Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 09.03.04 Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., доцент
_____А.А.Замышляева
«____»_____2019 г.

Разработка Android приложения для работников федеральной сети секций робототехники «Лига роботов»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.04.2019.114.ПЗ ВКР

Руководитель работы, доцент кафедры ПМиП
_____/ А.К. Демидов
«____»_____2019 г.

Автор работы
Студент группы ЕТ-414
_____/ А.Д. Юрьев
«____»_____2019 г.

Нормоконтролер, доцент кафедры ПМиП, к.т.н.
_____/Т.Ю. Оленчикова
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Юрьев А.Д. Разработка мобильного приложения для сети секций робототехники «Лига Роботов». – Челябинск: ЮУрГУ, ЕТ-414, с. 44, ил. 20, библиогр. список – 20 наим., прил. 2.

Цель данной работы – разработать мобильное приложение для дальнейшего использования работниками сети секций робототехники «Лига Роботов». В процессе работы над квалификационной работой спроектированы интерфейс и архитектура приложения, разработаны и описаны основные алгоритмы решения. На основании выполненной работы произведена программная реализация системы, произведено тестирование и отладка.

Первый раздел работы содержит описание предметной области, анализ системных требований к программному обеспечению и анализ работы секции робототехники, обоснование выбора средств разработки.

Второй раздел состоит из диаграмм системы, отображающих, как взаимодействуют пользователь с системой, а также как взаимодействуют узлы системы.

Третий раздел полностью посвящен реализации разработки мобильного приложения: в данном разделе описываются алгоритмы приложения, этап тестирования и отладки приложения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Число пользователей мобильных устройств с каждым днем стремительно растет, это приводит к расширению рынка мобильных приложений. Ежедневно запускаются сотни приложений, каждый месяц появляются новые тренды в дизайне и принципы построения интерфейса, каждый год изменяются платформы для их разработки – развитие этой области непрерывно и обусловливает актуальность изучения данной темы в этой области. Практически любую функцию, выполняемую в прошлом компьютером, сейчас способно выполнить мобильное приложение: различные жанры игр, просмотр фильмов, чтение книг и многое другое. Но несмотря на то, что приложения имеют обширный функционал, основная часть функций направлена на общение пользователей между собой и в меньшей степени на освоение полезных знаний и умений [1].

Мир постепенно движется к тому, чтобы все сферы деятельности человека помещались в его телефоне. Не обошло это и образовательную деятельность. Одним из сильнейших стимулов к таким изменениям является информатизация образования – это целенаправленно организованный процесс обеспечения сферы знаний технологией и практикой создания оптимального использования научно-технических и методологических разборок [2].

Не отстают в этой сфере и дополнительные образовательные секции. Драматические кружки, кружки по созданию фотографий, танцы, спортивные секции и, конечно же, робототехника – неотъемлемая часть современного образования. Вариантов интересных дополнительных образовательных секций на сегодня огромное множество. Они способны развить таланты ребенка, раскрыть весь его потенциал, развить в нем интересную и сформированную личность.

Но для достижения высоких результатов эта сфера должна быть хорошо продумана и должна находиться под постоянным контролем. Если в традиционной школе все органы управления и контроля сформированы годами практики, то дополнительное образование сейчас только нарабатывает опыт.

Одной из главных задач является оповещение преподавателей обо всех изменениях в расписании и составе групп, возможность отмечать посещаемость или отправить фотографии с занятий. В этом может помочь мобильное приложение, которое будет доступно преподавателю в любое время и в любой день, а котором также будут данные об обучающихся и их контактная информация.

1 ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПРЕПОДАВАТЕЛЕЙ

1.1 Описание предметной области

1.1.1 Характеристика федеральной сети секций робототехники «Лига Роботов» в городе Челябинск

Город Челябинск – образовательный город, здесь находится множество гимназий, лицеев, высших учебных заведений, а также дополнительных образовательных секций. «Лига Роботов» основана в 2011 году, основателем стал Пак Николай Юрьевич, магистрант кафедры Автоматики и Вычислительной техники НГТУ. Обучение «держится» на трех «КИТах» (КИТ – клуб инженерного творчества)

Первая ступень системы «КИТ 1.0» направлена на обучение детей от шести до восемнадцати лет квалифицированными педагогами, которые являются студентами и выпускниками технических университетов. Методисты тщательно прорабатывают материал для проведения занятий, для этого используется инструменты LEGO WeDo 1.0, WeDo 2.0, EV3 с использованием графических языков программирования.

Вторая ступень «КИТ 2.0», основным направлением которого является повышение квалификации молодых инженеров, – обучение детей из «КИТ 1.0», обеспечение их поддержкой со стороны профессионалов и обеспечение всем необходимым оборудованием.

Третья ступень «КИТ 3.0» – для профессиональных разработчиков, которые выполняют коммерческие инженерные заказы и выступают в качестве менторов для студентов из «КИТ 2.0»

Деятельность организации направлена на:

- формирование у обучающихся интереса к инженерным профессиям;
- формирование высокого уровня знаний робототехники;
- создание развивающей образовательной сферы для реализации творческих способностей.

1.1.2 Описание структуры и основных бизнес-процессов

Управление организацией осуществляется в соответствии с законодательством Российской Федерации и Челябинской области с учетом особенностей, установленных Федеральным законом «Об образовании в Российской Федерации».

Непосредственное управление осуществляется директором организации, на основании договора с Учредителями ФССР «Лига Роботов».

Организация обладает автономией, под которой понимается самостоятельность в осуществлении образовательной, научной, административной, финансово-экономической деятельности, разработке и принятии локальных нормативных актов.

Непосредственное управление организацией осуществляет директор, который является владельцем ООО «Робочел».

Организационная структура организации, которая должна обеспечить эффективное взаимодействие всех участников образовательного процесса (рисунок 1.1).

Направление деятельности школы робототехники как секции дополнительного образования определяется с согласия директора организации на общих собраниях преподавателей, администратора, методиста, руководителя центра робототехники и заместителя директора.

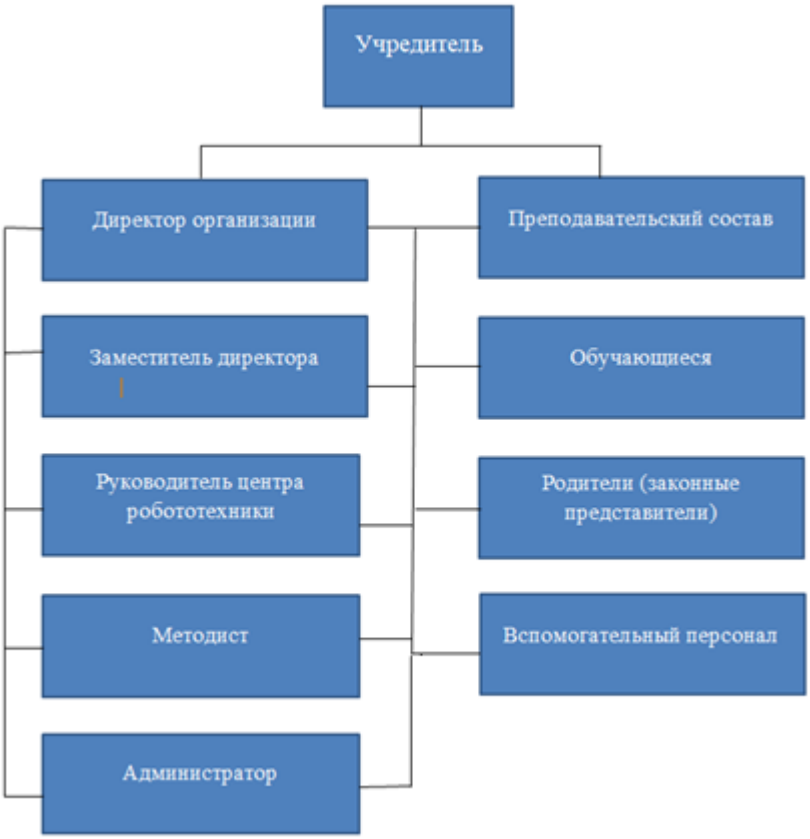

Рисунок 1.1 – Организационная структура ООО «Робочел»

К компетенции Совета относятся:

- участие в разработке программы обучения;
- утверждение основных направлений педагогической деятельности;
- осуществление контроля за ходом и результатами образовательного процесса;
- рассмотрение конфликтных ситуаций, возникающих между администрацией ООО «Робочел», педагогическим составом и родителями (законными представителями), принятие мер по их решению;
- внесение предложений по улучшению роботы, создание поощрительной системы, проведение мероприятий для привлечения новых клиентов.

Собрания преследуют цель развития и совершенствования учебного процесса, повышения уровня мастерства, творческого роста, повышения

интереса к робототехнике, достижения высоких результатов в муниципальных и региональных соревнованиях по робототехнике.

Моделирование бизнес-процессов – это эффективное средство поиска путей оптимизации деятельности образовательной секции, позволяющее определить, как протекает работа в целом и как организована деятельность каждого филиала.

1.2 Описание АльфаCRM

АльфаCRM – CRM-система для разгрузки преподавателей и менеджеров от рутинных задач: ведение расписания, работы с должниками, вести учет клиентов, расчет зарплаты по различным ставкам, формирование отчета, заполнение журналов. Система автоматизирует основные процессы учебного центра, а также является помощником в развитии бизнеса. Сервис поддерживет политику простоты, доступности и дружелюбной технической поддержки.

Возможности АльфаCRM:

- удобная воронка продаж;
- отслеживание источников новых клиентов;
- интеграция с социальными сетями;
- онлайн запись на занятия;
- возможность настроить рассылку электронных писем и СМС-сообщений;
- график работы педагога;
- доходы и расходы по статьям;
- расчет заработной платы;
- возможность легко настроить права доступа;
- ведение нескольких офисов и филиалов.

1.3 Сравнительный анализ аналогичных программных продуктов

Для создания лучшего продукта необходимо иметь четкое представление о том, какой функционал, прежде всего, реализуется в приложениях, разработанных для преподавателей. В данном разделе рассмотрим два мобильных приложения под управлением операционной системы Android.

Целью сравнения представленных ниже конкурирующих приложений – определить сильные и слабые места при реализации функционала, для улучшения своего программного продукта.

