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 «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2019 г.

Разработка мобильного приложения для корпоративной информационно-аналитической системы Универис

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____И.Л. Кафтанников
«___»_____2019 г.

Автор работы,
студент группы КЭ-222
_____Г.В. Севостьянов
«___»_____2019 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____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 «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2019 г.

ЗАДАНИЕ
на магистерскую диссертацию
студенту группы КЭ-222
Севостьянова Глеба Валерьевича
обучающемуся по направлению
09.04.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка мобильного приложения для корпоративной информационно-аналитической системы Универис» утверждена приказом по университету от 25 апреля 2019 г. № 899

2. **Срок сдачи студентом законченной работы:** 21 июня 2019 г.

3. **Исходные данные к работе:** статьи, книги, техническое задание.
   - Тепляков, С., Паттерны проектирования на платформе .NET. – СПб.: Питер, 2015. – 320 с.;
   - Жемеров, Д., Kotlin в действии. / пер. с англ. Киселев А. Н. – М.: ДМК Пресс, 2018. – 402 с.: ил.;

- Усов, В. А., Swift. Основы разработки приложений под iOS и macOS. 4-е изд., доп. и перераб. – СПб.: Питер, 2018 – 448 с.: ил. – (Серия «Библиотека программиста»);
- сайт КИАС Универис (http://www.univeris.susu.ru)

4. **Перечень подлежащих разработке вопросов:**
   - изучить опыт разработок в области создания мобильных приложений для информационных систем вуза;
   - анализ современных программных технологий для создания мобильных приложений;
   - разработка собственного мобильного приложения для КИАС Универис;
   - оценка эффективности разработанного мобильного приложения.

5. **Дата выдачи задания:** 1 декабря 2018 г.

Руководитель работы _____/*И.Л. Кафтанников*/

Студент _____/*Г.В. Севостьянов*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2019	
Разработка модели, проектирование	01.04.2019	
Реализация системы	10.04.2019	
Тестирование, отладка, эксперименты	15.05.2019	
Компоновка текста работы и сдача на нормоконтроль	18.06.2019	
Подготовка презентации и доклада	19.06.2019	

Руководитель работы _____/*И.Л. Кафтанников*/

Студент _____/*Г.В. Севостьянов*/

Аннотация

Г.В. Севостьянов. Разработка мобильного приложения для корпоративной информационно-аналитической системы Универис. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 93 с., 25 ил., библиогр. список – 48 наим.

В рамках магистерской диссертации производится детальный анализ современных технологий разработки мобильных приложений для информационно-аналитических систем вузов. Организуется разработка мобильного приложения для корпоративной информационно-аналитической системы Универис. Производится выборка и анализ результатов работы системы в домене специально разработанных задач. Рассматриваются преимущества и недостатки, технологий разработки мобильных приложений. Строится архитектура мобильного приложения. Производится анализ эффективности приложения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В работе рассматривается тема разработки мобильного приложения для корпоративной информационно-аналитической системы под названием Универис[1], эта система организовывает управление бизнес-процессов Южно-Уральского государственного университета (Национального исследовательского университета)[2], в том числе предоставляет студентам и преподавателям информацию об учебном процессе (расписание, учебные планы и т. д.).

Актуальность темы, связанной с разработкой мобильного приложения для КИАС Универис заключается в том, что на сегодняшний день, когда цифровые технологии становятся все быстрее и мобильнее, быстрое получение доступа до нужной информации становится основной проблемой для пользователей. Так, мощная информационная система Универис – это в первую очередь сайт, сайт не позволяет быстро найти, например, расписание занятий, скажем где-нибудь на улице или в коридорах университета, ведь для этого нужно открыть браузер на смартфоне, найти нужную ссылку, авторизоваться и только потом найти нужный раздел с расписанием. Сразу же появляется задача ускорить этот процесс и сделать удобнее, мобильное приложение здесь подходит как нельзя лучше.

Кроме того, система Универис постоянно развивается и переходя в мобильную разработку для него открывается новые возможности в развитии, ведь мобильные устройства зачастую оснащены различными мультимедийными технологиями, начиная от видеокамеры и заканчивая устройством GPS. А значит такие задачи, как создание интерактивной карты для навигации по корпусам университета, уже могут быть реализованы.

Не будем забывать о том, что в отличие от веб-интерфейса, мобильное устройство позволяет сохранять информацию на физическом носителе, например, расписание занятий для конкретного пользователя можно сохранить в локальную базу данных и оно будет доступно ему без дальнейшего подключения к сети интернет.

Из вышесказанного следует, что мобильное приложения для Универиса – это не только удобное решение, но и необходимое в современных реалиях. Забегая вперед, в главе 1 будет приведен статистический анализ использования мобильных приложений в России, что только подтвердит актуальность задачи.

Цель, представленной выпускной квалификационной работы – разработать мобильное приложение для КИАС Универис и адаптировать в нем разделы информационной системы в удобном виде для студентов и сотрудников.

Для достижения цели были выполнены следующие задачи:

1. Изучить опыт разработок в области создания мобильных приложений для информационных систем вузов.
2. Проанализировать технологии, которые будут использоваться при разработке приложения, определить их преимущества и недостатки.
3. На основе полученных данных, разработать и реализовать мобильное приложение для КИАС Универис.
4. Проанализировать полученные результаты и определить эффективность работы приложения.

Для решения поставленных задач были просмотрены и оценены различные аналоги приложения в самых популярных магазинах платформ Android[3] и iOS[4], на основе опыта других разработок и потребностей пользователей Универиса были сформулированы требования к

разрабатываемому приложению. Исходя из требований были сформулированы проблемы и особенности приложения, подробнее об этом в главах 3 и 4.

В главе 1 рассказывается о предметной области разработки мобильных приложений, производится обзор существующих решений, для каждого аналога будут сформулированы достоинства и недостатки, этот опыт будет позже использоваться в реализации и тестирование мобильного аналога КИАС Универис. В этой же главе рассматриваются различные технологии создания приложений на мобильные устройства, подбираются варианты, которые будут использоваться в проектирование.

Из полученных данных в главе 1 будут сформулированы требования к мобильному приложению для Универис и детально описаны в главе 2, там же буде представлена диаграмма прецедентов.

Третья глава работы посвящена проектированию приложения. В ней будет описана архитектура, а также будет представлена структура базы данных приложения.

В глава 4 будет подробно рассказано о реализации приложения, будут представлены скриншоты приложения.

Глава 5 посвящена тестированию и определению эффективности работы мобильного приложения.

Источником информации были книги по разработке мобильных приложений на различных платформах, а также были использованы паттерны проектирования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Прежде чем говорить о разработки мобильного приложения, определим, что же представляет из себя корпоративная информационно-аналитическая система Универис.

Южно-Уральский государственный университет (Национальный исследовательский университет) это огромный вуз:

- 14 высших школ и институтов;
- 4 филиала;
- 114 кафедр;
- более 50000 студентов;
- более 5000 сотрудников;
- более 5000 компьютеров;
- множество внутренних подразделений.

Процесс управления таким вузом должен быть адекватен текущей ситуации в обществе, государстве, мире.

В то же время – удобен, ясен и эффективен для студентов, преподавателей и сотрудников университета.

Без введения комплексного подхода к организации управления университетом, без использования новейших информационных технологий и разработок в этой сфере, оптимизирующих и упрощающих управленческий процесс, образовательная система во многом отстает от требований времени и ожиданий общества. Современный университет использует современные стандарты управления, основанные на информационных технологиях.

Проблемы управления вузом:

- множество разнопрофильных подразделений;

– использование разных специализированных решений для отдельных подразделений;
– динамично изменяющиеся условия;
– отсутствие однородной информации для анализа и управления.

Цель создания такой системы – обеспечить руководство университета и его подразделения эффективным инструментарием информационной поддержки формирования, контроля и реализации государственной политики в сфере образования.

Начиная с 2002 г. по приказу ректора ВЦ приступает к разработке корпоративной информационно-аналитической (КИАС) «Универис» на современной технологической и программной платформе с удобным пользовательским интерфейсом. Создаются модули для автоматизации работы учебно-методического управления, отдела кадров сотрудников и студентов, мобилизационного управления, студенческого кампуса, а также комплексная система приема и зачисления абитуриентов. С 2006 г. реализованы публикация рейтинга поступающих абитуриентов на сайте университета и sms-уведомления о ходе проверки документов и зачислении абитуриентов. В 2012 г. запущен Личный кабинет сотрудника. За шесть лет функциональность и содержание личного кабинета значительно расширились. Сейчас в нём есть возможность заполнять информацию о своих публикациях, личных достижениях, разрабатываемых учебных программах и пр. В 2012 г. также разработан Личный кабинет студента, с помощью которого студенты могут просматривать свои учебные планы, заполнять информацию о личных достижениях.[5]

Задачи проекта Универис:

– повышение эффективности деятельности университета;
– повышение качества подготовки специалистов;

– повышение оперативности управления, планирования и использования ресурсов университета;
– организация эффективного взаимодействия подразделений университета с Федеральным агентством по образованию, статистическими налоговыми и другими организациями и предприятиями.

В основе работы системы – единая база данных. Система управления базой данных – FIREBIRD[6]. Система работает под управлением операционной системы LINUX[7]. Среда разработчика программных приложений – DELPHI[8].

Наличие единой базы данных позволяет:

– выполнять мгновенный доступ ко всей необходимой информации, в том числе полученной из различных источников, с необходимостью ее ввода всего один раз
– обеспечивает высокое качество информации по всем бизнес-процессам вуза
– позволяет предоставлять информацию в нужное время, в нужном месте и в виде, наиболее удобном для принятия решений.

На рисунке 1 показана схема взаимодействия подразделений.

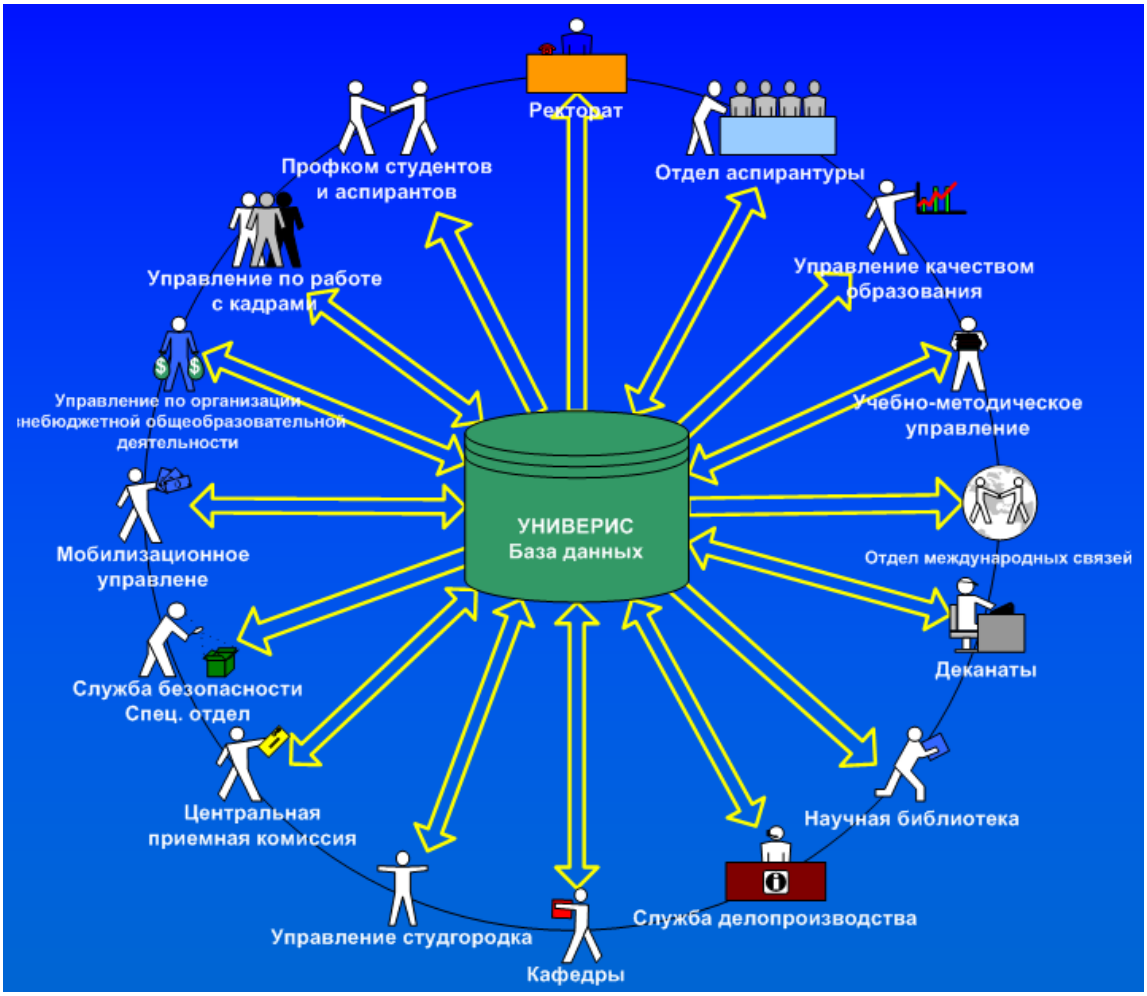

Рисунок 1 – Взаимодействие подразделений университета

Теперь вернемся к теме данной работы.

Опираясь на источник [9], за последние годы все больше людей стали пользоваться смартфонами и мобильным интернетом. Аналитическое агентство We Are Social и крупнейшая SMM-платформа Hootsuite совместно подготовили пакет отчетов о глобальном цифровом рынке Global Digital 2018. По представленным в отчетах данным, сегодня во всем мире интернетом пользуется более 4 миллиардов человек.

Больше половины населения земного шара теперь онлайн, и около четверти миллиарда из них вышли в сеть впервые в 2017 году. Самые высокие

темпы роста наблюдаются в Африке — количество пользователей интернета на континенте увеличилось больше чем на 20% по сравнению с аналогичным периодом прошлого года.

Одними из ключевых факторов роста интернет-аудитории в этом году стали доступные смартфоны и недорогие тарифы на мобильный интернет. В 2017 году более 200 миллионов человек впервые стали владельцами мобильных устройств, и теперь две трети из 7,6 млрд мирового населения имеют мобильный телефон.

Более половины из используемых сегодня мобильных устройств относятся к классу «умных», поэтому людям становится все проще получить доступ ко всем возможностям, которые предлагает интернет, где бы они ни находились.

Российский цифровой рынок в 2018 году также вторит глобальным трендам. Мобильным интернетом в России активно пользуются 91,4 млн человек. Доля трафика со смартфонов составляет 21%, что почти на треть больше прошлогоднего показателя, а пользователи планшетов стали немного реже выходить с них в сеть (-9%). Трафик с ноутбуков и ПК сократился на 5% (рисунок 2).

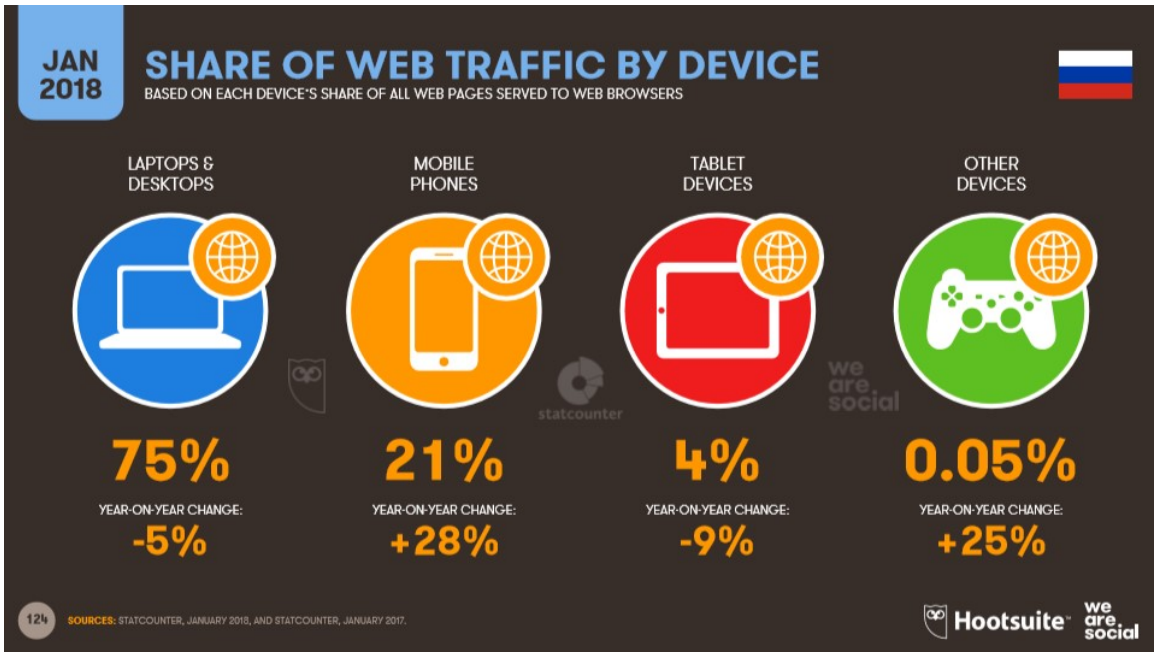

Рисунок 2 – Доля веб-трафика на устройстве

Наблюдается трансформация онлайн-потребления: интернет-пользователи становятся мобильнее, настольные устройства планомерно заменяются более удобными устройствами, которые можно носить с собой. Вследствие этого покупки плавно перетекают в онлайн, веб утрачивает позиции, уступая часть трафика приложениям, а социальные сети играют более значимую роль – это ценные сведения для бизнеса.

Как было сказано ранее, разработка мобильного приложения для КИАС Универис – актуальная задача. Но прежде чем заниматься составлением списка требований к приложению, найдем существующие аналогичные решения среди уже созданных мобильных приложений для информационно-аналитических систем (далее ИАС) вузов.

