О РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ ОПТИМИЗАЦИИ РАБОТЫ ЛОГИСТИЧЕСКИХ КОМПАНИЙ НА БАЗЕ МОБИЛЬНОЙ ПЛАТФОРМЫ IOS

Ткачев И. Е., Рудикова Л. В.
Кафедра современных технологий программирования, Гродненский государственный университет имени Янки Купалы
Гродно, Республика Беларусь
E-mail: tkachev_ie_15@mf.grsu.by, rudikowa@gmail.com

Излагаются общие подходы к реализации мобильного приложения на платформе iOS, которое предназначено для оптимизации рабочих процессов в логистических компаниях и упрощения коммуникации между сотрудниками. Приводится основная функциональность программного обеспечения, связанного с разработкой указанного мобиль-ного приложения.

Введение

В настоящее время сфера высоко интегрированных приложений для портативных компьютеров, смартфонов и сотовых телефонов развивается достаточно интенсивно. Однако, актуальными являются разработки для поддержки задач в специализированных предметных областях. Так, например, логистические компании, занимающиеся доставкой грузов наземным путем. В настоящее время спрос на логистические услуги постоянно возрастает. Это вызвано ростом экономики и усилением конкуренции среди производителей. Проблема конкурентоспособности решается путем совершенствования процедур складирования, транспортировки, распределения товаров между потребителями. Перед логистами встает задача регулировки и контроля доставки каждого груза, ведь, все поставки должны быть выполнены в срок и без непредвиденных ситуаций. Кроме того, должна поддерживаться постоянная связь с водителями, которых в компании может насчитываться более сотни. Все эти факты делают невозможным нормальное функционирование, централизованный контроль и развитие компании.

I. Этапы разработки приложения

Работу над реализацией приложения «Delievery Assistant» можно разбить на следующие этапы: проектирование веб-сервиса для хранения и обновления информации, связанной с доставкой, а также обеспечивающего связь между пользователями; проектирование базы данных; получение модели функций разрабатываемого приложения; разбиение приложения на модули в соответствии с выполняемыми функциями, проектирование интерфейса; реализация базы данных и логики взаимодействия с ней; реализация интерфейсов представлений; создание классов, используемых для привязки полученных данных к представлениям; реализация связи базы данных с интерфейсом. При первом запуске пользователю необходимо создать аккаунт, используя свою электроную почту, и провести авторизацию. Это главный аккаунт, который имеет возможность управлять системой и создавать подчиненные аккаунты соответствующего уровня. Далее приложение создает базу данных, а также необходимые таблицы, которые в будущем заполняются данными. В процессе работы приложения модифицируется содержимое базы данных, информация, полученная с веб-сервера, заносится в локальную базу данных. Это необходимо для доступа пользователя к собственным данным при отсутствии соединения с сетью. Следует отметить, что при добавлении данных пользователем в базу при отсутствии доступа к сети, в дальнейшем при наличии доступа в сеть, произойдет синхронизация данных с веб-сервером.

II. Общие подходы к архитектуре реализации приложения

Рассмотрим основные компоненты архитектуры мобильного приложения для оптимизации ра-боты логистических компаний. Приложение «Delievery Assistant» состоит из следующих частей: базы данных, в которой хранятся данные; веб-сервера, сохраняющего пользовательские данные; веб-сервера, обеспечивающего связь между пользователями; пользовательского интерфейса, предоставляющего пользователю доступ к функциям приложения; сервиса, предоставляющего GPS данные; авторизации через электронную почту. Для того чтобы пользователь мог использовать функционал приложения, ему необходимо совершить авторизацию через свою электронную почту. Авторизация реализована через технологию OAUTH 2.0, которую поддерживает SPRING BOOT (на его основе построен веб-сервер), что подразумевает генерацию token-авторизации. Далее уведомление о входе будет выслано на почту пользовате-ля. Для начала работы приложение должно получить token пользователя, который хочет авторизироваться в приложении. В основе приложения лежит архитектурный принцип SOLID, поэтому приложение разбито на модули, каждый из которых выполняет свою функцию. При этом моду-

