|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУ-КФ «Информатика и управление»\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_ ***ИУ5-КФ «Системы обработки информации»***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

***Проектирование и разработка мобильного приложения для навигации по университету***

по дисциплине ***Математическая логика и теория алгоритмов***

Студент гр. ИУК5-81Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Щербаков Д. В.)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Федоров В. О.)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2021 г.

Калужский филиал   
федерального государственного бюджетного образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»   
(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУ5-КФ\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Вершинин Е. В.)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине ***Математическая логика и теория алгоритмов***

Студент **Щербаков Д. В. ИУК5-81Б**

(фамилия, инициалы, индекс группы)

Руководитель **Федоров В. О.**

(фамилия, инициалы)

График выполнения проекта: 25% к\_4\_нед., 50% к\_7\_нед., 75% к\_10\_нед., 100% к\_12\_нед.

***1. Тема курсового проекта***

***Проектирование и разработка мобильного приложения для навигации по университету***

***2. Техническое задание***

*«Разработать приложение для студентов, которое поможет им ориентироваться по университету и следить за их учебным планом. Клиентское приложение должно иметь следующий функционал: аккаунт пользователя с информацией о нем, интерактивная двухмерная схема корпусов университета. Пользователь может ввести в поиск любую аудиторию и проложить до нее маршрут, ему будет показано расстояние до аудитории.»*

***3. Оформление курсового проекта***

3.1. Расчетно-пояснительная записка на\_\_\_\_\_\_ листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.) \_презентация,

графический материал на листах А1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Руководитель курсового проекта \_\_\_ \_/\_ Федоров В. О.\_\_/

(подпись) (Ф.И.О.)

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_ Щербаков Д. В.\_\_\_\_\_\_/ «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2021\_г.

(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc70032632)

[1. Техническое задание 6](#_Toc70032633)

[1.1.1 Полное наименование системы 6](#_Toc70032634)

[1.1.2 Краткая характеристика области применения 6](#_Toc70032635)

[1.1.3 Заказчик 6](#_Toc70032636)

[1.1.4 Исполнитель 6](#_Toc70032637)

[1.1.5 Плановые сроки 6](#_Toc70032638)

[1.1.6 Порядок оформления и предъявления заказчику результатов работ по созданию системы 6](#_Toc70032639)

[1.2 Назначение и цели создания системы 7](#_Toc70032640)

[1.2.1 Назначение системы 7](#_Toc70032641)

[1.2.2 Цели создания системы 7](#_Toc70032642)

[1.3 Характеристики объекта автоматизации 7](#_Toc70032643)

[1.4 Требования к системе 7](#_Toc70032644)

[1.4.1 Требования к функциям, выполняемым системой 7](#_Toc70032645)

[1.4.2 Требования к пользовательскому интерфейсу 8](#_Toc70032646)

[1.4.3 Требования к надежности и безопасности 8](#_Toc70032647)

[1.4.4 Требования к эргономике и технической эстетике 8](#_Toc70032648)

[1.4.5 Требования по сохранности информации 9](#_Toc70032649)

[1.4.6 Требования к техническим средствам 9](#_Toc70032650)

[1.4.7 Требования к программному обеспечению 9](#_Toc70032651)

[1.4.8 Требования к математическому обеспечению 10](#_Toc70032652)

[1.5 Состав и содержание работ по созданию и развитию системы 10](#_Toc70032653)

[1.6 Порядок контроля и приёмки системы 11](#_Toc70032654)

[1.6.1 Методы испытания системы 11](#_Toc70032655)

[1.6.2 Общие требования к приёмке работы по стадиям 12](#_Toc70032656)

[1.6.3 Статус приемочной комиссии 12](#_Toc70032657)

[1.7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие 12](#_Toc70032658)

[1.8 Требования к документированию 12](#_Toc70032659)

[1.9 Источники разработки 12](#_Toc70032660)

[2. Исследовательская часть 14](#_Toc70032661)

[2.1. Постановка задачи проектирования 14](#_Toc70032662)

[2.2. Описание предметной области 14](#_Toc70032663)

[2.3. Анализ существующих аналогов 15](#_Toc70032664)

[2.4. Выбор системы позиционирования 19](#_Toc70032665)

[2.5. Перечень задач, подлежащих решению в процессе разработки 20](#_Toc70032666)

[2.6. Обоснование выбора платформы для разработки 21](#_Toc70032667)

[2.7. Выбор инструментов для разработки 21](#_Toc70032668)

[2.8. Алгоритм поиска кротчайшего пути 22](#_Toc70032669)

[2.9. Модели поведения мобильных пользователей 25](#_Toc70032670)

[3. Проектно-конструкторская часть 28](#_Toc70032671)

[3.1. Модель данных 28](#_Toc70032672)

[3.2. Архитектура приложения 29](#_Toc70032673)

[3.3. Разработка структуры приложения 29](#_Toc70032674)

[3.4. Разработка пользовательского интерфейса 29](#_Toc70032675)

[3.4.1. Сценарий поведения пользователя 30](#_Toc70032676)

[3.4.2. Основные элементы пользовательского интерфейса 31](#_Toc70032677)

[3.4.3. Создание прототипов и тестирование интерфейса 31](#_Toc70032678)

[3.4.4. Создание меню навигации 32](#_Toc70032679)

[3.5. Подключение базы данных 36](#_Toc70032680)

[3.5.1. Регистрация пользователя 37](#_Toc70032681)

[3.6. Реализация навигации по приложению 39](#_Toc70032682)

[3.7. Реализация алгоритма поиска кратчайшего пути 41](#_Toc70032683)

[3.8. Реализация схемы университета 41](#_Toc70032684)

[4. Проектно-технологическая часть 45](#_Toc70032685)

[4.1. Руководство пользователя 45](#_Toc70032686)

[4.2. Порядок внедрения приложения 52](#_Toc70032687)

[ЗАКЛЮЧЕНИЕ 53](#_Toc70032688)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc70032689)

[ПРИЛОЖЕНИЕ А 55](#_Toc70032690)

# ВВЕДЕНИЕ

В современном мире системы точной навигации стали неотъемлемой частью нашей жизни. С различными системами навигации человек сталкивается постоянно. В повседневной жизни, с помощью смартфона мы можем проложить маршрут в навигаторе, заказать такси с помощью приложения и посмотреть, где мы находимся, а где едет водитель.

Наш смартфон позволяет быстрее и комфортнее ориентироваться в незнакомом месте.

Цель данной работы – разработка приложения для локального позиционирования внутри университета.

Каждый год в университеты страны поступают тысячи абитуриентов и все они оказываются в незнакомой обстановке. У каждого студент сейчас есть смартфон, поэтому приложение на него поможет решить ряд проблем позиционирования внутри университета. Поскольку новые студенты поступают в университет каждый год, приложение будет востребовано и использоваться на постоянной основе.

Мобильное приложение для навигации по университет будет актуально, так как других аналогичных решений в стенах ВУЗов не предусмотрено. Единственное что можно увидеть в университетах это таблички со стрелочками, которые указывают направления в какой стороне корпус, но они дают минимальную информацию.

# Техническое задание

* 1. **Общие сведения**

# Полное наименование системы

Наименование программы – «Навигация по университету».

# Краткая характеристика области применения

Данное приложение может быть использовано в любых учебных заведениях, которые имеют большую площадь и требуют дополнительных способов навигации.

# Заказчик

КФ МГТУ им. Н.Э. Баумана.

# Исполнитель

Студент КФ МГТУ им. Н.Э. Баумана Щербаков Д.В., гр. ИУК5-81Б.

# Плановые сроки

Плановые сроки по созданию системы:

Начало работ – 1 сентября 2020 года.

Окончание работ – 1 июня 2021 года.

# Порядок оформления и предъявления заказчику результатов работ по созданию системы

По мере разработки приложения результаты будут показываться научному руководителю на консультациях по дипломной работе.

# Назначение и цели создания системы

# Назначение системы

Разрабатываемая система поможет студентам ориентироваться по университету.

# Цели создания системы

Разработать систему навигации по университету, что позволит:

1. Уменьшить время на поиск необходимых объектов (аудитории, кафедры, деканата, столовой и так далее).
2. Уменьшить время передвижения по университету за счет выбора оптимальных маршрутов.

# Характеристики объекта автоматизации

Объектом автоматизации является поиск оптимального маршрута.

# Требования к системе

# Требования к функциям, выполняемым системой

Приложение должно обеспечивать реализацию следующих функций:

1. Аккаунт пользователя, в котором будут его персональные данные: ФИО студента, название группы, почта.
2. Выбор начальной и конечной точки маршрута на схеме университета, построение кротчайшего пути между ними.
3. Расчет времени и расстояния при построении маршрута, отображение их на экране.
4. Функция поиска по точкам маршрута (аудиториям и другим объектам).
5. Сканирование QR кода для определения местоположения.

6) Отображение расписание пользователя.

# Требования к пользовательскому интерфейсу

Пользовательский интерфейс должен быть оптимизирован под использование на мобильных платформах.

Синтаксическая структура, реализованная в интерфейсе должна постоянно находиться под контролем пользователя, никакие действия последнего не должны приводить к тупиковой ситуации или зависанию программы.

Интерфейс приложения должен обеспечивать возможность лёгкого исправления ошибок ввода, не должен требовать повторного ввода данных.

Левый верхний угол необходимо использовать для размещения важной информации, элементов управления и поиска, в то время как нижняя граница экрана - для навигации.

Так как большую часть времени пользователи держат устройство правой рукой и используют для работы с экраном только большой палец, необходимо продумать эргономичную компоновку важных элементов приложения для качественного использования и восприятия информации.

Так как пользователь может использовать приложение на ходу, необходимо сделать контрастные иконки, а также расположить противоположенные по контексту значки на расстоянии друг от друга, что поможет избежать случайных нажатий.

