МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

**«Приложение для просмотра погоды «Weather»»**

Выполнил студент Ильин Никита Сергеевич

(Ф.И.О.)

Руководитель проекта асс. Уласевич Н.И. (учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. Кафедрой к.т.н., ст. преп. Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: асс. Уласевич Н.И. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс. Уласевич Н.И. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Содержание

[Введение 5](#_Toc167082178)

[1 Анализ аналогичных решений 6](#_Toc167082179)

[1.1 Яндекс. Погода 6](#_Toc167082180)

[1.2 WeatherBug 7](#_Toc167082181)

[1.3 Today Weather 7](#_Toc167082182)

[2 Проектирование приложения 9](#_Toc167082183)

[2.1 Проектирование базы данных 9](#_Toc167082184)

[2.2 Описание информационных объектов и ограничений целостности 10](#_Toc167082185)

[2.2.1. Таблица users 10](#_Toc167082186)

[2.2.2. Таблица userCities 10](#_Toc167082187)

[2.3 Проектирование API 10](#_Toc167082188)

[2.4 Проектирование мобильного приложения 11](#_Toc167082189)

[2.5 Выводы по главе «Проектирование программного продукта» 12](#_Toc167082190)

[3 Программная реализация приложений 13](#_Toc167082191)

[3.1 Технические средства для разработки 13](#_Toc167082192)

[3.2 Разработка глобальной базы данных 13](#_Toc167082193)

[3.3 Разработка API 13](#_Toc167082194)

[3.4 Разработка мобильного приложения 15](#_Toc167082195)

[3.5 Вывод по главе «Программная реализация приложения» 16](#_Toc167082196)

[4 Тестирование приложения 18](#_Toc167082197)

[5 Руководство пользователя 22](#_Toc167082198)

[5.1 Руководство гостя 22](#_Toc167082199)

[5.2 Руководство администратора 23](#_Toc167082200)

[5.3 Руководство пользователя 25](#_Toc167082201)

[Заключение 30](#_Toc167082202)

[Список используемых источников 31](#_Toc167082203)

[ПРИЛОЖЕНИЕ А. Диаграмма вариантов использований 32](#_Toc167082204)

[ПРИЛОЖЕНИЕ Б. Листинг удаления города у пользователя 33](#_Toc167082205)

# Введение

В последнее время наблюдается рост популярности мобильных приложений, позволяющих пользователю получить необходимые услуги прямо с телефона, имея доступ к интернету. Это стало возможным благодаря быстрому развитию науки и технологий, а также стремлению людей экономить время и получать важную информацию без необходимости покидать дом. Разработка мобильного приложения для мониторинга погоды дает ряд преимуществ: удобство доступа к актуальным погодным условиям, возможность планировать свои действия в зависимости от погоды, уменьшение рисков, связанных с непредвиденными погодными условиями. Приложение для мониторинга погоды позволяет пользователям всегда быть в курсе последних изменений погоды, повышая их удовлетворенность и лояльность к приложению.

Задачи курсового проекта:

* Изучение существующих мобильных приложений для мониторинга погоды, анализ их достоинств и недостатков для формирования требований к новому приложению;
* Проектирование приложения, позволяющее визуализировать предварительную концепцию и спланировать разработку программного продукта, а также распределить необходимые ресурсы и время на разработку;
* Реализация программного продукта, который будет представлять собой мобильное приложение для мониторинга погоды;
* Разработка руководства пользователя, чтобы помочь пользователям ознакомиться с основными функциями и принципами работы приложения. Это важный элемент разработки, поскольку некоторые пользователи предпочитают заранее изучить функционал перед его использованием.

По завершении изучения и формулирования задач курсового проекта, окончательная цель проекта заключается в предоставлении пользователям удобного и функционального инструмента для мониторинга погодных условий с помощью мобильного приложения.

# Анализ аналогичных решений

В современном мире приложения погоды являются достаточно популярными веб–сервисами. Их основная задача – это просмотр погоды в нужном городе. При разработке приложение для своего приложения я обратил внимания на самые успешные примеры популярнейших приложений погод. Аналоги рассмотрены ниже.

## Яндекс. Погода

Одним из значимых аналогов в сфере мобильных приложений для мониторинга погоды является «Яндекс. Погода». Это приложение предоставляет пользователю детальный прогноз погоды, включая данные о температуре, влажности, скорости ветра и осадках. Особенности приложения включают автоматическое определение местоположения, настройки для кастомизации отображаемой информации, интерактивные карты погодных условий, а также уведомления о приближающихся экстремальных погодных явлениях.

Интерфейс приложения показан на рисунке 1.1.

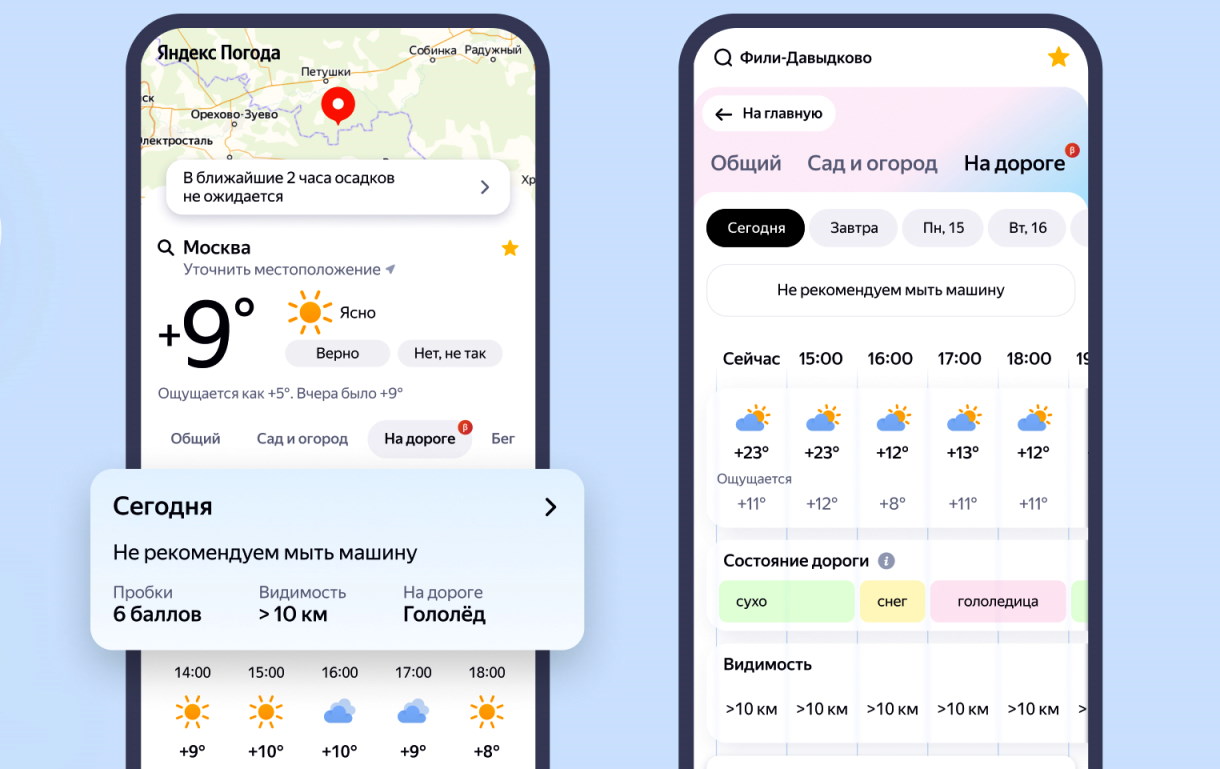


Рисунок 1.1 – Интерфейс приложения «Яндекс. Погода»

Технологически приложение выделяется своей способностью к интеграции многочисленных источников данных и обновлением прогнозов в реальном времени. Несмотря на широкий функционал и высокую точность прогнозов, пользователи могут столкнуться с недостатками, такими как наличие рекламы в бесплатной версии и ограничение доступности только для мобильных устройств, что исключает использование приложения на настольных компьютерах.

