МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Директор ООО «Наполеон Айти»

_____ П.С. Подкорытов

“___”_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

“___”_____2019 г.

РАЗРАБОТКА IOS-КОНСТРУКТОРА ДЛЯ СОЗДАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ РОЗНИЧНЫХ СЕТЕЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.04.02.2019.308-150.ВКР

Научный руководитель,
доцент кафедры СП, к.п.н.
_____ О.Н. Иванова

Автор работы,
студент группы КЭ-220
_____ А.Г. Михайлов

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____2019 г.

Челябинск–2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ

Зав. кафедрой СП

_____ Л.Б. Соколинский

09.02.2019

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистра
студенту группы КЭ-220 Михайлову Артему Геннадьевичу,
02.04.02 «Фундаментальная информатика и информационные технологии»
(магистерская программа «Технологии разработки высоконагруженных систем»)

1. **Тема работы** (утверждена приказом ректора от 25.04.2019 № 899)

Разработка iOS-конструктора для создания мобильных приложений для розничных сетей

2. **Срок сдачи студентом законченной работы:** 05.06.2019.

3. **Исходные данные к работе:**

1. Марк Д., Наттинг Дж. iOS 6 SDK Разработка приложений для iPhone, iPad и iPod touch. – США: APress, 2013. – 672 p.

2. Арлоу Дж., Нейштадт А. UML 2 и унифицированный процесс. – М.: Символ-Плюс, 2007. – 624 p.

3. Hegarty P. Standford University Lecture – Developing iOS7 App. [Электронный ресурс] URL: https://www.youtube.com/watch?v=ZqKbN_C4Yvg&list=PLnOdYr35FyvhDUAIW17vo7nGfHJAyikUp (дата обращения: 11.03.2019).

4. **Перечень подлежащих разработке вопросов**

4.1. Произвести постановку задачи по созданию iOS-конструктора.

4.2. Произвести сравнительный обзор существующих аналогов для разрабатываемого iOS-конструктора.

4.3. Изучить современные платформы и средства разработки для операционной системы iOS.

4.4. Определить требования и спроектировать iOS-конструктор.

4.5. Реализовать и протестировать iOS-конструктор.

5. Дата выдачи задания: 08.02.2019.

Научный руководитель

Доцент кафедры СП, к.п.н. О.Н. Иванова

Задание принял к исполнению А.Г. Михайлов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Индустрия мобильных приложений занимает громадную часть электронного бизнеса в современных компьютерных сетях. В последние годы наблюдается существенный рост рынка мобильных приложений во всех его областях. Так, в 2012 году рынок мобильных приложений оценивался в 53 миллиарда долларов, а в 2016 году он составил уже около 100 миллиардов долларов [20], а в 2018 только за один год пользователями был потрачен 101 миллиард долларов в магазине приложений [8], что является на 75% больше, чем такой же показатель в 2016 году. Поэтому, имеются перспективы создания новых мобильных приложений, т.к. доля пользователей именно такими устройствами постоянно растет, а также мобильные технологии постоянно развиваются и совершенствуются.

Частичный или полный перенос сферы продаж в интернет-пространство является логичным и целесообразным для любой компании, т.к. это позволяет сэкономить издержки на открытие новых магазинов, повышает узнаваемость бренда и расширяет географию продаж. Но разработка брендированных мобильных приложений зачастую может оказаться слишком дорогой, особенно для малого и среднего бизнеса.

В связи с этим было принято решение создать конструктор мобильных приложений, который позволял бы выпустить готовое мобильное приложение быстро и дешево.

Цель и задачи

Целью работы является разработка iOS-конструктора для создания мобильных приложений для розничных сетей. Для достижения цели должны быть решены следующие задачи:

1) произвести постановку задачи по созданию iOS-конструктора;
2) произвести обзор существующих аналогов для разрабатываемого iOS-конструктора;

3) изучить современные платформы и средства разработки для операционной системы iOS;

4) определить требования и спроектировать iOS-конструктор;

5) реализовать iOS-конструктор;

6) протестировать iOS-конструктор.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, библиографического списка и одного приложения. Объем работы составляет 43 страницы, объем библиографии – 21 источник. Объем приложения – 3 страницы.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Постановка задачи

Разрабатываемый продукт должен представлять собой iOS-конструктор для создания мобильных приложений для розничных сетей магазинов. Он будет предоставлять возможность быстрого создания мобильных приложений с базовым функционалом, а также возможность легкого и быстрого внедрения дополнительного функционала для конкретного приложения.

iOS-конструктор должен включать в себя возможность автоматической сборки мобильного приложения на основе настроек, введенных для конкретного магазина с помощью внешнего интерфейса, и последующей загрузки собранного мобильного приложения в магазин приложений App Store [1].

В базовый функционал мобильного приложения, созданного на основе конструктора могут быть включены следующие модули: каталог продукции, торговые точки, дисконтная (либо бонусная) карта, обратная связь, бронирование и оплата заказа, акции розничной сети, о магазине (либо бренде).

В модуле каталога будет доступен общий каталог продукции, список категорий и подкатегорий (при наличии), список товаров категории, возможность фильтрации и интеллектуального поиска товаров, а также подробная информация о товаре.

В модуле торговых точек должна быть доступна карта торговых точек с возможностью применения различных фильтров и добавление торговых точек в избранное. Также в платформе должна быть реализована возможность построения маршрута до торговой точки в различных приложениях-навигаторах: карты Apple, Я.Карты и др.

В модуле обратной связи должны быть реализованы различные средства связи с представителями розничной сети: социальные сети, электронная почта, звонок на указанный номер и др.

В модуле дисконтной карты должна быть реализована возможность привязки карты пользователя к устройству для дальнейшего ее использования при покупке, как альтернатива пластиковой карты. Также, пользователю должен иметь возможность просматривать данные карты, например, количество бонусов или текущий размер скидки.

В модуле «О магазине (бренде)» должна располагаться общая информация о бренде либо торговой точке.

В модуле акций платформа должна отображать список акций торговой сети в виде баннеров с возможностью просмотра дополнительной информации об акции.

В административной панели должна быть возможность конфигурировать следующие части мобильного приложения: основной цвет приложения, количество и порядок функциональных модулей мобильного приложения, названия некоторых визуальных элементов (заголовков, кнопок и т. д.), а также некоторые элементы, относящиеся к бренду, выпускаемому с помощью конструктора (иконка и название мобильного приложения, ссылки на программу лояльности и политику конфиденциальности и т. д.).

В административной панели должна быть возможность запустить сборку мобильного приложения.

1.2. Обзор аналогов

В настоящий момент рынок платформ для создания мобильных приложений представлен следующим спектром платформ.

«Эквид» – это платформа для создания интернет-магазина, простой и быстрый способ начать продавать онлайн.

Достоинства:

- помимо мобильного приложения есть возможность запустить сайт интернет-магазина;

- наличие бесплатного плана, что позволяет попробовать возможности платформы;

- к мобильному приложению также предлагается административная панель для управления данными приложения;

- поддержка на русском языке.

Недостатки:

- недружественный дизайн;

- в представленных на сайте платформы примерах нет ни одного приложения;

- мобильное приложение можно запустить только при подписке на самый дорогой тарифный план;

- функционал приложения сильно ограничен.

«GoodBarber» – это платформа для создания нативных приложений для публикации контента и взаимодействия с пользователями.

Достоинства:

- наличие различных тарифных планов;

- наличие тридцатидневного бесплатного периода, что позволяет попробовать возможности платформы;

- плохие отзывы в магазинах приложений.

Недостатки:

- недружественный дизайн приложений;

- отсутствие русскоязычной поддержки;

- выпущенные приложения работают медленно и нестабильно;

- функционал приложений ограничен.

«Mobium» – это платформа для создания нативных мобильных приложений для магазинов.

Достоинства:

- много примеров приложений, запущенных на этой платформе;

- наличие бесплатной демоверсии;

- информативный сайт.

Недостатки:

- недружественный дизайн приложений;

- дизайн приложений не адаптирован под различные разрешения экранов;

- плохие оценки в магазинах приложений;

- выпущенные приложения работают медленно и нестабильно;

- функционал приложений ограничен.

«ImShop» – это платформа для создания мобильных приложений для магазинов.

Достоинства:

- наличие бесплатной демоверсии;

- информативный сайт.

Недостатки:

- нет примеров выпущенных мобильных приложений;

- функционал приложений ограничен;

- очень дорогие тарифные планы.

Сведем некоторые параметры рассмотренных платформ для создания приложений в таблицу 1.

Табл. 1. Обзор аналогов платформ для создания приложений

№ п/п	Критерий	**Эквид**	**GoodBarber**	**Mobium**	**ImShop**
1.	Функциональность приложений	2/5	4/5	3/5	3/5
2.	Удобство пользования	–	3/5	2/5	–
3.	Гибкость тарифных планов	2/5	3/5	3/5	2/5
4.	Доступность тарифных планов	5/5	4/5	3/5	1/5
5.	Оценки приложений*	–	4/5	2/5	–

* – по информации на 15.02.2019 из [1]

Из таблицы видно, что рассмотренные платформы по многим параметрам не достигают даже среднего уровня. В связи с этим было принято решение создать платформу с тем дизайном, который будет наиболее удобен и привычен для пользователя операционной системы iOS, а также обладающее широким функционалом и гибким тарифным планом, чтобы клиент мог выбрать только тот функционал, который ему нужен и при этом не переплачивать за ненужные функции.

1.3. Средства разработки

Компания Apple предоставляет собственную платформу для разработки мобильных приложений для операционной системы iOS – Xcode. Данная платформа включает в себя все необходимое для разработки приложений, а именно инструменты для написания и рефакторинга исходного кода, программирования интерфейсов, отладки и тестирования приложений [11]. Кроме платформы Xcode, существуют также платформы для кроссплатформенной разработки мобильных приложений, например, Xamarin [10]. Но кроссплатформенные платформы не предоставляют той полноты инструментов, которые есть в Xcode, а именно: сложность написания автоматизированных тестов (чтобы реализовать тесты, необходимо создавать отдельное приложение, которое будет запускаться и тестировать необходимые участки нашего приложения), нестабильность симуляторов устройств (частые аварийные завершения работы, некорректность поведения в некоторых случаях). Нестабильность была проверена на собственном опыте при разработке проекта по предмету «Основы технологии программирования .NET», аварийный завершения работы происходили не реже чем каждый час работы симулятора. Кроме того, в таких платформах затрудняется верстка приложения и реализация его бизнес-логики, из-за возможности кроссплатформенной разработки, что повышает общую нестабильность мобильного приложения. Так, например, добавляя какой-либо элемент, мы не можем

точно сказать, как он будет выглядеть на разных устройствах и разных системах, так как его настройка является общей для всех платформ, а тонкости реализации верстки для разных платформ различны.

