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 «Системы автоматического управления»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
д.т.н., профессор
_____/ В.И. Ширяев
«____» _____ 2019 г.

Разработка автоматизированной системы электронных услуг для объектов студенческого городка ЮУрГУ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 09.03.01.2019.140.00 ПЗ ВКР

Консультант
директор студ. городка ЮУрГУ
_____/ Л.Н. Задорина
«____» _____ 2019 г.

Руководитель работы
доцент каф. САУ, к.т.н.
_____/ О.О. Павловская
«____» _____ 2019 г.

Автор работы
студент группы КЭ-451
_____/ С.А. Сукина
«____» _____ 2019 г.

Нормоконтролер
доцент каф. САУ, к.т.н.
_____/ О.О. Павловская
«____» _____ 2019 г.

Челябинск 2019

АННОТАЦИЯ

Сукина С.А. Разработка автоматизированной системы электронных услуг для объектов студенческого городка ЮУрГУ. – Челябинск: ЮУрГУ, ВШ ЭКН; 2019, 50 с., 15 ил., библиогр. список – 23 наим., 14 листов слайдов презентации ф.А4, 1 прил.

В рамках выпускной квалификационной работы разработана и реализована автоматизированная система электронных услуг для объектов студенческого городка ЮУрГУ.

В первой главе проведен сравнительный анализ существующих на рынке автоматизированных систем электронных услуг.

Во второй главе составлено техническое задание на разработку автоматизированной системы электронных услуг для объектов студенческого городка ЮУрГУ.

В третьей главе проведена разработка программного обеспечения автоматизированной системы, реализованной на языках Kotlin, C#.

Спроектирована база данных для хранения информации, которая используется автоматизированной системой электронных услуг.

Автоматизированная система электронных услуг для объектов студенческого городка ЮУрГУ проходит тестирование с целью поиска недостатков перед публикацией на сервисе Google Play Market.

Пояснительная записка к выпускной квалификационной работе оформлена в текстовом редакторе MS Word 2016.

					09.03.01.2019.140.00 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Сукина С.А.			Разработка автоматизированной системы электронных услуг для объектов студенческого городка ЮУрГУ	Лит.	Лист 4 / Листов 50
Провер.		Павловская О.О.				Д	
Н. Контр.		Павловская О.О.				ЮУрГУ Кафедра САУ	
Утверд.		Ширяев В.И.					

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Основу современной мировой экономики составляет конкуренция. Предприятия соперничают за ограниченные ресурсы: сырье, рабочую силу, капитал, рынки сбыта. От того, насколько рационально эти ресурсы используются, зависит конкурентоспособность компании на рынке сбыта и обеспечение необходимого ей для развития размера прибыли.

Размер прибыли компании напрямую определяется объемом продаж и ценой реализации продукции. В рыночных условиях компания не может формировать цену реализации своей продукции путём простого добавления к производственной себестоимости необходимой ей сбытовой наценки, а вынуждена ограничивать предельную цену реализации среднерыночным уровнем цен. Таким образом, конкурентное преимущество получают те производители, себестоимость продукции которых оказывается ниже, чем у конкурентов [21]. Исходя из этого, можно утверждать, что основным инструментом поддержания конкурентоспособности компании на рынке является снижение себестоимости производства продукции.

Известно, как снизить себестоимость продукции промышленного предприятия [20]: увеличить масштаб производства, снизить цены закупки сырья и материалов для производства, снизить производственные затраты. Однако снижение себестоимости продукции обеспечивается, прежде всего, за счет повышения производительности труда. Ведь с ростом производительности сокращается удельный вес зарплаты в структуре себестоимости.

Задача снижения себестоимости и повышения качества продукции актуальна и для компаний, оказывающих услуги населению. Здесь повышение производительности компании трактуется как увеличение объема оказываемых услуг, что в настоящее время достигается за счет внедрения автоматизированных систем электронных услуг.

В последнее время электронные услуги пользуются повышенным спросом у населения. Объяснить это можно просто: автоматизированные системы упрощают механизм оказания услуг (вся информация подается в удобном виде, ее изучение занимает минимум времени), услугу может получить пользователь сети Интернет в любое удобное время. Таким образом, системы электронных услуг — надежные помощники для большого количества людей, которые ценят время.

Что касается списка электронных услуг, которые предоставляются гражданам и компаниям, то их число постоянно растет и сегодня превышает 300. В первую очередь это системы электронных услуг, обеспечивающие автоматизацию деятельности различных государственных и муниципальных организаций. Так, например, од-

ной и самых популярных и востребованных автоматизированных систем электронных услуг в нашей стране является «Портал государственных услуг Российской Федерации». Данная система обеспечивает доступ к сведениям о государственных и муниципальных услугах в Российской Федерации, а также осуществляет предоставление в электронной форме государственных и муниципальных услуг физическим и юридическим лицам [17].

Известно, что услуги в электронном виде могут предоставляться в нескольких формах:

- через интернет-ресурсы;
- через многофункциональные центры;
- через информаты (киоски для предоставления информации);
- по телефону (call-центры) и т.д.

Однако, практически в каждой организации есть услуги, которые необходимо перевести в электронный вид. В том числе и услуги организаций, чья деятельность связана с предоставлением мест временного проживания человека, например, гостиницы, общежития, хостелы и т.д. Несмотря на то, что услуги, предоставляемые гостиницами и общежитиями очень похожи, невозможно адаптировать известные на рынке программные продукты, обеспечивающие автоматизацию оказания услуг предприятий гостиничного бизнеса, на рабочие и студенческие общежития. Причина проста — гостиницы предполагают проживание человека, приехавшего по делам или на отдых, в течение некоторого короткого времени в незнакомом городе. В общежитиях же люди живут в основном несколько лет, во время учебы или работы при невозможности приобрести собственное жилье.

Также следует сказать, что тиражирование уже существующих на рынке систем электронных услуг сдерживается следующими факторами:

- высокая стоимость лицензий и технической поддержки покупных программных изделий (платформ) иностранного производства;
- отсутствие высококвалифицированных кадров для сопровождения и технической поддержки систем электронных услуг, для поддержания в актуальном состоянии общих информационных ресурсов системы.

Эти проблемы следует устранить уже на этапе планирования разработки системы электронных услуг для объектов Студгородка ЮУрГУ. Данная система собственной разработки позволит организовать для студентов не только просмотр новостной ленты, информации о культурно-массовых мероприятиях, информации об общежитиях, но и обеспечит студентов возможностью пользования личным кабинетом. Также разработка данной системы предполагает введение в общежитии системы цифровых пропусков, организацию электронных очередей в прачечную, на услуги сантехника, электрика, слесаря/плотника. Пользователи электронных

услуг, предоставляемых общежитиями – студенты – отдают предпочтение более современным формам: через телефон и ноутбук (ПК).

По назначению и принципам проектирования автоматизированная система электронных услуг является разновидностью автоматизированной системы обработки информации и управления (АСОИУ), поэтому **цель** данной выпускной квалификационной работы – разработать автоматизированную систему электронных услуг для объектов Студгородка ЮУрГУ.

Для достижения цели необходимо решить следующие **задачи**:

1. Провести анализ известных на рынке автоматизированных систем электронных услуг. Определить концепцию системы электронных услуг для объектов студгородка ЮУрГУ.
2. Сформировать техническое задание на разработку системы электронных услуг.
3. Выбрать программно-аппаратные средства.
4. Разработать архитектуру программного обеспечения системы.
5. Собрать информационные ресурсы и организовать хранение данных.
6. Программная реализация системы.

Объект исследований: Система информационного обеспечения жителей студенческого городка ЮУрГУ

Предмет исследований: Система электронных услуг

Методы исследований: Исследования, проводимые в работе, основаны на комплексном использовании знаний в таких областях как: теория информационных систем, теория автоматического управления, объектно-ориентированное программирование, базы и хранилища данных, программирование на языках высокого уровня, иностранный язык, клиент-серверные приложения, оптимизация приложений на языке C#, современные подходы к созданию графических интерфейсов пользователя, распределённые системы контроля версий.

Новизна результатов работы состоит в том, что разработана и реализована в форме мобильного приложения система электронных услуг для жителей объектов студгородка ЮУрГУ, в которой кроме функций предоставления актуальной информации реализована электронная очередь на оказание коммунальных услуг и оцифрованная версия пропусков.

Практическая ценность: автоматизация системы информационного обеспечения жителей студенческого городка ЮУрГУ, обеспечивающая

- упрощение процедур записи на оказание услуг;
- расширение целевой аудитории, улучшение качества информирования о культурно-массовых мероприятиях;
- изменение формы доступа (введение оцифрованной версии пропусков).

1 АНАЛИЗ СУЩЕСТВУЮЩИХ НА РЫНКЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ ЭЛЕКТРОННЫХ УСЛУГ

В настоящее время на рынке существует множество различных сервисов. Чаще всего подобные системы реализуются как:

- мобильные приложения;
- сайты.

В нашем случае стоит рассмотреть мобильные приложения различных отелей и приложения, созданные для студентов. Так как мобильное приложение – это современная и более удобная для пользователей форма реализации системы. Проведем анализ существующих решений, разделив их на две группы:

1. Мобильные приложения для гостиничного бизнеса.
2. Мобильные приложения для студентов.

Проанализируем, какими возможностями обладают существующие системы, и выясним не станет ли наша разработка очередным дубликатом.

1.1 Мобильные приложения для гостиничного бизнеса

Приложение «Radisson Rewards»

Данное приложение создано для компании Radisson Hotel Group (рисунок 1.1). Его основная возможность — бронирование номеров в любом из отелей компании, поиск отелей компании по текущему местоположению, система бонусов. Также предусмотрена система личного кабинета, где пользователь может просмотреть всю необходимую информацию о бронировании, или персональных бонусах.

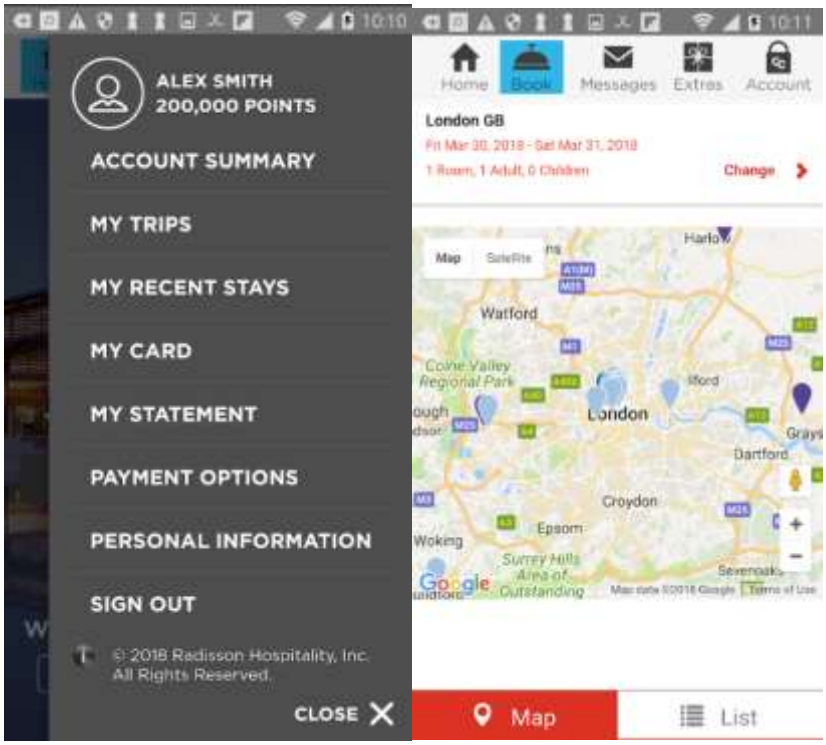

Рисунок 1.1 — Приложение Radisson Rewards

Приложение «Best Western To Go»

Данное приложение создано специально для отелей бренда Best Western Hotels & Resorts (рисунок 1.2). Аналогично предыдущему в данном продукте предусмотрены такие решения как: бронирование любого из действующих отелей, просмотр отелей на карте, проверка личного кабинета и баланса участника BWR. Кроме того, у пользователей есть возможность поиска ресторанов, достопримечательностей и транспорта поблизости с местом проживания.

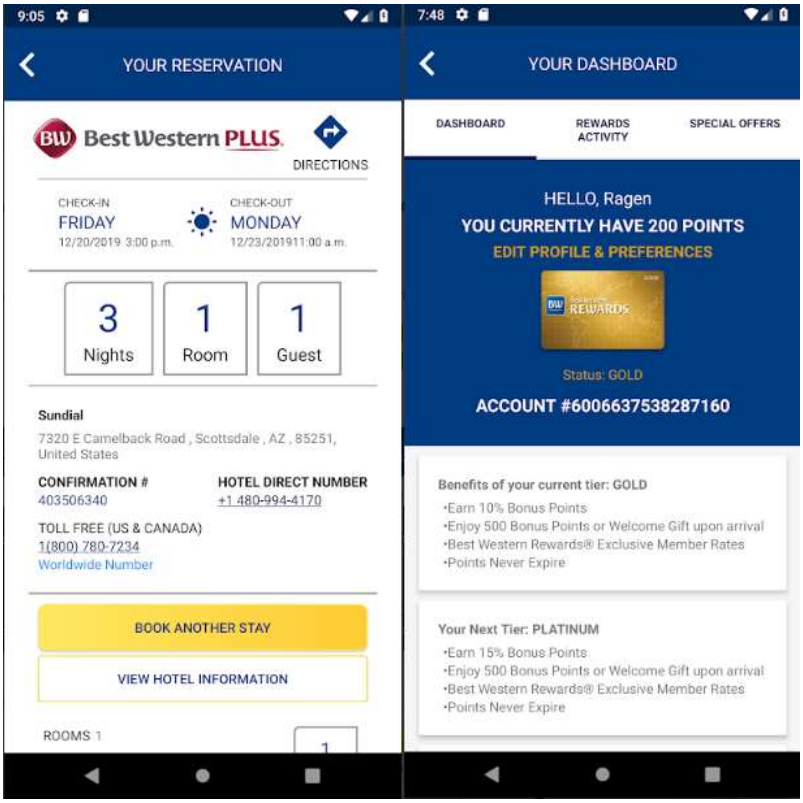

Рисунок 1.2 — Приложение «Best Western To Go»

Приложение «NH Hotel Group – Book your hotel»

Приложение компании NH Hotel Group (рисунок 1.3). Его функционал немного шире, чем у двух предыдущих решений. У пользователя есть возможность совершать звонки с помощью приложения для того, чтобы забронировать столик в одном из некоторых ресторанов или для согласования проведения частного мероприятия или встречи.