## WeatherBug

Мобильное приложение «WeatherBug» предлагает широкий спектр метеорологических данных, включая подробные прогнозы, информацию о погодных условиях в реальном времени, и карты радаров осадков. Особенности этого приложения охватывают наличие подробных графиков температуры и ветра, а также уникальную функцию предупреждений о молниях, что делает его полезным инструментом для людей, проводящих много времени на открытом воздухе. Также WeatherBug включает интеграцию с платформами социальных сетей, позволяя пользователям делиться погодной информацией с друзьями.

Интерфейс приложения показан на рисунке 1.2.

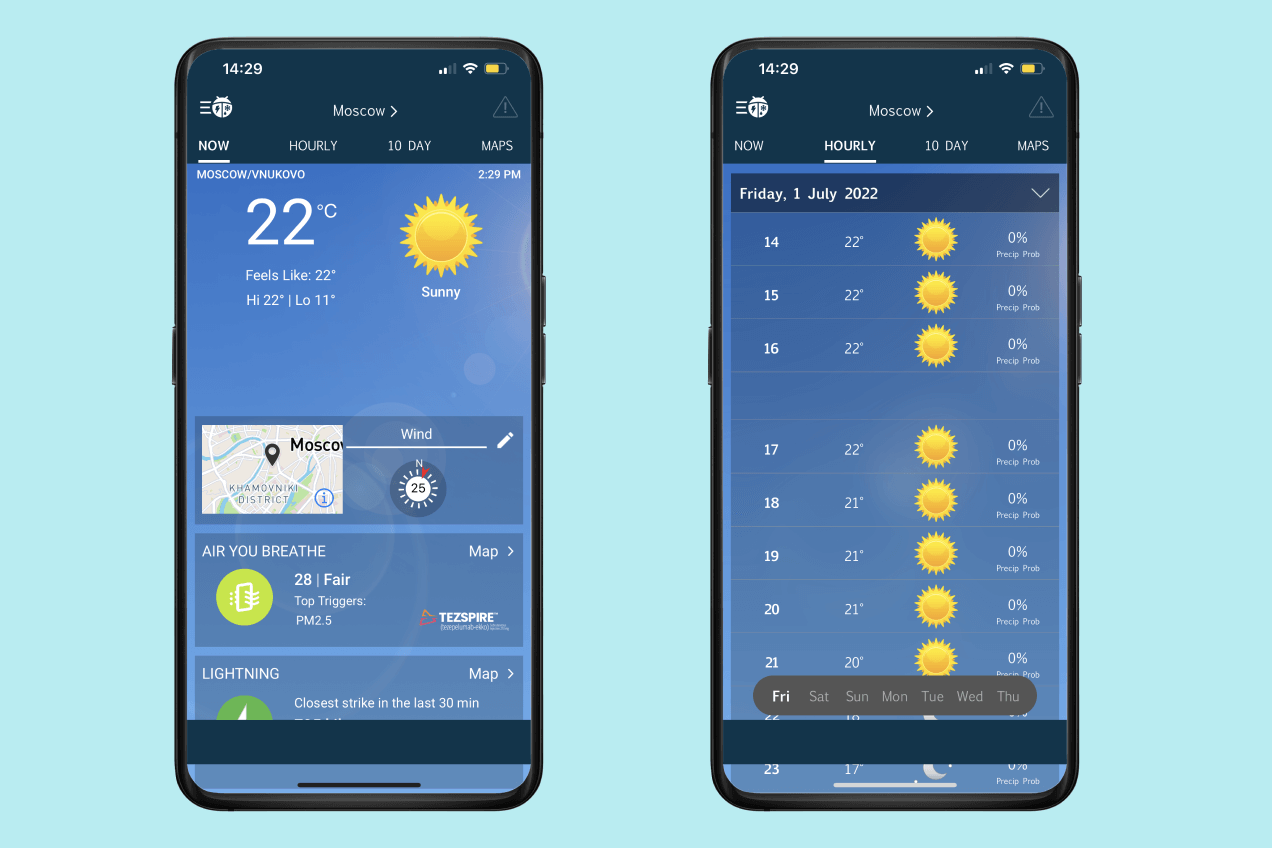


Рисунок 1.2 – Интерфейс приложения «WeatherBug»

Однако среди недостатков приложения можно выделить его относительно высокое потребление батареи и системных ресурсов, что может быть критичным для устройств с ограниченными возможностями. Кроме того, некоторые пользователи отмечают избыточную сложность интерфейса, что может усложнить использование приложения для новых пользователей.

## Today Weather

Мобильное приложение «Weather Radar & Forecast» предоставляет пользователям подробные и актуальные данные о погоде, включая анимированные радары, прогнозы и предупреждения о неблагоприятных погодных условиях. Оно отличается возможностью просматривать погодные условия на интерактивной карте, что делает его особенно полезным для планирования путешествий или мероприятий на открытом воздухе.

Интерфейс приложения показан на рисунке 1.3.

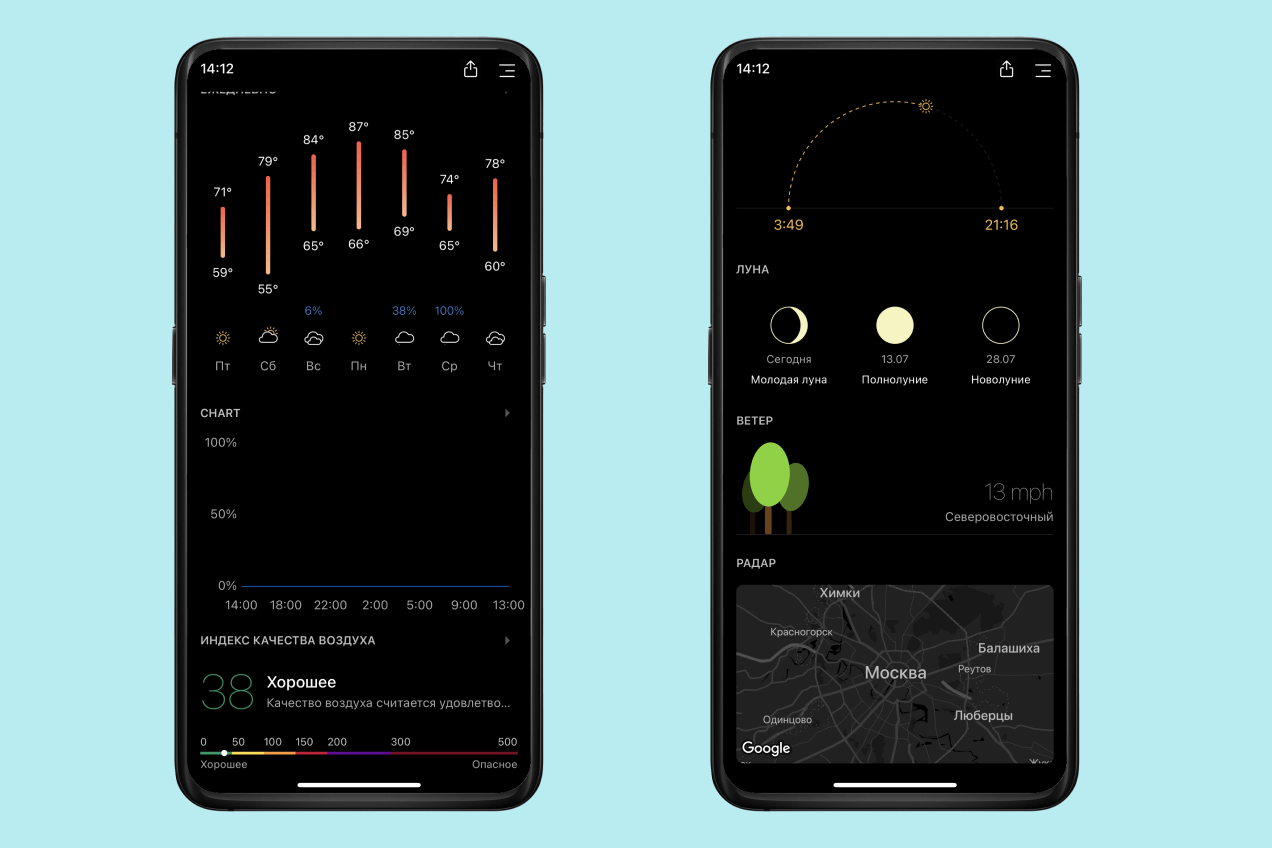


Рисунок 1.3 – Интерфейс приложения «Weather Radar & Forecast»

Приложение также предлагает виджеты для быстрого доступа к погодной информации прямо с главного экрана смартфона. Однако среди недостатков стоит отметить наличие в приложении рекламы, что может отвлекать пользователей и снижать общее удобство использования. Также пользователи иногда сообщают о неточностях в прогнозах, особенно в менее популярных или удаленных локациях.