# Требования к надежности и безопасности

Гарантированное построение существующего и оптимального маршрута.

# Требования к эргономике и технической эстетике

Для удобства и соблюдения единого стиля необходимо использовать один шрифт, а для выделения отдельных частей контента менять кегль шрифта.

Переключение между страницами приложения должно осуществляться с помощью нижнего меню навигации, в котором будут основные страницы: профиль, карта, расписание, камера. При переходе на страницу карты, все элементы управления и навигации должны быть реализованы в выдвижном меню, чтобы освободить пространство экрана.

Все страницы приложения должны быть реализованы в единой цветовой палитре. Все текстовые надписи должны быть на русском языке. Цвета, строки, стили должны быть вынесены в отдельные xml файлы (colors, strings, styles).

Все классы, отвечающие за управление экранами, должны называться – НазваниеActiviy. Xml файлы экранов должны называться activity\_название, фрагментов – fragment\_название.

# Требования по сохранности информации

Осуществить настройку статуса и активности резервного копирования сервиса Firebase.

# Требования к техническим средствам

Для клиента устройство на операционной системе Android минимальная версии 4.4 уровня API 19, рекомендованная версия 9.0 уровня API 28.

# Требования к программному обеспечению

Использование сервиса Google Firebase для хранения расписания студентов, регистрации и авторизации пользователей.

В качестве среды разработки должна быть использована Android Studio. В проект Android Studio необходимо добавить следующие библиотеки:

Material Design – для элементов навигации.

Firebase-database – для базы данных firebase.

Firebase-auth – для авторизации пользователя.

Приложение должно быть написано на языке Java.

# Требования к математическому обеспечению

Для поиска оптимального пути необходимо использовать граф, поиск по которому будет осуществляться одним из существующих алгоритмов обхода взвешенного графа.

# Состав и содержание работ по созданию и развитию системы

Таблица 1 – Календарный план

|  |  |  |
| --- | --- | --- |
| Стадии | Этапы работ | Даты |
| 1. Формирование требований к АС | 1.1. Обследование объекта и обоснование необходимости создания АС | 7 сентября 2020 года |
| 1.2. Формирование требований пользователя к АС | 14 сентября 2020 года |
| 1.3. Оформление отчета о выполненной работе и заявки на разработку АС | 21 сентября 2020 года |
| 2. Разработка концепции АС | 2.1. Изучение объекта | 1 октября 2020 года |
|  | 2.2. Проведение необходимых научно-исследовательских работ | 8 октября 2020 года |
|  | 2.3. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя | 15 октября 2020 года |
|  | 2.4. Оформление отчета о выполненной работе | 23 октября 2020 года |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| 3. Техническое задание | 3.1. Разработка и утверждение технического задания на создание АС | 30 октября 2020 года |
| 4. Эскизный проект | 4.1. Разработка предварительных проектных решений по системе и ее частям | 1 ноября 2020 года |
|  | 4.2. Разработка документации на АС и ее части | 8 ноября 2020 года |
| 5. Технический проект | 5.1. Разработка проектных решений по системе и ее частям | 1 декабря 2020 года |
|  | 5.2. Разработка документации на АС и ее части | 15 декабря 2020 года |
|  | 5.3. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации | 1 января 2021 года |
| 6. Рабочая документация | 6.1. Разработка рабочей документации на систему и ее части | 1 апреля 2021 года |
|  | 6.2. Разработка или адаптация программ | 14 апреля 2021 года |

# Порядок контроля и приёмки системы

# Методы испытания системы

В процессе разработки APK файлы приложения загружаются и тестируются на Android устройствах разных версий API.

# Общие требования к приёмке работы по стадиям

Место проведения приемки работ КФ МГТУ им. Н.Э. Баумана.

Сроки проведения приёмки работ – консультации в соответствии с расписанием, а также другие даты, согласованные с заказчиком и исполнителем.

# Статус приемочной комиссии

В окончательной приемке работы участвует Государственная аттестационная комиссия.

# Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Для ввода приложения в действие необходимо опубликовать приложение в сервисе Google Play.

# Требования к документированию

Разработку системы требуется сопровождать расчетно-пояснительной запиской и графической частью.

Расчетно-пояснительная записка должна состоять из четырех частей:

1. Техническое задание.
2. Исследовательская часть.
3. Проектно-конструкторская часть.
4. Проектно-технологическая часть.

Графическая часть должна состоять из:

1. Семи или более листов формата А1 графического материала.
2. Презентация.

# Источники разработки

**ГОСТ 34.602-89** Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

**ГОСТ 34.601-90** Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.

# Исследовательская часть

# Постановка задачи проектирования

Цель данной работы заключается в проектировании системы навигации по университет, так как других аналогичных решений в стенах ВУЗа не предусмотрено.

Поскольку новые студенты поступают в университет каждый год, приложение будет востребовано и использоваться на постоянной основе.

Данное решение позволит новым студентам экономить время на ориентирование в незнакомом месте, а остальным лучше планировать свое время и проще контролировать учебный план, поэтому приложение будет актуально не только в начале, но и в течение учебного года.

# Описание предметной области

Существует два основных раздела в области навигация – это навигация внешняя (outdoor) и внутренняя (indoor).

Outdoor навигация осуществляется с помощью GPS модуля на основании данных со спутника. Примеры такой навигации - это карты от компании Google и Yandex.

Эти готовые решения способны решить основную долю проблем позиционирования на местности и построения оптимальных маршрутов. Но когда дело доходит до навигации в закрытых помещениях, то данные решения уже не применимы, на что влияют следующие факторы.

Сами разработчики заявляют о погрешности определения местоположения в несколько метров. Когда речь идет о широких улицах и расстояниях в сотни метров или даже километров, на эту погрешность никто не обратит внимание, но в рамках одного здания погрешность даже в пару метров может перенести положение пользователя по-другому адресу.

Окна, кирпичные стены, бетонные перегородки и перекрытия создают большой уровень помех, поэтому система GPS будет совсем не эффективна.

На примере университета можно это отчетливо увидеть (рис. 1).

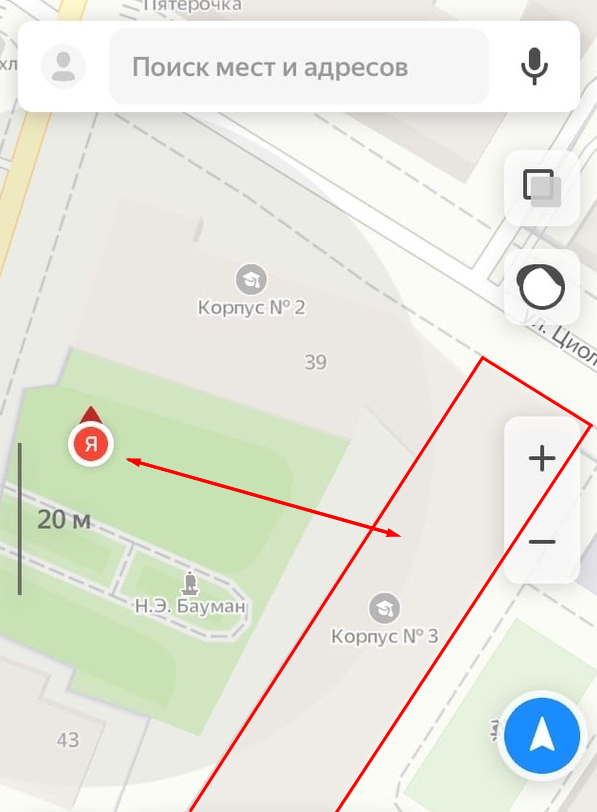


Рисунок 1 – Неточное положение при использовании GPS

Поэтому существует другая сторона этой области, а именно indoor навигация, список технологий которой будет рассмотрен в дальнейшем.

# Анализ существующих аналогов

**Indoors** – точная система позиционирования в помещении в реальном времени.

Indoo.rs предоставляет профессиональные [решения для внутреннего позиционирования в реальном времени для мобильных приложений](https://indoo.rs/solution/indoor-positioning-system/). Эта передовая технология основана на сочетании интеллектуальных алгоритмов, iBeacon и датчиков смартфонов и обеспечивает надежные результаты позиционирования в любых условиях внутри помещения.

Indoo.rs эффективно реализовал проекты Mobile Wayfinding по всему миру, в различных отраслях, от [здравоохранения](https://indoo.rs/industries/healthcare/) до [транспорта](https://indoo.rs/industries/transport/), [мероприятий](https://indoo.rs/industries/events/), [розничной торговли](https://indoo.rs/industries/retail/) и т. д. Примеры некоторых из последних проектов с крупными и известными клиентами, такими как [Amtrak](https://indoo.rs/indoor-navigation-penn-station/), [Kaleida Health](https://indoo.rs/mobile-wayfinding-kaleida-health/) или [Desjardins](https://indoo.rs/desjardins-office-digitalization-with-indoor-positioning/).

[Команда](https://indoo.rs/solution/professional-services/) indoors обеспечивает бесперебойное управление проектом и точную установку на месте для наиболее точной точки на карте.

**Три столпа успешного опыта indoo.rs**

Внутреннее позиционирование

Обеспечьте мобильную навигацию, чтобы улучшить ориентацию внутри вашего здания и повысить комфорт для пользователей.

Внутренняя аналитика

Получите представление о закономерностях движения и поведении посетителей, чтобы оптимизировать планировку места проведения, маркетинговые кампании и цены в зависимости от местоположения [5].

Внутреннее картографирование

Преобразуйте свой статический план этажа в интерактивные цифровые карты для интеграции в мобильные приложения.

Данная система несмотря на точность позиционирования имеет ряд недостатков. Перовое – это высокие тарифные планы стандарт - $499, профессионал – $999.

Также, самое большое ограничение приложение – оно доступно только для платформы Android (ввиду ограничений на устройствах Apple для работы с Bluetooth стеком). Поэтому при разработке версии приложения для IOS будет необходимо разрабатывать новую систему с самого начала.