ли используют общие сервисы, которые внедряются в них извне, что позволяет ослабить зависимости между ними, и повысить возможность их дальнейшего переспользования. Работа приложения начинается с модуля Login, в ходе работы которого проверяется, был ли пользователь уже авторизирован или нет. Если пользователь уже авторизовался, то в работу включается модуль Main и задействует первую вкладку приложения, которая также является отдельным моду-лем и называется Status. Пользователь получает доступ к основному функционалу приложения. Однако, если при обращении к UserDefaults (используется в приложении для создания именованных ассоциативных массивов типа «ключ – значение», которые могут быть использованы различными компонентами приложения) отсутствует token, то пользователю предлагается произвести авторизацию. Интерфейс авторизации довольно прост и понятен. Login модуль обращается к LoginHelper (класс, который инкапсулирует логику авторизации пользователя). Если же у пользователя логин отсутствует, ему будет предложено завести новый аккаунт и в случае подтверждения будет вызван модуль CreateAccount. После ввода комбинации логина-пароля, на почту пользователю будет выслано сообщение с кодом подтверждения. Если же авторизация прошла успешно, то полученный token из CreateAccount модуля передается в Main. Однако, если при авторизации произошла ошибка (т.е. после запроса авторизовать пользователя, веб-сервер вернул ошибку), то выводится сообщение об ошибке. После того как был получен token из Login или CreateAccount модуля, Main модуль обращается к UserDefaults для записи полученого token в хранилище типа «ключ-значение». Это необходимо для того, чтобы пользователь каждый раз не вводил свою комбинацию логин/пароль. Далее Main вызывает модуль Status для предоставления основного функционала приложения пользователю, либо предлагает настроить аккаунт, если аккаунт был только что создан. Main модуль является контейнером для других модулей и отображает их в виде вкладок. Такая архитектура упрощает синхронизацию данных между модулями. Само представление данного модуля наследуется от класса UITabBarController. Он содержит следующие вкладкти: Status, Chat, Map, Settings. Каждая вкладка – это самостоятельный модуль, который выполняет строго свою функцию и построен на основе VIPER. После того как Status-модуль был вызван, он обращается к NetworkService (общий для всех мо-дулей сервис для работы с сетью) для загрузки данных. Далее NetworkService проверяет наличие Интернета и, если он отсутствует, то все данные читаются из локальной базы данные, а, если соединение присутствует, то NetworkService обращается к веб-серверу для получения данных. Status модуль отоб-ражает всю необходимую информацию о текущих рейсах. Chat модуль реализует чат с другими сотрудниками организации. Map отображает текущее расположение грузов (использует Google Maps API), Setting позволяет осуществлять настройки аккаунта. NetworkService обращается к веб-серверу, где в качестве идентификации пользователя используется User-id, сгенерированный на основе адреса электронной почты. Полученные данные хранятся в формате JSON, поэтому для удобной работы с ними используется стандартная библиотека которая структурирует данные для удобного пользования. После загрузки данных NetworkService обращается к DataBaseManager для того, чтобы записать полученные с веб-сервера данные в локальную базу данных. Это необходимо для использования приложения без соединения с Интернетом. DataBaseManager взаимодействует с базой данных SQLite используя встроенный инструмент CoreData, который упрощает работу с базой данных. После сохранения данных в базу данных, полученная информация передается непосредственно в модуль. Для загрузок картинок используется класс ImageLoader, инкапсулирующий логику загрузки и кэширования картинок. Модуль передает в ImageLoader адрес картинки в сети и UIView, в который необходимо загрузить файл картинки. ImageLoader использует NSCache для хранения картинок в оперативной памяти устройства. После того как был передан URL и UIImageView, загрузчик картинок, прежде всего, проверяет наличие файла картинки в кэше: если картинка отсутствует, то загрузчик, используя HttpClient, загружает ее. После загрузки картинка помещается в кэш. Для того, чтобы картинка не загружалась дважды и при следующем обращении к картинке, она уже не будет загружаться из сети, а будет извлекаться из кэша, что обеспечивает экономию трафика. После загрузки картинка присваивается указному UIImageView.

III. Заключение

Таким образом, разработанное приложение предназначено для использования в логистических компаниях, которые заинтересованы в оптимизации их внутренней работы и контроля рабочих процессов. Приложение «Delievery Assistant» предоставляет возможность создания и настройки списка текущих доставок, отслеживания доставляемых грузов и общения между сотрудниками компании, вести централизованный учет расходов. В дальнейшем приложение может быть адаптировано и использоваться для связи грузоотправителя и грузоперевозчика, заключения между ними контракта, ведение учета на складах.