# Проектирование приложения

Проектирование программного средства — это процесс создания детального плана или концепции программного продукта до его реализации, включающий определение структуры и организации программы, разработку основных компонентов и модулей, их взаимосвязей и функций. На этапе проектирования выбираются архитектурные решения, которые могут быть монолитными, микросервисными, распределенными и т.д., а также подходящие паттерны проектирования для эффективной реализации задач. Проектирование интерфейсов, включающее создание пользовательских интерфейсов (UI) и программных интерфейсов (API), играет ключевую роль в обеспечении взаимодействия между компонентами системы и пользователем. Важно уделить внимание удобству и интуитивной понятности пользовательских интерфейсов, а также четкому и понятному взаимодействию между модулями через API. Определение функциональности системы включает разработку подробного описания всех функций и возможностей, которые должна предоставлять система, определение бизнес-логики, основных сценариев использования и требований к производительности.

## Проектирование базы данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Модель данных играет ключевую роль в проектировании базы данных, так как она определяет структуру хранения данных и способы их обработки.

Для данного курсового проекта была спроектирована база данных. Логическая модель базы данных представлена на рисунке 2.1.

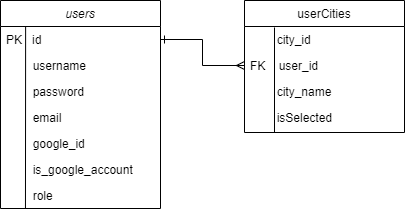


Рисунок 2.1 – Модель БД

Созданная база данных содержит в себе 2 таблицы: Users и userCities.

## **Описание информационных объектов и ограничений целостности**

Далее будет описана структура каждой таблицы. В таблице users хранятся данные для идентификации пользователей, зарегистрированных в приложении, userCities– данные о городах, добавленных пользователем.

### **Таблица users**

Таблица users состоит из следующих полей:

* id. Хранит уникальный идентификатор пользователя;
* email. Хранит email пользователя;
* username. Хранит уникальное имя пользователя;
* password. Хранит захэшированный пароль;
* role. Хранит роль пользователя.;
* google\_id. Хранит уникальный идентификатор пользователя в системе google;
* is\_google\_account. Хранит булево значение, означающее был ли создан аккаунт до авторизации через google.

### **Таблица userCities**

Таблица userCities состоит из следующих полей:

* city\_id. Хранит уникальный идентификатор города;
* user\_id. Хранит уникальный идентификатор пользователя, который выбрал город;
* city\_name. Хранит название города;
* isSelected. Хранит булево значение, отвечающее за проверку выбранного пользователем основного города.

Таблица userCilies содержит внешний ключ на таблицу users в поле id.

## Проектирование API

Для эффективной синхронизации работы локальной и глобальной БД на Android-клиенте было выбрано спроектировать серверное приложение. Оно должно обладать WEB API, который обеспечивает доступ к каждой сущности базы данных. Чтобы обмен данных между клиентами и сервером был успешным, необходимо передавать информацию в формате JSON.

Архитектурный стиль REST является оптимальным для разработки серверного приложения и позволяет обеспечить взаимодействие компонентов распределенного приложения в сети. Сервер разработан таким образом, чтобы иметь сервисы для взаимодействия с каждой сущностью базы данных.

При обращении к серверу с заданным маршрутом и описанным HTTP методом, сервер определяет, с какой сущностью базы данных необходимо взаимодействовать и какие действия должны быть выполнены.

Мы жестко разделяем уровни ответственности нашего приложения на 3 слоя, чтобы добиться расширяемости, лучшей тестируемости и гибкости приложения:

* Уровень доступа к данным;
* Бизнес логика приложения;
* Уровень представления. В нашем случае это URL, по которому будут обращаться Android и Web-клиенты, чтобы взаимодействовать с данными.

В целом, серверное приложение является неотъемлемой частью системы, гарантирующей эффективное и безукоризненное взаимодействие между различными компонентами в распределенной сети.

## Проектирование мобильного приложения

Клиентская часть приложения не менее важна, чем серверная. Это то, с чем будет контактировать пользователь, то, от чего в первую очередь зависят хорошие или плохие впечатления при использовании приложения. Здесь стоит понимать, что исходя из специфики проекта, в первую очередь разрабатывается мобильное приложение и самые строгие требования со стороны пользователя будут предъявляться именно к нему.

Точно так же, как и серверная часть приложения, как и практически любой другой проект, который имеет риск сильно разрастись в будущем, лучше всего разбить логику клиентской части приложения на части таким образом, чтобы на этой основе можно было каждую часть реализовывать последовательно и независимо от других частей.

Навигация – очень важный для мобильного приложения элемент, при её проектировании необходимо поддерживать логическую связь между экранами.

При первом запуске приложения пользователь попадает на Activity авторизации, с которой он может попасть в Activity регистрации. После успешной авторизации/регистрации пользователь попадет в Activity текущей погоды, где может посмотреть текущую погоду в городе по часам.

В Activity настроек пользователь может просмотреть свои персональные данные и поменять их, изменить пароль.

Activity поиска позволяет найти и выбрать главный город, для которого будет отображаться погода.

Общая схема взаимодействия между страницами Android-клиента представлена на рисунке 2.2.

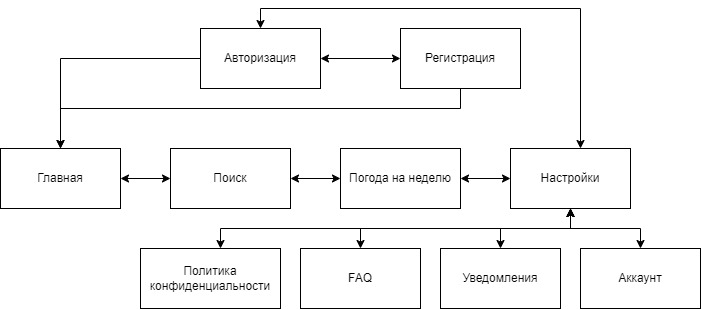


Рисунок 2.2 – Схема навигации для пользователя

Диаграмма вариантов использований представлена в приложении А.

## Выводы по главе «Проектирование программного продукта»

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения.

Была также спроектирована база данных, где были определены необходимые таблицы, охарактеризованы поля каждой из них. Также были установлены связи между таблицами, заключающиеся в создании первичных и внешних ключей.

При проектировании серверного приложения был выбран архитектурный стиль взаимодействия между клиентом и сервером – клиент-серверное взаимодействие. Это означает, что сервер предоставляет определенные услуги и ресурсы, которые клиент может запросить и использовать.

# **Программная реализация приложений**

## Технические средства для разработки

Для разработки глобальной базы данных были использованы следующие средства:

* SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционных БД;
* SQLite– система управления реляционными БД;

Средства для разработки Web API:

* Visual Studio Code – среда для разработки;
* Node.js – это среда выполнения JavaScript, основанная на движке V8 от Google.
* Express – это минималистичный и гибкий фреймворк для создания веб-приложений с использованием Node.js.

Средства для разработки Android-клиента:

* Android Studio – среда для разработки Android приложений, с хорошим статическим анализатором кода;
* Flutter – это UI фреймворк, разработанный компанией Google, который позволяет создавать кроссплатформенные мобильные приложения.
* Dart – это язык программирования, разработанный компанией Google, который используется для создания мобильных, веб- и настольных приложений.

Почти все вышеперечисленные средства для разработки являются современными и популярными, поэтому выбор и пал на них.

## Разработка глобальной базы данных

Для хранения данных будет использоватьсяSQLite.

Основные обращения к базе данных будут осуществляется через хранимые процедуры, которые позволяют инкапсулировать сложную логику внутри процедуры и предоставить удобный интерфейс для взаимодействия с БД

## Разработка API

Для данного курсового проекта был разработан сервер, который служит важной основой для функционирования системы. Серверная часть обеспечивает управление данными, взаимодействие между клиентскими приложениями и внешними сервисами, а также выполнение бизнес-логики. В этом проекте сервер был разработан в среде разработки Visual Studio Code с использованием технологии Node.js и фреймворка Express.