1.1. ОБЗОР АНАЛОГОВ

Поиск аналогов был проведен в Google Play. Google Play[3] (прежнее название — Android Market) — магазин приложений, игр, книг, музыки и фильмов компании Google и других компаний, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения.

На данную тематику в магазине находится очень много приложений разных производителей и разного качества, были выбраны самые популярные приложения различных вузов. Из всех подобранных аналогов была сформирована таблица 1.

Таблица 1 – Обзор существующих решений

Приложение	**Разработчик**	**Доступность**	**Оценка**	**Особенности**
Расписание Сибстрин[10]	Старшекурсники	Google Play, Apple App Store	4,8	Настройка цветовой темы
Studify – расписание вузов[11]	Studify and Talks	Google Play, Apple App Store	4,5	Поддерживает расписание множества вузов
СтудЖурнал – Расписание занятий[12]	Roman Sytnyk	Google Play	4,4	Быстрый импорт и экспорт данных в виде архива (бэкап), который сразу можно отправить друзьям или в личные хранилища
Skedy – расписание занятий[13]	Эльвин Аскяров	Google Play, Apple App Store	4,3	Таймер до звонков

Приложение	Разработчик	Доступность	Оценка	Особенности
Университет ИТМО[14]	Университет ИТМО. ДИТ	Google Play	3,5	Новости и мероприятия университета; Стипендия и зарплата; Информация о сотрудниках и подразделениях университета; Таймлайн с расписанием занятий и персональными новостями
Студент СФУ[15]	Глеб Мандров	Google Play	3,1	Срочные уведомления от преподавателей

Как видно из таблицы, приложения имеют разную степень доступности (на двух платформах или только на Android). Также можно заметить, что разработчиками может быть либо команда или фирма, либо физическое лицо. Каждое приложение имеет свои особенности, об этом подробнее ниже.

В нашем обзоре мы не будем рассказывать про все найденные аналоги, так как решения с низкими оценками нас мало интересует, однако такие приложения также имеют место быть, так например, мобильное приложение Университет ИТМО имеет очень полезную функцию, которая называется Таймлайн[16], что означает визуальное представление каких-либо событий, явлений, лиц или предметов в хронологическом порядке.

Детально рассмотрим приложения с самыми высокими оценками.

Мобильное приложение Расписание Сибстрин – это коммерческое приложение, разработанное командой Старшекурсники, как видно из таблицы,

оно доступно на всех платформах, у него достаточно высокие оценки. Главная особенность этого приложения – это то, что оно позволяет меня цвет оформления интерфейса, в остальном типичное приложения для расписания. Команда разработчиков не стала делать универсальное приложение для всех университетов, но они сделали шаблон, который адаптируют под разные вузы. Интерфейс приложения показан на рисунке 3.

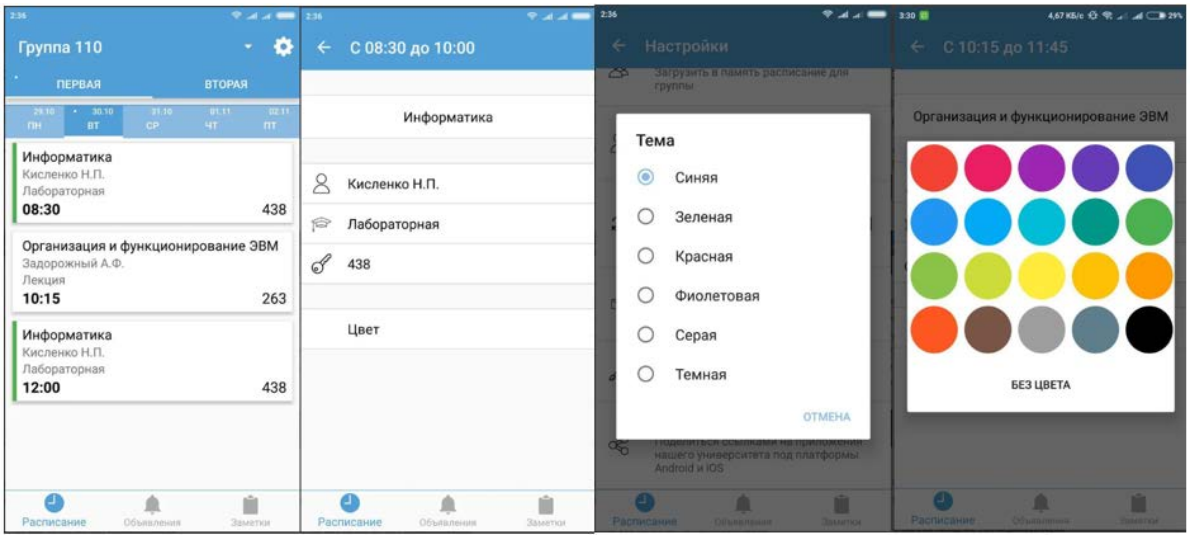

Рисунок 3 – Интерфейс «Расписание Сибстрин»

Достоинствами данного приложения являются:

- загружать в память расписания для студента или преподавателя:
- загружать в память несколько расписаний;
- просматривать расписания без доступа к интернету;
- просматривать даты в расписании;
- автоматически обновлять расписание;
- настраивать цветовую тему приложения из 6-ти предложенных;
- выбирать цвет для определенного предмета;
- связываться с администрацией приложения через "Сообщения" в приложении;
- писать заметки, которые связаны с учебой;

– включать или отключать оповещения об изменении в выбранном расписании.

Недостатками приложения являются:

– отсутствие расписания сессии;
– отсутствует расписание преподавателей;
– встроенная реклама (приложение коммерческое);
– малоинформативное расписание занятий (отсутствует тип занятия);
– расписание нельзя редактировать;
– приложение позиционирует себя как просмотр расписания, а не как ИАС.

Следующим аналогом является Studify – расписание вузов. В отличие от предыдущего решения, Studify – расписание вузов, представляет собой сборник расписаний различных высших учебных заведений, которые сотрудничаю с командой разработчиков, Studify and Talks. Начать сотрудничество не сложно, для этого есть специальная форма на их официальном сайте [11]. Приложение также доступно на всех платформах и имеет высокую среднюю оценку. Интерфейс представлен на рисунке 4.

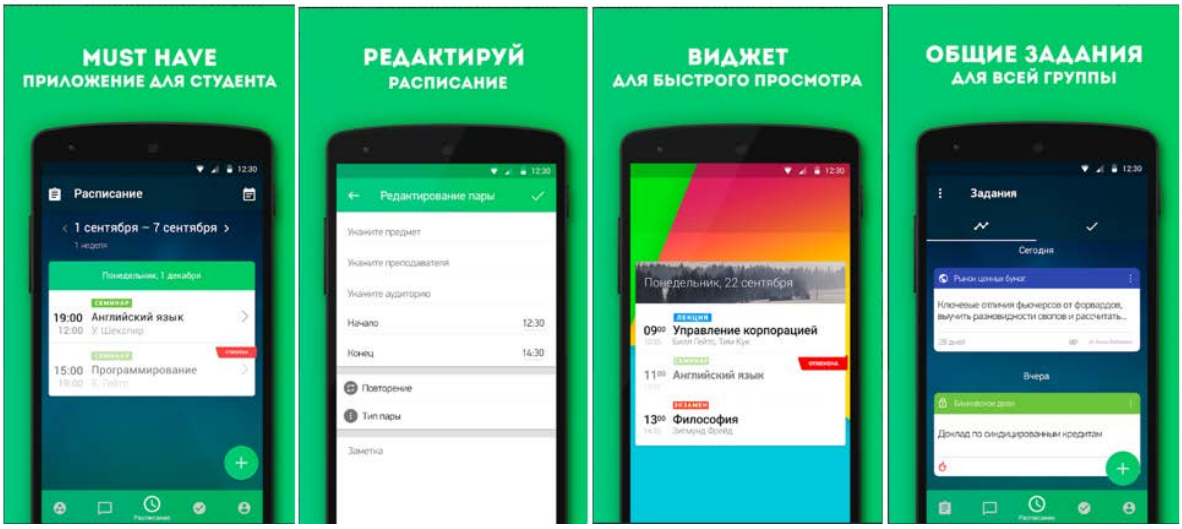

Рисунок 4 – Интерфейс «Studify – расписание для вузов»

Собственно, кроме универсальности для разных вузов, достоинства приложения не заканчиваются:

- более 440 университетов и институтов, более 35 000 групп с расписанием;
- актуальное расписание группы всегда доступно без подключения интернет;
- виджет;
- совершенное отображение расписания:
  - нумерация недель;
  - календарь;
  - понедельный просмотр расписания;
  - карточка деталей пары;
  - маркировка типов пар.
- управление расписанием – возможно делать изменения прямо в приложении;
- расписание преподавателей;
- журнал посещаемости группы;
- возможность записывать задания по предметам;
- установка сроков выполнения заданий.

Недостатки:

- приложение не поддерживает расписание сессии
- приложение ограниченно, поддерживает только расписание занятий, о ИАС и речи не идет
- приложение невозможно адаптировать под КИАС Универис
- платное предоставление расписания

1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.2.1 Анализ технологий разработки мобильных приложений

Информация взята из источника [17]. Рынку мобильных приложений уже больше десяти лет, однако он до сих пор бурно развивается. Спрос на создание мобильных приложений со стороны компаний постоянно растёт, и он всё ещё заметно превышает предложение, что приводит к постоянному удорожанию разработки. Одно из решений в удешевлении этого процесса — кроссплатформенная разработка, когда один и тот же код используется на всех платформах.

Естественные

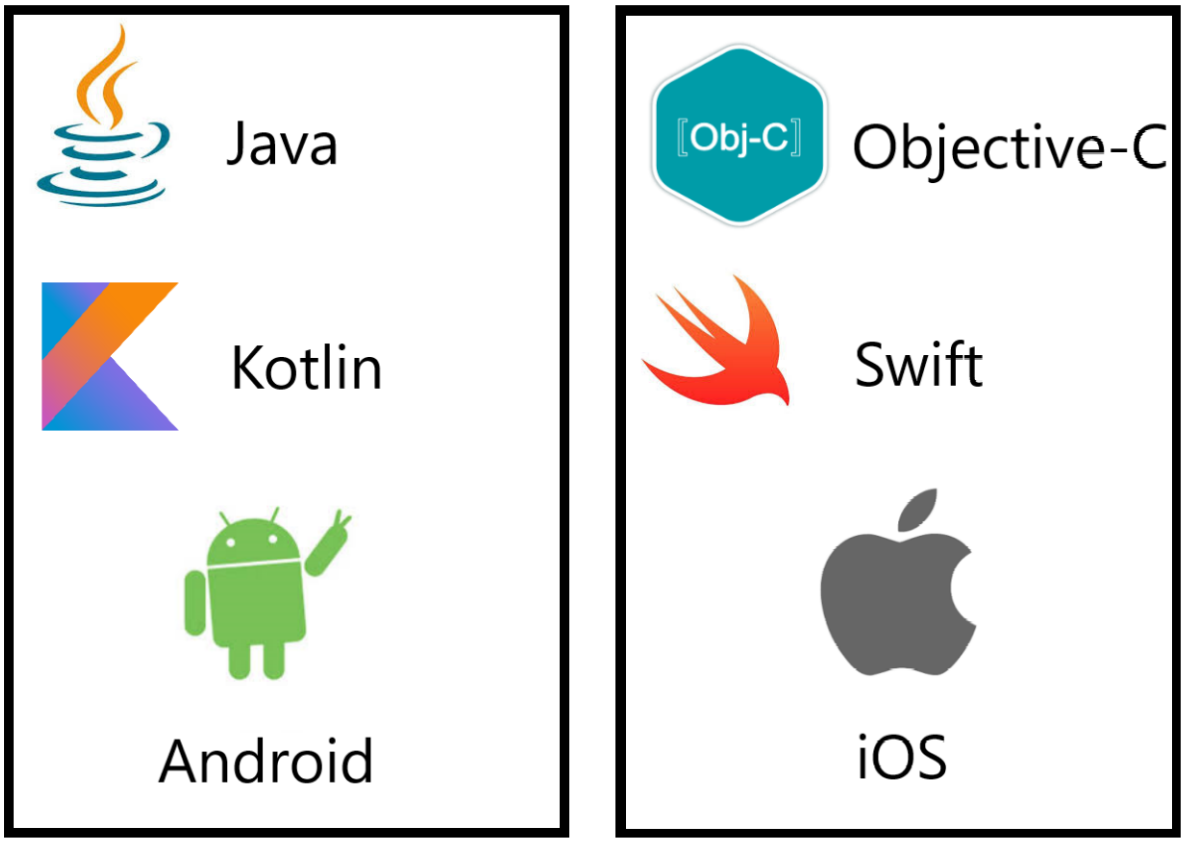

Рисунок 5 – Естественные языки

Если разработчики в процессе написания приложения пользуются принятым для конкретной платформы языком программирования, будь то Objective-C[18] и Swift[19] для iOS или Java[20] или Kotlin[21] для Android, такое приложение будет называться нативным (от англ. native — родной, естественный)(рисунок 5).

Преимущества нативных приложений:

- скорость работы и отклика интерфейса. Приложение реагирует на нажатия мгновенно, практически отсутствуют задержки в анимации, скроллировании, получении и выводе данных;
- понятный и простой доступ к функциям и датчикам устройства. Для разработчика не представляет проблемы работа с геолокацией, пуш-уведомлениями, съёмкой фото и видео через камеру, звуком, акселерометром и другими датчиками;
- возможность углублённой работы с функциями смартфона. Как и в предыдущем пункте, такие вещи, как анимации, создание сложных интерфейсов и работа нейросетей прямо на устройствах реализуются, может быть, и не просто, но прогнозируемо;
- родной для платформы интерфейс. Нативные приложения обычно оперируют «платформенными» элементами интерфейса: меню, навигация, формы и все остальные элементы дизайна берутся от операционной системы и потому привычны и понятны пользователю.

Недостаток один — дороговизна разработки и поддержки. Для каждой платформы надо писать свой код. С ростом рынка мобильных приложений разработчики стали не просто дороги, а очень дороги.

И неестественные

Кроссплатформенные приложения пишутся сразу для нескольких платформ на одном языке, отличном от нативного. Как такой код может работать на разных устройствах? Тут тоже есть два подхода.

Первый заключается в том, что на этапе подготовки приложения к публикации он превращается в нативный для определённой платформы с помощью транспилера. Фактически один кроссплатформенный язык программирования «переводится» на другой.

Второй — в том, что к получившемуся коду добавляется определённая обёртка, которая, работая уже на устройстве, на лету транслирует вызовы из неродного кода к родным функциям системы.

Предполагается, что большая часть такого кода может переносится между платформами — очевидно, что, например, логика совершения покупок, сохранения товара в корзину, просчёта маршрута для такси, написания сообщения в мессенджер не меняется в зависимости о того, Android у клиента или iOS. Нужно лишь доработать UI и UX для платформ, но сейчас, в определённых пределах, даже это можно объединить — например, меню-гамбургер активно используется как на Android, так и на iOS. Так что даже внесений исправления в интерфейс для того, чтобы приложение отвечало духу и букве нужной платформы — вопрос желания, необходимой скорости и качества разработки.

Преимущества:

- стоимость и скорость разработки. Так как кода надо писать заметно меньше, то и стоимость работ снижается;
- возможность использовать внутренние ресурсы компании. Как мы покажем дальше, кроссплатформенную разработку мобильных

приложений зачастую можно осуществить силами уже существующих у вас программистов.

Недостатки:

- неродной интерфейс или, как минимум, необходимость работы с интерфейсом каждой платформы отдельно. У каждой системы свои требования к дизайну элементов и иногда они взаимоисключающи. При разработке это необходимо учитывать;
- проблемы в реализации сложных функций или возможные проблемы работы даже с простыми процедурами в силу ошибок самих фреймворков разработки. Кроссплатформенная среда лишь транслирует запросы к системным вызовам и интерфейсам в понимаемый ею, системой, формат, и потому на этом этапе возможны как сложности с пониманием, так и возникновение ошибок внутри самого фреймворка;
- скорость работы. Так как кроссплатформенная среда является «надстройкой» над кодом (не всегда, но в определённых ситуациях), в ней возникают свои задержки и паузы в отработке действий пользователя и выводе на экран результатов. Это было особенно заметно несколько лет назад на смартфонах, более маломощных относительно сегодняшних, однако сейчас, с ростом производительности мобильных устройств, этим уже можно пренебречь.

Как видите, эти два метода практически являются зеркальным отражением друг друга — то, что плюсы у нативной разработки, минусы у кроссплатформенной, и наоборот.

Популярные платформы и инструменты кроссплатформенной разработки

Как мы написали выше, есть два подхода — превращение кода в нативный на этапе сборки или добавление определённой обёртки, транслирующей вызовы к системе и от неё.

Cordova и PWA — два инструмента, работающие как раз в идеологии обёртки (рисунок 6).

