МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения 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 КУРСОВОГО ПРОЕКТА:**

По дисциплине «Программирование Мобильных Систем»

Тема **«**Мобильное приложение для заказа междугородних маршруток»

**Исполнитель**

студент 3 курса группы 8 Кирпиченко В.А.

подпись, дата

**Руководитель**

ассистент Уласевич Н.И.

должность, ученая степень, ученое звание подпись, дата

Допущен(а) к защите \_

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

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Уласевич Н.И.

подпись дата инициалы и фамилия

**Содержание**

[Введение 5](#_Toc167038679)

[1 Постановка задачи 6](#_Toc167038680)

[2 Обзор аналогичных решений 7](#_Toc167038681)

[2.1 Анализ приложения «Атлас» 7](#_Toc167038682)

[2.2 Анализ приложения «Naminsk.by» 11](#_Toc167038683)

[2.3 Вывод по разделу 15](#_Toc167038684)

[3 Проектирования приложения 16](#_Toc167038685)

[3.1 Проектирование базы данных 16](#_Toc167038686)

[3.2 Проектирование ролей 19](#_Toc167038687)

[3.3 Проектирование навигации 19](#_Toc167038688)

[3.4 Описание структуры проекта 20](#_Toc167038689)

[3.5 Вывод по разделу 22](#_Toc167038690)

[4 Реализация приложения 23](#_Toc167038691)

[4.1 База данных и взаимодействие с ней 23](#_Toc167038692)

[4.2 Авторизация, регистрация, личный кабинет пользователя 24](#_Toc167038693)

[4.3 Панель администратора 25](#_Toc167038694)

[4.4 Процесс поиска и бронирования билета 26](#_Toc167038695)

[4.5 Страница заказов пользователя 29](#_Toc167038696)

[4.6 Вывод по разделу 29](#_Toc167038697)

[5 Тестирование приложения 30](#_Toc167038698)

[5.1 Гость 30](#_Toc167038699)

[5.2 Авторизация и просмотр заказов 31](#_Toc167038700)

[5.3 Использование панели водителя 33](#_Toc167038701)

[5.4 Использование панели администратора 34](#_Toc167038702)

[5.5 Вывод по разделу 35](#_Toc167038703)

[6 Краткое описание приложения для демонстрации 36](#_Toc167038704)

[7 Руководство пользователя 37](#_Toc167038705)

[7.1 Руководство для гостя 38](#_Toc167038706)

[7.2 Руководство для авторизованного пользователя 39](#_Toc167038707)

[7.3 Руководство для водителя 41](#_Toc167038708)

[7.4 Руководство для администратора 41](#_Toc167038709)

[7.5 Вывод по разделу 43](#_Toc167038710)

[Заключение 44](#_Toc167038711)

[Список используемых источников 45](#_Toc167038712)

[Приложение А. Классы для работы с базой данных 46](#_Toc167038713)

# **Введение**

В настоящее время автовокзалы являются одним из самых популярных и востребованных транспортных узлов. Ежедневно множество людей предпочитают путешествовать на маршрутках, и для обеспечения высокого уровня организации и управления необходимо использовать современные базы данных и программное обеспечение. В этом контексте курсовой проект, имеет огромную практическую значимость для автовокзалов, поскольку он позволяет улучшить обслуживание пассажиров и оптимизировать работу персонала, что, в свою очередь, способствует росту доходов организации.

Одним из ключевых аспектов улучшения обслуживания является использование мобильного приложения для бронирования и покупки билетов. Такое приложение предоставляет значительное преимущество перед конкурентами, которые все еще предлагают более традиционные способы обслуживания. Благодаря мобильному приложению, автовокзал сможет собирать и анализировать данные о предпочтениях и поведении пассажиров, что позволит адаптировать услуги под потребности клиентов и повысить удовлетворенность пользователей.

Основной целью данного курсового проекта является разработка мобильного приложения для автовокзала и эффективное хранение информации, необходимой для его работы. Готовое приложение должно предоставлять пользователям возможность находить и бронировать нужные им билеты для поездок в другие города.

Для разработки приложения была выбрана платформа Flutter, которая обеспечивает адаптивность под различные пользовательские устройства. Это позволит охватить широкую аудиторию пользователей. Кроме того, разработка мобильного приложения на платформе Flutter обеспечит единый код, который можно использовать как для iOS, так и для Android, что ускорит процесс разработки.

С уверенностью можно сказать, что данное мобильное приложение для автовокзала сможет улучшить опыт пассажиров, обеспечивая им удобный способ бронирования и покупки билетов, а также поможет автовокзалу адаптировать свои услуги под потребности клиентов и повысить качество обслуживания.

1. **Постановка задачи**

Задачей курсового проекта является разработка мобильного приложения на платформе Flutter, которое содержит четыре роли: гость, авторизованный пользователь, водитель и администратор.

Для роли «Гость» следует реализовать следующую функциональность:

* просмотр информации о рейсах;
* выполнение регистрации;
* выполнение авторизации.

Для роли «Авторизованный пользователь» следует реализовать следующую функциональность:

* просмотр информации о рейсах;
* управление бронью, в которое входит оформление брони и отказ от брони;
* управление личной информацией, в которое входит изменение и просмотр личной информации;
* возможность подачи заявки на становление водителем.

Для роли «Водитель» следует реализовать следующую функциональность:

* просмотр информации о рейсах;
* просмотр рейсов, к которым прикреплен водитель;
* отметка севших клиентов.

Для роли «Администратор» следует реализовать следующую функциональность:

* управление маршрутками;
* управление городами;
* управление рейсами;
* управление маршрутами;
* возможность ответа на заявку пользователя стать водителем;
* управление заказами.

Помимо этого, следует реализовать личный кабинет пользователя, в котором будет хранится личная информация.

Необходимо сделать вкладку «Рейсы», для пользователя, в которой будет возможность просмотра всех забронированных рейсов, а также возможность отказа от рейса на который была совершена бронь.

Также требуется реализовать возможность авторизации и регистрации неавторизованных пользователей.

Важно отметить, что приложение должно быть разработано с использованием Flutter, базирующегося на виджетах, и иметь адаптивный интерфейс и навигацию внутри приложения. Данные должны храниться в реляционной базе данных.

# **Обзор аналогичных решений**

В данном разделе будет рассмотрен список мобильных приложений автовокзалов, которые выполняют аналогичные пользовательские функции, необходимые для разработки в рамках данного курсового проекта. Приложения были протестированы на устройстве Huawei P Smart 2021, Android 10.

* 1. **Анализ приложения «Атлас»**

При открытии приложения, основное внимание уделяется форме поиска рейсов. Пользователь может заполнить эту форму, указав необходимые параметры, и после этого ему будут представлены соответствующие рейсы в виде списка. Дополнительно, на нижней части страницы расположено горизонтальное меню с разделами приложения. Главная страница приложения "Атлас" изображена на рисунке 2.1.

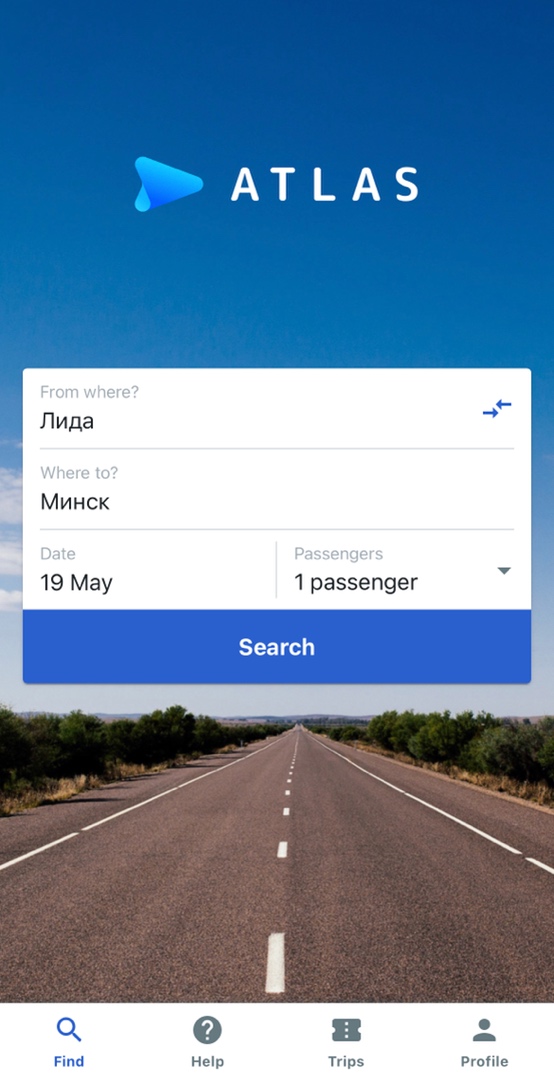
****

Рисунок 2.1 – Главная страница приложения

Нажав на кнопку «Войти», появится форма регистрации по номеру телефона. Можно также заметить надпись: «Запомнить меня на сайте». Это является очень необычным решением, так как данное приложение все-таки мобильное. Раздел регистрации представлен на рисунке 2.2.

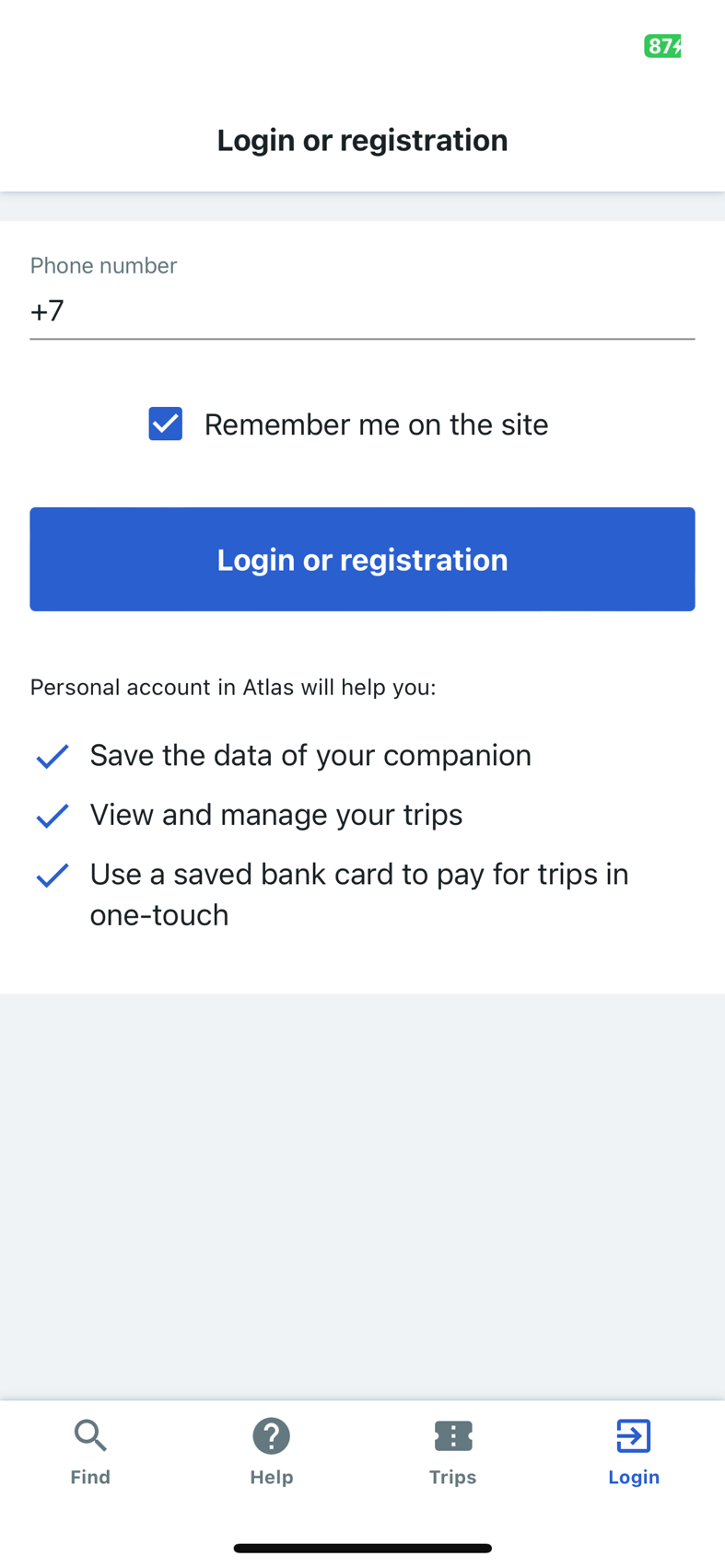


Рисунок 2.2 – Страница регистрации приложения

После нажатия на кнопку "Поиск" будет отображен список рейсов, включающий краткую информацию о каждом из них. Дополнительно, в приложении предусмотрена функция фильтрации и изменения дня, позволяющая пользователю выбрать нужный день путешествия с помощью элемента слайдера, представленного в виде дней месяца. Страница с перечнем рейсов изображена на рисунке 2.3.

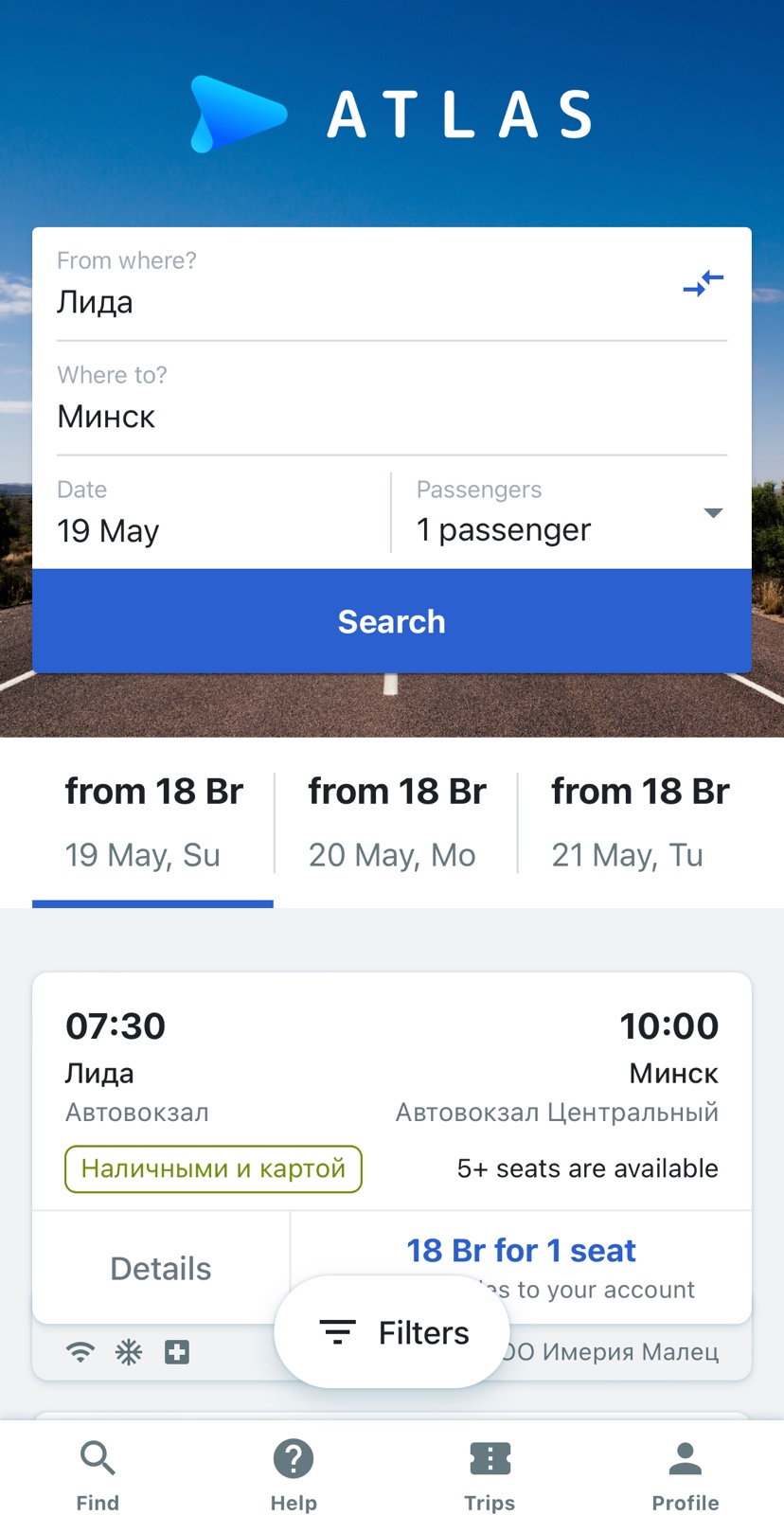


Рисунок 2.3 – Страница найденных рейсов

При выборе определенного рейса, у нас откроется подробная информация о рейсе, включающая пункт отправления, пункт назначения, транспортное средство, а также удобства, которые предоставляет данное транспортное средство. Также присутствует возможность выбора места посадки и места высадки с помощью карты. Но отсутствует возможность заказать маршрутку без авторизации, нам в любом случае потребуется зарегистрироваться. Страница с информацией о рейсе представлена на рисунке 2.4.