Node.js был выбран для разработки сервера по нескольким причинам. Во-первых, Node.js позволяет создавать высокопроизводительные серверные приложения благодаря своей асинхронной архитектуре и однопоточному циклу событий. Это особенно важно для обработки большого количества запросов одновременно. Во-вторых, Node.js имеет богатую экосистему модулей и библиотек, доступных через npm (Node Package Manager), что ускоряет разработку и упрощает интеграцию различных функциональностей.

Фреймворк Express был использован для создания серверного приложения, так как он предоставляет удобные инструменты для обработки HTTP-запросов, маршрутизации и управления промежуточным ПО (middleware). Express также обладает гибкостью и расширяемостью, что позволяет легко добавлять новые функции и масштабировать приложение.

Серверная часть проекта состоит из нескольких компонентов:

* database.js – Этот файл содержит все функции для работы с базой данных, включая вызовы хранимых процедур. Взаимодействие с базой данных осуществляется с помощью библиотеки SQLite3, которая обеспечивает надежное и эффективное хранение данных.
* serverApi.js – Этот файл содержит обработку запросов для определенных маршрутов. Здесь определены маршруты для получения, добавления, обновления и удаления данных. Каждый маршрут обрабатывается соответствующей функцией, которая взаимодействует с базой данных и возвращает результаты клиенту.
* node\_modules – Эта папка содержит все установленные зависимости Node.js. Здесь находятся все пакеты, которые используются на сервере, включая Express, SQLite3, Axios (для взаимодействия с внешними API), bcrypt (для хеширования паролей) и другие.
* package-lock.json – Этот файл содержит информацию об установленных версиях пакетов и их зависимостях, необходимую для корректной работы проекта. Это позволяет обеспечить воспроизводимость окружения разработки и избежать конфликтов версий.
* package.json – Этот файл содержит информацию о проекте, включая его имя, автора, зависимости и скрипты для запуска и сборки. Здесь также указаны основные команды для установки зависимостей и запуска сервера.

Для реализации базы данных в данном проекте использовалась SQLite. SQLite была выбрана благодаря её лёгкости, простоте и отсутствию необходимости в отдельном сервере базы данных. SQLite хранит всю базу данных в одном файле, что упрощает процесс разработки и тестирования. SQLite является встроенной базой данных, которая хранит все данные в одном файле, что делает её идеальной для небольших и средних проектов. Она не требует установки серверного ПО, что упрощает настройку и использование. SQLite поддерживает большинство команд SQL, что облегчает миграцию и интеграцию с другими системами. Благодаря своей архитектуре, SQLite обеспечивает высокую производительность для небольших и средних проектов.

Основная задача сервера заключается в обеспечении безопасного и эффективного управления данными. Например, сервер обрабатывает запросы на получение информации о погоде для определенного города, выбранного пользователем. Для этого сервер взаимодействует с внешним API OpenWeatherMap, чтобы получить актуальные данные о погоде, и возвращает эти данные клиентскому приложению.

Важной частью функциональности сервера является управление пользователями. Сервер позволяет пользователям регистрироваться, входить в систему, а также управлять своим аккаунтом. Для этого используются маршруты, которые обрабатывают запросы на регистрацию и авторизацию пользователей, а также на изменение данных профиля и пароля. Для обеспечения безопасности сервер хеширует пароли перед их сохранением в базу данных, используя библиотеку bcrypt.

Одной из задач сервера является предоставление административного интерфейса для управления пользователями. Администратор может просматривать список пользователей, искать пользователей по имени или email, а также удалять пользователей из системы. Это реализовано с помощью маршрутов для получения списка пользователей и удаления пользователей по их идентификатору.

Пример реализации одного из маршрутов для ц представлен на листинге 3.3:

app.post("/login", (req, res) => {

const { username, password } = req.body;

const sql = "SELECT \* FROM users WHERE username = ?";

db.get(sql, [username], (err, row) => {

if (err) {

res.status(500).json({ error: err.message });

return;

}

if (row && bcrypt.compareSync(password, row.password)) {

res.json({ message: "success", data: row });

} else {

res.status(404).json({ message: "User not found or password incorrect" });

}

});

});

Листинг 3.3 – Обработка входа пользователя

Если имя пользователя и пароль совпадают, функция возвращает JSON-ответ с сообщением "success" и данными пользователя. Если имя пользователя не найдено или пароль неверен, функция возвращает ответ с кодом состояния 404 (Not Found) и сообщением "User not found or password incorrect".

## Разработка мобильного приложения

Мобильное приложение построено с использованием нескольких папок, каждая из которых выполняет определенную функцию и обеспечивает структурированное и логичное разделение кода.

* Model включает классы, которые описывают данные и бизнес-логику приложения. Здесь находятся классы моделей данных, отображающие структуру данных приложения, и логика обработки этих данных.
* Services содержит компоненты, которые обеспечивают работу с сервером и глобальной базой данных. Реализованы возможности отправки HTTP-запросов и обработки полученных ответов. Например, асинхронная функция selectCity выполняет запрос к серверу для установки выбранного пользователем города и возвращает объект, содержащий информацию о выполнении функции.
* Utils представляет классы-помощники, используемые во всем приложении и содержащие логику, такую как сокрытие клавиатуры или преобразование даты в нужный формат. Эти утилиты обеспечивают удобство и консистентность кода, облегчая выполнение часто используемых операций.
* View отвечает за взаимодействие с пользовательским интерфейсом и состоит из двух подпапок: Components и Pages. Components включает в себя отдельные элементы интерфейса, такие как кнопки, текстовые поля и другие виджеты, используемые на различных страницах. Pages содержит классы, которые определяют страницы приложения и их функциональность, например, страницы авторизации, регистрации и просмотра продуктов.

Реализация асихронной функции, которая выполняет запрос к серверу для получения информации о продукте, представлена в листинге 3.5.

Future<void> selectCity(int userId, int cityId) async {

    final response = await http.post(

      Uri.parse('${Config.url}/setSelectedCity'),

      headers: <String, String>{'Content-Type': 'application/json'},

      body: json.encode({'userId': userId, 'cityId': cityId}),

    );

    if (response.statusCode == 200) {

      print('Selection updated successfully');

      var newCities = await getUserCities(userId);

      setState(() {

        cities = newCities;

      });

    } else {

      throw Exception('Failed to update city selection');

    }

  }

Листинг 3.5 – Установка основного города пользователя

Пример реализации функции удаления города у пользователя представлен в листинге, который находится в приложении Б.

Функция отправляет HTTP DELETE-запрос к серверу, передавая идентификаторы пользователя и города.

Если сервер успешно выполняет удаление (ответ с кодом состояния 200), функция обновляет список городов пользователя, уменьшая счетчик городов и показывая уведомление об успешном удалении города.

В случае ошибки функция показывает уведомление с сообщением об ошибке.

## Вывод по главе «Программная реализация приложения»

В данной главе подробно рассмотрен процесс разработки всех компонентов приложения. Вначале было проведено исследование и отбор необходимых технологий для реализации курсового проекта. После выбора технологий приступили к разработке структуры базы данных, определив ключевые таблицы и связи между ними для обеспечения целостности и эффективности работы с данными. Были разработаны запросы, необходимые для взаимодействия с базой данных, включая запросы на получение, добавление, обновление и удаление данных.

Далее было разработано серверное приложение, которое играет ключевую роль в обработке данных и взаимодействии между клиентской частью и базой данных. В этой части работы была представлена графическая структура серверного приложения, описаны все используемые классы и методы, а также их функциональность. Обсуждение включало описание логики обработки запросов, маршрутизации, аутентификации пользователей и других важных аспектов работы сервера.

После завершения серверной части проекта, было приступлено к разработке мобильного приложения. В этом разделе также была представлена графическая структура приложения, рассмотрены все классы и методы, их роль и взаимодействие. Было уделено внимание интерфейсу пользователя, взаимодействию с сервером, обработке данных и обеспечению удобства использования приложения. Подробно рассматривались компоненты приложения, такие как модели данных, сервисы, утилиты и представления, что позволило получить полное представление о работе мобильного приложения.