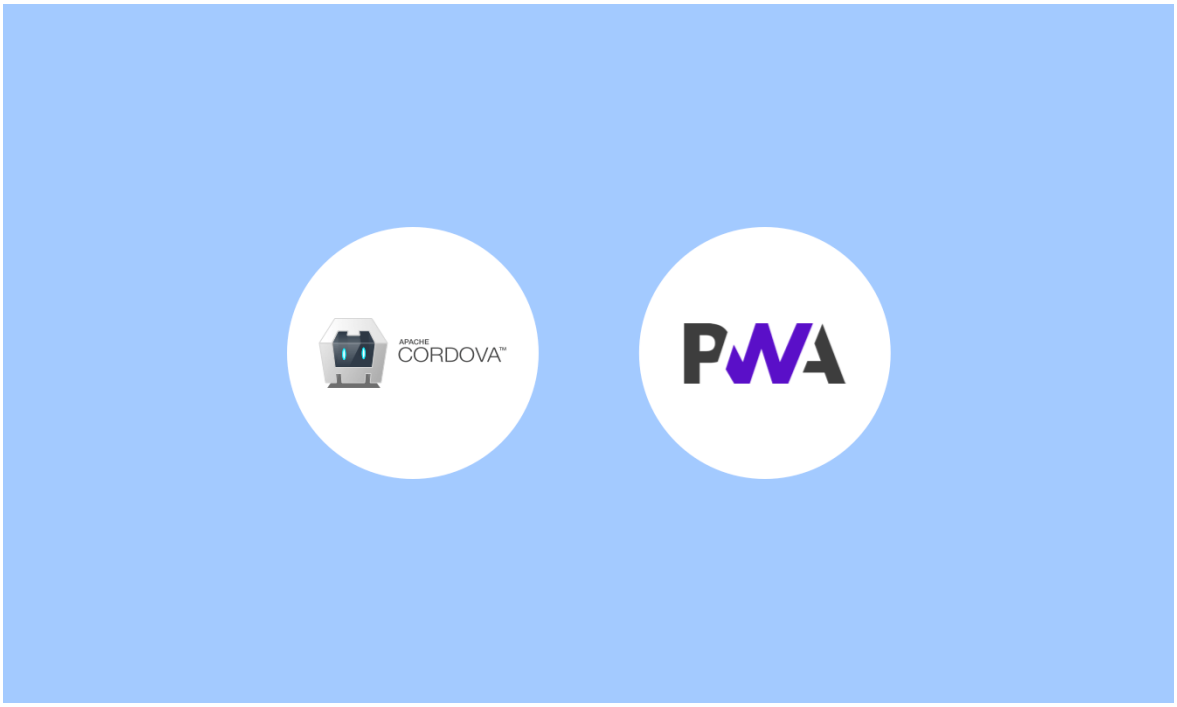

Рисунок 6 – Cordova и PWA

Cordova и HTML5

Одно из самых популярных направлений в кроссплатформенном программировании, которое часто по-народному называют PhoneGap. Фактически создаётся мобильный сайт, который «оборачивается» небольшим платформенным кодом, транслирующим вызовы от системы к приложению и обратно.

Все недостатки и достоинства тут выражены как нигде ярко. Вы можете использовать веб-разработчиков (HTML, CSS и JavaScript как основные

технологии) и за месяц или даже пару недель сделать первую версию приложения за относительно небольшие деньги. Да, она будет подтормаживать в работе, возможно, в ней будет не совсем точная геолокация, но она будет работать на всех устройствах и позволит вам, как минимум, протестировать спрос со стороны клиентов на мобильных устройствах.

Для такого подхода создано огромное количество фреймворков, но все они делают фактически одно и тоже. Различие между ними в том, что Cordova (PhoneGap) не задаёт ограничений и шаблонов на логику и UI для вашего HTML5-проекта, а фреймворки оперируют собственными готовыми UI-элементами, имитирующими мобильные платформы, и своей логикой разработки. В качестве примера такого подхода можно указать: Ionic Framework — обёртка; Framework7, Mobile Angular UI, Sencha Touch, Kendo UI — интерфейсные фреймворки.

PWA

Модная технология от Google — это те же самые веб-приложения, но за счёт использования определённых технологий (в первую очередь это так называемые Service Worker — работающие в фоновом режиме скрипты, и Web App Manifest — описание веб-приложения в понятном для мобильной системы виде) они без обёртки из PhoneGap могут работать как нативные. Они могут устанавливаться на домашний экран в обход магазина приложений, работать в офлайне, работать с пуш-уведомлениями, с нативными функциями.

Проблема в том, что не все платформы даже сейчас поддерживают эти «определённые технологии». В первую очередь это касается Apple, которой, видимо, очень не нравится возможность распространять приложения в обход App Store.

Учтя все недостатки HTML5-решений, многие компании создали инструменты, которые позволяют писать код на одном, не нативном, языке, а он

потом транслируется в нативный. Так убивается два зайца одновременно: кодовая база получается одна, а приложения получаются максимально близки к нативному.

Далее речь пойдет о платформах без веб-разработки (Рисунок 7).

Рисунок 7 – Платформы без веб-разработки

Xamarin

Платформа компании Microsoft. Используется стандартный для Enterprise-разработки язык программирования C#[22], кроссплатформенная среда разработки — Visual Studio. На выходе — нативные приложения для iOS, Android и Windows. Правда, относительно большого размера.

React Native

Платформа от Facebook[23] — приложения пишутся на JavaScript и с использованием CSS-подобных стилей. Интерфейс получается родной, а код интерпретируется уже на платформе, что придаёт ему нужную гибкость.

Будучи относительно молодой платформой, React Native пока очевидно (хоть и не катастрофически) страдает от недостатка средств разработки и документации.

Flutter

Естественно, не мог обойти тему кроссплатформенной разработки Android и iOS-приложеий и такой гигант, как Google. Flutter, пока, правда, существующий только в бета-версии, исповедует отличный от React Native и Xamarin подход. Он не превращает исходный код в нативный, который выполняется платформой, а на самом деле рисует окно на экране смартфона и отрисовывает все элементы сам. В качестве языка используется «фирменный» Dart, который Google создал как усовершенствованную версию JavaScript.

У этого есть как преимущества (например, внешне идентичные интерфейсы), так и недостатки (например, перерисовка интерфейса требует определённых затрат памяти и процессорного времени).

Платформа быстро развивается и Google вкладывает в это много сил и средств. Но по сравнению с Flutter даже React Native кажется вполне устоявшейся и впечатляющей экосистемой.

1.2.2 Анализ технологий разработки сервиса

Для мобильного приложения, чтобы загружать данные из базы данных, требуется веб-служба Application Programming Interface (API)[24] в ней будут прописаны инструкции для оправки запросов в базу данных и обработка полученных ответов. Так как требуется загружать данные из сети, хорошая практика использовать HTTP-запросы для отправки и получения данных, для этого удобнее всего использовать веб-сервис на архитектуре REST[25]. Сервисы для Универис были написаны на языке C# и платформе ASP.NET[26], следовательно, для удобства сервис мобильного приложения должен быть

разработан с использования тех же компонентов. В частности, компания Microsoft предлагает два решения, а именно Windows Communication Foundation (WCF)[27] и ASP.NET Web API[28]. Представим некоторые особенности этих технологий в Таблица 2[29]

Таблица 2 – WCF vs ASP.NET Web API

WCF	ASP.NET Web API
Включает службы сборки, которые поддерживают несколько транспортных протоколов (HTTP, TCP, UDP и пользовательский транспорт) и допускают переключение между ними.	Только HTTP. Первоклассная модель программирования для HTTP. Больше подходит для доступа из различных браузеров, мобильных устройств, д Включение широкая доступность.
Включает службы сборки, которые поддерживают разные кодирования (текст, MTOM и двоичные) одного типа сообщений и допускают переключение между ними.	Позволяет создавать сетевые API-интерфейсы, которые поддерживают большое количество различных типов содержимого, в том числе XML, JSON и т. д.
Поддерживает создание служб по таким стандартам WS-*, как надежный обмен сообщениями, транзакции и безопасность сообщений.	Использует основные протоколы и форматы, такие как HTTP, WebSockets, SSL, JSON и XML. Отсутствует поддержка протоколов высокого уровня, таких как надежный обмен сообщениями и транзакции.
Поддерживает шаблоны обмена сообщениями «запрос-ответ», «односторонний» и «дуплексный».	HTTP — запрос/ответ, но дополнительные шаблоны, которые могут поддерживаться через SignalR[30] и интеграции WebSockets.
Службы WCF SOAP могут быть описаны в языке WSDL, что позволяет автоматическим средствам создавать прокси клиентов даже для служб со сложными схемами.	Имеются различные способы описания Web API – от автоматически формируемых html-страниц справки с описанием фрагментов до структурированных метаданных для интеграции API в OData.
Поставляется вместе с платформой	Поставляется вместе с платформой

WCF	ASP.NET Web API
.NET Framework.	.NET Framework, но имеет открытый код и доступна также по внешним каналам как независимая загрузка.

1.2.3 Анализ технологий разработки мобильных приложений

Для каждого мобильного приложения, не зависимо от платформы, требуется локальная база данных, где будет хранится информация, загруженная из сервиса. Это одно из требований – иметь доступ к требуемым данным без повторной загрузки через сеть Интернет.

Разберем какие системы управления базами данных существуют, какие у них преимущества и недостатки [31].

SQLite

Легко встраиваемая в приложения база данных. Так как это система базируется на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с этой СУБД обращения происходят напрямую к файлам (в этих файлах хранятся данные), вместо портов и сокетов в сетевых СУБД. Именно поэтому SQLite очень быстая, а также мощная благодаря технологиям обслуживающих библиотек.

Типы данных SQLite:

- NULL – значение NULL;
- INTEGER – знаковое целочисленное значение, использует 1, 2, 3, 4, 6, или 8 байт в зависимости от порядка числа;
- REAL – число с плавающей точкой, занимает 8 байт для хранения числа в формате IEEE;

– TEXT – текстовая строка, при хранении используются кодировки UTF-8, UTF-16BE или UTF-16LE;
– BLOB – тип данных BLOB, массив двоичных данных (предназначенный, в первую очередь, для хранения изображений, аудио и видео).

Преимущества SQLite:

– Файловая структура – вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
– Используемые стандарты – хотя может показаться, что эта СУБД примитивная, но она использует SQL. Некоторые особенности опущенны (RIGHT OUTER JOIN или FOR EACH STATEMENT), но основные все-таки поддерживаются;
– Отличная при разработке и тестировании – в процессе разработки приложений часто появляется необходимость масштабирования. SQLite предлагает всё что необходимо для этих целей, так как состоит всего из одного файла и библиотеки написанной на языке C.

Недостатки SQLite:

– отсутствие системы пользователей – более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях;
– отсутствие возможности увеличения производительности – опять, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

MySQL

MySQL – это самая распространенная полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая

успешно работает с различными сайтами и веб приложениями. Обучиться использованию этой СУБД довольно просто, так как на просторах интернета вы легко найдете большее количество информации.

Заметка: стоит заметить, что благодаря популярности этой СУБД, существует огромное количество различных плагинов и расширений, облегчающих работу с системой.

Несмотря на то, что в ней не реализован весь SQL функционал, MySQL предлагает довольно много инструментов для разработки приложений. Так как это серверная СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL.

Типы данных MySQL:

- TINYINT – очень малые целочисленные значения;
- SMALLINT – малые целочисленные значения;
- MEDIUMINT – средние целочисленные значения;
- INT или INTEGER – стандартные целочисленные значения;
- BIGINT – большие целочисленные значения;
- FLOAT – маленькие значения с плавающей точкой (точность до одного значения после точки). Всегда знаковые значения;
- DOUBLE, BOUBLE PRECISION, REAL – Стандартные значения с плавающей точкой. Всегда знаковые;
- DECIMAL, NUMERIC – распакованное значение с плавающей точкой, всегда знаковое;
- DATE – дата;
- DATETIME – дата и время в одном значении;
- TIMESTAMP – временная отметка timestamp;
- TIME – время;
- YEAR – год, 2 или 4 числа (4 – по-умолчанию);

- CHAR – строковое значение фиксированной длины, справа всегда добавляются пробелы до указанной длины при сортировке;
- VARCHAR – строковое значение переменной длины;
- TINYBLOB, TINYTEXT – значение типа BLOB или TEXT, 255 (2^8 – 1) символов – максимальныя длина;
- BLOB, TEXT – значение типа BLOB или TEXT, 65535 (2^16 – 1) символов – максимальныя длина;
- MEDIUMBLOB, MEDIUMTEXT – значение типа BLOB или TEXT, 16777215 (2^24 – 1) символов – максимальныя длина;
- LONGBLOB, LONGTEXT – значение типа BLOB или TEXT, 4294967296 (2^32 – 1) символов – максимальныя длина;
- ENUM – перечисление;
- SET – множество.

Преимущества MySQL:

- Простота в работе – установить MySQL довольно просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД;
- Богатый функционал – MySQL поддерживает большинство функционала SQL;
- Безопасность – большое количество функций обеспечивающих безопасность, которые поддерживается по умолчанию;
- Масштабируемость – MySQL легко работает с большими объемами данных и легко масштабируется;
- Скорость – упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки MySQL:

- Известные ограничения – по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях;
- Проблемы с надежностью – из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности;
- Медленная разработка – Хотя MySQL технически открытое ПО, существуют жалобы на процесс разработки. Стоит заметить, что существуют другие довольно успешные СУБД созданные на базе MySQL, например MariaDB.

PostgreSQL

PostgreSQL является самым профессиональным из всех трех рассмотренных нами СУБД. Она свободно распространяемая и максимально соответствует стандартам SQL. PostgreSQL или Postgres стараются полностью применять ANSI/ISO SQL стандарты своевременно с выходом новых версий.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID).) Благодаря мощным технологиям Postgre очень производительна. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие ACID. PostgreSQL очень легко расширять своими процедурами, которые называются хранимые процедуры. Эти функции упрощают использование постоянно повторяемых операций.

Хотя PostgreSQL и не может похвастаться большой популярностью в отличии от MySQL, существует довольно большое число приложений,

облегчающих работу с PostgreSQL, несмотря на всю мощность функционала. Сейчас довольно легко установить эту СУБД используя стандартные менеджеры пакетов операционных систем.

Типы данных PostgreSQL:

- bigint – знаковое 8-ми битное целочисленное значение;
- bigserial – автоматически инкрементируемое 8-ми битное целочисленное значение;
- bit[(n)] – строка постоянной длины;
- bit varying [(n)] – строка переменной длины;
- boolean – булево значение (true/false);
- box – прямоугольник на плоскости;
- bytea – бинарные данные (массив байтов);
- character varying [(n)] – строковое значение переменной длины;
- character [(n)] – строковое значение постоянной длины;
- cidr – IPv4/IPv6 сетевой адрес;
- circle – круг на плоскости;
- date – календарная дата (год, месяц, день);
- double precision – число с плавающей точкой двойной точности (8 байт);
- inet – IPv4/IPv6 адрес хоста;
- integer – знаковое 4-ех байтовое целочисленное значение;
- interval [fields][(p)] – отрезок времени;
- line – бесконечная прямая на плоскости;
- lseg – отрезок на плоскости;
- macaddr – MAC адрес;
- mony – валютное значение;

– numeric [(p, s)] – точное численное значение с выбранной точностью;
– path – геометрическая кривая на плоскости;
– point – геометрическая точка на плоскости;
– polygony – многоугольник на плоскости;
– real – число с плавающей точкой одинарной точности (4 байта);
– smallint – знаковое целочисленное значение (4 байта);
– serial – автоматические инкрементируемое целочисленное значение (4 байта);
– text – строковое значение переменной длины;
– time [(p)] [without time zone] – время суток (без часового пояса);
– time [(p)] with time zone – время суток (включая часовой пояс);
– timestamp [(p)] [without time zone] – дата и время (без часового пояса);
– timestamp [(p)] with time zone – дата и время (с часовым поясом);
– tsquery – текстовый поисковый запрос;
– tsvector – документ текстового поиска;
– txid_snapshot – пользовательский снимок транзакции с ID;
– uuid – универсальный уникальный идентификатор;
– xml – XML данные.

Достоинства PostgreSQL:

– Открытое ПО соответствующее стандарту SQL – PostgreSQL – бесплатное ПО с открытым исходным кодом. Эта СУБД является очень мощной системой;
– Большое сообщество – существует довольно большое сообщество, в котором вы запросто найдёте ответы на свои вопросы;

– Большое количество дополнений – несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими;
– Расширения – существует возможность расширения функционала за счет сохранения своих процедур;
– Объектность – PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого.

Недостатки PostgreSQL:

– Производительность – при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL;
– Популярность – по своей природе, популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество;
– Хостинг – в силу выше перечисленных факторов иногда довольно сложно найти хостинг с поддержкой этой СУБД.

1.3. ВЫВОД

Анализ предметной области дал нам представление о существующих решений в данной области. Было произведено сравнение шести мобильных приложений, которые так или иначе относятся к информационным системам различных вузов. Каждое приложение имеет свои достоинства и недостатки. Основной недостаток приложений, что они позиционируют себя не как приложения для полноценных ИАС вузов, а обычные приложения для

просмотра расписания занятий для студентов. Но в задачу текущей работы входит куда более широкая задача, а именно разработка мобильного приложения для корпоративной информационно-аналитической системы Универис, то есть, кроме просмотра расписания занятий для студентов или преподавателей, приложение должно способно обеспечить комфортное решение задач различных подразделений учебного заведения. Такие как: прохождение опросов, просмотр учебных планов, получение доступа в научную библиотеку и др. Другим недостатком стало то, что не все аналоги не имеют авторизацию в приложение, то есть любой может скачать себе и пользоваться, что не является безопасным решением.

Как было рассказано в подразделе 1.2.1 анализа технологических решений, технологии разработки мобильных приложений делятся на два типа: кроссплатформенные и естественные. Также в этом разделе были представлены достоинства и недостатки таких решений. В частности, естественная разработка приложений под все платформы есть дорогое и требующее большего количества времени решение данной задачи, нежели кроссплатформенные решения. Однако проанализировав недостатки кроссплатформенной разработки, были сравнены риски и принято решение, что естественная разработка куда более надежный способ.

Каждая платформа в данный момент поддерживает по два естественных языка Objective-C и Swift для iOS и Java или Kotlin для Android.

Определимся сначала с iOS. Как сказано в этом источнике [32], Swift — это новый мощный язык программирования для iOS и OS X, который позволяет разработчикам создавать потрясающие приложения с ещё большей лёгкостью. Swift спроектирован таким образом, чтобы помогать разработчикам в создании более безопасного и надёжного кода, устраняя при этом целые категории распространённых программных ошибок. Он может сосуществовать с кодом

Objective-C, позволяя разработчикам легко интегрировать Swift в уже созданные ими приложения. Следовательно, ничего не мешает выбрать этот язык для реализации клиентского приложения под iOS.