**NAVIGINE SDK**

Компания Navigine предлагает разработчикам программный комплекс SDK для внутренней навигации в помещении, который внедряется в мобильные приложения для расширения их функциональных возможностей. Navigine SDK представляет собой набор алгоритмов, решающих широкий спектр задач на базе геоданных. С его помощью можно создавать удобные приложения для смартфонов на Android и IOS, оснащенные функцией indoor-навигации внутри зданий.

Возможности приложения, оснащенного navigine sdk

Комплект алгоритмов открывает неограниченные возможности в отслеживании перемещения людей и объектов внутри помещений. Благодаря предлагаемым инструментам можно создавать передовые приложения для смартфонов с применением технологии iBeacon. К ключевым возможностям программной платформы относятся:

Определение месторасположения точностью до 3–5 метров.

Точное местонахождение определяется за счет внешней радиочастотной инфраструктуры, моделей движения людей и плана этажей. Помимо маячков, Wi-Fi и Bluetooth, система использует внутренние датчики смартфона (барометр, акселерометр, датчик магнитного поля и др.).  
Пошаговая навигация в интерфейсе смартфона в режиме реального времени.  
Карты маршрутов можно увеличивать и уменьшать, они способны охватывать все здания и этажи локации. Пошаговые инструкции (автоматически отображаются и обновляются в зависимости от положения пользователя) позволяют легко ориентироваться и перемещаться по помещениям. Конфигурация и корректировка выполняются с помощью редактора карт, что обеспечивает высокую гибкость в изменении содержимого и визуализации в любое удобное время.

Построение маршрута до нужной точки на карте.

Маршрутизация с использованием параметров начала пути и места назначения осуществляется через библиотеку маршрутов. Позиционирование возможно как внутри, так и снаружи помещений. Применение векторных карт вместо растровых сокращает время загрузки и обеспечивает автономные возможности геолокации с легким набором данных.

Фиксация времени входа в запрещенные зоны.

Посредством системы можно четко определять, в какое время сотрудник вошел в запрещенную зону. Такой подход повышает безопасность труда, предупреждает несанкционированное проникновение персонала в зоны особого доступа, обеспечивает снижение количества внештатных ситуаций [8].

**DECK WAYZ** - это система навигации внутри помещений и трекинга [7].

С помощью системы вы сможете:

* дать пользователям возможность понять где они находятся и как им пройти до нужного объекта
* инициировать различные бизнес сценарии в зависимости от их местоположения
* получать аналитику по передвижениям пользователей и использовать ее для повышения эффективности бизнеса

Варианты использования системы:

* ритейл
* музеи, выставки, конференции
* навигация в "запутанных" зданиях и комплексах
* логистика
* интеграция с системами контроля и управления доступом

Рассмотренные выше аналоги, а также остальные решения на рынке требуют:

1. платную подписку на sdk
2. покрытие помещений BLE маячками
3. плотную сеть Wi-Fi

Вышеперечисленные пункты складываются в высокую стоимость разработки и внедрение, поэтому принято решение разработать собственную систему навигации. Для этого необходимо определиться с системой позиционирования для собственного приложения.

# Выбор системы позиционирования

В процессе изучения предметной области было выделены следующие технологии.

**BLE маяки (iBeacon)**

Маяки необходимо расставлять в помещениях таким образом, чтобы расстояние между ними не превышало 10 метров, поэтому необходимо чтобы в каждой комнате был как минимум один маяк. Стоимость маяка варьируется от 1000 до 5000 рублей [5].

***Плюсы:***

Существуют сервисы с готовым приложением.

Пользователь может отслеживать свое положение в реальном времени с максимальной точность.

***Минусы:***

Для нескольких зданий университета понадобится большое количество маячков, а также лицензия на приложение, что делает эту технологию дорогостоящей.

Такие сервисы содержат большое количество разных готовых решений, поэтому в них имеется слишком много лишнего функционала.

**Wi-fi триангуляция положения**

Существуют системы использующие эту технологию, которая довольно точно может определить местоположение человека внутри помещения.

***Плюсы:***

Пользователь может отслеживать свое положение в реальном времени с хорошей точность.

***Минусы:***

Позиционирование по WiFi работает только на Android, возможности, предоставляемые iOS разработчику, ограничены в этом плане.

Высокая сложность разработки приложения

Необходимо большое количество точек wi-fi, которые должны постоянно работать.

**QR коды**

Данная технология не дает возможность отслеживать положение в реальном времени во время передвижения. Чтобы понять свое местоположение нужно найти и отсканировать QR код.

***Плюсы:***

Низкая сложность разработки

Дешевое внедрение QR кодов, по сравнению с датчиками BLE или точками Wi-fi.

***Минусы:***

Пользователь может отслеживать свое положение только от места с QR кодом, но не может видеть передвижение в реальном времени.

Таблица 2 - Подведение итогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название технологии | Сложность разработки | Стоимость внедрения | Определение местоположения |
| BLE | Можно купить готовое приложение | Высокая | В реальном времени |
| Wi fi | Высокая | Средняя | В реальном времени |
| QR коды | Низкая | Низкая | От стационарной точки |

Исходя из этого, технология с QR кодами соответствует нашим требованиям.

# Перечень задач, подлежащих решению в процессе разработки

Для разработки приложения необходимо:

* Составить техническое задание
* Проанализировать предметную область
* Определить набор средств для разработки
* Спроектировать пользовательский интерфейс
* Разработать систему регистрации и авторизации пользователя
* Реализовать систему поиска кратчайшего пути

# Обоснование выбора платформы для разработки

Данное приложение проектируется для мобильных платформ. Разработка будет осуществляться под операционную систему Android исходя из средств разработки и удобства тестирования.

В качестве среды разработки используется Android studio как официальное средство разработки Android приложений. Она находится в свободном доступе и имеет подробную документацию.

# Выбор инструментов для разработки

В качестве базы данных для мобильного приложения под Android был выбран Google Firebase. Firebase – это облачный сервис, сочетающий в себе множество функций: аутентификацию, базу данных в реальном времени, хранение файлов, уведомления и прочие. Google Firebase обладает рядом особенностей:

* **Синхронизация в реальном времени для данных JSON**

База данных Firebase Realtime представляет собой облачную базу данных NoSQL, которая позволяет хранить и синхронизировать данные между пользователями в режиме реального времени.

* **Совместно работать с устройствами**

Синхронизация в реальном времени позволяет пользователям получать доступ к своим данным с любого устройства: веб или мобильного.

* **Создание приложений без сервера**

 База данных Realtime поставляется с мобильными и веб-SDK, поэтому разработчик может создавать приложения без необходимости серверов.

* **Оптимизирован для автономного использования**

Когда пользователи переходят в автономный режим, SDK базы данных Realtime использует локальный кэш на устройстве для обслуживания и сохранения изменений. Когда устройство подключается к сети, локальные данные автоматически синхронизируются.

* **Сильная пользовательская безопасность**

База данных Realtime интегрируется с Firebase Authentication для обеспечения простой и интуитивной аутентификации для разработчиков.

При работе можно столкнуться с некоторыми недостатками этого сервиса:

* его область применения намного меньше, чем у NoSQL-решения;
* Firebase сильно ограничивает вас при выборке данных и при необходимости записать данные в несколько мест одновременно;
* далеко не со всеми структурами данных удобно работать в Firebase.

Стоит заметить, что Firebase это no-sql база данных, поэтому формат хранения полностью отличается от формата SQL. Данные представляются в виде агрегатов.

В отличие от реляционной модели, которая сохраняет логическую бизнес-сущность приложения в различные физические таблицы в целях нормализации, NoSQL хранилища оперируют с этими сущностями как с целостными объектами. Достоинство JSON базы данных - это возможность хранить объекты в том виде, в каком с ними работает приложение. Недостаток такого типа базы данных по сравнению с SQL – это отсутствие возможности писать очень сложные выборки, учитывающие связи внутри базы.

# Алгоритм поиска кротчайшего пути

Для поиска кратчайшего необходимо реализовать алгоритм поиска по графу. Алгоритм Дейкстры – это алгоритм на графах, который находит кратчайшие пути от одной из вершин графа до всех остальных. Рассмотрим работу этого алгоритма.

Каждой вершине из графа сопоставим метку — минимальное известное расстояние от этой вершины до А.

Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.  Работа алгоритма завершается, когда все вершины посещены.

Инициализация.

Метка самой вершины А=0, метки остальных вершин — бесконечности.

Это отражает то, что расстояния от А до других вершин пока неизвестны.

Все вершины графа помечаются как непосещённые.

Шаг алгоритма.

Если все вершины посещены, алгоритм завершается.

В противном случае, из ещё не посещённых вершин выбирается вершина U, имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых U является предпоследним пунктом. Вершины, в которые ведут рёбра из U, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки U и длины ребра, соединяющего U с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину U как посещённую и повторим шаг алгоритма [6].