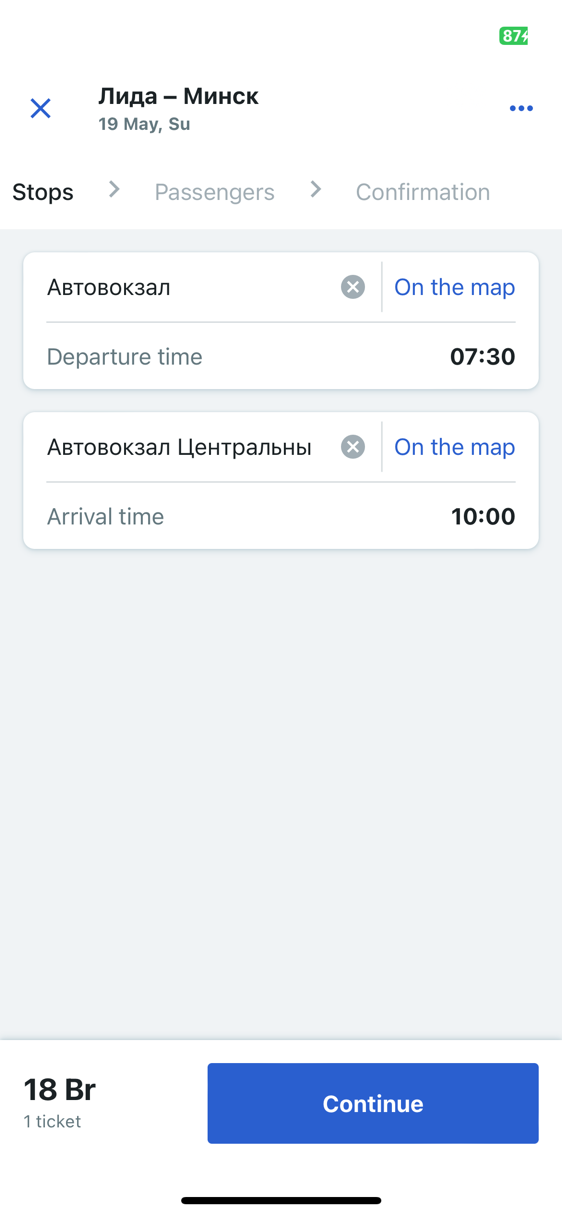


Рисунок 2.4 – Страница с данными о рейсе

Так как разрабатываемое приложение не ставит перед собой цель обеспечения оплаты билетов и взаимодействия с банковскими счетами, не будем рассматривать дальнейшие действия и закончим обзор данного аналога.

* 1. **Анализ приложения «Naminsk.by»**

При запуске приложения, нас встречает экран авторизации, представленный на рисунке 2.5, на котором есть поля для ввода номера и пароля, а также возможность входа без авторизации.



Рисунок 2.5 – Приветственный экран приложения «Naminsk.by»

После приветственного экрана, на главной странице приложения можно увидеть навигацию внизу экрана, раздел с поиском рейса, историей поездок, страницу для связи с диспетчерами, раздел с информацией по поводу посадок, а также раздел с профилем.

Главная страница приложения «Naminsk.by» представлена на рисунке 2.6.

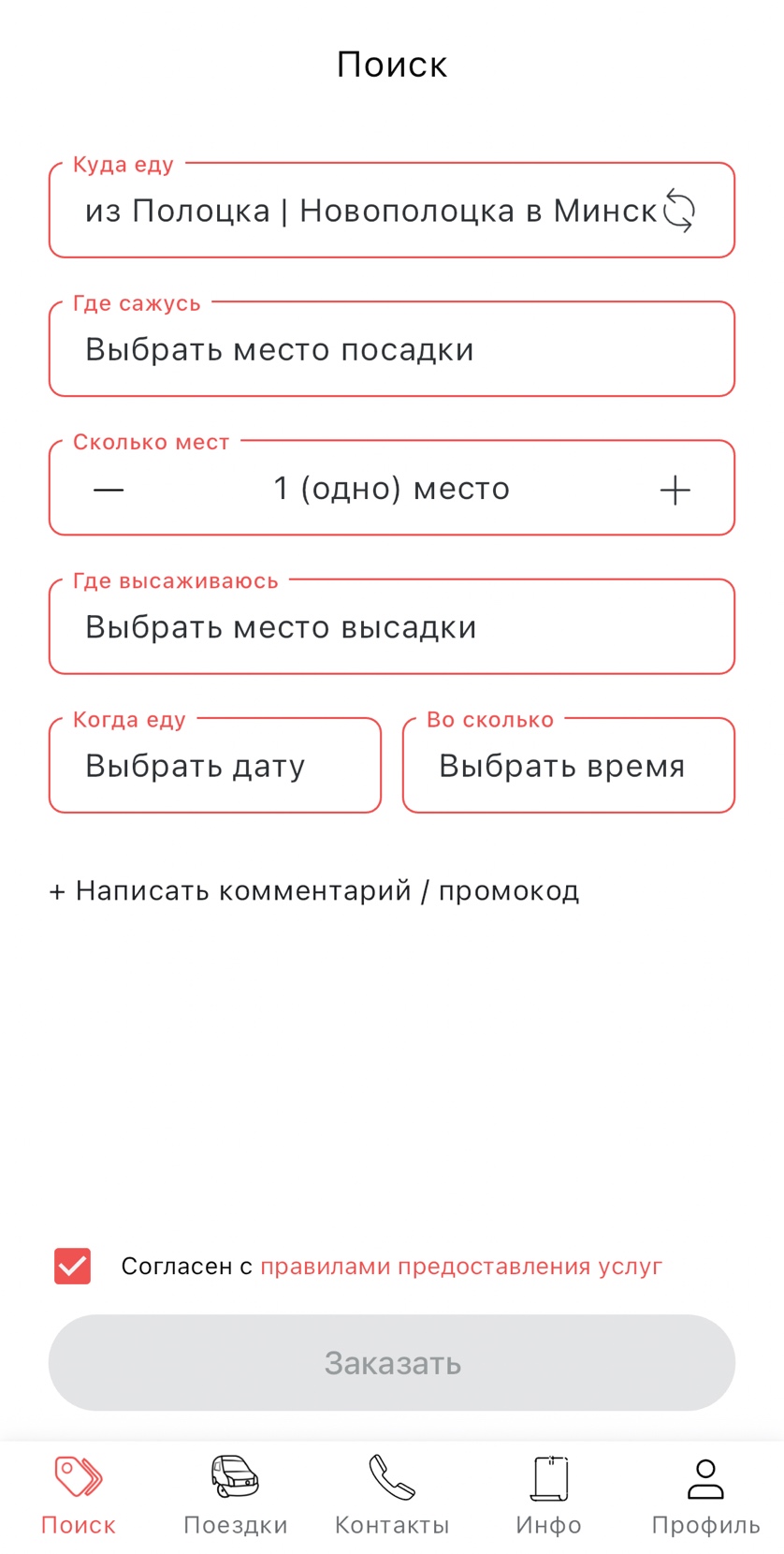


Рисунок 2.6 – Главная страница приложения

При заполнении всех полей, и последующим нажатии кнопки «Заказать», появится список подходящих под требования рейсов и краткой информации о каждом. Если все места уже будут заняты, то есть возможность нажатия на кнопку «Уведомить», после чего, при отказе иного пассажира, вам позвонит диспетчер. Также есть возможность нажатия на кнопку «Заказать», для заказа билета.

Страница с найденными рейсами представлена на рисунке 2.7.

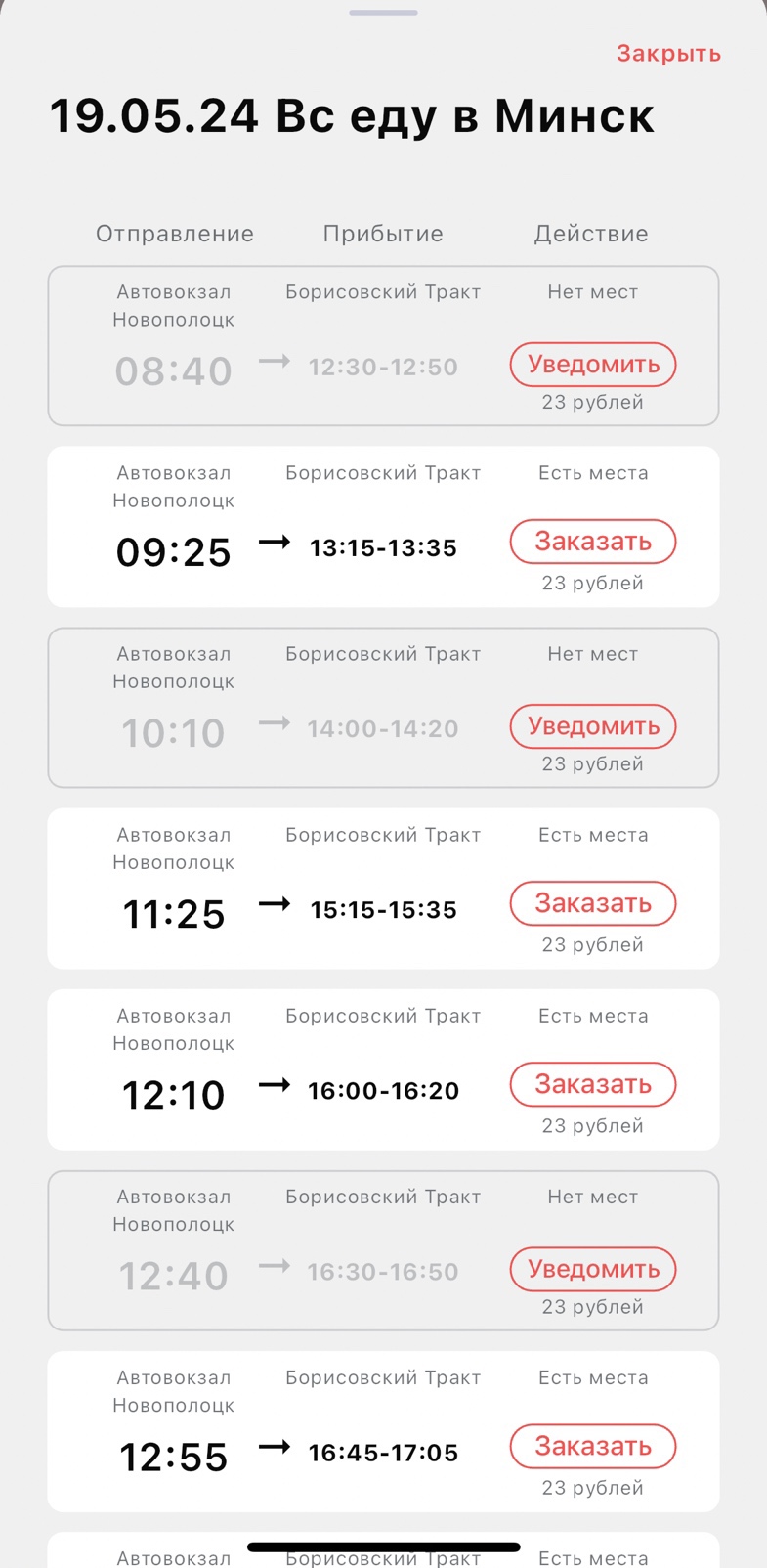


Рисунок 2.7 – Страница с рейсами

После выбора рейса, нам необходимо пройти регистрацию для оформления заказа.

На странице «Поездки», предоставляется информация о заказах пользователя, а также история заказов. Данная страница с информацией о заказах представлена на рисунке 2.8.

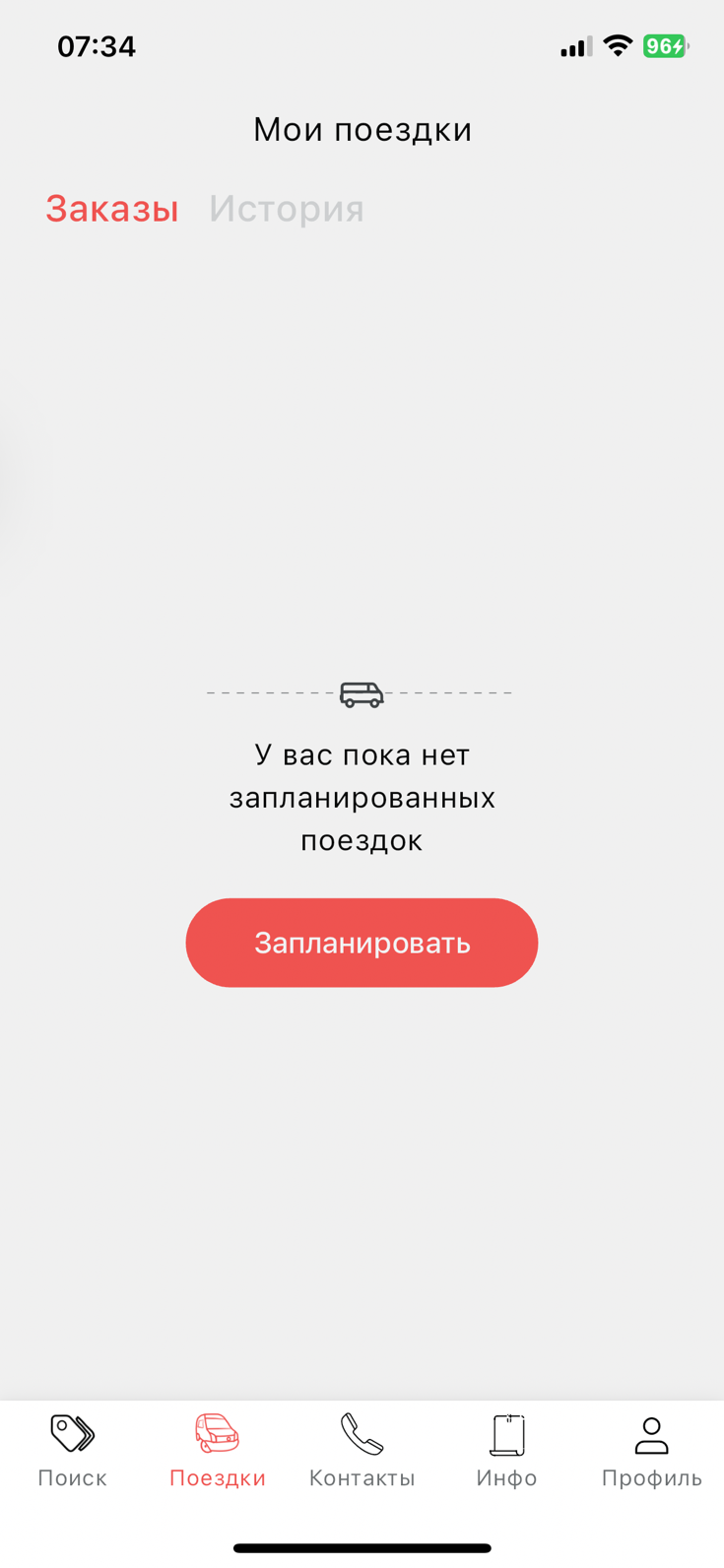


Рисунок 2.8 – Страница с информацией о заказах

Существует страница контакты со всеми возможными способами связи с администрацией, такие как, номера телефонов МТС и А1, сайты и социальные сети компании.