Аналогичная ситуация с платформой Android. В источнике [33] сказано, что разработчики Kotlin ставят перед собой задачу создать язык, «лишенный ограничений Java, неисправимых в связи с проблемами обратной совместимости». Благодаря статичности типов Kotlin будет безопаснее, чем Java, а кроме того, новый язык будет лаконичнее. Еще одна цель – сделать его более простым, чем «главный конкурент» – Scala[34]. Компилятор Kotlin и подключаемый модуль для IntelliJ[35] будут предлагаться в открытых кодах по лицензии Apache[36]. В JetBrains[37] собираются обеспечить возможность расширения Kotlin различными способами. Язык Kotlin был выбран для написания приложения под Android.

В подразделе 1.2.2 был проведен анализ двух технологий для создания веб-службы API обращения к базе данных и обработке полученной информации. Из Таблица 2 можно сделать вывод, что технология WCF более универсальная, имеет расширенный функционал, что и сыграло роль в выборе данной технологии.

Подраздел 1.2.3 был посвящен анализу существующих СУБД, которые можно включить в мобильное приложение. Исходя из всех преимуществ и недостатков, наиболее подходящей стало СУБД SQLite.

Таким образом можно точно сформулировать задачу по разработке мобильного приложения.

Требуется мобильные приложения под все платформы (Android и iOS), которое будет сделано на естественных для каждой платформы языках, оно должно включать в себя реализацию различных разделов КИАС Универис, вход в приложение должен быть с помощью аккаунтов Универиса, приложение

должно быть удобным и безопасным. Также к мобильному приложению требуется веб-служба для работы с данными, которая будет поддерживать HTTP-запросы, иметь архитектуру REST-сервиса и написана на основе технологии WCF. Вся информация на устройстве должна храница в локальной базе данных под управлением SQLite.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации данной системы необходим следующий набор подсистем:

1. *HTTP-сервер IIS,* на нем реализованный, с помощью платформы Windows Communication Foundation (WCF), построенный на архитектуре REST, веб-служба (сервис). С помощью веб-сервиса обеспечивается организация взаимодействия с пользователем и данными, выдача, получение результатов в формате JSON[38].
2. *Сервис для авторизации.* Для безопасности нужен отдельный сервис для авторизации пользователь, на той же технологии WCF.
3. *Клиентское приложение с графическим интерфейсом под различные платформы* – мобильные приложения под Android и iOS.
4. *SQLite* база данных для каждого приложения на всех платформах, обеспечивающая хранение данных о расписание и другой информации прямо на устройстве.

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

В данном подразделе определим какой функционал будет в мобильном приложение. Для этого воспользуемся диаграммой прецедентов (вариантов использования).

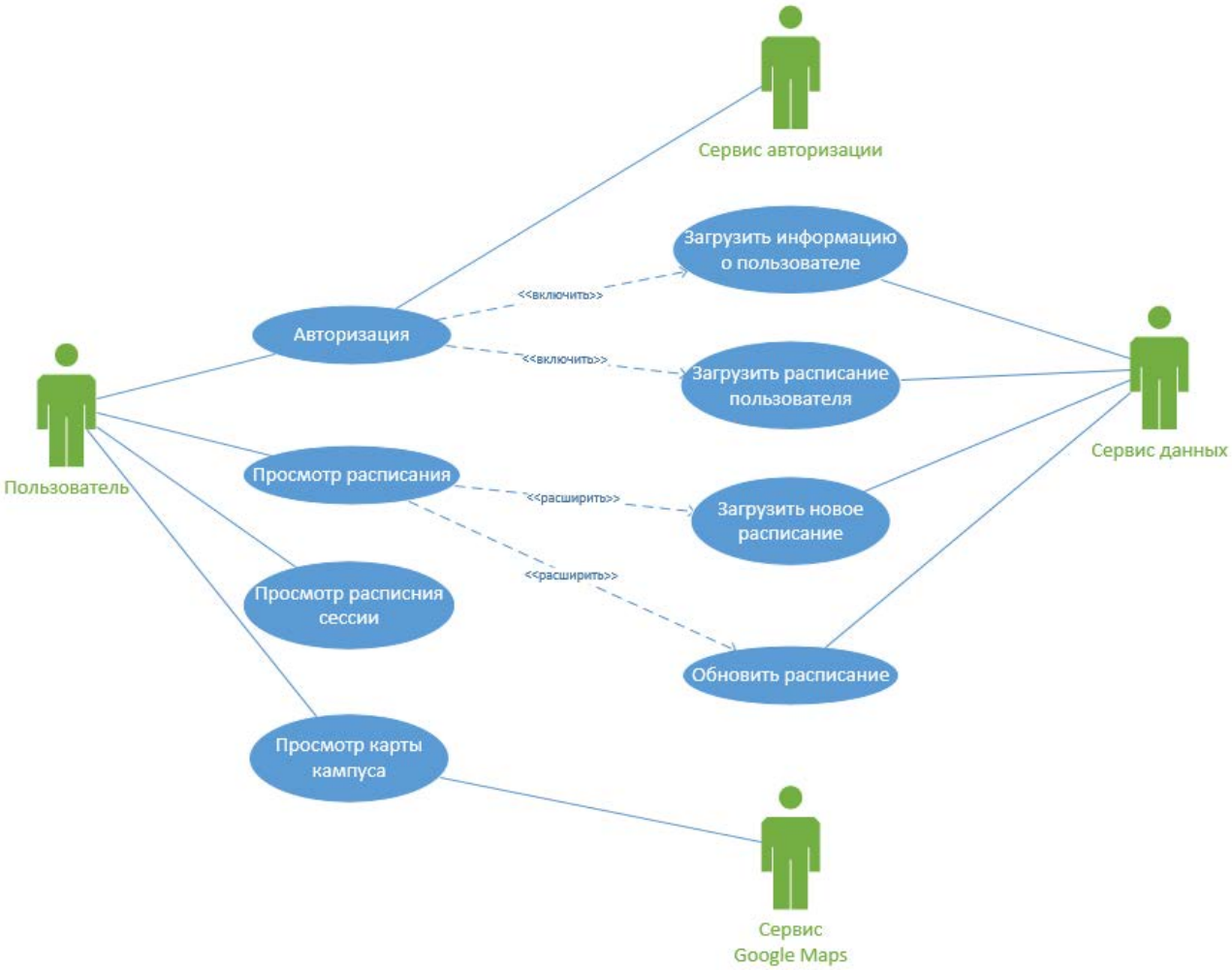

Рисунок 8 – Диаграмма вариантов использования мобильного приложения

На Рисунок 8 изображена диаграмма вариантов использования мобильного приложения. Данная диаграмма является верной для всех приложений, не зависимо от платформы. Но стоит заметить, что данная диаграмма уже может быть не актуальна, на момент прочтения, так как приложения постоянно развивается и пополняется функционалом, в данной работе будет описан функционал на момент написания работы. Можно выделить следующих ключевых актеров, взаимодействующих с системой:

1. *Пользователь*: пользователь мобильного приложения – это сотрудник или студент ФГАОУ ВО «ЮУрГУ (НИУ)», который имеет доступ к КИАС Универис. В зависимости от роли в университете пользователь наделяется

разными правами доступа, так, например, сотруднику при Загрузке информации о пользователе возвращается информация о его должности и структурном подразделение, где он работает, если сотрудник имеет роль преподавателя, то ему возвращается расписание его занятий, а студенту – номер группы, где он учится и расписание занятий.

2. *Сервис авторизации*: первое, что требуется пользователю при старте работы с мобильным приложением – это авторизоваться с помощью аккаунта КИАС Универис. Как видно из диаграммы, сервис авторизации работает автономно от другого *Сервиса данных*, сделано это из соображения безопасности. Как только пользователь авторизуется, ему приходит токен с функцией устаревания, этот токен оправляется на *Сервис данных* при каждом запросе, для получения данных. Как только токен устаревает, производится повторная авторизация, без участия *Пользователя*, если *Пользователь* все еще имеет доступ к КИАС Универис с тем же логином и паролем.
3. *Сервис данных*: основной сервис для получения информации из базы данных, написанный на технологии WCF, в дальнейшем, возможно, набор автономных сервисов для разных задач.
4. *Сервис Google Maps*: еще один API сервис от компании Google, подробнее в источнике [39], он предназначен для загрузки карты с корпусами и общежитиями студгородка ФГАОУ ВО «ЮУрГУ (НИУ)».

Рассмотрим прецеденты использования разрабатываемой системы.

Пользователю необходимо *авторизоваться,* чтобы иметь возможность делать запросы на сервис для получения данных: для этого он вводит в соответствующем окне системе свои логин и пароль учетной записи КИАС Универис.

После процедуры авторизации пользователю автоматически загружается информация о нем, а если он имеет роль преподавателя или студента, то еще и расписание его занятий, если таково имеется. Далее он имеет возможность просматривать свое расписание, загружать новое (не имеет значение какая роль у пользователя, он может загрузить расписание любой студенческой групп или преподавателя) или обновить текущее (по умолчанию, обновление происходит автоматически). Кроме того, ему доступно расписание сессии, но оно отображается только, когда его составят на сайте. Если пользователь плохо ориентируется в студгородке, ему доступна карта кампуса.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

В системе должна быть предусмотрена защита от следующих типов угроз безопасности:

1. Просмотр информации из КИАС Универис неавторизованным пользователем.
1. Анонимная отправка запросов.
2. Перехват данных для авторизации в теле запроса.

Для начала поясним в чем проблема этих угроз.

Если допустить, что загрузка будет доступно неавторизованному пользователю, то это может вызвать повышенную нагрузку на сервер, а также предоставить доступ к данным, которые возможно должны быть срыты для определенной группы пользователей. К этому добавляется невозможность установить роль пользователя и его ограничения.

Анонимная отправка запроса, то есть тоже самое, что и отправка запросов неавторизованного пользователя, создаст трудности в логирование деятельности сервиса и, соответственно, мобильного приложения.

Решением проблемы 1 и 2 является простая авторизация, которая будет происходить с помощью аккаунта КИАС Универис.

Проблемы в пункте 3 решаются путем шифрования SSL[40] информационного потока (RSA, DSA, AES, DES).

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

При описании архитектуры разрабатываемого мобильного приложения была спроектирована общая диаграмма компонентов для отображения взаимодействия логических частей приложения, представленная (рисунок 9).

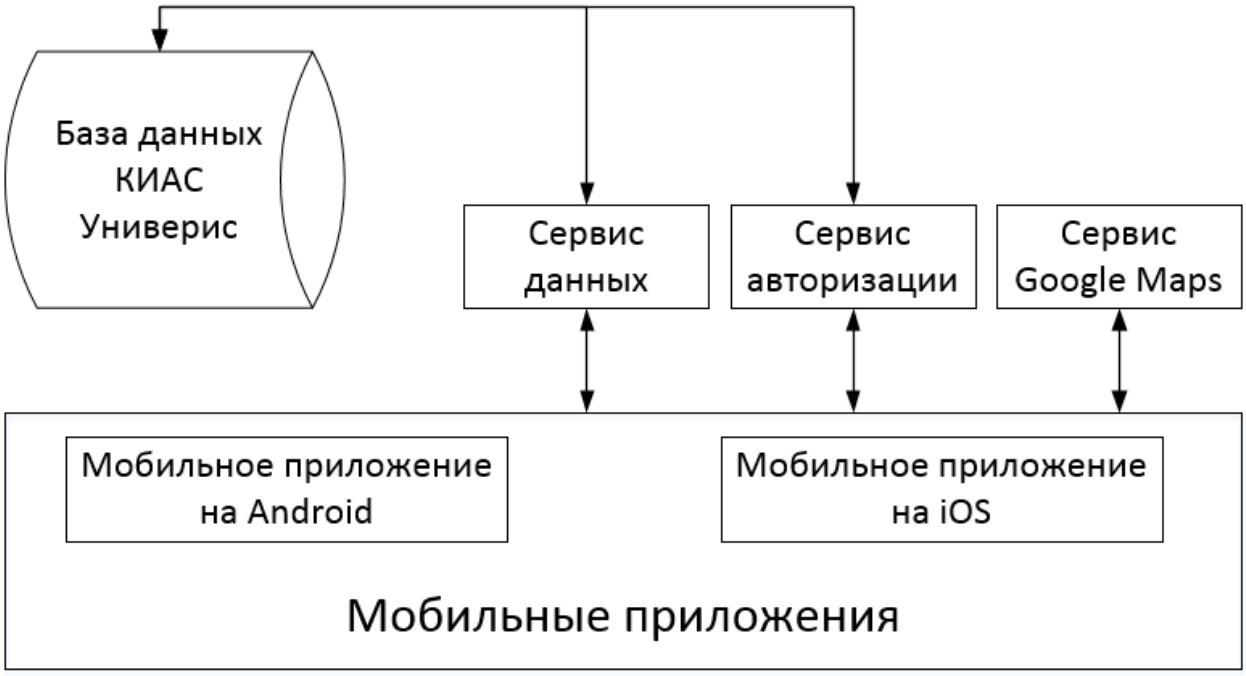

Рисунок 9 – Диаграмма компонентов проектируемой системы

Данная схема является общей и не подробной, но чётко показывает из чего состоит архитектура приложения. Условно ее можно разделить на три группы:

– База данных КИАС Универис – это единая база, где хранится вся требуемая информация, начиная от данных о пользователе и его учетной записи и заканчивая расписанием занятий, а также другие данные, которые в приложение пока не используются;

- Набор сервисов, в данном случае их пока три, но приложения реализованы таким образом, что подключаемых сервисов может быть сколько угодно, об этом подробнее в главе 4. Как было уже сказано все сервисы работают независимо и автономно от друг друга;
- Мобильные приложения, соответственно, приложение под Android и под iOS.

Разработка базы данных КИАС Универис не входит в данную работу, так как она уже давно разработана и работает с 2012 года [5], поэтому не будем подробно о ней говорить. Про сервис Google Maps также не будем говорить, ссылка, где можно про него прочитать, выше.

Рассмотрим подробно из чего состоит сервис данных, Рисунок 10.

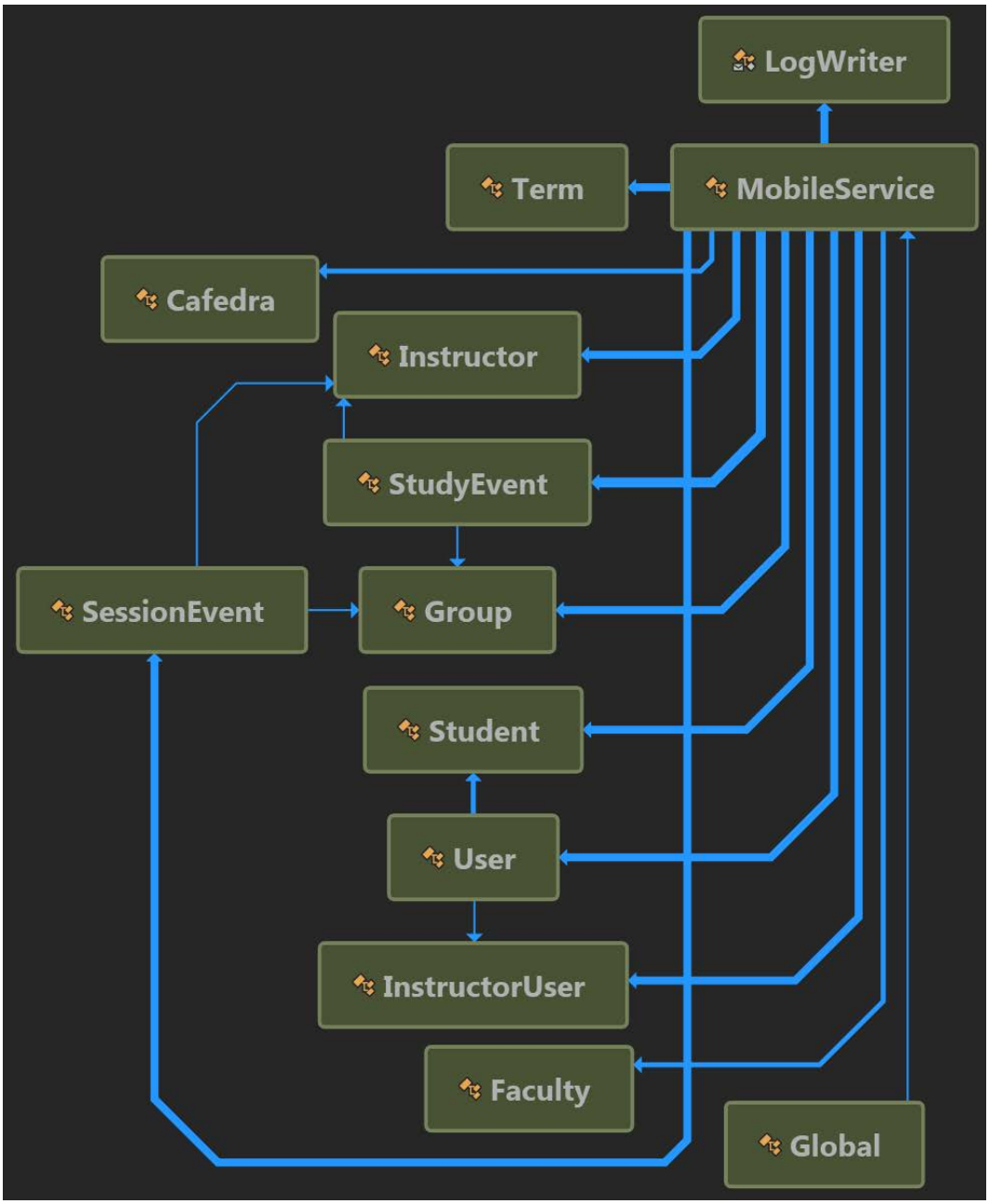

Рисунок 10 – Сервис данных

Как видно из рисунка, сервис имеет общий компонент MobileService, это класс где данные загружаются из базы и передаются в понятные мобильному приложению модели представления данных с соответствующими названиями (Faculty – факультет, Group – группы и т. д.). Также сервис имеет возможность ведения логов, класс LogWriter.

