**Содержание**

[Введение 5](#_Toc167100269)

[1 Аналитический обзор литературы 6](#_Toc167100270)

[1.1 Аналог «Пегас Туристик» 6](#_Toc167100271)

[1.2 Аналог «Travelata» 7](#_Toc167100272)

[1.3 Аналог «Expedia» 7](#_Toc167100273)

[2 Проектирование приложения 9](#_Toc167100274)

[2.1 Проектирование базы данных 9](#_Toc167100275)

[2.2 Описание информационных объектов и ограничений целостности 10](#_Toc167100276)

[2.2.1 Коллекция users 10](#_Toc167100277)

[2.2.2 Коллекция tours 10](#_Toc167100278)

[2.2.3 Коллекция hotels 10](#_Toc167100279)

[2.2.4 Коллекция bookings 11](#_Toc167100280)

[2.3 Проектирование мобильного приложения 11](#_Toc167100281)

[3 Реализация приложения 14](#_Toc167100282)

[3.1 Технические средства разработки 14](#_Toc167100283)

[3.2 Разработка мобильного приложения 15](#_Toc167100284)

[4 Тестирование приложения 17](#_Toc167100285)

[5 Руководство по использованию 20](#_Toc167100286)

[В данном разделе будет описано руководство по использованию для пользователя, администратора и главного администратора. 20](#_Toc167100287)

[5.1 Руководство пользователя 20](#_Toc167100288)

[5.2 Руководство администратора 22](#_Toc167100289)

[Заключение 24](#_Toc167100290)

[Список использованных источников 25](#_Toc167100291)

[ПРИЛОЖЕНИЕ А - UML диаграмма использований 26](#_Toc167100292)

[ПРИЛОЖЕНИЕ Б - Класс AuthService 26](#_Toc167100293)

# Введение

В современном мире, где путешествия стали неотъемлемой частью жизни, мобильные приложения играют ключевую роль в планировании и организации поездок. Они предлагают удобный доступ к информации о направлениях, предоставляют возможность бронирования билетов и отелей, а также позволяют управлять поездками в режиме реального времени. Таким образом, мобильные приложения для туристических агентств становятся незаменимым инструментом для привлечения клиентов и повышения их удовлетворенности.

Разработка мобильного приложения для туристического агентства позволит клиентам легко находить и бронировать туры, получать доступ к информации об отелях, изучать достопримечательности и управлять своим профилем. Благодаря удобному интерфейсу и интуитивно понятной навигации, приложение станет надежным помощником в планировании путешествий.

В рамках курсового проекта поставлены следующие задачи:

* Разработка системы ролей пользователей, включающей в себя роли главного администратора, администратора и пользователя;
* Реализация функционала добавления, удаления и изменения информации о турах;
* Реализация функционала добавления, удаления и изменения информации об отелях;
* Реализация функционала изменения имени и фотографии профиля пользователя;
* Реализация системы бронирования туров;

Основной целью курсового проекта является разработка мобильного приложения для туристического агентства, которое будет удовлетворять потребности как клиентов, так и сотрудников агентства, предоставляя им удобный и эффективный инструмент для взаимодействия.

# 1 Аналитический обзор литературы

Изучение уже существующих решений – это ключевой этап в разработке любого программного продукта. Анализ аналогов позволяет нам разобраться в том, какие решения уже предлагаются, какие функции они реализуют и какие потребности удовлетворяют а также увидеть, какие решения оказались неудачными и какие проблемы они создавали, чтобы избежать их повторения в своем проекте.

## **Аналог «Пегас Туристик»**

Пегас Туристик – это мобильное приложение, разработанное компанией Besttours LLC, представлено на рисунке 1.1. Оно предназначено для клиентов турагентства «А-ТУР» и предлагает более тысячи туров в различные страны мира.

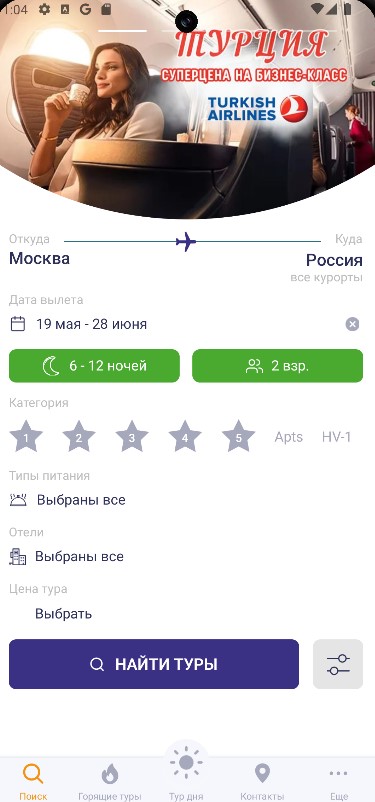


Рисунок 1.1 – Приложение «Пегас Туристик»

Преимуществами этого приложения является широкие возможности поиска тура, специальная вкладка «Тур дня», предлагающая один выгодный клиенту тур, а также служба поддержки, для связи с которой в приложении есть несколько способов.

Из недостатков, следует отметить, что поиск туров занимает длительное время, список вариантов пребывания в отеле может достигать большой длины, при этом нет возможности их сортировки или фильтрации, наличие аббревиатур и сокращений усложняет восприятие информации.

## **1.2 Аналог** «Travelata»

Мобильное приложение Travelata – это приложение от компании Ads Travelata, представленное на рисунке 1.2, предназначенное для поиска горящих туров по всему миру. Оно предлагает поиск по более чем 120 туроператорам, с постоянной поддержкой. Кроме того, в нём предоставляется выбор из туров с Ж/Д.