Помимо недостатков, описанных выше, стоит учитывать своевременность обновлений приложения. Когда выходит новая версия операционной системы, необходимо поддерживать ее новые возможности для поддержания спроса на приложение. Xcode обновляется одновременно с обновлением операционной системы, что позволяет сразу начать использовать новые функции и возможности в разрабатываемых приложениях. В то время как при разработке на платформе Xamarin нет возможности оперативно выпустить обновление для приложения, так как необходимо ждать, пока в платформе появятся соответствующие инструменты [21]. Именно поэтому для разработки продукта была выбрана платформа Xcode.

Для разработки мобильных приложений под семейство iOS существуют 2 языка программирования – Swift и Objective C. Для реализации продукта был выбран язык Swift, так как этот язык создавался компанией Apple прежде всего для разработки приложений на iOS. Он является более легким для чтения и более устойчивым к ошибкам программиста, нежели его предшественник – Objective C.

Objective C же, являясь более старым языком, содержит конструкции, которые не всегда можно легко прочитать, что затрудняет его поддержку и разработку нового функционала. Также, по сравнению с Objective C, Swift имеет намного больше конструкций, которые применяются при разработки мобильных приложений на iOS, что значительно уменьшает время разработки.

Язык Swift имеет ряд преимуществ:

1) язык является мультипарадигмальным, включая в себя объектно-ориентированную, протокол-ориентированную и функциональную парадигмы;

2) легкость и скорость разработки и сопровождения (встроенная в среду разработки документация языка с удобной навигацией и стабильно растущее сообщество программистов);

3) высокая читаемость и компактность кода;

4) развитая система контроля ошибок и рефакторинга кода.

Для создания приложения будет использоваться фреймворк Cocoa Touch [2], который является частью iOS SDK – комплекта средств для разработки приложений для системы iOS.

В состав фреймворка входят следующие ключевые библиотеки:

1) Foundation Kit – библиотека, содержащая основные классы (классы с префиксом NS);

2) UIKit – библиотека, которая содержит GUI-классы (специфичные для iOS);

3) CoreAnimation – библиотека, позволяющая создавать анимации для пользовательских интерфейсов;

4) CoreAudio – библиотека для воспроизведения, записи и обработки звука.

Также фреймворк Cocoa Touch включает в себя следующие технологии и механизмы: MapKit для работы с картами и навигацией, многозадачность, iAd для внедрения контекстной рекламы в мобильные приложения.

Для работы с данными внутри приложения будет использоваться фреймворк Realm [7]. Он скрывает реализацию взаимодействия с физическим хранилищем данных на устройстве, предоставляя лишь оболочку для работы мобильного приложения с хранимыми данными. Это позволяет легко добавлять новые сущности, атрибуты в существующие сущности, а также строить связи между ними.

Таким образом, нами была осуществлена постановка задачи, произведен анализ аналогичных программных продуктов, имеющихся на рынке, а также сделан выбор платформы и языка программирования для реализации мобильного приложения.

2. ПРОЕКТИРОВАНИЕ iOS-КОНСТРУКТОРА ДЛЯ СОЗДАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ РОЗНИЧНЫХ СЕТЕЙ

2.1. Архитектура iOS-конструктора

На рис. 1 представлена архитектура iOS-конструктора.

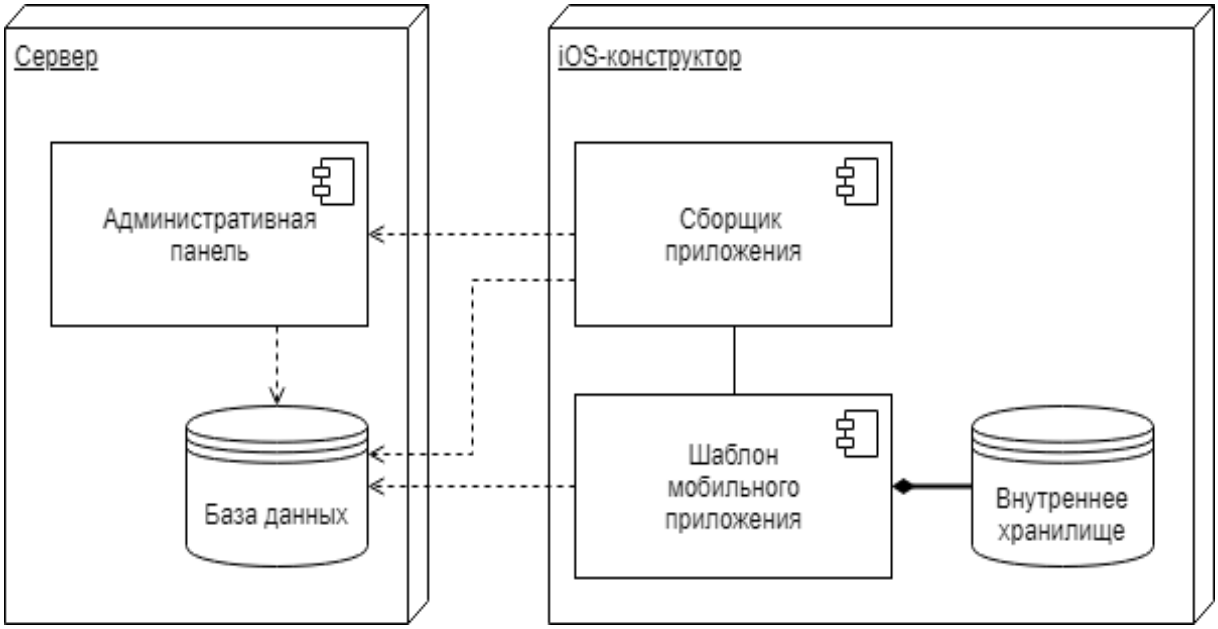

Рис. 1. Архитектура iOS-конструктора

iOS-конструктор состоит из двух основных компонентов – сборщик мобильного приложения и само мобильное приложение.

Менеджер заполняет настройки для клиентского приложения с использованием административной панели, после чего они записываются в базу данных.

Сборщик приложения использует административную панель, как внешний интерфейс, с помощью которого менеджер может запустить сборку мобильного приложения.

Сборщик приложения подставляет данные с настройками клиентского приложения, задает их в компоненте мобильное приложение.

После этого сборщик запускает процедуру сборки клиентского мобильного приложения и загрузки его в магазин приложений App Store.

2.2. Определение требований к проектируемому шаблону мобильного приложения

В ходе проектирования конструктора были определены функциональные возможности, которые могут быть предоставлены пользователю конечного мобильного приложения.

Приложение должно отображать пользователю каталог товаров розничной сети. Каждый товар должен содержать следующую информацию: название: краткое описание, мера измерения товара, изображения и другие данные, необходимые пользователю.

Приложение должно автоматически определять город по геопозиции пользователя в случае, если пользователь разрешил отслеживание своей геопозиции в приложении. Также пользователь должен иметь возможность посмотреть список доступных городов и изменить город, выбранный вручную или автоматически.

Мобильное приложение должно показывать список магазинов в виде маркеров на карте, также у пользователя должна быть возможность посмотреть список магазинов, отображаемых на карте и краткую информацию о каждом магазине: адрес, время работы, описание.

Пользователь должен иметь возможность добавить товар в корзину для последующего оформления заказа. Пользователь должен иметь возможность оформить заказ, указав свои контактные данные: ФИО, номер телефона и e-mail, а также данные заказа: способ доставки, в случае, если магазин поддерживает и самовывоз и доставку курьером, время, когда он хочет забрать заказ либо оформить доставку, адрес магазина, в котором он хочет забрать заказ при самовывозе либо адрес для доставки содержимого заказа курьером и комментарий. Также пользователь должен иметь возможность выбрать способ оплаты заказа, если магазин поддерживает несколько способов оплаты. Приложение должно отправить оформленный пользователем заказ на сервер для последующей передачи этих данных в выбранный магазин или курьеру.

Пользователь должен иметь возможность посмотреть информацию о совершенных им ранее заказах.

Пользователь должен иметь возможность связаться с представителем компании, для которой было создано мобильное приложение, по каналам связи, выбранным при сборке приложения.

Пользователь должен иметь возможность посмотреть информацию о проходящих в сети акциях.

Среди нефункциональных требований можно выделить следующие.

Приложение должно функционировать на устройствах iPhone седьмого поколения (iPhone 5S) и выше.

Приложение должно функционировать на ОС iOS версии не ниже 10.0.

Приложение должно быть доступно только в портретной ориентации.

Приложение должно загружать информацию во внутреннюю базу данных из внешней базы данных, хранящей информацию, отображаемую пользователю в приложении: данные о товарах, магазинах, проходящих акциях и др.

Также в ходе проектирования конструктора были определены функциональные возможности, предоставляемые менеджеру.

Менеджер должен иметь возможность заполнить или изменить настройки для конкретного мобильного приложения, выпускаемого на основе конструктора.

Менеджер должен иметь возможность запустить автоматическую процедуру выпуска мобильного приложения, которая включает в себя применение введенных настроек, создание сборки мобильного приложения и последующей ее загрузки в магазин мобильных приложений.

2.3. Разработка диаграммы вариантов использования

Диаграмма прецедентов (вариантов использования) отражает отношения между актерами и прецедентами системы и позволяет описать систему

на концептуальном уровне. Прецедент – возможность моделируемой системы, часть ее функциональности, благодаря которой актер может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. Диаграмма прецедентов обычно применяется для спецификации внешних требований к системе [14].

В ходе проектирования в системе было выделено два актера.

