А.А. Козловский, А.А. Ступников

Тюменский государственный университет, г. Тюмень
УДК 004.9

РАЗРАБОТКА СРЕДСТВ КОНФИГУРИРОВАНИЯ И СБОРКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Аннотация. В статье описываются основные инструменты и подходы при разработке системы управления контентом и динамически конфигурируемого мобильного приложения под ОС Android. Приводится пример разработанной системы.

Ключевые слова. Динамически конфигурируемое мобильное приложение, система управления контентом, сборка мобильного приложения под ОС Android, веб-разработка, мобильная разработка.

Введение

В условиях развития мобильных технологий, растет заинтересованность предпринимателей в расширении своего бизнеса с помощью рынка мобильных приложений [1]. Как правило, мобильное приложение выступает в качестве инструмента для взаимодействия компании со своими клиентами. Для небольших компаний основной проблемой является необходимость разрабатывать мобильное приложение на заказ или самостоятельно в течении определенного времени, имеющее во многом схожий функционал. Идеей решения может выступить разработка готового решения в виде динамически настраиваемого мобильного приложения. Для функционирования сложных и динамичных мобильных приложений, непосредственно работающих с обновляемыми данными, необходимо взаимодействие с сервером. Таким образом формируется клиент-серверная архитектура, в которой в качестве клиента выступает мобильное приложение. Следовательно, помимо разработки

мобильного приложения необходимо также разрабатывать и Web API. Также одной из проблем, с которой сталкиваются разработчики мобильных приложений, является необходимость в обновлении уже выпущенного в онлайн приложения при изменении визуальных элементов или каких-либо ресурсов, при этом используя только возможности SDK. В данном случае можно использовать кроссплатформенную разработку, при этом лишившись таких достоинств нативной разработки как высокая производительность и инструментарий для работы с ОС [2]. Одним из подходов к решению данной проблемы является концепция Server-Driven UI, позволяющая сделать экраны нативного приложения динамически изменяемыми с помощью запросов на сервер.

Требования к системе и функциональные возможности

Система разрабатывается применительно к наиболее распространенным областям ведения бизнеса, что формирует следующие требования к функционалу: возможность показывать новости и акции клиентам, настройка продуктов и услуг компании, реализация обратной связи с клиентами в виде сообщений, уведомлений, отзывов и оценок, отслеживание поступающих заказов, записей и заявок.

Основные технологии и решения

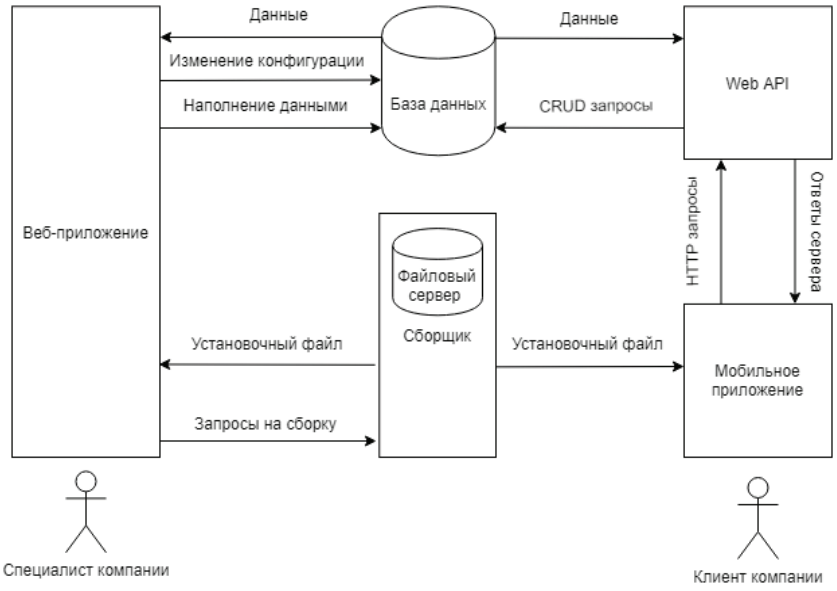

Рис. 1. Общая схема работы системы

Разрабатываемая система в общем виде (рис. 1) состоит из нескольких модулей:

- База данных компаний. В качестве СУБД используется кроссплатформенная и свободно распространяемая PostgreSQL.