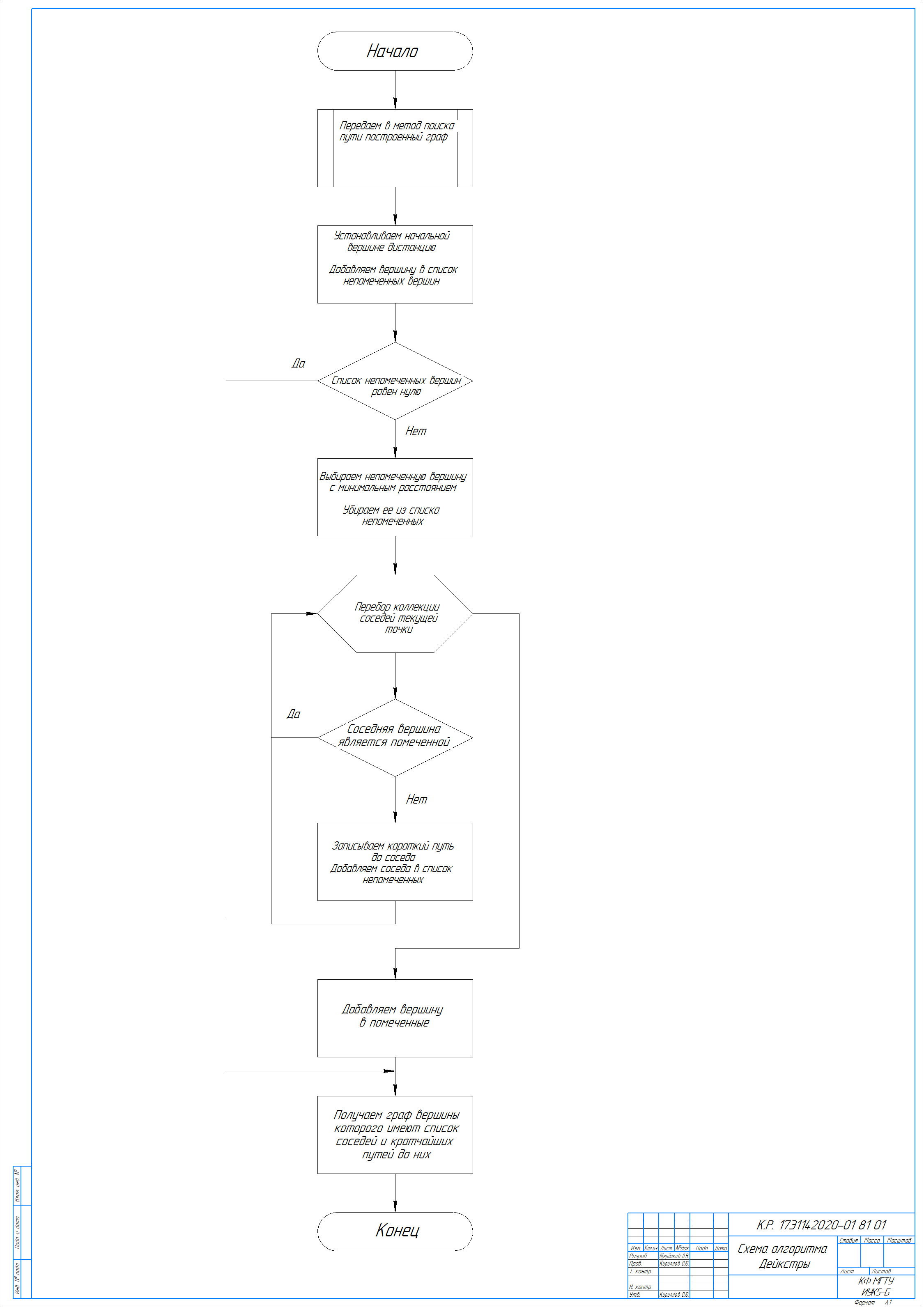


Рисунок 2 – Блок схема алгоритма Дейкстры

# Модели поведения мобильных пользователей

Для удобного расположения элементов интерфейса при разработке приложения необходимо заранее продумывать модель поведения пользователя.

Существует три основных модели поведения мобильных пользователей. Repetitive Now, Bored Now и Urgent Now.

Выбор в пользу одной из трех стартовых моделей поведения при разработке мобильных приложений позволяет сэкономить время и ресурсы при выборе оптимальных способов донести информацию о своих услугах и продуктах целевой аудитории именно тогда и в той форме, когда и как это востребовано самими пользователями.

**Модель Repetitive Now**

Пользователи, чьи действия соответствуют модели Repetitive Now, как правило используют мобильные приложения для поиска текущей, обновляющейся и повторяющейся информации. Свои интернет-планшеты и смартфоны они используют, чтобы оставаться в курсе определенных событий, просмотра новостей и для того, чтобы отслеживать постоянные изменения. Чаще всего такие пользователи отслеживают результаты спортивных состязаний, прогнозы погоды, курсы валют и акций. Их интересуют какие-то фрагменты, блоки, цитаты, небольшие «кусочки» информации, получив которые, они тут же отключают / покидают мобильное приложение или виджет.

Пользователи, чьи действия укладываются в паттерн Repetitive Now, постоянно просматривают один и тот же тип данных, но в разные даты и промежутки времени. Им в основном нужна актуальная информация и ничего более. В контексте мобильной коммерции, паттерн Repetitive Now чаще всего встречается в приложениях, которые информируют о распродажах, скидках, ценах на регулярные билеты и т.п.

Для ритейлеров модель поведения Repetitive Now дает возможность заработать путем создания мобильного приложения или секции на сайте, где размещаются уведомления об акциях, скидках, сделках дня. Кроме того, данную поведенческую модель можно применить путем представления новых товаров одной и той же линейки или новых скидок в одни и те же дни недели / часы.

**Модель Bored Now**

Поведенческая модель Bored Now — основа использования мобильных приложений, основная цель которых: отвлечение внимания, развлечение или предоставление кратковременного подключения к сети и соцсервисам с помощью мобильного устройства. Чтобы увидеть эту модель в действии, достаточно проехаться в общественном транспорте: в метро, автобусе, трамвае, поезде вы увидите множество примеров применения мобильных устройств именно в рамках данной поведенческой модели. А еще — в приемной у врача, в организациях и учреждениях, в залах ожидания аэропортов и вокзалов, в бистро и барах, на автобусных остановках. Здесь пользователи с мобильными телефонами в руках совершают множество действий, которые занимают менее 1 минуты времени: пишут текстовые сообщения, общаются в социальных сетях, проверяют электронную почту и т.п. Модель Bored Now идеально вписывается в поведение тех пользователей, которым нужно просто «убить время», пока не наступил черед заняться чем-то более важным / интересным, и они берут первую пришедшую на ум возможность современного digital-мира, чтобы отвлечься в ожидании этого «более важного».

Для мобильной коммерции модель Bored Now чаще всего срабатывает в период быстрого шоппинга и пре-шоппинга. Впрочем, удержать внимание пользователей, которые чаще всего подпадают в действиях под этот паттерн, не так-то просто. Тут важно понять, чем именно заполняют кратковременные промежутки ожидания ваша целевая аудитория: проверяет ли она электронную почту, а может, они чаще сидят в социальных сетях или составляют списки дел и покупок.

**Модель Urgent Now**

В случае нашего приложения это будет модель Urgent Now. К модели Urgent Now относятся те, кому срочно надо получить доступ или информацию, и они очень ограничены по времени. В рамках этой поведенческой модели пользователи часто ищут информацию, которая чувствительна ко времени суток, социальному или пространственно-временному контексту. К примеру, для нашего случая, это поиск ближайшей аудитории, столовой, перехода в другой корпус и так далее. Поиск оптимального маршрута и расчет времени передвижения по университету. Все это примеры использования мобильных приложений.

Люди, чье поведение и информационные потребности соответствуют модели Urgent Now, принимают быстрые решения и интересуются точной, проверенной, актуальной и максимально подробной, но при этом лаконичной информацией.

# Проектно-конструкторская часть

# Модель данных

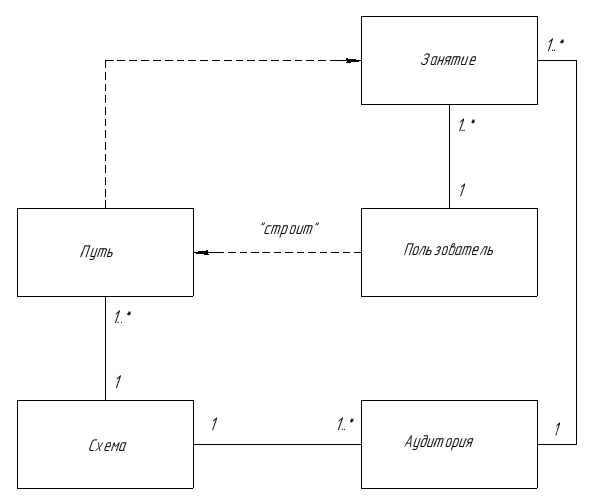


Рисунок 3 – Модель данных

При проектировании модели данных были выделены следующие сущности:

* Пользователь – пользователем приложения является студент, который строит маршруты к необходимым аудиториям.
* Аудитория – это начальная или конечная точка маршрута для студента.
* Схема – это схема университета, которая состоит из множества аудиторий.
* Путь – это путь, который студент прокладывает от начальной до конечной точки его маршрута.
* Занятие – в зависимости от занятия, студент выбирает до какой аудитории ему необходимо построить путь.

# Архитектура приложения

В мобильном приложении применяется двухзвенная архитектура. Вся бизнес-логика и логика представления находится на стороне клиента. Сервер служит для удаленного хранения данных.

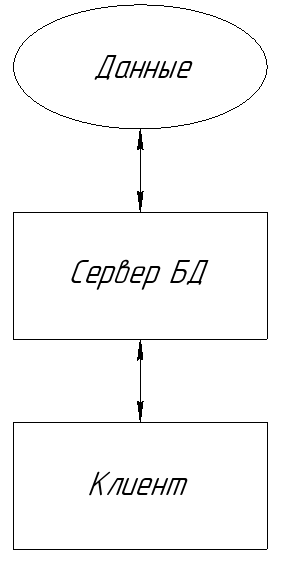


Рисунок 4 – Архитектура приложения

# Разработка структуры приложения

Разработка мобильного приложения разделена на следующие основные этапы:

1. Разработка пользовательского интерфейса
2. Подключение БД для регистрации и авторизации
3. Реализация алгоритма поиска кратчайшего пути
4. Разработка схемы университета

# Разработка пользовательского интерфейса

На основании изучения предметной области в первую очередь проектируется диаграмма использования приложения, по которому будет разработан прототип пользовательского интерфейса.