Страница с контактными данными представлена на рисунке 2.9.

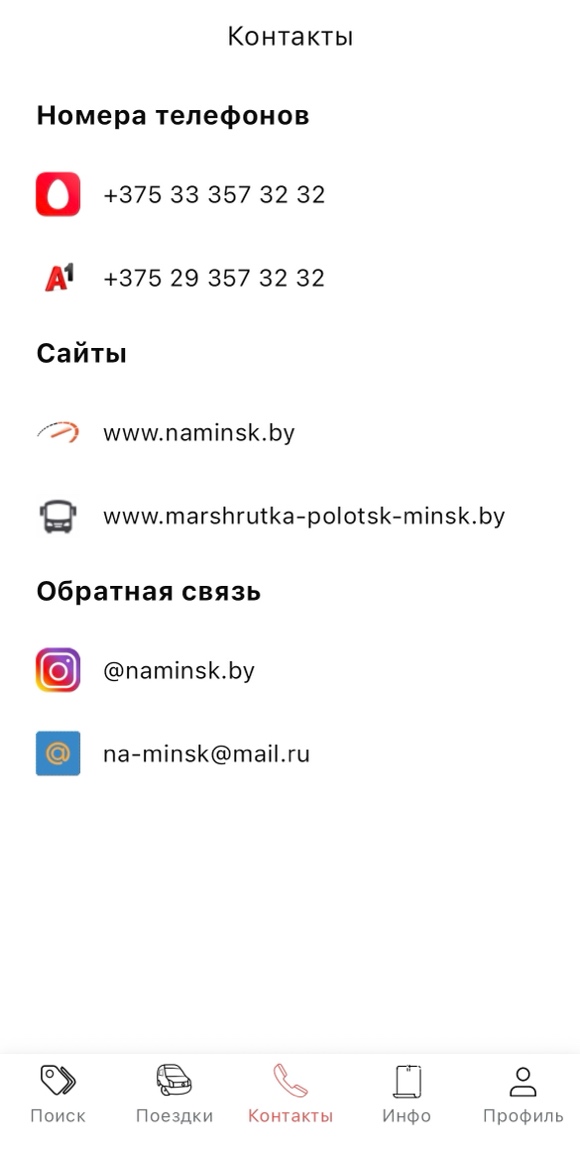


Рисунок 2.9 – Страница с контактными данными

* 1. **Вывод по разделу**

После анализа сравнительных примеров можно заключить, что они успешно включают все необходимые функции, которые требуется реализовать в конечном приложении. В этих примерах применяются навигационные элементы, механизмы авторизации и регистрации, а также функция поиска рейсов с заданными условиями.

В разрабатываемом приложении необходимо внедрить навигационные элементы и создать все необходимые формы, включающие регистрацию, авторизацию, поиск рейсов и бронирование билетов.

1. **Проектирования приложения**

Данный раздел посвящен основным аспектам проектирования планируемого приложения. В нем будет освещена структура приложения, выбор применяемых технологий и схема навигации внутри приложения. Эта часть поможет более глубоко понять принципы работы разрабатываемого приложения и определить необходимые задачи для успешной его реализации.

* 1. **Проектирование базы данных**

Для разработки мобильного приложения в рамках курсового проекта требуется база данных с необходимой конфигурацией сущностей для хранения всей необходимой информации.

Структуру базу данных, ограничения целостности, связи и поля можно увидеть на рисунке 3.1.

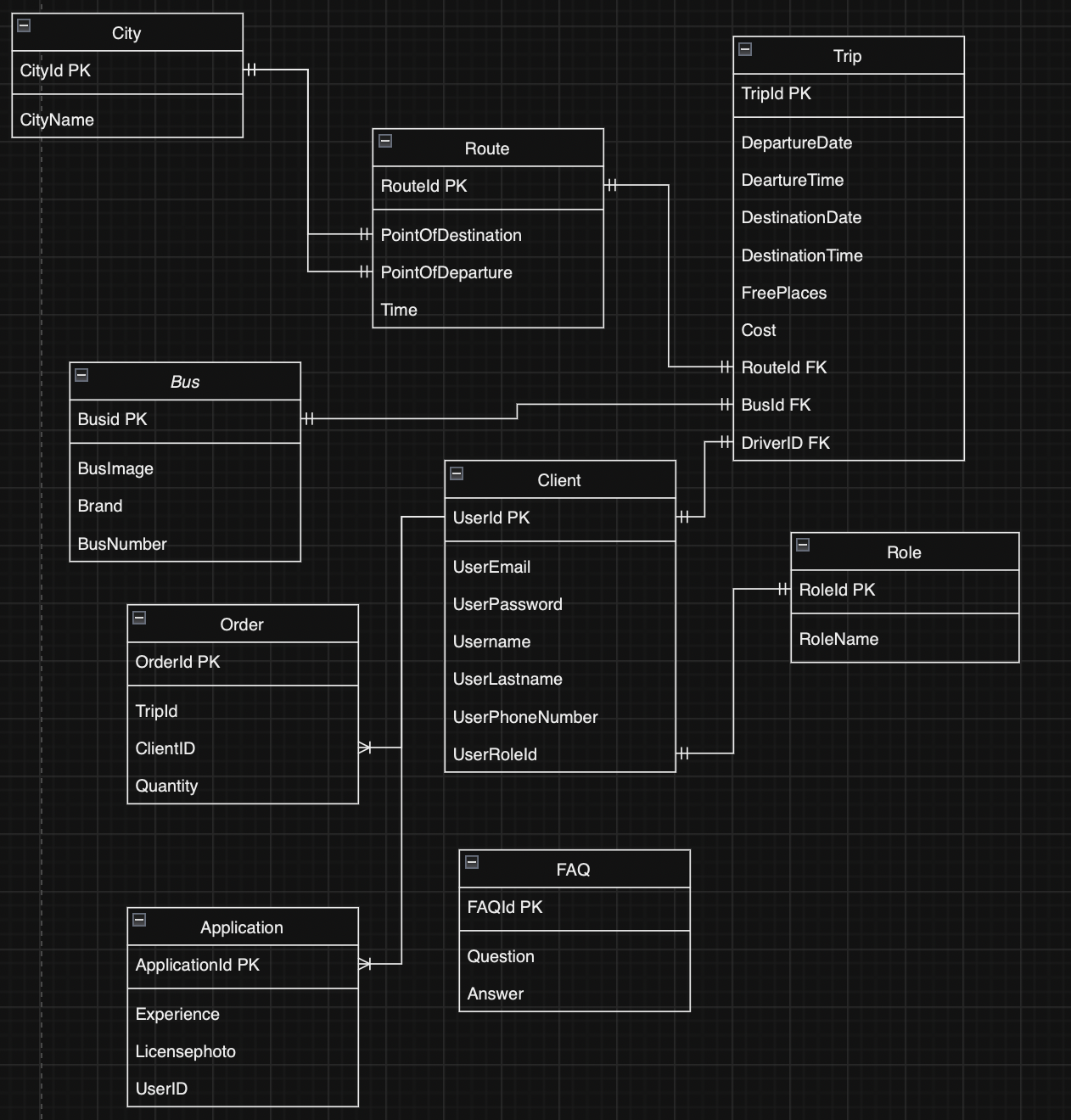


Рисунок 3.1 – Логическая схема базы данных

В качестве СУБД используется SQLite – самая используемая СУБД для мобильных устройств за счет своей простоты и легковесности, а также пакет sqflite для работы с SQLite через Flutter.

Опишем все таблицы, перейдя к физической реализации – типы данных, а также описание того, что будет храниться в конкретном столбце.

В таблице Client хранятся все пользователи, как зарегистрированные, так и не зарегистрированные, но заказавшие билет, за разделение пользователей отвечает поле USERSTATUS. Пароль пользователя хранится в зашифрованном виде. Физическая реализация данной таблицы представлена в таблице 3.1.

Таблица 3.1 – Содержание таблицы Client

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| USERID | INTEGER | Уникальный идентификатор пользователя |
| USEREMAIL | TEXT | Почта пользователя |
| USERPASSWORD | TEXT | Пароль пользователя |
| USERNAME | TEXT | Имя пользователя |
| USERLASTNAME | TEXT | Фамилия пользователя |
| USERPHONENUMBER | TEXT | Телефон пользователя |
| USERROLEID | INTEGER | Уникальный идентификатор роли |

В таблице Application хранятся все заявления пользователей на становление водителем маршрутки, а именно фото лицензий, опыт работы. Физическая реализация данной таблицы представлена в таблице 3.2.

Таблица 3.2 – Содержание таблицы Application

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| APPLICATIONID | INTEGER | Уникальный идентификатор заявления |
| EXPERIENCE | INTEGER | Опыт работы водителя |
| LICENSEPHOTO | BLOB | Фотографии лицензий |
| USERID | INTEGER | Уникальный идентификатор пользователя |

В таблице Order хранится информация о заказах определенного пользователя. Физическая реализация данной таблицы представлена в таблице 3.3.

Таблица 3.3 – Содержание таблицы Order

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ORDERID | INTEGER | Уникальный идентификатор заказа |
| TRIPID | INTEGER | Название рейса |
| CLIENTID | INTEGER | Уникальный идентификатор клиента |
| QUANTITY | INTEGER | Количество заказов |

В таблице Bus содержится информация о маршрутках: модель маршрутки, номер маршрутки и фотография маршрутки. Физическая реализация данной таблицы представлена в таблице 3.4.

Таблица 3.4 – Содержание таблицы Bus

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| BUSID | INTEGER | Уникальный идентификатор маршрутки |
| BRAND | TEXT | Модель маршрутки |
| BUSNUMBER | TEXT | Номер маршрутки |
| BUSIMAGE | BLOB | Количество мест |

В таблице Route хранится информация о маршрутах: город отправления, город прибытия и время пути. Физическая реализация данной таблицы представлена в таблице 3.5.

Таблица 3.5 – Содержание таблицы Route

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ROUTEID | INTEGER | Уникальный идентификатор маршрута |
| POINTOFDEPARTURE | INTEGER | Уникальный идентификатор города отправления |
| POINTOFDESTINATION | INTEGER | Уникальный идентификатор города прибытия |
| TIME | TEXT | Время пути |

В таблице Trip содержится информация о рейсах, которая содержит даты и время отправления и прибытия, количество свободных мест, цену, маршрут, маршрутку и водителя привязанных к данному маршруту. Физическая реализация данной таблицы представлена в таблице 3.6.

Таблица 3.6 – Содержание таблицы Trip

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| TRIPID | INTEGER | Уникальный идентификатор рейса |
| DEPARTUREDATE | TEXT | Дата отправления |
| DESTINATIONDATE | TEXT | Дата прибытия |
| DEPARTURETIME | TEXT | Время отправления |
| DESTINATIONTIME | TEXT | Время прибытия |
| FREEPLACES | INTEGER | Количество свободных мест |
| COST | REAL | Цена |
| ROUTEID | INTEGER | Уникальный идентификатор маршрута |
| BUSID | INTEGER | Уникальный идентификатор маршрутки |
| DRIVERID | INTEGER | Уникальный идентификатор водителя |

В таблице City содержится информация о городах. Записи содержат уникальный идентификатор города, а также название города. Физическая реализация данной таблицы представлена в таблице 3.7.

Таблица 3.7 – Содержание таблицы City

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| CITYNAMEID | INTEGER | Уникальный идентификатор города |
| CITYNAME | TEXT | Название города |

В таблице Role хранятся уникальный идентификатор роли, а также название роли пользователей приложения. Физическая реализация данной таблицы представлена в таблице 3.8.

Таблица 3.8 – Содержание таблицы Role

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ROLEID | INTEGER | Уникальный идентификатор роли |
| ROLENAME | TEXT | Название роли |

В таблице FAQ содержится информация о часто задаваемых вопросах. Записи содержат часто задаваемый вопрос и ответ на него. Физическая реализация данной таблицы представлена в таблице 3.9.

Таблица 3.9 – Содержание таблицы FAQ

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| FAQID | INTEGER | Уникальный идентификатор записи |
| QUESTION | TEXT | Название вопроса |
| ANSWER | TEXT | Название ответа |

Ниже приведен список используемых ограничений целостности в таблицах:

* на всех столбцах с идентификатором стоит autoincrement при создании;
* всем столбцам, не являющимися первичным ключом, назначено ограничение целостности not null;
* некоторым столбцам назначено ограничение целостности foreign key, которые привязываются к конкретным primary key для обеспечения связи между таблицами.

Данные таблицы обеспечивают необходимый функционал, который требуется от программного средства.

* 1. **Проектирование ролей**

Возможности каждой роли представлены на диаграмме использования в приложении А. Данная схема содержит три роли: авторизованный пользователь, водитель и администратор, а также соответствует задачам, установленным в разделе постановки задач.

* 1. **Проектирование навигации**

Приложение будет содержать некоторое количество страниц, которые должны быть как-то связаны между собой. Схема навигации выглядит так: на главной странице у нас будет элемент с навигационным меню – BottomNavigationBar, каждая его вкладка представляет собой отдельную страницу. С главной страницы мы можем получить доступ к четырем основным страницам приложения. Страница «Профиль» имеет условный рендер: если пользователь авторизован, то отображается страница с личной информацией пользователя, если не авторизован, то страница авторизации. Со страницы авторизации можно попасть на страницу регистрации. Страница «Главная» содержит форму для поиска рейсов. Путем взаимодействия с элементами данной страницы происходит процесс бронирования билета на рейс. Страница «Заказы» также имеет условный рендер: если пользователь авторизован, то отображаются его билеты, если нет – сообщение о необходимой авторизации. Страница «Помощь» содержит кнопку для вызова диспетчера, а также ответы на часто задаваемые вопросы.

* 1. **Описание структуры проекта**

Pubspec.yaml – это конфигурационный файл, используемый в проектах Flutter и Dart. Этот файл содержит метаданные о проекте, его зависимости, а также другую важную информацию, необходимую для управления и сборки проекта. Содержимое данного файла представлено на листинге 3.1.

Пакет path используется для расширенного управления путями приложения, sqflite позволяет работать с базой данных SQLite. С помощь provider можно предавать между страницами приложения наш единственный экземпляр класса, который будет обеспечивать взаимодействие с базой данных – DatabaseNotifier.

Пакет url\_launcher предоставляет возможность навигации данного приложения на другие приложения, например на телефон, при нажатии кнопки вызова диспетчера на странице «Помощь». Пакет image\_picker позволяет реализовать выбор картинок из галереи. Содержимое файла pubspec.yaml представлено на листинге 3.1.

|  |
| --- |
| dependencies:  flutter:  sdk:fluttercupertino\_icons:^1.0.2  sqflite:  path:  provider:  url\_launcher:^6.2.6  image\_picker: |

Листинг 3.1 – Содержимое файла pubspec.yaml

Организация папки lib проекта представлена на рисунке 3.2. Она разделена на разделы: Models, Services и Views. В директории Models хранятся классы, аналогичные таблицам базы данных. В директории Services хранятся классы для работы с базой данных, и авторизацией пользователя. В директории views находится директория widgets, а также страницы приложения. В папке widgets находятся виджеты приложения.

