Санкт-Петербургский государственный университет

***ОБОЙЩИКОВА Екатерина Андреевна***

**Выпускная квалификационная работа**

***Разработка приложения на базе Android для адаптации студентов из КНР в Санкт-Петербурге***

Уровень образования: бакалавриат

Направление 09.03.03 *«Прикладная информатика»*

Основная образовательная программа СВ.5078

*«Прикладная информатика в области искусств и гуманитарных наук»*

Профиль «Прикладная информатика в области искусств и гуманитарных наук»

Научный руководитель: доцент, Кафедра информационных систем в искусстве и гуманитарных науках, к. пед. н, Гаевская Елена Георгиевна

Консультанты:

ст. преподаватель, Кафедра информационных систем в искусстве и гуманитарных науках, Мбого Ирина Анатольевна

Рецензент: генеральный директор, Общество с ограниченной ответственностью «Резолла», Столяров Денис Андреевич

Санкт-Петербург

2023

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Утверждаю

Заведующий Кафедрой

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

ЗАДАНИЕ

по подготовке выпускной квалификационной работы студента: Обойщиковой Екатерины Андреевны

1. Тема работы: Разработка приложения на базе Android для адаптации студентов из КНР в Санкт-Петербурге.
2. Срок сдачи студентом законченной работы: июнь 2023.
3. Исходные данные к работе: информация в Интернете.
4. План-график выполнения дипломной работы

|  |  |
| --- | --- |
| **Номера и содержание этапов работы** | **Плановая дата сдачи** |
| * + - 1. Формирование требований к мобильному приложению. Поиск и отбор аналогов и референсов. Создание технического задания. | Декабрь 2022 |
| * + - 1. Проектирование архитектуры приложения. | Декабрь 2022 |
| * + - 1. Реализация компонента Мессенджер. | Декабрь 2022 |
| * + - 1. Реализация мобильного приложения. | Апрель 2023 |
| * + - 1. Написание теоретической части диплома. | 15-20 мая 2023 |
| 1. Защита теоретической и практической частей выпускной квалификационной работы «Разработка приложения на базе Android для адаптации студентов из КНР в Санкт-Петербурге». | Июнь 2023 |

Консультанты по работе: старший преподаватель Мбого Ирина Анатольевна.

Руководители от Кафедры: кандидат педагогических наук, доцент Гаевская Елена Георгиевна.



**Оглавление**

[Определения, сокращения и обозначения 10](#_Toc135491769)

[Введение 11](#_Toc135491770)

[Глава 1. Анализ требований 12](#_Toc135491771)

[1.1 Функциональные требования 12](#_Toc135491772)

[1.2 Нефункциональные требования 15](#_Toc135491773)

[1.2.1 Требования к надежности 15](#_Toc135491774)

[1.2.2 Требования к производительности 15](#_Toc135491775)

[1.2.3 Требования к безопасности 15](#_Toc135491776)

[1.2.4 Требования к эргономике и технической эстетике 16](#_Toc135491777)

[1.2.5 Требования к возможности и простоте локализации 16](#_Toc135491778)

[1.2.6 Требования к модульности приложения или системы 16](#_Toc135491779)

[Глава 2. Анализ технологий для проектирования и реализации приложения 17](#_Toc135491780)

[2.1 Анализ геоинформационных систем 17](#_Toc135491781)

[2.1.2 Требования к геоинформационной системе 17](#_Toc135491782)

[2.1.2 Вывод 18](#_Toc135491783)

[2.2 Анализ систем регистрации и аутентификации пользователей 18](#_Toc135491784)

[2.2.1 Выбор способа аутентификации пользователей 18](#_Toc135491785)

[2.2.2 Требования к аутентификации пользователей 19](#_Toc135491786)

[2.2.3 Выбор системы регистрации и аутентификации пользователей 19](#_Toc135491787)

[2.3 Анализ баз данных 20](#_Toc135491788)

[2.3.1 Реляционные базы данных 20](#_Toc135491789)

[2.3.2 Нереляционные базы данных 20](#_Toc135491790)

[2.3.3 Выбор базы данных 21](#_Toc135491791)

[2.4 Выбор среды разработки 22](#_Toc135491793)

[2.4.1 Симметричное и ассиметричное шифрование 23](#_Toc135491794)

[2.4.2 Организация шифрования в компоненте Мессенджер 25](#_Toc135491795)

[2.5 Выбор языка программирования 25](#_Toc135491798)

[2.6 Выбор среды разработки 26](#_Toc135491799)

[Глава 3. Проектирование архитектуры 28](#_Toc135491800)

[3.1 Организация хранения данных 29](#_Toc135491801)

[3.2 Организация клиентской части приложения 30](#_Toc135491802)

[Глава 4. Разработка приложения 32](#_Toc135491803)

[4.1 Обзор структуры Android-приложения 32](#_Toc135491806)

[4.2 Настройка мультиязычности 35](#_Toc135491807)

[4.3 Организация хранения ключей шифрования 36](#_Toc135491808)

[4.4 Особенности организации пользовательских представлений 38](#_Toc135491809)

[4.5 Управление маршрутами 41](#_Toc135491810)

[Заключение 44](#_Toc135491811)

[Использованные источники 45](#_Toc135491812)

[Приложения 47](#_Toc135491813)

[Приложение №1. Диаграмма вариантов использования 47](#_Toc135491814)

[Приложение №2. Схема шифрования сообщений 48](#_Toc135491815)

[Приложение №3. Схема базы данных 49](#_Toc135491816)

[Приложение №4. Диаграмма классов 51](#_Toc135491817)

АННОТАЦИЯ

выпускной квалификационной работы

Обойщиковой Екатерины Андреевны

Разработка приложения на базе Android для адаптации студентов из КНР в Санкт-Петербурге

Работа направлена на исследование и анализ технологий для реализации и проектирования мобильных приложений на базе Android и разработку самого мобильного приложения. В работе представлен сбор и анализ требований к приложению и его архитектуре, проектирование архитектуры приложения, изучение различных технологий для проектирования и реализации и последующий выбор необходимых инструментов, а также описана разработка приложения.

Первая глава посвящена изучению функциональных и нефункциональных требований. В результате было получено описание того, что и как должно выполнять приложение.

Вторая глава представляет собой анализ технологий необходимых для проектирования и реализации приложения. Был проведен сравнительный анализ геоинформационных систем, способов аутентификации, а также баз данных. В соответствие с результатами анализа были выбраны технологии.

Третья глава посвящена проектированию архитектуры мобильного приложения «1698». Здесь рассматриваются такие вопросы как выбор архитектуры, организация хранения данных, а также организация клиентской части приложения.