После создания прототипа проводится юзабилити-тестирование, рассчитанное на выявление проблемных мест интерфейса.

В соответствии с пользовательским интерфейсом разрабатывается структура взаимодействия его элементов.

# Сценарий поведения пользователя

Пользователем приложения является студент возраста от 18 до 24 лет. Для удобства и экономии времени он скачивает мобильное приложение, которое поможет ему ориентироваться по университету, строить оптимальные маршруты.

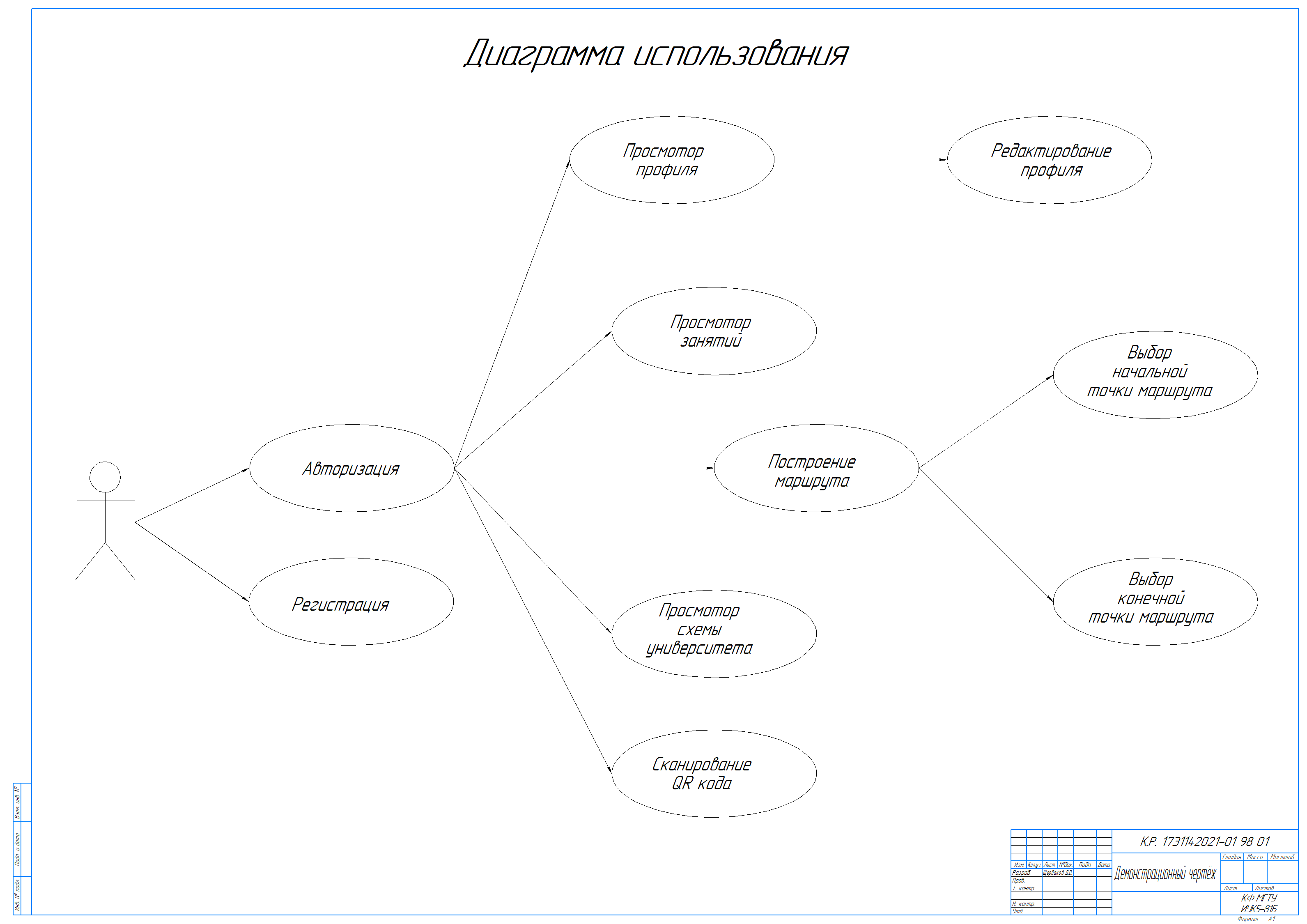


Рисунок 5 – Диаграмма использования

При запуске приложения пользователь проходит авторизацию или регистрацию.

Далее открывается главный экран профиля, пользователь может выбирать интересующие его вкладки с помощью нижнего меню навигации. Если ему надо найти маршрут, то он выбирает нужную вкладку карты в меню навигации и нажимает кнопку перехода на схему помещения.

На карте есть специальное боковое меню навигации между этажами. Для построения маршрута пользователь может нажать на аудиторию. Выдвинется информационная панель, где будет записано начало маршрута. Далее выбирается пункт назначения, а на выдвижном меню нажимается кнопка построения маршрута. Если же пользователь не знает, где находится нужная ему аудитория на схеме помещения, он вводит ее название в поиск. Информация записывается в поисковую панель.

# Основные элементы пользовательского интерфейса

На основании модели поведения пользователя разрабатывается интерфейс приложения. Для создания пользовательского интерфейса используется язык разметки XML. Создание макета интерфейса будет осуществляться через редактор макетов Android Studio, который позволяет как написать код вручную, так и перетаскивать View объекты на экран разметки.

Разработку пользовательского интерфейса можно разделить на несколько этапов, а именно создание:

* окна регистрации и авторизации
* окна профиля
* меню навигации
* интерактивной схемы

# Создание прототипов и тестирование интерфейса

После создания прототипа интерфейса, была собрана фокус группа для проведения юзабилити-тестирования. Фокус группа состояла из студентов вуза.

В результате тестирования был собран ряд отзывов на основании которых были внесены изменения в итоговый пользовательский интерфейс приложения.

# Создание меню навигации

Создаем новый ресурс меню. Это xml файл с корневым тегом <menu> состоящий из одного или нескольких элементов <item>.

Создадим элементы нашего меню для профиля, карты, занятий, камеры. Задаем каждому элементу атрибуты id, заголовок (title) и иконку (icon) из заранее созданных векторных изображений.

Далее создадим ресурс навигации с тегом <navigation> и определим следующие фрагменты: профиль, карта, занятия, камер. Каждый фрагмент имеет параметры id, name, label. Определяем layout который будет соответствовать фрагменту.

Далее создаем ActivityMain и создаем элемент библиотеки material BottomNavigationView в параметре меню которого указываем наше созданное меню bottom\_nav\_menu.

Создадим fragment который будет заполнять область выше меню навигации и в параметре графа навигации укажем mobile\_navigation.

Теперь при нажатии кнопок меню навигации на экране будут отображаться соответствующие фрагменты.

**Окно авторизации**

Создаем новый layout для окна авторизации activity\_login. Добавляем макет LinearLayout, который выравнивает все дочерние объекты в одном направлении. Выбираем ориентацию объектов Vertical, чтобы они располагались друг под другом.

Создаем следующие объекты:

ImageView – для отображения логотипа приложения.

TextView – отображает название приложения

EditView – поле ввода почты

EditView – поле ввода пароля

Buttom – для кнопка “войти”

TextView – кнопка “регистрация”

**Окно регистрации**

Аналогичным способом создаем activity\_register для окна регистрации. Добавляем следующие элементы:

EditView – для поля ввода почты, пароля, имени и группы.

Buttom – для кнопка “регистрация”

TextView – кнопка “назад”

**Схема этажей**

Для первого этажа нашей схемы создаем layout activity\_1st\_floor. Используем элемент Button для отображения точек маршрута (аудиторий, лестниц, и т.д.). В свойстве text указываем необходимые названия элементов. Создаем сетку, по которой будем располагать помещения с помощью LinearLayout и получившийся набор элементов вкладываем в слой RelativeLayout, чтобы при обращении к нему, можно было двигать и масштабировать сразу все вложенные элементы схемы.

Создаем layout второго этажа activity\_2nd\_floor и делаем всё по аналогии с первым.

Также на этом экране необходим ряд кнопок для элементов управления:

* поиск для ввода необходимой точки маршрута

А также кнопки:

* “назад” для перехода с экрана схемы в профиль
* выдвижного меню для его вызова
* выбора этажей
* масштабирования
* возврата схемы в центр экрана

Для построения линии маршрута созданы специальные View объекты. Они необходимы для визуализации графа путей, по которому мы строим линии маршрута. На экране они прозрачные, они служат только для получения координат на экране. Уже исходя из этих координат с помощью классов, предназначенных для работы с графикой, рисуется линия (рис 6).

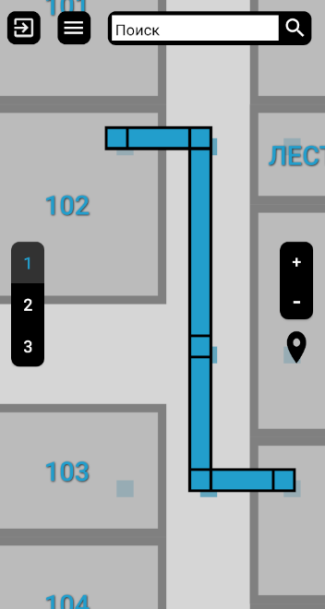


Рисунок 6 – Элементы View по которым строится линия пути,

**Выдвижное меню**

Нижний экран bottom\_sheet – это компонент, который выдвигается снизу экрана, для отображения дополнительной информации. Этот экран содержит в себе следующие элементы:

* Два элемента типа TextView – текстовые поля “От” и “До” в которое записывается начальная и конечная точки маршрута.
* TextView – в который записывается расстояние маршрута.
* Две кнопки типа Button – кнопка построения маршрута “Маршрут” и кнопка “Очистить”, которая убирает путь и очищает все поля.

