков, зарегистрированных на портале developer.1c.ru.

С помощью этого сервиса можно собрать мобильное приложение за несколько минут, не погружаясь глубоко в нюансы сборки мобильных приложений.

Процесс подготовки приложения к сборке реализован с помощью мастера, который последовательно проводит вас по всем этапам сборки. При этом на каждом шаге выдается только та информация, которая нужна в данный момент, что делает сборку понятной и простой.

Вы можете собрать приложения для устройств, работающих под управлением Android и iOS, причем для сборки приложения под iOS вам не нужен компьютер с macOS. Для сборки и публикации мобильных приложений нужен только компьютер с установленной платформой «1С:Предприятие» и доступ в Интернет.

Далее будет показан пример сборки приложения мобильного клиента с автономным режимом для устройств, работающих на платформе Android. Разработка приложений этого вида рассматривалась в третьей главе книги. В результате вы получите установочный apk-файл, предназначенный для корпоративного распространения.

Итак, для работы с сервисом запустите любую (можно даже и пустую) информационную базу в режиме «1С:Предприятие», вызовите из системного меню Функции для технического специалиста и запустите стандартную обработку Сервис сборки мобильных приложений. Затем авторизуйтесь (или зарегистрируйтесь, если у вас еще нет учетной записи) на сайте developer.1c.ru в сервисе (рис. 5.1).

Рис. 5.1. Начало работы с сервисом сборки мобильных приложений

После этого вы увидите все собранные вами приложения в верхней таблице Приложения. При выделении строки с конкретным приложением в нижней таблице Сборки будет показан список всех сделанных вами сборок этого приложения. Для начала сборки нового приложения нажмите Создать приложение над верхней таблицей (рис. 5.2).

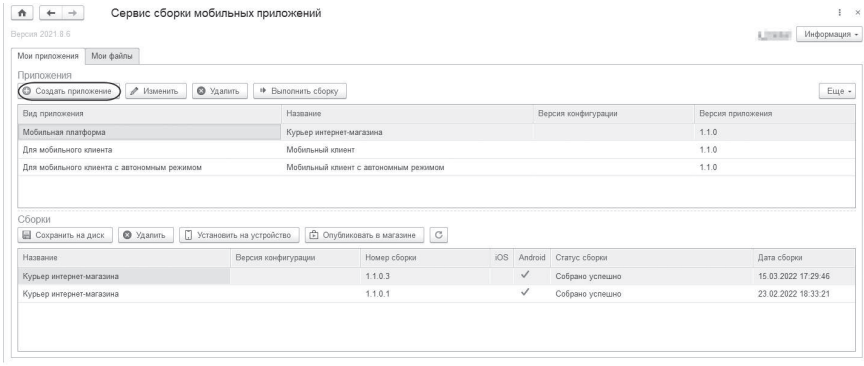


Рис. 5.2. Сервис сборки мобильных приложений

Откроется мастер сборки Параметры мобильного приложения, и будет активизирована его первая страница Начало. Познакомившись со справочной информацией и заполнив все требующиеся параметры сборки, нажмите Далее на каждом шаге мастера или же просто активизируйте следующую страницу сборщика в списке закладок слева.

На странице Начало выберите Упрощенный режим мастера настройки (рис. 5.3).

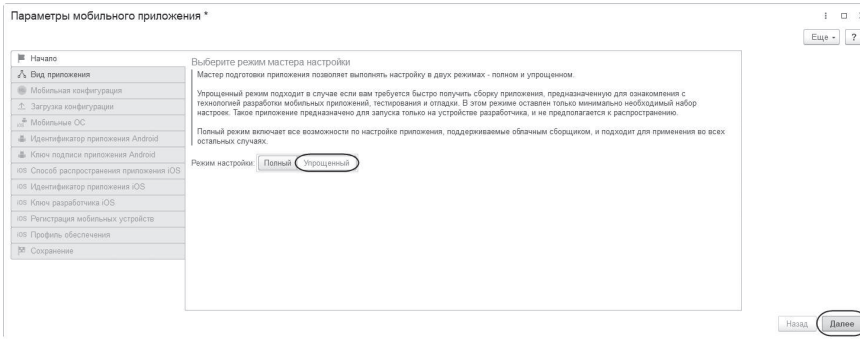


Рис. 5.3. Выбор режима настройки

На странице Вид приложения укажите вид собираемого приложения – Мобильный клиент (рис. 5.4).