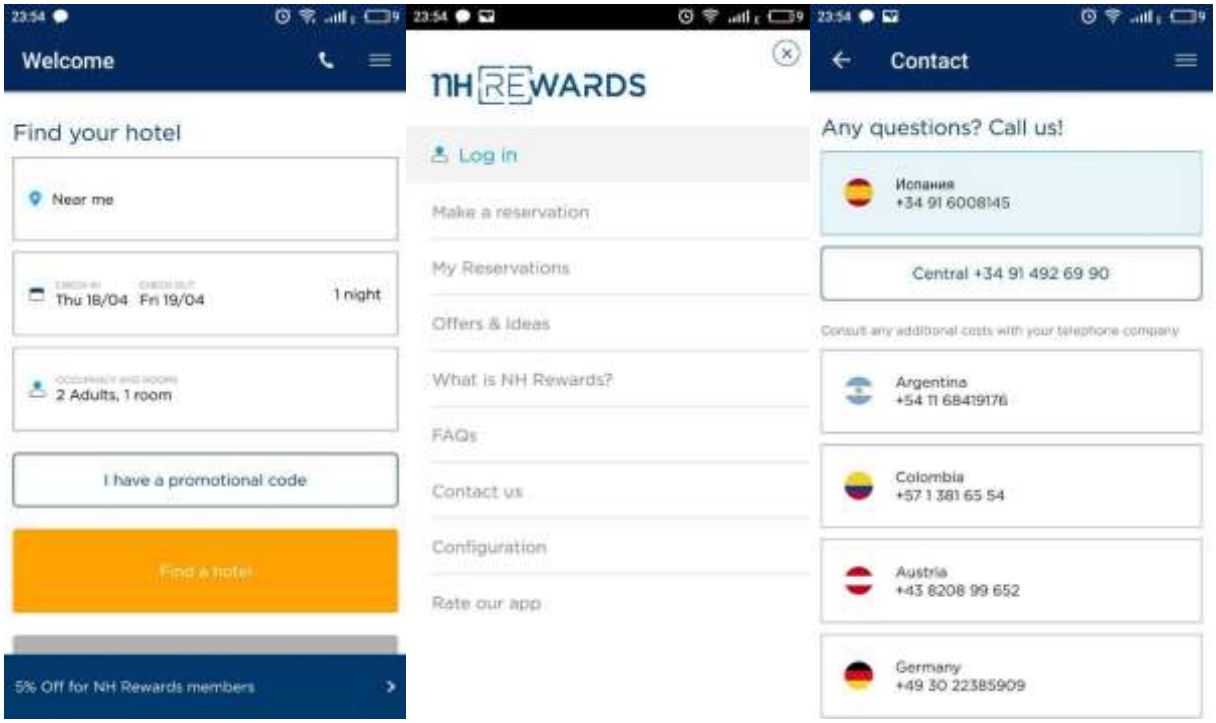

Рисунок 1.3 — Приложение «NH Hotel Group – Book your hotel»

Приложение «Melia – Hotel Bookings & more»

Приложение компании Meliá Hotels International имеет широкий функционал, и создано для посетителей отелей данной компании (рисунок 1.4). В возможности приложения входит: управление бронированием, онлайн регистрация в отеле, цифровой ключ от номера, запрос гостиничных услуг (специальные подушки, туалетные принадлежности, чайник и т.д), бронирование столиков в гостиничных ресторанах, мест в СПА, просмотр информации о персональных бонусах.

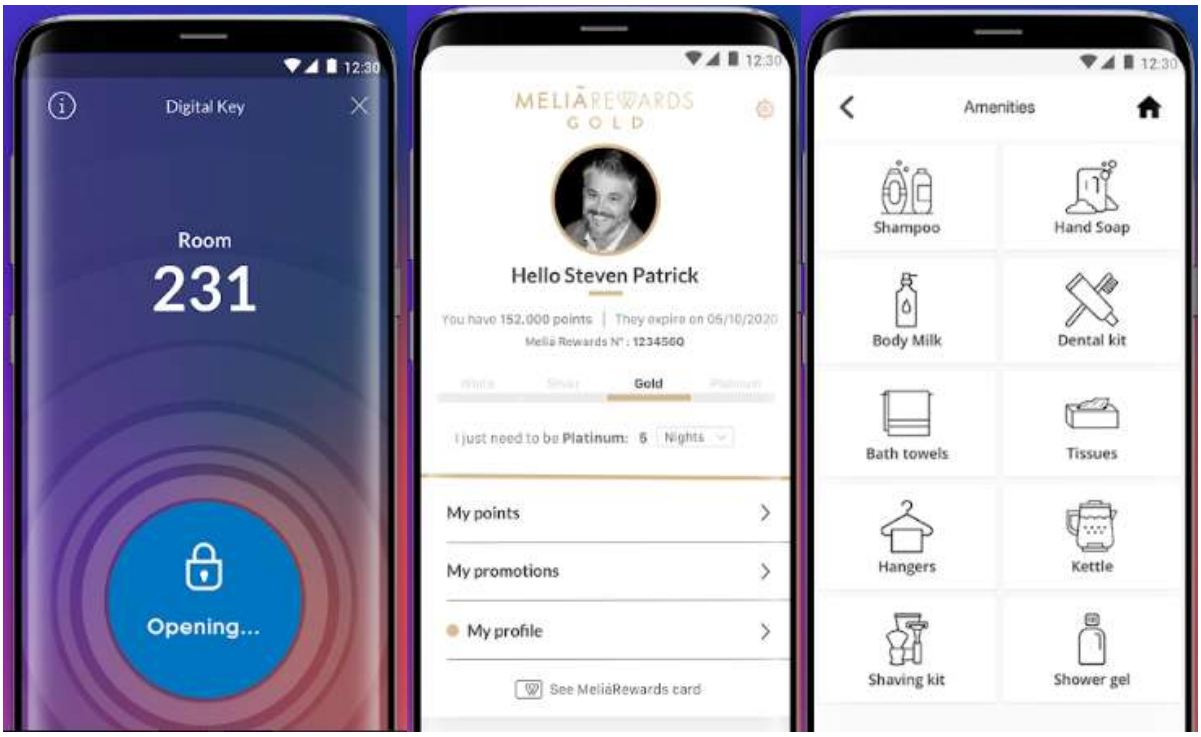

Рисунок 1.4 — Приложение «Melia – Hotel Bookings & more»

1.2 Мобильные приложения для студентов

Приложение «Общежитие ЛЭТИ 10-11»

Мобильное приложение для студентов, проживающих в общежитиях №10 и №11 ЛЭТИ (рисунок 1.5). Данный продукт еще находится в разработке, однако функционал уже обозначен и предполагает такие возможности: новостная лента; интернет-магазин необходимых вещей; информация о правилах и условиях проживания.

К недостаткам данного конкретного продукта можно отнести то, что продукт направлен на студентов только 2х общежитий.

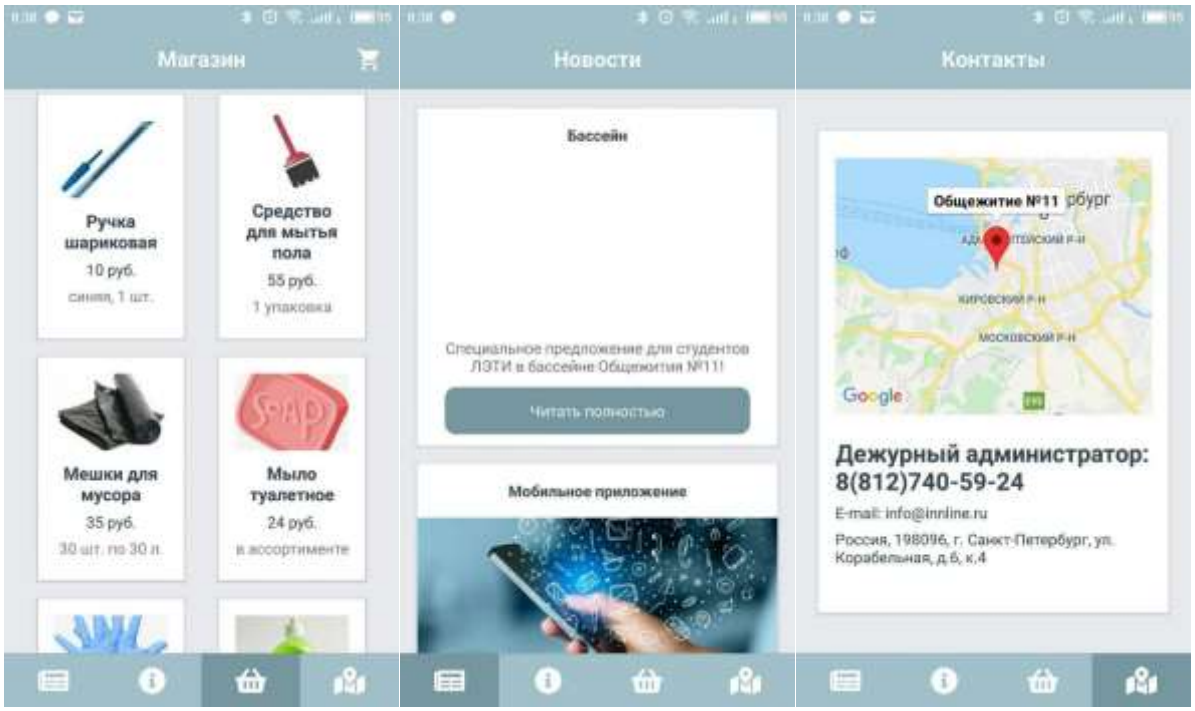

Рисунок 1.5 — Приложение «Общежитие ЛЭТИ 10-11»

Приложение «БФУ им. И. Канта»

Приложение создано для студентов БФУ им. И. Канта (рисунок 1.6). Данный продукт обладает следующим функционалом:

- заказ различных видов справок;
- просмотр сведений о задолженностях по оплате за общежитие и за обучение;
- расписание учебных занятий;
- расположение корпусов и общежитий университета;
- общая информация об инфраструктуре (Wi-Fi, почта, и т.д.).

Недостатки этого продукта:

- пользователи жалуются на нестабильную работу приложения;

– невозможность пользоваться приложением при отсутствии сканера отпечатка пальца на устройстве.

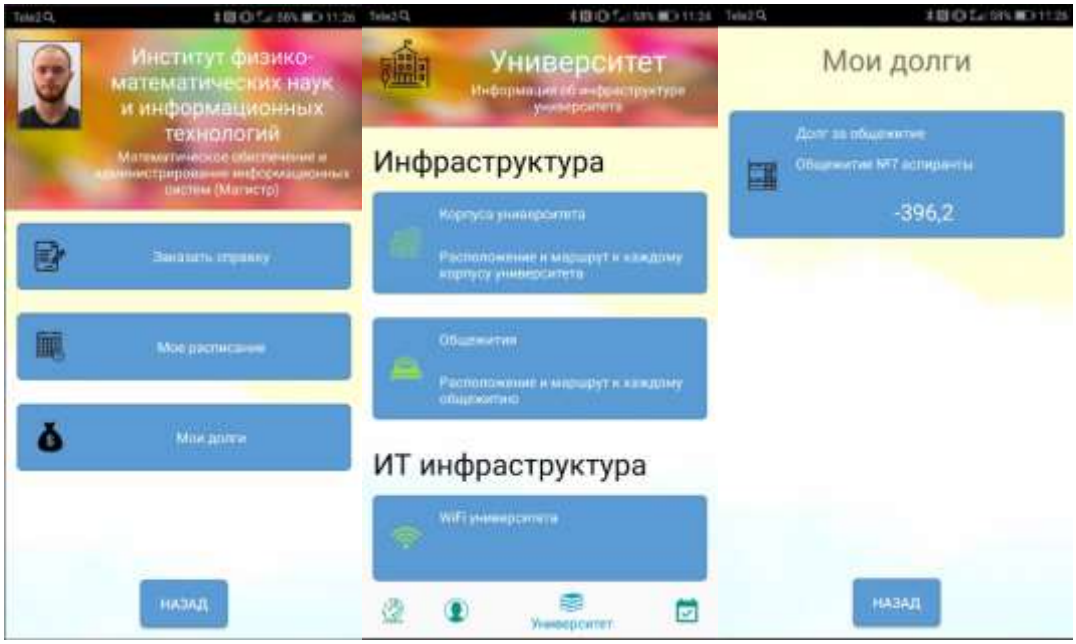

Рисунок 1.6 — Приложение «БФУ им. И. Канта»

Приложение «МАИ»

С помощью этого приложения (рисунок 1.7) пользователь может просматривать расписание занятий своей группы. Также студент может узнать информацию о библиотеках, столовых, общежитиях, спортивных секциях и прочей инфраструктуре МАИ.