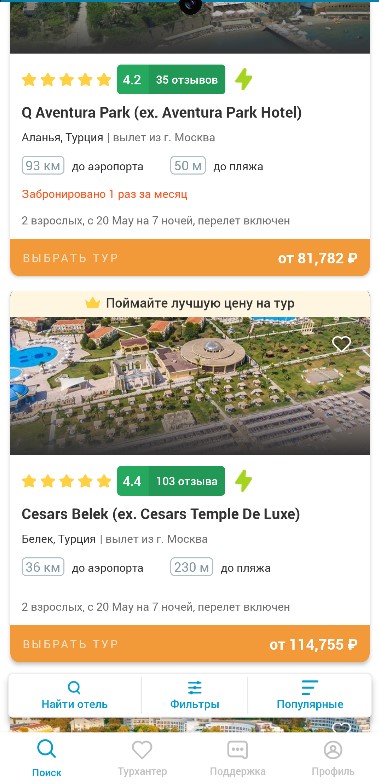


Рисунок 1.2 – Приложение «Travelata»

Преимущество данного приложения в том, что в нём отображены отзывы пользователей о турах. Кроме того, Travelata позволяет узнать такую актуальную информацию, как расстояние до аэропорта и до пляжа. Ещё один плюс приложения – возможность добавить тур в понравившиеся, что обеспечивает быстрый доступ к туру.

Основной недостаток приложения Travelata – скудный функционал для незарегистрированного пользователя.

## **1.3 Аналог** «Expedia»

Мобильное приложение Expedia – это приложение от компании Expedia, предназначенное для поиска туров, транспорта, а также организации путешествия на мобильных устройствах. Оно имеет широкий функционал и дополнительные возможности, такие как просмотр тематических видео, встроенный чат-GPT.

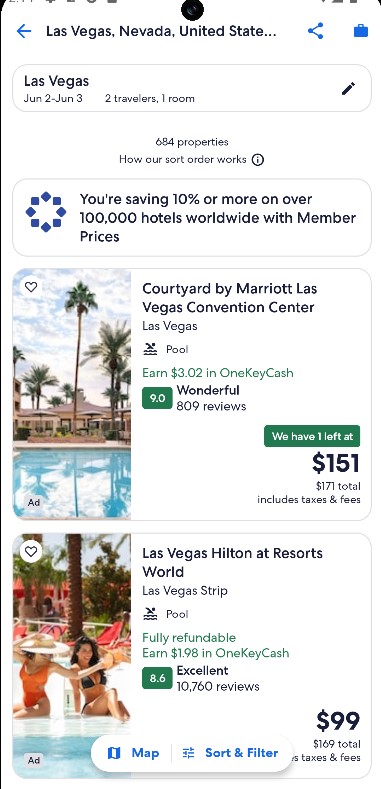


Рисунок 1.3 – Приложение «Expedia»

Плюсы приложения многочисленны: удобный интерфейс, наличие просмотра мест путешествий на карте, наличие раздела с уведомлениями и сообщениями от отелей, внутренний кошелёк, привязанный к профилю.

Недостатком приложения можно назвать его низкую производительность, особенно на старых мобильных устройствах.

# 2 Проектирование приложения

Разработка архитектуры программного продукта является ключевым этапом, который заложит основу для всего процесса создания. Тщательно спланированный дизайн предотвращает возникновение ошибок и способствует ускорению и оптимизации процесса разработки. В ходе этой фазы происходит определение ключевых функций приложения, его структуры, а также логики взаимодействия с пользователем.

## Проектирование базы данных

Создание структуры данных для NoSQL систем хранения информации представляет собой задачу выстраивания архитектуры данных, которая оптимально соответствует потребностям программного обеспечения. В контрасте с классическими реляционными системами, NoSQL решения часто отходят от табличной организации и фиксированных схем, что дает возможность разрабатывать более адаптивные и расширяемые структуры данных. Разрабатывая архитектуру данных для NoSQL, критично принимать во внимание её вид (например, ориентированная на документы, колонки, пары ключ-значение или графы), а также ожидания по скорости работы, масштабированию, надёжности и особенности программы, которая будет её использовать.