В четвертой главе описываются общие принципы разработки приложения, а также подробно рассматриваются решения наиболее трудных задач реализации.

Прикладным результатом ВКР является мобильное приложение на базе Android, включающее в себя компонент Карта, позволяющий пользователю находить на карте места, связанные с китайской культурой, а также компонент Мессенджер, с помощью которого пользователи могут общаться как в личных, так и в групповых чатах.

В процессе работы использованы следующие программы: Android Studio, Firebase, GitHub Client.

Объем работы: 51 страница текста, 10 рисунков, 14 источников литературы и 4 приложения.

Ключевые слова: мобильная разработка, Java, Android, анализ требований, проектирование архитектуры, геоинформационная система.

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись (фамилия, имя, отчество)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись (фамилия, имя, отчество)

Abstract

Final project

Oboishchikova Ekaterina

Development of an Android application for the students from China adaptation to Saint Petersburg

The work is aimed at the research and analysis of technologies for the implementation and design of mobile applications based on Android and the development of the mobile application itself. The paper presents the collection and analysis of requirements for the application and its architecture, the design of the application architecture, the study of various technologies for design and implementation and the subsequent selection of the necessary tools. It also describes the development of the application.

The first chapter is the study of functional and non-functional requirements. As a result, a description of what and how the application should perform was obtained.

The second chapter is the analysis of the technologies necessary for the design and implementation of the application. A comparative analysis of geodata systems, authentication methods, as well as databases. Technologies were selected in accordance with the results of the analysis.

The third chapter is devoted to the design of the architecture of the mobile application "1698". Here we consider such issues as the choice of architecture, the organization of data storage, as well as the organization of the client-orientated part of the application.

The fourth chapter describes the general principles of application development, and also discusses in detail the solutions to the most difficult implementation tasks.

The applied result of this final project is an Android-based mobile application that includes a Map part that allows users to find places connected with Chinese culture on the map, as well as a Messenger part with which users can communicate in both personal and group chats.

The following programs are used in the process of work: Android Studio, Firebase, GitHub Client

Volume of work: 51 pages, 10 drawings, 14 literature sources and 4 appendices.

Keywords: mobile development, Java, Android, requirements analysis, architecture design, geographic information system.

# **Определения, сокращения и обозначения**

В данной работе используются следующие термины и сокращения:

Таблица 1

| Термин/сокращение | Определение / пояснение |
| --- | --- |
| Аутентификация | Процедура проверки подлинности. Например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных |
| Фронтенд | Разработка клиентской части приложения, которая отвечает за взаимодействие между пользователем и сервером |
| Бэкенд | Разработка серверной части приложения, которая отвечает за передачу данных между пользователями или ресурсами |
| БД | База данных |
| Activity | Отдельный экран в Android |

Определения, сокращения и обозначения

# **Введение**

В настоящее время в СПБГУ обучаются более двух тысяч студентов из КНР. Нередко они отмечают, что русская и китайская культура действительно отличаются и процесс адаптации к новой обстановке является для них сложным. В связи с этим появилась идея создать мобильное приложение “1698”, которое бы облегчало процесс адаптации студентов из КНР на первых этапах.

Создаваемое приложение позволяет найти места с родной едой, культурой, архитектурой, найти соотечественников в Санкт-Петербурге и пообщаться с людьми, которым также не хватает общения и контакта на родном языке. Приложение не случайно названо “1698”. Эта китайская числовая фраза обозначает выражение “пойдем вместе”, что еще раз подчеркивает цель создания приложения.

Целью данной работы является разработка мобильного приложения на базе Android, служащего облегчить адаптацию студентов из КНР в Санкт-Петербурге.

В работе решаются следующие задачи:

* сбор и анализ требований к системе;
* выбор технологий для проектирования и реализации;
* проектирование архитектуры приложения;
* разработка приложения.

# **Глава 1. Анализ требований**

## **Функциональные требования**

Функциональные требования определяют, каким должно быть поведение приложения в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи. Для их описания воспользуемся языком UML - стандартизированном языком моделирования при проектировании программ. С помощью него будут составлены диаграммы вариантов использования (use-case diagram) – диаграммы, описывающие, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Изображение выглядит как диаграмма

Автоматически созданное описаниеВ приложении существует два класса пользователей (см. Рисунок 1):

Рисунок 1 – Классы пользователей и их права доступа.

* Гость – незарегистрированный пользователь, который обладает правами:

1. Регистрация в мобильном приложении.
2. Вход в мобильное приложение.

* Авторизованный пользователь, с доступом к основным двум компонентам приложения – Карта и Мессенджер, а также с правом редактирования личного профиля.

**Изображение выглядит как диаграмма

Автоматически созданное описание**

Рисунок 2 – Варианты использования главной страницы приложения.

На первом шаге работы с приложением пользователь попадает на страницу авторизации. После прохождения авторизации пользователь попадает на интерактивную карту, которая предоставляет возможность использования поиска, доступ ко всем местам, отмеченными на карте и к отображению текущей геолокации (после соответствующего разрешения пользователя). Из компонента Карта пользователь может перейти в Мессенджер и Профиль.

Отдельный интерес представляет возможность взаимодействия с местами на карте, которые предварительно отобраны администраторами ресурса. После ее выбора пользователю предоставляется возможность перейти в диалог данного места, а также возможность открыть страницу построения маршрутов к этой точке.

Построение маршрута начинается с выбора начальной точки одним из трех способов: поиск по адресу, выбор собственной геолокации, указание точки на карте. Помимо этого, пользователю необходимо выбрать способ передвижения для расчета времени и расстояния: пешком (установлен по умолчанию), автомобильный маршрут.

В компоненте Мессенджер предусмотрено два типа пользователей: пользователь, который к текущему моменту уже участвовал в переписке, и пользователь, не имеющий диалогов. При входе в Мессенджер происходит проверка существования диалогов текущего пользователя.

Новый пользователь, не имеющий диалогов, и пользователь, уже вступивший в переписку, видят разные окна и к ним применяются разные сценарии обработки. Для нового типа пользователя на главной странице будет отображаться приветственная страница, для второго – все диалоги, в которых он участвует.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 3 – Варианты использования компонента Мессенджер*.*

На страницу с диалогами можно попасть двумя способами: с использованием поиска по пользователям или путем выбора существующего пользовательского диалога.