1.3.1 Приложение Tallanto teachers

Мобильное приложение для преподавателей с возможностью контролировать учеников с планшета и телефона на операционных системах Android и iOS, а также существует версия для браузера. Пользователь в любое время может просматривать свое расписание, записанных на занятие учеников, отмечать посещаемость, проставлять оценки за занятия, составлять домашнее задание и делать его рассылку ученикам на электронную почту.

Реализованный функционал мобильного приложения:

- возможность фильтровать свои знания;
- отмечать пропущенные учеником занятия;
- возможность выставлять оценки за занятия;
- возможность оставлять комментарии к записям;
- отмечать выполнение домашнего задания;
- выполнять рассылку на электронную почту учеников;
- подводить итоги занятия.

Во вкладке «Мои занятия» преподаватель видит расписание всего учебного центра и имеет возможность отфильтовать только необходимые ему.

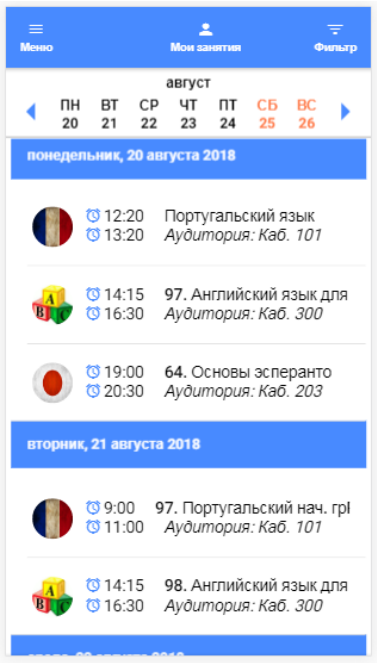

Рисунок 1.2 – Окно приложения
с расписанием занятий

В фильтре преподавателей можно выбрать ответственного за проведение занятия или проведения, каких-либо мероприятий (рисунок 1.2).

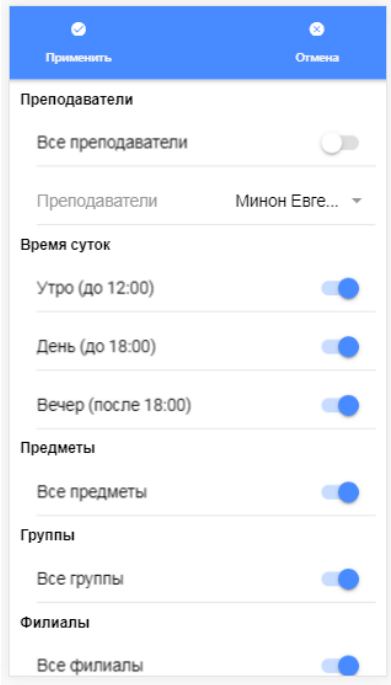

Рисунок 1.3 – Выбор
ответственного преподавателя

В разделе отметок о посещениях есть несколько статусов (рисунок 1.3):

- предварительно записан;
- записан бесплатно;
- аванс;
- полная предоплата;
- посетил бесплатно;
- посетил за баллы;
- прогул;
- не придет;
- в очереди.

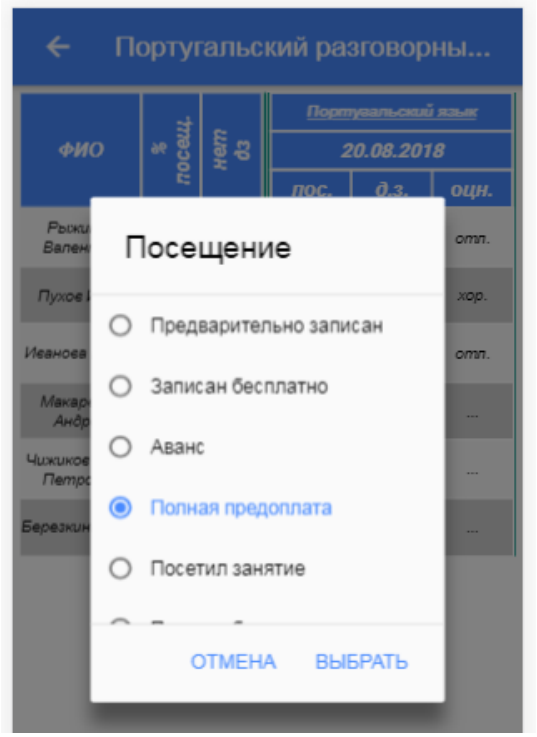

Рисунок 1.4 – Окно для отметок посещаемости

Возможность осуществлять контроль над выполнением домашнего задания, можно осуществить при помощи нескольких нажатий. Таким же образом формируется оценка у учеников, есть возможность оставлять к ним комментарии (рисунок 1.4).

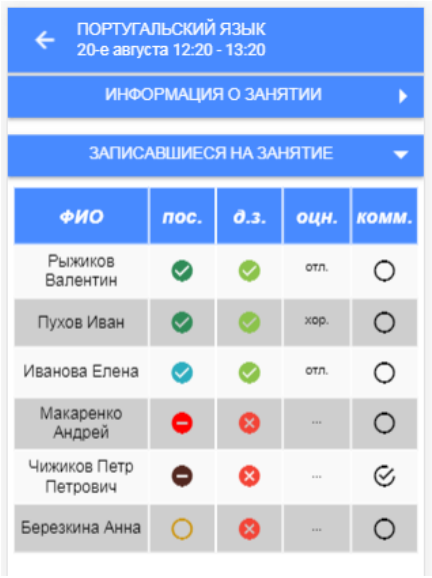

Рисунок 1.5 – Контроль над выполнением домашнего задания

По окончанию урока преподаватель задает домашнее задание и может продублировать его в электронной рассылке (рисунки 1.5, 1.6). Итог всего занятия ученики имеют возможнсть увидеть в своем смартфоне.

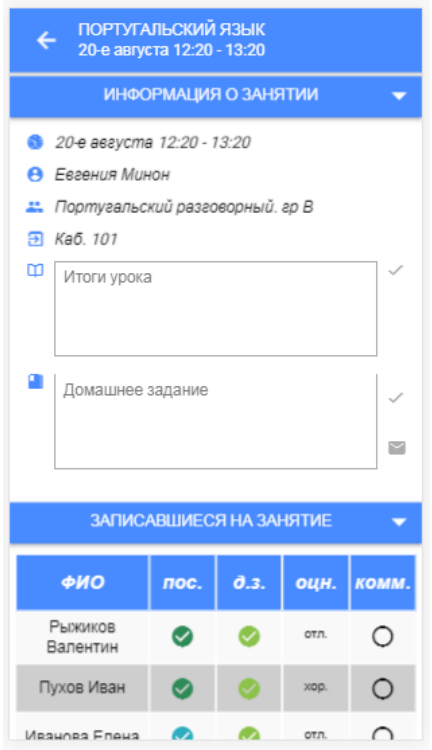

Рисунок 1.6 – Подведение итогов занятия

1.3.2 Приложение MustHave

Следующим приложением для сравнения является приложение MustHave. Расписание, в котором является достаточно удобным, есть возможность выбрать длительность уроков, даты экзаменов и записывать домашнее задание.

Рисунок 1.7 – Окно с расписаниями занятий

Кроме общего расписания, можно выбрать преподавателя и посмотреть только его расписание. С отображениями дней недели и временем начала и конца занятия (рисунок 1.7).

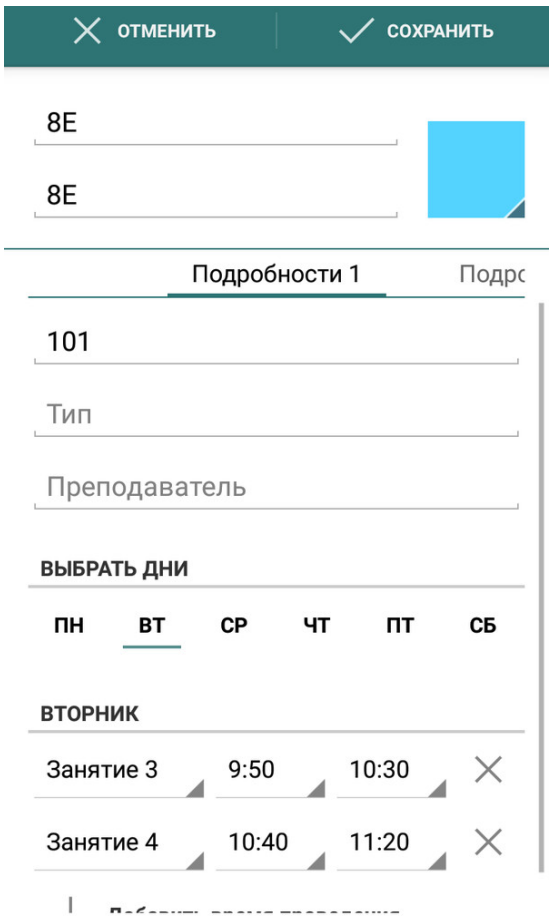

Рисунок 1.8 – Расписание
для преподавателя

Реализованный функционал мобильного приложения (рисунок 1.8):

- создание группы с учениками;
- загрузка материалов в приложение;
- назначение заданий каждому ученику;
- публикаация последних новостей;
- возможность обмена опыта между учителями;
- работа с бейджами.

1.4 Обзор мобильных операционных систем

Разрабатываемое приложение предназначено для использования на мобильных устройствах, для его реализации нужно определиться с выбором

операционной системы. Рассмотрим самые популярные и широко известные на сегодняшний день операционные системы.

1.4.1 Операционная система iOS

iOS – мобильная операционная система для смартфонов, переносимых пригрывателей, планшетов и других устройств, которые разрабатывает и продает компания Apple. Чтобы разработать мобильное приложение для данной оперативной системы необходимо выполниить ряд требуемых условий. Таких как, наличие фирменного компьютера Apple с операционной системой Mac OS. Лишь на данной оперативной системе представляется реальным создать приложение под iPhone и iPad. Конечно, есть вариант установить виртуальную машину на любой другой компьютер с предустановленной Mac OS. Но стоить помнить, что Хакинтош запрещен, так как macOS должно быть только на Apple.