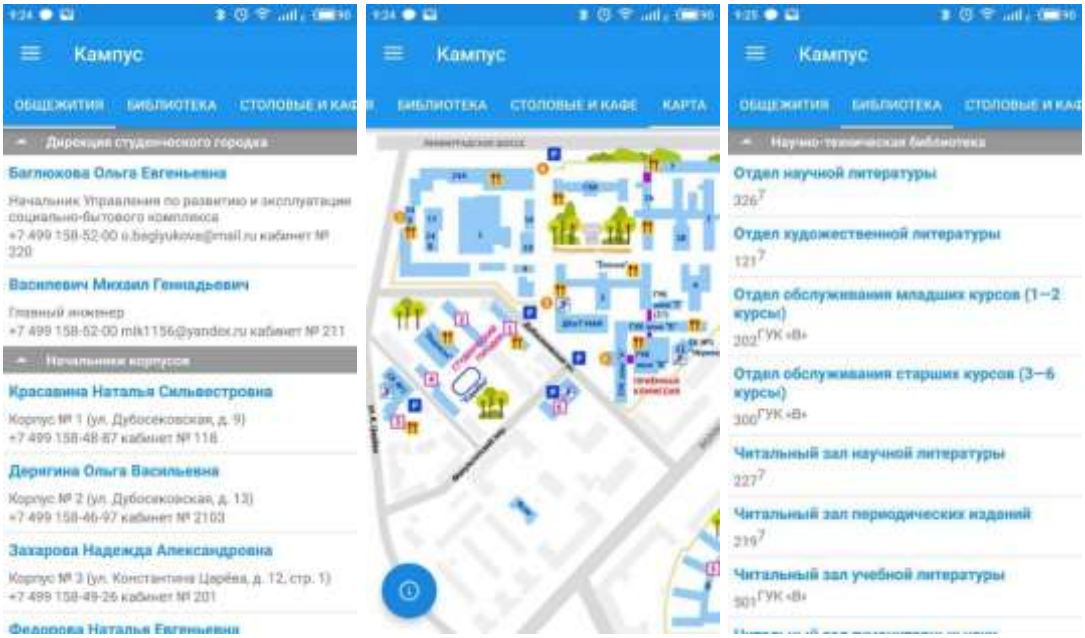

Рисунок 1.7 — Приложение «МАИ»

Недостатки продукта:

– дублирует информацию, которую возможно разместить на сайте университета;

– данные об общежитиях краткие и недостаточно информативные;

После анализа существующих на рынке систем электронных услуг в сфере предоставления мест для проживания можно увидеть некоторый схожий, а также отличительный функционал, который необходим гостиницам и общежитиям (таблица 1.1).

Таблица 1.1 – Перечень услуг гостиниц/общежитий

Гостиница	Общежитие
Схожий функционал	
– личный кабинет; – оцифрованный ключ от номера; – вызов сантехника; – вызов электрика; – вызов слесаря/плотника.	– просмотр общего списка общежитий; – информации о каждом в отдельности; – просмотр общежитий на карте; – личный кабинет; – оцифрованный пропуск в общежитие; – запись в прачечную; – вызов сантехника; – вызов электрика; – вызов слесаря/плотника.
Отличительный функционал	
– поиск ближайших отелей; – управление бронированием; – поиск ресторанов, достопримечательностей, транспорта поблизости; – звонки из приложения; – онлайн регистрация; – бронирование столика в ресторане/СПА; – запрос в номер специальной подушки, туалетных принадлежностей, чайника и т.д.; – заказ такси/трансфера; – заказ побудки.	– просмотр новостной ленты; – просмотр информации о культурно-массовых мероприятиях.

Оценим достоинства и недостатки готовых решений. Для наглядности сведем в таблицу рассмотренные продукты (таблица 1.2).

Таблица 1.2 – Достоинства и недостатки готовых решений

Возможности / Мобильное приложение	Просмотр новостной ленты	Просмотр списка объектов на карте	Просмотр информации о каждом объекте	Оцифрованный пропуск	Бытовые услуги(сантехник, прачечная, электрик и т.д.)	Информация о культ.-массовых мероприятиях	Авторизация пользователя
Radisson Rewards	–	+	+	–	–	–	+
Best Western To Go	–	+	+	–	–	–	+
NH Hotel Group – Book your hotel	–	+	+	–	–	–	+
Melia – Hotel Bookings & more	–	+	+	+	+	–	+
Общежитие ЛЭТИ 10-11	+	–	–	–	–	–	–
БФУ им. И. Канта	–	+	–	–	–	–	+
МАИ	–	+	+	–	–	–	–

Вывод по главе один

1. Анализ известных на рынке систем электронных услуг, выполненных как мобильные приложения для отелей («Radisson Rewards», «Best Western To Go», «NH Hotel Group – Book your hotel», «Melia – Hotel Bookings & more»), показал, что они не могут быть использованы как системы электронных услуг для жителей общежития (у них нет требуемого функционала), а приложения, разработанные для студентов ВУЗов в большинстве своем, являются простой копией сайта ВУЗа, где требуемый функционал (запись на стирку, подача заявки электрику, слесарю, плотнику, оцифрованный пропуск) не реализуется.

2. В качестве прототипа автоматизированной системы электронных услуг для объектов студгородка взят продукт «Melia – Hotel Bookings & more», так как он обладает широким функционалом, а именно:

- просмотр объектов на карте;
- информация о каждом объекте;

– авторизация пользователя;
– оцифрованный пропуск в номер;
– бытовые услуги.

Однако в этом продукте не реализованы следующие возможности:

– просмотр новостной ленты;
– информация о культурно-массовых мероприятиях,

которые являются важной частью системы электронных услуг студенческого городка ЮУрГУ. Эти возможности мы постараемся реализовать в создаваемой системе.

2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ СИСТЕМЫ

Название системы

«Автоматизированная система электронных услуг для объектов Студгородка ЮУрГУ».

Основные требования

Данная система должна состоять из двух подсистем, различного функционала:

- подсистема «Android-Клиент», функционирующая с планшетов и смартфонов на платформах Android, предназначена для конечного пользователя системы;
- подсистема «Web-сервер» предназначена для служебного использования, доступ к ее интерфейсу может быть получен сотрудником общежития с целью просмотра заявок по бытовым вопросам. Такой доступ может осуществляться с ПК с любой ОС.

Требования к подсистемам «Android-Клиент» и «Web-сервер»:

- минимальная поддерживаемая версия ОС Android 5.0;
- публикация Android-приложения в менеджере пакетов Play Market;
- публикация Web-сервера на хостинге.

Возможности подсистем «Android-Клиент»:

- просмотр новостей;
- список общежитий и информации о каждом общежитии отдельно;
- карта расположения общежитий;
- авторизация пользователей;
- взаимодействие с системой «Универис»;
- интеграция приложения с социальной сетью ВКонтакте;
- наличие возможности получать Push notifications;
- просмотр информации о следующих культурно-массовых мероприятиях:
  - «Мистер студгородок ЮУрГУ»;
  - «Мисс студгородок ЮУрГУ»;
  - «Лучшая комната студгородка»;
- оцифровка пропуска в общежитие;
- запись в прачечную;
- возможность оставлять заявки по бытовым вопросам:
  - вызов сантехника;
  - вызов электрика;
  - вызов слесаря/плотника.

3 РАЗРАБОТКА СИСТЕМЫ ЭЛЕКТРОННЫХ УСЛУГ ДЛЯ ОБЪЕКТОВ СТУДГОРОДКА ЮУРГУ

3.1 Разработка архитектуры программного обеспечения системы

На сегодняшний день одной из самых часто используемых архитектур, при разработке различных приложений, является архитектура «клиент-сервер».

Архитектура «клиент-сервер» подразумевает разделение нагрузки сервера и клиента. Клиентские компьютеры реализуют связь, позволяющую пользователю компьютера запрашивать службы сервера и представлять результаты, возвращаемые сервером. Серверы ждут появления запросов от клиентов, а затем возвращают ответ. Эта архитектура полезна в основном в тех случаях, когда клиенты и сервер обычно выполняют обособленные задачи [1].

Существует три типа «клиент-серверной» архитектуры: одно-, двух- и трехуровневая. Наилучшей среди них является трехуровневая архитектура (рисунок 3.1).

Рисунок 3.1 — Модель трехуровневой архитектуры

Трехуровневая архитектура обеспечивает множество преимуществ в разработке за счет выделения трех слоев:

- уровень представления (presentation tier) — уровень представления состоит из пользовательского интерфейса. Он при помощи графики отображает контент и информацию, полезную для конечного пользователя. Этот уровень часто строится с использованием таких технологий как HTML5, JavaScript, CSS (в мобильной разработке: XML, Kotlin, Java) и взаимодействует с другими уровнями через API-вызовы;
- уровень приложения (application tier) — уровень приложения содержит функциональную бизнес-логику, которая управляет основными возможностями приложения. Он часто написан на Java, .NET, C#, Python, C++ и т. д.;

– уровень данных (data tier) — уровень данных состоит из системы хранения данных (базы данных) и организации доступа к ним. Примерами таких систем являются MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB и др. Доступ к данным осуществляется на уровне приложения с помощью API-вызовов.

Такое разделение делает систему гибкой, позволяя разработчикам обновлять определенную её часть независимо от других. Например, пользовательский интерфейс может быть переработан или модернизирован без ущерба для лежащей в его основе функциональной бизнес-логики и логики доступа к данным. Такая архитектура идеальна для внедрения программного обеспечения от третьей стороны в уже существующие проекты.

Типичная структура проекта на базе трехуровневой архитектуры будет иметь уровень представления, развернутый на рабочем столе, ноутбуке, планшете или мобильном устройстве либо через веб-браузер, либо через приложение. Базовый уровень приложения обычно размещается на одном или нескольких серверах, но также может размещаться в облаке или на выделенной рабочей станции в зависимости от сложности и необходимой вычислительной мощности. А также слой данных, состоящий из одной или нескольких реляционных баз данных, источников Big data или других типов систем баз данных, размещенных локально или в облаке.

Использование трехуровневой архитектуры имеет такие преимущества как скорость разработки, масштабируемость, производительность, доступность. Так же повышается надежность системы за счет независимости компонентов проекта.

По этим причинам архитектура программного обеспечения создаваемой системы электронных услуг также будет представлена тремя уровнями (рисунок 3.2).

Рисунок 3.2 — Архитектура ПО

Как показано на рисунке 3.2. взаимодействие компонентов будет осуществляться следующим образом: Android-клиент в соответствии с пользовательским запросом посылает запрос к Backend, в свою очередь Backend обращается за необходимыми данными к Базе данных, обрабатывает полученную из БД информацию и отправляет ответ Android-клиенту.

В свою очередь каждый из уровней архитектуры программного обеспечения также будет иметь свою архитектуру, а также обращается к различным сторонним сервисам, поэтому на рисунке 3.3 покажем расширенную архитектуру системы.

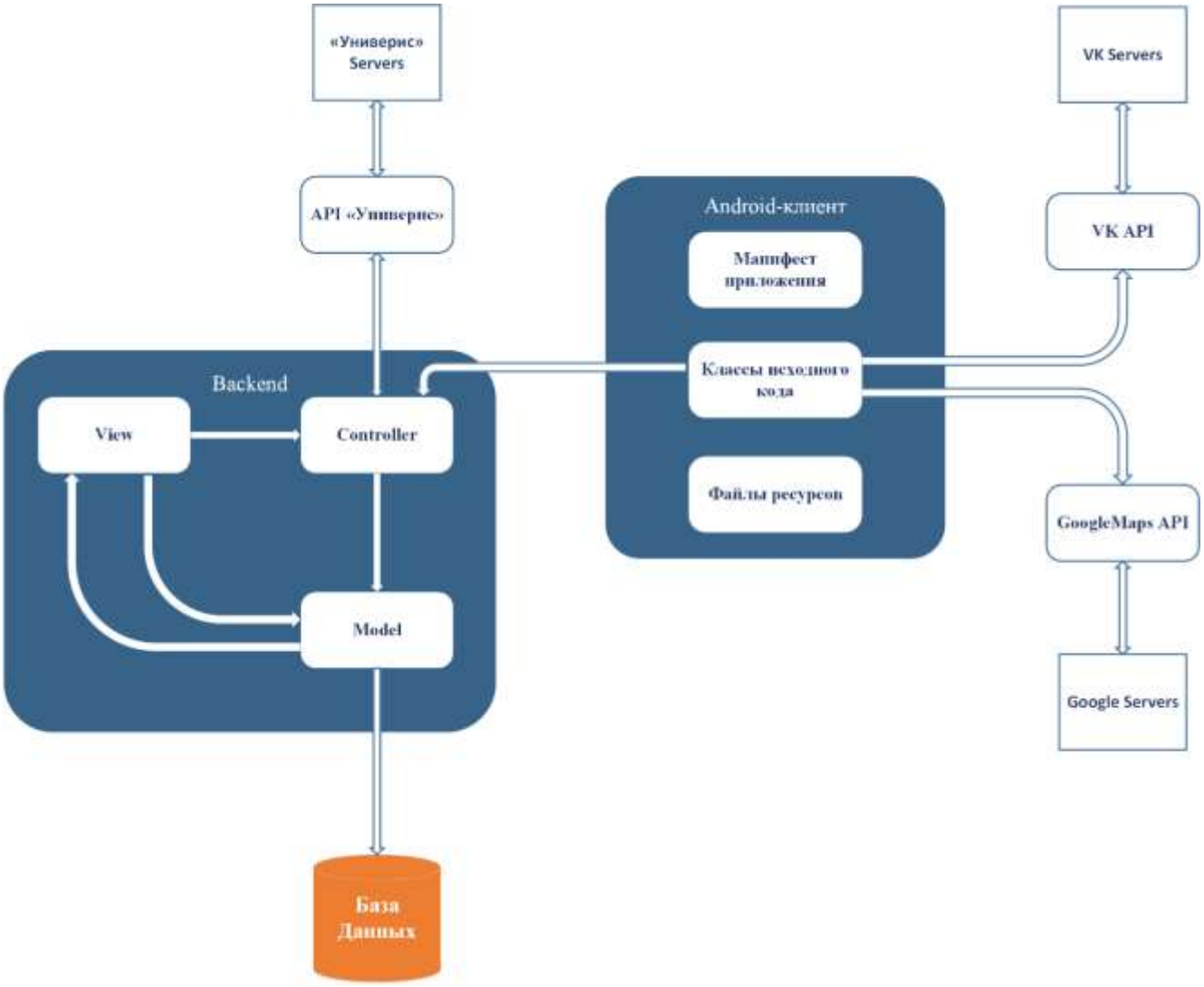

Рисунок 3.3 — Расширенная архитектура системы

Из рисунка 3.3 видно, как происходит взаимодействие компонентов системы. Android-клиент обращается к картографическому сервису, сервису социальной сети, а также к Backend части системы. Затем Android-клиент обрабатывает полученные данные, чтобы с помощью файлов собственных ресурсов отобразить эти данные пользователю.