- Веб-приложение для управления контентом и настройки конфигурации мобильного приложения. Для разработки используется ASP.NET Core MVC, язык программирования – C#. Для верстки используются CSS/HTML, фреймворк Bootstrap. Для работы с таблицами данных, постраничной навигацией и формами используется jQuery с использованием технологии AJAX и встроенные Tag Helper. Разделы «Администрирование», «Конфигурация» и «Сборщик» определены в соответствующие области (Area).

- Web API. Для разработки используется ASP.NET Core MVC, язык программирования – C#. Для достижения высокой производительности веб части системы и ускорения разработки было принято решение включить

API в проект вместе с веб-приложением, тем самым добившись возможности для веб-приложения получать данные напрямую обращаясь к БД, а не отправлять отдельный HTTP запрос на сервер. Соответствующие контроллеры и модели были определены в отдельную область (Area).

- Мобильное приложение для взаимодействия клиентов с компанией.

Для разработки мобильного приложения под ОС Android используются Android SDK, язык программирования Kotlin как приоритетный язык [3], среда разработки Android Studio. Общая архитектура приложения – Single Activity [4], упрощающая работу с жизненным циклом Activity, используя его как жизненный цикл всего приложения. Тем самым, все необходимое для бизнес-логики легко привязать к состоянию приложения, полностью контролируя запуски экранов с помощью транзакций Fragment'ов. Основной паттерн – MVP (Model-View-Presenter), используемый для разделения ответственности в коде приложения: Model предоставляет данные для Presenter. View реагирует на команды UI элементов, передавая эти события в Presenter, и изменяет свое содержимое по требованию Presenter. Применительно к Android приложениям паттерн MVP снимает часть ответственность с Activity или Fragment – работа с асинхронными задачами отводится Presenter'у. Вся бизнес-логика располагается в Presenter и Model. Такими образом, Single Activity в связке с MVP, позволяет полностью отделить работу с UI и бизнес-логику и реализовать навигацию с помощью Fragment транзакций. При запуске мобильное приложение отправляет запрос с названием своего пакета на сервер и в ответе получает ссылку для работы с API и основные параметры конфигурации приложения. Далее для получения данных на определенных экранах, приложение использует полученную ссылку для отправки соответствующих запросов. На рис. 2 представлена общая схема работы мобильного приложения с паттерном MVP и архитектурой Single Activity.