1.4.2 Операционная система Android

Операционная система Android разработана компанией Google для современных электронных устройств, таких как: смартфон, планшет, электронная книга, умные часы, очки Google, телевизор и многих других устройств. В настоящее время операционная система Android занимает 86% объема всего рынка [3]. Данная оперативная система хороша тем, что она поддерживается большим количеством устройств, а также тем, что она является доступной в разработке и не требует больших вычислительных затрат. Еще одним большим преимуществом средств разработки под Android является то, что они являются бесплатными в отличие от конкурентов в лице компании Apple. Библиотеки для работы со сторонними ресурсами операционной системы Android являются бесплатными, например, Horizontal Calendar, AwesomeBar, Android-SwitchIcon, Bridge, ObjectBox, Google Actions

Java SDK и др. Для операционной сисемы Windows Phone Mobile таких библиотек нет.

1.4.3 Операционная система Windows Mobile

Операционная система Windows Mobile – разработанная компанией Microsoft мобильная операционная система для произведенных своей компанией аппаратных платформ Pocket PC, карманные компьютеры и коммуникаторы, а также для смартфонов собстенного производства. На сегодняшний день данная платформа не поддерживается и не поддерживается.

1.5 Средства разработки

Для разработки полноценного приложения с действующим фунционалом, приятным глазу дизайном наиболее важную часть занимает выбор средст разработки. От того какой язык программирования будет выбран во многом зависит успешность реализации проекта. Рассмотрим два наиболее популярных языка программирования на операционной системе Android – Java и Kotlin.

1.5.1 Язык программирования Java

Java является высокоуровневым языком программирования, разработан в 1995 году компанией Sun Microsystems. Может работать на разных платформах, например на Windows, Mac OS и даже UNIX [4].

Среди достоинств языка программирования Java хотелось бы выделить:

- обектно-ориентированность. В языке Java все является объектом. Дополнения могут расширяться, так как язык основывается на объектной модели;
- платформонезависимость. Java, в отличние от многих остальных языков программирования, в том числе C и C++, не компилируется в

платформе и на конкретной машине, а в независимом коде. Такой байт код интерпретируется в работающий на сегодняшний день Java Virtual Machine;

- простота языка. Язык программирования Java остается простым и понятным для разработчиков, если он разбирается в концепции объектно-ориентированного программирования;
- безопасность. Проверка подлинности основана на шифровании с открытым ключом;
- архитектурно-нейтральный. В компиляторе генерируется архитектурно-нейтральные объекты, это позволяет компилировать код, который будет исполняться во многих процессорах;
- портативность. Независимость от реализации аспектов спецификации и архитектурно-нейтральный – это то, что делает портативным язык программирования Java;
- прочность. Для устранения ошибок выполняется тщательная проверка;
- многопоточность. Есть возможность писать программы, в которых выполняются одновременно множество задач;
- интерпретированность. Код, написанный на языке Java переводитсяв машинные инструкции, при этом не сохраняясь. Такой метод делает процесс создания более быстрым и аналитическим, потому что связывание происходит как дополнительное;
- высокопроизводительность. Вводится компилятор Just-In-Time, что позволяет получить еще более высокую производительность;
- распостраненность. Предназначен для распределенной среды Интернет;
- динамичность. Программирование на Java является более динамичным, чем на языках класса С. Язык программирования Java предназначен для адаптации к меняющимся условиям. Программы могут выполнять большое количество во время обработки информации,

использующейся для проверки доступка к каждому объекту во время их выполнения.

Необходимые инструменты, которые понадобятся в процессе программирования:

Прежде всего, конечно же, компьютер Pentium 200 МГц с минимальным объемом оперативной памяти 64 Мб. Рекомендуемый обьем оперативной памяти 128 Мб.

Программное обеспечение:

- Linux 7.1, Windows 95/98/2000/7/8 и выше или другая операционная система;
- JDK 5 и выше;
- Текстовый редактор. Например, Блокнот или Notepad.

Недостатки программирования на языке Java:

- платное коммерческое использование. Компания Oracle объявила о взымании платы за коммерческое использование. Такое нововведенеие начнет действовать с 2019 года. Плата будет зависеть от количества пользователей или от количества компьютеров, платить придется так же за обновления и исправление ошибок;
- низкая производительность. У языков высокого уровня производительность является довольно низкой. В языке Java, это не единственная причина низкой производительности. Напимер, приложение, которое очищает память, занимает 20% времени процессора. Плохо настроенная функция кэширования может отнимать значительное количество времени;
- отсутствует нативный дизайн. При создании Android приложений используется Android Studio, в которой есть возможность создания нативного дизайна. Но при создании пользовательского интерфейса для создания нативного дизайна на персональном компьютере нет Java-инструмента;

• многословность и сложность кода. На первый взгляд может показаться, что многословность является преимуществом, однако, сложные и длинные предложения делают чтение кода более сложным.

1.5.2 Язык программирования Kotlin

Kotlin – статически типизированный язык программирования. Язык начал разрабатываться с 2010 года, представлен был лишь в 2011. Название произошло от острова Котлин в Финском заливе, на котором расположен город Кронштадт. Основная философия языка Kotlin в том, что он во всех проявлениях будет лучше, чем Java.

Преимущества языка Kotlin:

• лаконичность языка. Kotlin похож на Swift. Swift в свою очередь является очень лаконичным языком. Придерживается идеалогии, что код читается как английский текст;

• расширения, касты, именованные аргументы и множество других киллер-фич языка;

• имеет поддержку со стороны компании Google, на сегодняшний день вышло несколько стабильных версий;

• совместимо с Java. Классы Java можно использовать в Kotlin и наоборот. Пользоваться данной функцией можно даже в одном проекте;

• не является академическим языком. Kotlin – это индустриальный язык, созданный разработчиками специально для разработки приложений под Android.

Недостатком разработки приложений на языке программирования Kotlin является относительно небольшое сообщество и слабое продвижение.

Еще полгода назад недостатков было гораздо больше. Kotlin постоянно совершенствуется.

1.6 Выбор средств разработки

Не малое значение в разработке приложений имеет выбор среды разработки. Среда необходима для облегчения и упрощения процесса разработки. Код можно писать даже в блокноте, но если допустить ошибку и поставить отступ не в том месте, то код уже не выполнится, как задумывалось ранее. Среда разработки не допускает такого. Рассмотрим две из них: Android IDE и Android Studio.

1.6.1 Android IDE

Android IDE – среда разработки мобильных приложений для платформы Android. Основана на интегрированной среде разработки приложений Eclipse. Имеет встроенные инструменты для создания, компиляции, сборки и отладки разрабатываемых приложений. В настоящий момент прекращается поддержка инструментов для разработки в операционной системе Android для среды разработки Android IDE.

Преимущества Android IDE:

- настраиваемый интерфейс;
- простая установка;
- большое количество доступной документации;
- неограниченна и бесплатная лицензия;
- большая библиотекаа плагинов.

Недостатки Android IDE:

- при разработке и отладки может происходить зависание, за счет работы с Eclipse, требуется перезапускать программу;
- прекращение обновления и поддержки создателями;
- устаревший сборщик;
- сложность работы с эмулятором.

1.6.2 Android Studio

Среда разработки Android Studio – это среда разработки под операционную систему Android мобильных приложений. Основанна на интегрированной среде разработки программного обеспечения IntelliJ IDEA. Так же, как и в Android IDE, в Android Studio содержится встроенный инструментарий для разработки и отладки приложений.

Бонусом к основным возможностям в Android Studio существуют следующие возможности:

- поддержка автоматической сборки Gradle;
- инструментарий для поиска и устранения проблем;
- система рефакторинга кода;
- поддержка облачной платформы Google Cloud Platform.
- окно просмотра, позволяющего запустить приложение в реальном времени сразу на нескольких устройствах.

Преимущества Android Studio:

- понятный интерфейс;
- удобный в пользовании конструктор для интерфейсов, при помощи которого можно просматривать отображение экрана на других устройствах, в том числе телевизоров и часов. При этом отображение будет выглядеть так же, как на данной версии операционной системе;
- встренный комплекс средств разработки SDK, выдает уведомление о необходимости установки API для запуска предыдущего проекта;
- понятная и удобная структура проекта;
- логи для отслежования ошибок потоков и процессов;
- большое количество литературы на русском языке.

Недостатки Android Studio:

- нет интерфейса на русском языке;

- время компиляции замедляется за счет постоянной автоматической сборки (Gradle);
- высокие системные требования к компьютеру;
- требовательные стандартные эмуляторы, которые не обладают всеми возможностями современных смартфонов.

По соотношению преимуществ и недостатков Android IDE и Android Studio, становится очевидным, что разработка мобильного приложения для операционной системы Android должна происходить в среде разработки Android Studio.

1.7 Постановка задачи

Целью данной работы является создание клиент-серверного мобильного приложения для преподавателей федеральной сети секций робототехники «Лига Роботов» в городе Челябинск на операционной системе Android. Благодаря такому приложению у преподавателя будет личный кабинет, через который они смогут просматривать информацию о своих группах, о каждом отдельно взятом обучающемся, филиале в котором преподаватель работает, просматривать актуальные новости.

Для достижения поставленной цели необходимо решить слующие задачи:

- провести сравнительный анализ аналогичных программных продуктов;
- провести анализ существующих методов разработки мобильных приложений;
- разработать мобильное приложение;
- провести тестирование и отладку программного продукта;
- составить необходимую документацию.

1.8 Вывод по разделу

В данном разделе была поставлена цель создания мобильного приложения для преподавателей сети секций робототехники «Лига Роботов» и задачи для ее достижения. Рассмотрены аналогичные мобильные приложения, выделены их особенности, что повлияет на дальнейшую разработку собственного приложения.

Были рассмотрены современные средства мобильной разработки, мобильные операционные системы, с описанием из преимуществ и недостатков. Выбор был остановлен на операционной системе Android. Так как, данная операционная система является наиболее популярной и простой.

Произведен сравнительный анализ двух популярнейших языков программирования, после которого был выбран язык программирования Kotlin. А также, определена среда разработки приложения Android Studio.

2 РАЗРАБОТКА ПРИЛОЖЕНИЯ

2.1 Интеграция с АльфаCRM

2.1.1 Интеграция АльфаCRM в разрабатываемое приложение

Безопасность АльфаCRM обеспечивается многофакторной авторизацией, доступом по протоколу HTTPS. В АльфаCRM существует интеграционный API, при помощи которой одна программа может взаимодействовать с другой, есть возможность интегрировать с сайтом, лендингом, телефонией, социальными сетями. API реализуется сервисом операционной системы или отдельной программой библиотекой.