Backend в свою очередь построен на принципах паттерна MVC (Model-View-Controller). Паттерн MVC предполагает три основных объекта для использования в разработке программного обеспечения [12].

– Модель, представляющая базовую логическую структуру данных в программном приложении. Эта объектная модель не содержит никакой информации о пользовательском интерфейсе.

– Представление, которое представляет собой набор классов, представляющих элементы пользовательского интерфейса. Т. е. всё, что пользователь может видеть на экране и на что может реагировать (кнопки, окна отображения и т. д) [20].
– Контроллер, представляющий классы, соединяющие модель и представление. Он определяет, какое представление должно быть отображено [20].

Backend при помощи моделей взаимодействует с базой данных приложения, а с помощью контроллеров обеспечивает интеграцию с системой «Универис».

3.2 Особенности Android разработки

В качестве системы сборки в Android-разработке используется **Gradle**, поэтому в проекте имеются файлы с раширением .gradle [7]. В таких файлах описываются настройки сборки проекта, а именно:

– используемая версия языка программирования;
– пути и методы загрузки требуемых плагинов;
– другие зависимости необходимые для осуществления сборки;

также в них описываются настройки сборки конкретного разрабатываемого приложения, такие как:

– описание используемых в коде сторонних библиотек;
– информацию о необходимой конфигурации Android-устройств, которые могут использовать приложение.

Основой для любого приложения Android является файл манифеста: **AndroidManifest.xml**. Он описывает все компоненты, из которых состоит приложение – Activity, service и т.д [6]. Также указывается какие разрешения пользователя необходимы для корректной работы приложения.

Приложения Android состоят из слабо связанных компонентов, их связывает манифест приложения. Компоненты, составляющие основу всех Android-приложений следующие:

– **Activity**. С их помощью строится пользовательский интерфейс приложения. Activity используют фрагменты (fragments) и представления (views) для компоновки и отображения информации, а также для реагирования на действия пользователя.
– **Service** (службы). Компоненты служб работают без пользовательского интерфейса. Они обновляют источники данных, вызывают уведомления и широковещательные намерения. Используются для выполнения длительных задач или задач, не требующих взаимодействия с пользователем (например, задач, которые необходимо выполнять, даже если Activity приложения не активны или не видны) [8].

– **Intents** (намерения). Это мощная структура передачи сообщений между приложениями. Намерения используются для запуска и остановки Activity и Service, для широковещательной рассылки в масштабах всей системы или для запроса действия, которое необходимо выполнить с определенными данными.

В Android-разработке принято отдельно хранить различные ресурсы, такие как строки, цвета, изображения, анимации, темы и макеты пользовательского интерфейса. Такое хранение ресурсов облегчает их обслуживание, обновление и управление ими. Это также позволяет создавать альтернативные значения ресурсов, например, для поддержки интернационализации. Когда приложение запускается, система Android автоматически выбирает нужные ресурсы из доступных альтернатив.

По мере перехода пользователя от одного Activity к другому или из одного приложения в другое, Activity также переходят в разные состояния жизненного цикла. Методы обратного вызова позволяют Activity понять, что система создает, останавливает или возобновляет Activity, или уничтожает процесс, в котором оно находится.

В рамках методов обратного вызова жизненного цикла можно объявить, как будет вести себя Activity при различных действиях пользователя. Другими словами, каждый метод обратного вызова позволяет выполнять определенные задачи, соответствующие данному изменению состояния [11]. Выполнение верных задач в нужное время и правильная обработка переходов делают приложение более надежным и эффективным. Иллюстрация жизненного цикла Activity представлена на рисунке 3.4.

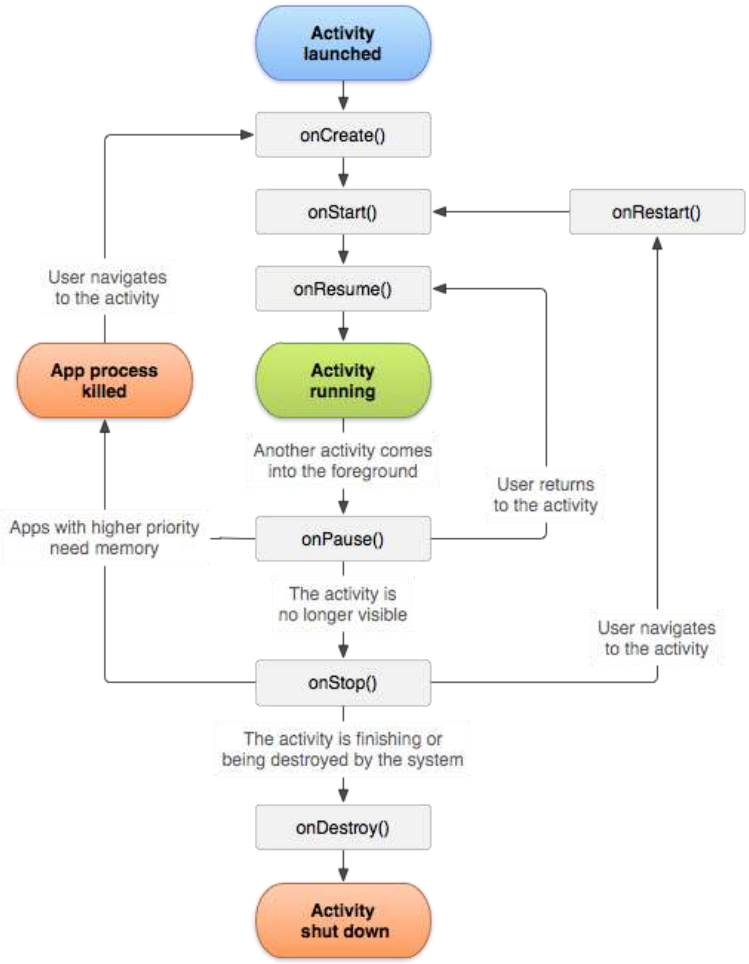

Рисунок 3.4 – Иллюстрация жизненного цикла Activity

Когда пользователь покидает Activity, система вызывает методы для «демонтажа» Activity. Если пользователь возвращается к Activity, оно возобновляется с того места, где пользователь закончил работу. Вероятность того, что система убьет данный процесс — вместе с находящимся в нем Activity — зависит от состояния Activity в данный момент.

В зависимости от сложности решаемой задачи, не всегда требуется реализовывать все методы жизненного цикла [2]. Необходимо понимать назначение каждого из них, а реализовать те, которые гарантируют, что создаваемое приложение будет вести себя так, как задумано. Пользователь ожидает, что при повороте устройства, переходе в многооконный режим, все изменения его интерфейса сохранятся, а он вернется к работе с ними позже.

Однако при подобных изменениях система по умолчанию уничтожает Activity, а при временном переключении на другое приложение, система может уничтожить весь процесс вашего приложения, пока пользователь отсутствовал. Поэтому необходимо сохранять переходные состояния, для корректной работы приложения.

3.3 Выбор программно-аппаратных средств разработки

Для того, чтобы начать разработку системы необходимо определить, какие инструменты должны быть использованы в процессе разработки, а именно определить:

- средства разработки программного обеспечения;
- аппаратные характеристики пользовательских устройств.

3.3.1 Среда разработки Android-клиента

При выборе среды разработки стоит обращать внимание на многие факторы. Например, необходимо подумать об организации, отвечающей за обслуживание Android IDE, о том, как долго существует эта организация, может ли она в обозримом будущем продолжать поддерживать IDE.

Большинство разработчиков предпочитают использовать язык программирования, который им уже знаком, поэтому языки программирования, поддерживаемые средой разработки это один из важнейших факторов.

Также стоит оценить удобство использования интерфейса, встроенные возможности эмулятора и тестирования, интегрированные средства сборки и отладки, системные требования среды разработки [5]. В таблице 3.1 сравним самые популярные IDE по некоторым критериям.

Таблица 3.1 – Основные характеристики сред разработки

Среда Android разработки	Поддерживаемые языки	ОС на которой работает среда	Цена
Android Studio	Java, C/C++, Kotlin	Windows, MacOS, Linux	Бесплатно
Eclipse	Java, C/C++/C#, JavaScript, Python	Любая ОС с поддержкой Java	Бесплатно
Visual Studio (with Xamarin)	C/C++/C#, Visual Basic, PHP, JavaScript	Windows, MacOS, Linux	Бесплатно или от $2,999+
IntelligJ IDEA	Java, Scala, Groovy, Kotlin, JavaScript TypeScript, SQL	Windows, MacOS, Linux	Бесплатно или от $499/год

Из таблицы 3.1 следует, что наиболее подходящей средой разработки Android-клиента является Android Studio, так как, в сравнении с остальными средами, она обладает следующими достоинствами:

- бесплатная;
- является свободно распространяемым программным обеспечением;

- осуществляет поддержку необходимых языков программирования;
- имеет гибкую систему сборки на основе Gradle;
- быстрый и многофункциональный эмулятор;
- широкий выбор инструментов и фреймворков для тестирования;
- встроенная поддержка облачной платформы Google, что упрощает интеграцию Google Cloud Messaging и App Engine.

3.3.2 Язык разработки Android-клиента

Выбранная нами среда разработки поддерживает три языка: Java, Kotlin, C/C++. Рассмотрим их подробнее:

- Java – Java на сегодняшний день является самым популярным языком программирования, который позволяет создавать различные приложения широкого спектра: веб-сайты и веб-сервисы, Desktop-приложения, мобильные приложения для ОС Андроид. Это универсальный кроссплатформенный язык, поэтому приложения на Java будут работать на большинстве известных платформ как Windows, Linux, MacOS.
- Kotlin – был разработан JetBrains, чешской компанией, известной своей популярной IDE, IntelliJ IDEA. Команда Android Google недавно объявила, что они официально добавляют поддержку языка программирования Kotlin. Kotlin был разработан для решения некоторых проблем, встречающихся в Java [6]. Синтаксис Kotlin проще, чище по сравнению с Java. Простота языка и его гибкость дает разработчику больше возможностей для написания быстрого, но качественного кода. Кроме того, существует возможность использования Kotlin и Java вместе в одном проекте.
- C/C++ – из поддерживаемых Android Studio языков C/C++ является наиболее сложным. Однако это мощное средство, позволяющее манипулировать низкоуровневыми операциями, такими как работа с памятью. Необходим при использовании Android NDK (Native Development Kit).

Так как с Android Native Development Kit работать в данном проекте не предполагается, поэтому выбор можно сократить до Java и Kotlin.

На сегодняшний день выбор между Java и Kotlin зависит от индивидуальных предпочтений разработчика. Поэтому в данном случае составим список личных требований, и сравним два языка разработки Android-приложений на предмет удовлетворения этим требованиям (таблица 3.2).

Из таблицы 3.2 следует, что возможности Kotlin полностью удовлетворяют всем требованиям для разработки Android-приложения. Код проще и понятнее, а то, что Kotlin работает поверх виртуальной машины Java позволит при необходимости

также использовать Java во время разработки. Поэтому остановим выбор на языке программирования Kotlin.

Таблица 3.2 – Основные критерии для сравнения Java и Kotlin

	Простота	Краткость	Безопасность	Официальная поддержка	Поддержка ООП	Поддержка процедурного программирования	Отсутствие get()/set()
Java	+	–	–	+	+	–	–
Kotlin	+	+	+	+	+	+	+

3.3.3 Инструментарий для разработки Web-сервера

Model-View-Controller (MVC) — это архитектурный шаблон, который разделяет приложение на три основных логических компонента: модель, представление и контроллер. Каждый из этих компонентов предназначен для обработки конкретных аспектов разработки приложения [20]. MVC является одним из наиболее часто используемых отраслевых стандартов Web-разработки для создания масштабируемых и расширяемых проектов.

Наиболее популярными средствами для разработки MVC проектов являются платформа ASP .NET Core и фреймворк PHP YII, для проведения сравнительно анализа занесем их основные характеристики в таблицу (таблица 3.3)

Таблица 3.3 — Основные характеристики ASP .NET Core и PHP YII

	Масштабируемость	Тип языка	Скорость	Надежный и строго типизированный язык	Тесная интеграция с Visual Studio	Удобный синтаксис
ASP .NET Core	+	Компилируемый	+	+	+	+
PHP YII	–	Интерпретируемый	–	–	–	–

Исходя из сравнения данных, представленных в таблице 3.3, для разработки Web-сервера выберем платформу ASP .NET Core. Работа с этой платформой будет осуществляться в среде разработки Visual Studio 2017 версии Community, так как она обладает всеми необходимыми возможностями и является бесплатной.