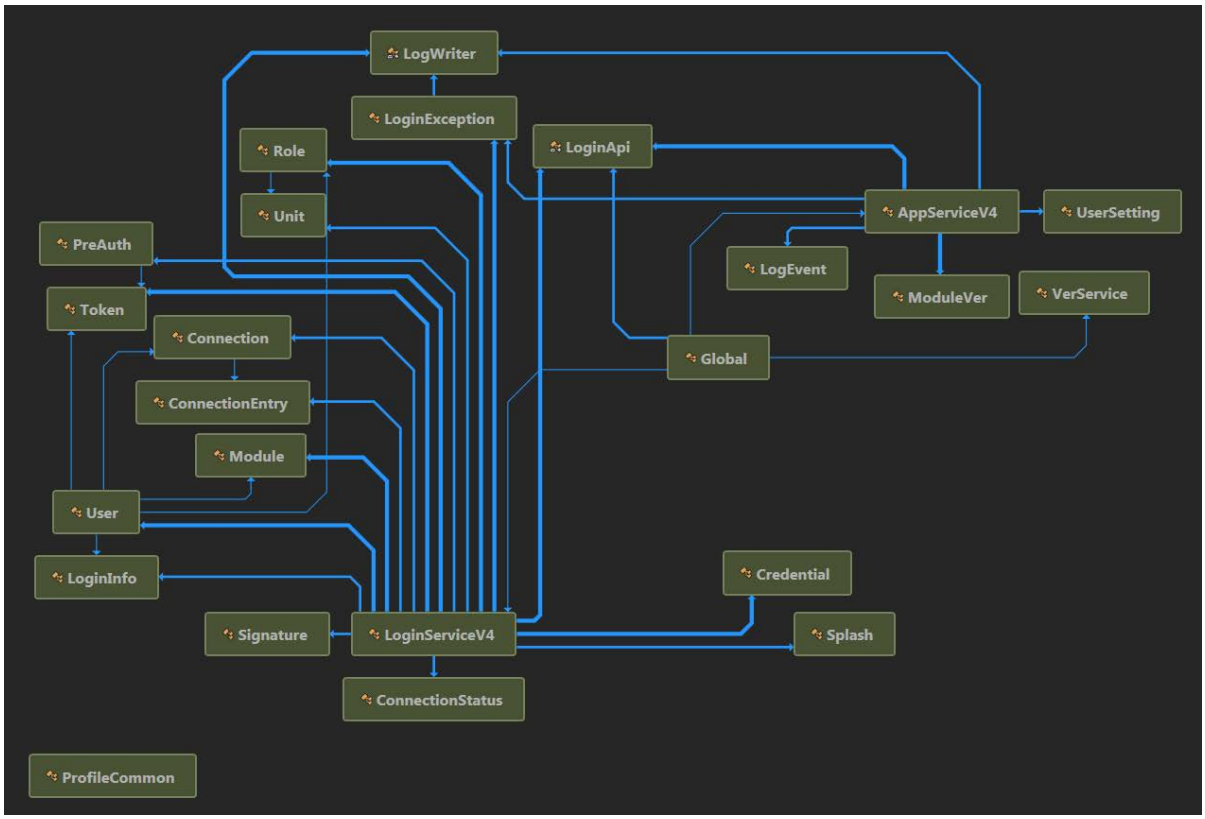

Рисунок 11 – Сервис авторизации

Рисунок 11 показывает схему компонентов классов сервиса авторизации, данный сервис также используется в основном сайте КИАС Универис, аналогично сервису данных можно заметить, что присутствует основной класс управления, а именно LoginServiceV4, а от него идут классы моделей представления данных, которые требуются для авторизации. Как видно присутствует класс для обработки исключений LoginExeption из него формируется отчет в класс логирования LogWriter. Ключевыми классами являются следующие:

– LoginInfo – в классе хранится информация о авторизованном пользователе;
– User – класс храни информацию об пользователе и предает ее в класс LoginInfo;

– Token – класс, где формируется токен, который позже будет использоваться при отправке запросов с мобильного приложения.

Основой для построения архитектуры мобильных приложения является модель представления Model-View-Controller (MVC)[41]. Мы не будем описывать всю архитектуру, приложения очень объемные по количеству компонентов, мы расскажем общую идею.

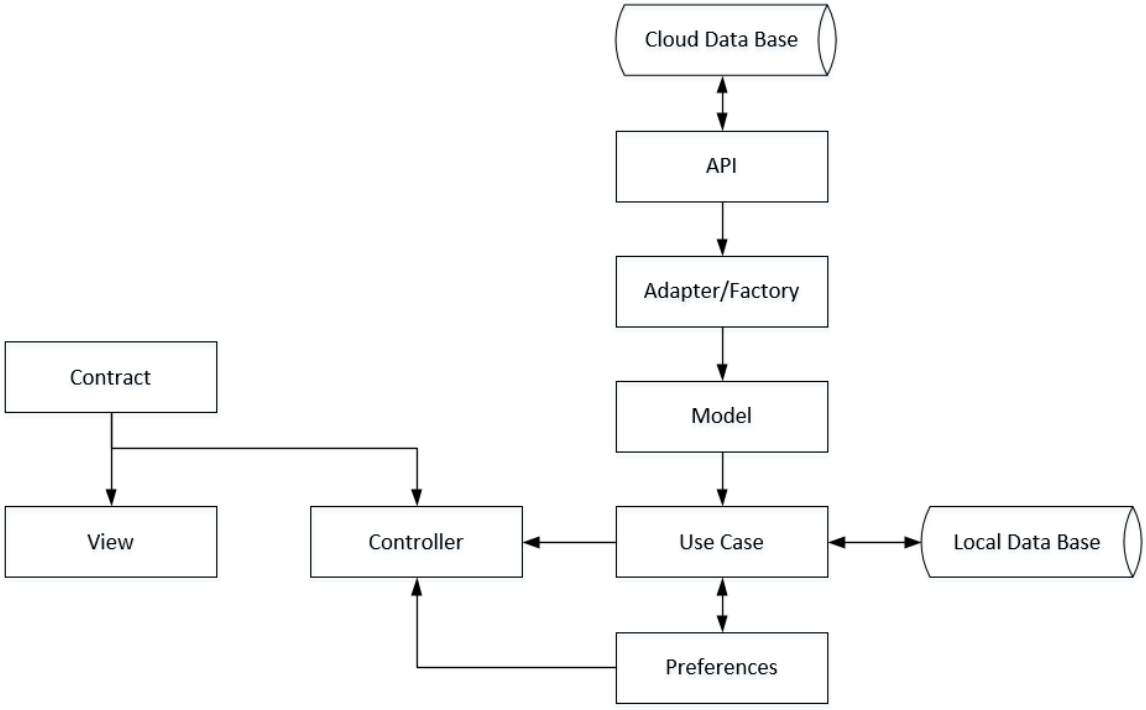

Рисунок 12 – Архитектура мобильного приложения

На Рисунок 12 показана схема компонентов мобильного приложения, которая формирует ее архитектуру, каждый компонент, кроме баз данных, это класс, от которого могут наследоваться другие классы.

Опишем назначение каждого класса.

Cloud Data Base – это та самая база данных КИАС Универис, о которой было много сказано выше.

API – это набор классов, для каждой сущности, где прописаны все требуемые запросы к Сервису данных, о котором тоже было много написано

выше. Каждый запрос запрограммирован таким образом, чтоб данные из сервиса приходили в формате JSON.

Adapter или Factory – Factory это образующий класс, который реализует паттерн проектирования Фабричный метод[42], он включает в себя класс Adapter, то есть класс внутри класса. Класс Adapter содержит методы для считывания возращенных данных в формате JSON и записывает в классы модели представления.

Model – это собственно и есть класс данных, в нем не содержится функций, только сущности.

UseCase – этот функциональный класс, он содержит в себе какую-то одну функцию, в зависимости от задачи, которая производит набор действий, например, функция авторизации включает в себя выполнение ряда задач: отправка логина и пароля, загрузка информации о пользователе и др. В зависимости от задачи, класс UseCase записывает данные в различной форме: в локальную базу данных, в конфигурационный файл на устройстве, либо направляет данные в контроллер.

Local Data Base – это локальная база данных, под управлением СУБД SQLite. В этой базе данных содержится информация, которую загрузил пользователь.

Preferences – это другой вид хранилища, он представляет из себя конфигурационный файл и содержит значения, которые не удобно хранить в БД, такие как, например, текущий семестр или день недели. Создавать для таких данных новые таблицы нецелесообразно и ведет к снижению производительности и нерациональное использование памяти устройства.

Contract – это не совсем класс, это интерфейс, от которого наследуются два класса: Controller и View. Сделано это для того, чтоб можно было легко взаимодействовать между этими классами.

Controller – этот класс отвечает обработку данных полученных из UseCase для передачи в класс View.

View – класс отображения элементов на экране, здесь прописывается разметка интерфейса.

В приложениях A-G будет представлен пример исходного кода данной архитектуры на языке Kotlin.

3.2. ОПИСАНИЕ ДАННЫХ

Все данные в приложение принимаются и отправляются в формате JSON. Такой формат был выбран из-за удобства чтения и легковесности, нежели, например, XML[43]. Как было сказано со стороны приложения считыванием данных занимается класс Adapter. Сервис отправляет данные в виде ответов на HTTP-запросы, в зависимости от заголовков запросов возвращаться они могут в двух форматах в JSON и XML.

Описание базы данных. Как было сказано, приложение имеет локальную базу данных. Это, в данный момент, не большая база из шести несвязанных таблиц.

Список таблиц и значений:

- GroupTable – в этой таблице хранится информация о студенческой группе;
    - id (TEXT) – поле идентификатор (первичный ключ);
    - facultyId (TEXT) – идентификатор факультета;
    - name (TEXT) – название группы;
    - number (INTEGER) – номер группы;
    - courseNumber (INTEGER) – номер курса;
    - studyForm (TEXT) – форма обучения;
    - termDate (INTEGER) – дата начала семестра;

- changedDate (INTEGER) – дата смены семестра;
- termNumber (INTEGER) – номер семестра;
- updateDate (INTEGER) – дата смены расписания;
- isModified (INTEGER) – флаг модификации;
- scheduleHash (TEXT) – хеш расписания.

– LecturerTable – в этой таблице хранится информация о преподавателе;
- id (TEXT) – поле идентификатор (первичный ключ);
- firstName (TEXT) – имя преподавателя;
- middleName (TEXT) – отчество преподавателя;
- lastName (TEXT) – фамилия преподавателя;
- termDate (INTEGER) – дата начала семестра;
- updateDate (INTEGER) – дата смены расписания;
- departmentName (TEXT) – название корпуса;
- scheduleHash (TEXT) – хеш расписания.

– StudentEventTable – таблица занятий студентов;
- id (TEXT) – поле идентификатор (первичный ключ);
- groupId (TEXT) – идентификатор группы (внешний ключ);
- subjectName (TEXT) – название предмета;
- lecturerId (TEXT) – идентификатор преподавателя (внешний ключ);
- eventDate (INTEGER) – дата занятия;
- timeStart (INTEGER) – время начала;
- timeEnd (INTEGER) – время конца;
- weekNumber (INTEGER) – номер недели;
- weekDay (INTEGER) – номер дня;
- termNumber (INTEGER) – номер семестра;

- termPart (INTEGER) – номер части семестра;
- room (TEXT) – аудитория;
- building (TEXT) – корпус;
- pairNumber (INTEGER) – номер пары;
- subGroupNumber (INTEGER) – номер подгруппы;
- eventType (TEXT) – тип занятия;
- lecturerFirstName (TEXT) – имя преподавателя;
- lecturerMiddleName (TEXT) – отчество преподавателя;
- lecturerLastName (TEXT) – фамилия преподавателя.

– LecturerEventTable – таблица занятий преподавателя;
- id (TEXT) – поле идентификатор (первичный ключ);
- groupId (TEXT) – идентификатор группы (внешний ключ);
- subjectName (TEXT) – название предмета;
- lecturerId (TEXT) – идентификатор преподавателя (внешний ключ);
- eventDate (INTEGER) – дата занятия;
- timeStart (INTEGER) – время начала;
- timeEnd (INTEGER) – время конца;
- weekNumber (INTEGER) – номер недели;
- weekDay (INTEGER) – номер дня;
- termNumber (INTEGER) – номер семестра;
- termPart (INTEGER) – номер части семестра;
- room (TEXT) – аудитория;
- building (TEXT) – корпус;
- pairNumber (INTEGER) – номер пары;
- subGroupNumber (INTEGER) – номер подгруппы;
- eventType (TEXT) – тип занятия;

- groupNames (TEXT) – название группы.

– StudentSessionEventTable – таблица расписания экзаменационной сессии студентов;
  - id (TEXT) – поле идентификатор (первичный ключ);
  - groupId (TEXT) – идентификатор группы (внешний ключ);
  - subjectName (TEXT) – название предмета;
  - lecturerId (TEXT) – идентификатор преподавателя (внешний ключ);
  - eventDate (INTEGER) – дата занятия;
  - timeStart (INTEGER) – время начала;
  - timeEnd (INTEGER) – время конца;
  - termNumber (INTEGER) – номер семестра;
  - room (TEXT) – аудитория;
  - building (TEXT) – корпус;
  - eventType (TEXT) – тип занятия;
  - lecturerFirstName (TEXT) – имя преподавателя;
  - lecturerMiddleName (TEXT) – отчество преподавателя;
  - lecturerLastName (TEXT) – фамилия преподавателя.

– LecturerSessionEventTable – таблица расписания экзаменационной сессии преподавателей.
  - id (TEXT) – поле идентификатор (первичный ключ);
  - groupId (TEXT) – идентификатор группы (внешний ключ);
  - subjectName (TEXT) – название предмета;
  - lecturerId (TEXT) – идентификатор преподавателя (внешний ключ);
  - eventDate (INTEGER) – дата занятия;
  - timeStart (INTEGER) – время начала;

- timeEnd (INTEGER) – время конца;
- termNumber (INTEGER) – номер семестра;
- room (TEXT) – аудитория;
- building (TEXT) – корпус;
- eventType (TEXT) – тип занятия;
- groupNames (TEXT) – название группы.

Все таблицы были сгенерированы программным кодом с помощью класса android.database.sqlite.SQLiteDatabase[44].

4. РЕАЛИЗАЦИЯ

4.1. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ НА ANDROID

Приложение было реализовано в среде разработке Android Studio[45], это среда, разработанная российскими разработчиками JetBrains, компания Google признана ее официальной средой разработки приложений на Android. На Рисунок 13 представлен интерфейс среды.

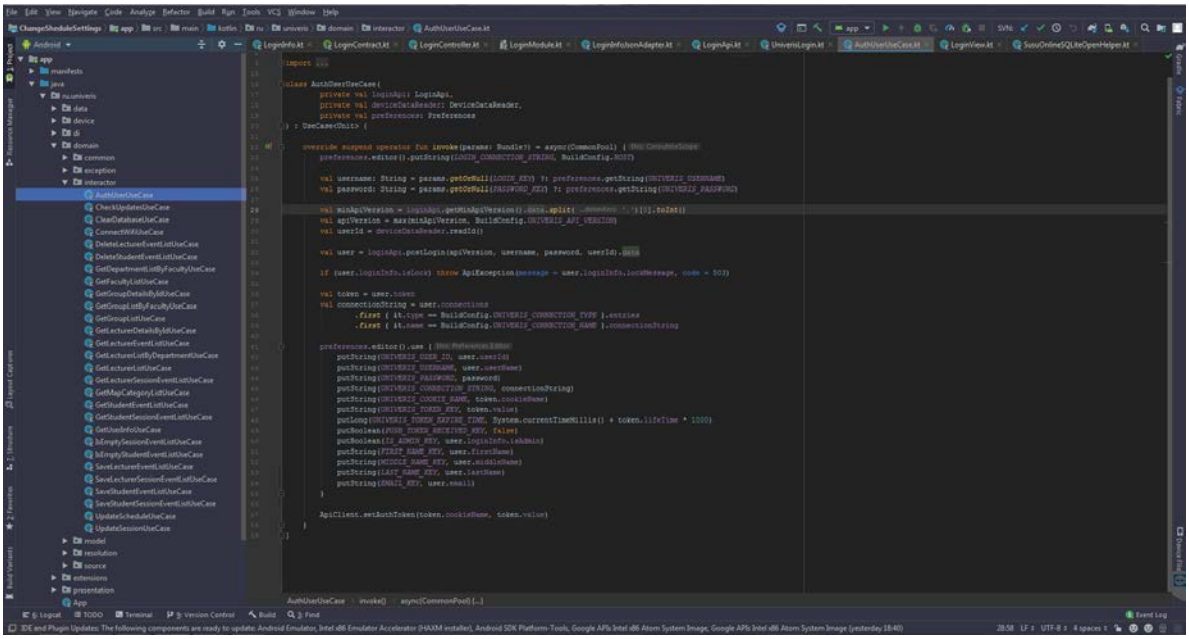

Рисунок 13 – Главный экран среды Android Studio

Приложение было написано на языке Kotlin, этот язык был разработан той же фирмой JetBrains и считается вторым официальным языком программирования мобильных приложений под Android.

Для реализации приложения использовались множества библиотек, все их перечислять не будем, только самые основные:

- com.crashlytics.sdk.android – библиотека для сбора статистики сбоев приложения;
- io.fabric.tools – плагин для работы с Crashlytics;

- com.github.salomonbrys.kodein – это библиотека для реализации внедрения зависимостей (DI);
- com.google.android.gms – библиотека для взаимодействия с сервисом Google Maps;
- org.jetbrains.anko – вспомогательная библиотека для работы с СУБД SQLite;
- com.bluelinelabs:conductor – библиотека, позволяющая писать разметку на языке Kotlin, а не в формате XML, как по умолчанию;
- com.squareup.okhttp3 – библиотека для работы с HTTP-запросами;
- com.mikepenz:iconics-core – библиотека иконок для меню;
- com.android.tools.build:gradle – мощный компоновщик для сборки приложения и формирования APK;
- com.orhanobut:hawk – библиотека шифрования строк;
- com.squareup.moshi:moshi-kotlin – библиотека для преобразования JSON в Java.