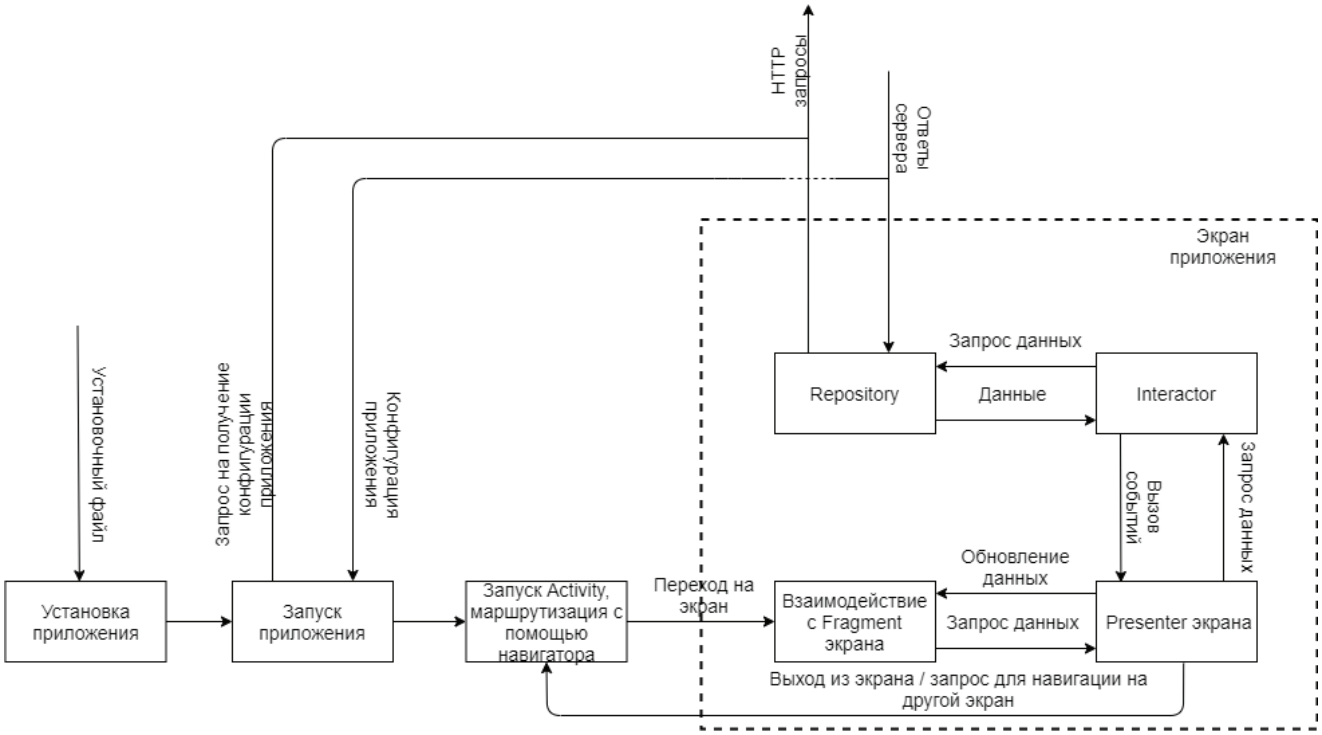

Рис. 2. Общая схема работы мобильного приложения с паттерном MVP и архитектурой Single Activity.

Для создания динамически настраиваемых экранов используется RecyclerView вместе с паттерном AdapterDelegate: для каждого отдельного компонента создается свой AdapterDelegate, ответственный за логику отображения на экране. Затем все эти компоненты помещаются в список элементов основного Adapter’a, управляющего созданием AdapterDelegate. Таким образом, во время получения данных, основной адаптер выбирает какому делегату предоставить отображение в зависимости от типа данных или других условий (например, параметров конфигурации). При этом на одном экране можно иметь несколько RecyclerView, каждый из которых имеет Adapter с некоторыми AdapterDelegate. Для параметров, влияющих на отображение всего экрана реализуются отдельный Fragment. На рис. 3 представлен пример экрана новостей NewsFragment с двумя способами отображения, реализованный с помощью паттерна AdapterDelegate.

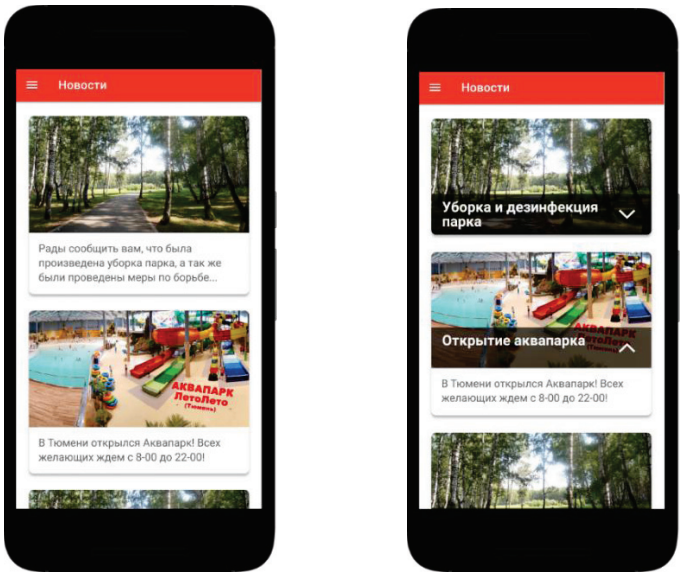

Рис. 3. Экран новостей с двумя способами отображения: а) карточками б) расширяющимися карточками

Во фрагменте NewsFragment к основному RecyclerView прикрепляется Adapter, который инициализирует CardsAdapterDelegate и ExpandableCardsAdapterDelegate, каждый из которых отображает компонент при условии, что данные для отображения являются объектом News и выбран соответствующий параметр конфигурации. Например, для CardsAdapterDelegate условие примет вид: `items[position] is News && newsViewModeParam == “Cards”,` где items[position] – элемент на текущей позиции в списке новостей.

- Сборщик. Для сборки мобильного приложения под ОС Android необходим ряд зависимостей и средств. Также в рамках системы нужно поддерживать непрерывную работу сборщика, не переустанавливая каждый раз при запуске компоненты для сборки. Подходящим инструментом является Docker, позволяющий в автоматическом режиме управлять контролируемой средой без повторной комплектации необходимых ресурсов и зависимостей. Основные компоненты для сборки мобильного приложения под ОС Android: Gradle, JDK, Kotlin, Android SDK. Для работы скриптов по замене информации о сборке в файлах

проектах требуется Python, для обновления проекта с мобильным приложением необходим Git. Образ контейнера создается на основе ОС Ubuntu. Алгоритм работы сборки представлен на рис. 4.