Затем перешли к разработке веб-приложения. Здесь также была представлена его графическая структура и основные файлы. Обсуждение включало описание архитектуры веб-приложения, взаимодействие с сервером и базой данных, а также реализацию пользовательского интерфейса. Были рассмотрены классы и методы, используемые для обеспечения функциональности веб-приложения, включая обработку пользовательских запросов, отображение данных и другие важные аспекты.

Таким образом, в данной главе детально описан весь процесс разработки приложения, начиная с выбора технологий и заканчивая реализацией всех компонентов – от базы данных до мобильного и веб-приложения. Это позволило получить полное представление о создании комплексного программного обеспечения, включающего все необходимые этапы и технологии для успешной реализации проекта.

# Тестирование приложения

Тестирование мобильных приложений важно и необходимо для обеспечения их качества и производительности. Оно помогает уменьшить риск сбоев приложения, гарантирует отсутствие ошибок и улучшает восприятие пользователей, что в свою очередь способствует увеличению количества загрузок и успешному запуску приложения на рынке.

Для начала протестируем страницу авторизации. Если пользователь не зарегистрирован, ввел неверные данные или не ввел какое-либо поле, то он получит предупреждение, показанное на рисунке 4.1.

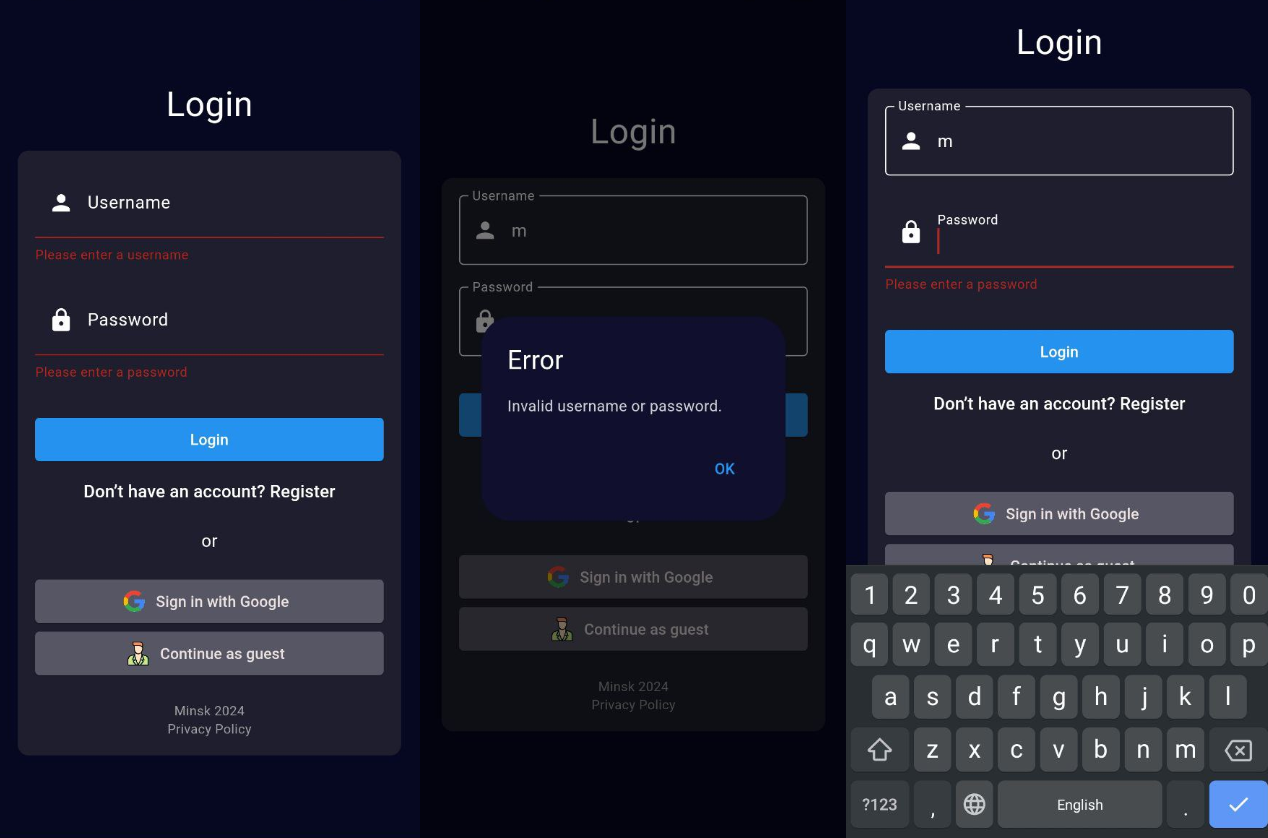


Рисунок 4.1 – Тестирование авторизации

Далее протестируем страницу регистрации. Пользователь может зарегистрироваться с уникальным email и именем, также ему необходимо повторить введённый пароль. Роль пользователя автоматически считается User. Кроме того, есть проверка на корректный email и пароль. Тестирование продемонстрировано на рисунке 4.2.

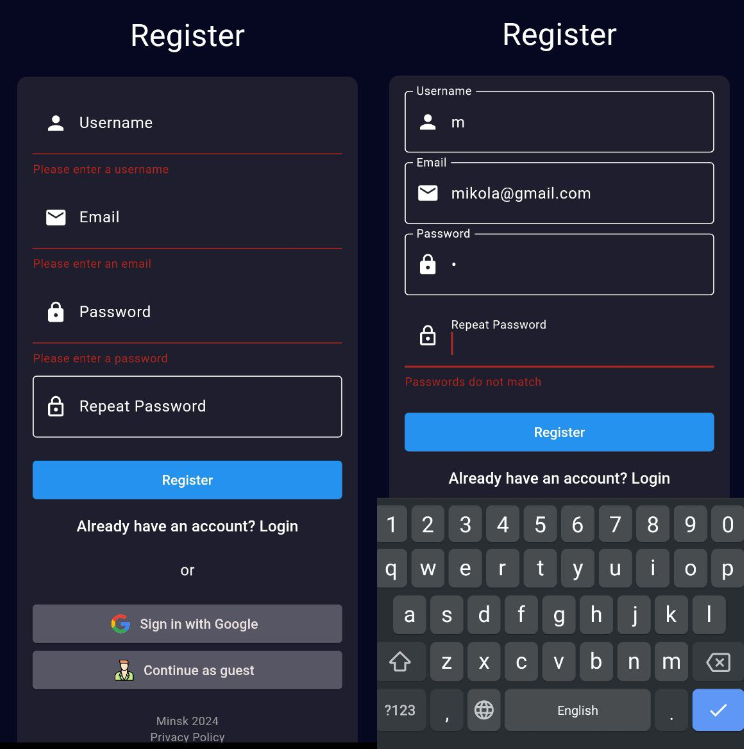


Рисунок 4.2 – Тестирование регистрации

После авторизации или регистрации пользователь попадает на главную страницу с информацией о погоде в выбранном городе. Перейдем на страницу с поиском городов. Протестируем добавление нового города. При добавлении вызывается функция, которая проверяет не был ли данный город уже добавлен у пользователя. Если выбранный город уже был добавлен, то пользователю показывается модальное окно с информацией. Тестирование продемонстрировано на рисунке 4.3.

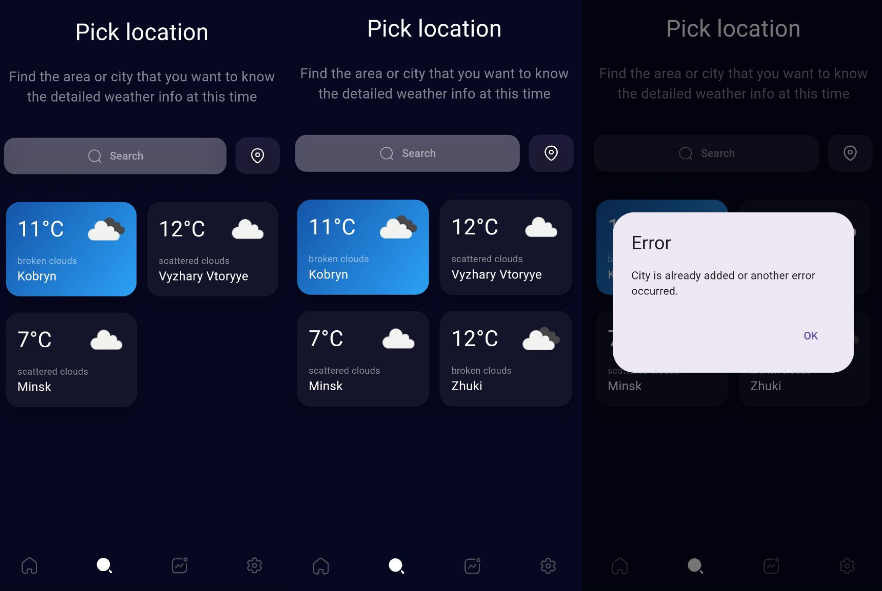


Рисунок 4.3 – Тестирование добавления города

Кроме рассмотренных возможностей приложения, обработаны возможные ошибки при редактировании профиля. Например, нельзя поменять информацию на точно такую же.