Для вызова нижнего меню необходимо нажать необходимую кнопку, убрать его можно либо нажатием на туже кнопку, либо свайпом вниз. Данный элемент позволяет освободить пространство экрана для более удобного пользования схемой.

**Фрагмент fragment\_home**

Фрагмент fragment\_home является профилем пользователя. Профиль студента должен содержать в себе текстовые поля: имя, почты, группы. А также из профиля должен осуществляться переход к фрагменту настроек fragment\_settings и выход из аккаунта.

Для этого создаем два View объекта, которые будут выполнять роль кнопок. Присваиваем им id logout (выхода) и settings (настроек) для дальнейшего обращения к ним через код.

Далее добавляем три элемента TextView, которые являются текстовыми полями и будут отображать почту, группу и имя. Добавляем им id textEmail, textGroup, textName.

Для каждого элемента создаем векторные изображения, которые отражают суть текстового поля или кнопки, и размещаем их рядом.

**Фрагмент fragment\_settings**

Создаем элемент выравнивания LinearLayout, указываем вертикальную ориентацию для того, чтобы объекты на экране располагались друг под другом. Добавляем три элемента EditText, которые позволяют вводить текст и редактировать его. Присваиваем элементам id edit\_name, edit\_mail, edit\_group для обращения к ним через код.

Добавляем элемент Button для кнопки “сохранения” и кнопки “назад”. Присваиваем id settings\_save и settings\_back.

**Фрагмент fragment\_timetable**

В фрагменте timetable необходимо отображать расписание пользователя. Для этого необходимо создать разметку таблицы. Для этого используем разметку TableLayout, которая позиционирует свои дочерние элементы в строки и столбцы. Полученные строки и ячейки столбцов используются как контейнеры для хранения объектов TextView, в которые будут записываться данные о расписании.

**Фрагмент fragment\_camera**

Для добавления сканера QR кода необходимо добавить в проект дополнительную библиотеку. В файл Gradle уровня модуля добавляем новую зависимость implementation 'com.budiyev.android:code-scanner:2.1.0'.

Создаем фрагмент и добавляем новый view элемент из загруженной библиотеки CodeScannerView. Это область, в которой будет отображаться камера смартфона.

# Подключение базы данных

Firebase служит базой данных, которая изменяется в реальном времени и хранит данные в JSON. Любые изменения в базе данных тут же синхронизируются между всеми клиентами, или девайсами, которые используют одну и ту же базу данных. Вместе с хранилищем, Firebase также предоставляет пользовательскую аутентификацию, и поэтому все данные передаются через защищенное соединение SSL. Мы можем выбрать любую комбинацию email и пароля для аутентификации, будь то Facebook, Twitter, GitHub, Google, или что-то другое.

Для подключения нашего проекта к Firebase необходимо:

* Необходимо указать имя пакета приложения и указать его SHA-1 сертификат.
* Скачать и добавить в проект google-services.json файл.
* В файле Gradle уровня приложения необходимо добавить правила для включения модуля служб Google, а для файла Gradle уровня модуля объявить зависимости продуктов Firebase [1].

classpath "com.android.tools.build:gradle:4.0.2"

implementation platform('com.google.firebase:firebase-bom:26.8.0')  
implementation 'com.google.firebase:firebase-analytics'  
implementation 'com.google.firebase:firebase-database:19.5.1'  
implementation 'com.google.firebase:firebase-auth:20.0.1'

* В меню инструментов Android Studio выбрать вкладку Firebase и нажать кнопку подключения приложения.

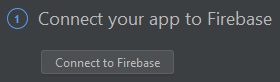


Рисунок 7 – Подключение Firebase к проекту

# Регистрация пользователя

В аккаунте Firebase выбираем вкладку Authentication и выбираем метод авторизации Email/Password с использование почты (в качестве логина) и пароля.

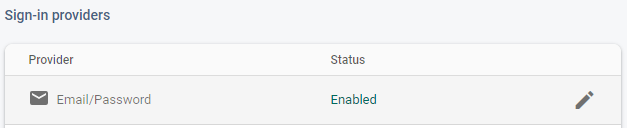


Рисунок 8 – Настройки авторизации

В нашем проекте создаем класс User. Создаем поля email, password, group, fullName.

Далее создаем два класса LoginActivity и RegisterActivity для управления экранами авторизации и регистрации.

В классе RegisterActivity создаем экземпляр класса FirebaseAuth. Поскольку нам необходима регистрация по почте и паролю, то вызываем метод этого класса createUserWithEmailAndPassword(email, password).

Для авторизации в классе LoginActivity создаем вызываем метод signInWithEmailAndPassword(email, password).

После регистрации в Realtime Database нашего Firebase проекта будет создана запись с информацией о пользователе.

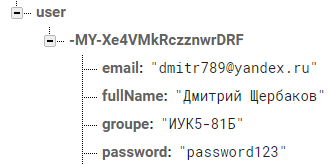


Рисунок 9 – Фрагмент кода Firebase

**Правила доступа базы данных**

Настроим правила доступа к базе данных таким образом, чтобы доступ на чтение мог получить только авторизированный пользователь:

{  
 "rules": {  
 ".read": "auth != null",  
 ".write": "false"  
 }  
}

Данное правило дает возможность чтения любому авторизированному пользователю, права на изменение не имеет никто (изменять запись в рамках статьи можно напрямую из FB консоли).

**Структура базы данных**

Users:

user\_id:121jhg12h12

email: ""

fullName: ""

group: "[group\_id]"

password: ""

user\_id:12321g12h45

...

Groups:

group\_id: 3213jhg21

friday:

пара1: ""

...

monday:

пара1: ""

...

saturday:

пара1: ""

...

thursday:

пара1: ""

...

tuesday:

пара1: ""

...

wednesday:

group\_id: 3213jhg21

...

# Реализация навигации по приложению

**Класс MainActivity**

Этот класс служит для управления экраном activity\_main. В методе onCreate создаем экземпляры классов BottomNavigationView, NavController для меню навигации. Вызываем setOnNavigationItemSelectedListener слушатель, и задаем условия:

При нажатии соответствующих кнопок открывать фрагмент navigation\_home, navigation\_timetable, navigation\_camera. При нажатии на кнопку карты запускаем активность, управляемую классом MapActivity.

**Класс HomeFragment**

Данный класс служит для управления фрагментом fragment\_home. В нем создаем три поля типа TextView textEmail, textGroup, textName, а также два View объекта logout и settings. Инициализируем используя id элементов фрагмента.

Добавляем слушатель на logout, который при нажатии осуществляет выход из аккаунта и переходит на экран авторизации activity\_login.

Слушатель на settings открывает фрагмент настройки fragment\_settings.

Создаем объект класса DatabaseReference userRef и добавляем ему слушатель addValueEventListener.

Он вызывает метод onDataChange(DataShapshot dS) в котором мы ищем нашего пользователя и получаем значения почты, группы и имени. Методом setText записываем полученные значения в наши TextView.

**Класс SettingsFragment**

Создаем три поля EditText для пароля, почты и группы. Вызываем метод onDataChange(DataShapshot dS) объекта DatabaseReference и заполняем поля уже имеющимися значениями.

Добавляем два поля типа Button для кнопок “сохранить” и “назад”. Чтобы сохранять измененные данные полей EditText, добавляем слушатель на кнопку сохранения и методом setValue() перезаписываем значения в базе данных.

**Класс TimetableFragment**

В классе, управляющем фрагментом расписания, создаем объект DatabaseReference. С помощью метода child(String id) получаем ссылку на данные расписания по id группы пользователя, методом getValue() получаем значения и заполняем текстовые поля TextView.

**Класс CameraFragment**

Класс CameraFragment служит для управления фрагментом со сканером QR кода. В нем создаем экземпляр класса CodeScanner и устанавливаем слушатель setCallback(new DecodeCallback()), который при сканировании кода вызывает метод onDecoded(final Result result). При сканировании кода получаем строку следующего вида: “id\_места номер\_этажа название”, её разделаем на три части по пробелам. Далее запускаем активность схемы университета.

Чтобы передать полученную строку в другой класс, используем объект Intent. Через метод putExtra() добавляем ключ и связанное с ним значение. В методе onCreate() класса MapActivity создаем объект Bundle и методом get() получаем переданные значения. Эти значения передаем в параметры метода buttonAction(String id, int floor, String info), который выбирает нужную точку на схеме.

# Реализация алгоритма поиска кратчайшего пути

**Классы Node и Graph**

Node – это класс, который описывает вершину графа. Он содержит такие поля, как name – название вершины, х – координата по Х, y – координата по Y. Список вершин shortestPath, которые описывают кротчайший путь, рассчитанный из исходной вершины, и словарь adjacentNodes, в который добавляются соседние вершины и расстояние до них.

Класс Graph cодержит в себе хэш-таблицу nodes для хранения вершин графа.

**Класс Dijkstra**

Класс Dijkstra реализует в себе алгоритм поиска кратчайшего пути Дейкстры. Алгоритм реализуется в методе calculateShortestPathFromSource().

Метод getLowestDistanceNode() возвращает вершину с наименьшим расстоянием от заданного количества неустановленных вершин, а метод calculateMinimumDistance() сравнивает фактическое расстояние с вновь рассчитанным, следуя недавно исследованному пути [4].

# Реализация схемы университета

**Класс MovementScaling**

Данный класс содержит в себе функции, которые реализуют масштабирование и передвижение схемы университета по экрану. Он содержит следующие методы:

* actualPosition(View v, float x, float y) перемещает объект на необходимые координаты. В его параметры передается View объект и координаты, которые ему необходимо присвоить.
* actualScale(View v, float x, float y) задает необходимый размер объекта на экране. В его параметры передается View объект и масштаб, который необходимо установить.
* getScale(View v) получает актуальный масштаб объекта
* scaleUp(View v) увеличивает масштаб объекта в два раза
* scaleDown(View v) уменьшает масштаб объекта в два раза
* resetPosition(View v) размещает объект в центре экрана (в координатах 0:0)
* moveOnTouch(View v, MotionEvent event) перемещает объект по экрана. Обработчик MotionEvent необходим для обработки касаний экрана. Если выполняется условие MotionEvent.ACTION\_DOWN (палец коснулся экрана), мы получаем координаты объекта. Если выполняется условие MotionEvent.ACTION\_MOVE (палец движется), то меняем координаты объекта.