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

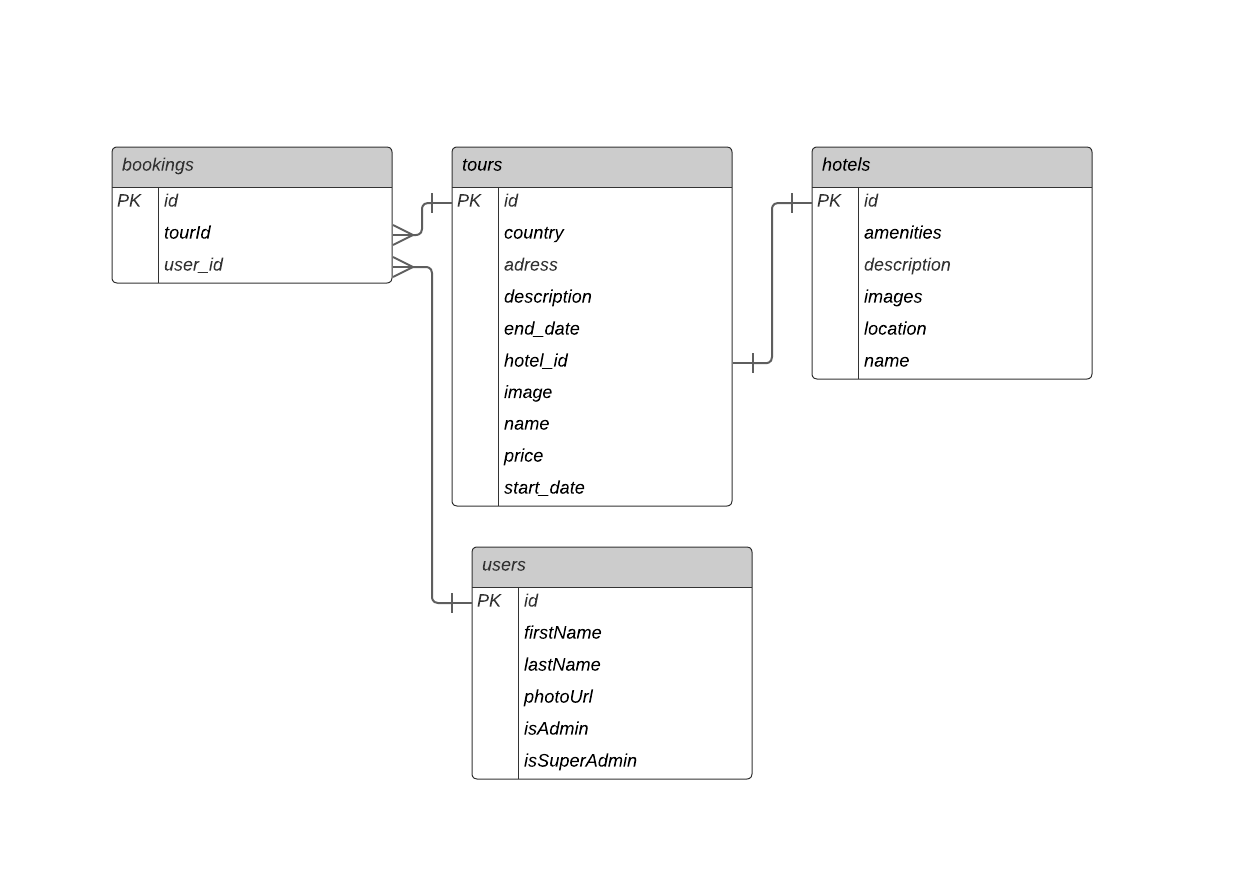


Рисунок 2.1 – Модель БД

Созданная база данных содержит в себе 4 коллекции: users, tours, hotels, bookings. Коллекция hotels содержит список amenities и images.

2.2 Описание информационных объектов и ограничений целостности

Далее будет описана структура каждой коллекции. В коллекции users хранятся данные для идентификации пользователей, зарегистрированных в приложении, в коллекции tours хранится основная информация туров, в hotels выделена информация, относящаяся к отелям, коллекция bookings хранит информацию о бронирования отелей пользователями.

### **2.2.1 Коллекция users**

Коллекция состоит из следующих полей:

* id. Хранит уникальный идентификатор пользователя;
* firstName. Хранит имя пользователя;
* lastName. Хранит фамилию пользователя
* photoUrl. Хранит ссылку на фотографию профиля пользователя;
* isAdmin. Является ли пользователь администратором

− isSuperAdmin. Является ли пользователь главным администратором.

### **2.2.2 Коллекция tours**

Коллекция products состоит из следующих полей:

* id. Хранит уникальный идентификатор тура;
* country. Хранит страну тура;
* description. Хранит описание тура;
* end\_date. Хранит дату окончания тура;
* hotel\_id. Хранит ссылку на соответствующий туру отель;
* image. Хранит ссылку на изображение тура;
* name. Хранит название тура;
* price. Хранит цену тура;
* start\_date. Хранит дату начала тура;

### **2.2.3 Коллекция hotels**

В состав коллекции входят следующие поля:

* id. Хранит уникальный идентификатор отеля;
* amenities. Хранит список удобств отеля;
* description. Хранит описание отеля;
* images. Хранит ссылки на изображения отеля;
* location. Хранит информацию о месте расположения отеля.
* name. Хранит название отеля

### **2.2.4 Коллекция bookings**

В состав коллекции входят следующие поля:

* tourId. Хранит ссылку на тур
* userId. Хранит ссылку на пользователя

## 2.3 Проектирование мобильного приложения

Проектирование мобильных приложений включает в себя множество аспектов: определение ролей, архитектуры приложения, его бизнес-задач, а также проектирование страниц приложения.

Ограниченное пространство экрана мобильных устройств требует особого внимания при проектировании каждой страницы. Ключевые принципы - интуитивность и лаконичность. Информация должна быть структурирована так, чтобы пользователь легко ее воспринимал, не испытывая информационной перегрузки.

Дизайн должен быть простым и чистым, с грамотным балансом между текстом, изображениями и свободным пространством.

Навигация в мобильных приложениях играет ключевую роль в удобстве использования. Она должна быть простой и интуитивной, позволяя пользователям легко перемещаться между страницами.

Стандартные элементы навигации, такие как меню, вкладки и кнопка "Назад", должны быть использованы грамотно, учитывая контекст приложения и потребности пользователей. Правильно спроектированная навигация делает приложение максимально комфортным и удобным в использовании.

При первом входе в приложение пользователя встречает домашняя страница, на которой расположен список всех доступных туров.

Если пользователь является администратором, то есть кнопка добавления тура, а также туры могут быть изменены или удалены.

Далее идёт страница поиска. Здесь можно применить фильтры к существующим турам, например фильтрация по цене или по стране назначения, и получить список туров с определёнными характеристиками.

На странице закладок можно увидеть сделанные пользователем закладки, если они есть. Если их нет, то сообщение об этом будет отображено пользователю.

На странице профиля отображается текущая информация о пользователе или гостевая информация. Если пользователь авторизован, то он может изменить свои имя и фамилию, а также фотографию профиля. Также есть возможность выйти из учетной записи.

На рисунке 2.2 представлена схема навигации гостя.