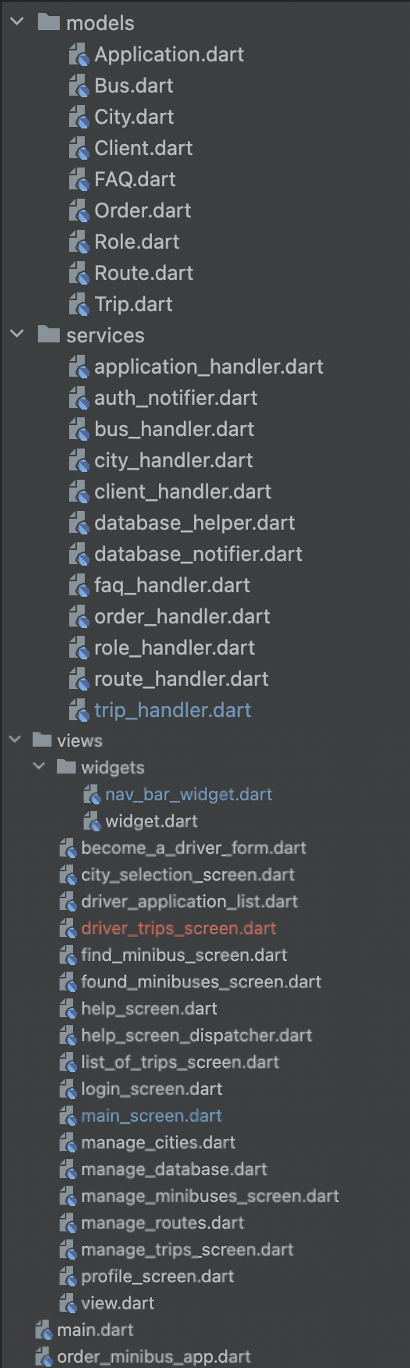


Рисунок 3.2 – Структура папки lib проекта

Файл main.dart определяет процесс инициализации приложения, а DatabaseNotifier организует взаимодействие с базой данных, создавая экземпляр DatabaseHelper в конструкторе, который в свою очередь в методе init вызывает остальные конструкторы классов для работы с базой данных.

* 1. **Вывод по разделу**

В данном разделе мы тщательно изучили ключевые элементы нашего проекта. Особое внимание уделялось проектированию базы данных приложения, включая его физическую и логическую структуру, которая обеспечит все необходимые функции приложения. Мы также определили роли в контексте приложения, выделили различные уровни доступа и функциональность для каждой роли, и создали диаграмму использования для наглядного представления.

Мы разработали модель навигации приложения, используя схему, которая обеспечивает интуитивно понятный и эффективный пользовательский интерфейс. Это позволит пользователям легко перемещаться по приложению и быстро находить нужные функции.

Важной частью нашего проектирования была организация структуры проекта. Мы определили все необходимые зависимости проекта, обосновав их использование, и создали файловую структуру проекта, чтобы обеспечить четкость и эффективность кода.

В целом, данный раздел предоставляет нам прочную основу для дальнейшего развития и реализации проекта. Этот тщательно продуманный и спроектированный подход позволит нам успешно продвигаться вперед и достичь наших целей.

В этом разделе мы также уделили внимание анализу требований и функциональности приложения. Мы провели детальное исследование, чтобы понять, какие задачи должны быть выполнены для успешной реализации проекта. Это включало в себя определение основных функций, которые приложение должно предоставлять, а также выявление специальных требований или ограничений, которые нужно учесть при разработке.

Важным аспектом было также рассмотрение масштабируемости и производительности приложения. Мы анализировали ожидаемую нагрузку, определяли возможные узкие места и предлагали решения для обеспечения высокой производительности и отзывчивости системы при увеличении числа пользователей.

В конечном итоге, этот раздел предоставляет нам полное представление о проекте, его функциональности, требованиях, безопасности и производительности. Это позволяет нам строить дальнейшую разработку и реализацию на основе основательного анализа и планирования.

1. **Реализация приложения**

Данный раздел содержит в себе описание шагов разработки приложения и его элементов. Разработка приложения будет выполнена в IDE Android Studio, версия Flutter SDK – 3.19.5. Первоначальная реализация будет протестирована на Android-устройстве. Также приложение будет запускаться на Macos, iOS.

* 1. **База данных и взаимодействие с ней**

Для работы с базой данных разработан класс DatabaseNotifier, который в конструкторе создает экземпляр класса DatabaseHelper. Рассмотрим содержимое DatabaseNotifier– Листинг 4.1. Полный скрипт работы с базой данных представлен в приложении Б.

|  |
| --- |
| class DatabaseNotifier extends ChangeNotifier {  late DatabaseHelper \_databaseHelper;  DatabaseNotifier() {  \_databaseHelper = DatabaseHelper();  }  Future<void> initializeDatabase() async {  await \_databaseHelper.init("database.db");  notifyListeners();  }  DatabaseHelper get databaseHelper => \_databaseHelper;  } |

Листинг 4.1 – Содержимое класса DatabaseNotifier

В данном классе мы определяем экземпляр DatabaseHelper, который будет единственным в приложении. DatabaseNotifier. Класс также расширяет ChangeNotifier для уведомления слушателей о завершении инициализации. Класс DatabaseHelper хранит поле db и метод init. В методе init происходит инициализация поля db, а также создание таблиц и триггеров, если они еще не существуют путем вызова методов классов, отвечающих за работу с конкретными таблицами.

Экземпляр класса DatabaseNotifier используется как глобальное состояние, чтобы передавать один и тот же экземпляр между разными страницами. Для этого используется пакет Provider. Он позволяет получать экземпляр класса DatabaseNotifier из контекста виджета, предварительно поместив его в контекст приложения следующим способом. На листинге 4.2 представлен метод main, в котором происходит создание провайдера.

|  |
| --- |
| void main() {  runApp(  MultiProvider(  providers: [  ChangeNotifierProvider(create: (context) => AuthNotifier()),  ChangeNotifierProvider(create: (context) => DatabaseNotifier()),  ],  child: OrderingMinibusesApp(),  ),  ); } |

Листинг 4.2 – Настройка и использование класса Provider

Далее мы сможем выполнить любой метод, содержащийся в классе DatabaseHelper исходя из наших нужд.

Также созданы классы, которые будут представлять модель нашей базы данных. Их названия и наличие можно увидеть на рисунке 3.2 в директории Database. Более полное содержимое классов можно найти в приложении Б.

* 1. **Авторизация, регистрация, личный кабинет пользователя**

Авторизация, регистрация и личный кабинет создаются для разделения ролей, поэтому и должны быть связаны между собой, т.е. должны располагаться в одном месте. В данном случае мы будем иметь дело с условным рендерингом – если пользователь не авторизован – отображать ему страницу входа, а если авторизован – личный кабинет. Также следует учитывать, что личный кабинет для администратора, авторизованного пользователя и водителя будут отличаться

Реализована страница ProfileScreen, на которой пользователя приветствует окно для регистрации, пользователь должен ввести поля username, userLastname, password, confirmPassword, email, phoneNumber, после чего нажать на кнопку «Зарегистрироваться». После данных действий информация о пользователе сохраняется в базу данных после чего эта информация используется для отображения интерфейса. Чтобы разделить экраны для пользователя, диспетчера и водителя был добавлен класс AuthNotifier который хранит состояние авторизованного пользователя. Для личного кабинета каждой из ролей пользователя разработан уникальный виджет. Также на странице есть кнопка для входа в аккаунт, которая переносит пользователя на форму для захода в аккаунт, после чего переменная класса AuthNotifier \_currentUser меняется на экземпляр объекта текущего пользователя.

Для неавторизованного пользователя(Гостя), также реализована смена интерфеса. Он может просматривать рейсы, смотреть часто задаваемые вопросы, а также проходить этап регистрации. Для неавторизованного пользователя предусмотрено поле \_isAuthenticated, в классе AuthNotifier, которое хранит в себе состояние пользователя.

* 1. **Панель администратора**

В соответствии от того, выполнен ли вход под пользователем с ролью обычного пользователя, водителя или администратора будет отрисована уникальная страница для каждой роли. Для пользователя с ролью «Администратор» появится возможность просматривать всех пользователей приложения на экране «Profile». Внешний вид панели администратора представлен на рисунке 4.1.

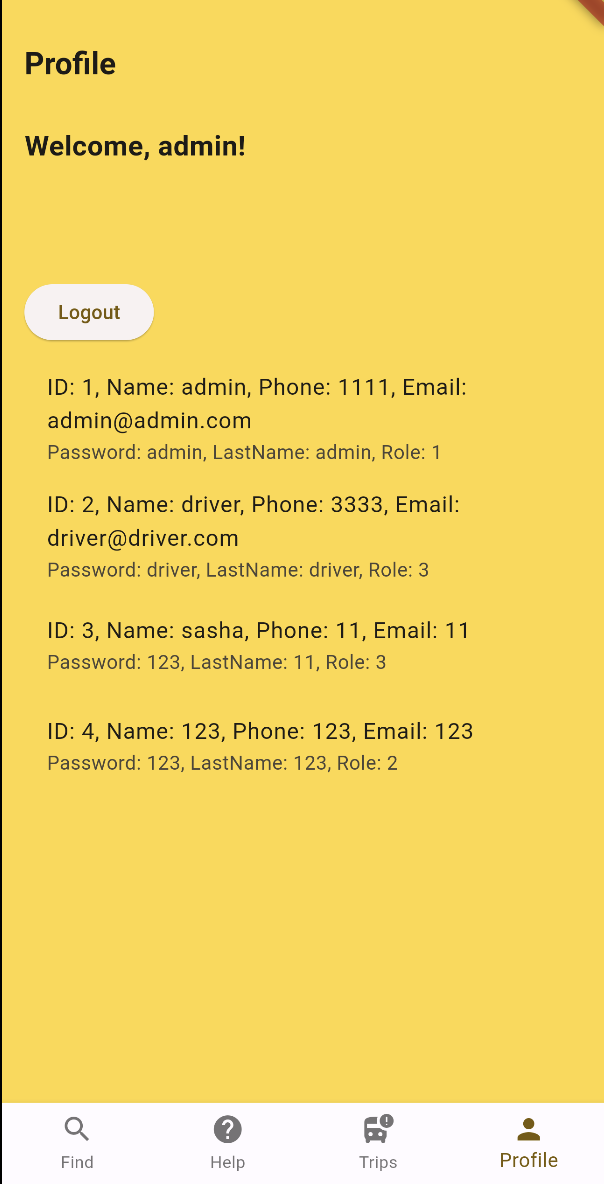


Рисунок 4.1 – Панель администратора

Следующее что доступно администратору это создание записей в базе данных. Администратор может создавать записи в таблицах BUS, CITY, ROUTE, TRIP, FAQ.

Внешний вид панели администратора представлен на рисунке 4.2.

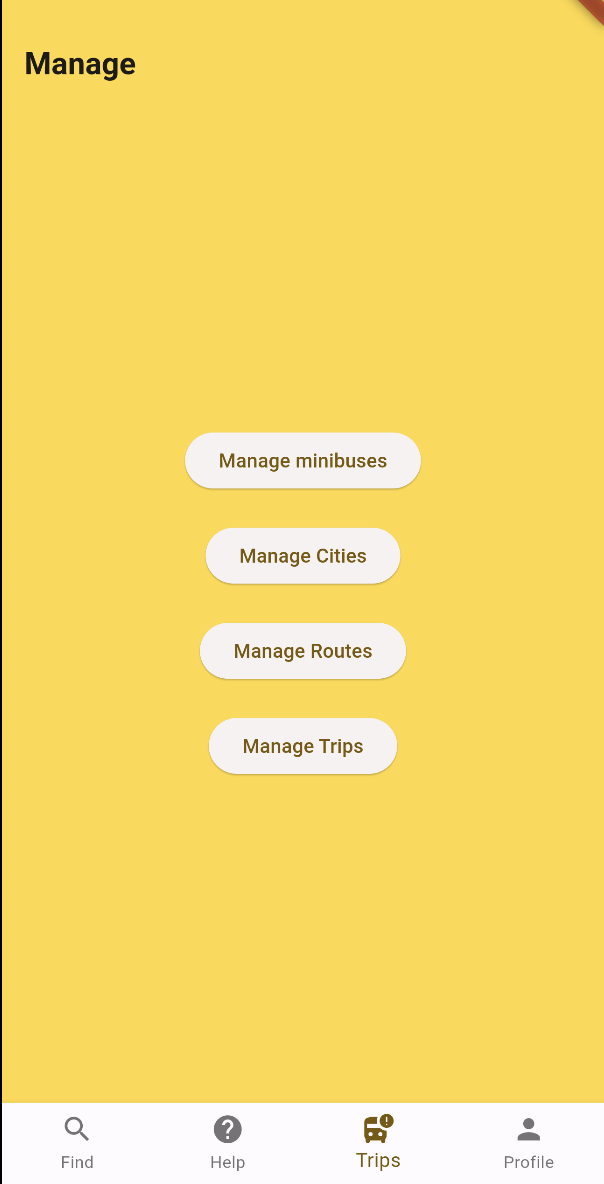


Рисунок 4.2 – Панель администратора

* 1. **Процесс поиска и бронирования билета**

В процессе поиска рейса будет задействовано две страницы нашего приложении: FindMinibusScreen и FoundMinibusesScreen. На странице FindMinibusScreen пользователь должен заполнить поля «Откуда», «Куда», «Дата», «Количество пассажиров». В поиске города на полях «Откуда» и «Куда» поможет CitySelectionScreen. На рисунке 4.3 представлен виджет с поиском.

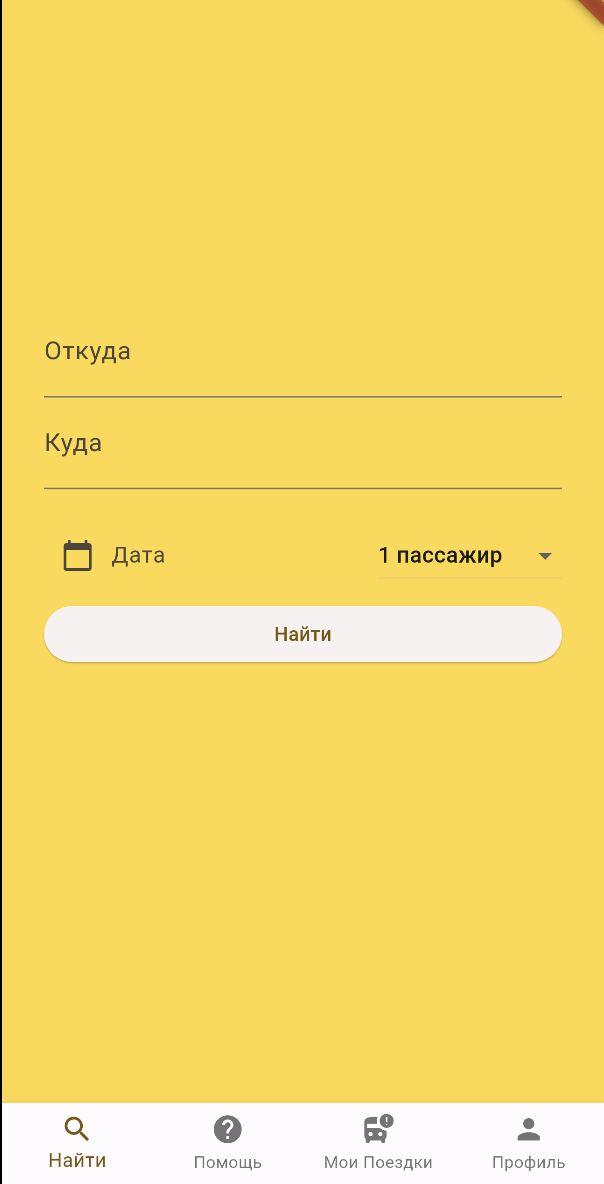


Рисунок 4.3 – Виджет с формой поиска

По нажатию на кнопку «Найти» выполняется поиск рейса путем запроса к базе данных, включающим связанные таблицы Trip, Route и City, а также будет отображена страница FoundMinibusesScreen с результатами поиска. В виджете будут отображены результаты поиска рейсов на указанный день. Построение будет происходить динамически – на основании результата поиска рейсов с необходимыми условиями в виджете ListView, каждый элемент которого включает данные, относящиеся к определенному рейсу. На рисунке 4.4 представлен виджет с результатом поиска.