Пользователь – пользователь мобильного приложения, которому доступна возможность использования всех его функций.

Менеджер – пользователь административной панели, который заполняет настройки для выпуска мобильного приложения.

На основе функциональных требований к разрабатываемой платформе была создана диаграмма вариантов использования пользователя, которая представлена на рис. 2 и диаграмма вариантов использования менеджера, представленная на рис. 3.

Выбрать город – выбрать город для просмотра списка магазинов данного города и возможности оформить заказ в этом городе.

Выбрать магазин – выбрать магазин для возможности оформить заказ в этом магазине.

Просмотреть информацию о товарах – посмотреть различную информацию о товарах сети или магазина: название, стоимость товара, краткое описание, некоторые характеристики и др.

Просмотреть информацию о проходящих акциях – посмотреть проходящие акции магазина или сети.

Добавить товар в корзину – добавить товар в корзину с указанием количества добавленного товара для последующей возможности оформить заказ.

Оформить заказ – оформить заказ, состоящий из ранее добавленных товаров в корзину с указанием своих контактных данных: ФИО, контактного номера телефона, адреса электронной почты, а также данных заказа: способа получения, места и времени получения, комментария к заказу.

Посмотреть историю заказов – посмотреть историю оформленных заказов.

Связаться с представителем компании – связаться с помощью различных каналов связи с представителем компании для получения ответа на интересующий вопрос.

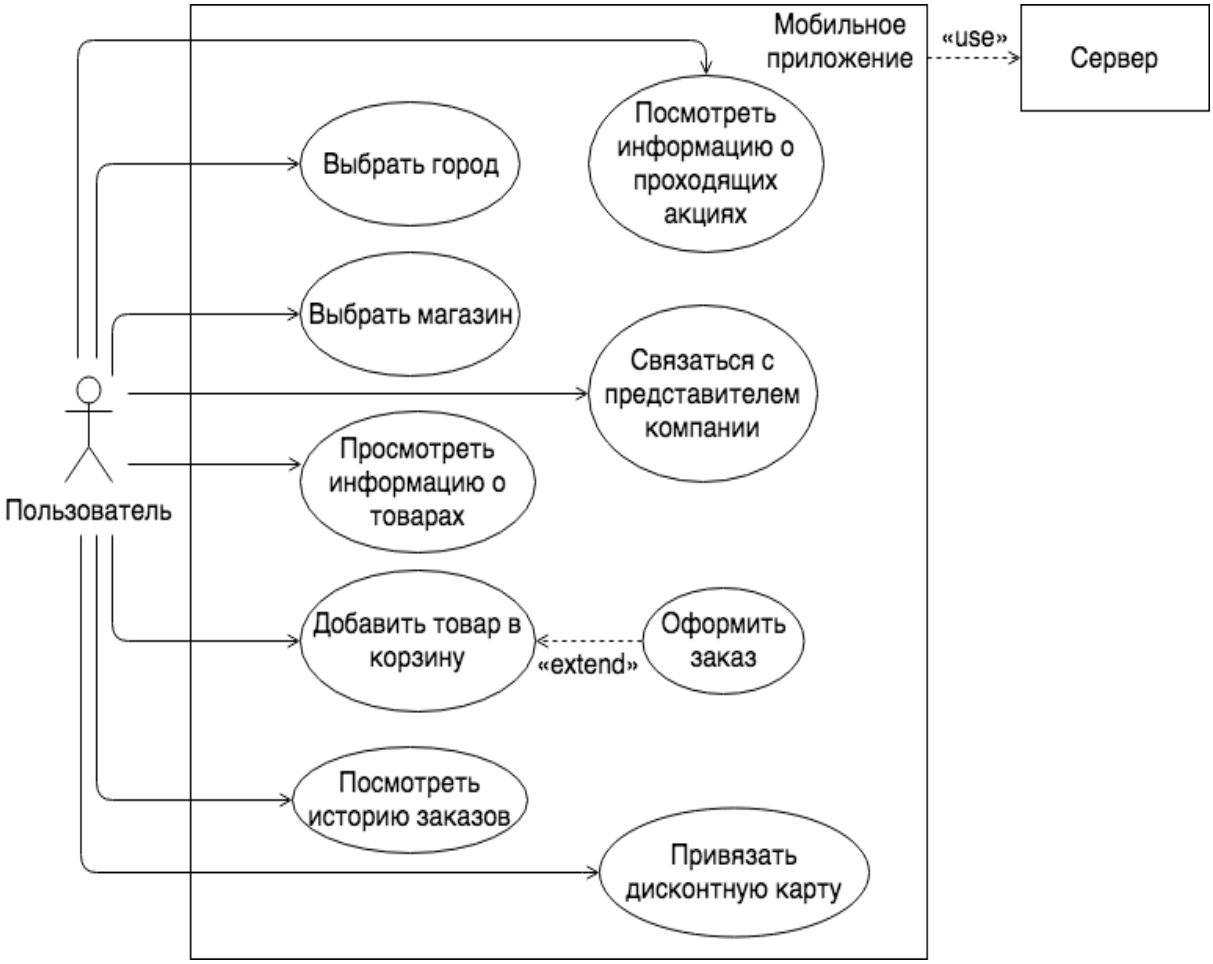

Рис. 2. Диаграмма вариантов использования пользователя

Привязать дисконтную карту – привязать карту к устройству для дальнейшего ее использования при покупке, как альтернатива пластиковой карты.

Работа мобильного приложения зависит от сервера. Все данные, отображаемые в приложении актуализируются с сервера каждый раз при заходе пользователем на экран, отображающий эти данные. Также в ходе работы приложения на сервер отправляются данные об оформленных пользователем заказах. Сервер сообщает мобильному приложению об успешности завершения этих действий, после чего приложение показывает соответствующее уведомление пользователю.

Заполнить настройки – заполнить настройки для мобильного приложения: иконку и название приложения, количество и порядок функциональных модулей, названия визуальных элементов, элементы, относящиеся к бренду (иконка и название мобильного приложения, ссылки на программу лояльности и политику конфиденциальности и т. д.).

Выпустить мобильное приложение – запустить автоматическую процедуру выпуска мобильного приложения, которая включает в себя применение введенных настроек, создание сборки мобильного приложения и последующей ее загрузки в магазин мобильных приложений.

Сохраненные настройки для конкретного мобильного приложения хранятся на сервере, также на сервере выполняется процедура автоматической сборки приложения.

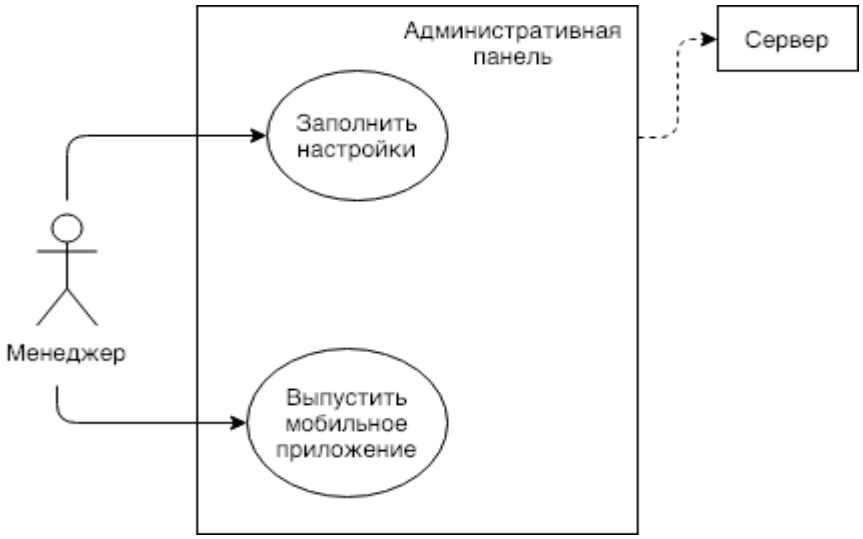

Рис. 3. Диаграмма вариантов использования менеджера

2.4. Разработка диаграммы взаимодействия

На рис. 4 представлена диаграмма взаимодействия, описывающая процесс оформления пользователем заказа. В данном процессе пользователь добавляет в корзину товары, которые он хочет включить в свой заказ. После добавления всех необходимых товаров, пользователь нажимает кнопку «Оформить заказ».