Рис. 4. Алгоритм работы сборки мобильного приложения.

В основной директории образа расположен проект мобильного приложения. Для замены названия, описания, пакета и версии приложения задействуются скрипты по изменению соответствующих файлов проекта. С помощью библиотек, работающих с графикой, корректируются под необходимые разрешения логотип и иконка приложения, и так же скриптами заменяются необходимые изображения в проекте. С помощью Git команд возможна поддержка актуальных версий проекта. После того, как была заменена информация в файлах проекта, запускается Gradle скрипт для сборки приложения.

Сценарий работы пользователя

При входе в систему пользователю будет предложено авторизоваться. Если пользователя нет в базе данных, то ему необходимо зарегистрироваться. При успешной регистрации в базе данных создаются параметры приложения со значениями по умолчанию, и, затем пользователю станут доступны три раздела системы: «Администрирование», «Конфигурация», и «Сборщик». В разделе «Сборщик» пользователь может заполнить необходимую для сборки мобильного приложения информацию. Далее в подразделе «Архив» пользователь может отправить запрос на сборку мобильного приложения.

Также в этом подразделе отображается история всех запросов на сборку и ссылки для скачивания установочных файлов (рис. 5).

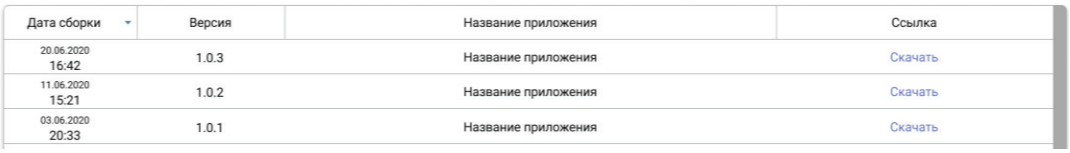

Рис. 5. Окно с информацией о сборках мобильного приложения

В разделе «Конфигурация» в подразделе «Функционал» пользователь может определить функционал приложения (рис. 6).

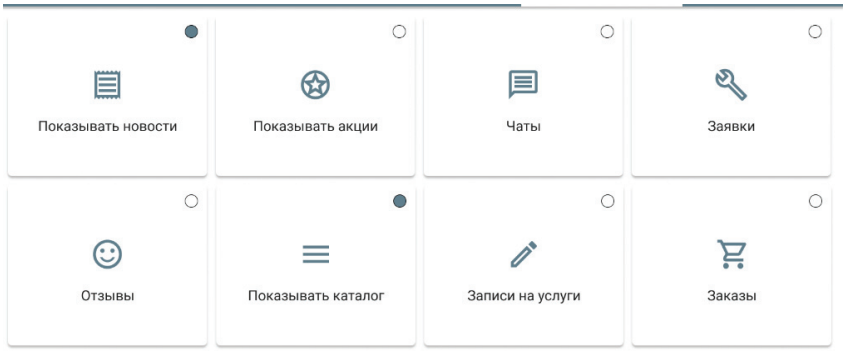

Рис. 6. Окно с выбором функционала приложения

В других подразделах представлен выбор параметров отображения некоторых элементов приложения. Например, в подразделе «Меню» (рис. 7) можно выбрать способ отображения главного меню мобильного приложения.