При добавлении городов осуществляется проверка на наличие такого города у пользователя. Пользователь получит предупреждение в случае повторного добавления города.

Когда пользователь меняет пароль, происходит валидация на соответствие введённых паролей, соответствие старого пароля и минимальное количество символов. Продемонстрировано на рисунке 4.4.

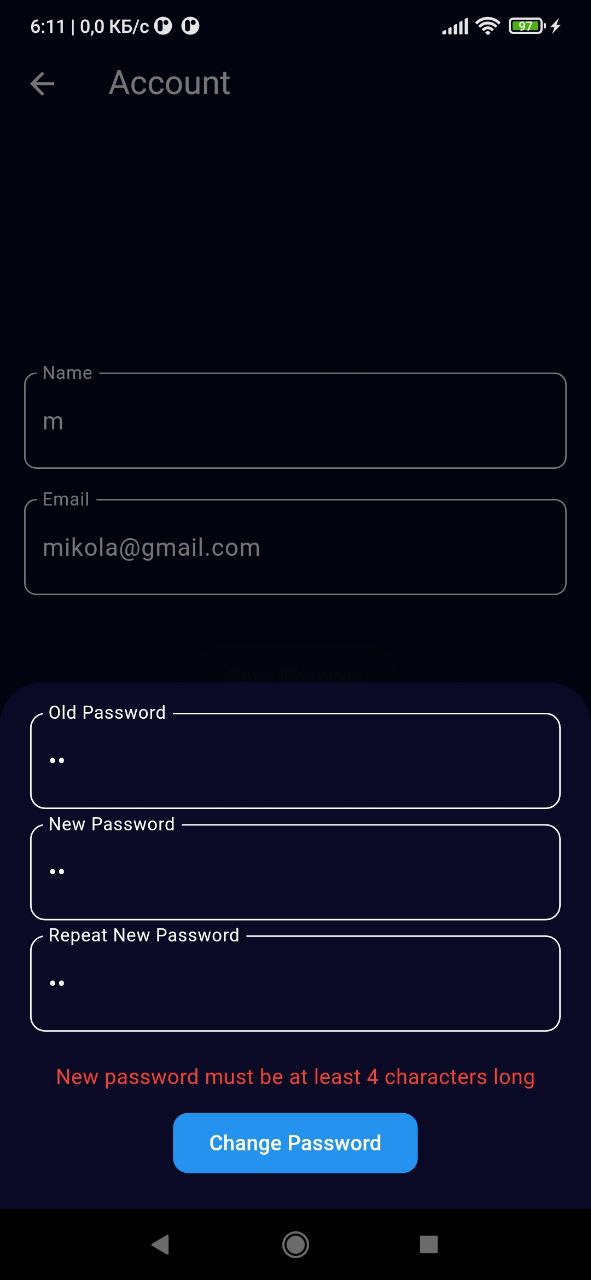


Рисунок 4.4 – Обработка ошибки при изменении пароля

Тестирование приложения является неотъемлемой частью процесса разработки, позволяющей обеспечить высокое качество продукта и удовлетворенность пользователей. В рамках данного проекта, благодаря тщательной валидации и обработке ошибок, было создано надежное и удобное в использовании мобильное приложение для фотохостинга, которое успешно выполняет свои функции и отвечает потребностям целевой аудитории.

# Руководство пользователя

В данном разделе будет описано руководство по использованию для гостя, пользователя и администратора.

## Руководство гостя

Для того чтобы пользователь зашел как гость, при авторизации или регистрации нужно нажать на кнопку «Продолжить как гость». После этого гость попадает на страницу, показанную на рисунке 5.1.



Рисунок 5.1 – Страница выбора города для гостя

При клике на кнопку, откроется модальное окно, содержащее все города, где гость может по поиску найти нужный ему город и добавить его. При желании добавить большее количество городов, необходимо нажать на кнопку авторизации. Детальная страница поста показана на рисунке 5.2.



Рисунок 5.2 – Детальная страница выбора города для гостя

Выбирать какой-либо еще город гость не может.

Гость может просмотреть основную информацию о выбранном городе или может перейти обратно на страницу авторизации/регистрации.

## Руководство администратора

Чтобы продолжить работу в приложении как администратор, пользователь должен иметь специальную роль и войти в приложение под соответствующей учетной записью. При входе в приложение как администратора будет открыта страница, показанная на рисунке 5.3. В отличие от гостя, у администратора появляется возможность искать и удалять любого пользователя.

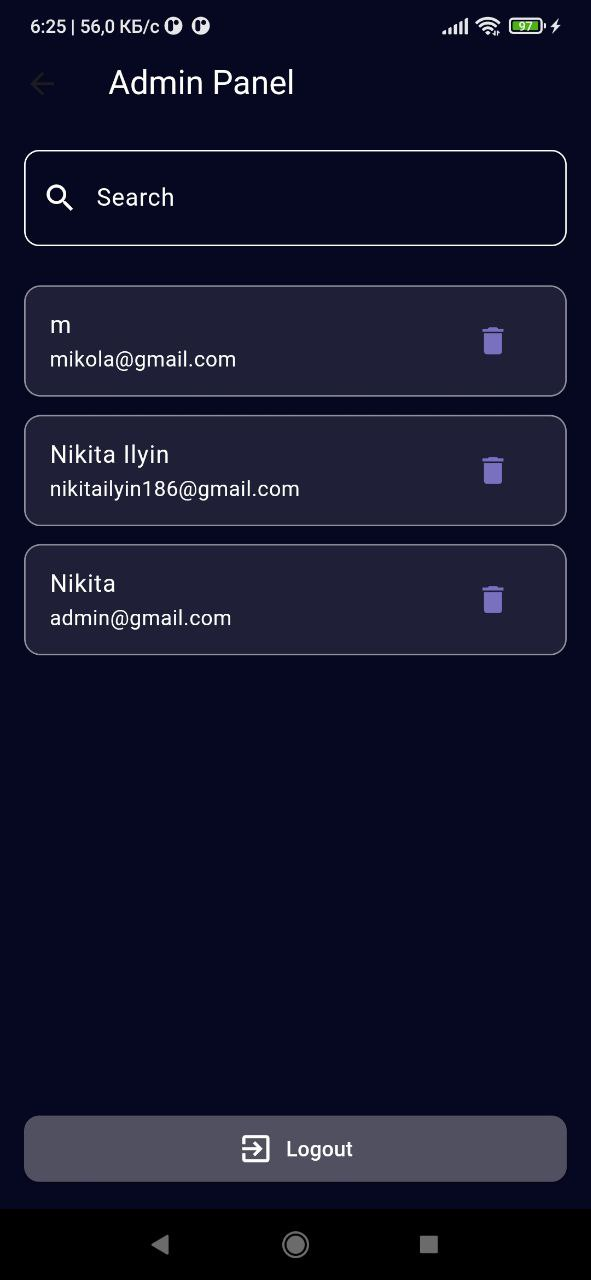


Рисунок 5.3 – Страница «Все пользователи» для администратора

При вводе ориентировочной строки в поле поиска, администратор может найти определенного пользователя. Показано на рисунке 5.4.