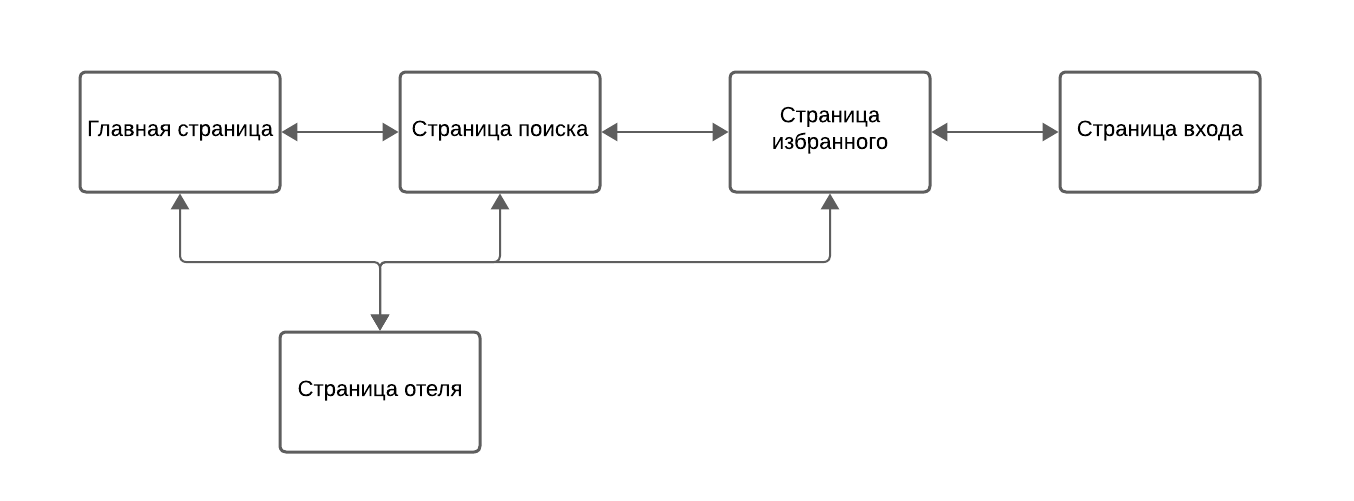
****

Рисунок 2.2 – Схема навигации для гостя

Страница приветствия показывается всем пользователям при первом входе.

На рисунке 2.3 представлена схема навигации пользователя.

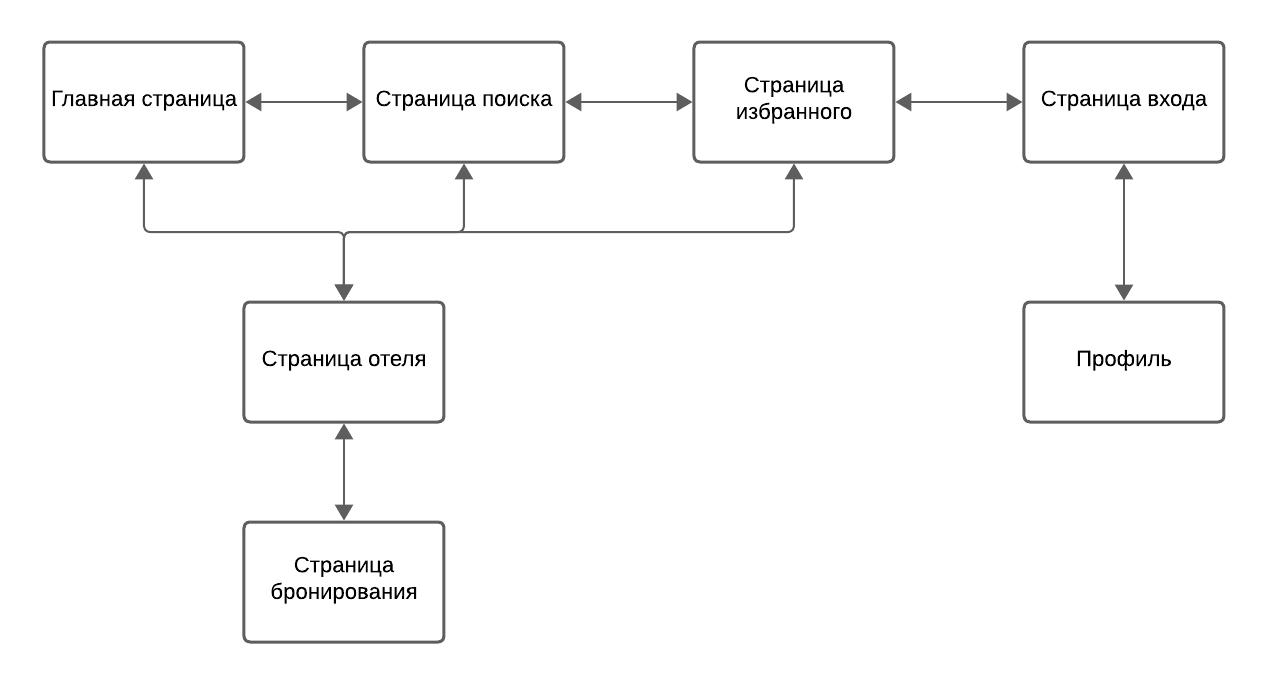


Рисунок 2.3 – Схема навигации для пользователя

На рисунке 2.4 представлена схема навигации администратора.