3.3.4 Аппаратные требования к системе

Главную роль в обеспечении совместимости Android-приложения с устройствами, на которых оно может быть установлено, играет уровень API (API level). Данный уровень обозначается целочисленным числом и зависит от версии системы установленной на устройстве, а при разработке приложения задается разработчиком вручную. На рисунке 3.5 представлена информация о том, как соотнесены версии системы Android с уровнями API и общим распределением устройств, поддерживающих эти версии.

Исходя из данных, представленных на рисунке 3.5, уровень API 21 поддерживается 85,0% устройств. Выберем его в качестве минимально используемого. Поэтому устройства, которые будут использовать систему должны иметь установленную версию Android не ниже 5.0.

Что касается служебного Web-сервера, то аппаратные требования для его работы не большие. Необходим компьютер, имеющий устройства ввода и вывода информации (клавиатура, мышь, монитор), и выход в глобальную сеть.

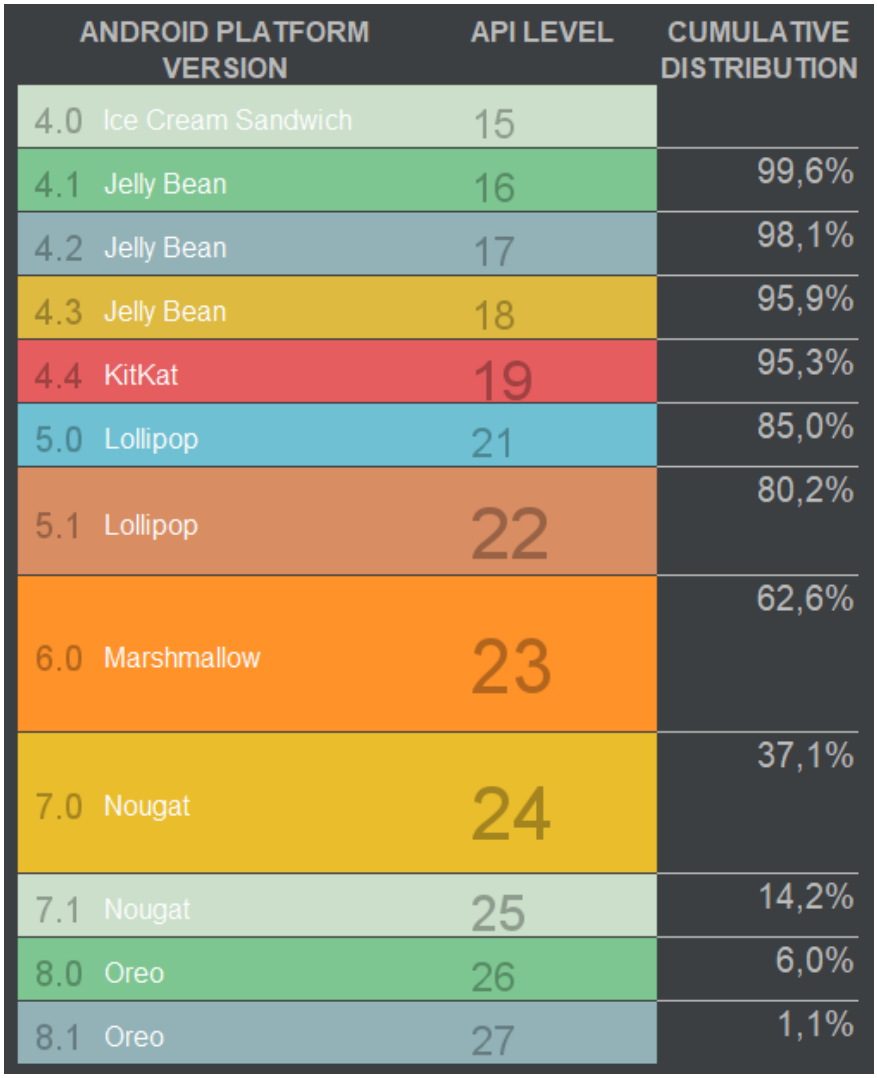

Рисунок 3.5 — Соотношение версий Android с уровнем API и процентом поддерживающих устройств

3.4 Формирование информационных ресурсов

Система электронных услуг студгородка ЮУрГУ должна активно использовать информационные ресурсы. В первую очередь необходимо определить, какая информация должна будет отображаться в Android-приложении.

1. На карте будет отображаться местоположение каждого общежития.

Соответственно для этого необходима следующая информация:

- координаты общежития (ширина, долгота);
- название общежития (его номер).

2. Будет доступна информация о каждом общежитии.

Следует определить, какая именно информация должна быть доступна:

- название общежития (номер);
- адрес;
- номер телефона;
- количество этажей;
- число студентов;
- фотография.

3. В приложении будет отображаться информация о действующем студенческом совете общежития.

В каждом общежитии существует студенческий совет – группа студентов, образующая одну из форм инициативной, самостоятельной, ответственной общественной деятельности студентов, направленной на решение важных вопросов жизнедеятельности студентов, развитие её социальной и культурной активности. Соответственно необходимо снабдить пользователя информацией о членах студенческого совета и их должностях:

- фамилия, имя, отчество;
- фото;
- должность.

Помимо тех данных, которые должны доходить до конечного пользователя, необходимы и некоторые служебные данные, такие как информация об администраторах системы:

- логин;
- пароль.

Соответственно, необходимо определиться, как хранить и как структурировать эти данные.

Для решения вопроса о хранении данных необходимо использовать базу данных. А значит нужно определиться с тем, какую систему управления базами данных (СУБД) использовать.

Так как данные, которые мы собираемся хранить не большие, то не стоит использовать громоздкие и сложные в использовании системы управления базами данных, такие как Oracle, MySQL и т.д. Тем более большинство из них являются платными. Требуется компактная реляционная СУБД, с открытым исходным кодом. Этим условиям удовлетворяет SQLite.

SQLite – это встраиваемая кроссплатформенная система управления базами данных, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах [23]. Слово «встраиваемый» (embedded) означает, что движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа, что предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с базой данных обращения происходят напрямую к файлам и благодаря этому, SQLite легко встраивается в приложения и является мобильной.

После выбора СУБД необходимо определиться с тем, как будет структурирована информация в создаваемой базе данных. Для этого следует составить архитектуру базы данных (рисунок 3.6).

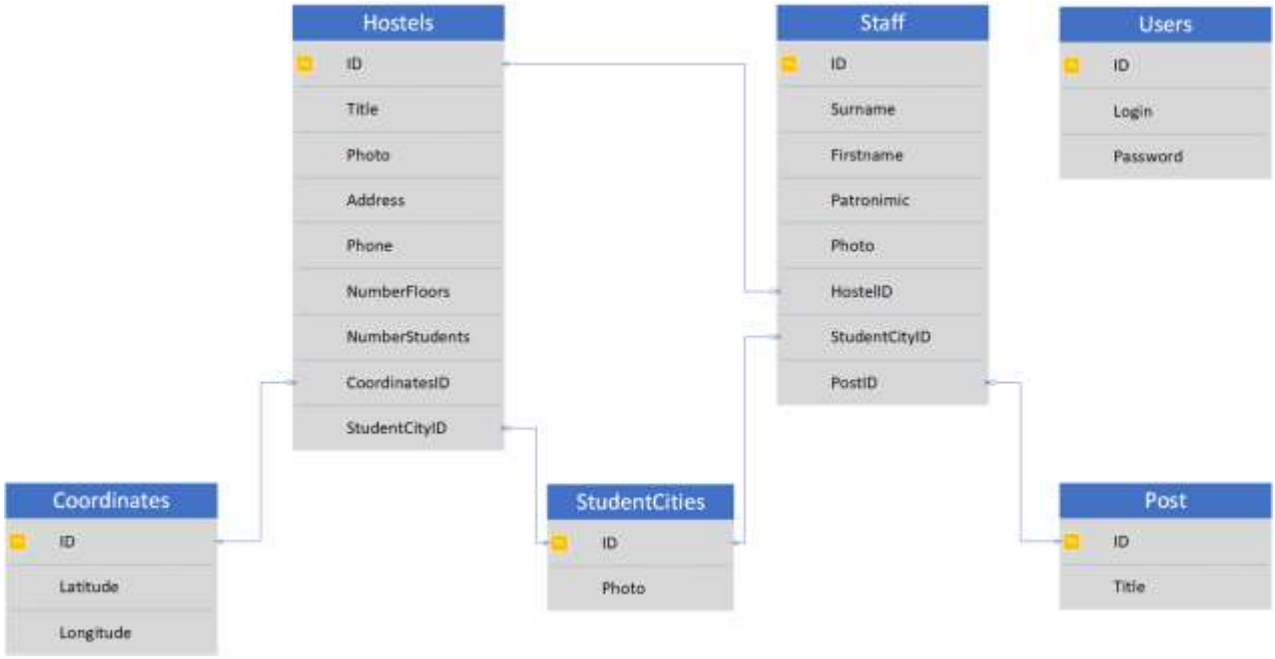

Рисунок 3.6 – Архитектура базы данных

Архитектура, представленная на рисунке 3.6, показывает, что база данных будет состоять из шести таблиц. Их назначение укажем в таблице 3.4

Таблица 3.4 – Назначение таблиц базы данных

Таблица	Назначение
Hostels	Имеет 9 полей. Предназначена для хранения данных о каждом общежитии (название общежития, фотография, адрес, телефон, количество этажей, количество проживающих студентов). Также имеются служебные поля, для связи с другими таблицами: – CoordinatesID – связывает общежитие с его координатами; – StudentCityID – связывает общежитие с принадлежностью к студгородку; – ID – первичный ключ таблицы.
Coordinates	Имеет 3 поля. Предназначается для записи координат общежития. В двух полях хранятся две координаты: ширина и долгота. Служебное поле ID – первичный ключ таблицы.
Post	Имеет 2 поля. Хранит информацию о том, какие должности существуют. Служебное поле ID – первичный ключ таблицы.
Staff	Имеет 8 полей. Предназначена для хранения информации об администрации общежития и о студенческом совете общежития (ФИО, фотография). Служебные поля выполняют следующие функции:
	– HostelID – связывает человека с общежитием, где он ведет деятельность; – StudentCityID – связывает человека с принадлежностью к студгородку, если это администрация студгородка или студсовет студгородка; – PostID – связывает человека с постом, который он занимает; ID – первичный ключ таблицы.
StudentCities	Имеет 2 поля. Служебное поле ID, выполняющее функции первичного ключа, и поле Photo – хранение фотографии студенческого городка. Таблица предназначена для того, чтобы обособить администрацию и студсовет студенческого городка от общежитий.
Users	Имеет 3 поля. Два поля хранят логины и пароли администраторов системы электронных услуг. Не имеет связей с другими таблицами. Служебное поле ID, выполняющее функции первичного ключа

Все таблицы, представленные на рисунке 3.6, необходимо заполнить соответствующей актуальной информацией. Для этого необходимо обратиться к председателю студенческого совета студгородка, а также директору студгородка, так как именно они располагают необходимой информацией. Однако координаты необходимо взять из любого современного картографического сервиса. Готовые таблицы приведены в приложении А.

3.5 Реализация программного обеспечения системы

Программная реализация системы подразумевает использование различных библиотек, сервисов и т.д. Поэтому на первом этапе программной реализации системы осуществим выбор необходимого инструментария.

3.5.1 Выбор сервисов

Выбор картографического сервиса

Существует достаточное количество картографических сервисов. В нашей системе необходимо реализовать следующее:

- определение местоположения пользователя;
- отображение на карте объектов студгородка и пользовательского местоположения.

Для того чтобы решить, какие карты лучше использовать в разработке, необходимо провести сравнение основных характеристик, которые потребуются в разработке.

В первую очередь необходима возможность встраивания карты в мобильное приложение, информация на карте должна отображаться на русском языке. Этим требованиям удовлетворяют все рассматриваемые сервисы.

Следующий важнейший критерий — это бесплатное использование. Google Maps позволяет использовать свой сервис в мобильных приложениях бесплатно [4]. Компания Яндекс выдвигает определенные условия бесплатного использования своих библиотек для работы с картографическими данными [19]. Среди прочих есть некоторые условия, которые не позволяют использовать библиотеки Яндекс в разрабатываемой системе:

- регистрация в приложении должна быть доступна для любого пользователя .

Это условие не совпадает с нашими целями, поэтому использование этих библиотек нецелесообразно.

Компания 2GIS не позволяет использовать свои библиотеки без покупки доступа к ним.

Open street map — проект, создаваемый людьми на добровольной основе, и его поддержка также добровольна. Это ставит под вопрос надежность работы таких карт.

Для наглядности запишем все перечисленные требования в таблицу и обозначим удовлетворяет ли тот или иной сервис каждому требованию (таблица 3.5).

Таблица 3.5 – Требования к картографическим сервисам

	Использование в мобильном приложении	Бесплатное использование	Справочная информация	Русский язык	Стабильная поддержка
Google Maps	+	+	+	+	+
Яндекс карты	+	–	+	+	+
Open street map	+	+	–	+	–
2GIS	+	–	+	+	+

Сравнивая данные о каждом картографическом сервисе, представленные в таблице 3.5, понимаем, что наиболее подходящим вариантом для решения поставленной задачи об определении и отображении местоположения объектов студгородка и пользователя является сервис Google Maps.

Выбор сервиса для отображения новостей

В настоящее время широкое распространение получает такой вид маркетинга как SMM (Social Media Marketing, иначе маркетинг в социальных сетях). Основная цель SMM – улучшить коммуникацию с пользователями, повысить узнаваемость бренда и увеличить охват потенциальной аудитории. Студенческий городок ЮУрГУ уделяет данному виду маркетинга достаточное количество внимания, это связано с тем, что в стенах студгородка ежегодно проводится множество мероприятий, самыми глобальными из которых можно назвать «Мистер Студгородка ЮУрГУ» и «Мисс Студгородка ЮУрГУ». Для подобных мероприятий пиар-компания и привлечение спонсоров – один из важнейших факторов, так как это обеспечивает привлечение аудитории к участию или просто явке на мероприятие.