В системе существует два типа диалогов: персональный и групповой. Функциональная нагрузка персонального диалога совпадает с обычной функциональностью индивидуальных диалогов в мессенджерах. На странице персонального диалога можно написать и отправить сообщение, а также просмотреть информацию об участнике. Групповой диалог предназначен для использования несколькими пользователями для назначения встреч с привязкой к месту, связанному с точкой на карте. Каждый такой диалог доступен как из компонента Карта, так и из компонента Мессенджер. В групповом диалоге доступна информация с описанием выбранного места.

Обобщенная диаграмма вариантов использования представлена в Приложении №1 (Диаграмма вариантов использования).

## **Нефункциональные требования**

Нефункциональные требования – это чёткие критерии того, **как** система должна работать, в отличие от функциональных, которые описывают, **что** система должна делать.

### **1.2.1 Требования к надежности**

Приложение должно функционировать круглосуточно. Учитывая зависимость работы приложение от интернета, приложение может утрачивать основные функции при нестабильном или полностью отсутствующем соединении Интернет.

Все данные приложения должны храниться на сервере для сохранения данных и их возможного восстановления при потере доступу к устройству.

Приложение должно сохранять работоспособность при некорректных действиях конечных пользователей.

Приложение должно обеспечивать восстановление работоспособности и восстановление данных при появлении сбоев, аварий и отказов, возникающих на сервере и сетевом аппаратном обеспечении.

### **1.2.2 Требования к производительности**

Время реакции приложения на действия пользователя должны быть меньше 5 секунд.

### **1.2.3 Требования к безопасности**

Только авторизованный пользователь имеет доступ к контенту приложения.

Личная переписка должна хранить в базе данных в зашифрованном виде.

Во время работы пользователям приложения должны быть доступны только необходимые функции.

### **1.2.4 Требования к эргономике и технической эстетике**

Интерфейс приложения должен быть прост, нагляден, интуитивно понятен, легок в освоении.

Интерфейс приложения должен отвечать следующим требованиям:

* однозначно определять наименование пунктов меню;
* сигнализировать об ошибках системы или выполнении ошибочных действий пользователя в виде индикаций на экране;
* отображать вспомогательную индикацию при выполнении длительных процессов.

### **1.2.5 Требования к возможности и простоте локализации**

Интерфейс приложения должен быть реализован на трех языках: русский, китайский, английский. Пользователь должен выбирать язык при первом запуске приложения на устройстве.

### **1.2.6 Требования к модульности приложения или системы**

Приложение должно состоять из двух основных компонентов – Карта и Мессенджер, а также служебного компонента Профиль. Основные компоненты не должны быть зависимыми друг от друга.

Компонент Карта должен включать модуль Управление маршрутами, доступ к которому возможен только из основного компонента.

# **Глава 2. Анализ технологий для проектирования и реализации приложения**

## **2.1 Анализ геоинформационных систем**

Одним из основных компонентов в приложении будет Карта, на которой будет представлена информация о местах, близких студентам из КНР. Карта позволит пользователю найти место с помощью поиска, а также построить маршрут до него.

### **2.1.2 Требования к геоинформационной системе**

На основе требований к функционалу компонента Карта были выдвинуты следующие базовые требования к ГИС:

* близкое к максимальному покрытие на территории Санкт-Петербурга и Ленинградской области;
* возможность отображения местоположения пользователя;
* возможность добавления на карту собственных маркеров и их персонализации;
* возможность добавления собственных пользовательских действий при нажатии на маркер;
* возможность построения маршрутов и получения информации о них (время прохождения маршрута, расстояние);
* возможность геокодировать точку на карте (получить точную геопозицию по адресу и наоборот);
* возможность наложения на карту отрезков.

Кроме того, важнейшим критерием выбора ГИС является возможность интеграции данной системы в Android-приложение. Предпочтение отдается системам, которые имеют SDK — набор готовых решений для разработчика, уже преднастроенный внешней системой. Внутри SDK могут содержаться один или несколько API, фрагменты кода, библиотеки, документация и другие инструменты.

На соответствие данным критериям были протестированы следующие системы: Google Maps, Яндекс.Карты, 2ГИС, Here.

### **2.1.2 Вывод**

Несмотря на включение дополнительных сторонних сервисов, было принято решение использовать Google Maps, так как они позволяют полностью реализовать функционал компонента Карта.

## **2.2 Анализ систем регистрации и аутентификации пользователей**

Использование Мессенджера требует использования регистрации, аутентификации и авторизации пользователей. Каждый пользователь имеет свой профиль, через который можно будет найти соотечественников в Санкт-Петербурге и пообщаться с ними.

### **2.2.1 Выбор способа** **аутентификации пользователей**

Прежде всего, необходимо выбрать способ аутентификации. На данный момент популярны следующие способы:

* номер телефона + SMS-код;
* email + пароль;
* логин + пароль;
* email + код;
* вход через социальные сети.

Способы, использующие пароль, являются довольно стрессовыми для пользователя (большой шанс потери пароля), поэтому было принято решение от них отказаться.

Решение использовать аутентификацию пользователя с помощью социальных сетей является довольно привлекательным, так как позволяет получить больше персональной информации о нем. Однако, этот способ накладывает строгие ограничения и теперь регистрация доступна не всем.

Также при выборе способа аутентификации необходимо учитывать особенности устройства, на котором оно будет использовано. Так, при использовании email и кода, у пользователя не всегда будет возможность быстро и удобно зайти на почту, чтобы ввести код.

Оптимальным вариантом для данного приложения будет аутентификация по номеру телефона и SMS-кода, который на него поступит. Таким образом, мы сможем организовать систему «один пользователь – один аккаунт», что усложнит создание ботов и фейков в приложении.

### **2.2.2 Требования к аутентификации пользователей**

При выборе системы аутентификации были сформулированы следующие требования:

* аутентификация по номеру телефона и SMS-коду;
* поддержка как местных, так и иностранных номеров;
* надежность сервиса;
* наличие бесплатной или льготной версии.

Приоритет в выборе отдается тем системам, которые предоставляют SDK и являются наиболее простыми в эксплуатации.

### **2.2.3 Выбор системы регистрации и аутентификации пользователей**

Ввиду большого спроса на подобные инструменты большинство сервисов оказались платными. Из бесплатных предпочтение было отдано Firebase App, так как он удовлетворяет всем раннее описанным критериям и помимо этого поддерживает SDK для Android.

## **Анализ баз данных**

Хранение данных реализовано с использованием базы данных. На первом этапе необходимо было выбрать тип базы данных. В мире технологий баз данных существует два основных направления: SQL и NoSQL, реляционные (SQL) и нереляционные (NoSQL) базы данных. Различия между ними заключаются в том, как они спроектированы, какие типы данных поддерживают, как хранят информацию.