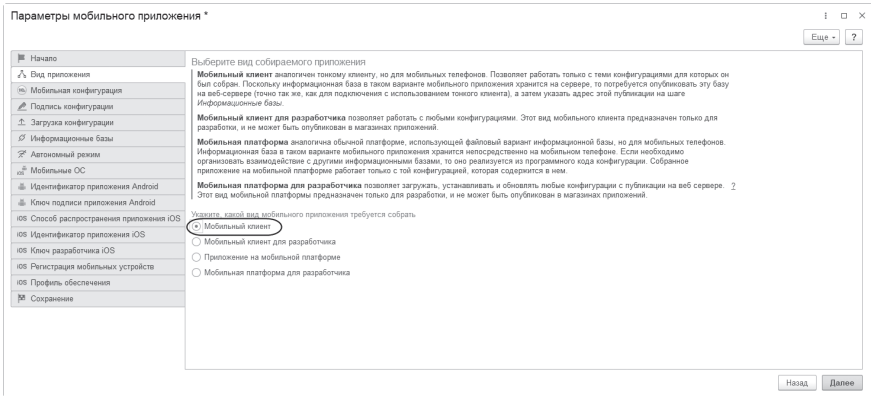


Рис. 5.4. Выбор вида собираемого приложения

На странице Мобильная конфигурация выберите способ получения мобильной конфигурации – Получить автоматически из информационной базы (рис. 5.5).

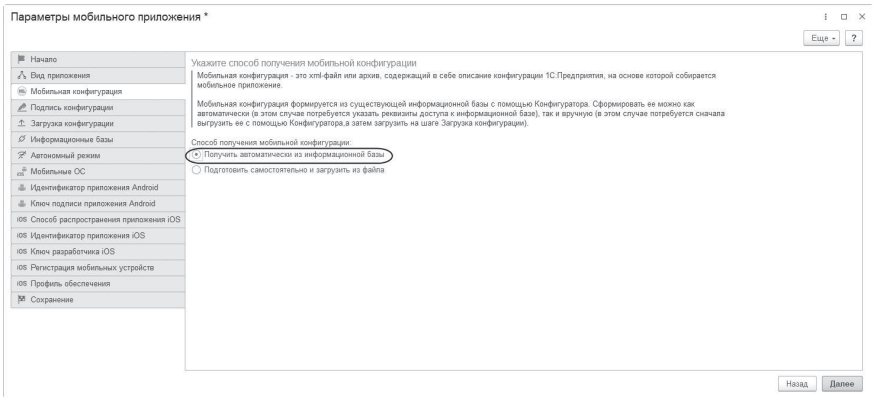


Рис. 5.5. Выбор способа получения мобильной конфигурации

На странице Подпись конфигурации нужно задать уникальный ключ подписи конфигурации, который будет помещен внутрь собранного мобильного приложения. Оставьте способ подписи конфигурации по умолчанию

в значении Автоматически при каждой сборке и нажмите кнопку Создать новый ключ. Заполните описание и пароль ключа и нажмите Создать ключ. Пароль запомните, так как им будет подписываться эта конфигурация (или ее новые версии). Созданный вами ключ отобразится в поле Ключ подписи мобильного клиента (рис. 5.6).

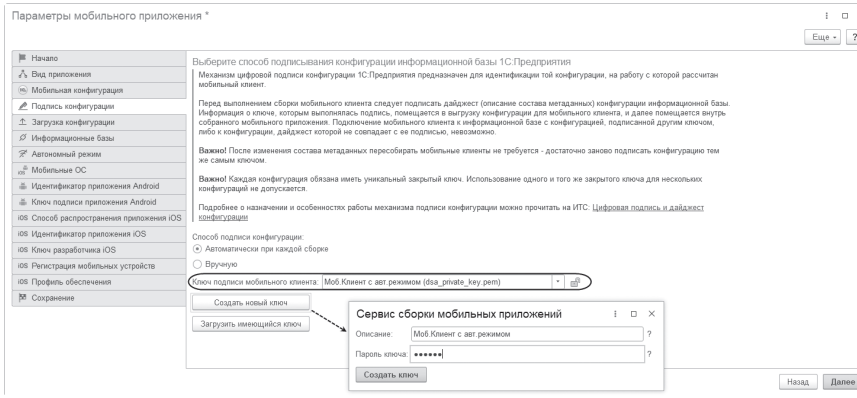


Рис. 5.6. Установка ключа подписи конфигурации

На странице Загрузка конфигурации загрузите мобильную конфигурацию, из которой будет собрано мобильное приложение, нажав кнопку Настроить подключение к информационной базе (рис. 5.7).