**Класс DrawView**

Для того, чтобы рисовать линии маршрута создаем класс DrawView, наследуем его от класса View. Класс содержит поля ArrayList<Node> pointF и pointF2 для хранения координат точек, по которым будут рисоваться линии маршрута.

Данный класс содержит следующие методы:

* drawLine(Canvas canvas, ArrayList<Node> points, Paint paint\_1, Paint paint\_2), который рисует линию между двумя точками, передаваемыми из списков.
* onDraw(Canvas canvas) наследуемы от View предназначен для работы с графикой. В нем создаем кисти, стили линий (цвета, толщина, контур), а также вызываем метод driwLine(), в который рисует линию между двумя точками. В его параметры передаем список точек, кисти.

**Класс MapFunctions**

Для осуществления поиска маршрута необходимо создать взвешенный граф, вершины которого будут олицетворять реальные объекта, а вес связей этих точек будет отражать расстояние между этими объектами. Класс MapFunctions содержит методы необходимые для создания этого графа.

* Вершинами графа являются view объекты, расположенные на экране. Методы floorGetX и floorGetY в качестве параметров получают список этих view объектов. С помощью методов getX и getY получаем их координаты. Данные методы возвращают списки cooryX и coordY.
* Метод buildArray создает массив вершин типа Node, каждой вершине присваивается имя, координаты Х и У. Метод возвращает заполненный массив.
* Метод buildGraph создает граф из вершин. Метод addDestination() класса Dijkstra создает связь между вершинами и задает расстояние между ними. Далее создается объект класса Graph, который отражает наш граф. Метод addNode() класса Graph добавляет вершины к самому графу.

**Класс MapInitialization**

Класс MapInitialization хранит в себе массивы:

* idList, idList2 – эти массивы хранят в себе id точек по которым будет строиться маршрут.
* viewsArray, viewsArray2 – массивы объектов типа View
* nodes, nodes2 – массивы объектов типа Node

**Класс MapActivity**

Данный класс служит для управления экранами схемы университета, а именно активностью activity\_1st\_floor, activity\_2nd\_floor. Этот класс имеет следующие методы:

* В методе onCreate(Bundle savedInstanceState) инициализируем текстовые поля и кнопки, а также создаем экземпляры классов, функции которых нам необходимы.
* initializationFloor(Integer[] idList, View[] viewsArray) инициализируем точки по которым строим наш граф
* initializationForFloors(ArrayList<View> views, RelativeLayout relativeLayout) получаем с экрана координаты Х и У каждой точки
* buildFloor1() присваиваем координаты Х и У каждому элементу массива nodes
* buildGraph() передаем в наш граф массивы с нодами и строим граф, возвращаем его
* buildWay(String start) считаем все короткие пути от точки Старт в нашем графе и возвращаем граф с путями
* drawWay(String end, Graph graph, int floorCheck) рисуем путь от точки Старт до нужной нам точки
* drawLines() создаем пустой View элемент на весь экран, на которым рисуем линию
* changeFloor(int layoutNumber, int floor) меняет схему этажа
* buttonClear()
* buttonAction(String id, int floor, String info)

# Проектно-технологическая часть

# Руководство пользователя

Первый экран, который открывается после запуска приложения – это экран авторизации. Пользователь вводит свою почту и пароль и нажимает кнопку “Войти”, если он зарегистрирован. Если не зарегистрирован, то нажимает кнопку “Регистрация” (рис. 10).

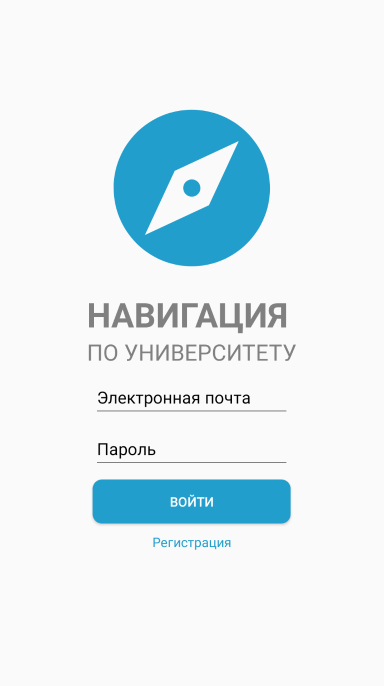


Рисунок 10 – Окно авторизации

После нажатия на копку “регистрация” открывается окно регистрации пользователя (рис. 11), в котором необходимо заполнить следующие поля:

* Электронная почта
* Пароль
* ФИО студента
* Группу (выбрать из выдвигающегося списка)

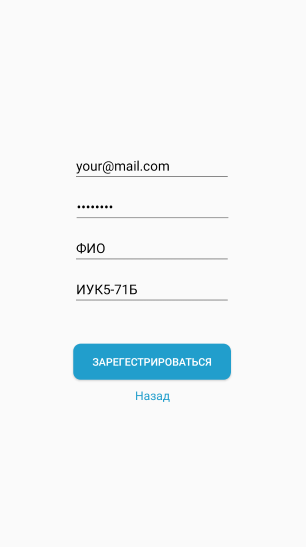


Рисунок 11 – Окно регистрации

После авторизации открывается окно профиля студента. В нем отображается данные, указанные при регистрации. Внизу располагается меню навигации с основными вкладками: профиль, карта, расписание, камера (рис. 12).

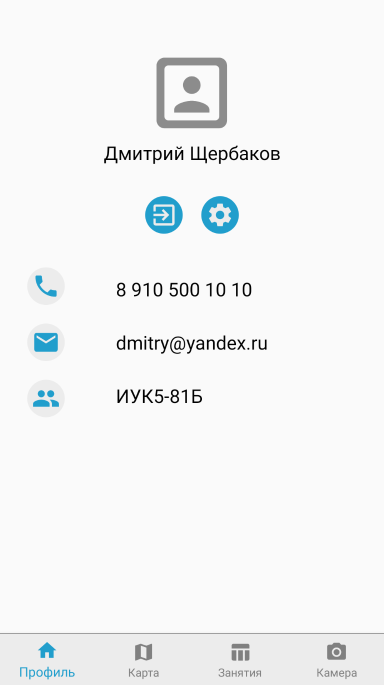


Рисунок 12 – Окно регистрации

Иконка выбранной вкладки окрашена голубым цветом, неактивные вкладки серым.

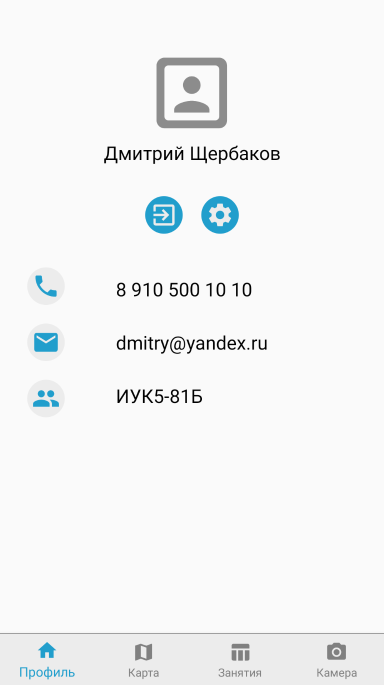


Рисунок 13 – Главная страница

Для того, чтобы выйти из аккаунта, пользователю необходимо нажать на кнопку со стрелочкой в верхней части экрана, после этого будет открыто окно авторизации.

Если пользователь хочет поменять пароль или почту, то ему необходимо нажать на кнопку с изображением шестеренки, чтобы перейти к настройкам.

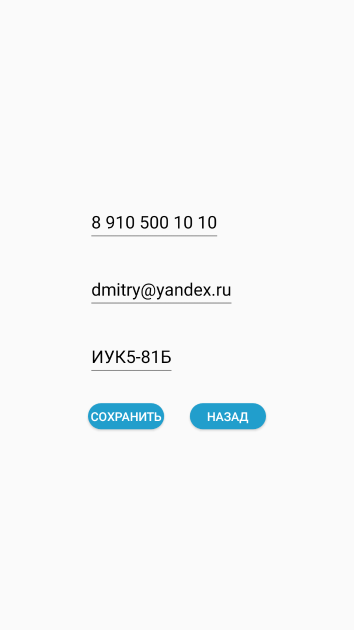


Рисунок 14 – Настройки

Для того, чтобы отсканировать QR код необходимо в меню навигации нажать кнопку камеры. В открывшемся окне при первом запуске нужно дать разрешение на использование камеры приложением. После этого откроется окно с выделенной зоной по середине. В ней нужно расположить сканируемый код. После сканирования откроется схема университета с отмеченной точкой.

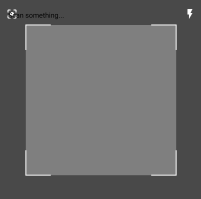


Рисунок 15 – Область сканера

При переходе к схеме университета открывается следующая вкладка.