Способность интегрировать заявки с сайта поддерживается сторонними разработчиком. Часто реализутся на PHP, но может быть адаптирован под любой другой язык программирования.

В реализуемом мобильном приложении АльфаCRM используется вместо базы данных. Из АльфаCRM будет загружаться расписание занятий, возможность загружать списки групп и отмечать посещаемость и оценки за занятие.

Для дальнейшего внедрения в систему была выбрана АльфаCRM потому, что в ней реализован удобный интерфейс, есть возможность интеграции с телефонией, социальными сетями. Расписание представленно в нескольких вариантах реализации: регулярное, групповое, разовое, индивидуальное. Таким образом, нет необходимости переносить и настраивать дополнительные итерации вручную. Как и в любой другой системе в АльфаCRM так же есть небольшие недочеты, например, в ней нет возможности устанавливать сторонние модули интеграции. Есть возможнсть сделать интеграцию, через какой-либо промежуточный сервис.

В ходе реализации мобильного приложения создавалось отдельное API для формирования расписания занятий, добавления новых филиалов.

2.2 Разработка архитектуры системы

Мобильное приложение под устройства Android ОС состоит из активности, на которое наложены фрагменты, каждому из которых соответствует экран приложения. Каждый фрагмент представлен в проекте в виде класса, реализованного на языке Kotlin, хранящемся в одноименном файле с расширением .kt. [5]Каждому фрагменту соответствует xml файл-описание, в котором описано расположение визуальных объектов. При запуске активности система автоматически распознает размер экрана мобильного устройства и приводит выводимый контент приложения в соответствии с разметкой, описанной в xml-файле. Таким образом, одна и та же активность будет выглядеть одинаково независимо от диагонали экрана используемого устройства. Также, для каждого Android приложения должен быть создан файл с расширением xml, в котором будут прописаны минимальные системные требования, а так же активность, вызываемая при старте приложения.

Мобильное приложение Android использует архитектуру взаимодействия «клиент-сервер». Общая схема взаимодействия представлена на рисунке 2.1

Рисунок 2.1 – Общая схема
взаимодействия «клиент-сервер»

Клиентом является приложение на мобильном устройстве, которое умеет формировать понятный серверу запрос и читать полученный от него ответ.

Сервер представляет собой некую программу, работающую на удаленном компьютере, которая реализует функционал «общения» с приложениями-клиентами, то есть, фактически, слушает запросы, распознает переданные параметры и корректно на них отвечает. Интерфейсом взаимодействия является некий формат получения/передачи ответов/запросов обеими сторонами.

Для получения данных организация-заказчик предоставила сервер. Приложение будет при каждом запуске проверять наличие Интернет-соединения на устройстве, и, если соединение присутствует, приложение будет отправлять запрос серверу, чтобы обновить контент.

2.2.1 Диаграмма классов

Функционально, приложение состоит из приведенных ниже модулей (фрагментов). Фрагмент является схемой представления Android-приложений.

Каждый экран пользовательского интерфейса представлен классом Fragment и по сути является отдельной формой приложения. Android-приложение способно состоять из нескольких фрагментов и может переключаться между ними во время выполнения приложения:

- MainActivity активность на котором располагаются фрагменты, так же на этой активности расположена нижнее меню (оно скрыто до тех пор, пока пользователь не авторизован);
- StartFragment данная активность предназначена для неавторизованного пользователя, запускается на MainActivity, если приложение запущено в первый раз или пользователь вышел из системы. Предлагает пользователю зайти в кабинет или ознакомиться с информацией об организации;
- InfoFragment на данной активности пользователь может озна-комиться с информацией об организации и перейти на следующий фрагмент для записи в секцию;

• EnrollFragment фрагмент представляет из себя WebView, на которой открывается веб-форма записи в секцию;

• OtpConfirmFragment после ввода номера мобильного телефона на StartFragment пользователю необходимо подтвердить свою личность путем ввода четырех символов из смс сообщения;

• MainTeacherFragment когда пользователь уже авторизовался он попадает на данную активность, где содержится информация о пользователе, а снизу появляется меню для перехода на другие активности;

• ScheduleFragment активность отображает расписание преподавателя;

• NewsFragment на данном фрагменте преподаватель может ознакомиться с новостями организации «Лига роботов»;

• FilialsFragment фрагмент представляет из себя карту, на которой с помощью маркеров отображены местоположение филиалов секции.

2.2.2 Диаграмма активности

Поскольку основными элементами приложения для Android ОС являются активности, то схема работы реализуемого приложения представляет собой схему связей между активностями. На рисунках 2.3, 2.4 показана диаграмма активностей UML, демонстрирующая работу всего приложения.

2.1 Выводы по разделу

В данном разделе рассмотрена архитектура и реализация приложения. Выявлены активности, каждую из которых можно отнести к определенному классу, реализованному на языке программирования Kotlin. Так же приведено описание схемы взаимодействия сервера и приложения, диаграмма активности позволяет увидеть схему связей между активностями, которые демонстрируют работу всего приложения.

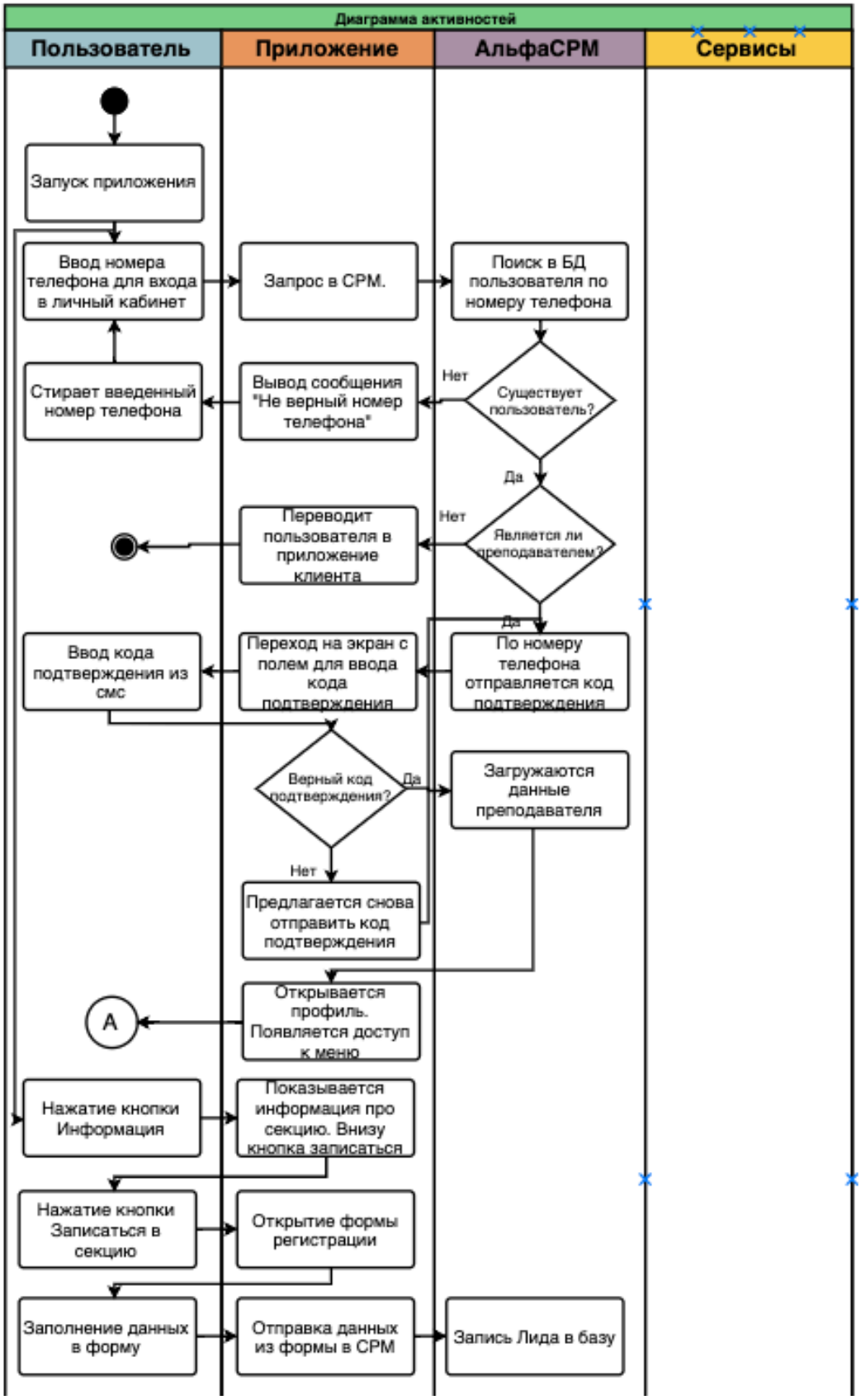

Рисунок 2.3 – Диаграмма активности (начало)