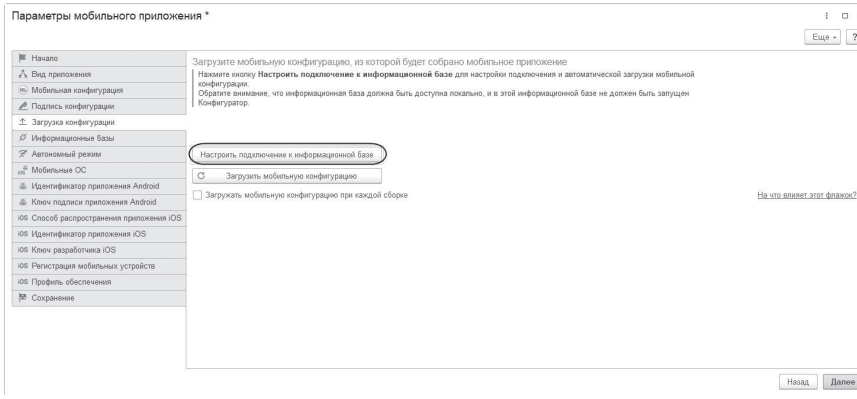


Рис. 5.7. Загрузка мобильной конфигурации

В открывшемся диалоге на закладке Информационная база укажите, где находится база с вашей конфигурацией, а также имя и пароль (в данном примере его нет) пользователя информационной базы – Администратор. На закладке Платформа для выгрузки каталог платформы оставьте без изменений, если вы его не меняли при установке (рис. 5.8).

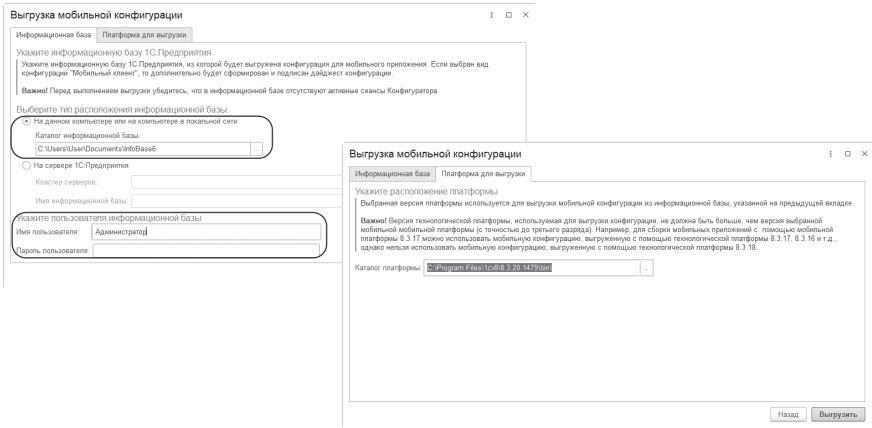


Рис. 5.8. Загрузка мобильной конфигурации

После небольшого ожидания на странице Загрузка конфигурации появится имя загруженной мобильной конфигурации, которое вы можете изменить, нажав на значок карандаша (рис. 5.9).

Рис. 5.9. Загрузка мобильной конфигурации

На странице Информационные базы укажите адрес, по которому вы публиковали информационную базу на веб-сервере (например, http://192.168.0.105/MCA). В данном случае адрес публикации указан в формате: http://<IP-адрес компьютера в беспроводной сети>/<Каталог веб-сервера, на котором опубликована информационная база> (рис. 5.10).

Рис. 5.10. Адрес публикации информационной базы

На странице Автономный режим нужно установить флажок Использовать автономный режим, так как вы собираете приложение для мобильного клиента с автономным режимом (рис. 5.11).

Рис. 5.11. Использование автономного режима

На странице Мобильные ОС укажите, что вы хотите собрать мобильное приложение для платформы Android (рис. 5.12).

Рис. 5.12. Выбор платформы для работы приложения

На странице Идентификатор приложения Android задайте строковый идентификатор собираемого приложения в формате <com>.<e1c>.<имя приложения на латинице> (рис. 5.13).