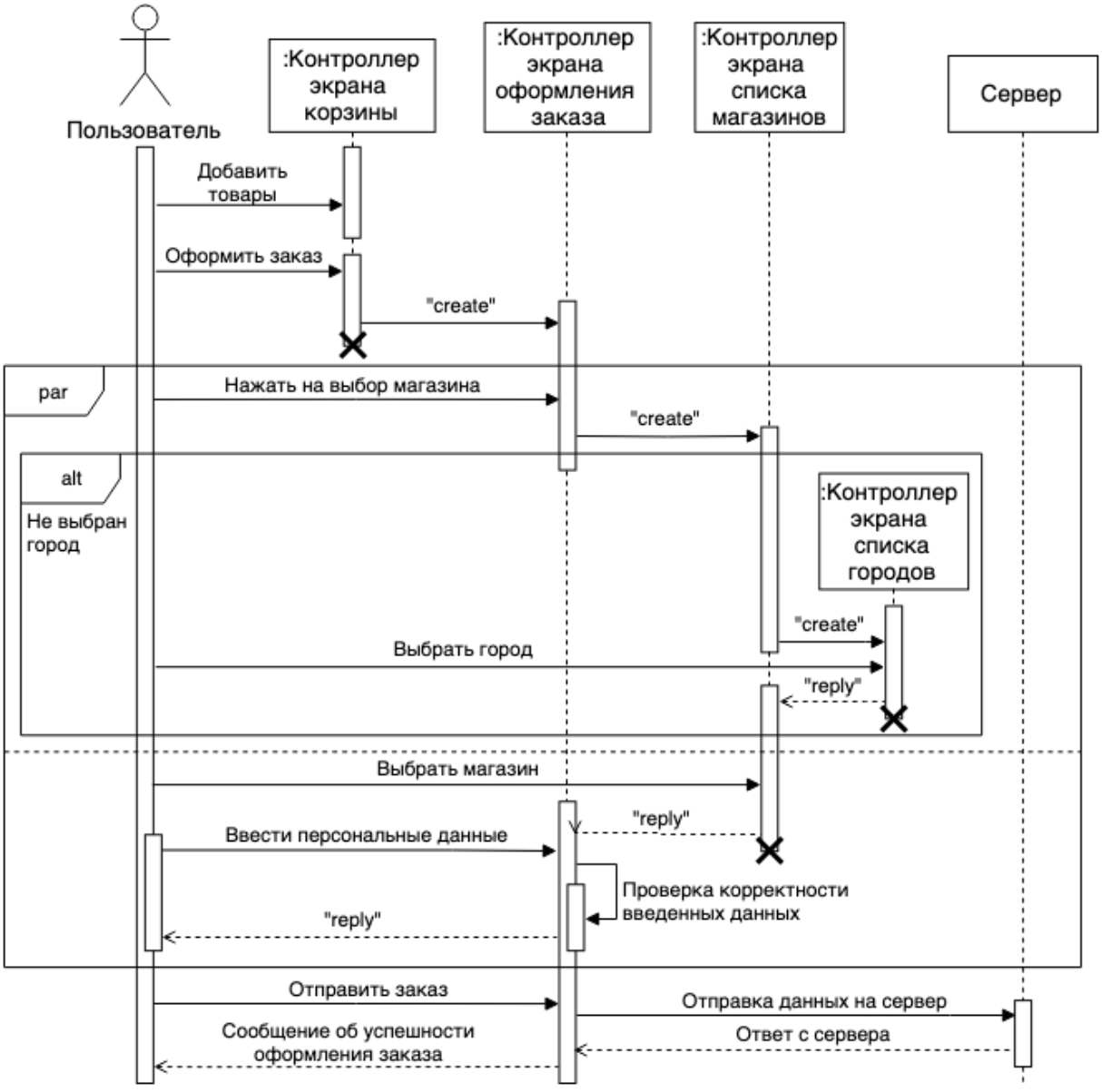

Рис. 4. Диаграмма взаимодействия при оформлении заказа

В момент нажатия кнопки «Оформить заказ» осуществляется переход на экран оформления заказа, где пользователь вводит персональные данные – ФИО, мобильный телефон, и, по желанию, комментарий. После ввода

пользователем данных происходит их проверка на корректность: ФИО должно быть непустым, номер телефона должен соответствовать стандартному формату. Если хотя бы одно поле содержит некорректные данные, приложение уведомит пользователя о необходимости исправить эту информацию.

Также пользователь выбирает магазин, в котором он хочет забрать заказ. При нажатии на кнопку выбора магазина открывается экран выбора магазина. Если до этого момента пользователем не был выбран город, то происходит автоматический переход на экран выбора города, где пользователю будет необходимо выбрать город. После выбора города, происходит автоматический возврат на экран выбора магазина. После выбора магазина происходит автоматический возврат на экран оформления заказа.

Если все введенные данные являются корректными, и пользователь выбрал магазин, то по нажатию на кнопку оформления заказа приложение отправляет информацию о заказе на сервер, получает ответ от сервера об успешности оформления заказа и показывает соответствующее уведомление пользователю.

2.5. Разработка диаграммы деятельности

На рис. 5 представлена диаграмма деятельности мобильного приложения в каталоге при оформлении пользователем заказа.

Начальным состоянием системы является запуск мобильного приложения. При запуске приложения пользователю показывается экран списка категорий. В случае, если категория содержит подкатегории, нажав на любую категорию из списка, пользователь переходит на экран списка подкатегорий. Если выбранная категория не содержит подкатегорий, пользователь попадает на экран списка товаров выбранной категории, где может добавить товар в корзину.

После добавления товаров в корзину пользователь может перейти на экран корзины, где отобразятся все добавленные им товары. При нажатии

на кнопку «Оформить заказ», пользователь попадает на экран оформления заказа. Нажав на экране кнопку «Выбрать магазин», пользователь переходит на экран списка магазинов. При этом, если в приложении ранее не был выбран город, то после этого пользователь сразу попадает на экран списка городов. После выбора города происходит возврат на экран списка магазинов. После выбора магазина происходит возврат на экран оформления заказа. При вводе персональных данных и нажатии на кнопку «Оформить заказ» происходит переход в конечное состояние.

Рис. 5. Диаграмма деятельности

3. РЕАЛИЗАЦИЯ iOS-КОНСТРУКТОРА ДЛЯ СОЗДАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ РОЗНИЧНЫХ СЕТЕЙ

3.1. Методология разработки

Для разработки iOS-конструктора была выбрана инкрементальная модель разработки программного обеспечения.

В инкрементной модели полные требования к системе делятся на различные сборки. Терминология часто используется для описания поэтапной сборки ПО. Имеют место несколько циклов разработки, и вместе они составляют жизненный цикл «мульти-водопад».

Цикл разделен на более мелкие легко создаваемые модули. Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования. Процедура разработки по инкрементной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов». Процесс продолжается до тех пор, пока не будет создана полная система [16, 17, 18].

Данная методология позволяет быстро выпустить первую версию мобильного приложения и легко добавлять новый функционал, а также адекватно рассчитывать трудозатраты и стоимость разработки.

При реализации проекта на первом этапе была выпущена первая версия, включавшая модуль «каталог», предназначенного для просмотра информации о товарах, и модуль «доска», предназначенного для отображения баннеров с различным содержимым и возможностью перехода на различные экраны мобильного приложения, а также на внешние ресурсы.

Также на первом этапе был реализован сборщик приложений, в который далее добавлялись настройки приложения при реализации соответствующих компонентов в мобильном приложении.

На последующих этапах последовательно добавлялись: модуль «магазины» для отображения различной информации о магазинах, модуль «корзина» для возможности формирования пользователем списка товаров и

оформления заказа, включающего добавления товаров, модуль «Лояльность», для возможности привязать дисконтную карту пользователя к мобильному приложению и использовать электронную версию вместо пластиковой при совершении покупок, небольшие модули, содержащие некоторые функциональные возможности для пользователей, а также добавление новых возможностей в уже существующие модули.

3.2. Сборщик приложений

На рис. 6 представлена схема потоков данных сборщика приложений.

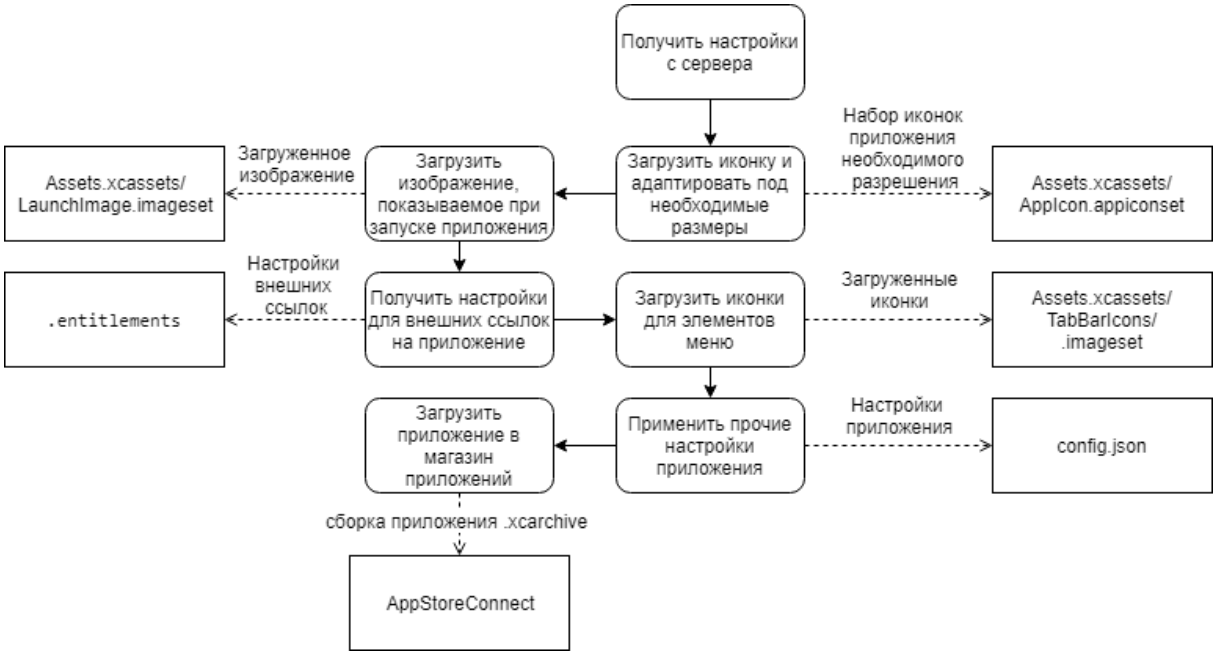

Рис. 6. Схема потоков данных сборщика приложений

В процессе сборки приложения сначала сборщик получает от сервера настройки конкретного приложения, которые были предварительно заполнены с помощью внешнего интерфейса (административной панели).

Далее сборщик загружает иконку мобильного приложения и делает ее копии под различные разрешения экранов, после чего добавляет их в папку проекта мобильного приложения, содержащую иконку.

После этого сборщик загружает изображение, показываемое при запуске мобильного приложения и добавляет его в папку проекта, отвечающую за изображение при запуске.

Далее происходит обновление настроек внешних ссылок путем обновления файла .entitlements.

После этого загружаются иконки для все элементов меню, которые должны содержать иконки, отличные от стандартных, они также добавляются в соответствующие папки.

Далее происходит обновление прочих настроек приложения путем обновления файла config.json, отвечающего за прочие настройки приложения.

Наконец, происходит сборка мобильного приложения и загрузка его в магазин приложений App Store.

3.3. Структура проекта мобильного приложения

Проект мобильного приложения содержит 7 основных компонентов и 2 вспомогательных, взаимодействие которых представлено на рис. 7.