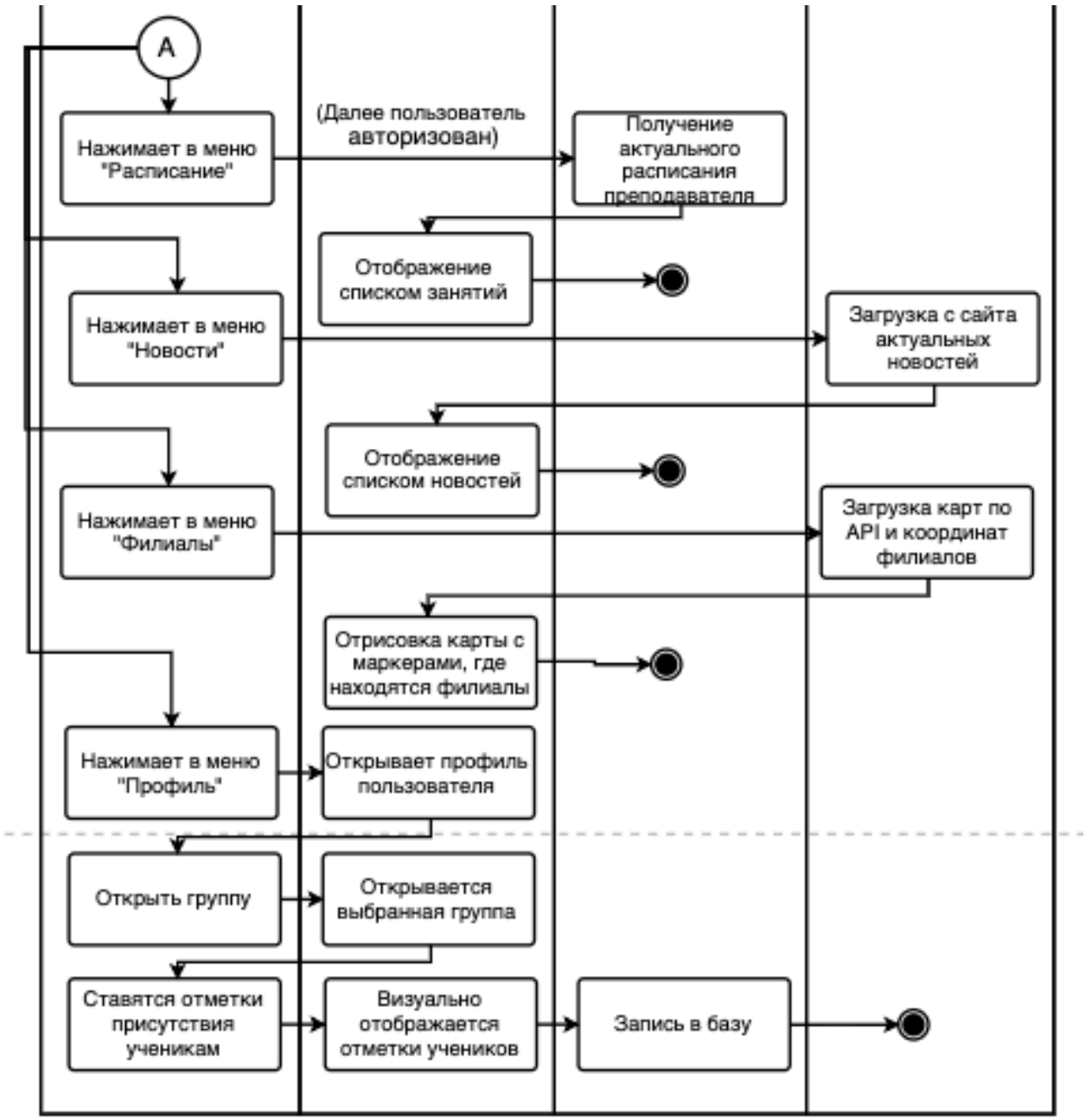

Рисунок 2.4 – Диаграмма активности (окончание)

3 РАЗРАБОТКА И ТЕСТИРОВАНИЕ ПРОГРАММЫ

3.1 Разработка пользовательского интерфейса

В прототипе реализованы главные вкладки приложения, например: «Личный кабинет» (рисунок 3.1), «Информация» (рисунок 3.3), «Записаться в секцию» (рисунок 3.4), «Главная» (рисунок 3.6), «Расписание» (рисунок 3.7), «Новости» (рисунок 3.8), «Филиалы» (рисунок 3.9).

Для создания пользовательского интерфейса необходимо учитывать важные факторы: наглядность, интуитивная понятность интерфейса, приятный дизайн.

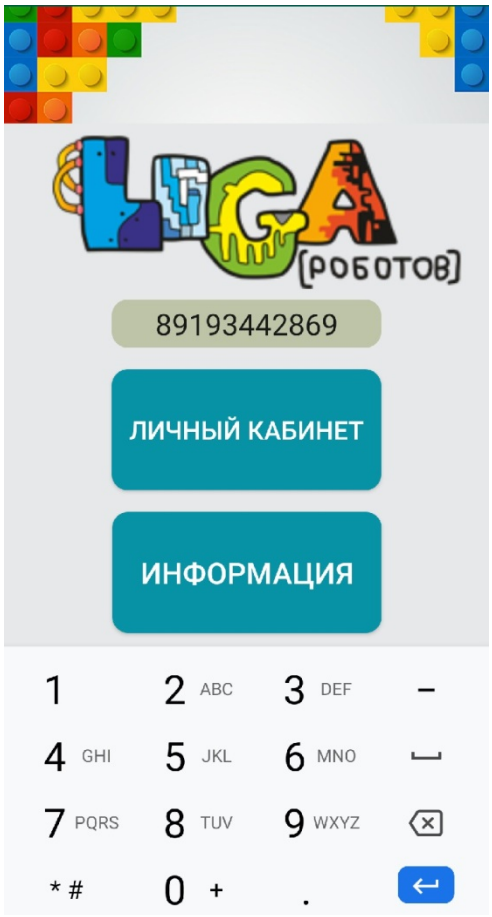

Рисунок 3.1 – Вкладка «Личный кабинет»

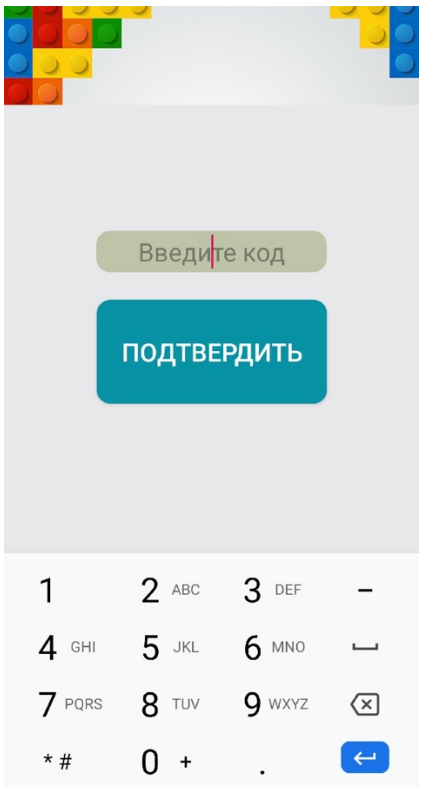

Рисунок 3.2 – Окно ввода кода авторизации

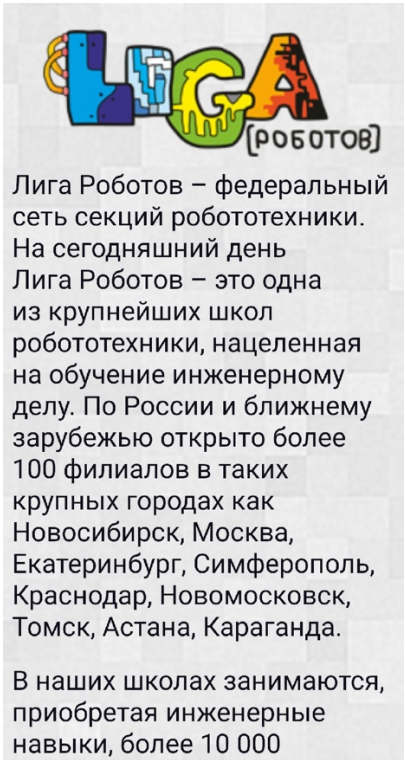

Рисунок 3.3 – Вкладка «Информация»

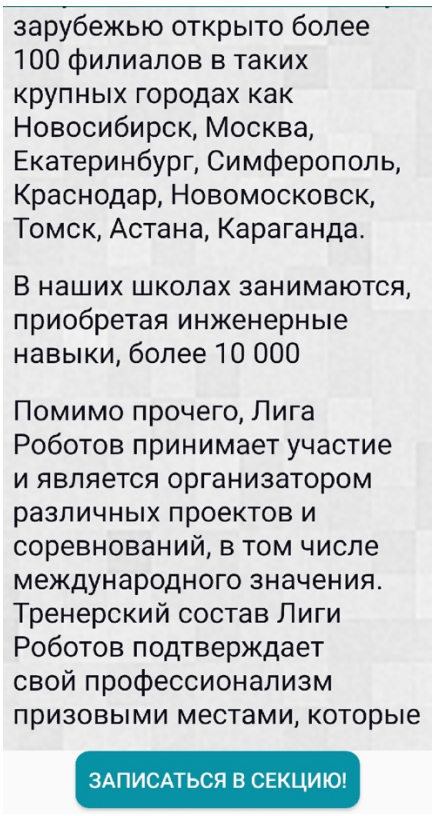

Рисунок 3.4 – Запись в секцию робототехники «Лига Роботов»

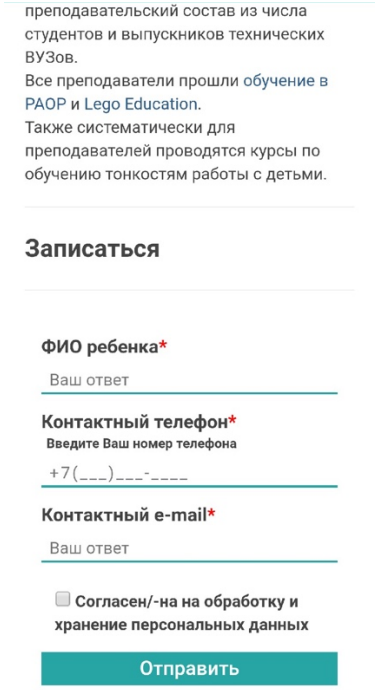

Рисунок 3.5 – Вкладка «Записаться в секцию»

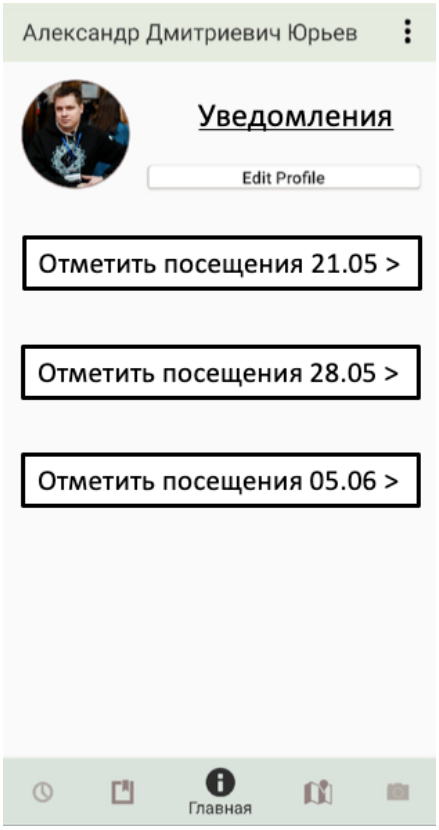

Рисунок 3.6 - Вкладка «Главная»