### **Реляционные базы данных**

Рассмотрим более подробно особенности каждого вида. В реляционных базах данных данные хранятся в формате таблиц, они строго структурированы и связаны друг с другом. В таблице есть строки и столбцы, каждая строка представляет отдельную запись, а столбец — поле с назначенным ей типом данных. Достоинствами таких баз данных являются надежность и неизменяемость данных, низкий риск потери информации, так как они соответствуют ACID – специальным требованиям к транзакционным системам.

Однако, у таких баз данных есть и ряд ограничений. Реляционные БД имеют строгие правила хранения данных. Реляционная БД может иметь сложную структуру, а скорость обработки запроса зависит от числа таблиц, к которым запрос обращается. Сложная структура сильно замедляет работу такой системы. Еще одним ограничением является то, что в них сложно организовать хранение и обработку сущностей с произвольным набором атрибутов или иерархических данных.

### **Нереляционные базы данных**

Нереляционные базы данных устроены иначе. Рассмотрим документо-ориентированные базы как наиболее распространенные. То есть вместо столбцов и строк мы описываем все данные в одном документе. Если нам нужно было бы добавить новые данные в таблицу реляционной базы данных, пришлось бы изменять ее схему данных. В случае с документами нужно только добавить в них дополнительные пары ключ-значение. То, что в реляционной БД будет разбито на несколько взаимосвязанных таблиц, в нереляционной может храниться в виде целостной сущности. Важным отличием является то, что такие базы данных не накладывают ограничения на тип хранимой информации, а также они имеют распределенную архитектуру, поэтому хорошо масштабируются горизонтально и отличаются высокой производительностью.

Но данный тип баз данных имеет и ряд недостатков. Так, мы не можем поддерживать четкую структурированность данных, а в связи с этих не можем гарантировать и их целостность.

### **Выбор базы данных**

Основными требованиями к БД в разрабатываемом стали скорость работы, а также отсутствие строгой структуры. Первый параметр, позволит улучшить скорость работы всего приложения, что является важным критерием выбора приложения пользователем. Второй параметр позволит легко масштабировать проект. Исходя из выдвинутых требований, было принято решение использовать NoSQL базу данных.

Одна из особенностей приложения – постоянное обновление данных внутри него. В компоненте Мессенджер часто появляются новые сообщения, эти данные должны сразу же обновляться у всех пользователей, которые вовлечены в диалог. Для удобства и реализации быстрого обновления информации на устройствах всех пользователей было принято использовать базу данных Firebase Realtime — это облачная NoSQL база данных компании Firebase. Данные хранятся в формате JSON и синхронизируются в режиме реального времени с каждым подключенным клиентом. Немаловажным фактором выбора стало то, что безопасность и проверка данных доступны через правила безопасности баз данных, которые администратор пишет самостоятельно. Такие, основанные на выражениях правила, проверяются при чтении или записи данных. Кроме того, у данного сервиса есть SDK для Android-приложений, что делает разработку более удобной.



## **Выбор среды разработки**

В настоящее время одним из критериев выбора приложения пользователем является его безопасность. Пользователи все чаще отдают предпочтение таким мессенджерам, как WhatsApp, Telegram. Одна из причин выбора таких приложений – использование в них современных методов защиты пользовательской информации.

Выделим критерии, по которым можно оценить безопасность мессенджера:

* отсутствие сбора данных и метаданных;
* наличие шифрования сообщений и вложений;
* возможность доступа к сообщениям со стороны администратора мессенджера.

Полностью отказаться от сбора данных пользователя в приложении не представляется возможным и является даже чрезмерной мерой: одна из целей приложения “1698” – поиск соотечественников и знакомство с ними, именно поэтому приложение предполагает хранение информации о пользователях – никнейм, описание, возраст, фотография по желанию. Однако, минимизация сбора данных присутствует: было принято решение отказаться от хранения мобильных контактов пользователя, местоположения, информации о социальных сетях в пользу обеспечения безопасности.

Для обеспечения выполнения указанных двух критериев необходимо ввести шифрование сообщений.

### **2.4.1 Симметричное и ассиметричное шифрование**

Для выбора способа шифрования обратимся к имеющейся классификации. Криптографические системы защиты данных в настоящее время разделены на две основные области исследования: симметричное и асимметричное шифрование.

Симметричное шифрование работает следующим образом: существует алгоритм шифрования, на вход которого подается текст и закрытый ключ. На выходе получается зашифрованный текст. Для расшифровки применяется тот же самый закрытый ключ, но используется алгоритм расшифрования. Закрытый ключ – это последовательность символов, построенная по правилу используемого алгоритма, которая должна быть известна только владельцу и не должен передаваться в открытом виде по сети.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 4 – Схема симметричного шифрования.

Ассиметричное шифрование предполагает существование двух связных между собой ключей: открытого и закрытого. Причем, открытый ключ используется для шифрования сообщений, а закрытый для расшифрования. Отличие открытого ключа в том, что мы можем распространять его публично и передавать другим пользователям.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 5 – Схема асимметричного шифрования.

У каждого способа есть свои достоинства и недостатки. Так, из плюсов симметричного шифрования можно выделить большую скорость работы, простоту используемых алгоритмов. Ассиметричное шифрование работает дольше, требует большей вычислительной мощности компьютера, имеет большую длину ключа, однако, решает проблему распределения ключей: теперь не требуется передавать закрытую информацию обоим пользователем.

В разрабатываемом компоненте Мессенджер важна как скорость работы, так и качество шифрования, поэтому принято решение использовать гибридное шифрование: с помощью ассиметричного алгоритма пользователи обмениваются ключами для симметричного шифрования, после чего обмен информацией происходит по симметричному алгоритму.

Такая схема позволит сохранять требуемую скорости работы и при этом позволит не раскрывать закрытые ключи и не хранить их в явном виде.

В качестве используемых алгоритмов были выбраны RSA (Rivest, Shamir, Adleman) для формирования и отправки ключей и AES (Advanced Encryption Standard) для обмена информацией.

Полная схема итогового варианта шифрования представлена в Приложении №2 (Схема шифрования сообщений).

### **2.4.2 Организация шифрования в компоненте Мессенджер**

Далее необходимо описать несколько особенностей использования шифрования в компоненте Мессенджер.

Шифрование будет использовано только для личных диалогов. Такое решение было принято исходя из того, что групповые чаты являются открытыми – зайти в них и отправить туда сообщение может любой пользователь.

Обмен ключами между пользователями будет происходить в момент отправки первого сообщения в пустой диалог. После чего ключ будет сохранен для данного диалога и будет оставаться неизменным.



## **Выбор языка программирования**

Среди языков программирования для Android наиболее распространены Java и Kotlin.