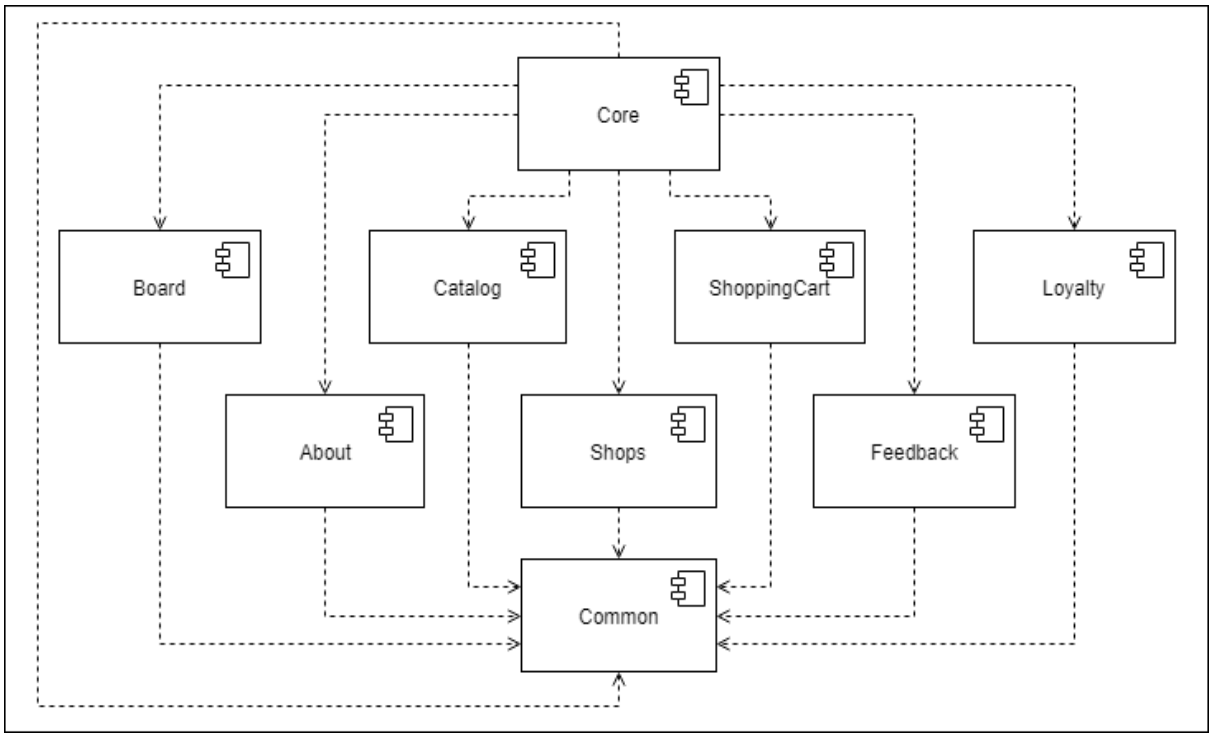

Рис. 7. Диаграмма компонентов

Проект содержит 7 основных компонентов – Board, Catalog, ShoppingCart, Shops, About, Feedback и Loyalty, и 2 вспомогательных – Core и Common. Каждый из основных компонентов реализует определенные функциональные возможности мобильного приложения.

Компонент Board реализует отображение баннеров различного типа с возможностью перехода по ним на другой экран мобильного приложения или на сторонний ресурс: стороннее мобильное мобильное приложение, социальная сеть или веб-сайт.

Компонент Catalog реализует возможность отображения категорий продукции и различной информации о товарах, содержащихся в этих категориях: описание, различные характеристики, детальные изображения, стоимость, наличие и др.

Компонент ShoppingCart реализует возможность формирования пользователем списка товаров и последующего оформления заказа с добавленными товарами.

Компонент Shops содержит экраны карты с магазинами и детальной информации о магазине, а также списка городов в случае, если торговые точки компании-заказчика расположены в более чем одном городе.

Компонент About содержит экран общей информации о компании, включающий в себя изображения и текстовые данные.

Компонент Feedback позволяет пользователю связаться с представителем компании в социальных сетях либо с помощью других средств связи – электронной почты, мессенджера или телефона.

Компонент Loyalty содержит экраны и логику привязки пользователем дисконтной или бонусной карты к приложению.

Во вспомогательном компоненте Common содержатся классы или методы, которые используются в более чем одном основном компоненте. Также в компоненте Common подключены все сторонние библиотеки, используемые в других компонентах.

Компонент Core отвечает за связь основных компонентов между собой, конфигурацию конструктора в готовое мобильное приложение для клиента и главное меню приложения – Tab Bar.

3.4. Архитектура мобильного приложения

Мобильное приложение реализовано с использованием архитектурного шаблона VIPER [9], адаптированного под особенности разработки для операционной системы iOS. Таким образом, каждый экран приложения представлен в виде VIPER-модуля, схема которого представлена на рис. 8.

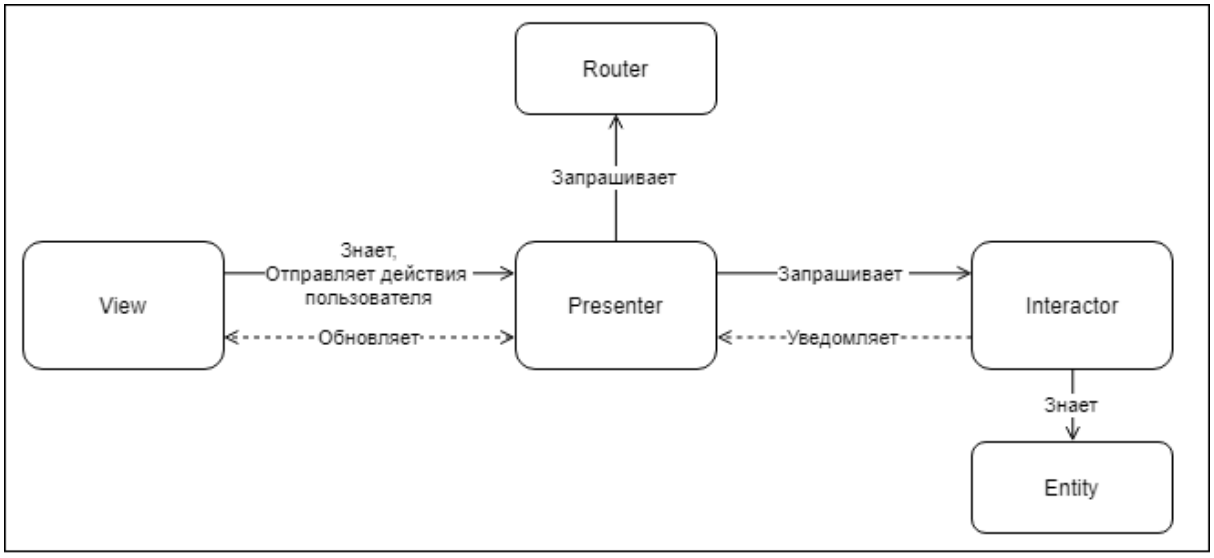

Рис. 8. Схема VIPER-модуля

Каждый VIPER-модуль состоит из 5 основных компонентов - View, Interactor, Presenter, Entity и Router.

Компонент View отвечает за представление экрана мобильного приложения, в рамках разработки для операционной системы iOS он представлен стандартным классом фреймворка Cocoa Touch [2] UIViewController или его дочерними классами.

Компонент Interactor содержит бизнес логику для управления объектами (Entity), чтобы выполнить определенную задачу. В iOS-приложениях чаще всего этот компонент используется для получения информации с сервера и записи ее во внутреннее хранилище данных приложения.

Компонент Entity - объекты, которыми управляет Interactor. Он представляет из себя класс или структуру с определенными полями, но не содержащую бизнес-логику.

Компонент Presenter содержит логику отображения интерфейса, он отправляет запросы компоненту Interactor, получает от него результаты, преобразует их в состояние, которое будет наиболее эффективным для отображения и передает его в компонент View. Также Presenter собирает и обрабатывает входные данные от взаимодействия пользователя с мобильным приложением.

Компонент Router содержит логику навигации между экранами мобильного приложения.

3.5. Модель внутреннего хранилища данных мобильного приложения

На рис. 9 представлена полная схема модели внутреннего хранилища данных в мобильном приложении разрабатываемого конструктора.

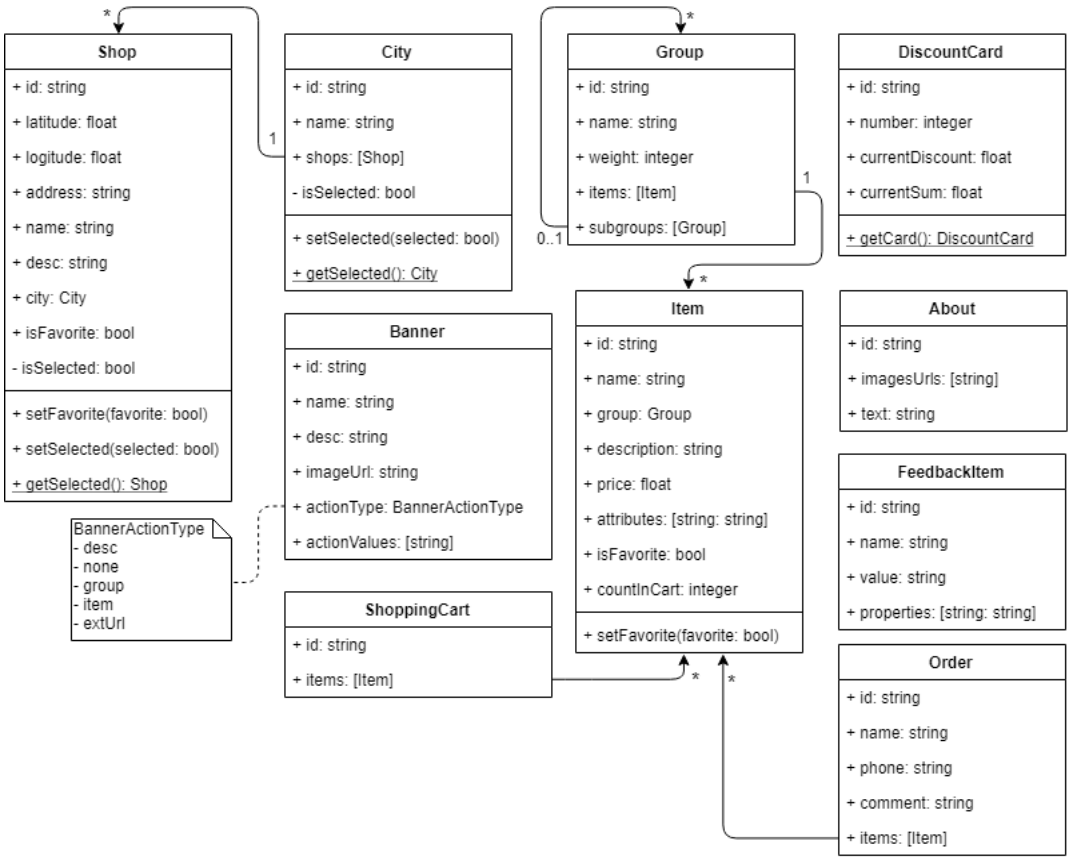

Рис. 9. Схема модели внутреннего хранилища данных

Сами модели расположены в различных компонентах в зависимости от их назначения, но хранятся в едином хранилище данных, которое управляется компонентом Common.