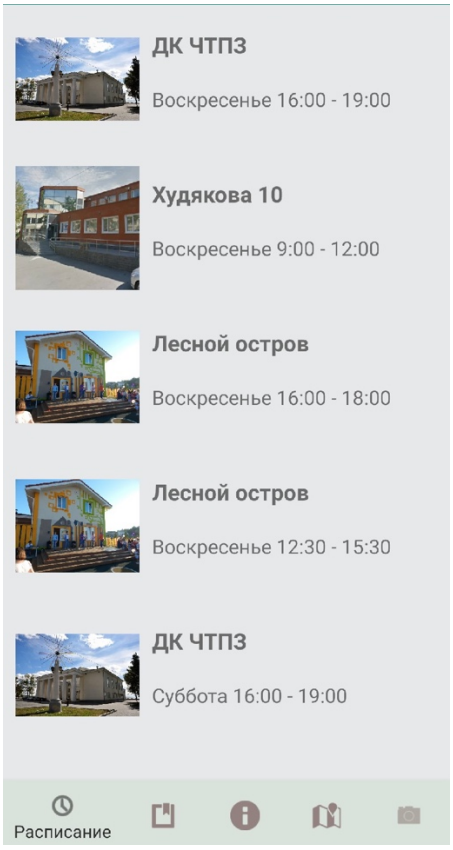

Рисунок 3.7 – Вкладка «Расписание»

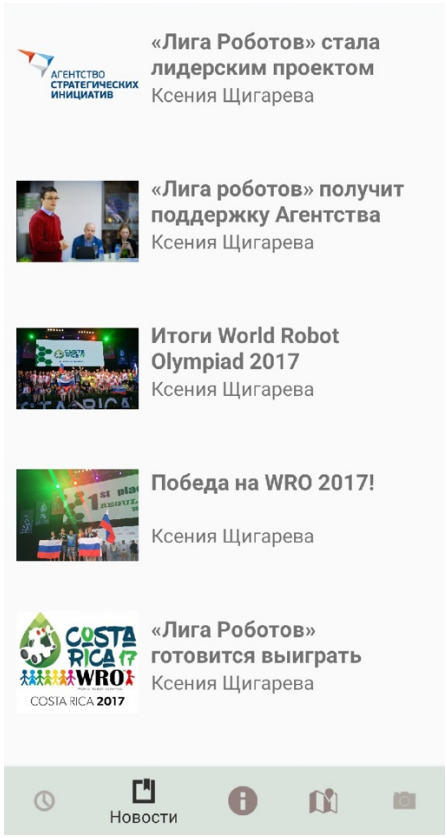

Рисунок 3.8 – Вкладка «Новости»

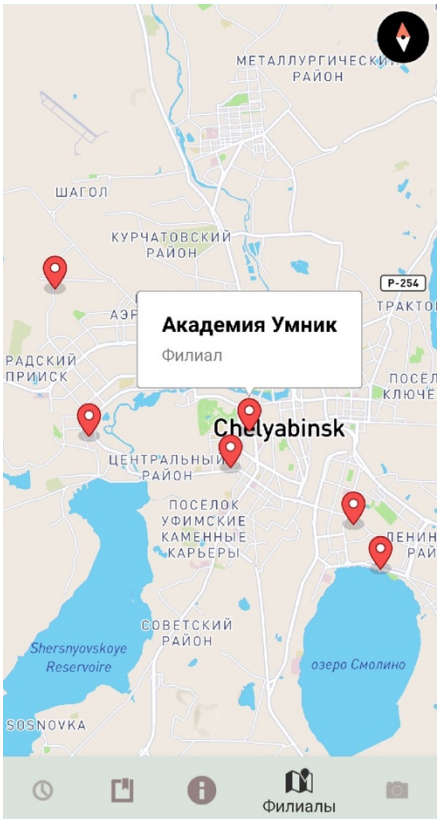

Рисунок 3.9 – Вкладка «Филиалы»

3.2 Разработка функциональности

Для того чтобы реализовать приложение для преподавателей сети секций робототехники необходимо выделить некоторые модули, которые будут отвечать за роботоспособность функций. Приложение использует принцип обработки активностей [7].

Для того чтобы реализовать приложение для преподавателей сети секций робототехники возникла необходимость создать API из которого приложение имело бы возможность загружать необходимую информацию. Все взаимодействия через API осуществляются по протоколу REST.

Использовались параметры API:

- hostname – клиенский индикатор, для предоставления доступа в систему
- branch – ID филиала,в который происходит запрос
- api_key – ключ для авторизации в системе
- token – токен, полученный при авторизации

3.2.1 Обработка главного окна

При запуске приложения открывается главное окно, на котором есть две вкладки «Личный кабинет» и «Информация». При выборе вкладки «Информация» загружается страница с общей информацией сети секций робототехники «Лига Роботов». Так же, на данной странице есть кнопка «Записаться в секцию», при нажатии на которую открывается WebView, в которой можно записать ребенка онлайн.

Для перехода в личный кабинет, необходимо пройти авторизацию по номеру телефона. Авторизация реализована посредством API и осуществляется по протоколу REST. Формат данных JSON, прочие форматы не поддерживаются. API реализуется в MVC контроллере, имеет вид CRUD парадигмы.

Перед тем, как обращаться к методам CRUD нужно пройти авторизацию и получить временный код доступа. Данным кодом должен подписываться каждый CRUD запрос загаловке X-ALFACRM-TOKEN. Время, действия кода – пятнадцать минут. После истечения пятнадцати минут код доступа необходимо запрашивать заново, так как при вводе старого кода будет выдаваться ошибка. При удачном прохождении этапа авторизации загружается личный кабинет преподавателя.

3.2.2 Обработка вкладки «Расписание»

При выборе вкладки «Расписание» загружается окно с изображениями всех филиалов, днем, когда запланировано занятие и время начала и конца занятия. Таким образом, преподаватель может подтвердить время и день своего занятия, либо найти свободного преподавателя для замены. Вкладка «Расписание» реализовано посредством API.

3.2.3 Обработка вкладки «Новости»

Новостная лента с сайта «Лига Роботов» отображается в приложении благодаря разметки rss. RSS – является форматом передачи веб-контента, который позволяет регулярно публиковать и транслировать содержимое, опубликованное каким-либо веб-узлом. Происходит конвертация в JSON, после чего приложение получает JSON.

3.2.4 Обработка вкладки «Главная»

При выборе вкладки «Главная» в приложении отображается личный профиль преподавателя с его фотографией и личным расписанием занятий. Во время летних каникул данное окно остается пустым, так как семестр заканчивается в мае и расписание становится неактуальным. Для следующего семестра расписание формируется заново. Карта города Челябинск загружена в приложение при помощи API mapbox.

3.2.5 Обработка вкладки «Филиалы»

При выборе вкладки «Филиалы» загружается карта города Челябинск с метками и адресами всех филиалов. Карта города Челябинск загружена в приложение при помощи API mapbox.

3.3 Тестирование программы

В процессе разработки приложения на каждом этапе разработки проводилось тестирование, для несоответсвия техническому заданию и выявлению программных ошибок. Для тестирования был использован встроенный эмулятор и физическое устройство на операционной системе Android. Для проверки корректности работы приложения оно было установлено на мобильное устройство, в процесе такого тестирования.

1. Каждая активность приложения была протестирована юнит-тестированием, для выявления ошибок которые вызваны нессответствием параметров полученных и ранее ожидаемых. На каждую активность были созданы специальные юнит-классы, проверяющие активность верными и неверными параметрами.
2. Приложение запускалось на различных устройствах, с различными версиями операционной системы Android, для выявления особенностей работы приложения на таких версиях операционной системы.
3. После окончания разработки, программный продукт успешно прошел тестирование, показывая стабильную работу.

3.4 Выводы по разделу

В данном разделе разработан прототип мобильного приложения, разработан интерфейс, описан функционал, а также обработка каждого раздела приложения.

ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты.

• Выполнен анализ аналогичных программных продуктов, выявлены их особенности, недостатки и достоинста, которые были учтены при разработке собственного мобильного приложения.

• Описана предметная область, приведена характеристика предприятия – заказчика программного продукта.

• Рассмотрены технологии реализации программного продукта. Произведен обзор современных операционных мобильных систем, по итогам которого была определена операционная система для разработки приложения (Android). Рассмотрены два современных языка программирования для операционной системы Android (Java и Kotlin), для разработки был выбран современный язык программирования Kotlin.

• Разработана архитектура системы мобильного приложения, созданы диаграммы активности, классов и вариантов использования.

• Приведено обоснование дальнейшей интеграции с системой АльфаCRM.

• Разработан прототип пользовательского интерфейса, а так же приведено описание функций программного продукта.

• Написана программная документация, включающая руководство пользователя, с инструкцией по использованию мобильного приложения.

В результате работы разработано программное обеспечение, которое состоит из клиентской части в виде мобильного приложения для операционной системы Android.

Таким образом, все поставленные задачи были выполнены успешно, на практике применены занания и умения, полученные в процессе обучения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Томашевский, Д. Смартфоны с Android для чайников / Д. Томашевский. – Москва: Диалектика, 2018 – 336 с.

2 МакГрат, М. Создание приложений на Android для начинающих / М, МакГрат. – Москва: Эксмо, 2016. – 192 с.

3 Филлипс, Б. Стюарт.К, Марсикано.К Android программирование для профессионалов / Б. Филлипс, К. Стюарт, К. Марсикано. – М.-СПб: Питер, 2017. – 688 с.

4 Жемеров, Д. Kotlin в действии / Д. Жемеров, С. Исакова. – Москва: ДМК Пресс, 2017. – 570 с.

5 Клифтон, Я. Проектирование пользовательского интерфейса в Android / Я. Клифтон. – Москва: ДМК Пресс, 2017. – 452 с.

6 Ераннсон.А Эффективное использование потоков в операционной системе Android. Технологии асинхронной обработки данных. / А. Ераннсон, – Москва: ДМК Пресс, 2017. – 304 с.

7 Нейл.Т Мобильная разработка. Галерея шаблонов / Т. Нейл, – Санкт-Петербург: Питер, 2013. – 326 с.