Оба языка являются бесплатными и распространяются как открытое программное обеспечение, являются языками со статической типизацией, поддерживают объектно-ориентируемой программирование.

Как у более нового языка, у Kotlin есть несколько преимуществ: код, написанный на нем, намного компактнее по сравнению с Java — на 30–40 %, Kotlin защищен от NullPointerException (именно этот тип ошибки является самой частой причиной сбоев приложений) и др.

Однако, Java все еще использует большая часть разработчиков, язык разрабатывается и поддерживаются уже много лет, и поэтому, на нем существует больше библиотек, которые можно использовать при разработке, что является весомым аргументом для выбора его в качестве основного языка для разработки компонента Мессенджер.

## **Выбор среды разработки**

На данный момент существует 3 наиболее популярные среды разработки под Android:

* Eclipse;
* IntelliJ IDEA;
* Android Studio.

Eclipse – это бесплатная среда разработки от некоммерческой организации Eclipse Foundation. На данный момент у этой IDE имеются следующие преимущества по сравнению с другими средами разработки:

* официальная русификация интерфейса и документации;
* большое число дополнений (например, для работы с сервером, базой данных и т. д.);
* возможность групповой разработки.

Однако, с появлением Android Studio она больше не поддерживается как официальная среда разработки Android.

IntelliJ IDEA, как и Eclipse даёт возможность создавать программы на нескольких языках программирования. У этой среды также имеются свои преимущества:

* автозаполнение;
* наличие рефакторинга;
* более удобный интерфейс.

Несмотря на удобство IntelliJ IDEA, было принято решение использовать в разработке Android Studio, так как она была создана на основе предыдущей среды разработками с рядом дополнений, и, кроме того, является официальной средой разработки Android.

# **Глава 3. Проектирование архитектуры**

Для данного приложения лучшим решением является клиент-серверная архитектура.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 6 – Схема клиент-серверной архитектуры.

Кроме того, было принято решение использовать BaaS-систему в данном приложении. BaaS или Backend as a Service — это платформа, которая автоматизирует внутреннюю разработку и заботится об облачной инфраструктуре. Поставщики BaaS выстраивают мост между фронтендом приложения и различными облачными бэкендами с помощью единого API и SDK. Идея BaaS состоит в том, что вместо того, чтобы самим разрабатывать и в дальнейшем поддерживать серверные сервисы, можно воспользоваться готовыми, набор которых вместе формируют необходимый универсальный кросс-платформенный бэкенд для любого проекта. Соответственно, не нужно взаимодействовать с сервером приложения, базой данных, клиент-серверной библиотекой, хостингом, необязательно писать административную панель, продумывать дизайн своего API.

В качестве BaaS-системы была выбрана Firebase, значительным преимуществом которой является наличие сервера приложений с хорошо документированным API. Он включает в себя базу данных, упомянутою ранее, хостинг, систему аутентификации пользователей, систему аналитики и другие сервисы.

Из сказанного выше следует, что разработка приложения будет сосредоточена на клиентской части. Взаимодействие с бэкендом реализуется на основе API Firebase, с использованием его контрактов: методов, параметров, возвращаемых значений.

## **3.1 Организация хранения данных**

Проанализировав функциональные требования мобильного приложения, было выделено несколько основных сущностей:

Место:

* ID места;
* название;
* категория;
* адрес;
* информация о ближайшем метро;
* координаты;
* фото;
* сайт;
* время работы.

Пользователь:

* ID пользователя;
* возраст;
* описание;
* диалект;
* родной город;
* имя и фамилия;
* никнейм;
* фото;
* ключи ассиметричного шифрования.

Диалог:

* ID диалога;
* тип диалога;
* участники диалога (id участников);
* сообщения;
* статус;
* закрытый ключ шифрования.

Сообщение:

* ID сообщения;
* отправитель;
* время отправления;
* текст сообщения.

На основе этих сущностей была построена схема базы данных, указанная в Приложении №2 (Схема базы данных).

## **3.2 Организация клиентской части приложения**

В официальных рекомендациях построения архитектуры Android-приложения сказано, что при проектировании архитектуры стоит опираться на такие общие принцип как разграничение между разными частями приложения и их ответственностями.

Так, разработчики Android рекомендуют выделять минимум два самых главных слоя:

* слой представления;
* слой данных.

Слой представления содержит в себе логику отображения данных на экране, отвечает за взаимодействие с пользователем и за другие процессы, связанные с UI. Например, обработка пользовательских событий, таких как нажатие на кнопку, будет обрабатываться в этом слое.

Слой данных содержит бизнес-логику приложения и открывает доступ к данным приложения. К этому слою относятся в том числе классы, отвечающие за получение данных с бэкенда. В нем эти данные могут обрабатываться и передаваться в слой представления, чтобы отобразить изменения для пользователя.

В соответствии с таким разделением ответственностей была составлена диаграмма классов. С ними можно ознакомиться в Приложении №3 (Диаграмма классов).

# **Глава 4. Разработка приложения**

В соответствие со всеми вышеперечисленными требованиями было разработано приложение.

Было принято решение разбить процесс разработки на несколько этапов:

1. Разработка модуля Профиль и настройка пользователя как объекта приложения.
2. Разработка компонента Карта.
3. Разработка модуля управление маршрутами.
4. Разработка модуля Мессенджер.

Для каждого компонента разработка велась в два этапа:

1. На первом этапе была разработана бизнес-логика, связанная с данными.
2. На втором этапе производилась разработка слоя представления данных, основанного на дизайн-проекте и конфигурационных XML-файлах.

Каждый следующий компонент использовал ресурсы предыдущего, поэтому в результате было получено связанное приложение.



## **Обзор структуры Android-приложения**

Структура каждого проекта Android по умолчанию представляет собой модуль с тремя ключевыми подпапками:

* manifests;
* java;
* res.

В папке manifests хранится файл AndroidManifest.xml, который описывает конфигурацию приложения и определяет каждый из компонентов данного приложения.

В папке java хранятся файлы кода на языке Java, которые могут быть структурированы по отдельным пакетам в соответствие с логикой приложения. В данной папке хранятся и файлы Activity. Activity — это отдельный экран в Android. Для каждой отдельной Activity необходим отдельный файл, который будет отвечать за ее жизненный цикл.