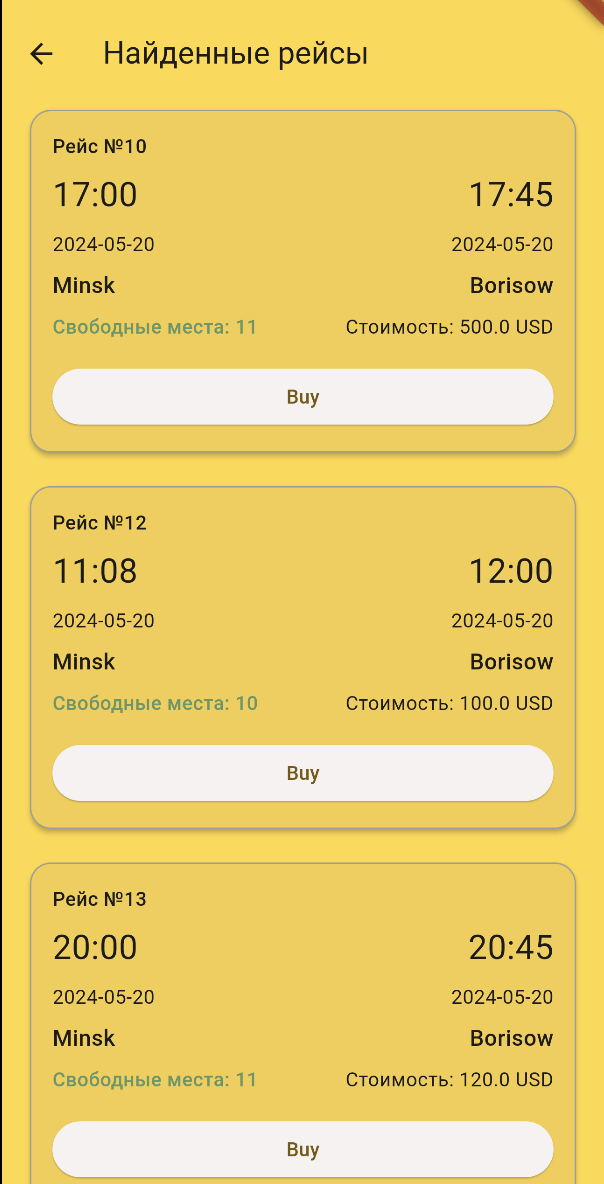


Рисунок 4.3 – Виджет с результатом поиска

При нажатии на кнопку «Заказать» появляется окно с подтверждением заказа. Если пользователь успешно авторизовался то заказ переносится во вкладку «Мои поездки», а также количество свободных мест на этот рейс снизиться на количество указанных пассажиров.

* 1. **Страница заказов пользователя**

Данная страница также имеет условный рендеринг, в зависимости от того, авторизован пользователь или нет. Если пользователь не авторизован, то появится надпись, что ресурс доступен только авторизованным пользователям. Если пользователь авторизован, то на данной странице будет отображаться список заказанных билетов этого пользователя. Страница заказов для авторизованных и неавторизованных пользователей представлена на рисунке 4.5.

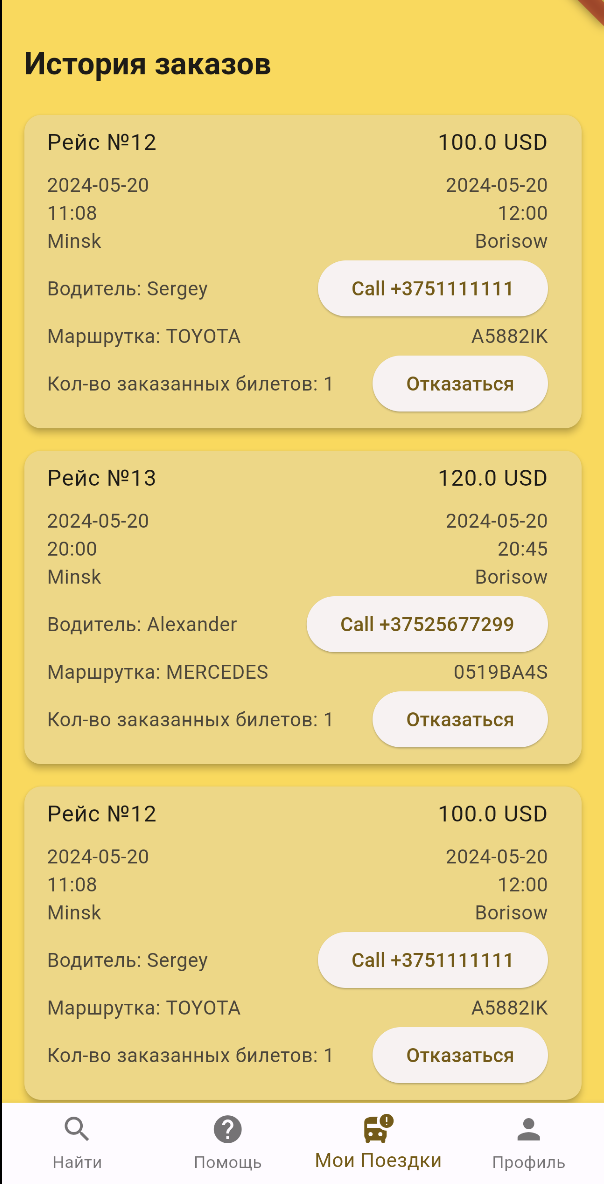


Рисунок 4.5 – Страница заказов

При нажатии на любой из билетов появится страница с полной информацией о рейсе. Авторизованный пользователь может отменить активную бронь, нажав на кнопку «Отказаться» и подтвердив действие в диалоговом окне.

* 1. **Вывод по разделу**

В данном разделе была рассмотрена реализация главных функций приложения – регистрация, авторизация, бронирование билетов, управление бронированием, панель администратора, личный кабинет пользователя, работа с API, а также технологии и подходы, которые обеспечили их реализацию.

1. **Тестирование приложения**

После создания приложения требуется проверить, что все работает как и было задумано. В данном разделе рассматриваются шаги, которые выполнит авторизованный пользователь, гость, водитель и администратор при скачивании приложения.

* 1. **Гость**

При запуске приложения перед гостем появится главная страница приложения. С помощью нижнего навигационного меню гость может перемещаться по странице. Проверка представлена на рисунке 5.1.

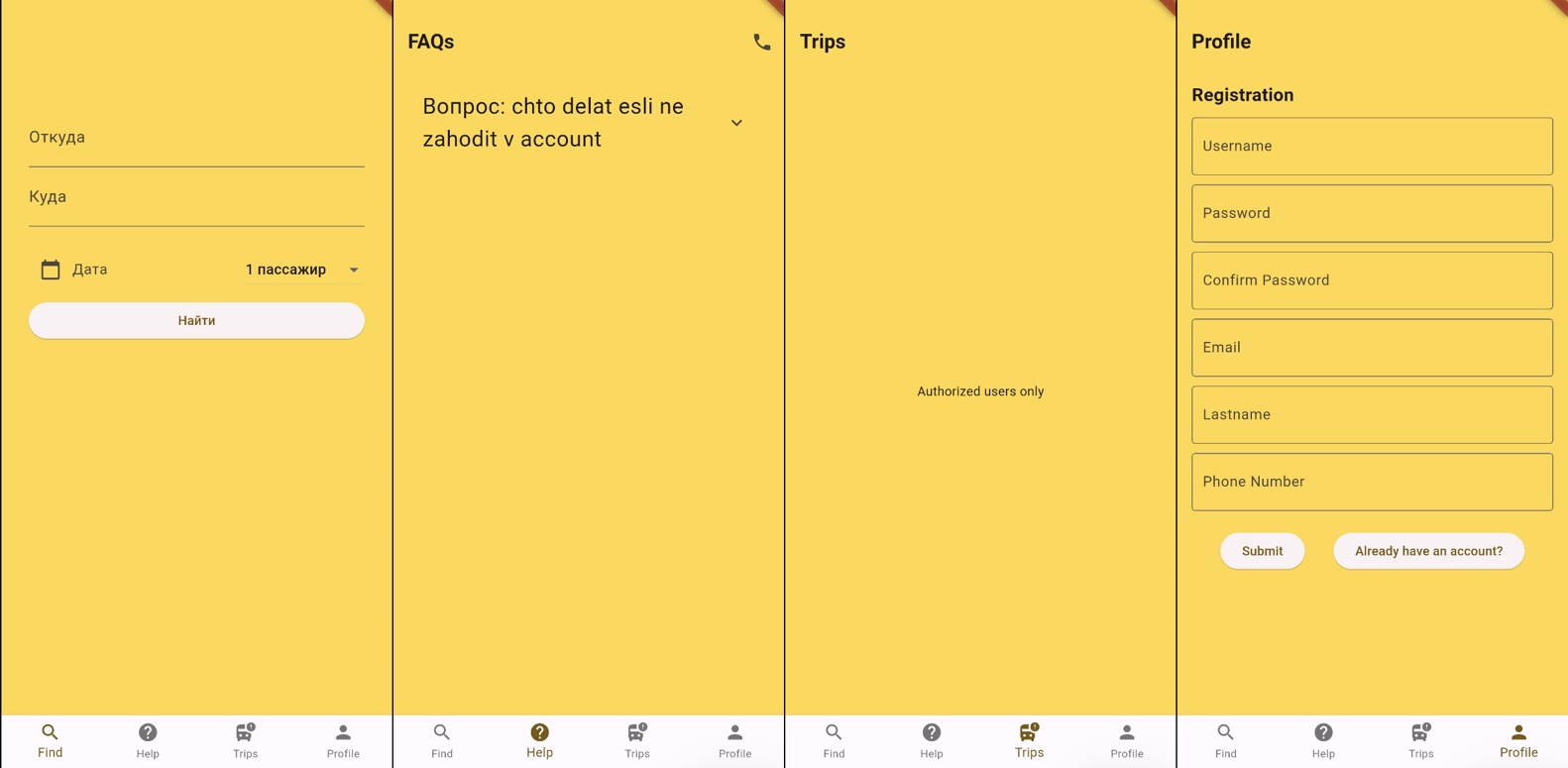


Рисунок 5.1 – Проверка возможностей Гостя

Как мы видим, Гость имеет доступ к некоторым элементам интерфейса, однако для полноценного функционирования приложения Гостю нужно зарегистрироваться.

Далее пользователю предстоит процесс регистрации. Гость вводит данные и нажимает на кнопку Зарегистрироваться, после чего, если они валидные, в базе данных появляется новая запись с информацией введенной пользователем информации.

Если Гость зарегистрировался он также может войти в свой аккаунт нажатием на кнопку с надписью «Уже зарегистрирован?»

Если пользователь ввел не все значения из формы, то он получит ошибки в соответствующих полях. Процесс регистрации и входа в аккаунт представлен на рисунке 5.2.

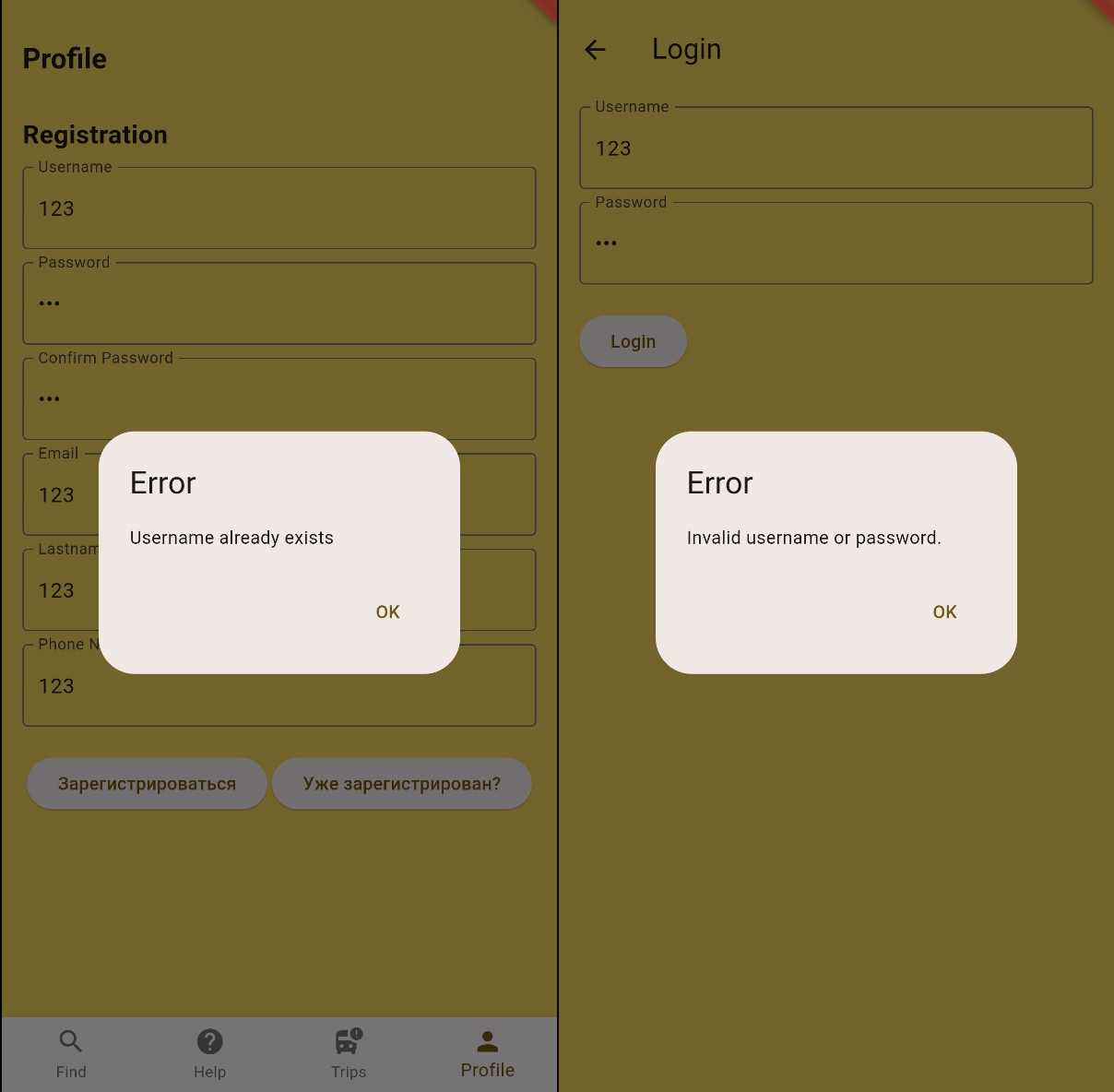


Рисунок 5.2 – Процесс регистрации и авторизации

После этого пользователь имеет учетную запись и может полноценно взаимодействовать с бронированием билетов: бронировать и отменять бронь билетов.

* 1. **Авторизация и просмотр заказов**

На странице входа пользователь выполняет авторизацию, причем, если не существует пользователя с таким номером телефона и паролем, то выводится ошибка. После ввода нужных данных учетной записи пользователь попадает на страницу профиля. На данной странице пользователь может подать заявление стать водителем.

Создание заявления на становление водителем, а также внешний вид страницы Профиль представлен на рисунке 5.3.