8 Разработка приложения для смартфонов под управлением операционной системы Android. – Дата обновления: 19.10.2018. URL: https://knowledge.allbest.ru/programming/3c0a65625b3ac79b5d53a89521206c27_0.html#text (дата обращения: 22.05.2019)

9 Магазин приложений от компании Google. – URL: https://play.google.com/ (дата обращения: 23.05.2019)

10 Bloch, J.J. Effective Java: Programming Language Guide / J.J Bloch, – M.: Лори, 2002. – 224 с.

11 Гослинг, Дж. The Java Language Specification, Java SE 8 Edition / Дж. Гослинг, Б. Джой, Г. Стил [и др.] – М.: «Вильямс», 2015. – 672 с.

12 Академия современного программирования. – URL: http://www.amse.ru/ (дата обращения: 26.05.2019).

13 Образовательный портал IT-сферы Geekbrains. – URL: https://geekbrains.ru/ (дата обращения: 23.04.2019).

14 Хорстманн, К. С. Java. Библиотека профессионала Том 1. Основы. / К.С. Хорстманн, Г. Корнелл. – М.: «Вильямс», 2008. – 864 с.

15 Веб-ресурс, посвещенный Android-разработке. – URL: https:// androidinsider.ru/. (дата обращения: 26.04.2019).

16 Сайт европейской группы веб-разработчиков Netguru. – URL: https:// www.netguru.co/. (дата обращения: 30.05.2019).

17 Буч, Г. Язык UML. Руководство пользователя. — 2-е изд./ Г. Буч, Дж. Рамбо, А. Джекобсон — М., СПб.: ДМК Пресс, Питер, 2004. — 432 с.

18 Учебник по разработке мобильных приложений для Android. – URL: http://startandroid.ru/. (дата обращения: 13.05.2019).

19 Сайт Александра Климова с уроками программирования. – URL: http://developer.alexanderklimov.ru/android/ (дата обращения: 12.11.2018).

20 Сайт для разработчиков Android-приложений. – URL: https://developer. android.com/. (дата обращения: 10.05.2019).

ПРИЛОЖЕНИЕ 1

ОПИСАНИЕ ПРОГРАММЫ

Мобильное приложение для преподавателей

секции «Лига роботов»

2019

ОГЛАВЛЕНИЕ

П1.1. ОБЩИЕ СВЕДЕНИЯ

Данное программное обеспечение является мобильными приложением для операционной системы Android. Написано в интегрированной среде разработки Android Studio 3.2.1.0 на языке программирования Kotlin.

П1.2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Клиент-серверное мобильное приложение для преподавателей федеральной сети секций робототехники «Лига Роботов» в городе Челябинск на операционной системе Android.

Возможности данного программного обеспечения.

1. Личный кабинет, через который преподаватель сможет просматривать информацию о своих группах, о каждом отдельно взятом обучающемся.

2. Просмотр на карте информации о филиалах сети секций.

3. Чтение актуальных новостей франшизы.

4. Просмотр собственного расписания занятий.

П1.3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

Данное мобильное приложение, как и другие Android-приложения, после компиляции упаковано в один исполняемый apk-файл, который содержит в себе все файлы проекта, такие как файл манифеста (AndroidManifest.xml), dex-файл, файл ресурсов (рисунок П1.1).

Файл манифеста содержит основную информацию о программе, которая необходима операционной системе Android. Dex-файл содержит скомпилированный код приложения. Файл ресурсов включает в себя xml-файлы, в которых находятся описания макетов всех окон приложения,

панели инструментов и контекстного меню, цвета, строковые константы, векторные изображения иконок, стили и т.д.

П1.4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Для функционирования программного продукта необходимо мобильное устройство, с установленной на нем операционной системой Android версии не ниже 4.0 «Ice Cream Sandwich».

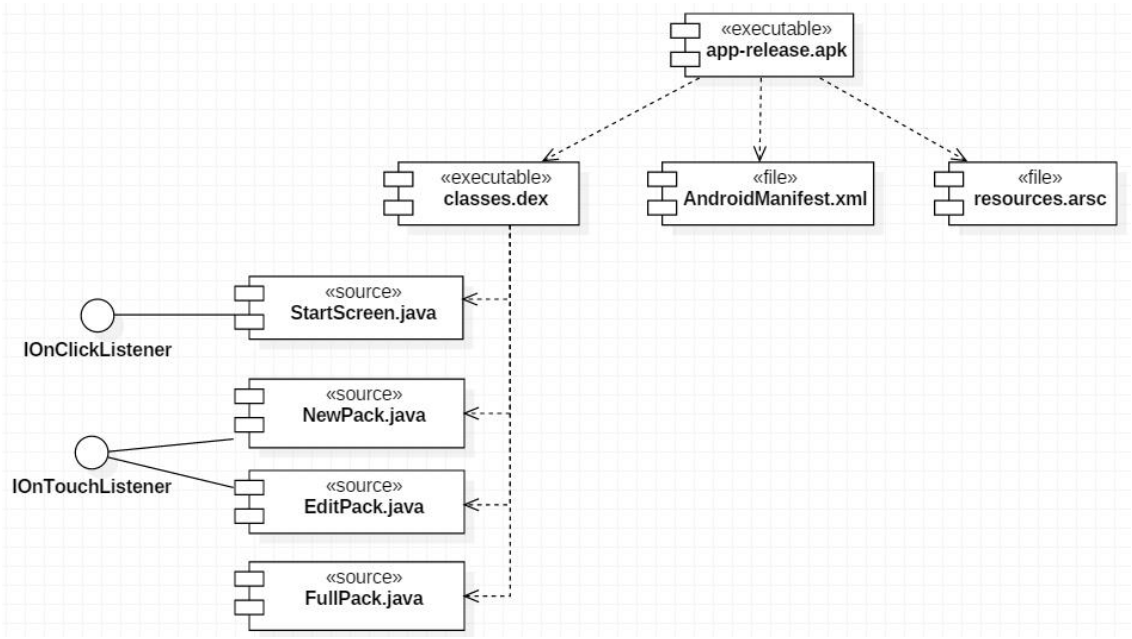

Рис. П1.1 – Диаграмма компонентов

П1.5. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входные данные: символьные строки, вводимые пользователем в специальные редактируемые поля, нажатие кнопок и другие касания пальцами пользователя по экрану его мобильного устройства.

Выходные данные: сохраненные данные и действия пользователя, представленные определенным образом на экране.

ПРИЛОЖЕНИЕ 2

ТЕКСТ ПРОГРАММЫ

2019

ОГЛАВЛЕНИЕ

MainActivity.kt

//Основная активность

```
package ru.ligarobotov.navigation
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.view.MenuItem
import android.view.View
import androidx.navigation.NavController
import androidx.navigation.Navigation
import kotlinx.android.synthetic.main.activity_main.*
import ru.ligarobotov.LRMobileApp
import ru.ligarobotov.R
import ru.ligarobotov.redux.Action
import ru.ligarobotov.state.AppState
import ru.ligarobotov.state.NAVIGATE_BACK
import ru.ligarobotov.state.NAVIGATE_TO
import ru.ligarobotov.state.NAVIGATE_TO_REPLACE
import kotlin.math.min
class MainActivity : AppCompatActivity() {
    private var navController: NavController? = null
    private var cacheNavigationStack: List<Int> = emptyList()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        bottom_navigation.setOnNavigationItemSelectedListener(::onTeacherMenuItemSelected)
        navController = Navigation.findNavController(this, R.id.navigation_host_fragment)
        getStore().subscribe(this, AppState::navigationStack) { activity, navigationStack ->
            if (navigationStack.isEmpty()) {
                cacheNavigationStack = navigationStack
                if (navController?.currentDestination != null && navController?.currentDestination?.id != 0) {
                    activity.getStore().dispatch(Action(NAVIGATE_TO, navController?.currentDestination?.id))
                } else {
                    activity.getStore().dispatch(Action(NAVIGATE_TO, R.id.startFragment))
                }
            } else if (navigationStack.size == 1 && navigationStack[0] == navController?.currentDestination?.id) {
                cacheNavigationStack = navigationStack
            } else {
                var matchFragmentsCount = min(navigationStack.size, cacheNavigationStack.size)
                for ((i, targetId) in cacheNavigationStack.withIndex()) {
                    if (i >= navigationStack.size) {
                        navController?.popBackStack(targetId, true)
```

```
            break
        } else if (targetId != navigationStack[i]) {
                navController?.popBackStack(targetId, true)
                matchFragmentsCount = i
                break
            }
        }
        for (i in (matchFragmentsCount until navigationStack.size)) {
            navController?.navigate(navigationStack[i])
        }
        cacheNavigationStack = navigationStack
      }
      toggleNavigationBarIfNeeded(cacheNavigationStack)
    }
  }
  override fun onDestroy() {
    getStore().unsubscribe(this)
    navController = null
    cacheNavigationStack = emptyList()
    super.onDestroy()
  }
  override fun onBackPressed() {
    if (cacheNavigationStack.size > 1) {
      getStore().dispatch(Action(NAVIGATE_BACK))
    } else {
      super.onBackPressed()
    }
  }
  private fun getStore() = (applicationContext as LRMobileApp).store
  private fun toggleNavigationBarIfNeeded(navigationStack: List<Int>) {
    if (navigationStack.isEmpty()) {
      bottom_navigation.visibility = View.GONE
      return
    }
    bottom_navigation.visibility = View.VISIBLE
    when (navigationStack.last()) {
      R.id.scheduleFragment -> bottom_navigation.selectedItemId = R.id.scheduleItem
      R.id.mainTeacherFragment -> bottom_navigation.selectedItemId = R.id.mainTeacherItem
      R.id.newsFragment -> bottom_navigation.selectedItemId = R.id.newsItem
      R.id.filialsFragment -> bottom_navigation.selectedItemId = R.id.filialsItem
      R.id.photoFragment -> bottom_navigation.selectedItemId = R.id.photoItem
      else -> bottom_navigation.visibility = View.GONE
    }
  }
  private fun onTeacherMenuItemSelected(menuItem: MenuItem): Boolean {
    when (menuItem.itemId) {
      R.id.scheduleItem -> getStore().dispatch(Action(NAVIGATE_TO_REPLACE, R.id.scheduleFragment))
      R.id.mainTeacherItem -> getStore().dispatch(Action(NAVIGATE_TO_REPLACE, R.id.mainTeacherFragment))
      R.id.newsItem -> getStore().dispatch(Action(NAVIGATE_TO_REPLACE, R.id.newsFragment))
      R.id.filialsItem -> getStore().dispatch(Action(NAVIGATE_TO_REPLACE, R.id.filialsFragment))
      R.id.photoItem -> getStore().dispatch(Action(NAVIGATE_TO_REPLACE, R.id.photoFragment))
    }
    return true
  }
}
```