Папка res содержит дополнительные файлы, необходимые для правильной работы приложения – ресурсные файлы. Все, хранящиеся там, ресурсы разделены на отдельные подпапки:

* drawable - предназначена для хранения изображений, используемых в приложении;
* layout - предназначена для хранения конфигурационных XML файлов, определяющих графический интерфейс;
* mipmap – предназначена для хранения изображений для создания иконки приложения при различных разрешениях экрана;
* values - предназначена для хранения конфигурационных XML файлов, которые определяют различные дополнительные ресурсы. По умолчанию здесь есть два файла и одна папка:
* файл colors.xml хранит описание цветов, используемых в приложении;
* файл strings.xml содержит строковые ресурсы, используемые в приложении;
* папка themes хранит две темы приложения - для светлую (дневную) и темную (ночную).

В папке values можно создать дополнительные папки, содержимое которых также будет корректно обрабатываться системой. При разработке приложения «1698» возникла необходимость использовать не предустановленные шрифты, а также дополнительно настраивать вид приложения. Для этого в папке values были созданы папки font и styles соответственно. В папке font хранятся конфигурационные XML файлов, описывающие шрифты. В папке styles хранятся XML файлы с настройками отображения экранов приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 7 – Структура каталогов в приложении «1698».

## **Настройка мультиязычности**

Одной из особенностей приложения «1698» является мультиязычность. Приложение должно поддерживаться на трех языках – русском, английском, китайском.

Перед началом разработки были подготовлены все строки, которые будут использоваться в приложении – надписи на кнопках, текстовые подсказки, названия полей, уведомления и другие необходимые тексты.

Мультиязычные приложения в Android организованы следующим образом:

1. Подготавливаются специальные конфигурационные файлы со списком строк на каждом отдельном языке, причем, одинаковые строки в различных переводах хранятся под одной переменной.
2. В момент запуска приложения определяется язык операционной системы на устройстве. Язык приложения будет соответствовать этому языку.

С большой долей вероятностью можно предположить, что обучающиеся будут использовать свой телефон на китайском языке, а наше приложение возможно захотят использоваться на каком-либо другом в целях обучения и адаптация. Поэтому классический вариант мультиязычности в Android оказался неподходящим для приложения «1698».

В целях избегания прямой зависимости от языка, установленного на телефоне, было принято решение управлять локализацией вручную. В нашем случае процесс управления языком в приложении выглядит следующим образом:

1. Подготавливаются специальные конфигурационные XML файлы со списком строк на каждом отдельном языке. Для каждой переводимой строки созданы уникальные имена параметров XML-тэгов. Каждый перевод находится в своем файле. Таким образом, для каждой строки извлекается уникальное имя и подставляется её значение.
2. При первом запуске приложения у пользователя появляется модальное окно выбора языка.
3. После того, как язык был выбран, информация о языке сохраняется в SharedPreferences. SharedPreferences - это постоянное хранилище на платформе Android, используемое приложениями для хранения своих настроек и других данных приложения.
4. В момент запуска любой Activity формируется обращение к SharedPreferences, после чего с помощью объекта Locale устанавливается необходимый язык.

На данном этапе разработки приложения возможность выбрать язык представляется лишь единожды – в момент первого запуска приложения, однако, в дальнейшем планируется внести это в постоянный функционал приложения.

## **Организация хранения ключей шифрования**

Рассмотрим подробнее вопрос генерации и хранения ключей шифрования. В приложении используется гибридное шифрование симметричное и ассиметричное. С помощью ассиметричного алгоритма пользователи обмениваются ключами для симметричного шифрования, после чего обмен информацией происходит по симметричному алгоритму.

1. На первом этапе, во время регистрации нового пользователя, для генерации ключей используется алгоритм ассиметричного шифрования. Закрытый ключ сохраняется на устройстве. Открытый ключ сохраняется в поле БД конкретного пользователя.
2. На втором этапе пользователь А отправляет первое сообщение в пустой диалог. В это время происходит генерация закрытого ключа симметричного шифрования.
3. С целью обеспечения безопасной передачи симметричного ключа он шифруется с помощью открытого ключа ассимметричного шифрования пользователя Б, которому направляется сообщение.
4. Закрытый ключ симметричного шифрования сохраняется на устройстве пользователя А, который отправляет сообщение. Зашифрованный закрытый ключ симметричного шифрования сохраняется в поле БД нового диалога.
5. После получения сообщения пользователем Б, он расшифровывает зашифрованный ключ симметричного шифрования и отправляет свое следующее сообщение уже в зашифрованном виде, используя симметричное шифрование. Расшифрованный ключ также сохраняется на устройстве пользователя.
6. Далее все взаимодействие будет шифроваться симметричным ключом.

Таблица 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Ключи | Место хранения | Время создания | Время передачи другому пользователю |
| Открытый ключ ассиметричного шифрования | БД | Момент регистрации нового пользователя | Момент отправления первого сообщения в новый диалог |
| Закрытый ключ ассиметричного шифрования | SharedPreferences (на устройстве пользователя) | Момент регистрации нового пользователя | - |
| Закрытый ключ симметричного шифрования | SharedPreferences (на устройстве пользователя) | Момент отправления первого сообщения в новый диалог | - |
| Зашифрованный ключ симметричного шифрования | БД | Момент отправления первого сообщения в новый диалог | Момент отправления первого сообщения в новый диалог |

Управление ключами шифрования

За отслеживание состояния диалога – зашифрован он или нет - отвечает поле «status».

## **Особенности организации пользовательских представлений**

В процессе разработки приложения появилась необходимость использовать не только статичные элементы View, но и динамические. Они использовались для таких решения таких задач, как:

* отображение списка сообщений и диалогов;
* отображение приветственного окна приложения;
* округление фотографий.

Эти задачи представляют отдельный интерес, так как для их решения необходимо было создавать классы, обрабатывающие правильное отображение элементов.

**4.3.1 Отображение списка сообщений и диалогов**

Для решения первых двух задач создавались классы-адаптеры. Адаптер – объект, который соединяет между собой набор данных и объект, который использует или отображает эти данные. В задаче отображения списка сообщений и диалогов данными выступали сообщения и диалоги, а объектом, который использует эти данные - View-компоненты, в которые они встраивались.

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Списки отображения сообщений и диалогов.

**4.3.2 Отображение приветственного окна приложения**

На приветственной странице приложения используется слайдер, в который включены элементы приветствия: картинки, тексты. В задаче отображения приветственного окна приложения данными выступали тексты и картинки, тогда как отображающим объектом был сам слайдер.

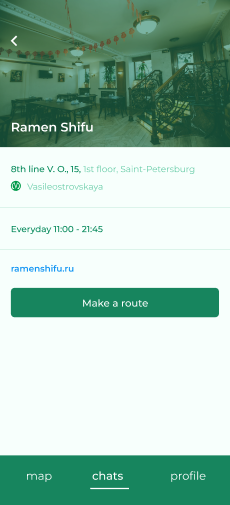


Рисунок 9 –  Приветственное окно приложения.