Мобильное приложение было успешно реализовано и внедрено в магазин приложений Google Play, с помощью сайта для разработчиков Google Play Console (Рисунок 14) и названо SUSU-online (ЮУрГУ-онлайн).

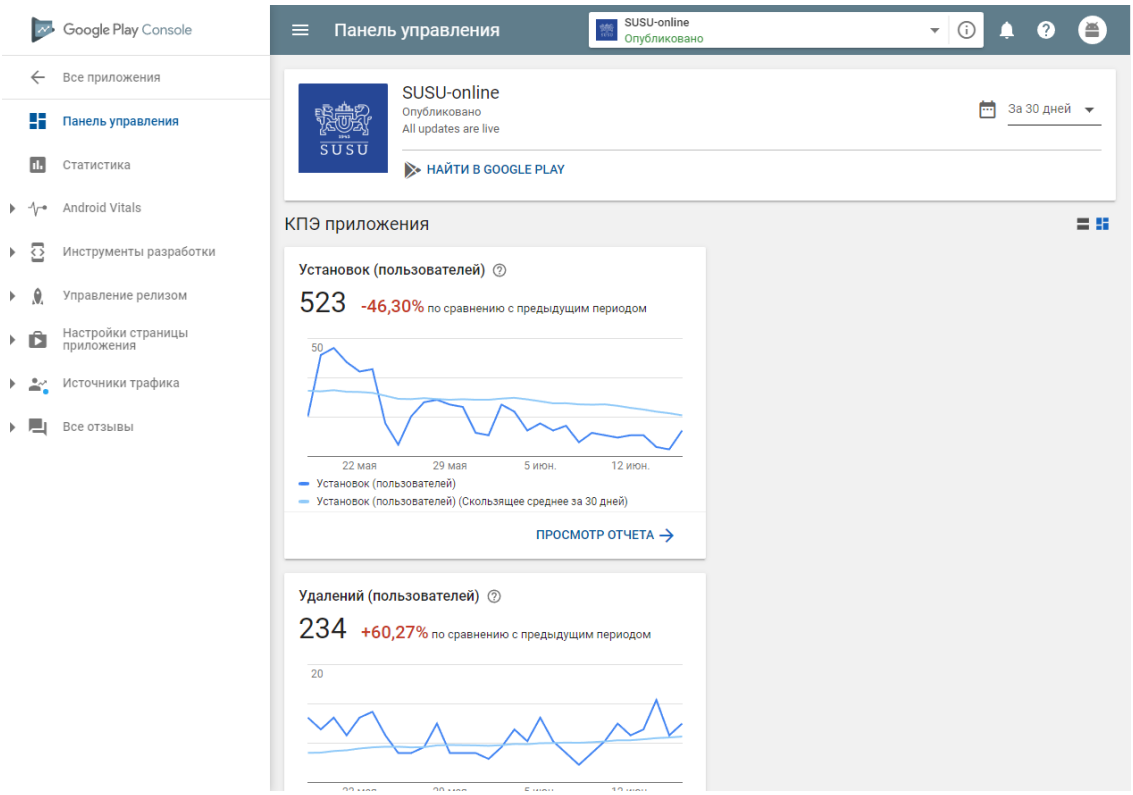

Рисунок 14 – Google Play Console – панель управления

Как видно из графика, за 30 дней уже более пятисот новых пользователей.

Приложение уже имеет следующий функционал (Рисунок 15 – Рисунок 21):

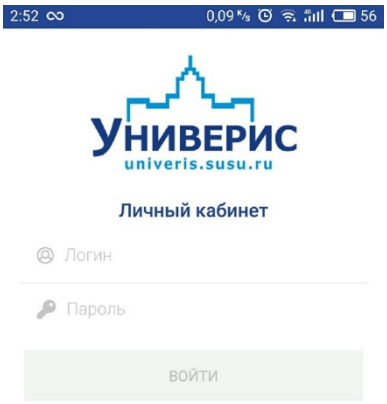

Рисунок 15 – Авторизация

Пользователь уже может зайти в приложение, авторизоваться, ему автоматически загрузится расписание его группы или занятий (если он студент или преподаватель).

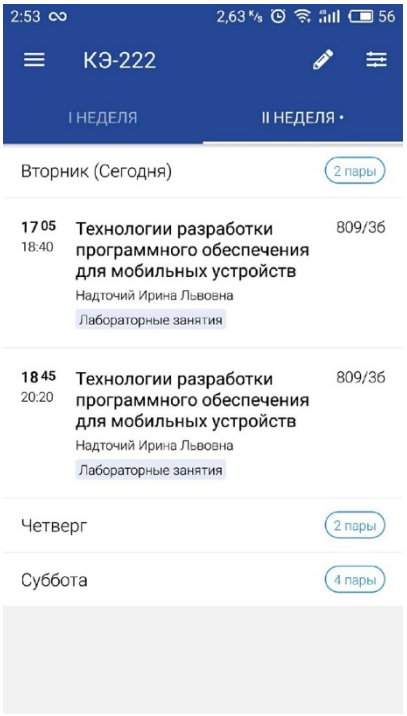

Рисунок 16 – Просмотр расписания занятий

На Рисунок 16 представлено окно с расписанием занятий.

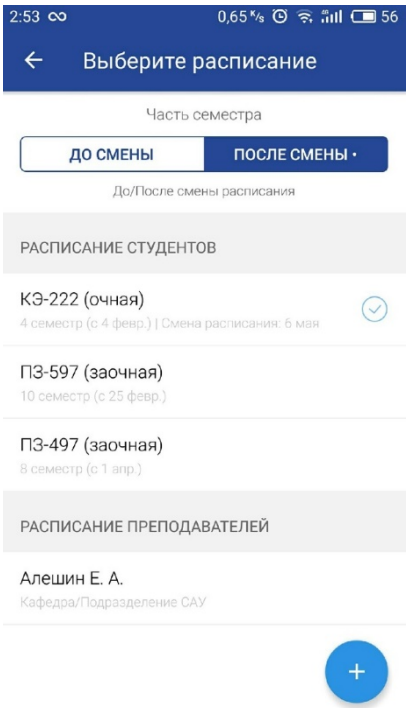

Рисунок 17 – Загрузка расписания

Также он может загрузить любое другое расписание.

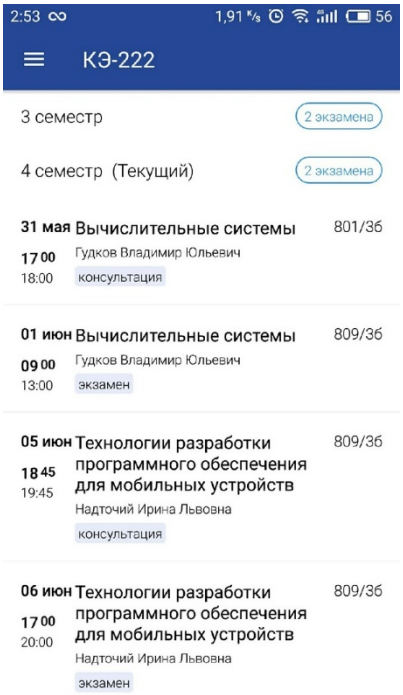

Рисунок 18 – Просмотр расписания сессии

Пользователь может просмотреть расписание сессии, если оно уже составлено.

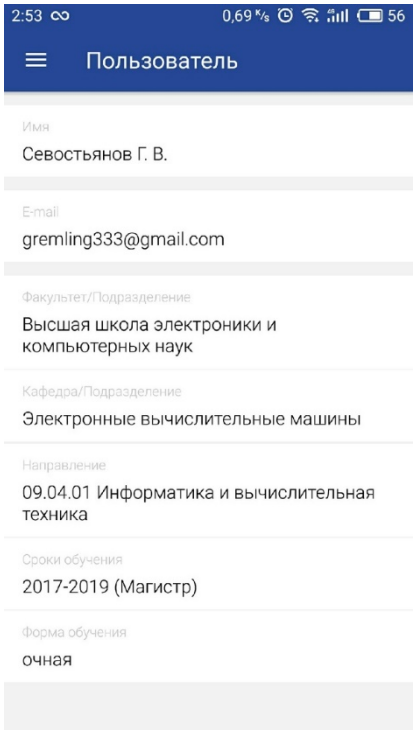

Рисунок 19 – Информация о пользователе

В разделе Пользователь предоставляется возможность просмотреть информацию о себе.

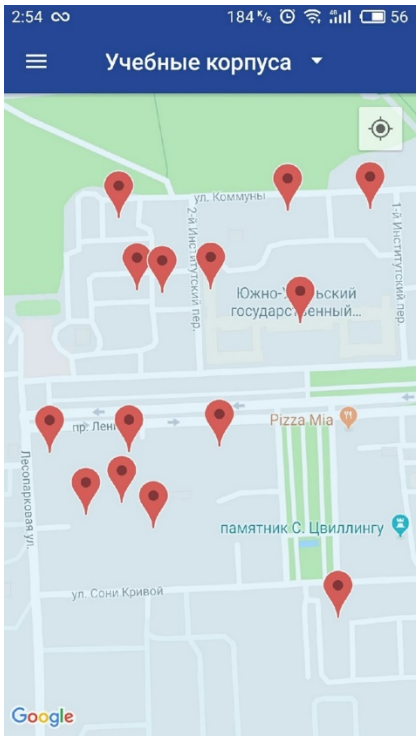

Рисунок 20 – Просмотр карты кампуса

Открыть карту кампуса и даже определить свое местоположение на ней.

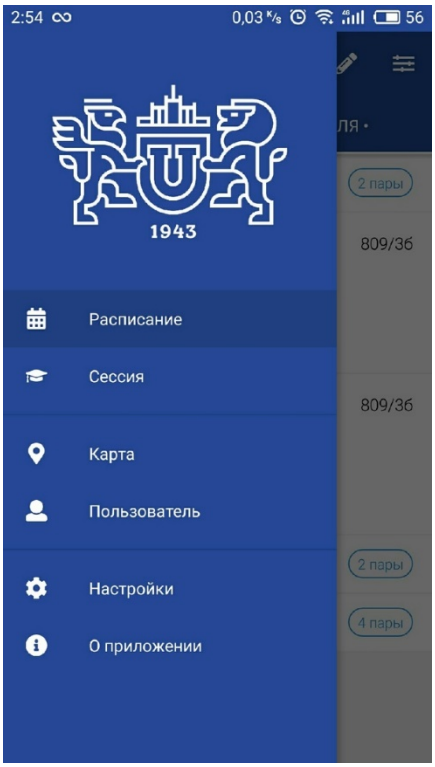

Рисунок 21 – Удобное меню навигации

Все это сопровождается удобной навигацией через меню сбоку.

5. ТЕСТИРОВАНИЕ

5.1. МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

Приложение на Android тестировалось автоматически с помощью внутреннего тестирования Google Play Console. Данный сервис составляет отчет о тестирование[46] после того, как установочный файл APK[47] был загружен в раздел Версии приложения.

Открытое, закрытое и внутреннее тестирование версий APK-файлов или наборов App Bundle позволяет выявлять проблемы, которые возникают в приложении на разных моделях устройств с разными версиями Android.

В отчетах о тестировании в Play Console представлена информация о таких проблемах, как:

- проблемы со стабильностью;
- проблемы совместимости с Android;
- проблемы с производительностью;
- проблемы с доступностью;
- уязвимости.

Принцип тестирования

После загрузки и публикации APK-файла или набора App Bundle для тестирования приложение запускается на тестовых устройствах и сканируется в течение нескольких минут. При этом в приложении автоматически выполняются базовые действия, такие как ввод текста, нажатие и пролистывание.

Результаты проверки появляются в разделе Отчет о тестировании в Play Console.

5.2. ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

Тестирование проходит в несколько этапов. В начале собираются общие сведения о APK файле (Рисунок 22).

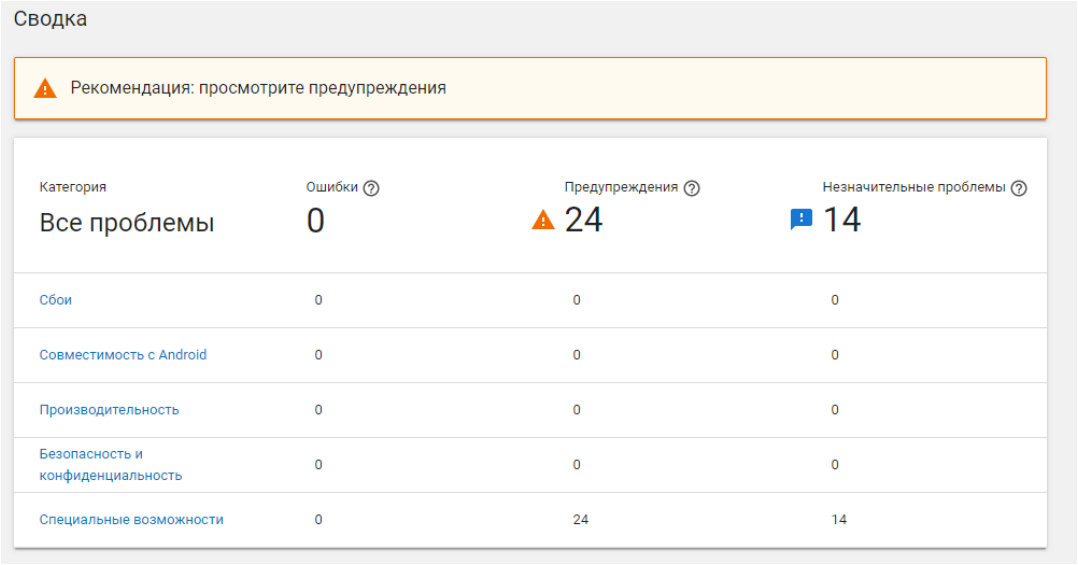

Рисунок 22 – Общие сведения

Затем проходит проверка на стабильность различных устройств (Рисунок 23).

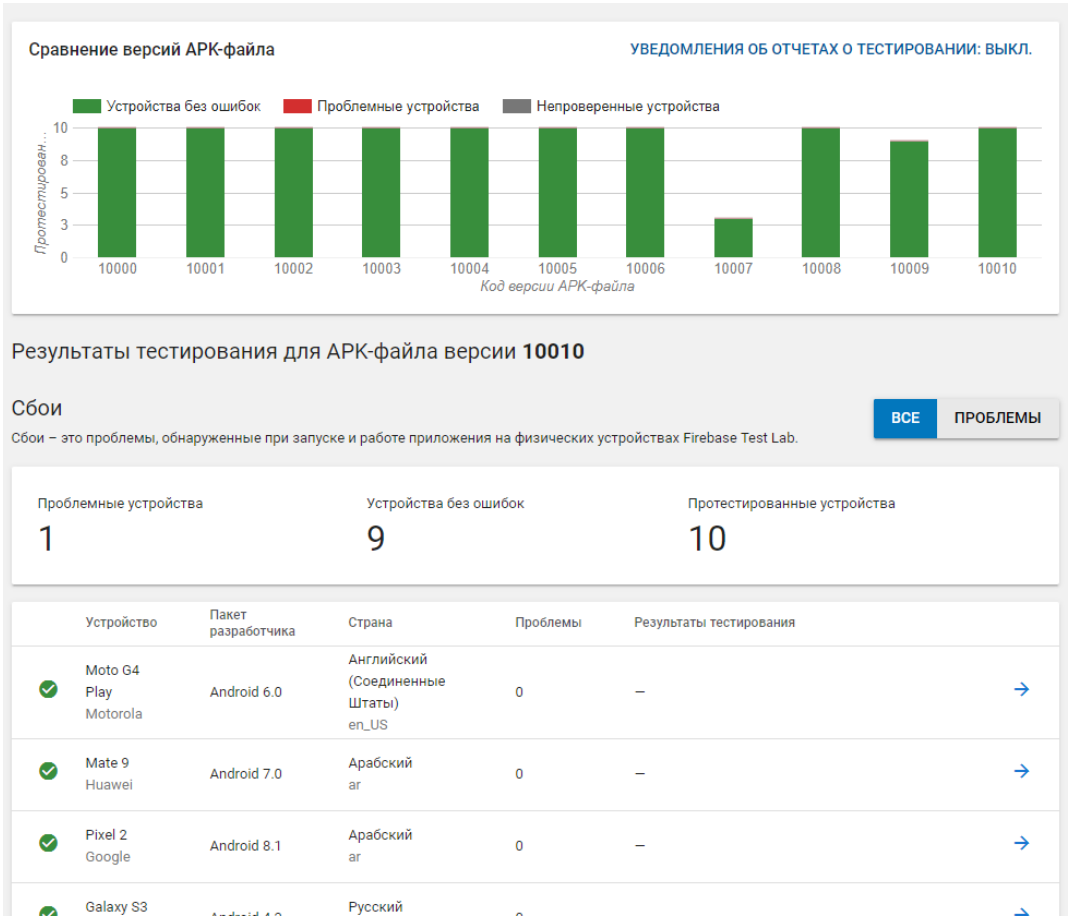

Рисунок 23 – Стабильность

Следующим шагом идет оценка эффективности на различных устройствах (Рисунок 24).