StartFragment.kt

//Стартовый фрагмент. Где пользователь вводит телефон

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import kotlinx.android.synthetic.main.fragment_start.*
import ru.ligarobotov.R
import ru.ligarobotov.navigation.BaseFragment
import ru.ligarobotov.redux.Action
import ru.ligarobotov.state.NAVIGATE_TO
lateinit var phone: String

class StartFragment : BaseFragment() {
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        return inflater.inflate(ru.ligarobotov.R.layout.fragment_start, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        PhoneItem.addTextChangedListener(object :TextWatcher{
            override fun afterTextChanged(s: Editable?) {
                Log.d("TAG", "After s = ${s}")
            }
            override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {
                Log.d("TAG", "Before s = ${s}")
            }
            override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
                Log.d("TAG", "onText s = ${s}")
            }
        })
        btnInfo.setOnClickListener{
            getStore().dispatch(Action(NAVIGATE_TO, R.id.infoFragment))
        }
        btnLogin.setOnClickListener{
            phone = PhoneItem.text.toString()
            if (phone == "89193442869") {
                Log.d("TAG", "ALEX UREV")
                getStore().dispatch(Action(NAVIGATE_TO, R.id.otpConfirmFragment))
            }
        }
    }
}
```

OtpFragment.kt

//Фрагмент ввода кода подтверждения

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import kotlinx.android.synthetic.main.fragment_otp_confirm.*
import ru.ligarobotov.R
import ru.ligarobotov.navigation.BaseFragment
import ru.ligarobotov.redux.Action
import ru.ligarobotov.state.NAVIGATE_TO
class OtpConfirmFragment : BaseFragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_otp_confirm, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        btnFurther.setOnClickListener{
            val Otp = OtpItem.text.toString()

            if(Otp == "0000") {
                getStore().dispatch(Action(NAVIGATE_TO, R.id.mainTeacherFragment))
            }
        }
    }
}
```

BaseFragment.kt

//Основная базовый фрагмент

```
package ru.ligarobotov.navigation
import android.support.v4.app.Fragment
import ru.ligarobotov.LRMobileApp
import ru.ligarobotov.state.AppState
open class BaseFragment : Fragment() {
    override fun onDestroyView() {
        super.onDestroyView()
        getStore().unsubscribe(this)
    }
    protected fun <M : Any> subscribe(mapper: (AppState) -> M,
                                      subscriber: (BaseFragment, M) -> Unit) = getStore().subscribe(this, mapper, subscriber)
    protected fun getStore() = (activity!!.applicationContext as LRMobileApp).store
}
```

InfoFragment.kt

//Основная фрагмент с информацией об организации

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import kotlinx.android.synthetic.main.fragment_info.*
import ru.ligarobotov.R
import ru.ligarobotov.navigation.BaseFragment
import ru.ligarobotov.redux.Action
import ru.ligarobotov.state.NAVIGATE_TO
class InfoFragment : BaseFragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_info, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        btnEnroll.setOnClickListener{
            getStore().dispatch(Action(NAVIGATE_TO, R.id.enrollFragment))
        }
    }
}
```

EnrollFragment.kt

//WebView записи ребенка в секцию

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import kotlinx.android.synthetic.main.fragment_enroll.*
import ru.ligarobotov.navigation.BaseFragment
class EnrollFragment: BaseFragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_enroll, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        WebViewEnroll.loadUrl("https://chel.ligarobotov.ru/sekcii/#reg")
    }
}
```

ScheduleFragment.kt

//Расписание преподавателя

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.google.gson.Gson
import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.disposables.Disposable
import io.reactivex.schedulers.Schedulers
import ru.ligarobotov.R
import ru.ligarobotov.adapter.*
import ru.ligarobotov.navigation.BaseFragment
var requestSchedule : Disposable? = null
lateinit var vRecViewSchedule: RecyclerView
class ScheduleFragment: BaseFragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_schedule, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        vRecViewSchedule = view.findViewById(R.id.schedule_list)
        val o = createRequest("https://api.rss2json.com/v1/api.json?rss_url=https%3A%2F%2Frobochel.do.am%2Fpubl%2F1%2Frss")
            .map { Gson().fromJson(it, Delta::class.java) }.subscribeOn(
                Schedulers.io()).observeOn(AndroidSchedulers.mainThread())
        requestSchedule = o.subscribe({
            for (item in it.items) {
                Log.w("test", "title: ${item.author}")
            }
            showRecView(it.items)
        }, {
            Log.e("test", "", it)
        })
    }
    private fun showRecView(deltaList: ArrayList<DeltaItem>){
        vRecViewSchedule.adapter = RecAdapterSchedule(deltaList)
        vRecViewSchedule.layoutManager = LinearLayoutManager(context)
    }
```

NewsFragment.kt

//Фрагмент новостей

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.google.gson.Gson
import io.reactivex.android.schedulers.AndroidSchedulers
import ru.ligarobotov.adapter.Feed
import ru.ligarobotov.adapter.FeedItem
import ru.ligarobotov.adapter.RecAdapter
import ru.ligarobotov.adapter.createRequest
import ru.ligarobotov.navigation.BaseFragment
var request : Disposable? = null
lateinit var vRecViewNews: RecyclerView
class NewsFragment: BaseFragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_news, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        vRecViewNews = view.findViewById(R.id.act_list)
        val o = createRequest("https://api.rss2json.com/v1/api.json?rss_url=https%3A%2F%2Fligarobotov.ru%2Fposts-
common%2Ffeed%2F")
            .map { Gson().fromJson(it, Feed::class.java) }.subscribeOn(
                Schedulers.io()).observeOn(AndroidSchedulers.mainThread())
        request = o.subscribe({
            for (item in it.items) {
                Log.w("test", "title: ${item.title}")
            }
            showRecView(it.items)
        }, {
            Log.e("test", "", it)
        })
    }
    private fun showRecView(feedList: ArrayList<FeedItem>){
        vRecViewNews.adapter = RecAdapter(feedList)
        vRecViewNews.layoutManager = LinearLayoutManager(context)
    }}
```

MainTeacherFragment.kt

//Личный кабинет преподавателя

```
package ru.ligarobotov.navigation.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import kotlinx.android.synthetic.main.fragment_main_teacher.*
import ru.ligarobotov.R
import ru.ligarobotov.navigation.BaseFragment

class MainTeacherFragment : BaseFragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_main_teacher, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        if(phone == "89193442869") TextLogin.text = "Alex Urev”
    }
}
```

Utils.kt

//Создание запросов по url

```
package ru.ligarobotov.adapter
import io.reactivex.Observable
import java.net.HttpURLConnection
import java.net.URL

fun createRequest(url: String) = Observable.create<String>{
    val urlConnection = URL(url).openConnection() as HttpURLConnection
    try {
        urlConnection.connect()

        if (urlConnection.responseCode != HttpURLConnection.HTTP_OK)
            it.onError(RuntimeException(urlConnection.responseMessage))
        else {
            val str = urlConnection.inputStream.bufferedReader().readText()
            it.onNext(str)
        }
    } finally {
        urlConnection.disconnect()
    }
}
```

NewsAdapter.kt

//адаптер для подгрузки новостей

```
package ru.ligarobotov.adapter

import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import com.squareup.picasso.Picasso
import ru.ligarobotov.R

class RecAdapter(val items: ArrayList<FeedItem>): RecyclerView.Adapter<RecHolder>(){
    override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecHolder {
        val inflater = LayoutInflater.from(p0.context)
        val view = inflater.inflate(R.layout.list_item, p0, false)
        return RecHolder(view)
    }
    override fun getItemCount(): Int {
        return items.size
    }
    override fun onBindViewHolder(p0: RecHolder, p1: Int) {
        val item = items[p1]
        p0.bind(item)
    }
}
class RecHolder(view: View) : RecyclerView.ViewHolder(view){

    fun bind(item: FeedItem){
        val vTitle = itemView.findViewById<TextView>(R.id.item_title)
        val vAuthor = itemView.findViewById<TextView>(R.id.item_desc)
        val vThumb = itemView.findViewById<ImageView>(R.id.item_thumb)
        vTitle.text = item.title
        vAuthor.text = item.author
         Picasso.with(vThumb.context).load(item.thumbnail).into(vThumb)
    }
}
class Feed(
    val items: ArrayList<FeedItem>
)
class FeedItem(
    val title: String,
    val link: String,
    val thumbnail: String,
    val description: String,
    val author: String
)
```

ScheduleAdapter.kt

//адаптер для подгрузки новостей

```
package ru.ligarobotov.adapter
import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import com.squareup.picasso.Picasso
import ru.ligarobotov.R
class RecAdapterSchedule(val items: ArrayList<DeltaItem>): RecyclerView.Adapter<RecHolderSchedule>(){
    override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecHolderSchedule {
        val inflater = LayoutInflater.from(p0.context)
        val view = inflater.inflate(R.layout.list_item, p0, false)
        return RecHolderSchedule(view)
    }
    override fun getItemCount(): Int {
        return items.size
    }
    override fun onBindViewHolder(p0: RecHolderSchedule, p1: Int) {
        val item = items[p1]
            p0.bind(item)
    }
}
class RecHolderSchedule(view: View) : RecyclerView.ViewHolder(view){
    fun bind(item: DeltaItem){
        val vTitle = itemView.findViewById<TextView>(R.id.item_title)
        val vDesc = itemView.findViewById<TextView>(R.id.item_desc)

        val vImg = itemView.findViewById<ImageView>(R.id.item_thumb)

        vTitle.text = item.title
        vDesc.text = item.content
        }
    }
}
class Delta(
    val items: ArrayList<DeltaItem>
)
class DeltaItem(
    val title: String,
    val description: String,
    val content: String,
    val author: String
)
```