**4.3.3 Округление фотографий**

Для загрузки и отображения изображений использовалась библиотека Picasso. Picasso - популярная библиотека Android с открытым исходным кодом для загрузки изображений в приложение. Она была выбрана, так как позволяет легко скачивать изображение по ссылке и встраивать его в компонент представления. Библиотека представляет множество функций для правильной настройки изображения в приложении: можно изменять размер, растяжение, центрирование картинки.

Для реализации некоторых задач приложения, например, отображения фото пользователя в Профиле этого функционала было достаточно. Однако, в приложении необходимо было использовать и скругленные фотографии пользователя, например, в компоненте Мессенджер. Хранить скругленные фотографии невозможно, так как тогда их невозможно было бы отображать в исходном виде. Для того, чтобы решить эту задачу, была переопределена функция transform объекта Transformation из используемой библиотеки. В ней высчитывались новые размеры изображения и, используя объект Bitmap, строилось новое изображение.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Пример использования оригинальной и округленной фотографии.

## **Управление маршрутами**

Компонент Карта включает в себя отдельный модуль для построения и управления маршрутами. Перейти в него можно выбрав одну из предложенных точек на карте и нажав в карточке маршрута «Отправиться»/ «Make a route»/«前往».

Для того, чтобы построить маршрут необходимо:

1. Указать способ передвижения (пешеходный маршрут/автомобильный маршрут)
2. Указать способ выбора отправления точки на карте (ввести адрес/выбрать конечную точку/использовать данные о моей геолокации). Кроме того, можно нажать на карту, указанная точка автоматически станет местом отправления.
3. При необходимости можно поменять местами точку отправления и точку прибытия.

Важной особенностью реализации модуля управления маршрутами в приложении «1698» является фиксация одной из точек. Целью компонента Карта является не замена традиционно использующихся карт для навигации, а поиск мест, с близкими китайским студентам культурой, поэтому точка отправления/прибытия в модуле управления маршрутами строго зафиксирована и ею является одно из подобранных мест на карте.

Для построения маршрутов и получения информации о них использовалcя сервис Directions API, который входит в Google Maps (подробнее см. 2.1). Для того, чтобы делать запросы использовался Retrofit – это библиотека для сетевого взаимодействия. Она поддерживает REST API и позволяет быстро и удобно настраивать запросы. Для правильного функционирования запросов было добавлено два вспомогательных класса:

* POJO-класс для обработки ответа;
* интерфейс для запроса.

POJO-класс – Java-класс, которые не наследуется от каких-то других объектов и не реализует никакие служебные задачи. В приложении данный класс нужен, чтобы обрабатывать ответ на запрос и получать информацию о маршруте. Интерфейс для запроса представляет собой команды-запросы для сервера. Команды комбинируются с базовым адресом сайта и получается полный путь к странице запроса.

После получения ответа на запрос на карте в модуле управления маршрутами строиться ломаная линия, обозначающая маршрут, а с помощью отрезков, из которых она состоит, высчитывается время и дистанция пути.

# **Заключение**

В результате проведенной работы выполнены задачи, поставленные перед ВКР.

Благодаря исследованиям, выполненными в рамках выпускной квалификационной работы, были решены задачи создания кросскультурного мобильного приложения с интерфейсами на трех языках, включая китайский.

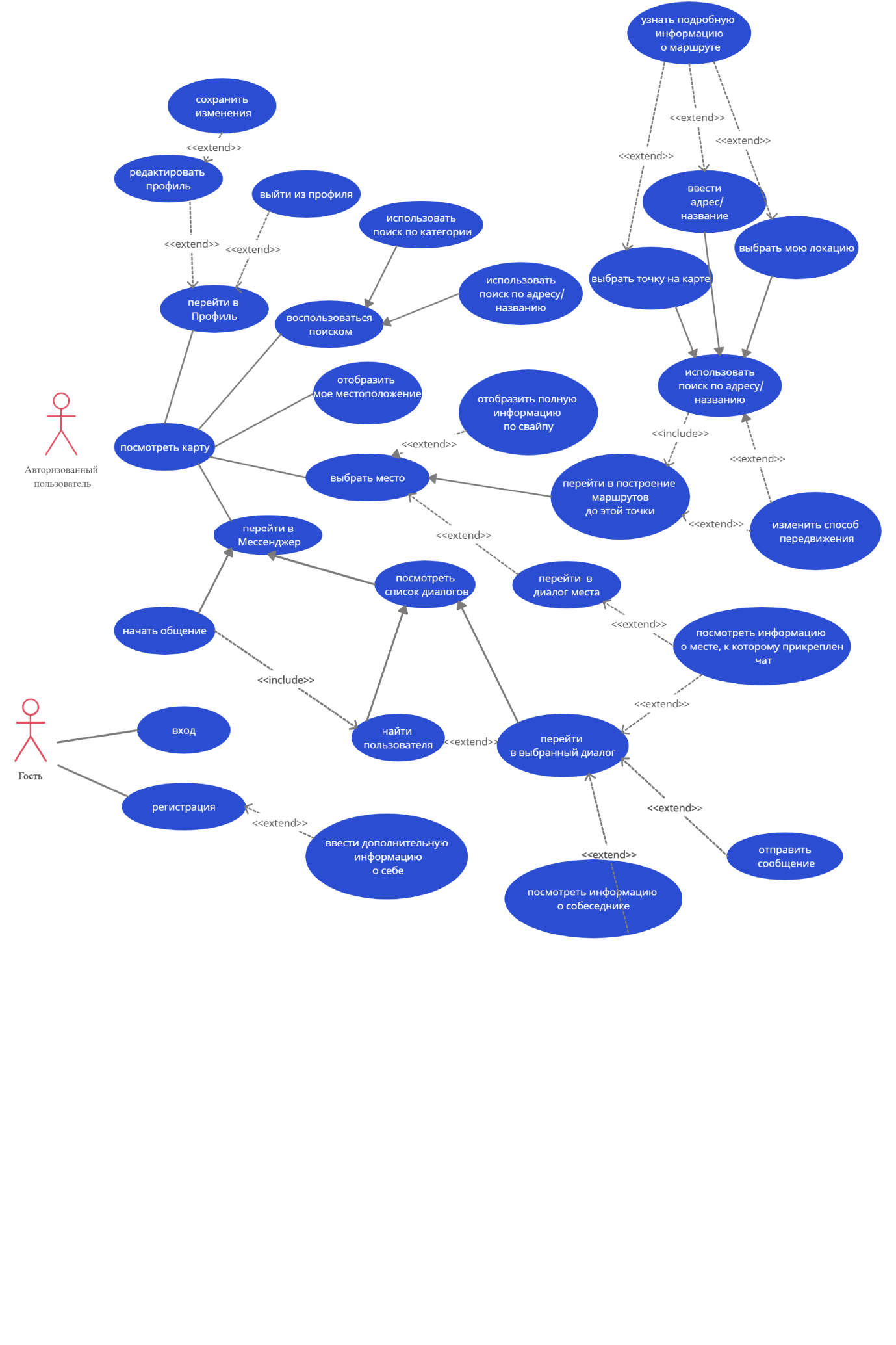
Приложение является инструментом, которое мотивирует пользователей к активному совместному изучению Санкт-Петербурга, и призвано облегчать адаптацию студентов из КНР, привлечь их внимание к культуре Китая, представленного в Санкт-Петербурге.