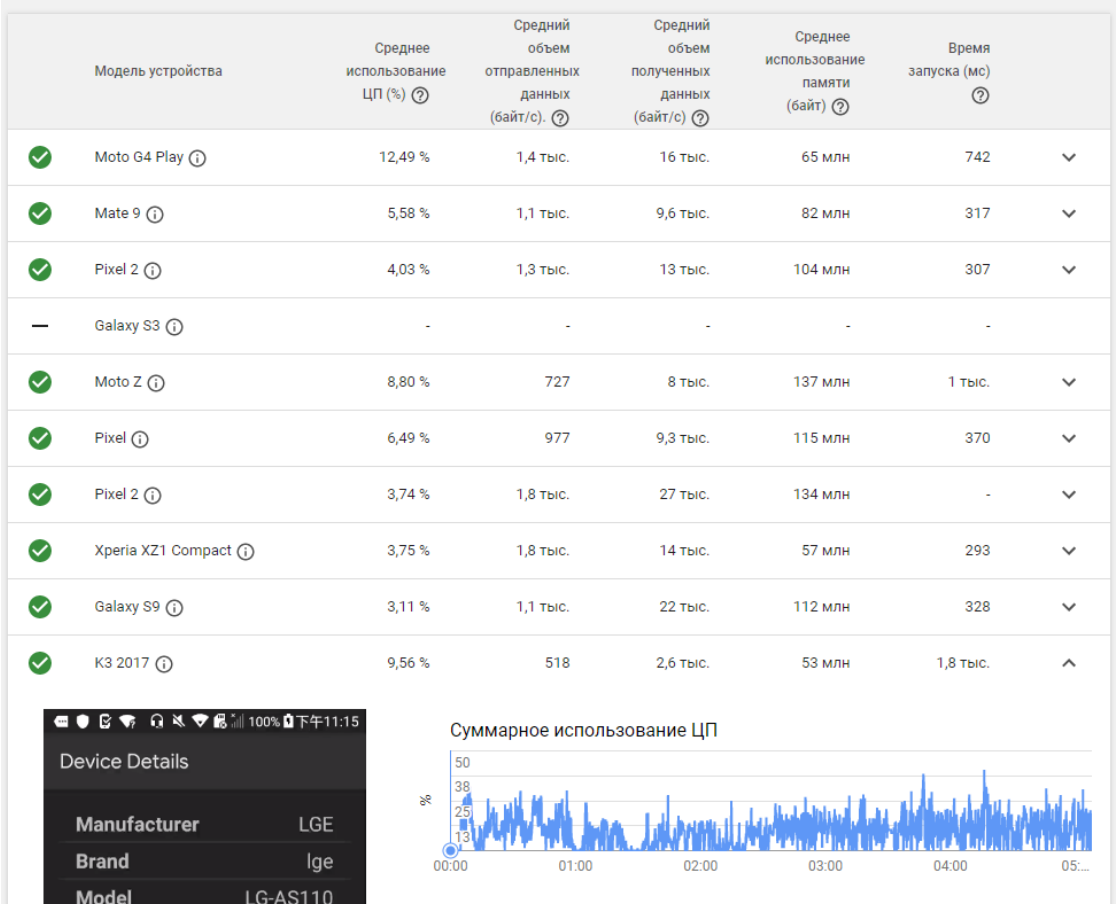

Рисунок 24 – Оценка эффективности

Последний шаг – это оценка доступности, здесь система анализирует качество отображения элементов на экране (яркость, размеры и т. д.) (Рисунок 25).

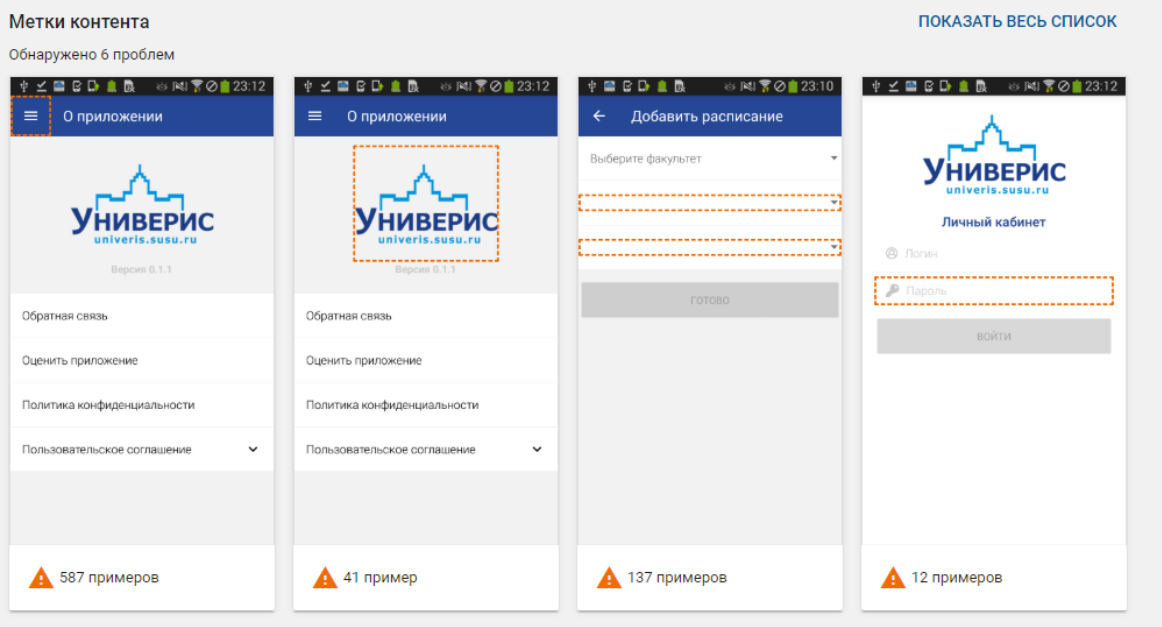

Рисунок 25 – Доступность

Из всего вышесказанного, можно сделать вывод, что внутреннее тестирование Google Play Console прекрасно подойдет для тех, у кого ограничен доступ к различным устройствам, а также удобно, что все происходит автоматически без участия программиста, что экономит ему время.

Стоит отметить, что недостатки все равно присутствуют, например, если в приложение нельзя войти без авторизации, такое тестирование далеко не продвинется. Этот недостаток компенсируется тем, что можно настроить тестировщик на ввод тестовой учетной записи, тогда тесты смогут пройти в полной мере, но все равно не везде, система должна определять компоненты на экране, чтобы ими пользоваться, для этого в коде программы на каждый отображаемый элемент необходимо ставить метку идентификатор.

Для ручного тестирования использовались обычные сценарии взаимодействия.

6. ЗАКЛЮЧЕНИЕ

В представленной работе были разобраны все основные темы, связанные с проектированием, разработкой и вводом в эксплуатацию мобильного приложения для КИАС Универис. Были выполнены следующие задачи:

1. Изучен опыт разработок в области создания мобильных приложений для информационных систем вузов.
2. Проанализированы технологии, которые будут использоваться при разработке приложения, определить их преимущества и недостатки.
3. На основе полученных данных, разработано и реализовано мобильное приложение для КИАС Универис.
4. Проанализированы полученные результаты и определена эффективность работы приложения.

Реализация мобильного приложения была выполнена частично, реализовано приложение только под Android. с учетом следующих этапов развития системы: планирования, разработки и внедрения.

На стадии разработки мобильного приложения возникли проблемы совместимости стиля программирования языка Kotlin и C#, так как в языке Kotlin стиль написания – camelCase[48], а в C# – PascalCase[48] из-за этого были проблемы в адаптере и десериализации из формата JSON. Однако, использование кастомного адаптера, позволило разрешить эту проблему.

По итогам работы, можно сделать следующий вывод:

- построена архитектура мобильных приложений на основе MVC;
- создано реализованное и внедренное в магазин приложение Google Play мобильное приложение под Android, которое получило название SUSU-online;
- организована авторизация приложения;

- организована автоматическая загрузка требуемой информации о пользователе;
- загрузка и обновление расписания занятий;
- просмотр расписания сессии;
- были разработаны сервисы для взаимодействия с базой данных КИАС Универис и сервис авторизации.

Данное приложение будет удобно для пользователей, привыкших работать мобильно, такие как студенты и преподаватели.

В дальнейшем планируется закончить реализацию приложения на iOS. В оба приложений добавить следующий функционал:

- подключение к сети Wi-Fi ФГАОУ ВО «ЮУрГУ (НИУ)» через мобильное приложение, используя аккаунт КИАС Универис
- отображение интерактивной карты аудиторий и прокладка маршрута к ним
- генерацию штрих-кодов в приложение для пропуска в общежития
- электронный аналог читательского билет в мобильном приложении.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Презентация КИАС Универис. – http://www.univeris.susu.ru/Documents/Forms/AllItems.aspx. Дата обращения: 18.06.2019.
2. Южно-Уральский государственный университет. – https://www.susu.ru/ru Дата обращения: 18.06.2019.
3. Google Play. – https://play.google.com/store. Дата обращения: 18.06.2019.
4. iTunes – Самые популярные бесплатные приложения в App Store – Apple (RU). – https://www.apple.com/ru/itunes/charts/free-apps. Дата обращения: 18.06.2019.
5. Календарь знаменательных дат и событий Южно-Уральского государственного университета. – https://is.gd/PeqC9t. Дата обращения: 18.06.2019.
6. Firebird. – http://www.ibase.ru/firebird. Дата обращения: 18.06.2019.
7. Lixux. – https://www.securitylab.ru/news/tags/Linux. Дата обращения: 18.06.2019.
8. Delphi (язык программирования) — Национальная библиотека им. Н. Э. Баумана. – https://is.gd/LBQCrm. Дата обращения: 18.06.2019.
9. Статистика интернета 2017-2018 в мире и в России. – https://www.web-canape.ru/business/internet-2017-2018-v-mire-i-v-rossii-statistika-i-trendy. Дата обращения: 18.06.2019.
10. Приложения в Google Play – Расписание Сибстрин. – https://play.google.com/store/apps/details?id=burov.sibstrin. Дата обращения: 18.06.2019.
11. Studify – расписание занятий для студентов и вузов. – https://studify.ru. Дата обращения: 18.06.2019.

12. Приложения в Google Play – СтудЖурнал – Расписание занятий. – https://play.google.com/store/apps/details?id=com.romansytnyk.studentstudio. Дата обращения: 18.06.2019.
13. Приложения в Google Play – Skedy – расписание занятий. – https://play.google.com/store/apps/details?id=ru.skedy. Дата обращения: 18.06.2019.
14. Приложения в Google Play – Университет ИТМО. – https://play.google.com/store/apps/details?id=ru.ifmo.main. Дата обращения: 18.06.2019.
15. Приложения в Google Play – Студент СФУ. – https://play.google.com/store/apps/details?id=ru.appril.sfu. Дата обращения: 18.06.2019.
16. Таймлайн – Викисловарь. – https://is.gd/wCNzd0. Дата обращения: 18.06.2019.
17. На чём писать кроссплатформенные приложения — Блог Live Typing. – https://livetyping.com/ru/blog/na-chem-pisat-krossplatformennye-prilozhenija. Дата обращения: 18.06.2019.
18. About Objective-C. – https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html. Дата обращения: 18.06.2019.
19. Swift – Apple (RU). – https://www.apple.com/ru/swift. Дата обращения: 18.06.2019.
20. Java – Объектно-ориентированный язык программирования / Хабр. – https://habr.com/ru/hub/java. Дата обращения: 18.06.2019.
21. Kotlin Programming Language – https://kotlinlang.org. Дата обращения: 18.06.2019.

22. Обзор языка C# – руководство по C#. – https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp. Дата обращения: 18.06.2019.
23. Facebook – Выполните вход или зарегистрируйтесь. – https://www.facebook.com. Дата обращения: 18.06.2019.
24. Что такое API в веб-приложениях и зачем он нужен | статьи о программировании mkdev. – https://mkdev.me/posts/chto-takoe-api-v-veb-prilozheniyah-i-zachem-on-nuzhen. Дата обращения: 18.06.2019.
25. Архитектура REST / Хабр. – https://habr.com/ru/post/38730. Дата обращения: 18.06.2019.
26. ASP.NET | Open-source web framework for .NET. – https://dotnet.microsoft.com/apps/aspnet. Дата обращения: 18.06.2019.
27. Что такое Windows Communication Foundation | Microsoft Docs. – https://docs.microsoft.com/ru-ru/dotnet/framework/wcf/whats-wcf. Дата обращения: 18.06.2019.
28. ASP.NET Web APIs | Rest API's with .NET and C#. – https://dotnet.microsoft.com/apps/aspnet/apis. Дата обращения: 18.06.2019.
29. WCF и ASP.NET Web API | Microsoft Docs. – https://docs.microsoft.com/ru-ru/dotnet/framework/wcf/wcf-and-aspnet-web-api. Дата обращения: 18.06.2019.
30. GitHub – SignalR/SignalR: Incredibly simple real-time web for .NET. – https://github.com/SignalR/SignalR. Дата обращения: 18.06.2019.
31. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных / DEVACADEMY. – https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql. Дата обращения: 18.06.2019.
32. Главные задачи нового языка программирования Apple Swift – скорость и простота разработки. – https://3dnews.ru/821428. Дата обращения: 18.06.2019.

33. Kotlin – конкурент Java и Scala | Открытые системы. СУБД | Издательство «Открытые системы». – https://www.osp.ru/os/2011/07/13010422. Дата обращения: 18.06.2019.
34. The Scala Programming Language. – https://www.scala-lang.org. Дата обращения: 18.06.2019.
35. IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. – https://www.jetbrains.com/idea. Дата обращения: 18.06.2019.
36. Welcome! – The Apache HTTP Server Project. JetBrains. – https://httpd.apache.org. Дата обращения: 18.06.2019.
37. JetBrains — Ведущий мировой производитель профессиональных средств разработки JetBrains. – https://jetbrains.ru. Дата обращения: 18.06.2019.
38. JSON. – https://www.json.org/json-ru.html. Дата обращения: 18.06.2019.
39. Google Maps Platform | Google Developers. – https://developers.google.com/maps/documentation. Дата обращения: 18.06.2019.
40. Как работает SSL-сертификат. Принцип работы шифрования | SSL.com.ua. – https://ssl.com.ua/info/how-ssl-works. Дата обращения: 18.06.2019.
41. Джесс Чедвик и др. ASP.NET MVC 4: разработка реальных веб-приложений с помощью ASP.NET MVC = Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC. — М.: «Вильямс», 2013. — 432 с. — ISBN 978-5-8459-1841-3.
42. Фабричный метод. – https://refactoring.guru/ru/design-patterns/factory-method. Дата обращения: 18.06.2019.
43. Что такое XML. – https://msiter.ru/tutorials/uchebnik-xml-dlya-nachinayushchih/chto-takoe-xml. Дата обращения: 18.06.2019.

44. SQLiteDatabase | Android Developers. – https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html. Дата обращения: 18.06.2019.
45. Meet Android Studio | Android Developers. – https://developer.android.com/studio/intro. Дата обращения: 18.06.2019.
46. Отчеты о тестировании – Справка – Play Console. – https://support.google.com/googleplay/android-developer/answer/7002270#sources. Дата обращения: 18.06.2019.
47. Что такое APK-файлы на Android и зачем они нужны? | AndroidLime. – https://androidlime.ru/apk-files. Дата обращения: 18.06.2019.
48. Стили написания составных слов в программировании | Techrocks. – https://techrocks.ru/2018/08/09/most-common-programming-case-types. Дата обращения: 18.06.2019.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД LOGINAPI.KT

Листинг А.1 - LOGINAPI.KT

```
package ru.univeris.data.login

import android.os.Build
import com.squareup.moshi.Moshi
import okhttp3.OkHttpClient
import ru.univeris.BuildConfig
import ru.univeris.data.api.ApiClient
import ru.univeris.data.api.ApiResponse
import ru.univeris.data.api.RequestConfig
import ru.univeris.data.api.RequestMethod
import ru.univeris.domain.common.Preferences
import ru.univeris.domain.model.UniverisLogin
import ru.univeris.extensions.decrypt
import ru.univeris.extensions.generateKey

// Класс LoginApi отвечает за формирование запросов к сервису данных
class LoginApi(baseUrlName: String, client: OkHttpClient, moshi: Moshi, preferences:
Preferences) :
        // Базовый класс LoginApi от него наследуются все API классы
        ApiClient(baseUrlName, client, moshi, preferences) {
    // Функция getMinApiVersion возвращает минимальную версию сервиса
    fun getMinApiVersion(): ApiResponse<String> = request(
        requestConfig = RequestConfig(
                method = RequestMethod.GET,
                path = "/ver",
                headers = jsonHeaders
            )
    )
    // Функция postLogin отправляет логин и пароль на сервис для авторизации, пароль
отправляется в зашифрованном виде
    fun postLogin(apiVersion: Int, username: String, password: String, userId: String):
ApiResponse<UniverisLogin> = request(
```

```
requestConfig = RequestConfig(
                method = RequestMethod.POST,
                 path                                                                                    =
"/v$apiVersion/User/$username/${BuildConfig.UNIVERIS_APPLICATION_NAME}/validate",
                 headers = jsonHeaders
            ),
        body = mutableMapOf<String, Any>().apply {
                 put("Password", password)
                 put("ModuleSecret",
BuildConfig.UNIVERIS_MODULE_SECRET.decrypt(generateKey(BuildConfig.PASSWORD)))
                 put("Identity", "${Build.MANUFACTURER} ${Build.MODEL}\\$userId")
            }
    )
}
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД AUTHUSERUSECASE.KT

Листинг В.1 - AUTHUSERUSECASE.KT

```
package ru.univeris.domain.interactor

import android.os.Bundle
import kotlinx.coroutines.experimental.CommonPool
import kotlinx.coroutines.experimental.async
import ru.univeris.*
import ru.univeris.data.api.ApiClient
import ru.univeris.data.login.LoginApi
import ru.univeris.domain.common.Preferences
import ru.univeris.domain.source.DeviceDataReader
import ru.univeris.domain.common.UseCase
import ru.univeris.domain.exception.ApiException
import ru.univeris.extensions.getOrNull
import kotlin.math.max