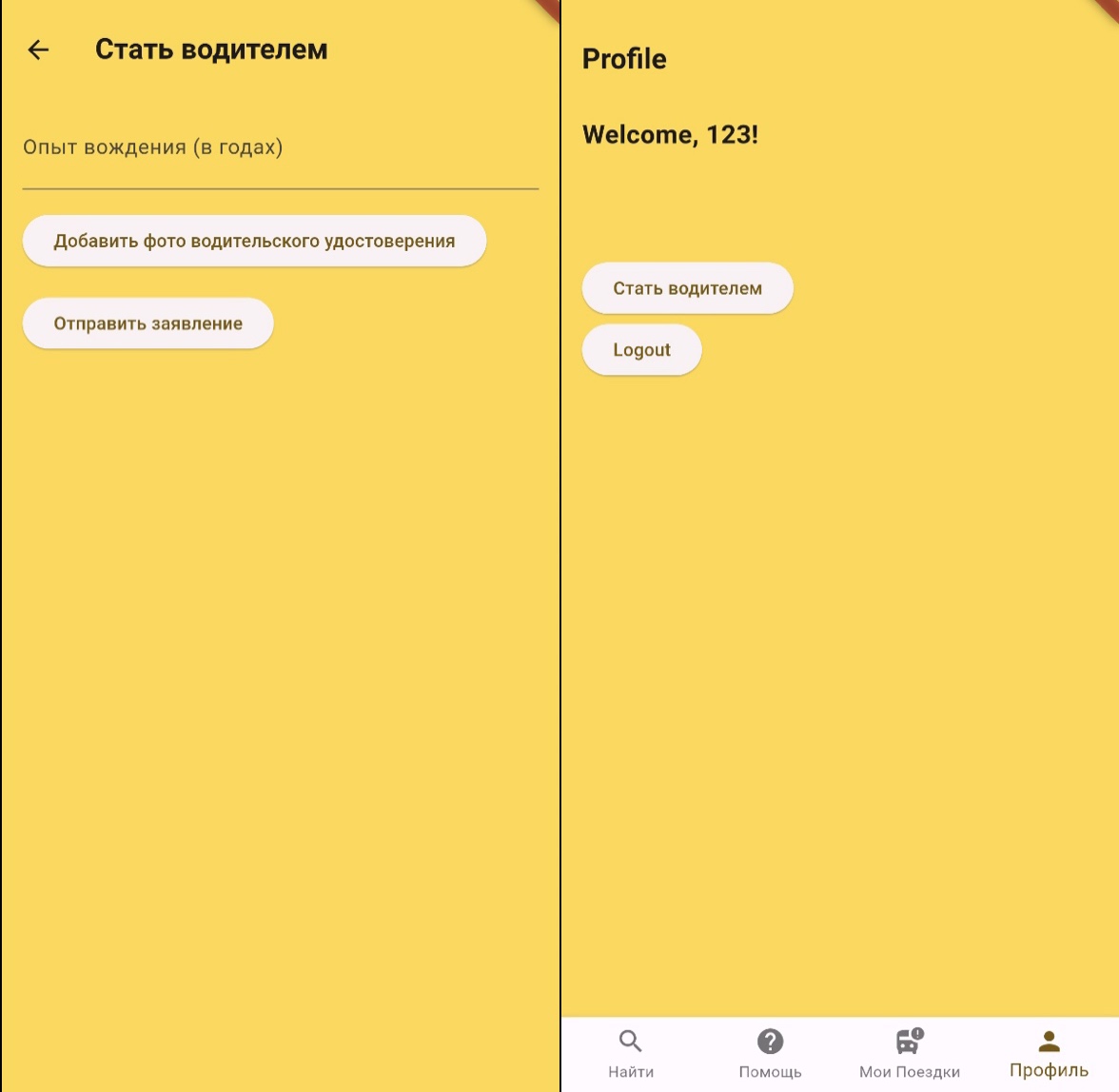


Рисунок 5.3 – Внешний вид страницы

При переходе на вкладку «Мои поездки» авторизованного пользователя отображаются билеты пользователя. Для авторизованного пользователя нечего тестировать, а вот для неавторизованного можно протестировать будут ли отображаться билеты авторизованных пользователей. Допустим у нас есть билет с идентификатором шесть у первого пользователя и с идентификатором семь у второго пользователя, билет с идентификатором шесть не должен отображаться у пользователя с идентификатором семь. Вкладка «Мои поездки» представлена на рисунке 5.4.

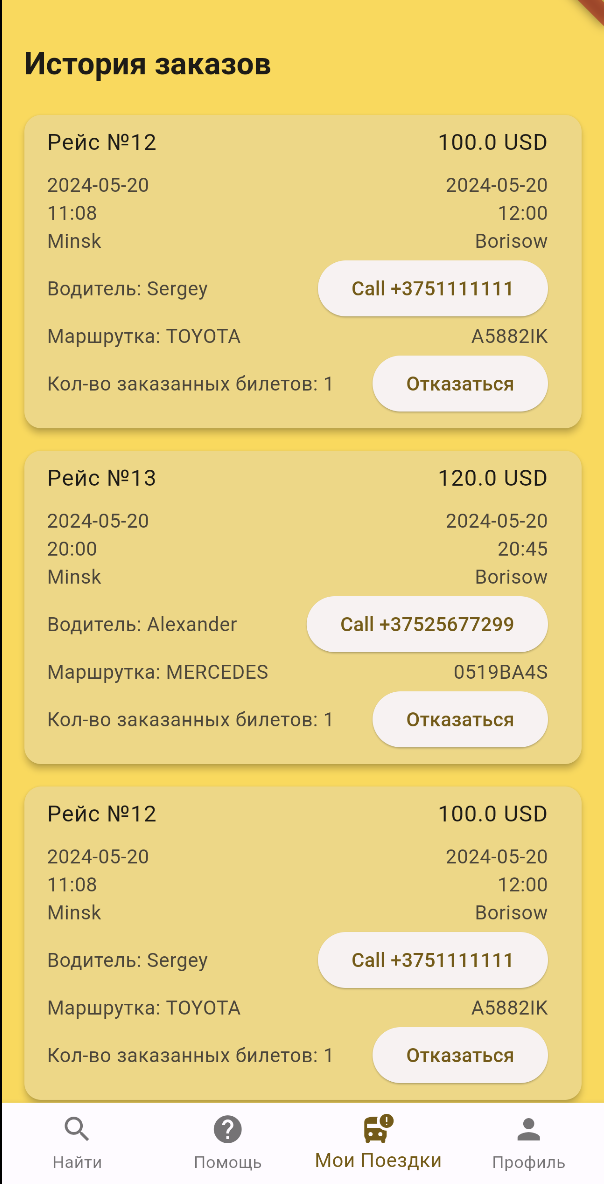


Рисунок 5.4 – Страница «Мои поездки»

Исходя из результатов можно сказать, что все работает так, как и должно.

* 1. **Использование панели водителя**

Для водителя реализован интерфейс просмотра всех пользователей определенного рейса, а также возможность набора номера этого пользователя для получения информации о текущем статусе пользователя, а также для подтверждения его заказа. Результат представлен ан рисунке 5.5.

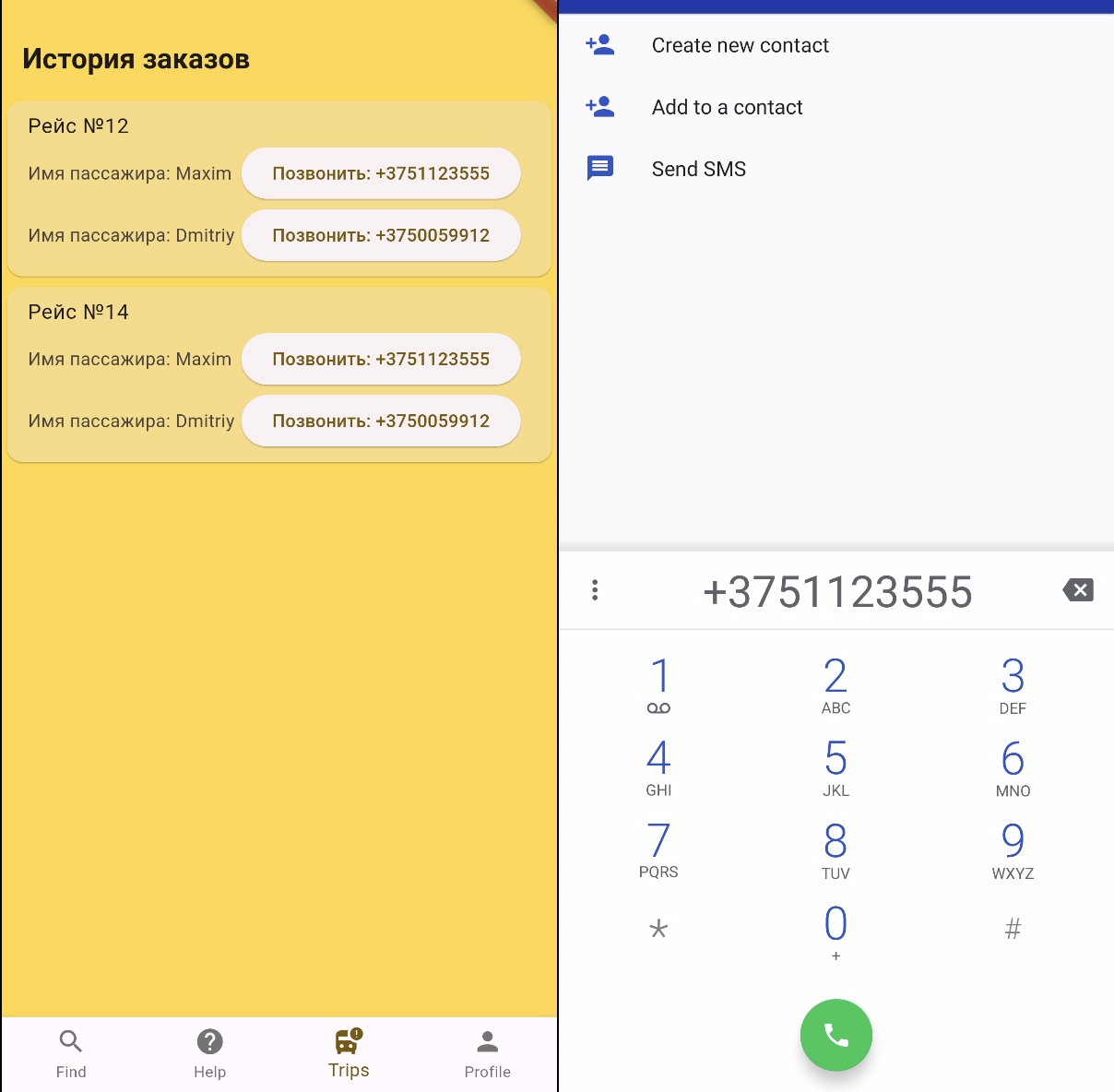


Рисунок 5.5 – Результат проверок панели водителя

Исходя из результатов, можно сделать вывод, что все работает корректно.

* 1. **Использование панели администратора**

Панель администратора открывается при вводе данных администратора в поля ввода логина и пароля для неавторизованных пользователей. Панель включает разделы для управления каждой моделью нашей базы данных.

В панели администратора соблюдены все функциональные требования так, чтобы не нарушать связь между таблицами базы данных.. При попытке добавления данных, которые уже существуют будет выводится сообщение об ошибке для уведомления пользователя. Результат проверок на добавление уже существующего объекта представлен на рисунке 5.6.

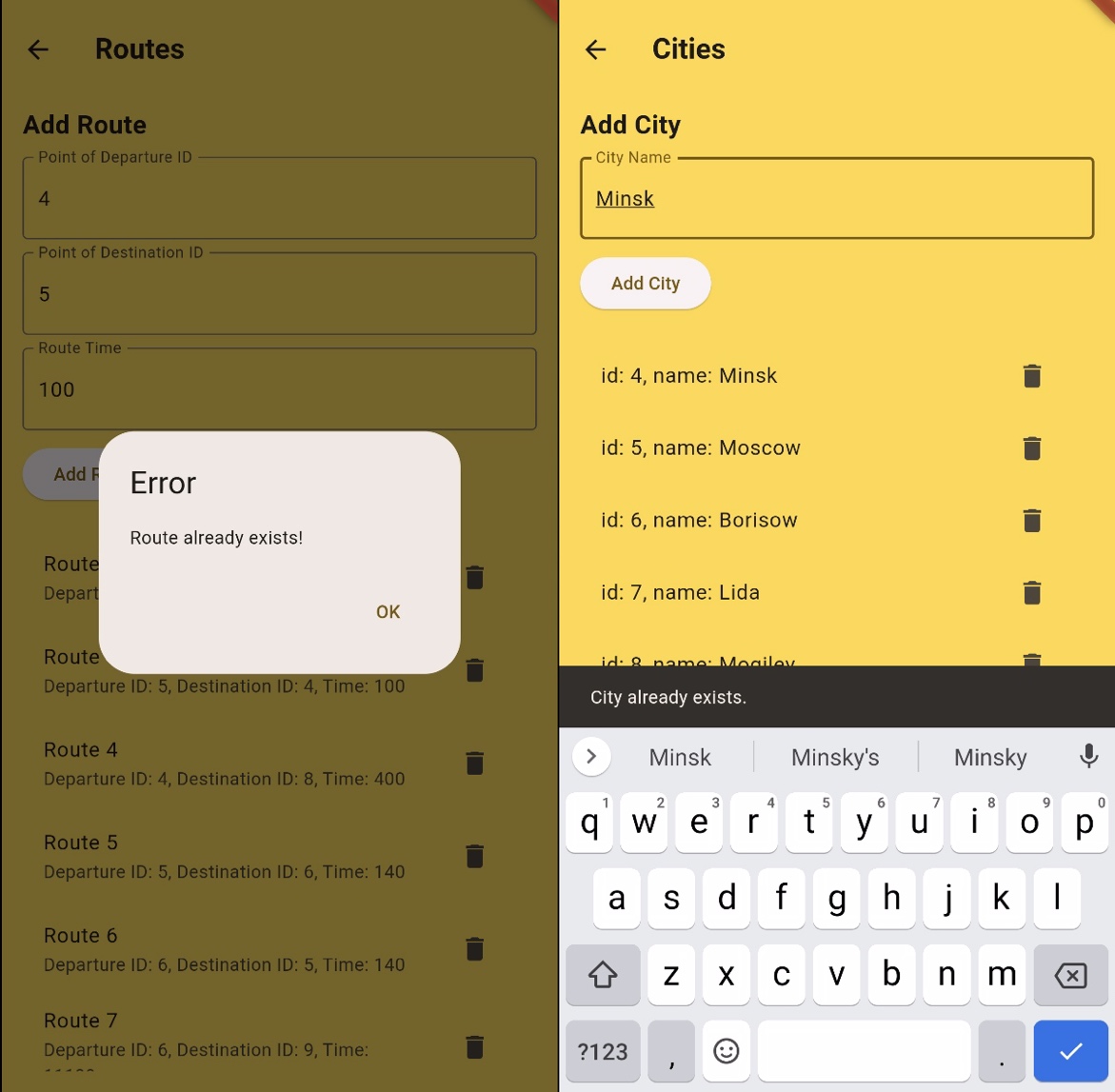


Рисунок 5.6 – Результат проверок добавления существующего

Исходя из результата можно заметить, что все работает как требуется. В результатах нет проверки добавления уже существующего рейса, т.к. добавлена проверка на занятость водителя, поэтому рейсы не могут быть точь-в-точь одинаковыми.

Исходя из результатов все проверок можно сделать вывод, что функциональность панели администратора работает полностью исправно.

* 1. **Вывод по разделу**

В данном разделе было описано тестирование работоспособности основных функций приложения: поиск рейса с условием, регистрация, авторизация, бронирование билета и управление бронированием, взаимодействие с данными через панель администратора, а также панель водителя.

1. **Краткое описание приложения для демонстрации**

Приложение автовокзала предлагает полный набор функций, о которых было заявлено: авторизацию и регистрацию пользователей, поиск билетов с определенными условиями, возможность бронирования билетов, а также управление бронированиями (просмотр и отмена) и управление данными в базе данных через административную панель.

При запуске приложения пользователь сразу попадает на домашнюю страницу, где находится форма для поиска билетов. Нижняя панель навигации позволяет пользователям перемещаться между страницами приложения: страница поиска и заказа билетов, страница просмотра заказанных билетов для авторизованных пользователей, страница поиска билетов для неавторизованных пользователей, страница входа (если пользователь не вошел в аккаунт) и страница профиля (если пользователь вошел в аккаунт). Если пользователь не зарегистрирован, на странице входа есть ссылка на страницу регистрации, где пользователь может заполнить форму для создания учетной записи в приложении.

Процесс бронирования билетов включает поиск нужного билета, выбор интересующего рейса и ввод персональных данных. Все формы имеют валидацию для каждого поля и обеспечивают автоматическую навигацию.

Панель водителя доступна только пользователям с ролью "водитель". Для получения доступа к ней пользователь должен нажать на кнопку "Панель водителя" в своем профиле. На этой странице отображается список рейсов для данного водителя, и после выбора рейса открывается информация о рейсе с указанием клиентов, связанных с этим рейсом.