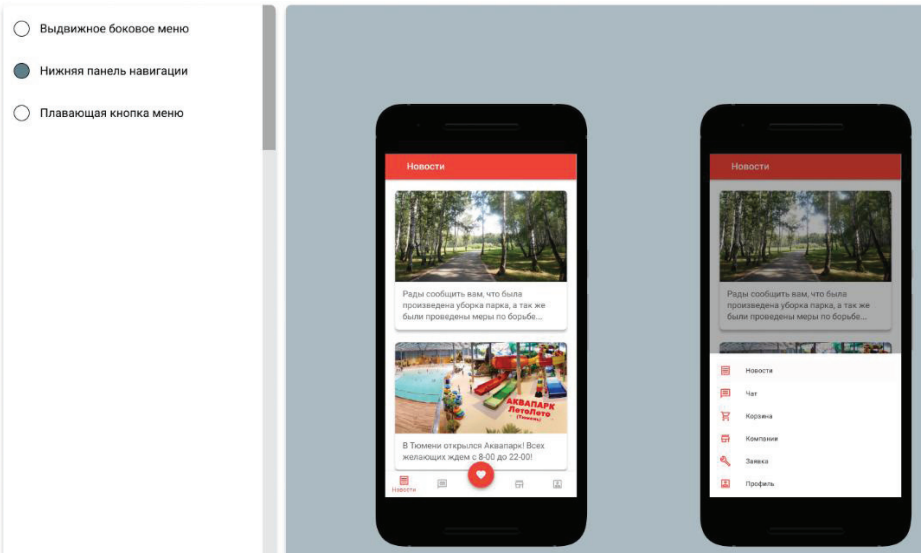

Рис. 7. Выбор способа отображения главного меню

В разделе «Администрирование» представлены подразделы с информацией о контенте приложения и инструменты для добавления в базу данных новых объектов, редактирования и удаления уже существующих. Например, в подразделе «Каталог» представлена информация о всех продуктах компании в виде таблицы (рис. 8).

Номер	Каталог	Наименование	Описание	Цена	Тип
1	Овощи	Товар #1	Описание товара	1000 руб.	Продукт
2	Услуги	Услуга #1	Описание товара	1000 руб.	Услуга
3	Фрукты	Товар #2	Описание товара	1000 руб.	Продукт

Рис. 8. Раздел «Администрирование». Раздел "Каталог"

Также в этом же подразделе можно добавить продукт с помощью окна добавления (рис. 9).

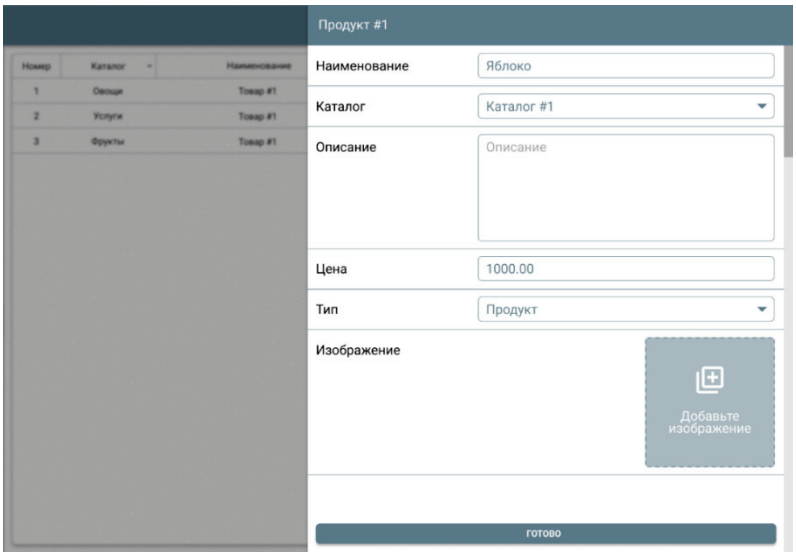

Рис. 9. Раздел "Администрирование". Окно добавления продукта.

Выводы

Предложенная автоматизированная система может дать возможность компаниям в кратчайшие сроки расширить свой бизнес через рынок мобильного приложений. С ее помощью можно управлять контентом, выбирать необходимый функционал для мобильного приложения, динамически настраивать в нем отображение элементов, а также получать сборку, готовую к выпуску в магазин приложений.

СПИСОК ЛИТЕРАТУРЫ

1. Report. The State of Mobile 2020 [Электронный ресурс] - URL: https://www.appannie.com/en/go/state-of-mobile-2020/ (Дата обращения: 30.05.2020)

2. PERFORMANCE ANALYSIS OF NATIVE AND CROSS-PLATFORM MOBILE APPLICATIONS [Электронный ресурс] - URL: https://www.researchgate.net/publication/320039474_PERFORMANCE_ANALYSIS_OF_NATIVE_AND_CROSS-PLATFORM_MOBILE_APPLICATIONS (Дата обращения: 30.05.2020)

3. Google I/O 2019: Empowering developers to build the best experiences on Android + Play [Электронный ресурс] - URL https://android-

developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html (Дата обращения: 30.05.2020)

4. Use Android Jetpack to Accelerate Your App Development [Электронный ресурс] - URL: https://android-developers.googleblog.com/2018/05/use-android-jetpack-to-accelerate-your.html?m=1 (Дата обращения: 30.05.2020)

5. Benchmark. How fast is Toothpick? [Электронный ресурс] - URL: https://github.com/stephanenicolas/toothpick/wiki/Benchmark (Дата обращения: 30.05.2020)