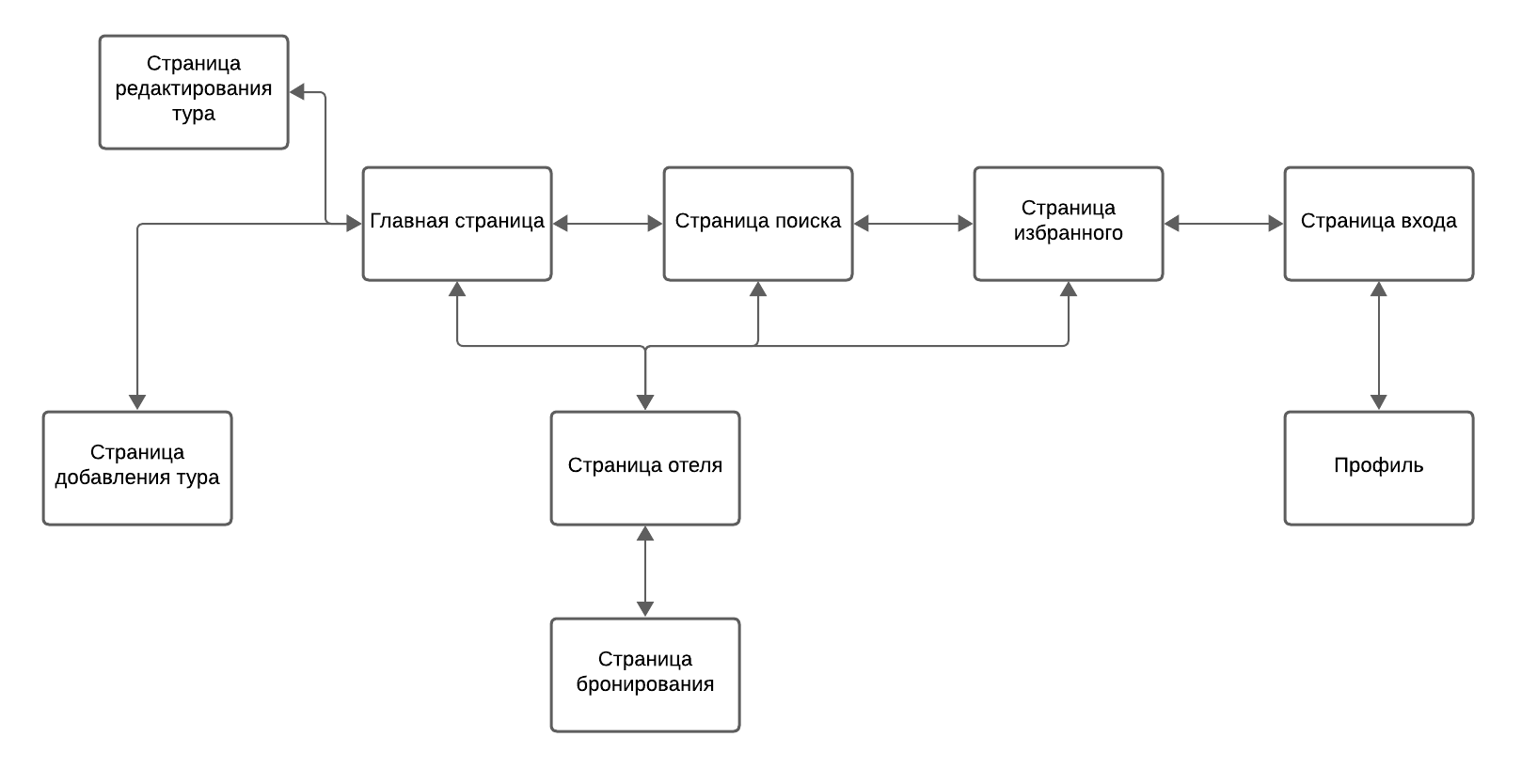


Рисунок 2.4 – Схема навигации администратора

Диаграмма вариантов использований представлена в приложении А.

# 3 Реализация приложения

Реализация приложения – это процесс, который превращает проект программного обеспечения в работающий продукт. Он включает в себя написание кода, интеграцию различных компонентов системы, тестирование для обнаружения и исправления ошибок, а также оптимизацию для повышения производительности. Реализация требует глубоких технических знаний, навыков программирования, а также понимания бизнес-требований и пользовательских потребностей. Это ключевой этап в разработке программного обеспечения, влияющий на качество и успех конечного продукта.

## 3.1 Технические средства разработки

Для реализации курсового проекта используется язык программирования Dart в связке с фреймворком Flutter.

Dart - язык программирования, разработанный Google. Представленный в 2011 году, он позиционируется как замена или альтернатива JavaScript. Dart - объектно-ориентированный, веб-фреймворк, сценарный, императивный, рефлексивный язык программирования, язык функционального программирования и мультипарадигмальный.

Основная область применения Dart сегодня - разработка графических приложений с помощью кроссплатформенного фреймворка Flutter. Dart предлагает современные возможности, такие как безопасность от null и сопоставление с образцом. Он также поддерживает горячую перезагрузку, позволяющую видеть результаты изменений в коде в режиме реального времени.

Flutter – это открытый UI-инструментарий, разработанный Google для создания красивых, компилируемых нативно приложений для мобильных устройств, веб и настольных компьютеров из одной кодовой базы. Flutter работает с существующим кодом и используется разработчиками и организациями по всему миру. Ключевые преимущества Flutter: быстрая разработка, адаптивность, кроссплатформенность.

Flutter поддерживается и используется Google, ему доверяют известные бренды по всему миру, а также активное сообщество разработчиков.

Firebase - платформа разработки приложений от Google, которая помогает создавать, улучшать и масштабировать приложения. Firebase предлагает ряд сервисов, включая Authentication, Firestore и Cloud Storage.

Firebase Authentication предоставляет бэкенд, SDK и готовые UI-библиотеки для аутентификации пользователей в вашем приложении. Он поддерживает аутентификацию с использованием паролей, номеров телефонов, популярных федеративных поставщиков идентификации, таких как Google, Facebook и Twitter.

Firestore – это гибкая, масштабируемая база данных для мобильных, веб- и серверных разработок от Firebase и Google Cloud. Firestore синхронизирует ваши данные на всех клиентских приложениях с помощью слушателей в реальном времени и предлагает поддержку в режиме офлайн для мобильных и веб-приложений.

Firestore – это гибкая, масштабируемая база данных для мобильных, веб- и серверных разработок от Firebase и Google Cloud. Firestore сохраняет ваши данные в синхронизации на всех клиентских приложениях с помощью слушателей в реальном времени и предлагает поддержку в режиме офлайн для мобильных и веб-приложений.

Cloud Storage for Firebase – это мощный, простой и экономичный сервис хранения объектов, построенный для работы в масштабе Google. SDK Firebase для Cloud Storage добавляет безопасность Google к загрузке и скачиванию файлов для ваших Firebase-приложений, независимо от качества сети. Вы можете использовать наши клиентские SDK для хранения изображений, аудио, видео или другого контента, созданного пользователями. Для данного приложения используются Firebase Authentication, Firebase Cloud и Firebase Storage.