# **Использованные источники**

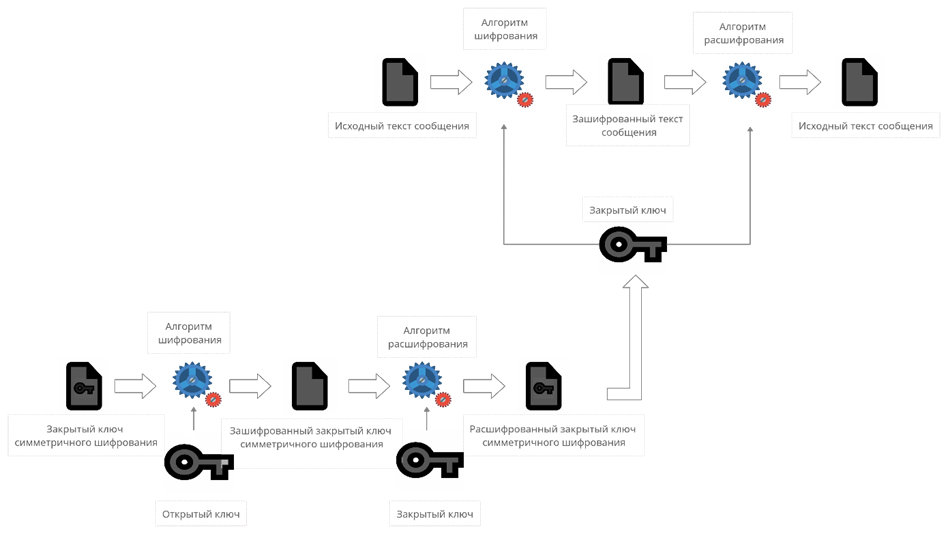
1. Использование диаграммы вариантов использования UML при проектировании программного обеспечения, - URL: https://habr.com/ru/post/566218/ (Дата доступа: 10.10.2022)
2. Вигерс Карл, Битти Джой, Разработка требований к программному обеспечению. 3-е изд., дополненное [Текст]; перевод с англ.— СПб.: БХВ-Петербург, 2014. — 736 стр.: ил. — Перевод изд.: Software Requirements Third Edition / Karl Wiegers and Joy Beatty. 2013. - 978-5-9775-3348-5 (в пер.)
3. Нефункциональные требования к программному обеспечению. Часть 1, - URL: https://habr.com/ru/articles/231961/ (Дата доступа: 10.10.2022)
4. Фаулер М., Садаладж П. Дж.л NoSQL: новая методология  разработки  нереляционных  баз  данных ; перевод с англ. Д. А. Клюшина ; – М.: Вильямс, 2013. – 192 с.: ил. – Перевод изд.: NoSQL Distilled A Brief Guide to the Emerging World of Polyglot Persistence / Pramod J. Sadalage, Martin Fowler. The United States, 2013. – 978-5-907144-91-0 (в пер.)
5. Ховард М., Лебланк Д., Вьега Дж 24 смертных греха компьютерной безопасности*:* как написать безопасный код – перевод с англ. - Front Cover, Питер, 2010 - Computers – 394с. – Перевод изд.: 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them / Michael Howard, David LeBlanc, John Viega. 2009.– 978-5-49807-747-5 (в пер.)
6. Ховард М., Лебланк Д. Защищенный код – перевод с англ. — 2е изд., испр. — М.: Издательство «Русская Редакция», 2005. — 704с. – Перевод изд.: Writing Secure Code Second Edition / Michael Howard, David LeBlanc.2003. – 978-5-7502-0238-6 (в пер.)
7. Криптография в Java. Класс KeyStore, - URL: https://habr.com/ru/post/445786/ (Дата доступа: 01.12.2022.)
8. Симметричное и ассиметричное шифрование: просто о сложном, - URL: https://otus.ru/nest/post/726/ (Дата доступа: 01.12.2022)
9. Сравниваем Java и Kotlin, - URL: https://habr.com/ru/company/otus/blog/580738/ (Дата доступа: 20.10.2022)
10. Документация для Firebase разработчиков, - URL: https://firebase.google.com/docs (Дата доступа: 10.10.2022)
11. BaaS (Backend-as-a-Service), - URL: https://ru.bmstu.wiki/BaaS\_(Backend-as-a-Service) (Дата доступа: 10.10.2022)
12. Первый проект в Android Studio, - URL: https://metanit.com/java/android/1.2.php (Дата доступа: 10.01.2023)
13. Create dynamic lists with RecyclerView, - URL: https://developer.android.com/develop/ui/views/layout/recyclerview (Дата доступа: 10.01.2023)
14. Picasso, - URL: https://developer.alexanderklimov.ru/android/library/picasso.php (Дата доступа: 10.01.2023)

# **Приложения**

## **Приложение №1. Диаграмма вариантов использования**



## **Приложение №2. Схема шифрования сообщений**



## **Приложение №3. Схема базы данных**

{

"places": {

idPlace: {

"address": string,

"firstCoord": float,

"label": string,

"metro": string,

"name": string,

"photoUrl": string,

"secondCoord": float,

"site": string,

"time": string

}

},

"users": {

idUser: {

"id": string,

"age": string,

"e": string,

"n": string,

"desc": string,

"dialect": string,

"hometown": string,

"name": string,

"nickname": string,

"photoUrl": string,

"dialogs": {

idDialog: {

"type": string,

"members": {

counter: id\_user,

},

"messages": {

idMessage: {

"from": string,

"idMessage": string,

"message": string,

"time": string

}

}

}

}

}

}

## **Приложение №4. Диаграмма классов**

Изображение выглядит как текст, снимок экрана, Параллельный, Прямоугольник

Автоматически созданное описание