// Класс AuthUserUseCase содержит всего одну функцию, которая выполняет набор операций такие как считывание с экрана авторизации строк логина и пароли, получение минимальной версии сервиса данных, считывания идентификатора устройства, отправка логина и пароля в запросе на авторизацию, из ответ на запрос считывается токен, адрес подключения, а также некоторая информация о пользователе записывается в память устройства
class AuthUserUseCase(
        private val loginApi: LoginApi,
        private val deviceDataReader: DeviceDataReader,
        private val preferences: Preferences
) : UseCase<Unit> {

    override suspend operator fun invoke(params: Bundle?) = async(CommonPool) {
        preferences.editor().putString(LOGIN_CONNECTION_STRING, BuildConfig.HOST)

        val username: String = params.getOrNull(LOGIN_KEY) ?:
preferences.getString(UNIVERIS_USERNAME)
        val password: String = params.getOrNull(PASSWORD_KEY) ?:
preferences.getString(UNIVERIS_PASSWORD)
```

```
        val minApiVersion = loginApi.getMinApiVersion().data.split('.')[0].toInt()
        val apiVersion = max(minApiVersion, BuildConfig.UNIVERIS_API_VERSION)
        val userId = deviceDataReader.readId()

        val user = loginApi.postLogin(apiVersion, username, password, userId).data

        if (user.loginInfo.isLock) throw ApiException(message = user.loginInfo.lockMessage, code = 503)

        val token = user.token
        val connectionString = user.connections
                .first { it.type == BuildConfig.UNIVERIS_CONNECTION_TYPE }.entries
                .first { it.name == BuildConfig.UNIVERIS_CONNECTION_NAME }.connectionString

        preferences.editor().use {
            putString(UNIVERIS_USER_ID, user.userId)
            putString(UNIVERIS_USERNAME, user.userName)
            putString(UNIVERIS_PASSWORD, password)
            putString(UNIVERIS_CONNECTION_STRING, connectionString)
            putString(UNIVERIS_COOKIE_NAME, token.cookieName)
            putString(UNIVERIS_TOKEN_KEY, token.value)
            putLong(UNIVERIS_TOKEN_EXPIRE_TIME, System.currentTimeMillis() + token.lifeTime * 1000)
            putBoolean(PUSH_TOKEN_RECEIVED_KEY, false)
            putBoolean(IS_ADMIN_KEY, user.loginInfo.isAdmin)
            putString(FIRST_NAME_KEY, user.firstName)
            putString(MIDDLE_NAME_KEY, user.middleName)
            putString(LAST_NAME_KEY, user.lastName)
            putString(EMAIL_KEY, user.email)
        }

        ApiClient.setAuthToken(token.cookieName, token.value)
    }
}
```

ПРИЛОЖЕНИЕ С

ИСХОДНЫЙ КОД LOGININFOJSONADAPTER.KT

Листинг С.1 - LOGININFOJSONADAPTER.KT

```
package ru.univeris.data.login

import com.squareup.moshi.JsonAdapter
import com.squareup.moshi.JsonReader
import com.squareup.moshi.JsonWriter
import ru.univeris.domain.model.LoginInfo

// Класс LoginInfoJsonAdapter содержит функционал для преобразования данных из формата JSON в модель представления и обратно
class LoginInfoJsonAdapter : JsonAdapter<LoginInfo>() {
    // Функция fromJson возвращает модель представления LoginInfo из строки в JSON формате
    override fun fromJson(reader: JsonReader): LoginInfo {
        var isAdmin = false
        var isLock = false
        var lockMessage: String? = null
        var hostAddress = ""
        reader.beginObject()
        while (reader.hasNext()) {
            when (reader.nextName()) {
                IS_ADMIN -> isAdmin = reader.nextBoolean()
                IS_LOCK -> isLock = reader.nextBoolean()
                LOCK_MESSAGE -> lockMessage = if (reader.peek() == JsonReader.Token.STRING) reader.nextString() else reader.nextNull()
                HOST_ADDRESS -> hostAddress = reader.nextString()
                else -> reader.skipValue()
            }
        }
        reader.endObject()

        return LoginInfo(isAdmin, isLock, lockMessage, hostAddress)
    }
    // Функция toJson возвращает строку в JSON формате из модели представления LoginInfo
```

```
    override fun toJson(writer: JsonWriter, value: LoginInfo?) {
```



```
       writer.beginObject()
        writer.name(IS_ADMIN).value(value?.isAdmin ?: false)
        writer.name(IS_LOCK).value(value?.isLock ?: false)
        writer.name(LOCK_MESSAGE).value(value?.lockMessage ?: "")
        writer.name(HOST_ADDRESS).value(value?.hostAddress ?: "")
        writer.endObject()
    }
   // Вспомогательный объект для считывания сериализованных в формат JSON из объекта, написанного на языке C#
    companion object {
        private const val IS_ADMIN = "IsAdmin"
        private const val IS_LOCK = "IsLock"
        private const val LOCK_MESSAGE = "LockMessage"
        private const val HOST_ADDRESS = "HostAddress"
    }
}
```

ПРИЛОЖЕНИЕ D

ИСХОДНЫЙ КОД LOGININFO.KT

Листинг D.1 – LOGININFO.KT

```
package ru.univeris.domain.model
// Класс LoginInfo – это модель представления для сущности LoginInfo, он содержит такие поля как проверка на администратора, проверка на доступ, сообщения закрытия доступа и адрес подключения
data class LoginInfo(
        val isAdmin: Boolean,
        val isLock: Boolean,
        val lockMessage: String?,
        val hostAddress: String
)
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД LOGINCONTRACT.KT

Листинг Е.1 – LOGINCONTRACT.KT

```
package ru.univeris.presentation.screen.login

import ru.univeris.presentation.common.BaseContract
import ru.univeris.presentation.resolution.UiResolution
// Интерфейс, от которого наследуются классы View и Controller, интерфейс содержит абстрактные методы, которые требуется переопределить в реализации этих классов
interface LoginContract : BaseContract {

    interface View : BaseContract.View<Controller> {

        val uiResolution: UiResolution
        // Функция showLoading показывает экран загрузки
        fun showLoading()
        // Функция hideLoading скрывает экран загрузки
        fun hideLoading()
    }

    interface Controller : BaseContract.Controller<View> {
       // Функция onSendPressed отправляет строки логина и пароля
        fun onSendPressed(login: String, password: String)
    }
}
```

ПРИЛОЖЕНИЕ F

ИСХОДНЫЙ КОД LOGINCONTROLLER.KT

Листинг F.1 – LOGINCONTROLLER.KT

```
package ru.univeris.presentation.screen.login

import android.os.Bundle
import android.view.ViewGroup
import com.crashlytics.android.Crashlytics
import com.github.salomonbrys.kodein.instance
import com.github.salomonbrys.kodein.with
import kotlinx.coroutines.experimental.android.UI
import ru.univeris.*
import ru.univeris.di.*
import ru.univeris.domain.common.Preferences
import ru.univeris.domain.common.UseCase
import ru.univeris.domain.model.Updates
import ru.univeris.domain.model.isNotEmpty
import ru.univeris.domain.resolution.launchWithErrorHandler
import ru.univeris.extensions.bundle
import ru.univeris.presentation.HOME
import ru.univeris.presentation.common.BaseController
import ru.univeris.presentation.common.ControllerRegistry
import ru.univeris.presentation.widget.updateScheduleWidgets
// Класс LoginController обрабатывает данные полученные из UseCase и отправляет во View
class LoginController(bundle: Bundle) : BaseController<LoginContract.View>(bundle),
    LoginContract.Controller {

    override val view: LoginContract.View by with(this).instance()
    private val univerisAuth: UseCase<Unit> by instance(AUTH)
    private val getUserInfo: UseCase<Updates> by instance(GET_USER_INFO)
    private val updateSchedule: UseCase<Unit> by instance(UPDATE_SCHEDULE)
    private val updateSession: UseCase<Updates> by instance(UPDATE_SESSION)
    private val controllerRegistry: ControllerRegistry by instance()
    private val publicPreferences: Preferences by instance(PUBLIC)
    private val privatePreferences: Preferences by instance(PRIVATE)
```

```
// Функция точка вхождения, выполняется при загрузке экрана и нужна, для того чтоб нарисовать все компоненты View
        override fun createView(container: ViewGroup) = view.createView(container)
// Функция onSendPressed отправляет строки логина и пароля в зашифрованном виде, авторизует пользователя, загружает информацию о пользователе и его расписание, если он есть, в случае ошибке авторизации произваодит откат всех загрузок
    override fun onSendPressed(login: String, password: String) {
            view.showLoading()
            launchWithErrorHandler(UI, view.uiResolution, block = {
                univerisAuth(bundle {
                    putString(LOGIN_KEY, login)
                    putString(PASSWORD_KEY, password)
                }).await()

                val updates = getUserInfo(bundle {
                    putBoolean(NEED_TO_SELECT_SCHEDULE, true)
                }).await()
                if (updates.isNotEmpty()) {
                    updateSchedule(bundle {
                        putParcelableArrayList(LECTURERS_NEED_TO_UPDATED_KEY,
    updates.lecturers.keys.toCollection(ArrayList()))
                        putParcelableArrayList(GROUPS_NEED_TO_UPDATED_KEY,
    updates.groups.keys.toCollection(ArrayList()))
                    }).await()
                }
                updateSession().await()

                val                                isEmptySession                                =
    privatePreferences.getBoolean(SESSION_EVENT_LIST_IS_EMPTY_KEY)
                setMenuItemEnabled(R.id.home_drawer_session, !isEmptySession)

                view.hideLoading()

                activity?.updateScheduleWidgets()
                Crashlytics.setUserName(login)
                replaceControllerWithAnimation(controllerRegistry.get(HOME))
            }, onError = {
```

```
            privatePreferences.editor().use {
                delete(UNIVERIS_USER_ID)
                delete(UNIVERIS_USERNAME)
                 delete(UNIVERIS_PASSWORD)
                delete(UNIVERIS_CONNECTION_STRING)
                delete(UNIVERIS_COOKIE_NAME)
                delete(UNIVERIS_TOKEN_KEY)
                delete(IS_ADMIN_KEY)
            }

            publicPreferences.editor().use {
                delete(CURRENT_SCHEDULE_TYPE_KEY)
                delete(GROUP_ID_KEY)
                delete(LECTURER_ID_KEY)
                delete(IS_STUDENT_KEY)
                delete(STUDENT_KEY)
                delete(INSTRUCTOR_KEY)
                delete(NEXT_CHECK_UPDATE_DATE)
            }

            view.hideLoading()
        })
    }
}
```

ПРИЛОЖЕНИЕ G

ИСХОДНЫЙ КОД LOGINVIEW.KT

Листинг G.1 – LOGINVIEW.KT

```
package ru.univeris.presentation.screen.login

import android.support.v4.view.GravityCompat
import android.text.InputType
import android.text.method.PasswordTransformationMethod
import android.view.Gravity
import android.view.View
import android.view.ViewGroup
import android.view.inputmethod.EditorInfo
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import com.mikepenz.fontawesome_typeface_library.FontAwesome
import com.mikepenz.iconics.IconicsDrawable
import org.jetbrains.anko.backgroundResource
import org.jetbrains.anko.dimen
import org.jetbrains.anko.dip
import org.jetbrains.anko.editText
import org.jetbrains.anko.frameLayout
import org.jetbrains.anko.hintResource
import org.jetbrains.anko.horizontalPadding
import org.jetbrains.anko.imageResource
import org.jetbrains.anko.imageView
import org.jetbrains.anko.lines
import org.jetbrains.anko.matchParent
import org.jetbrains.anko.progressBar
import org.jetbrains.anko.scrollView
import org.jetbrains.anko.sdk15.listeners.onClick
import org.jetbrains.anko.sdk15.listeners.onEditorAction
import org.jetbrains.anko.sdk15.listeners.textChangedListener
import org.jetbrains.anko.singleLine
import org.jetbrains.anko.support.v4.space
import org.jetbrains.anko.textColorResource
```

```
import org.jetbrains.anko.textResource
import org.jetbrains.anko.textSizeDimen
import org.jetbrains.anko.textView
import org.jetbrains.anko.verticalLayout
import org.jetbrains.anko.verticalMargin
import org.jetbrains.anko.view
import org.jetbrains.anko.wrapContent
import ru.univeris.R
import ru.univeris.presentation.extension.accentButton
import ru.univeris.presentation.extension.applyLightFontFamily
import ru.univeris.presentation.extension.applyMediumFontFamily
import ru.univeris.presentation.extension.color
import ru.univeris.presentation.extension.hideKeyboard
import ru.univeris.presentation.extension.hintColorResource
import ru.univeris.presentation.resolution.UiResolution
import ru.univeris.presentation.resolution.UiResolver
import ru.univeris.presentation.screen.home.HomeContract

// Класс LoginView отвечает за построение видимых компонентов на экране авторизации
class LoginView(
        override val controller: LoginContract.Controller,
        uiResolver: UiResolver
) : LoginContract.View, HomeContract.WithDrawerLockedView {

    override val uiResolution: UiResolution = UiResolution(uiResolver)

    private lateinit var loadingLayout: View
    private lateinit var loginEdit: EditText
    private lateinit var passwordEdit: EditText
    private lateinit var sendButton: Button

// Функция точка вхождения, выполняется при загрузке экрана и нужна, для того чтоб
нарисовать все компоненты View
    override fun createView(container: ViewGroup) = container.context.frameLayout {
        backgroundResource = R.color.background
        clipChildren = false
        id = R.id.home_content
```

```
scrollView {
clipChildren = false

  verticalLayout {
      horizontalPadding = dimen(R.dimen.padding_horizontal)

      imageView {
          imageResource = R.drawable.img_univeris_logo
          scaleType = ImageView.ScaleType.CENTER_CROP
      }.lparams(dip(200), dip(160)) {
          gravity = Gravity.CENTER
      }

      textView(R.string.univeris_title) {
          textColorResource = R.color.text_primary
          textSizeDimen = R.dimen.text_big
          applyMediumFontFamily()
      }.lparams {
          gravity = Gravity.CENTER
      }

      space().lparams(matchParent, dip(12))

      editText {
          id = R.id.login_username
          loginEdit = this
          background = null
          gravity = Gravity.CENTER_VERTICAL or GravityCompat.START
          horizontalPadding = dimen(R.dimen.padding_horizontal)
          textColorResource = R.color.text_black
          hintColorResource = R.color.text_gray_light
          hintResource = R.string.univeris_login
          textSizeDimen = R.dimen.text_large
          imeOptions = EditorInfo.IME_ACTION_NEXT
          singleLine = true
          setCompoundDrawablesWithIntrinsicBounds(
              IconicsDrawable(context, FontAwesome.Icon.faw_user_circle)
```

```
                        .color(color(R.color.gray_light))
                    .sizeDp(16), null, null, null)
                compoundDrawablePadding = dip(10)
                applyLightFontFamily()

                textChangedListener {
                    onTextChanged { _, _, _, _ ->
                        updateSendButton()
                    }
                }

                onEditorAction { _, actionId, _ ->
                    if (actionId == EditorInfo.IME_ACTION_NEXT) {
                        passwordEdit.requestFocus()
                        true
                    } else false
                }
            }.lparams(matchParent, dip(40))

            view {
                backgroundResource = R.color.background_dark
            }.lparams(matchParent, dip(1)) {
                verticalMargin = dip(5)
            }

            editText {
                id = R.id.login_password
                passwordEdit = this
                background = null
                gravity = Gravity.CENTER_VERTICAL or GravityCompat.START
                horizontalPadding = dimen(R.dimen.padding_horizontal)
                textColorResource = R.color.text_black
                hintColorResource = R.color.text_gray_light
                hintResource = R.string.univeris_password
                textSizeDimen = R.dimen.text_large
                inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD
```

```
                    transformationMethod = PasswordTransformationMethod.getInstance()
                    imeOptions = EditorInfo.IME_ACTION_DONE
                    lines = 1
                    maxLines = 1
                    setCompoundDrawablesWithIntrinsicBounds(
                            IconicsDrawable(context, FontAwesome.Icon.faw_key)
                                    .color(color(R.color.gray_light))
                                    .sizeDp(18), null, null, null)
                    compoundDrawablePadding = dip(10)
                    applyLightFontFamily()

                    textChangedListener {
                        onTextChanged { _, _, _, _ ->
                            updateSendButton()
                        }
                    }

                    onEditorAction { _, actionId, _ ->
                        if      (actionId      ==      EditorInfo.IME_ACTION_DONE      &&
loginEdit.text.isNotEmpty() && passwordEdit.text.isNotEmpty()) {
                            context.hideKeyboard(this)
                            controller.onSendPressed(loginEdit.text.toString(),
passwordEdit.text.toString())
                            true
                        } else false
                    }
                }.lparams(matchParent, dip(40))

                space().lparams(matchParent, dip(12))

                accentButton {
                    id = R.id.login_button
                    sendButton = this
                    textResource = R.string.univeris_sign_in
                    isEnabled = false

                    onClick {
```

```
            context.hideKeyboard(passwordEdit)
    controller.onSendPressed(loginEdit.text.toString().trim(),
passwordEdit.text.toString())
                    }
                }.lparams(matchParent, dip(60))
            }
        }.lparams(matchParent, wrapContent)

        frameLayout {
            loadingLayout = this
            backgroundResource = R.color.loading_overlay_background
            isClickable = true
            visibility = View.GONE

            progressBar().lparams {
                gravity = Gravity.CENTER
            }
        }.lparams(matchParent, matchParent)
    }
// Функция showLoading показывает экран загрузки
    override fun showLoading() {
        loadingLayout.visibility = View.VISIBLE
    }
// Функция hideLoading скрывает экран загрузки
    override fun hideLoading() {
        loadingLayout.visibility = View.GONE
    }
//  Функция  updateSendButton  активирует  кнопку  отправки  формы  только  если  все  поля
заполнены
    private fun updateSendButton() {
        sendButton.isEnabled          =          loginEdit.text.isNotEmpty()          &&
passwordEdit.text.isNotEmpty()
    }
}
```