SQLite – это небольшая, встроенная база данных, предназначенная для использования в настольных и мобильных устройствах. Она хорошо подходит для приложений, где требуется простота и эффективность, а также отсутствие зависимости от серверных баз данных. SQLite широко используется в приложениях Flutter благодаря своей простоте, надежности и интеграции с языком Dart.

## 3.2 Разработка мобильного приложения

Структура приложения состоит из нескольки папок: Screens, Widgets. Папка Screens содержит все основные экраны приложения, папка Widgets – элементы пользовательского интерфейса, не занимающие полный экран, такие как элементы списков. Кроме этого проект содержит несколько файлов, содержащих функции реализации связи с базами данных firestore, sqlite, а акже с сервисами firebase authenticator и firebase storage.

При реализации приложения использовано асинхронное программирование.

Данные хранятся локально и в удаленном хранилище Firestore. Для увеличения быстродействия данные в локальной БД синхронизируются с данными удаленной БД, после чего в приложении используется локальная БД.

Ниже приведена реализация метода, который синхронезирует базу данных firestore, хранящую данные о пользователях, с сервисом firebase authenticator. При регистрации нового пользователя в firestore создаётся новый документ, ключом которого является уникальный идентификатор пользователя в firestore.

Future<User?> registerWithEmailAndPassword(String email, String password) async {

try {

final UserCredential result = await \_auth.createUserWithEmailAndPassword(

email: email,

password: password,

);

if (result.user != null) {

await createUserDocument(result.user!); // Создаем документ пользователя в Firestore

}

return result.user;

} catch (e) {

print('Error registering: $e');

rethrow;

}

Листинг 3.1 – Синхронизация БД и firebase authenticator

Когда пользователь переходит на страницу, функционал которой различается в зависимости от роли пользователя, выполняется функция проверки, которая позволяет отличить администратора от обычного пользователя. Этот функционал реализован с помощью базы данных и хранимых в документах пользователей булевых полей.

База данных sqlite используется в проекте для хранения закладок пользователя. В ней созданы сущности, повторяющие коллекции туров и отелей, а также созданы дополнительные таблицы удобств и изображений. Это необходимо, потому что sql решения требуют иного подхода к организации хранения данныхю, в отличие от no-sql.

Листинг всего класса для работы с сервисом firebase authenticator представлен в приложении Б.

# 4 Тестирование приложения

Тестирование мобильного программного обеспечения - это процесс проверки функциональности, производительности, безопасности и других ключевых аспектов мобильных приложений перед их выпуском. Это включает в себя проверку приложения на различных устройствах, операционных системах и конфигурациях, чтобы убедиться, что оно работает корректно во всех возможных сценариях. Тестирование может включать в себя автоматические и ручные методы, а также различные типы тестов, такие как функциональное тестирование, тестирование производительности, тестирование безопасности и тестирование пользовательского интерфейса. Цель тестирования мобильного ПО - обеспечить высокое качество приложения и предотвратить проблемы, которые могут негативно повлиять на пользовательский опыт.

Ниже приведен результат тстирования страницы авторизации, рисунок 4.1. При вводе некорректных данных, поля подчеркиваются красным, появляются сообщение о том, какое условие ввода не выполнено сообщаеющий об этом.



Рисунок 4.1 – Тестирование авторизации

Далее приведен тест страницы регистрации. При попытке ввести некорректную информацию пользователь также получает ошибку, рисунок 4.2.

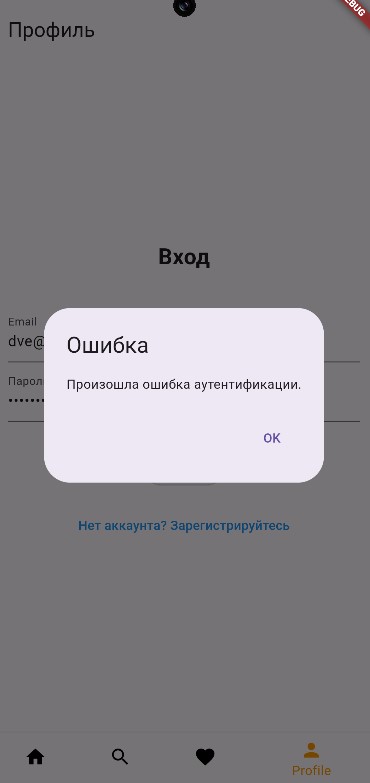


Рисунок 4.2 – Тестирование регистрации

Ниже также приведен тест сценария, когда неавторизованный пользователь (гость) пытается получить доступ к ресурсам, к которым у него нет доступа, таким как бронирование тура, рисунок 4.3.

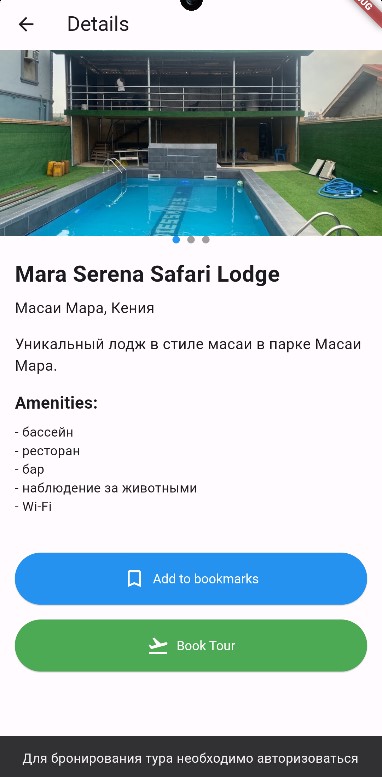


Рисунок 4.3 – Тестирование функционала гостя

Кроме выше описаных ситуаций, прложение также обрабатывает и другие, например при редактировании или добавлении туров и отелей.