В случае, если клиент не включит в своем приложении определенный функциональный модуль, соответствующий компонент не будет записывать в хранилище данных модели, относящиеся к этому компоненту, поскольку данные запрашиваются с сервера только при заходе на экран конструктора, на котором они отображаются и обрабатываются.

Система содержит 8 различных моделей данных - Category и Product в компоненте Catalog, Shop и City в компоненте Shops, FeedbackItem в компоненте Feedback, AboutItem в компоненте About, Banner и BannerGroup в компоненте Board.

3.6. Особенности реализации приложения

Мною были изучены особенности разработки iOS-приложений [3, 5, 19].

На рис. 10 представлено получение в сборщике приложений файлов иконок нужного разрешения из файла иконки, загруженной при заполнении настроек приложения.

Для этого из папки для иконок в проекте мобильного приложения читается файл конфигурации этих иконок, после чего считываются необходимые для приложения разрешения иконок, и загруженный файл иконки копируется с изменением размера на нужный.

Далее файл конфигурации иконок обновляется в соответствии с названием полученных файлов.

На рис. 11 представлена частичная реализация модели данных FeedbackItem, содержащая поля, хранящиеся в этой модели, инициализацию объекта этого типа и некоторые вспомогательные методы.

На рис. 12 представлена настройка сканера штрих-кода карты для последующей привязки и отображении ее в приложении. В представленной

части метода настраивается камера устройства, типы кодов, которые будут считываться, а также область сканирования кодов на экране.

На рис. 13 представлены методы протокола ESGroupsListInteractor-Input, который служит для связи Presenter и Interactor VIPER-модуля GroupsList, который реализует экран отображения категорий товаров в модуле «Каталог».

Помимо описания сигнатур методов протокола, протокол также содержит документацию этих методов, написанную с помощью стандартных средств документирования среды разработки Xcode, что позволяет получить подробную информацию об этих методах, не уходя с места, где эти методы вызываются.

```
echo $OLD_CONFIG_FILE_CONTENT | jq -c '.images[]' | while read config; do

    # Check if this config is unassigned
    unassigned=$(echo $config | jq '.unassigned')
    [ $unassigned = true ] && continue

    # Getting width (pt), height (pt) and scale of images from config
    widthpt=$(echo $config | jq '.size' | grep -oE '[0-9]+(\.[0-9]+)?' | head -n 1)
    heightpt=$(echo $config | jq '.size' | grep -oE 'x[0-9]+(\.[0-9]+)?' | head -n 1 | tail -c +2)
    scale=$(echo $config | jq '.scale' | grep -oE '[0-9]' | head -n 1)

    # Configuring name for resized image
    name="$APP_ICON_NAME_WITHOUT_EXT-$widthpt"x"$heightpt@$scale"x".png"

    # Coping image to assets folder
    cp $APP_ICON_NAME $APP_ICON_ASSETS_FOLDER/$name

    # Getting width and height in px and resizing image
    widthpx=$(echo $widthpt*$scale | bc)
    heightpx=$(echo $heightpt*$scale | bc)
    sips -z $widthpx $heightpx $APP_ICON_ASSETS_FOLDER/$name > /dev/null

    # Updating config file
    config=$(echo $config | jq ". + {\"filename\": \"$name\"}")
    config="[$config]"
    final=$(cat $TEMP_CONFIG_FILE | jq ".images |= . + $config")
    echo $final | jq '.' > $TEMP_CONFIG_FILE
done
```

Рис. 10. Получение иконок приложения нужного разрешения

На рис. 14 представлен реализация интерфейсов некоторых экранов модуля «Каталог». Все интерфейсы в конструкторе реализованы с использованием Interface Builder [4] - специального инструмента для создания интерфейсов, встроенного в среду разработки Xcode [11].

```
@objcMembers class ESFeedbackEntity: ObjectWithId, Decodable {

    //Realm properties
    dynamic var name: String = ""
    dynamic var value: String = ""
    dynamic var index = 0
    dynamic var oType: String = FeedbackType.website.rawValue
    var extra = List<FeedbackExtra>()

    var type: FeedbackType {
        get { return FeedbackType(rawValue: oType)! }
        set { oType = newValue.rawValue }
    }

    //Realm overrides
    override class func ignoredProperties() -> [String] {
        return ["type"]
    }

    //Init
    convenience init(name: String, value: String, index: Int, typeString: String, extra: List<FeedbackExtra>) {
        self.init()
        self.id = (value + name + typeString)
        self.name = name
        self.value = value
        self.index = index
        self.type = FeedbackType(rawValue: typeString)!
        self.extra = extra
    }
```

Рис. 11. Реализация модели FeedbackItem

```
do {
    guard let captureDevice = AVCaptureDevice.default(for: .video) else { return }
    let input = try AVCaptureDeviceInput(device: captureDevice)
    captureSession = AVCaptureSession()
    captureSession?.addInput(input)

    let output = AVCaptureMetadataOutput()
    captureSession?.addOutput(output)
    output.setMetadataObjectsDelegate(self, queue: .main)
    output.metadataObjectTypes = [.ean13]
    output.rectOfInterest = rectOfInterest

    if captureSession != nil {
        videoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession!)
    }
    videoPreviewLayer?.videoGravity = .resizeAspectFill
    videoPreviewLayer?.frame = view.layer.bounds
    if videoPreviewLayer != nil {
        previewView.layer.addSublayer(videoPreviewLayer!)
    }
    startScanning()
    output.rectOfInterest = videoPreviewLayer?.metadataOutputRectConverted(fromLayerRect: scanningRect) ??
        CGRect(x: 0.0, y: 0.0, width: 1.0, height: 1.0)
    drawFrame(withRect: scanningRect)
    needToSetupCamera = false
} catch {}
```

Рис. 12. Настройка сканирования штрих-кода

```swift
/// Asks interactor to load groups
///
/// - Parameters:
///   - id: id of parent category, if this paraneter is nil, root groups will be
    loaded, default value is nil.
///   - completion: completion block, contains success of the operation, array of
    loaded groups, if success is true and error otherwise.
func loadGroups(withParentGroupId id: String?, completion: ((_ success: Bool, _ groups:
    [ESGroup]?, _ error: NSError?) -> Void)?)

/// Asks interactor to load banners
///
/// - Parameters:
///   - completion: completion block, contains success of the operation, array of
    loaded banners, if success is true and error otherwise.
func loadBanners(completion: ((_ success: Bool, _ banners: [ESBanner]?, _ error:
    NSError?) -> Void)?)
```

Рис. 13. Методы протокола ESGroupsListInteractorInput

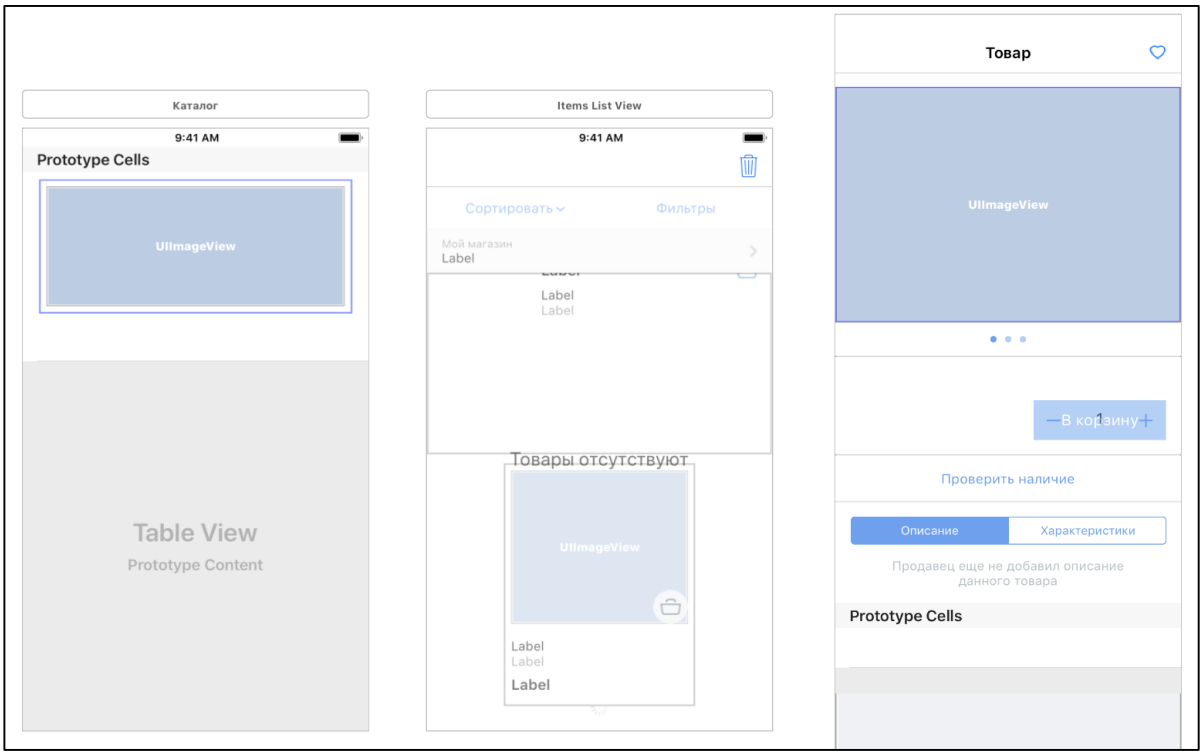

Рис. 14. Реализация интерфейсов экранов в Interface Builder

4. ТЕСТИРОВАНИЕ

4.1. Выбор способов тестирования системы

Из-за тесной связи логики мобильного приложения с его интерфейсом, unit-тестирование является неэффективным способом тестирования системы, т.к. не может обеспечить достаточный процент покрытия исходного кода. Для автоматизации тестирования iOS-приложения в среду разработки Xcode интегрирован фреймворк XCTest [12], который позволяет, в частности, разрабатывать UI-тесты. Для реализации UI-тестов в фреймворке используется механизм записи действий пользователя в режиме реального времени, который реализован в инструменте UI-Recorder, который также интегрирован в среду разработки. В ходе использования этого средства были выявлены следующие недостатки:

- нестабильная работа Xcode в ходе использования инструмента, аварийные завершения среды разработки наблюдались не реже, чем каждый час работы;

- сгенерированный код UI-тестов содержит большое количество ошибок, из-за которых тесты в последствии могут не запуститься;

- UI-тесты записывают конкретные экраны конкретного приложения, из-за наличия большого количества вариантов для конфигурации мобильного приложения и отображения информации, тесты не будут обеспечивать должного покрытия;

- из-за наличия связи приложения с сервером сильно усложняется автоматизированное тестирование, поскольку время получения данных для отображения может быть разным от запуска к запуску, из-за чего тесты в некоторых случаях не будут пройдены, так как произойдет обращение к еще неполученным объектам;

- написание UI-тестов вручную с использованием фреймворка – нетривиальная задача, поэтому высока вероятность допустить ошибку при реализации тестов.

В связи с этими недостатками было принято решение не использовать механизм автоматизированного UI-тестирования, а ограничиться ручным функциональным и интеграционным тестированием системы.

Также при тестировании был проверен сборщик приложений при сборке конечных мобильных приложений на предмет корректности проставления всех параметров, задаваемых в нем и на предмет корректности и работоспособности самой сборки мобильного приложения и загрузки ее в магазин мобильных приложений.

4.2. Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям [6]. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Используя методологию функционального тестирования, проверим работу мобильного приложения.

Тест № 1. Корректное отображение экрана баннеров при различном заполнении.

Входные данные: -

Цель: проверить корректность отображения экрана баннеров при различном количестве и типах баннеров. Баннеры всех трех имеющихся типов (горизонтальная коллекция, два квадратных баннера и одиночный горизонтальный баннер) должны отображаться корректно в соответствии с дизайном.

Ход проведения:

1) добавить в административной панели баннеры всех трех типов;

2) открыть мобильное приложение;

3) перейти на экран баннеров;

4) проверить, что баннеры всех типов отображаются корректно;

5) убрать один из типов баннеров в административной панели;

6) перезагрузить содержимое экрана с помощью жеста pull-refresh;

7) проверить, что баннеры оставшихся типов отображаются корректно;

8) повторить действия 5-7 для всех возможных комбинаций типов баннеров;

Результат: тест пройден.

Тест № 2. Корректность работы экрана обратной связи.

Входные данные: в административной панели заполнены данные всех возможных вариантов обратной связи.

Цель: проверить корректность работы экрана обратной связи, обновление данных на экране и работу приложения при выборе варианта обратной связи.

Ход проведения:

1) открыть мобильное приложение;

2) перейти на экран вариантов обратной связи;

3) убедиться, что показаны все возможные варианты обратной связи;

4) нажать на вариант обратной связи, убедиться в корректности его работы;

5) повторить действие 4 для всех вариантов обратной связи;

6) убрать один или несколько вариантов обратной связи в административной панели;

7) обновить содержимое экрана мобильного приложения с помощью жеста «pull-refresh»;

8) проверить, что убранные в административной панели варианты обратной связи исчезли с экрана.

Результат: тест пройден.

Тест № 3. Корректность работы экрана «О компании».

Входные данные: в административной панели заполнены данные для экрана «О компании».

Цель: проверить корректность отображения и обновления данных на экране «О компании».

Ход проведения:

1) открыть мобильное приложение;

2) перейти на экран «О компании»;

3) убедиться, что данные, заполненные в административной панели, отобразились корректно, в соответствии с дизайном;

4) поменять содержимое экрана в административной панели;

5) обновить содержимое экрана с помощью жеста «pull-refresh»;

6) убедиться, что данные обновились корректно.

Результат: тест пройден.

Тест № 4. Отображение экрана общего каталога.

Входные данные: -

Цель: проверить корректность отображения экрана общего каталога и корректность обновления данных на нем. В верхней части экрана должны быть расположены акции, которые получаются с сервера. Ниже должен располагаться список категорий, который также получается с сервера. В случае отсутствия акций, экран должен состоять только из списка категорий.

Ход проведения:

1) открыть мобильное приложение;

2) перейти на экран общего каталога;

3) убедиться, что на экране корректно отображаются акции и категории;

4) в административной панели убрать все акции;

5) обновить содержимое на экране каталога с помощью жеста «pull-refresh»;

6) убедиться, что акции исчезли с экрана, а список категорий теперь отображается на весь экран.

Результат: тест пройден.

Тест № 5. Корректность отображения данных блока «Вы недавно смотрели» на экране корзины.

Входные данные: -

Цель: убедиться в корректности работы блока «Вы недавно смотрели» на экране корзины. В блоке должно отображаться десять последних товаров, которые посмотрел пользователь и которые не добавлены в корзину.

Ход проведения:

1) открыть мобильное приложение;

2) зайти на экран карточки товара для десяти различных товаров;

3) перейти на экран корзины;

4) убедиться, что все просмотренные товары отображаются в блоке «Вы недавно смотрели» в порядке, обратном просмотру товаров в приложении (самый последний просмотренный товар должен быть первым в блоке);

5) добавить несколько товаров из блока в корзину;

6) убедиться, что добавленные товары исчезли из блока «Вы недавно смотрели».

Результат: тест пройден.

Тест № 6. Корректность работы фильтра по цене в категории товаров.

Входные данные: мобильное приложение открыто на экране категории товаров.

Цель: проверить корректность работы фильтра по цене.

Ход проведения:

1) перейти на экран фильтров;

2) выставить значения фильтра по цене, отличные от начальных;

3) применить фильтр;

4) убедиться, что на экране категории товаров отсутствуют товары, неудовлетворяющие примененному фильтру;

5) перейти на экран фильтров;

6) убедиться, что примененные значения фильтра по цене корректно отображаются на экране.

Результат: тест пройден.

Тест № 7. Корректность работы сортировки на экране категории.

Входные данные: мобильное приложение открыто на экране категории товаров.

Цель: проверить корректность работы сортировки товаров на экране категории.

Ход проведения:

1) убедиться, что по умолчанию выбрана сортировка по возрастанию цены;

2) убедиться, что товары отсортированы по возрастанию цены;

3) выбрать сортировку по алфавиту;

4) убедиться, что товары отсортированы по алфавиту;

5) выбрать сортировку по убыванию цены товаров;

6) убедиться, что товары отсортированы по убыванию цены;

Результат: тест пройден.

Тест № 8. Корректность оформления заказа.

Входные данные: приложение открыто на экране корзины, в корзину добавлен один или несколько товаров.

Цель: проверить корректность оформления заказа

Ход проведения:

1) перейти на экран оформления заказа;

2) убедиться, что заказ не оформляется без заполнения обязательных полей, соответствующие поля подсвечиваются красным цветом;

3) ввести данные в обязательные поля;

4) нажать на кнопку «Оформить заказ»;

5) убедиться, что заказ был успешно оформлен, осуществился автоматический переход на экран корзины, корзина была очищена, появилось уведомление об успешно оформленном заказе.

Результат: тест пройден.

Тест № 9. Корректность отображения последнего заказа, списка заказов и экрана деталей заказа.

Входные данные: приложение открыто на экране корзины, предварительно был оформлен заказ.

Цель: проверить корректность работы модуля совершенных заказов: блока последнего заказа на экране корзины, экрана со списком всех заказов пользователя и экрана деталей заказа.

Ход выполнения:

1) убедиться, что в корзине отображается совершенный заказ;

2) убедиться, что при нажатии на последний заказ осуществляется переход на экран деталей этого заказа;

3) вернуться на экран корзины, перейти на экран списка заказов;

4) убедиться, что список состоит из одного заказа, при нажатии на него осуществляется переход на экран деталей этого заказа;

5) совершить еще один заказ в приложении;

6) вернуться на экран корзины;

7) убедиться, что на экране корзины последний заказ поменялся на только что оформленный;

8) зайти на экран списка заказов, убедиться, что теперь он состоит из двух заказов.

Результат: тест пройден.

Тест № 10. Корректность привязки карты лояльности.

Входные данные: -

Цель: проверить корректность привязки карты лояльности к мобильному приложению.

Ход проведения:

1) открыть приложение;

2) перейти на экран ввода карты лояльности;

3) ввести карту, привязанную к доступному номеру телефона;

4) убедиться, что приложением осуществился переход на экран ввода SMS-кода подтверждения;

5) убедиться, что на этот номер телефона пришел код подтверждения привязки;

6) ввести код подтверждения;

7) убедиться, что автоматически осуществился переход на экран информации о карте лояльности;

Результат: тест пройден.

4.3. Интеграционное тестирование

Интеграционное тестирование – полная проверка программного продукта после его сборки с целью выявления ошибок, возникающих в процессе интеграции программных модулей или компонентов [6].

Мобильное приложение было проверено на всех возможных различных разрешениях экранов iOS-устройств, а именно: 640x1136px (iPhone SE), 750x1334px (iPhone 8), 1080x1920px (iPhone 8+), 2436x1125px (iPhone XS), 2688x1242px (iPhone XS Max), 1792x828px (iPhone XR), на предмет корректности отображения элементов интерфейса в различных конфигурациях отображения данных. Все элементы и экраны мобильного приложения были отображены без ошибок на всех вышеперечисленных разрешениях экрана. Также была произведена проверка мобильного приложения на последних трех поддерживаемых версиях операционных систем iOS: iOS 10, iOS 11 и iOS 12. В ходе проверки на разных версиях операционных систем ошибок также не было выявлено.