Сейчас рекламная компания проводится исключительно на базе социальных сетей ВКонтакте и Instagram. Пользователям зачастую сложно уделять достаточное внимание новостям, связанным с событиями в студенческом городке, из-за неимоверно большого количества контента в новостной ленте. Соответственно создавае-

мый Android-клиент поможет создать дополнительные возможности для оповещения студентов, проживающих в общежитиях студгородка, о надвигающихся мероприятиях, событиях и т.д.

Есть несколько вариантов, компоновки новостной ленты в создаваемой системе:

- создавать каждую новость вручную;
- использовать VK API и с его помощью подгружать актуальные новости из социальной сети ВКонтакте;
- использовать Instagram API и с его помощью подгружать новости из социальной сети Instagram.

Первый вариант является абсолютно не приемлемым, так как для его реализации необходимо добавлять обязанности человеку, который занимается информационными ресурсами студенческого городка ЮУрГУ, или в худшем случае возможно искать нового человека, который бы имел достаточную квалификацию и возможности для подобной работы.

Другие два варианта достаточно равнозначны. Однако контент, выкладываемый в обеих социальных сетях дублируется не часто, и более содержательной и полезной можно назвать информацию из официальной группы студенческого городка ЮУрГУ ВКонтакте. Также стоит отметить, что VK API удобнее в использовании тем, что его документация создана на русском языке, хотя это не является проблемой, но время, потраченное на изучение библиотеки, сокращается.

Таким образом, выберем в качестве сервиса для отображения новостей VK API по следующим причинам:

- пользователи и сообщества чаще всего для получения/предоставления важной информации используют социальную сеть ВКонтакте;
- документация VK API написана на русском языке.

3.5.2 Структура проекта Android-клиента

Структура любого Android проекта сложная. В первую очередь за счет разделения файлов кода и файлов ресурсов. На рисунке 3.7 представлена структура созданного Android проекта. Первое, что стоит отметить – манифест приложения, о его предназначении говорилось ранее (см. п. 3.2). Директория «java» содержит в себе все файлы исходного кода приложения. Классы могут сортироваться разработчиком по разным каталогам в зависимости от назначения. Поэтому в нашем проекте создано множество подкаталогов.

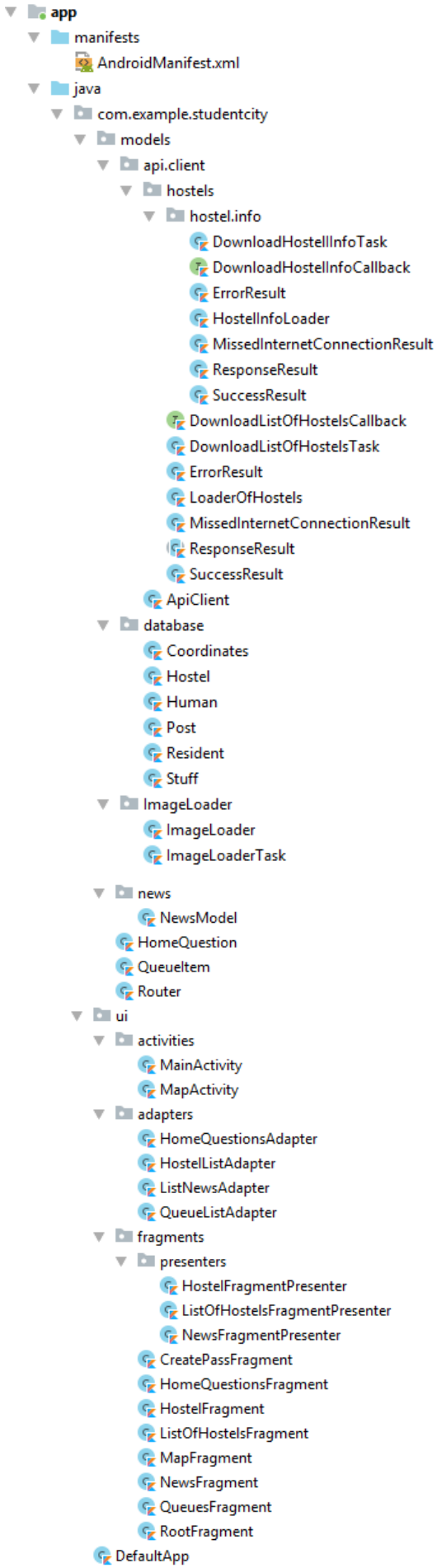

Рисунок 3.7 – Структура Android-проекта

Основное разделение идет на логику приложения (models) и пользовательский интерфейс (ui). Директория «ui» содержит в себе все классы, связанные с отображением информации пользователю. Это файлы Activity, адаптеры, фрагменты. Адаптер необходимы для связи массивов данных с набором элементов отображения. Фрагменты – это те части, на которые можно разделить Activity, так как организация приложения на основе одних только Activity это не всегда оптимальное решение. В директории «models» находится «мозг» приложения. Классы, расположенные в ней, отвечают за запрос и получение данных с сервера приложения, или использование API для получения данных от сторонних сервисов.

Кроме классов исходного кода в структуре проекта также существует директория, в которой хранятся все ресурсы приложения (рисунок 3.8).

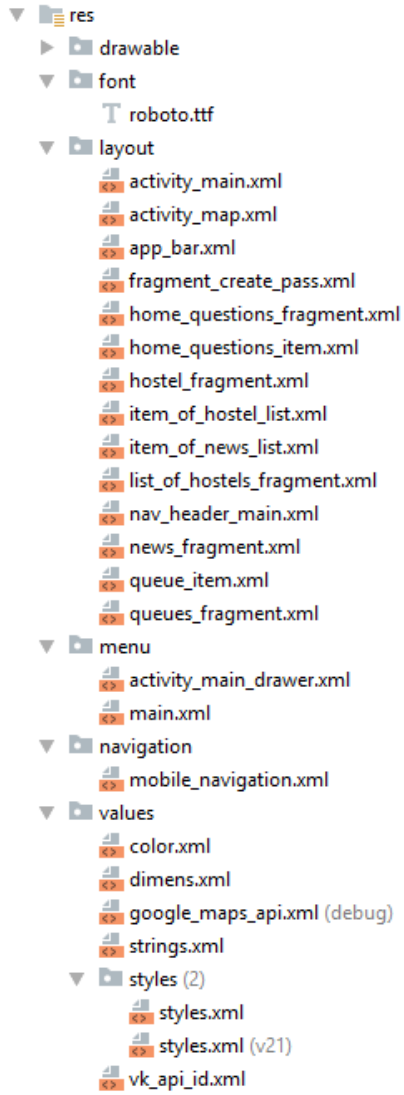

Рисунок 3.8 – Дерево файлов ресурсов

Директория имеет название «res» (от слова «resources» – ресурсы). Подкаталог «drawable» содержит в себе все файлы изображений, которые будет использовать приложение (иконка, фоновые рисунки, и т.д.). Подкаталог «font» хранит в себе все шрифты, которые использует приложение. Один из наиболее важных подкаталогов

– «layout». В нем хранятся все файлы разметки, созданные при помощи XML (eXtensible Markup Language – расширяемый язык разметки). Именно этими файлами компонуется внешний вид приложения. Также важным является подкаталог «values», в нем принято хранить все данные о цветах, отступах, стилях, и даже строках текста, которые использует приложение.

3.5.3 Интеграция картографического сервиса в Android-приложение

Для отображения на карте объектов студенческого городка будем использовать сервис Google Maps.

Необходимо встроить в приложение фрагмент карты, на котором маркерами будут отмечены все общежития студгородка. Всё, что для этого необходимо – координаты каждого общежития. Которые уже занесены в созданную базу данных.

В первую очередь необходимо запросить разрешение у пользователя на получение его местоположения. Такое разрешение называется permission и записывается так:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Далее в коде разметки необходимо отрисовать фрагмент карты (рисунок 3.9)

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.MapActivity" />
```

Рисунок 3.9 – XML код фрагмента карты

Затем требуется инициализировать этот фрагмент карты в коде основного файла данного Activity (рисунок 3.10).

```
private lateinit var mMap: GoogleMap

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_map)

    val mapFragment:SupportMapFragment = supportFragmentManager
        .findFragmentById(R.id.map) as SupportMapFragment
    mapFragment.getMapAsync(this)
}
```

Рисунок 3.10 – Инициализация карты

После того как карта будет инициализирована вызовется функция `showHostels()`, задача которой отобразить маркер местоположения каждого общежития на карте. Функция получает от сервера информацию обо всех общежитиях, если всё прошло успешно, то для каждого общежития из списка вызывается функция `showHostel()`. Она помещает маркер на карте, в соответствии с полученными координатами. Если

вдруг произошла ошибка при получении данных с сервера, пользователю будет выведено сообщение об ошибке (рисунок 3.11).

```
private fun showHostels() {
    val loaderOfHostels = LoaderOfHostels()
    loaderOfHostels.download( activity: this, object:DownloadListOfHostelsCallback{
        override fun onServerError() {
            showMessage("Ошибка загрузки данных")
        }

        override fun onFailInternetConnection() {
            showMessage("Отсутствует интернет соединение")
        }

        override fun onSuccess(hostels: Array<Hostel>) {
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
                LatLng(55.160063, 61.367763),  15f)
            )
            for (hostel in hostels) {
                showHostel(hostel)
            }
        }
    })
}

private fun showHostel(hostel:Hostel) {
    val coordinates : Coordinates = hostel.coordinates
    val hostelPosition = LatLng(coordinates.latitude, coordinates.longitude)

    mMap.addMarker(MarkerOptions().position(hostelPosition).title(hostel.title))

}
```

Рисунок 3.11 – Функция отображения маркеров каждого общежития

В результате получаем на экране смартфона фрагмент карты, где отмечены маркерами все общежития (рисунок 3.12).

Рисунок 3.12 – Скриншот приложения с фрагментом карты

3.5.4 Интеграция новостной ленты в Android-приложение

Для отображения новостей используем MVP (Model-View-Presenter) – один из самых популярных архитектурных шаблонов, используемых для разработки приложений. Он был создан для того, чтобы упростить тестирование модулей и разделить логику от общего представления.

MVP делает архитектуру проекта модульной, а значит она становится очень гибкой [8]. Это, в свою очередь, позволяет при необходимости добавлять новые функции быстрее, что в целом приводит к улучшению качества проекта.

Основные компоненты MVP:

- Model используется для работы с данными. Данные могут поступать либо с удаленного сервера, либо из базы данных. В нашем случае это будет сервер.
- View реализует отображение данных. Вся работа, связанная с пользовательским интерфейсом, происходит здесь.
- Presenter реализует взаимодействие между данными (моделью) и представлением (view).

Опишем модель, используемую для получения данных. Нам необходимо с помощью VK API получить текст новости, и изображение, прикрепленное к ней.

Для этого создали класс, наследуемый от класса `AsyncTask`. Класс `AsyncTask` предлагает простой и удобный механизм для перемещения трудоёмких операций в фоновый поток. Его следует использовать для не слишком продолжительных операций – загрузка небольших изображений, файловые операции, операции с базой данных и т.д [15]. В листинге (рисунок 3.13) представим код созданного класса.

Во фрагменте программы (см. рис. 3.13) получаем данные (текст и фотографию), и записываем их в список.

В результате работы Presenter и Views мы получаем отображение новостной ленты в нашем приложении (рисунок 3.14).

3.5.5 Взаимодействие с ИАС «Универис»

Планировалось, что некоторые данные о студентах (н-р: живет ли студент в общежитии) будут получены путем взаимодействия с системой «Универис». Однако в ходе разработки выяснилось, что разработчики системы исключили из системы API, позволяющий получить такие данные. Соответственно при отсутствии необходимого инструментария задача взаимодействия с ИАС «Универис» снимается.

```
private class Task internal constructor(
    private val posts: VKList<VKApiPost>?,
    private val callback: ConvertCallback
) : AsyncTask<Void, Void, ArrayList<NewsModel>>() {

    override fun doInBackground(vararg voids: Void): ArrayList<NewsModel>? {
        val news = ArrayList<NewsModel>()
        for (post in posts!!) {
            if (isCancelled) return null

            if (TextUtils.isEmpty(post.text) || TextUtils.isEmpty(getPhoto(post)))
                continue

            val textPost = post.text
            val photoPost = getPhoto(post)

            if(TextUtils.isEmpty(photoPost)) continue

            news.add(NewsModel(textPost, photoPost!!))
        }

        return news
    }

    override fun onPostExecute(news: ArrayList<NewsModel>?) {
        super.onPostExecute(news)
        if (news == null) return

        if (posts != null && posts.size != 0 && news.size == 0)
            callback.onFail()
        else
            callback.onConvert(news)

    }

    private fun getPhoto(post: VKApiPost): String? {
        if (post.attachments.count > 0) {
            for (attachment in post.attachments) if (attachment.type == "photo") {
                val photo = attachment as VKApiPhoto

                return when {
                    photo.photo_1280 != null -> photo.photo_1280
                    photo.photo_604 != null -> photo.photo_604
                    else -> null
                }
            }
        }
        return null
    }
}
```

Рисунок 3.13 – Класс `Task` с методами загрузки текста и фото новостей

Рисунок 3.14 – Скриншот отображения новостной ленты

3.5.6 Способ реализации электронных услуг в Android-приложении

В приложении необходимо реализовать следующие услуги:

- запись в прачечную;
- вызов сантехника;
- вызов электрика;
- вызов слесаря/плотника.

Для реализации подобного рода вещей хорошо подходит структура электронной очереди. Поэтому необходимо организовать запись на каждую из услуг, а также создать список активных услуг пользователя.

С этой целью создадим два адаптера: один будет отвечать за запись в прачечную (рисунок 3.15), другой за вызовы электрика, плотника, сантехника (рисунок 3.16).

В результате работы фрагментов программы, представленных на рисунках 3.15 и 3.16, имеем следующий результат (рисунок 3.17-3.19)

3.5.7 Структура проекта Web-сервера

При разработке архитектуры программного обеспечения системы мы уже сказали, что Backend нашей системы будет основан на шаблоне MVC. Соответственно и структура его проекта будет разбита на модели, представления и контроллеры. В целом Backend системы представлен Web-сервером и является основой всей системы, её ядром.