# 5 Руководство по использованию

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

## 5.1 Руководство пользователя

При входе в приложение пользователь попадает на главную страницу, рисунок 5.1.

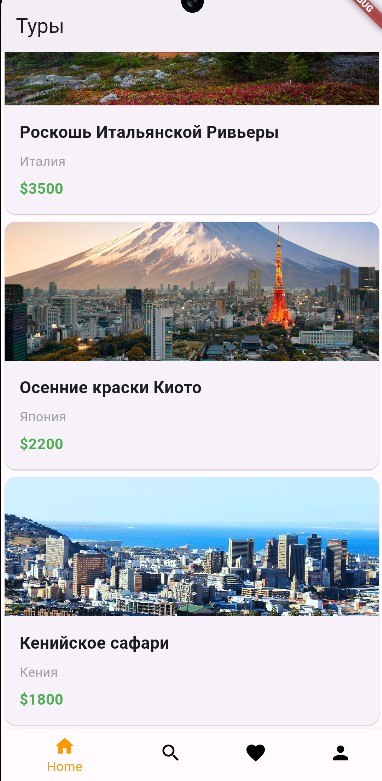


Рисунок 5.1 – Главная страница

Тут пользователь может перейти на страницу поиска, рисунок 5.2.

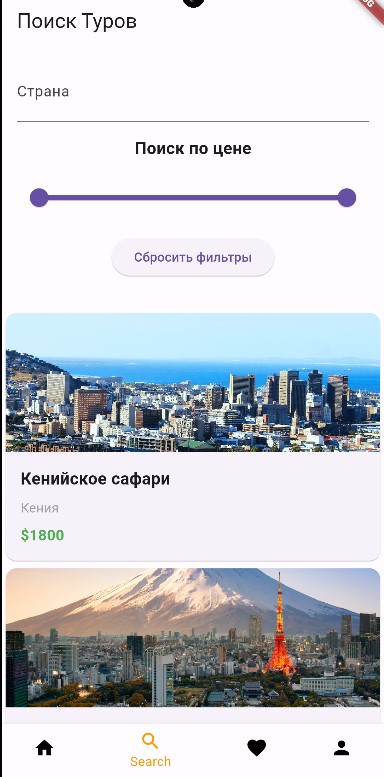


Рисунок 5.2 – Страница каталога

Пользователь может также перейти к странице закладок, рисунок 5.3.



Рисунок 5.3 – Страница поиска

Нажав на отель пользователь может посмотреть информацию о нем открыв страницу отеля, где пользователь может добавить его в избранное, рисунок 5.4.



Рисунок 5.4 – Страница товара

Из главной страницы можно перейти к странице профиля. Страница профиля представлена на рисунке 5.5.

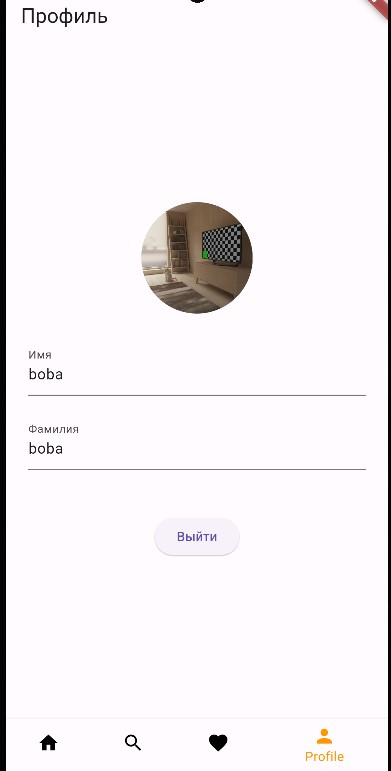


Рисунок 5.5 – Страница профиля

## 5.2 Руководство администратора

Нажав на опции меню сверху слева администратор может перейти к странице добавления нового тура, рисунок 5.6.

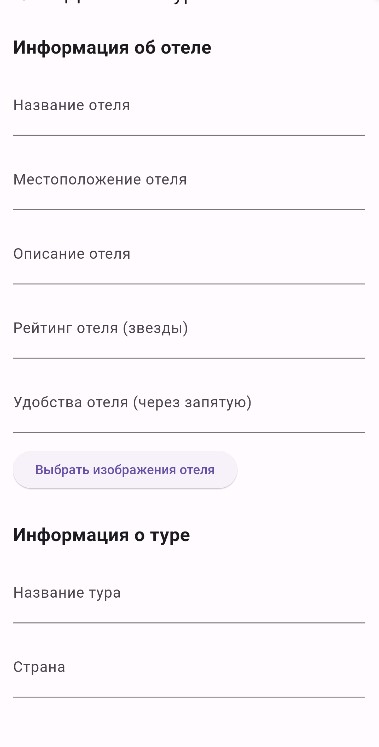


Рисунок 5.6 – Страница добавления тура

Администратор может также удалять и изменять туры, используя соответствующие кнопки на главной странице (home screen).

# Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение для турагентства.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* авторизация;
* просмотр и поиск туров;
* добавление туров в избранное, оформление бронирования;
* фильтрация туров по их параметрам;
* изменение профиля;

Администратору предоставляет следующие функциональные возможности:

* авторизация;
* просмотр и поиск туров;
* добавление товаров в избранное, оформление бронирования;
* изменение профиля;
* добавление, удаление и изменение туров и отелей;

Главный администратор обладает следующими возможностями:

* авторизация;
* просмотр и поиск туров;
* добавление товаров в избранное, оформление бронирования;
* изменение профиля;
* добавление, удаление и изменение туров и отелей;
* наделение пользователя правами администратора;

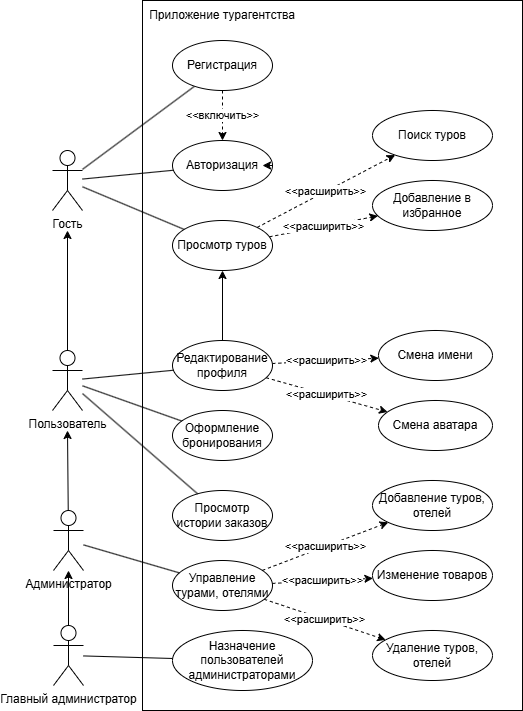
Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

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

# Список использованных источников

1. Flutter Documentation [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/ – Дата доступа: 19.03.2024
2. Dart Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dart.dev/guides/ – Дата доступа: 25.03.2024;
3. Firebase Documentation [Электронный ресурс]. – Режим доступа: https://firebase.google.com/docs/ – Дата доступа: 22.04.2024;
4. Sqlite Documentation [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/docs.html – Дата доступа: 27.04.2024.

# ПРИЛОЖЕНИЕ А - UML диаграмма использований



# ПРИЛОЖЕНИЕ Б - Класс AuthService

import 'dart:io';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:file\_picker/file\_picker.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:firebase\_storage/firebase\_storage.dart';

import 'package:image\_picker/image\_picker.dart';

class AuthService {

final FirebaseAuth \_auth;

AuthService({FirebaseAuth? auth}) : \_auth = auth ?? FirebaseAuth.instance;

Future<User?> signInWithEmailAndPassword(String email, String password) async {

try {

final UserCredential result = await \_auth.signInWithEmailAndPassword(

email: email,

password: password,

);

if (result.user != null) {

// Проверяем, существует ли документ пользователя в Firestore

final docSnapshot = await FirebaseFirestore.instance

.collection('users')

.doc(result.user!.uid)

.get();

if (!docSnapshot.exists) {

await createUserDocument(result.user!); // Создаем документ, если он не существует

}

return result.user;

}

} catch (e) {

print('Error signing in: $e');

rethrow;

}

}

Future<bool> isCurrentUserSuperAdmin() async {

final user = \_auth.currentUser;

if (user == null) {

return false;

}

final docSnapshot = await FirebaseFirestore.instance

.collection('users')

.doc(user.uid)

.get();

if (docSnapshot.exists) {

final isSuperAdmin = docSnapshot.data()!['isSuperAdmin'];

if (isSuperAdmin != null) {

return isSuperAdmin as bool; // Приводим к типу bool только если isSuperAdmin не null

} else {

return false; // Возвращаем false, если isSuperAdmin равно null

}

} else {

return false;

}

}

Future<User?> registerWithEmailAndPassword(String email, String password) async {

try {

final UserCredential result = await \_auth.createUserWithEmailAndPassword(

email: email,

password: password,

);

if (result.user != null) {

await createUserDocument(result.user!); // Создаем документ пользователя в Firestore

}

return result.user;

} catch (e) {

print('Error registering: $e');

rethrow;

}

}

Future<User?> getCurrentUser() async {

try {

final User? currentUser = \_auth.currentUser;

return currentUser;

} catch (e) {

print('Error getting current user: $e');

rethrow;

}

}

Future<void> signOut() async {

try {

await \_auth.signOut();

} catch (e) {

print('Error signing out: $e');

rethrow;

}

}

// auth\_service.dart

// ... (остальной код) ...

Future<void> createUserDocument(User user) async {

final firestore = FirebaseFirestore.instance;

// Создаем документ с ID, равным UID пользователя

await firestore.collection('users').doc(user.uid).set({

'firstName': user.displayName?.split(' ')[0] ?? '',

'lastName': user.displayName?.split(' ')[1] ?? '',

// Добавьте другие поля, которые хотите сохранить для пользователя

'isAdmin': false, // Добавляем поле isAuth со значением false по умолчанию

});

}

Future<void> updateUserProfile(

User user, String firstName, String lastName, PlatformFile? image) async {

final firestore = FirebaseFirestore.instance;

// Загружаем изображение профиля, если оно выбрано

String? photoURL;

if (image != null && image.path != null) {

// Проверяем, что image.path не null

final storageRef = FirebaseStorage.instance

.ref()

.child('user\_profiles/${user.uid}.jpg');

await storageRef.putFile(File(image.path!)); // Используем ! для указания, что image.path не null

photoURL = await storageRef.getDownloadURL();

}

// Обновляем поля firstName, lastName и photoURL

await firestore.collection('users').doc(user.uid).update({

'firstName': firstName,

'lastName': lastName,

if (photoURL != null) 'photoURL': photoURL,

});

// Обновляем displayName пользователя в FirebaseAuth

await user.updateDisplayName('$firstName $lastName');

// При необходимости, обновите photoURL пользователя в FirebaseAuth

if (photoURL != null) {

await user.updatePhotoURL(photoURL);

}

}

Future<bool> isCurrentUserAdmin() async {

final user = \_auth.currentUser;

if (user == null) {

return false; // Пользователь не авторизован (включая гостя)

}

final docSnapshot = await FirebaseFirestore.instance

.collection('users')

.doc(user.uid)

.get();

if (docSnapshot.exists) {

return docSnapshot.data()!['isAdmin'] as bool; // Проверяем поле isAdmin

} else {

return false; // Документ пользователя не найден (гость или ошибка)

}

}