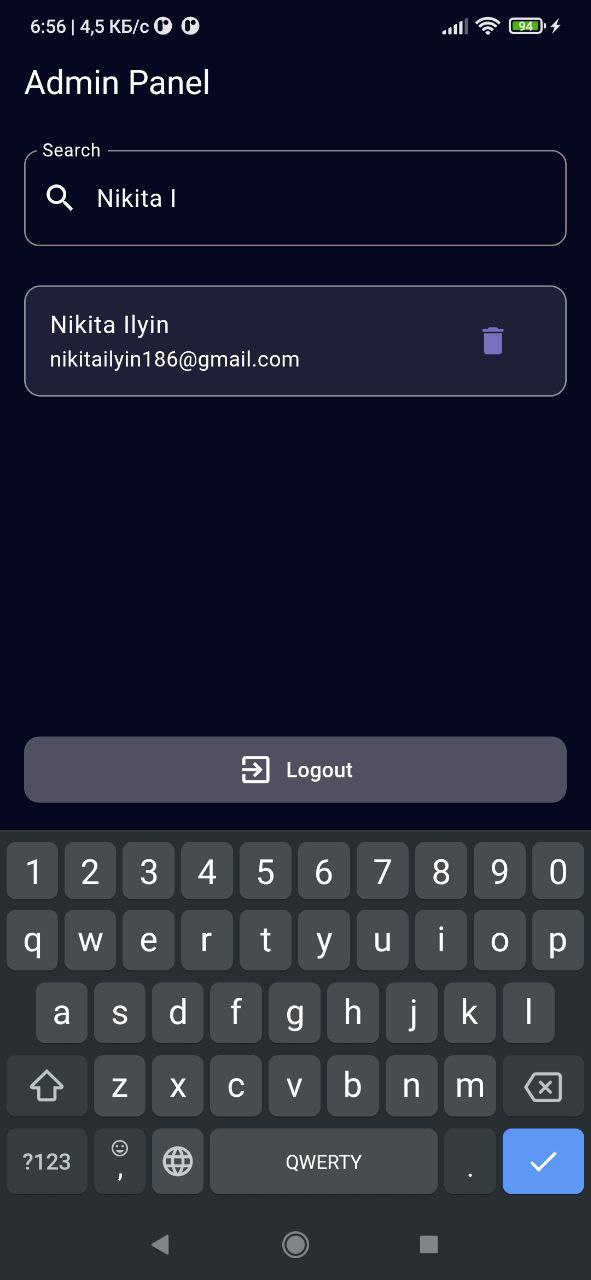


Рисунок 5.4 – Фильтрация пользователей для администратора

## Руководство пользователя

У пользователей ранее не зарегистрированных есть возможность пройти регистрацию. Он может ввести свой email и имя, а также придумать пароль для учетной записи. Страница регистрации показана на рисунке 5.5.

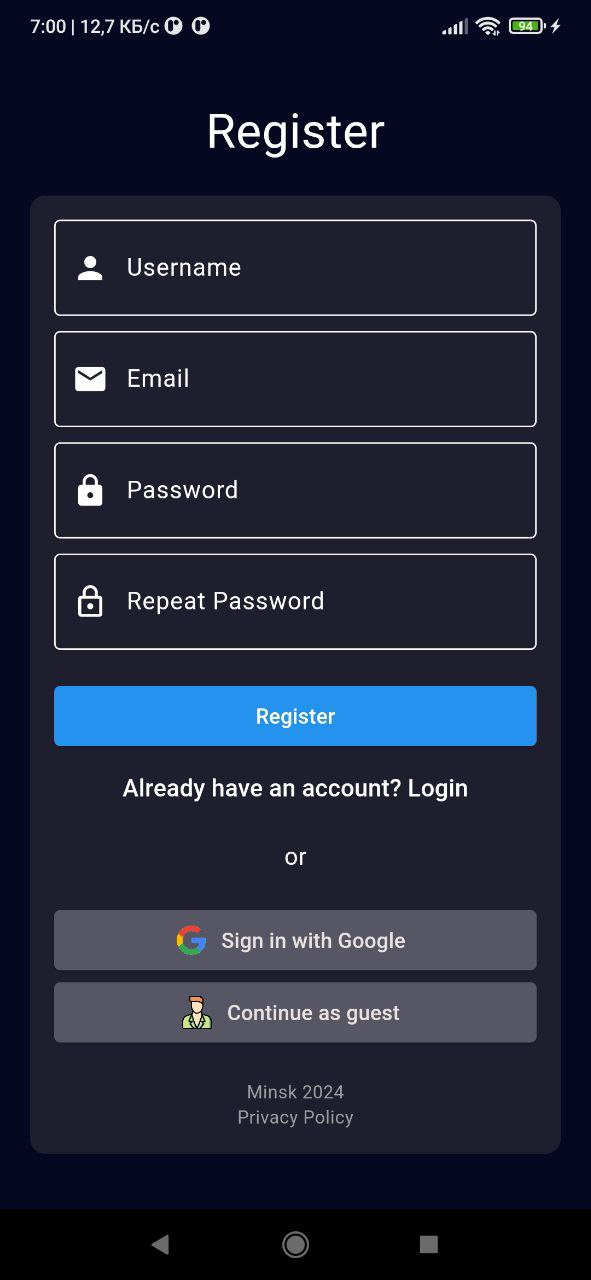


Рисунок 5.5 – Регистрация пользователя

После регистрации и авторизации пользователь переходит в основное окно приложения. Там он может просмотреть основную информацию о погоде в выбранном городе, имеется информация о температуре, скорости ветра и влажности, а также погоду на каждый час в течение суток. Показано на рисунке 5.6.



Рисунок 5.6 – Главная страница для пользователя

На странице поиска пользователь может добавить город для дальнейшего доступа к погоде в данном городе, выбирать основной город и удалять город. Можно увидеть на рисунке 5.7.



Рисунок 5.7 – Страница поиска для пользователя

На странице недельного прогноза погоды, пользователь может просмотреть информацию на каждый час в течение суток в выбранном городе, а также посмотреть температуру и облачность на каждый день на неделю вперед. Страница недельного прогноза показана на рисунке 5.8.



Рисунок 5.8 – Страница недельного прогноза погоды

Также пользователь может зайти на страницу настроек, где выбрать нужный ему пункт или выйти из аккаунта. Продемонстрировано на рисунке 5.9.

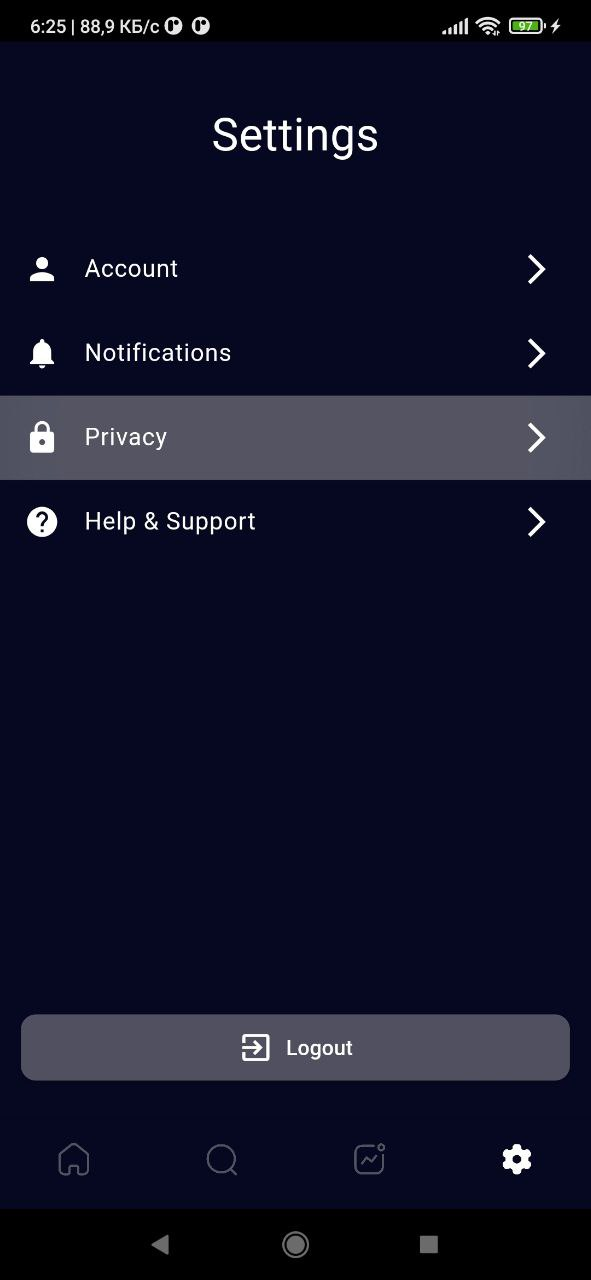


Рисунок 5.9 – Страница избранного

Также на главной странице можно просмотреть графики изменения метеорологических параметров в выбранном городе. Представлено на рисунке 5.10