```
class QueueListAdapter (private val context: Context,
                        private val queues: ArrayList<QueueItem>,
                        private val itemClickListenerListener:ItemClickListener): RecyclerView.Adapter<QueueListAdapter.ViewHolder>(), View.OnClickListener {

    override fun onCreateViewHolder(container: ViewGroup, p1: Int): ViewHolder {
        val itemView = LayoutInflater.from(context).inflate(R.layout.queue_item, container,  attachToRoot: false)
        return ViewHolder(itemView)
    }

    override fun getItemCount(): Int {
        return queues.size
    }

    override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {
        viewHolder.bindDate(queues[position].title, queues[position].time)
        viewHolder.itemView.setOnClickListener { itemClickListenerListener.onClick(queues[position]) }
    }

    override fun onClick(v: View?) {

    }

    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        fun bindDate(title: String, time: String) {
            itemView.titleQueueView.text = title
            itemView.timeQueueView.text = time
        }
    }

    interface ItemClickListener {
        fun onClick(queueItem: QueueItem)
    }
}
```

Рисунок 3.15 – Реализация записи в прачечную

```
open class HomeQuestionsAdapter(private val context: Context,
                                private val homeQuestions: ArrayList<HomeQuestion>,
                                private val itemClickListenerListener:ItemClickListener): RecyclerView.Adapter<HomeQuestionsAdapter.ViewHolder>(),
                                                                                          View.OnClickListener {

    override fun onCreateViewHolder(container: ViewGroup, p1: Int): ViewHolder {
        val itemView = LayoutInflater.from(context).inflate(R.layout.home_questions_item, container,  attachToRoot: false)
        return ViewHolder(itemView)
    }

    override fun getItemCount(): Int {
        return homeQuestions.size
    }

    override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {
        viewHolder.bindDate(homeQuestions[position].title, homeQuestions[position].description)
        viewHolder.itemView.setOnClickListener { itemClickListenerListener.onClick(homeQuestions[position]) }
    }

    override fun onClick(v: View?) {

    }

    class ViewHolder( itemView: View) : RecyclerView.ViewHolder(itemView) {
        fun bindDate(title:String, description:String) {
            itemView.titleView.text = title
            itemView.descriptionView.text = description
        }
    }

    interface ItemClickListener {
        fun onClick(homeQuestion: HomeQuestion)
    }

}
```

Рисунок 3.16 – Реализация вызова электрика/плотника/сантехника

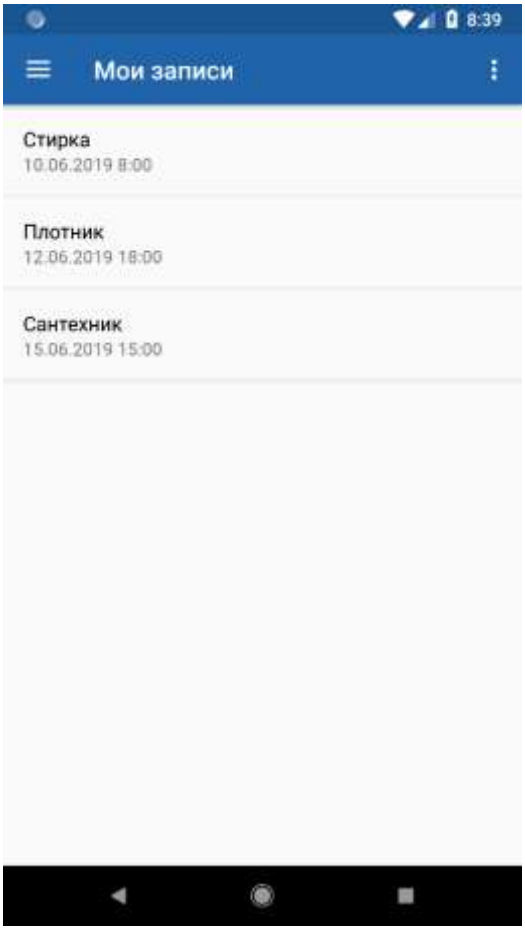

Рисунок 3.17 – Интерфейс активных услуг пользователя

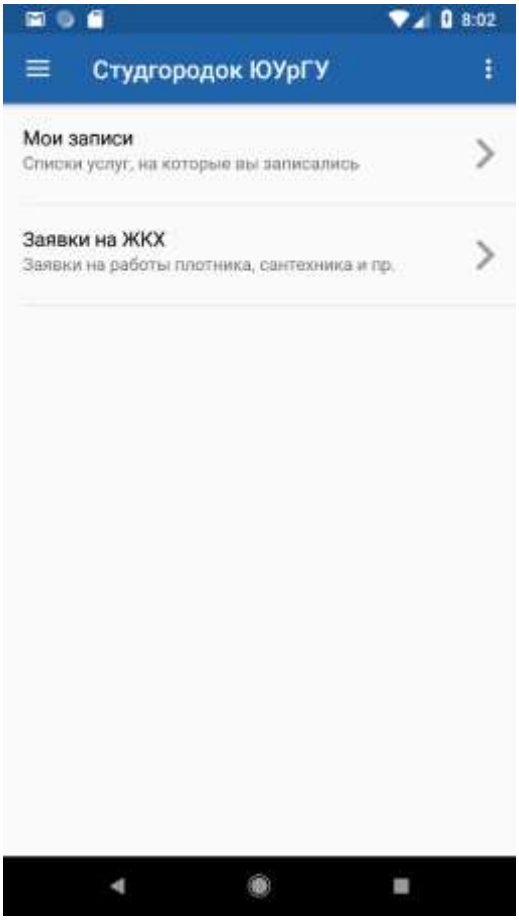

Рисунок 3.18 – Интерфейс выбора услуги

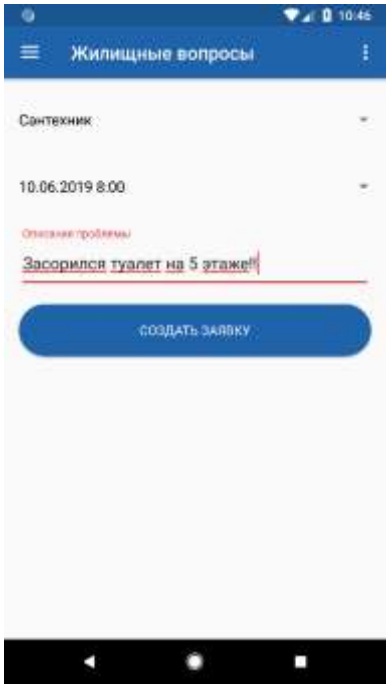

Рисунок 3.17 – Интерфейс подачи заявки на услугу

Именно он определяет большую часть функционала системы, отвечает за ее гибкость и масштабируемость, таким образом, при необходимости есть возможность расширить функционал системы, практически не затрагивая ее основные компоненты. На рисунке 3.20 представлена структура проекта Web-сервера.

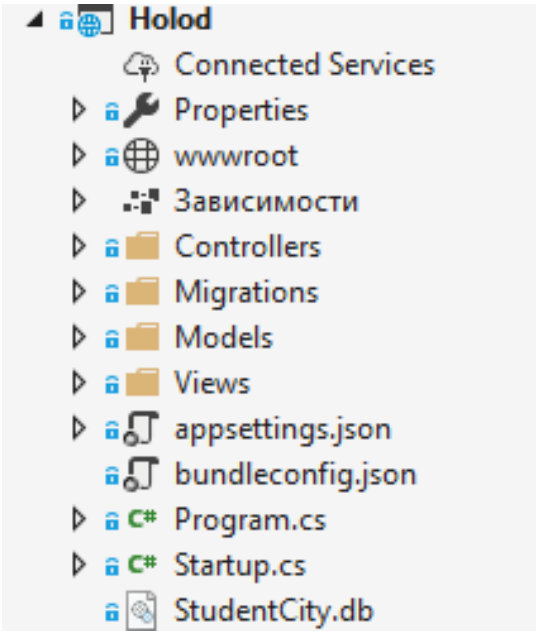

Рисунок 3.20 – Структура проекта Web-сервера

Рассмотрим подробнее структуру проекта (см. рис.3.20).

- **Зависимости –** все добавленные в проект пакеты и библиотеки
- **wwwroot** – этот узел предназначен для хранения статических файлов – скриптов javascript, изображений, css файлов и т.д., которые используются приложением.
- **Controllers** – папка для хранения контроллеров, используемых приложением
- **Models** – каталог для хранения моделей.
- **Views** – каталог для хранения представлений.
- **appsettings.json** – хранит конфигурацию приложения.
- **bundleconfig**.json –файл соединяет все существующие в проекте скрипты и стили в один файл (бандл).
- **Program.cs** – файл, определяющий класс Program, который инициализирует и запускает хост с приложением.
- **Startup.cs** – файл, определяющий класс Startup, с которого начинается работа приложения. То есть это точка входа в приложение.
- **StudentCity.db** – созданная нами база данных.
- **Migrations** – содержит файлы связывающие БД и модели.

3.5.8 Реализация электронных услуг на уровне Backend

Для создания электронной очереди, с помощью которой в данной системе реализуется запись в прачечную, вызов электрика, плотника или сантехника. Необходимо создать в Backend проекте соответствующие классы моделей (рисунок 3.21).

```
public class Queue
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<CellQueue> Cells { get; set; }
    [ForeignKey("HostelId")]
    public int HostelId { get; set; }
    public Hostel Hostel { get; set; }
}