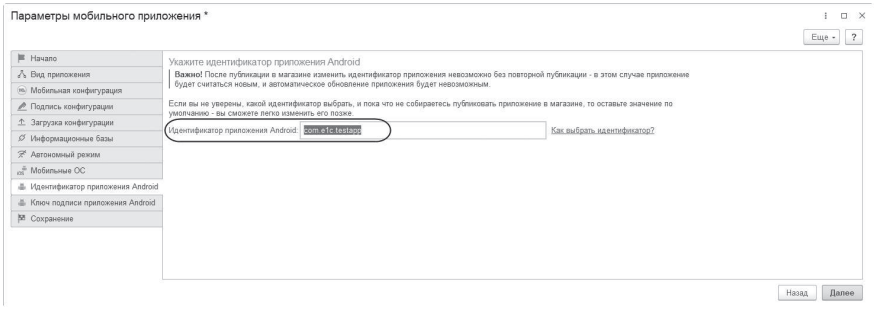


Рис. 5.13. Установка идентификатора собираемого приложения

На странице Ключ подписи приложения Android нужно задать ключ подписи, которым будут в дальнейшем подписываться все версии этого приложения. Поскольку вы собираете приложение в первый раз, создайте новый ключ (запомните его пароль), и при установке новых версий приложения оно будет обновляться. Нажмите кнопку Создать новый ключ. Заполните все необходимые поля и нажмите Сформировать ключ. Созданный вами ключ отобразится в поле Ключ подписи (рис. 5.14).

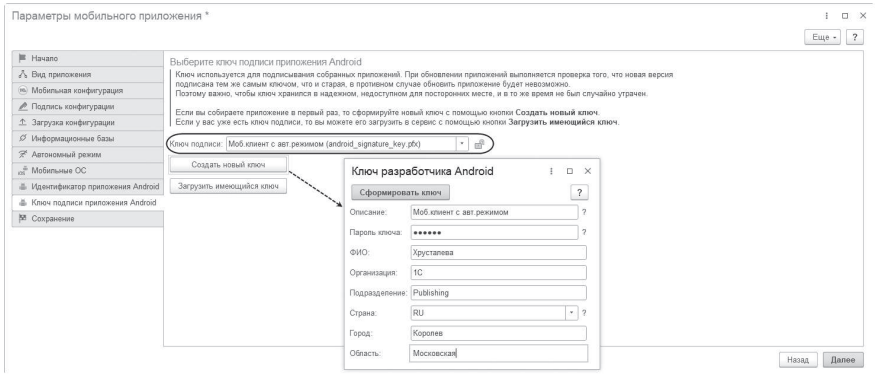


Рис. 5.14. Установка ключа подписи приложения

И в завершение на последней странице Сохранить нажмите Сохранить настройки.

Затем нажмите кнопку Выполнить сборку над верхней таблицей Приложения.

В результате через некоторое время в нижней таблице Сборки появится строка со сборкой вашего приложения. Чтобы установить приложение на мобильное устройство, выделите эту строку и нажмите кнопку Установить на устройство. В появившемся окне отсканируйте QR-код для приложения Android или просто скопируйте ссылку, нажав Скопировать ссылку, и затем передайте ее на мобильное устройство (рис. 5.15).

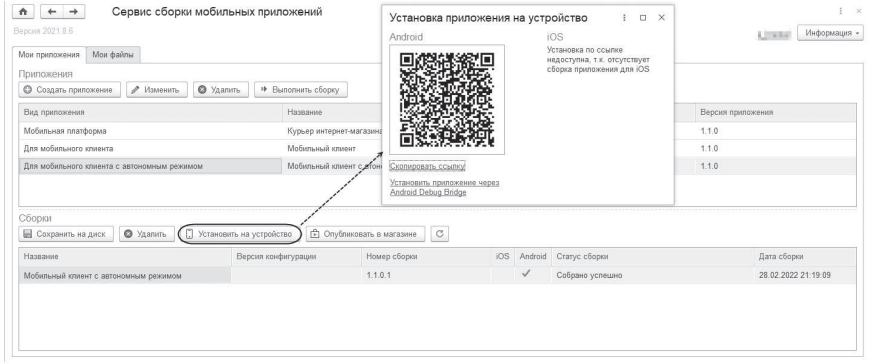


Рис. 5.15. Установка приложения на мобильное устройство

Затем откройте в браузере на мобильном устройстве полученную ссылку и нажмите Установить/Обновить (рис. 5.16).