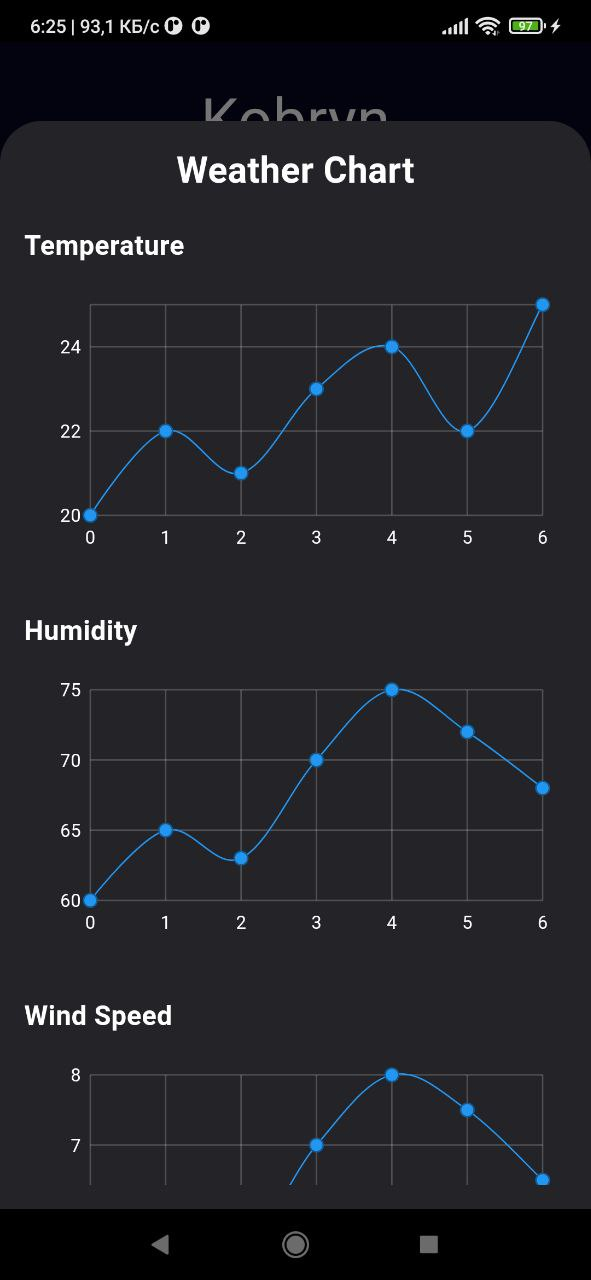


Рисунок 5.10 – Просмотр графиков метеорологических параметров

Таким образом, при первом запуске приложения пользователь попадает на экран Авторизация. Здесь он может ввести свои учетные данные (логин и пароль) для входа в систему.

Если у пользователя еще нет учетной записи, он может перейти на экран Регистрация. Для этого достаточно нажать на соответствующую кнопку. На экране регистрации пользователь вводит необходимую информацию для создания новой учетной записи.

После успешной авторизации пользователь попадает на Главную страницу. На этой странице отображается основная информация о текущей погоде и другая полезная информация.

С Главной страницы пользователь может перейти на страницу Поиск. Здесь он может искать информацию о погоде в различных городах. Для этого достаточно ввести название города в поисковую строку и нажать кнопку поиска.

Также с Главной страницы доступен переход на страницу Погода на неделю, где отображается прогноз погоды на ближайшие 7 дней для выбранного города. Пользователь может просматривать подробный прогноз на каждый день.

На странице Настройки пользователь может изменить параметры приложения в соответствии с его предпочтениями. Здесь доступны следующие подразделы: Политика конфиденциальности - информация о том, как приложение обрабатывает и защищает персональные данные пользователя; FAQ - ответы на часто задаваемые вопросы о работе приложения; Уведомления - настройка параметров уведомлений, которые пользователь хочет получать от приложения; Аккаунт - информация о профиле пользователя, где он может изменить свои учетные данные или выйти из учетной записи.

Авторизация и Регистрация взаимосвязаны, и пользователь может легко переключаться между ними.

# Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение, которое предоставляет собой приложение погоды.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* регистрация пользователей;
* авторизация пользователей;
* просмотр погоды на текущий день в выбранном городе;
* просмотр графиков погоды на неделю вперед по температуре, скорости ветра, влажности и давлению;
* выбирать несколько городов для просмотра погоды;
* просмотр погоды в выбранном городе на неделю;
* изменение данных аккаунта.

Администратору предоставляет следующие функциональные возможности:

* авторизация;
* просмотр всех пользователей;
* удаление пользователей;
* поиск пользователей.

Гость имеет возможность:

* регистрация;
* просмотр погоды только в одном городе.

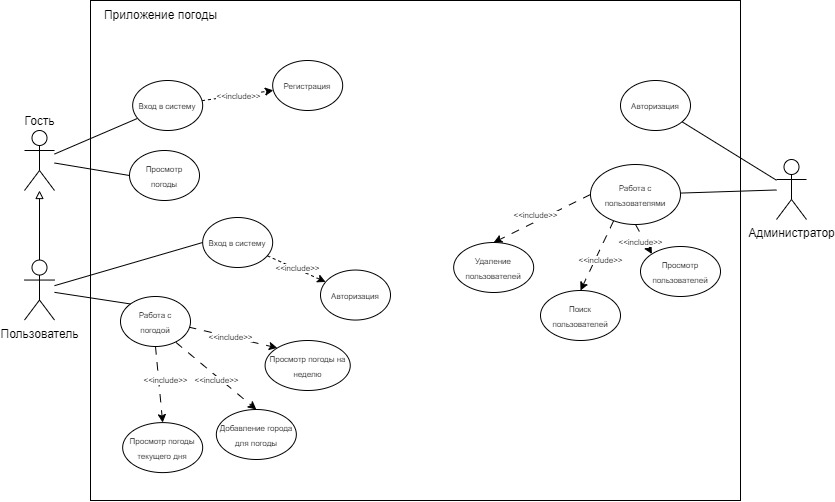
Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список используемых источников

1. Flutter [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/tools/devtools/overview – Дата доступа: 16.04.2024
2. Solutions on Stackoverflow [Электронный ресурс]. – Электронные данные. – Режим доступа: https://stackoverflow.com. – Дата доступа: 20.04.2024;
3. Чернышев, С.А. Основы Dart. 2–е издание / С.А. Чернышев. – Москва: Издательство "Наука", 2023. – 350 с.
4. Flutterfor [Электронный ресурс]. – Режим доступа: https://flutterfor.dev/ – Дата доступа: 20.03.2024
5. Dartflutter [Электронный ресурс]. – Режим доступа: https://dartflutter.ru/ – Дата доступа: 25.03.2024

# ПРИЛОЖЕНИЕ А. Диаграмма вариантов использований



# ПРИЛОЖЕНИЕ Б. Листинг удаления города у пользователя

Future<void> \_deleteCityFromUser(int userId, int cityId) async {

    final response = await http.delete(

      Uri.parse('${Config.url}/deleteCityFromUser'),

      headers: <String, String>{'Content-Type': 'application/json'},

      body: json.encode({'userId': userId, 'cityId': cityId}),

    );

    if (response.statusCode == 200) {

      final newCities = await getUserCities(userId);

      setState(() {

        cityCount -= 1;

        cities = newCities;

      });

      showDialog(

        context: context,

        builder: (BuildContext context) {

          return AlertDialog(

            title: Text('Success'),

            content: Text('City was removed'),

            actions: <Widget>[

              TextButton(

                child: Text('OK'),

                onPressed: () => Navigator.of(context).pop(),

              ),

            ],

          );

        },

      );

    } else {

      showDialog(

        context: context,

        builder: (BuildContext context) {

          return AlertDialog(

            title: Text('Error'),

            content: Text('Failed to remove the city.'),

            actions: <Widget>[

              TextButton(

                child: Text('OK'),

                onPressed: () => Navigator.of(context).pop(),

              ),

            ],

          );

        },

      );

    }

  }