Скриншоты основных экранов мобильного приложения представлены в приложении.

ЗАКЛЮЧЕНИЕ

Роль мобильных приложений в повседневной жизни растет постоянно. Многие люди пользуются несколькими десятками приложений ежедневно. Практически каждый день выпускаются новые мобильные приложения и обновления для старых. Скачивание приложений не требует долгого времени и особых навыков, установка также проста и понятна [15].

Целью работы являлась разработка iOS-конструктора для создания мобильных приложений для розничных сетей. Для достижения данной цели были решены следующие задачи:

1) произведена постановка задачи;

2) произведен обзор существующих аналогов;

3) изучены современные платформы и средства разработки для операционной системы iOS;

4) определены требования и спроектирован конструктор;

5) реализован конструктор;

6) протестирован конструктор;

Все поставленные задачи были решены, цель достигнута.

Разработанный конструктор имеет перспективы дальнейшего развития. В связи с усложнениями бизнес-процессов и ростом требований к мобильным приложениям у розничных сетей, возникает потребность в расширении функционала конструктора.

В перспективе планируется реализовать следующие возможности:

- внедрить механики лояльности для пользователей: списывание и получение баллов за заказы через мобильное приложение, возможность добавления к заказу товаров-подарков, персональную систему скидок для пользователя;

- внедрить интеллектуальный блок товаров-рекомендаций, который будет индивидуален для каждого пользователя;

- расширенную поддержку карт лояльности различных типов.

ЛИТЕРАТУРА

1. App Store. [Электронный ресурс] URL: https://itunes.apple.com/ru/genre/ios/id36?mt=8 (дата обращения: 01.05.2019).

2. Cocoa Touch. [Электронный ресурс] URL: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html (дата обращения: 08.08.2016).

3. Hegarty P. Standford University Lecture – Developing iOS7 App. [Электронный ресурс] URL: https://www.youtube.com/watch?v=ZqKbN_C4Yvg&list=PLnOdYr35FyvhDUAIW17vo7nGfHJAyikUp (дата обращения: 11.03.2017).

4. Interface Builder Built-In. [Электронный ресурс] URL: https://developer.apple.com/xcode/interface-builder/

5. iOS Developer Library. [Электронный ресурс] URL: https://developer.apple.com/library/ios/navigation/ (дата обращения: 11.03.2017).

6. Kaner Cem, Falk Jack, Nguyen Hung Quoc. Testing Computer Software. – USA: Wiley Computer Publishing, 1999. – 479 p.

7. Realm. [Электронный ресурс] URL: https://realm.io/docs/swift/latest/

8. The State of Mobile in 2019 – The Most Important Trends to Know. [Электронный ресурс] URL: https://www.appannie.com/ru/insights/market-data/the-state-of-mobile-2019/

9. VIPER Design Pattern in iOS, Swift 4. [Электронный ресурс] URL: https://medium.com/swift-india/viper-architecture-example-in-ios-in-swift-4-6f656a441f7c

10. Xamarin. [Электронный ресурс] URL: https://www.xamarin.com/ (дата обращения: 23.02.2017).

11. Xcode в AppStore для Mac. [Электронный ресурс] URL: https://itunes.apple.com/ru/app/xcode/id497799835?mt=12 (дата обращения: 09.02.2017).

12. XCTest. [Электронный ресурс] URL: https://developer.apple.com/reference/xctest (дата обращения: 15.05.2017).

13. Арлоу Дж., Нейштадт А. UML 2 и унифицированный процесс. – М.: Символ-Плюс, 2007. – 624 р.

14. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. – СПб.: Издательство «Питер», 2003. – 432 с.

15. Для чего нужны мобильные приложения? [Электронный ресурс] URL: http://for-gsm.ru/publ/igry_i_soft/dlja_chego_nuzhny_mobil-nye_prilozhenija/4-1-0-514 (дата обращения: 03.03.2017).

16. Закис А. RUP и другие методологии разработки ПО. Ч. 1. Сравнение методологий разработки ПО. // КомпьютерПресс, 2006. – № 3. – С. 174-175.

17. Закис А. RUP и другие методологии разработки ПО. Ч. 2. Сравнение методологий разработки ПО. // КомпьютерПресс, 2006. – № 8. – С. 158-159.

18. Закис А. RUP и другие методологии разработки ПО. Ч. 3. Сравнение методологий разработки ПО. // КомпьютерПресс, 2006. – № 11. – С. 170-173.

19. Марк Д., Наттинг Дж. iOS 6 SDK Разработка приложений для iPhone, iPad и iPod touch. – США: APress, 2013. – 672 р.

20. Обзор рынка мобильных приложений. [Электронный ресурс] URL: https://megamozg.ru/post/16122/ (дата обращения: 03.03.2016).

21. Разработка мобильного приложения для формирования и оформления заказов на продукцию в розничной сети магазинов. [Электронный ресурс] URL: http://omega.sp.susu.ru/publications/bachelorthesis/17-Mihaylov.pdf

ПРИЛОЖЕНИЕ

Рис. 1. Экран баннеров

Рис. 3. Экран категории

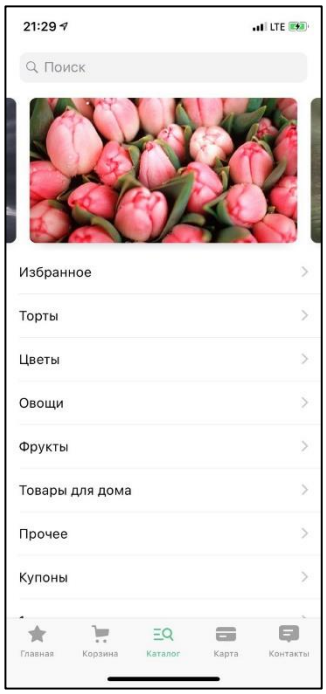

Рис. 2. Экран каталога

Рис. 4. Экран информации о акции

Рис. 5. Экран «О компании»

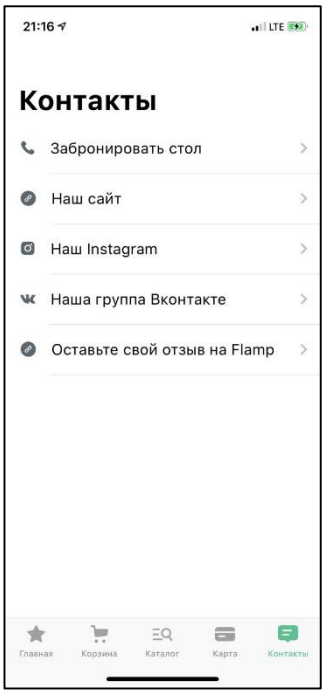

Рис. 6. Экран контактов

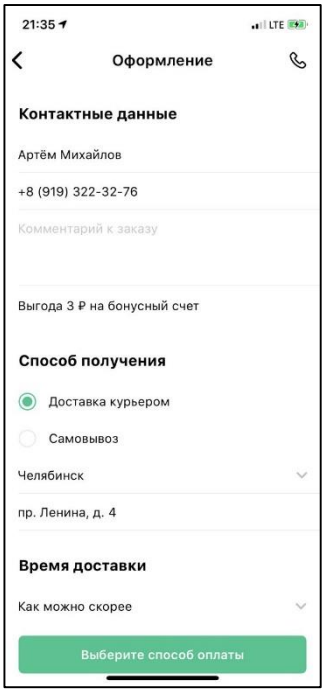

Рис. 7. Экран оформления заказа

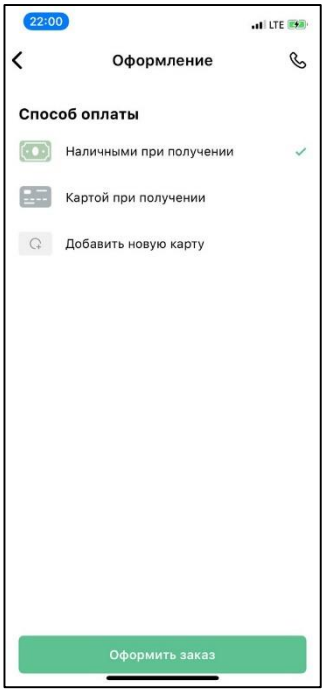

Рис. 8. Экран выбора способа оплаты

Рис. 9. Экран карточки товара

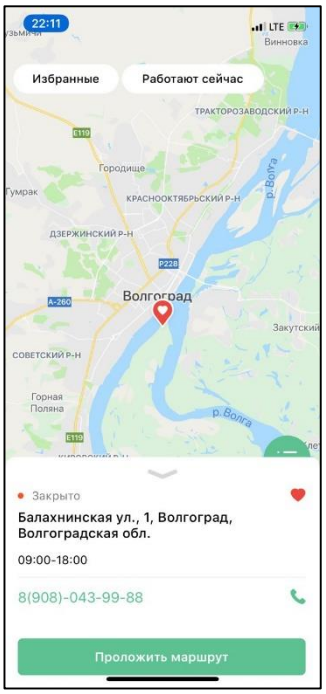

Рис. 10. Экран карты с магазин

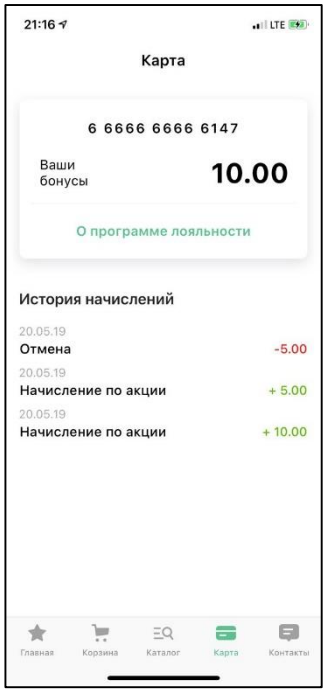

Рис. 11. Экран бонусной карты