Панель администратора доступна только пользователям с ролью "администратор". Для доступа к ней пользователь должен нажать на соответствующую кнопку в своем профиле. На этой странице представлены несколько вкладок, где можно выполнять операции управления данными в каждой из таблиц базы данных.

Приложение предоставляется в виде файла .apk для установки на устройства Android.

1. **Руководство пользователя**

В данном разделе описаны основные шаги по взаимодействию с созданным приложением.

Для начала работы требуется установить приложение при помощи .apk-файла и открыть его. При открытии приложения отобразится главная страница, которая представлена на рисунке 7.1

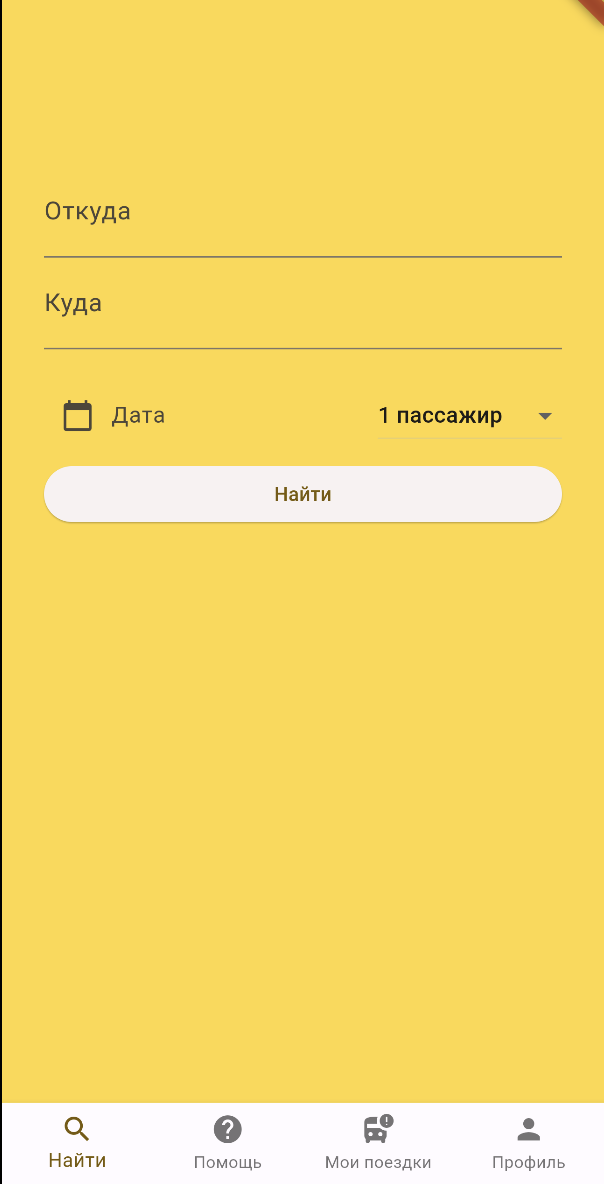


Рисунок 7.1 – Главная страница приложения

На главной странице располагается форма поиска билетов, взаимодействуя с которой можно забронировать билет.

* 1. **Руководство для гостя**

У неавторизованного пользователя есть возможность просматривать рейсы, а также регистрироваться и входить в аккаунт.

Для бронирования билетов на главной странице требуется указать требуемые для поиска параметры рейса: город отправления, город прибытия, дату отправления и количество пассажиров. После выбора всех параметров следует нажать на кнопку «Найти» для поиска. Появится список всех рейсов, которые удовлетворяют данным параметрам. Процесс просмотра рейсов указан на рисунке 7.2.

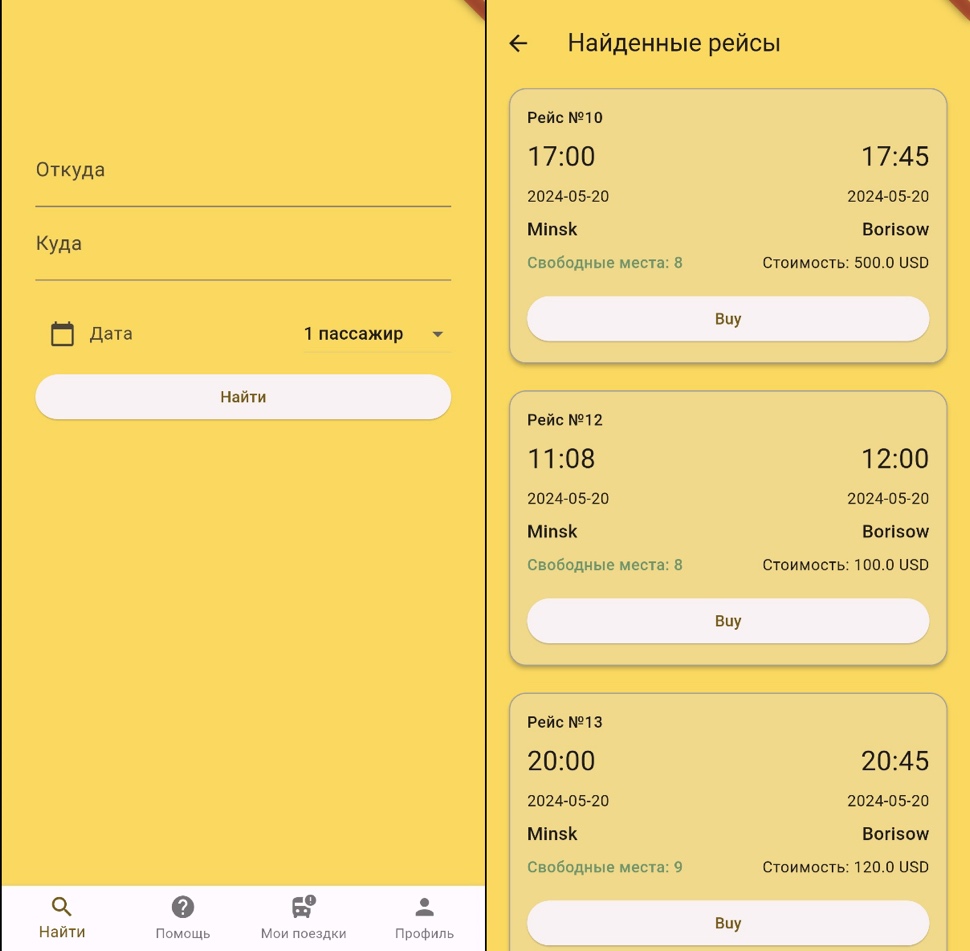


Рисунок 7.2 – Процесс просмотра рейсов

При регистрации требуется перейти на вкладку «Профиль», появится форма для регистрации аккаунта, при нажатии на кнопку рядом с кнопкой «Зарегистрироваться» откроется форма для входа в существующий аккаунт. Введя данные в форму регистрации и нажав кнопку «Зарегистрироваться» при корректных данных и успешном добавлении ново созданный пользователь добавляется в базу данных Процесс регистрации и входа представлен на рисунке 7.3.

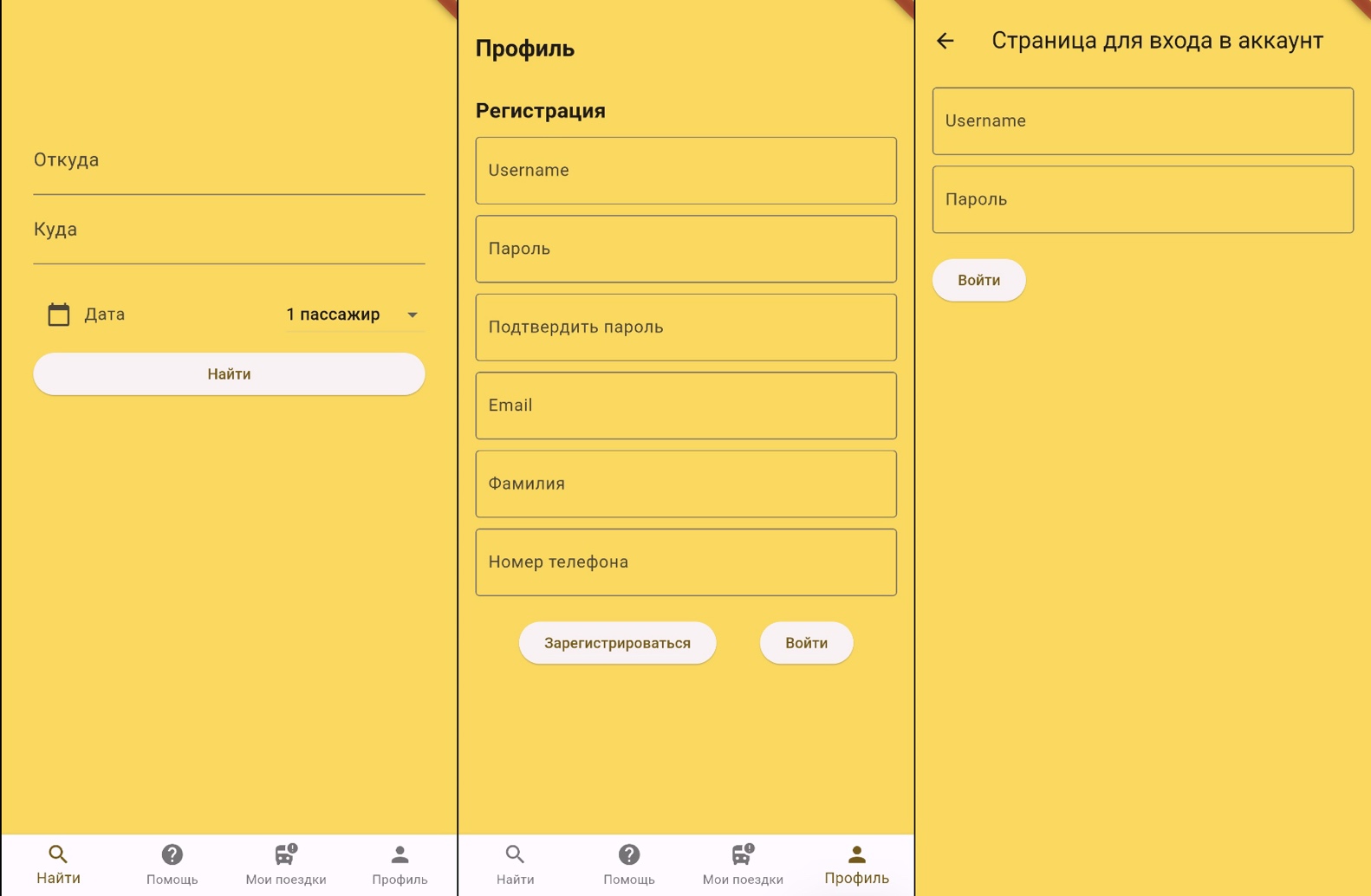


Рисунок 7.3 – Процесс регистрации и входа

После регистрации пользователь может, заполнив форму входа, попасть на страницу профиля, в котором хранятся пользовательские данные.

* 1. **Руководство для авторизованного пользователя**

Процесс заказа билета для авторизованного пользователя происходит таким образом, пользователь переходит во вкладку «Найти», затем вводит корректные данные в поля ввода, такие как «Откуда», «Куда», «Дата», «Количество пассажиров», затем нажимает кнопку «Найти». Во вкладке «Найденные рейсы» будут отображаться все возможные для заказа билеты. Информация о купленных билетах хранится на экране мои поездки. Так же на этом экране можно отказаться от билета либо же позвонить водителю для уточнения необходимой информации. Процесс работы с билетами представлен на рисунке 7.4.

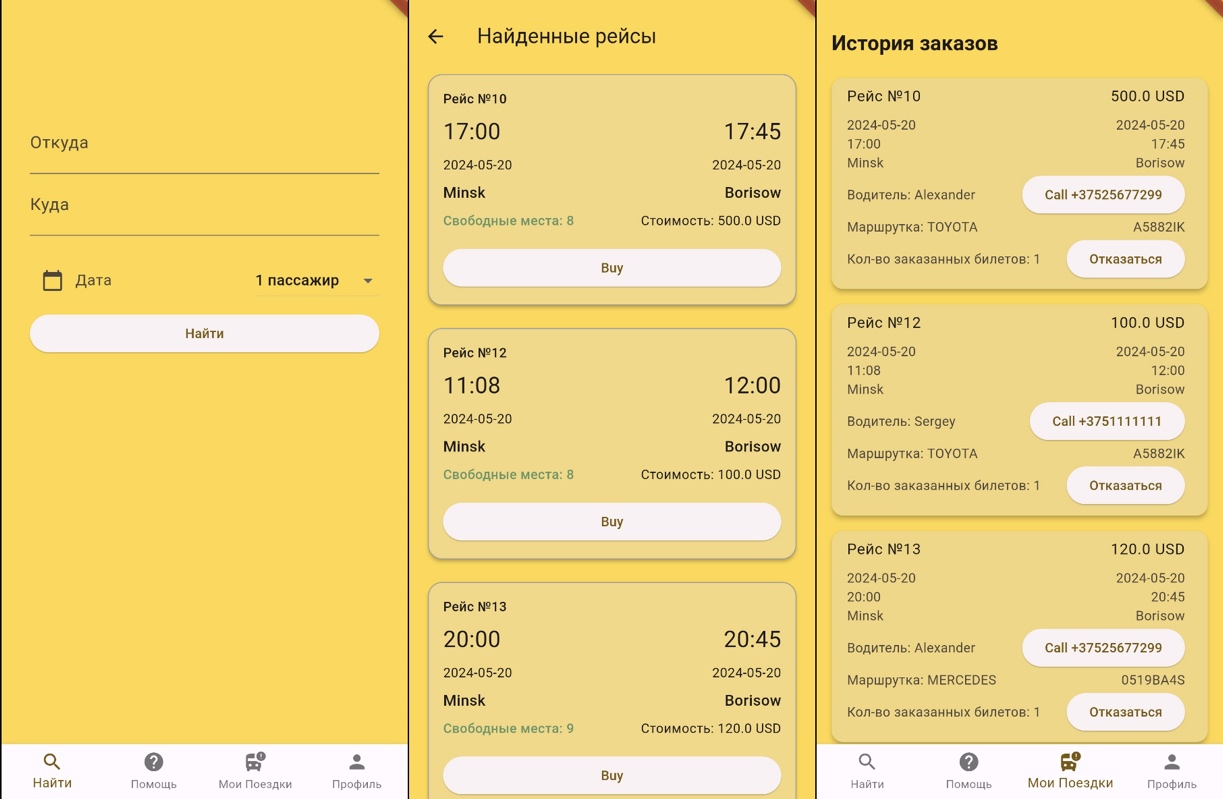


Рисунок 7.4 – Процесс работы с билетами

При переходе на вкладку «Помощь», откроется раздел, в котором будут представлены ответы на часто задаваемые вопросы а так же возможность набора номера диспетчера. Процесс изменения личной информации представлен на рисунке 7.5.

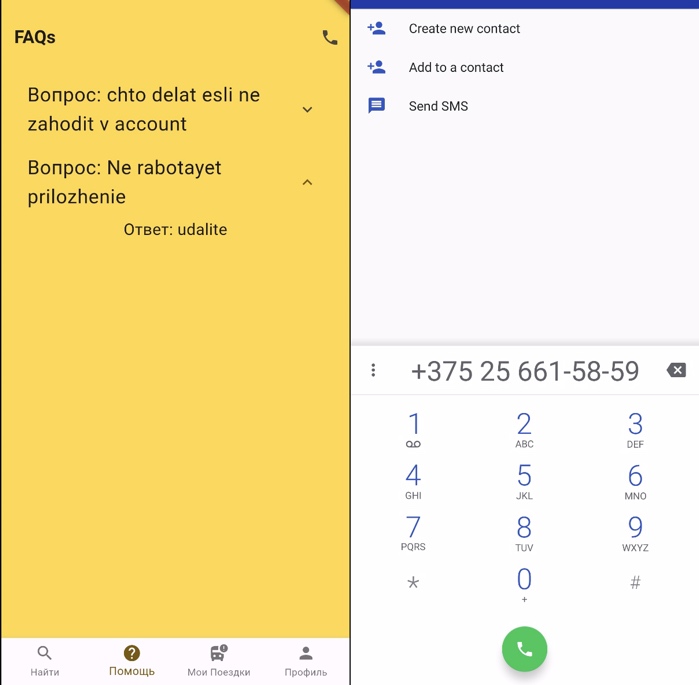


Рисунок 7.5 – Процесс связи с диспетчером

Теперь можно связаться с диспетчером по номеру телефона, или косвенно путем представленного раздела с ответами на часто задаваемые вопросы.