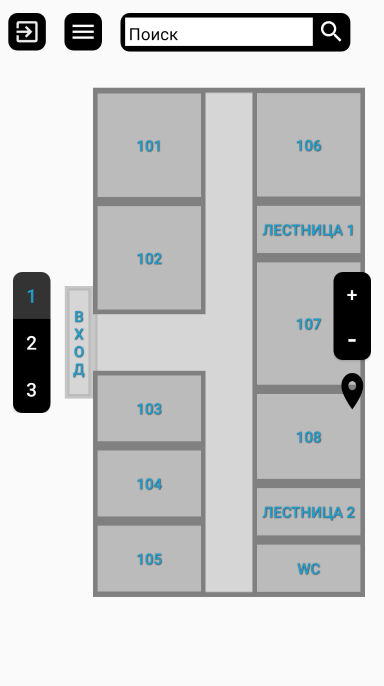
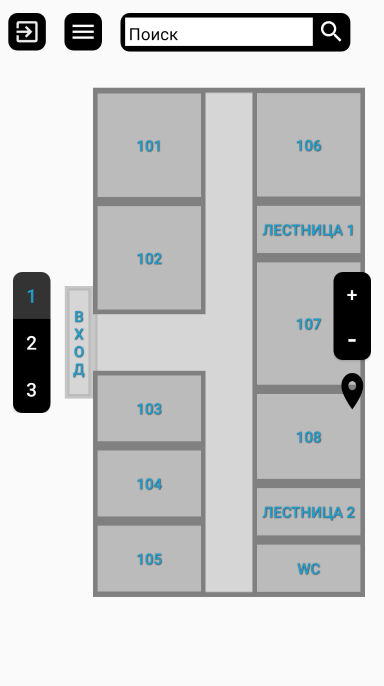
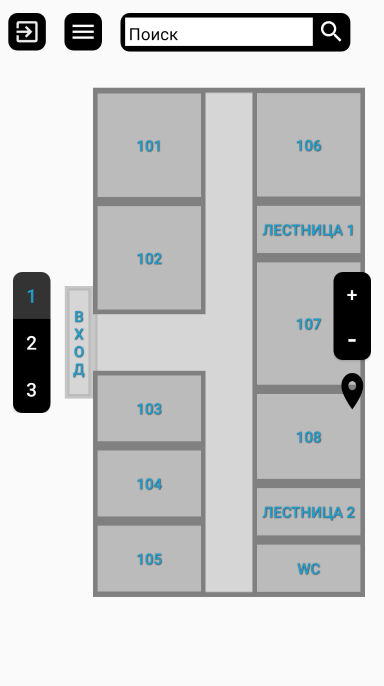
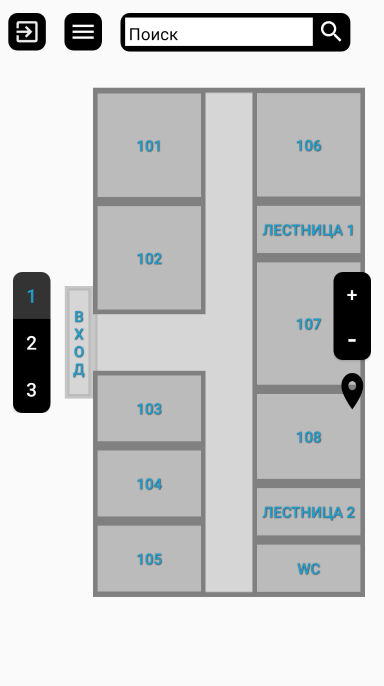
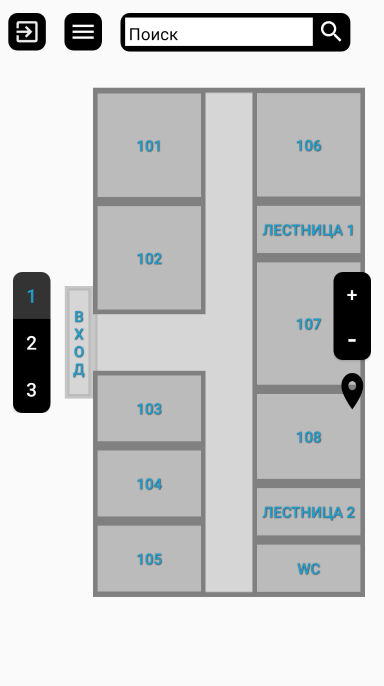


Рисунок 16 – Схема университета

На ней располагаются следующие элементы управления:

* Кнопка  - при нажатии возвращает в профиль.
* Кнопка  - открывает выдвижное меню (рис. 17).
* Панель поиска  – имеет поле для ввода, чтобы найти нужный объект на схеме необходимо ввести в строку его название и нажать на значок лупы справа.
* Слева на экране расположены кнопки “1,2,3” для перехода между этажами.
* Справа находится три кнопки: “+” и “-“ отвечают за приближение и отдаление схемы; кнопка  возвращает схему на центр экрана.

При выборе начальной точки маршрута, или через поиск, или через нажатие на соответствующую иконку, откроется информационная панель, в которой выводится следующая информация: начало, конец, расстояние маршрута (рис. 7).

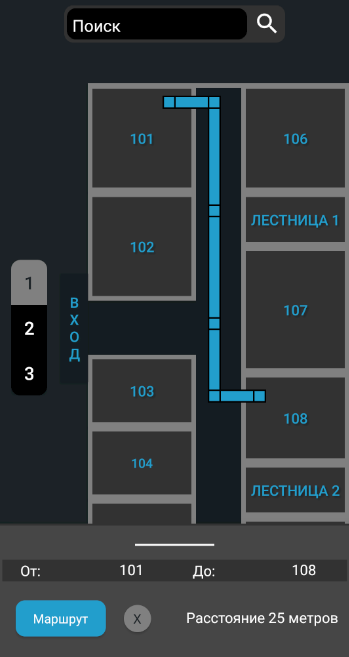


Рисунок 17 – Выдвижное меню

Также на информационной панели располагаются кнопки построения маршрута “Маршрут” и кнопка “Х” (очистить).

При нажатии на необходимый объект на нем появляется синий контур, выдвигается нижнее меню, а в поле “От”, если еще нет выбранного объекта, записывается его название (рис. 18). Если пользователь выбирает второй объект, его название записывается в поле “До”.



Рисунок 18 – Выбор начальной точки маршрута

При нажатии на кнопку “+”, схема увеличивается на экране в два раза (рис. 19).

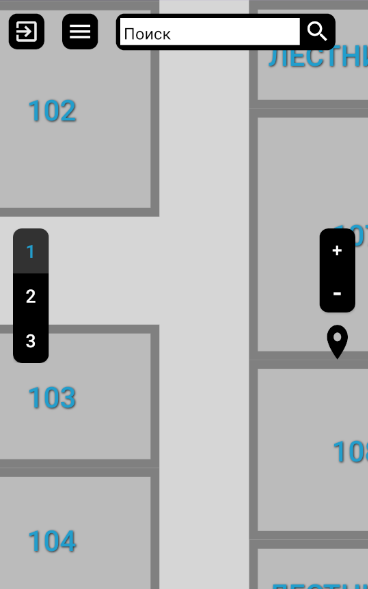


Рисунок 19 – Увеличение схемы

# Порядок внедрения приложения

**Процесс разработки и тестирования**

В процессии тестирования для использования приложения осуществляется его загрузка на смартфон APK файла через компьютер.

**Релиз приложения**

После окончания разработки и тестирования системы её внедрение заключается в размещении приложения на площадке Play Market для пользователей Android. Ссылка на скачивание будет доступна на сайте университета, который будет использовать разработанное приложение. Также скачивание можно будет осуществить при сканировании QR кода, который будет частью системы навигации.

**Публикация приложения в Play Market**

Для того, чтобы опубликовать приложение в Google Play, нужен аккаунт разработчика. Это особый тип аккаунта, который позволяет выкладывать приложение в Google Play. Создать его можно с помощью стандартного аккаунта Google. Необходимая часть процесса — внесение разовой оплаты за соглашение разработчика в размере 25$.

Далее необходимо выполнить ряд требований Google по оформлению описания приложения, скриншотов и иконок.

Размер apk-файла не должен превышать более 100 Мб.

Apk-файл должен быть подписан файлом цифровой подписи.

Цифровая подпись необходима для того, чтобы Google Play мог идентифицировать разработчика, и в дальнейшем только этот разработчик мог обновлять/изменять приложение.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было создано приложения для навигации по университету.

В результате проведения исследовательских работ была обоснована необходимость разработки специализированной системы, поскольку ни одна из существующих систем не удовлетворяет всем предъявляемым требованиям.

На этапе проведения проектно-конструкторских работ были решены основные архитектурные задачи, на основании которых было разработано мобильное приложение, решающие задачи навигации.

Разработка системы проводилась с учетом всех предъявляемых к ней требований, а ее функционал полностью соответствует техническому заданию.

Цель данной работы, состоящая в создание приложения для навигации по университету, была достигнута и все поставленные при проектировании задачи решены в полном объеме.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Google Firebase [Электронный ресурс] – <https://firebase.google.com/docs> (Дата обращения: 12.11.2020).
2. Документация Android Studio [Электронный ресурс] –<https://developer.android.com/guide?hl=ru> (Дата обращения: 08.10.2020).
3. Народный справочник Stack Overflow [Электронный ресурс] – <https://ru.stackoverflow.com/> (Дата обращения: 10.11.2020).
4. Информационный сайт Codeflow [Электронный ресурс]<https://www.codeflow.site/ru> (Дата обращения: 21.11.2020).
5. Корпоративный сайт Indoors [Электронный ресурс] – <http://www.indoo.rs> (Дата обращения: 15.11.2020).
6. Корпоративный сайт GlobalSat [Электронный ресурс] – <http://www.globalsat.ru/> (Дата обращения: 15.10.2020).
7. Корпоративный сайт DECK WAYZ [Электронный ресурс] – [<https://deck.lc/indoor>](http://www.globalsat.u/) (Дата обращения: 20.10.2020).
8. Корпоративный сайт Navigine [Электронный ресурс] – <http://nvgn.ru/platform/sdk> (Дата обращения: 15.10.2020).
9. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд. — М.: «Вильямс», 2007. — с. 459.

# ПРИЛОЖЕНИЕ А