public class CellQueue
{
    public int Id { get; set; }
    public DateTime RecordingTime { get; set; }
    [ForeignKey("ResidentId")]
    public int ResidentId { get; set; }
    public Resident Resident { get; set; }
    [ForeignKey("QueueId")]
    public int QueueId { get; set; }
    public Queue Queue { get; set; }
}
```

Рисунок 3.21 – Модель электронной очереди

3.6 Тестирование системы

Во время разработки работоспособность всего функционала Android-клиента тестировалась при его создании. Однако подобного тестирования, разумеется, недостаточно. Необходимо, чтобы испытания проводились группой людей. Так можно выяснить, какие недостатки присутствуют в работе системы в целом, в отдельных фрагментах системы, в пользовательском интерфейсе и т.д.

Наилучший вариант тестирования автоматизированной системы электронных услуг студенческого городка ЮУрГУ – тестирование системы на базе студенческих советов общежитий. Так как студенческий совет – это наиболее заинтересованный и активный круг лиц, проживающих в общежитиях. К тому же, так как в системе присутствуют элементы популяризации культурно-массовых мероприятий, которые организуются студенческим совет студгородка ЮУрГУ, их замечания к системе крайне важны.

После тестирования системы и отладки всех найденных недостатков необходимо будет приступить к публикации клиентского Android приложения. Наиболее популярная платформа для размещения Android-приложений – Google Play Market. Стоимость размещения приложения там равна 25 долларов.

Вывод по главе три

1. Система электронных …услуг разработана как продукт, имеющий трехуровневую архитектуру типа «клиент-сервер», и состоящий из компонентов: …, работоспособность которых определяется следующим:
   - подключением сторонних библиотек;
   - изменением файла манифеста;
   - учетом различий основных компонентов Android-приложений, а также нюансов работы Activity в операционной системе Android.

2. Для информационного наполнения системы сформирован банк информационных ресурсов. Архитектура базы данных, реализована при помощи СУБД SQLite.

3. Система электронных …услуг реализована с использованием языка Kotlin и среды разработки Android Studio; платформы ASP .NET Core и среды разработки Web-сервера (Microsoft Visual Studio 2017 Community), а также с использованием средств разметки пользовательских интерфейсов XML, HTML, CSS. Система требует наличие у пользователя устройства с операционной системой Android версии 5.0 и выше.

4. Перед публикацией на сервисе Google Play Market система электронных услуг для объектов студенческого городка ЮУрГУ проходит тестирование группой пользователей (совет студенческого городка ЮУрГУ.

ЗАКЛЮЧЕНИЕ

1. Анализ известных на рынке систем электронных услуг, выполненных как мобильные приложения для отелей («Radisson Rewards», «Best Western To Go», «NH Hotel Group – Book your hotel», «Melia – Hotel Bookings & more»), показал, что они не могут быть использованы как системы электронных услуг для жителей общежития (у них нет требуемого функционала), а приложения, разработанные для студентов ВУЗов в большинстве своем, являются простой копией сайта ВУЗа, где требуемый функционал (запись на стирку, подача заявки электрику, слесарю, плотнику, оцифрованный пропуск) не реализована.

2. В качестве прототипа автоматизированной системы электронных услуг для объектов студгородка взят продукт «Melia – Hotel Bookings & more», в котором не реализованы: просмотр новостной ленты; формирование электронной очереди на оказание коммунальных услуг, оцифровка пропусков.

3. Система электронных …услуг разработана как продукт, имеющий трехуровневую архитектуру типа «клиент-сервер», и состоящий из компонентов: …, работоспособность которых определяется следующим: подключением сторонних библиотек; изменением файла манифеста; учетом различий основных компонентов Android-приложений, а также нюансов работы Activity в операционной системе Android.

4. Для информационного наполнения системы сформирован банк информационных ресурсов. Архитектура базы данных, реализована при помощи СУБД SQLite.

5. Система электронных …услуг реализована с использованием языка Kotlin и среды разработки Android Studio; платформы ASP .NET Core и среды разработки Web-сервера (Microsoft Visual Studio 2017 Community), а также с использованием средств разметки пользовательских интерфейсов XML, HTML, CSS. Система требует наличие у пользователя устройства с операционной системой Android версии 5.0 и выше.

6. Перед публикацией на сервисе Google Play Market система электронных услуг для объектов студенческого городка ЮУрГУ проходит тестирование группой пользователей (совет студенческого городка ЮУрГУ.

7. Автоматизация системы информационного обеспечения жителей студенческого городка ЮУрГУ, обеспечивает: упрощение процедур записи на оказание услуг; расширение целевой аудитории, улучшение качества информирования о культурно-массовых мероприятиях; изменение формы доступа (введение оцифрованных пропусков).

8. Затраты на разработку системы и обеспечение её корректного функционирования составляют: 170 р./месяц за аренду хостинга для размещения на нем Web-сервера; $25 ≈ 1700 р. за публикацию приложения на сервисе Google Play Market.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 3-Tier Architecture: A Complete Overview. [Электронный ресурс]. – Режим доступа: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/ – Заглавие с экрана

2 Darwin, I.F. Android Cookbook. Problems and Solutions for Android Developers / I.F. Darwin. – O'Reilly Media, Inc., 2017. – 766 p.

3 Documentation for app developers. [Электронный ресурс]. – Режим доступа: https://developer.android.com/docs – Заглавие с экрана

4 Google Maps Platform Documentation. [Электронный ресурс]. – Режим доступа: https://developers.google.com/maps/documentation/?hl=ru – Заглавие с экрана

5 Harvey, C. Top Android IDEs for Developers. [Электронный ресурс]. – Режим доступа: https://www.developer.com/ws/android/development-tools/top-android-ides-for-developers.html – Заглавие с экрана

6 Meier, R. Professional Android, Fourth Edition. / R. Meier, I. Lake. – John Wiley & Sons, Inc., Indianapolis, Indiana, 2018. – 911 p.

7 Miloš Vasić. Mastering Android Development with Kotlin. / Miloš Vasić – Packt Publishing, 2017. – 421 p.

8 Model-View-Presenter: Our Choice of Architecture for Your Android App. [Электронный ресурс]. – Режим доступа: https://steelkiwi.com/blog/model-view-presenter-our-choice-of-android-app/ – Заглавие с экрана

9 Murphy, M. L. The Busy Coder's Guide to Android Development. / M.L. Murphy – CommonsWare, LLC, 2019. – 4236 p.

10 Suamont, P. The Joy of Kotlin. / P. Saumont – Manning Publications Co, 2018. – 480 p.

11 Understand the Activity Lifecycle. [Электронный ресурс]. – Режим доступа: https://developer.android.com/guide/components/activities/activity-lifecycle.html – Заглавие с экрана

12 What is MVC? [Электронный ресурс]. – Режим доступа: https://www.quora.com/What-is-MVC-1 – Заглавие с экрана

13 Албахари, Д. C# 7.0. Справочник. Полное описание языка / Д. Албахари, Б. Албахари. – М.: ООО «И.Д. Вильямс», 2018. – 1008 с.

14 Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт ; [пер. с англ. М.А. Райтмана]. – М.: Эксмо, 2017. – 480 с.

15 Класс AsyncTask. [Электронный ресурс]. – Режим доступа: http://developer.alexanderklimov.ru/android/theory/asynctask.php – Заглавие с экрана

16 Паттерны для новичков: MVC vs MVP vs MVVM. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/215605/ – Заглавие с экрана

17 Портал государственных услуг Российской Федерации. [Электронный ресурс]. – Режим доступа: https://www.gosuslugi.ru – Заглавие с экрана

18 СТО ЮУрГУ 04–2008 Стандарт организации. Курсовое и дипломное проектирование. Общие требования к содержанию и оформлению / составители: Т.И. Парубочая, Н.В. Сырейщикова, В.И. Гузеев, Л.В. Винокурова. – Челябинск: Изд-во ЮУрГУ, 2008. – 56 с.

19 Технологии Яндекса. [Электронный ресурс]. – Режим доступа: https://tech.yandex.ru/#catalog – Заглавие с экрана

20 Хлебенских, Л. В. Автоматизация производства в современном мире / Л.В. Хлебенских, М.А. Зубкова, Т.Ю. Саукова // Молодой ученый. – 2017. – С. 308-311.

21 Чистякова, Е.М. Современные методы экономии материальных ресурсов / Е.М. Чистякова. // Научное сообщество студентов XXI столетия. Экономические науки: сборник статей по материалам LXI студенческой международной научно-практической конференции. – Новосибирск: Изд. АНС «СибАК». – 2018. – С. 439-444.

22 Чамберс, Д. ASP .NET Core. Разработка приложений. / Д. Чамберсб, Д. Пэкет, С. Тиммс. – СПБ.: Питер, 2018. – 464 с.

23 SQLite – замечательная встраиваемая БД. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/149356/ – Заглавие с экрана

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. ИНФОРМАЦИОННЫЕ РЕСУРСЫ

Таблица с координатами общежитий (рисунок А.1).

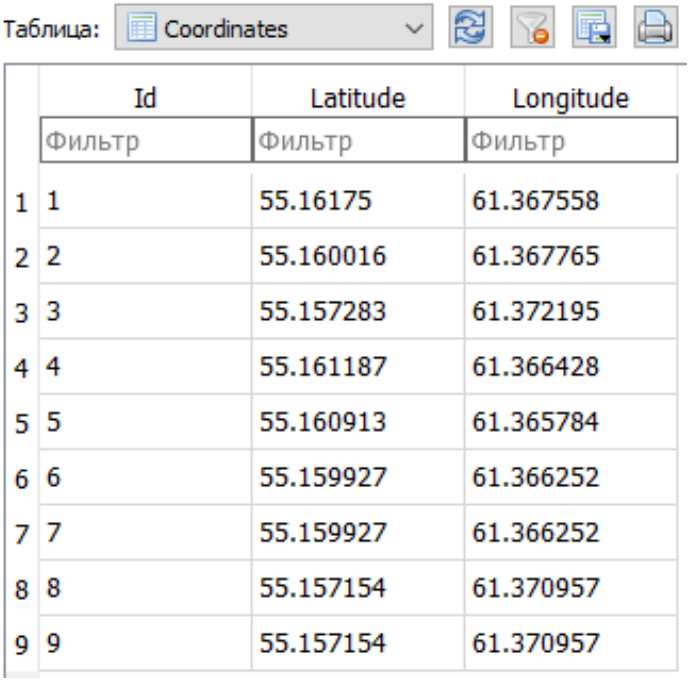
Таблица: Coordinates

	Id	Latitude	Longitude
	Фильтр	Фильтр	Фильтр
1	1	55.16175	61.367558
2	2	55.160016	61.367765
3	3	55.157283	61.372195
4	4	55.161187	61.366428
5	5	55.160913	61.365784
6	6	55.159927	61.366252
7	7	55.159927	61.366252
8	8	55.157154	61.370957
9	9	55.157154	61.370957

Рисунок А.1 – Таблица Coordinates

Таблица с информацией об общежитиях (рисунок А.2).

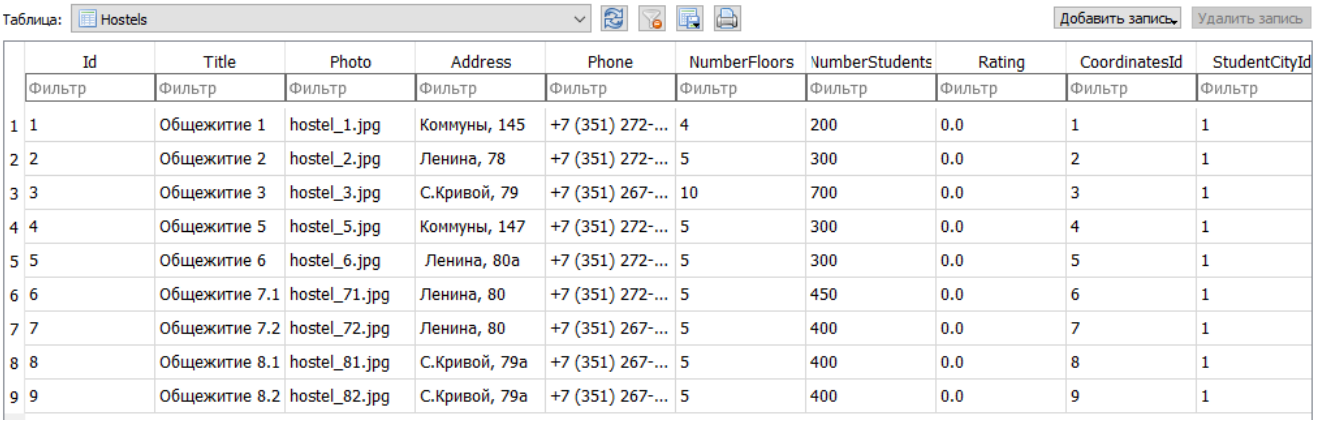
Таблица: Hostels

Добавить запись | Удалить запись

	Id	Title	Photo	Address	Phone	NumberFloors	NumberStudents	Rating	CoordinatesId	StudentCityId
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Общежитие 1	hostel_1.jpg	Коммуны, 145	+7 (351) 272-...	4	200	0.0	1	1
2	2	Общежитие 2	hostel_2.jpg	Ленина, 78	+7 (351) 272-...	5	300	0.0	2	1
3	3	Общежитие 3	hostel_3.jpg	С.Кривой, 79	+7 (351) 267-...	10	700	0.0	3	1
4	4	Общежитие 5	hostel_5.jpg	Коммуны, 147	+7 (351) 272-...	5	300	0.0	4	1
5	5	Общежитие 6	hostel_6.jpg	Ленина, 80а	+7 (351) 272-...	5	300	0.0	5	1
6	6	Общежитие 7.1	hostel_71.jpg	Ленина, 80	+7 (351) 272-...	5	450	0.0	6	1
7	7	Общежитие 7.2	hostel_72.jpg	Ленина, 80	+7 (351) 267-...	5	400	0.0	7	1
8	8	Общежитие 8.1	hostel_81.jpg	С.Кривой, 79а	+7 (351) 267-...	5	400	0.0	8	1
9	9	Общежитие 8.2	hostel_82.jpg	С.Кривой, 79а	+7 (351) 267-...	5	400	0.0	9	1

Рисунок А.2 – Таблица Hostels

Таблица с названиями должностей (рисунок А.3)

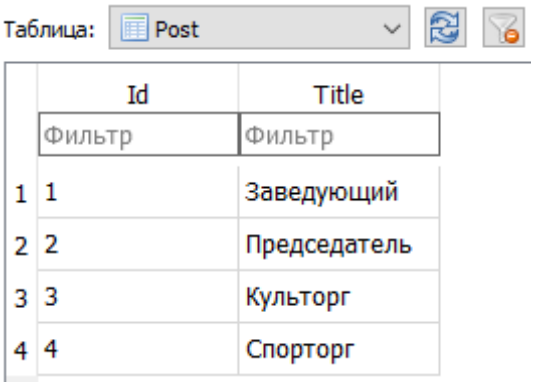
Таблица: Post

	Id	Title
	Фильтр	Фильтр
1	1	Заведующий
2	2	Председатель
3	3	Культорг
4	4	Спорторг

Рисунок А.3 – Таблица Post

Таблица с информацией о заведующих и членах студсоветов общежитий (рисунок А.4).

Таблица: Stuffs

	Id	Surname	Firstname	Patronymic	Photo	HostelId	StudentCityId	PostId
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Былкова	Ольга	Александровна	1.jpg	1	1	1
2	2	Аркашева	Валентина	Георгиевна	2.jpg	2	1	1
3	3	Селюнина	Мария	Васильевна	3.jpg	3	1	1
4	4	Кожевникова	Ольга	Николаевна	5.jpg	4	1	1
5	5	Гарбовская	Надежда	Михайловна	6.jpg	5	1	1
6	6	Зайцева	Антонина	Алексеевна	71.jpg	6	1	1
7	7	Приходько	Юлия	Владимировна	72.jpg	7	1	1
8	8	Пастухова	Наталья	Петровна	81.jpg	8	1	1
9	9	Иванова	Ольга	Николаевна	82.jpg	9	1	1
10	10	Шелков	Владимир	Михайлович	1_predsed.jpg	1	1	2
11	11	Горяева	Ксения	Андреевна	1_cultorg.jpg	1	1	3
12	12	Коломиец	Виктория	Юрьевна	1_sportorg.jpg	1	1	4
13	13	Гимаева	Вероника	Альбертовна	2_predsed.jpg	2	1	2
14	14	Камалов	Рамис	Азаматович	2_sportorg.jpg	2	1	4
15	15	Трусова	Екатерина	Сергеевна	2_cultorg.jpg	2	1	3
16	16	Пищик	Кирилл	Иванович	6_predsed.jpg	5	1	2
17	17	Жигайло	Кирилл	Денисович	6_cultorg.jpg	5	1	3
18	18	Цыплева	Екатерина	Алексеевна	6_sportorg.jpg	5	1	4
19	19	Самойлова	Ксения	Константинов...	81_predsed.jpg	8	1	2
20	20	Маркина	Юлия	Алексеевна	81_cultorg.jpg	8	1	3
21	21	Тимаев	Наиль	Ильдарович	81_sportorg.jpg	8	1	4
22	22	Минаева	Анжелетта	Александровна	72_predsed.jpg	7	1	2

Рисунок А.4 – Таблица Stuff