* 1. **Руководство для водителя**

Пользователь с ролью «driver» является расширением пользователя с ролью «user», т.к. у него помимо функционала пользователя имеется доступ к просмотру рейсов которыми он будет руководить. На данной странице водитель сможет отслеживать всех пользователей подписанных на рейс.

Процесс просмотра всех рейсов водителя представлен на рисунке 7.6.

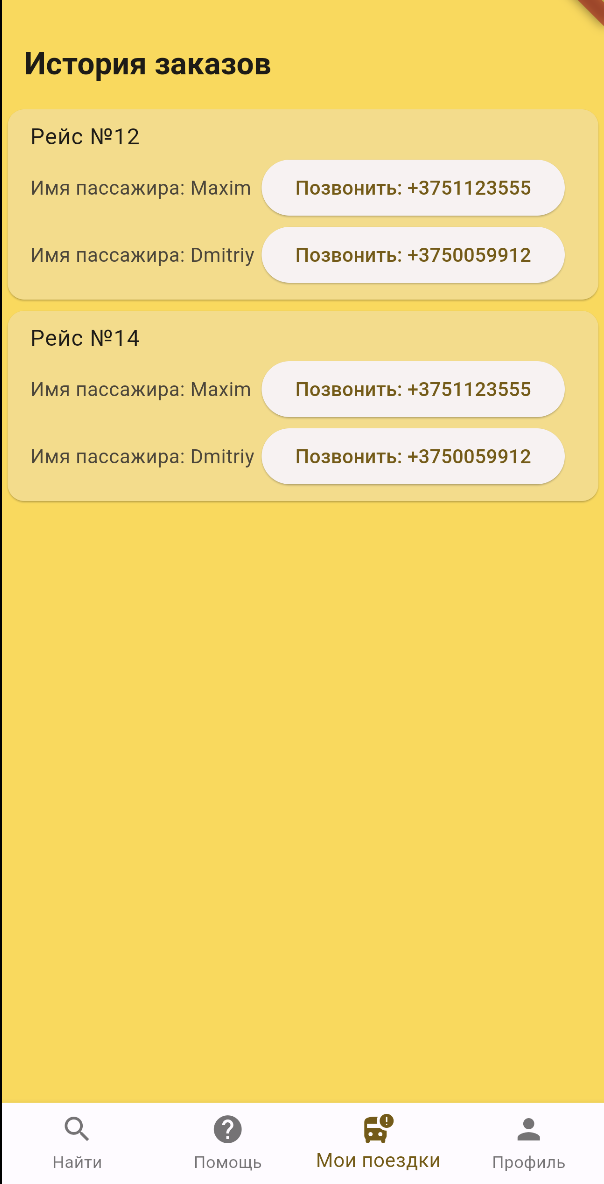


Рисунок 7.6 – Процесс работы с панелью водителя

Также водитель не может заказывать билеты на рейсы, к которым он прикреплен.

* 1. **Руководство для администратора**

У пользователя с ролью «Администратор» появляется доступ к панели администратора. В панели администратора присутствуют разделы для управления каждой из таблиц базы данных, рассмотрим управление таблицей City. Присутствует операция удаления, добавления, а также редактирования записи о городе. Для добавления города нужно перейти на вкладку DB потом нажать на кнопку Manage Cities, после чего на открытой странице ввести название города и нажать кнопку Add City. Для удаления записи о городе необходимо просто нажать на значок корзины. Для редактирования записи о городе необходимо нажать на запись после чего появится окно с редактированием таблицы, в данном окне изменить название города и нажать кнопку Save. Процесс работы с городами представлен на рисунке 7.7.

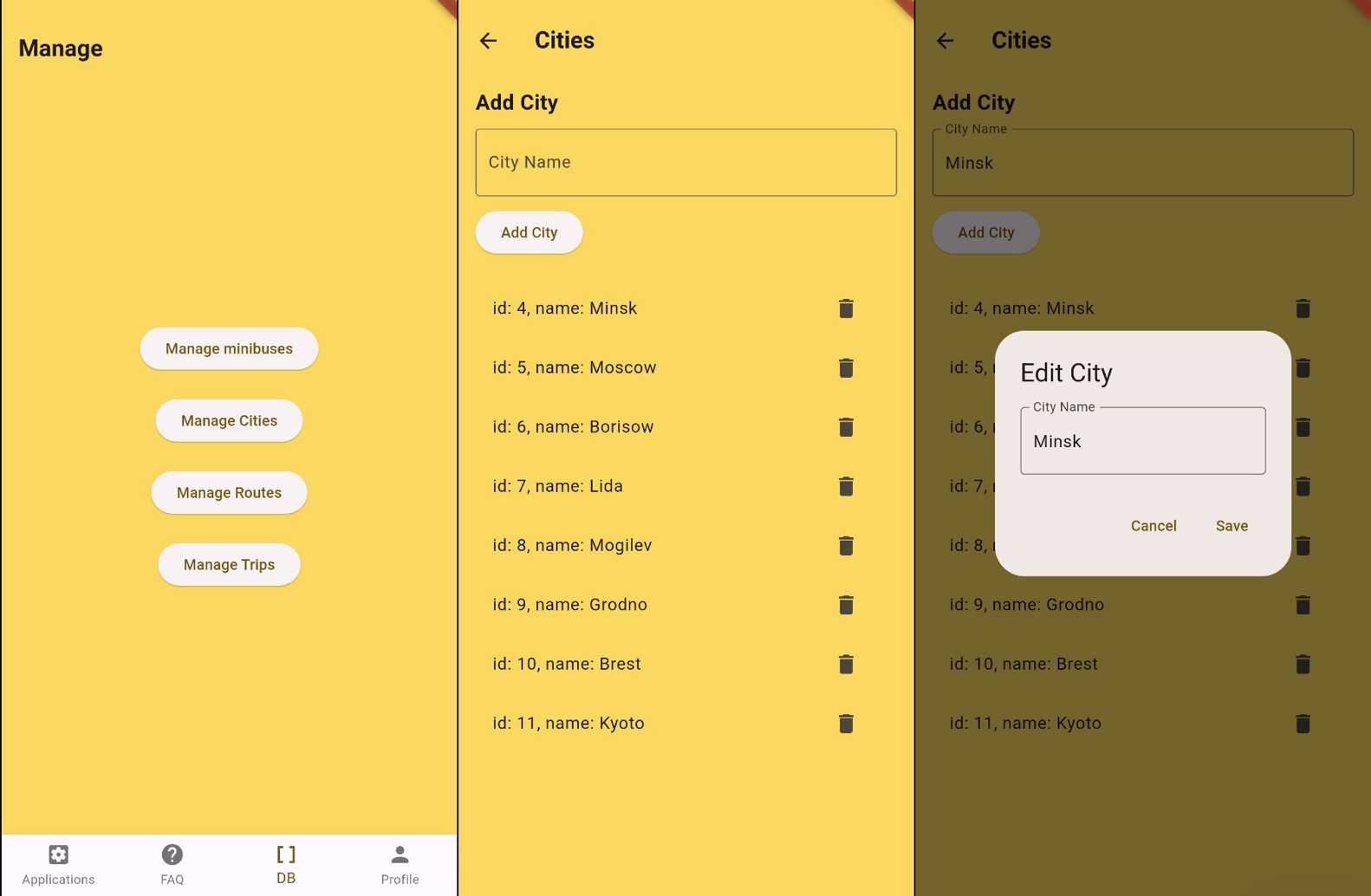


Рисунок 7.7 – Процесс работы с городами

Администратор также по аналогии может редактировать таблицы «BUS», «TRIP», «ROUTE».

Еще одной функцией администратора является возможность сделать пользователя водителем, если первый создаст заявление с указанием всей необходимой информации о своем водительском прошлом.

Процесс становления пользователя водителем происходит в несколько этапов. Для начала пользователь создает заявление, затем администратор переходит во вкладку «Applications», и в этой вкладке администратор решает одобрит ли заявление или же нет. Процесс перехода пользователя водителем предоставлен на рисунке 7.8.

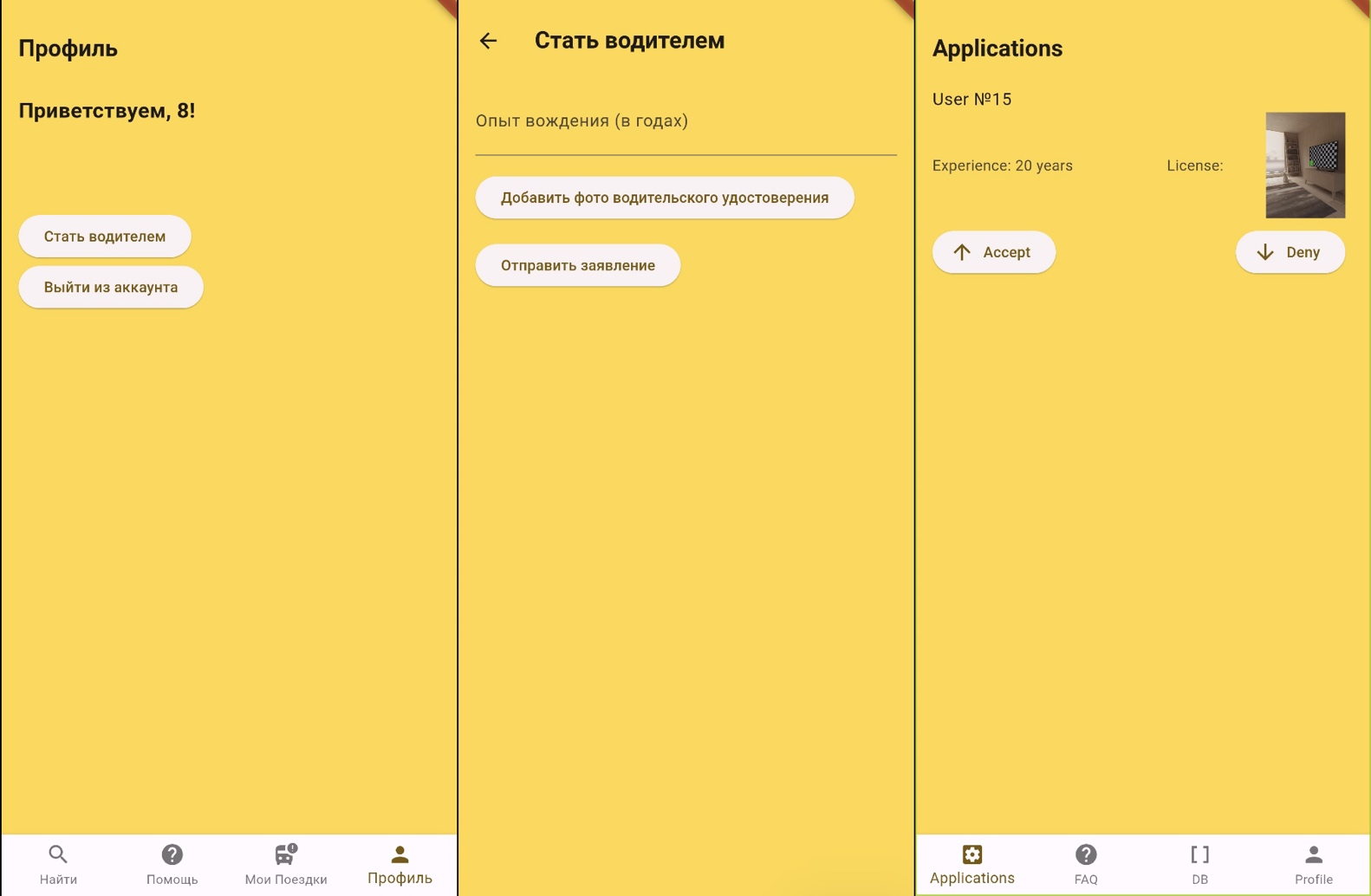
****

Рисунок 7.8 – Процесс становления пользователя водителем

Именно таким образом можно подать заявление и стать водителем.

* 1. **Вывод по разделу**

В данном разделе было представлено подробное руководство пользователя для каждой роли приложения с описание доступного функционала. Для администраторов были приведены ситуации, при которых нельзя проводить операции добавления, изменения и удаления данных из таблицы. Используя данное руководство пользователя, можно с легкостью разобраться с приложением и быстро ориентироваться, также данное руководство позволит освежить знания и узнать о новых возможностях приложения опытным пользователям.

# **Заключение**

В ходе разработки мобильного приложения для автовокзала на платформе Flutter были успешно реализованы следующие функциональные требования: авторизация и регистрация пользователей, поиск рейсов с условием, управление рейсами, личный кабинет пользователя, панель водителя и панель администратора.

Применение класса Provider пакета Provider, позволило передавать единственный экземпляр класса DatabaseNotifier, который обеспечивает взаимодействие с базой данных приложения и реализует порождающий шаблон проектирования – Singleton.Также класс AuthNotifier помог в определении авторизован пользователь или нет.

Спроектирована логическая и физическая схемы базы данных, описывающие необходимые объекты для полноценной реализации заявленных функций с разделением ролей пользователей.

Была создана страница с панелью водителя, облегчающая его работу и позволяющая просматривать все рейсы с информацией о клиентах которые сели на рейс. Также была разработана страница с панелью администратора, включающая элементы для взаимодействия со всеми сущностями базы данных.

Была разработана оптимальная схема навигации в приложении, благодаря которой пользователь может удобно получать доступ к нужным страницам. Определены UI/UX аспекты дизайна каждой страницы приложения, включая расположение интерактивных и информационных элементов, их стили, внешний вид и обработку событий взаимодействия.

Были получены навыки работы с формами и элементами пользовательского ввода, а также методы валидации входных данных.

Составлено руководство пользователя для каждой роли в приложении, а также успешно протестирована работоспособность приложения на разных программно-аппаратных платформах.

# **Список используемых источников**

1. Андрей Алеев, Быстрый старт Flutter-разработчика [Книга]/ А.В. Алеев. – Москва: Ridero, 2020. – 72 с. – Дата доступа: 29.02.2024
2. Alessandro Biessek. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2 [Книга]/ A. Biessek., 2019. – 450 с. Дата доступа – 05.03.2024.
3. Документация пакета sqflite [Электронный ресурс]. – Режим доступа: https://pub.dev/documentation/sqflite/latest/ – Дата доступа: 10.04.2024.
4. . Flutter API reference documentation [Электронный ресурс]. – Режим доступа: https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html – Дата доступа: 14.04.2024
5. Flutter & Dart from Stratch [Электронный ресурс]. –Pluralsight, 2022. – Режим доступа: https://www.pluralsight.com/courses/flutter-dart-getting-started – Дата доступа: 22.04.2024.

# **Приложение А. Классы для работы с базой данных**

|  |
| --- |
| class DatabaseHelper {  late Database db;   DatabaseHelper() {  init();  }   Future<void> init() async {  var databasesPath = await getDatabasesPath();  var fullPath = join(databasesPath, "database.db");  db = await openDatabase(fullPath, version: 1,  onCreate: (db, version) async {  await ClientHandler(db).createTable();  await RoleHandler(db).createTable();  await BusHandler(db).createTable();  await CityHandler(db).createTable();  await RouteHandler(db).createTable();  await TripHandler(db).createTable();  await FAQHandler(db).createTable();  await OrderHandler(db).createTable();  await DriverApplicationHandler(db).createTable();  });  }   Future<void> close() async => db.close(); } |