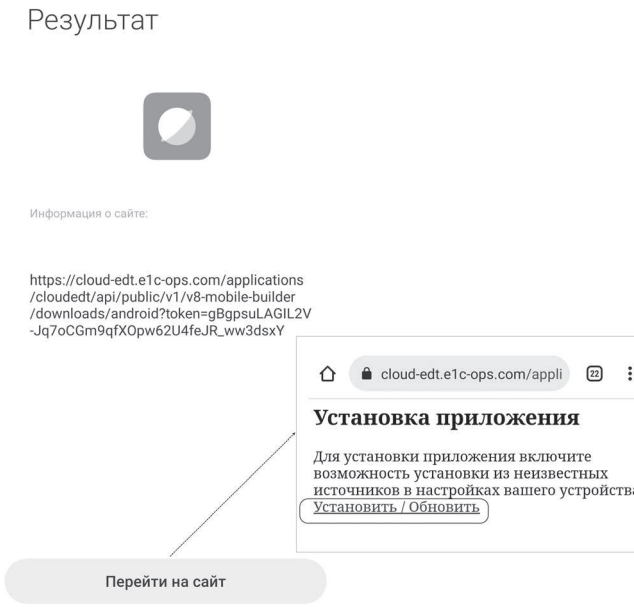


Рис. 5.16. Установка приложения на мобильное устройство

ВНИМАНИЕ

Для установки приложения на вашем устройстве должно быть включено разрешение на установку приложений из внешнего источника – интернет-браузера.

Чтобы сохранить установочные файлы на компьютер, с которого выполнялась сборка, нажмите кнопку Сохранить на диск, выберите архитектуру процессора мобильного устройства (ARM-64, ARM и т. п.) и нажмите Получить приложение (рис. 5.17).

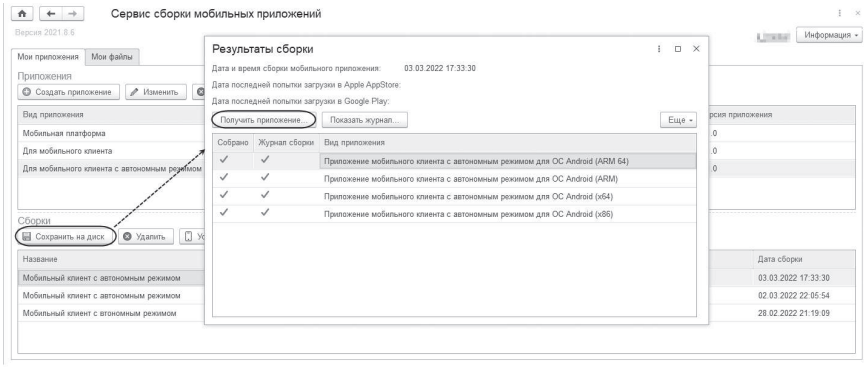


Рис. 5.17. Сохранение установочных файлов на компьютер

Итак, вы познакомились с работой сборщика мобильных приложений и быстро и легко собрали приложение одного из видов (мобильный клиент с автономным режимом), разработанных в книге.

Если вы собираетесь опубликовать собранное мобильное приложение в магазине приложений, то для его сборки нужно использовать полный режим мастера. Выбрать этот режим можно на первом шаге. Когда все шаги мастера будут пройдены, нажмите Опубликовать в магазине (см. рис. 5.3).

При желании выполнить более тонкую настройку вашего приложения выделите его в верхней таблице Приложения, нажмите Изменить и выберите на первом шаге Полный режим настроек мастера. Затем перейдите на еще не заполненные вами закладки, внимательно прочитайте доступную и понятную справочную информацию, выполните нужные вам настройки и сохраните их. Затем соберите новую версию приложения, нажав Выполнить сборку.

Фирма «1С»
123056, Москва, а/я 64, Селезневская ул., 21.
Тел.: (495) 737-92-57, факс: (495) 681-44-07.
1c@1c.ru, http://www.1c.ru/

Издательство ООО «1С-Паблишинг»
127434, Москва, Дмитровское ш., д. 9.
Тел.: (495) 681-02-21, факс: (495) 681-44-07.
publishing@1c.ru, http://books.1c.ru

Об опечатках просьба сообщать по адресу publishing@1c.ru.