

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения высшего
образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет _____ ИТР _____

Кафедра _____ ПИН _____

КУРСОВАЯ РАБОТА

По Разработка приложений для мобильных операционных систем

Тема АИС «Таксопарка»

Руководитель

Колпаков А.А

(фамилия, инициалы)

(подпись)

(дата)

Студент ПИН - 121

(группа)

Носков М.Ю

(фамилия, инициалы)

(подпись)

(дата)

Муром 2024

В данной курсовой работе необходимо было спроектировать приложение таксопарка. В качестве средств разработки базы данных была использована среда Android Studio. Язык разработки: Kotlin.

In this course work, it was necessary to design a taxi company application. The Android Studio environment was used as a database development tool. Development language: Kotlin.

Содержание

| | |
|--|----|
| Введение..... | 6 |
| 1. Анализ технического задания..... | 8 |
| 2. Разработка алгоритмов | 9 |
| 3. Руководство программиста..... | 12 |
| 4. Руководство пользователя..... | 26 |
| Заключение..... | 28 |
| Список используемой литературы..... | 29 |
| Приложение 1..... | 30 |
| Приложение 2..... | 32 |
| Приложение 3. Снимки окон программы..... | 33 |

| | | | | | | | | |
|-----------|------|--------------|---------|-----|----------------------------------|-----------------|------|--------|
| | | | | | МИВлГУ 09.03.04 - 0.015 | | | |
| | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дат | Распределённая ИС «Таксопарк» | Лит. | Лист | Листов |
| Разраб. | | Носков М.Ю | | | | | | |
| Провер. | | Колпаков А.А | | | | | 4 | 41 |
| Реценз. | | | | | | МИ ВлГУ ПИН-121 | | |
| Н. Контр. | | | | | | | | |
| Утверд. | | | | | | | | |

Введение

Основой жизнеобеспечения любого города является пассажирский транспорт. Если говорить о массовом пассажирском транспорте, таком как маршрутный транспорт (автобусы, троллейбусы), то основные критерии контроля и управления так или иначе связаны с конкретным маршрутом, «пиковыми» нагрузками на маршрут или его участки. Однако стоит уделить отдельное внимание такому виду транспорта как такси, где нет конкретного маршрута следования, остановочных пунктов, расписания и т.п.

Любой программный продукт, призванный автоматизировать систему управления службой такси, является узким, специализированным отраслевым программным продуктом. Это связано не только со спецификой такого вида транспорта как такси, но и с наличием «посредника» в оказании транспортной услуги - диспетчера.

Автоматизированная информационная система (АИС) таксопарка представляет собой комплексное решение, направленное на оптимизацию процессов управления такси. Система обеспечивает хранение и обработку данных о водителях, автомобилях и заказах, что позволяет диспетчерам эффективно управлять ресурсами и повышать качество обслуживания клиентов. В условиях растущей конкуренции на рынке такси внедрение АИС становится необходимостью для повышения эффективности работы.

Автоматизированные информационные системы становятся неотъемлемой частью современного бизнеса, и таксопарки не являются исключением. АИС таксопарка обеспечивает эффективное управление ресурсами, включая водителей, автомобили и заказы. В условиях растущей конкуренции и потребности в оптимизации процессов, создание такой системы на платформе Android с использованием языка Kotlin представляет собой актуальную задачу.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 6 |

Цель данной курсовой работы — разработка АИС «Таксопарка» на платформе Android с использованием языка программирования Kotlin. Система будет включать в себя функционал для клиента, позволяющий отслеживать состояние автомобилей и создавать заказы. Важным аспектом является также обеспечение надежности и удобства использования приложения.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| | | | | | | 7 |
| Изм. | Лис | № докум. | Подпись | Дат | | |

Анализ технического задания.

Для успешной реализации проекта необходимо детально рассмотреть каждое требование технического задания.

1. Функционал:

- Сведения об автомобилях: учет автомобилей на линии и в ремонте. Приложение будет отслеживать состояние автомобилей — на линии или в ремонте.

- Сведения о заказах и бронированиях: регистрация новых заказов, подтверждение бронирований, мониторинг статуса заказов. Приложение будет показывать текущий статус заказов.

2. Операционная система и языки программирования:

Android: Приложение будет разработано для платформы Android, что обеспечит доступ к широкой аудитории пользователей. Kotlin: Использование языка Kotlin позволит создать современное и безопасное приложение с лаконичным кодом.

3. Документация:

- Документирование кода с помощью инструмента Dokka. Данный инструмент предоставляет структурированный и удобный формат документации, который облегчит понимание и поддержку разрабатываемого приложения.

4. Система контроля версий:

- Размещение исходного кода в репозитории системы контроля версий GitHub. Это обеспечит централизованное управление версиями, возможность совместной работы над проектом и легкий доступ к истории изменений.

Реализация каждого этапа должна быть тщательно спланирована и согласована с заказчиком для достижения максимальной эффективности и качества разрабатываемой системы.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 8 |

2. Разработка алгоритмов

2.1. Общая структура приложения

Приложение АИС «Аксопарк» построено на архитектуре MVC (Model-View-Controller), которая разделяет логику работы программы, интерфейс пользователя и управление данными. Это позволяет упростить масштабирование и поддержку приложения.

Была создана база данных для приложения (приложение 1), являющееся основой при создании приложения.

Основные модули системы:

Модель (Model): отвечает за работу с базой данных (классы DatabaseHelper и DbHelepr2).

Контроллер (Controller): содержит бизнес-логику и обработку событий (основные классы MainActivity2, AuthActivity, RegistrationActivity и другие).

Представление (View): обеспечивает взаимодействие с пользователем (файлы макетов XML).

2.2. Логика работы с базой данных

Для управления данными используется встроенный механизм SQLite. База данных содержит таблицы:

Users: хранит данные о пользователях (логин, пароль, email и номер телефона).

Vehicles: информация об автомобилях (модель, номер, статус).

Drivers: данные о водителях (имя, номер лицензии, рейтинг и телефон).

Orders: хранит информацию о заказах (водитель, пользователь, маршрут, статус).

Bookings: хранит информацию о бронированиях.

Создание и обновление базы данных

Инициализация базы данных выполняется в классе DbHelepr2. При первом запуске приложения база копируется из ассетов, а при обновлении версии автоматически пересоздается:

```
override fun onCreate(db: SQLiteDatabase?) {
```

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 9 |

```
val createUsersTable = "CREATE TABLE Users (id INTEGER PRIMARY KEY, login TEXT, password TEXT, email TEXT, phoneNumber TEXT)"
db?.execSQL(createUsersTable) }
```

2.3. Алгоритмы авторизации и регистрации

Алгоритм регистрации

Пользователь вводит данные: логин, email, номер телефона, пароль.

Система проверяет заполнение всех обязательных полей.

Данные добавляются в таблицу Users.

Пользователь автоматически сохраняется как текущий в SharedPreferences.

Алгоритм авторизации

Пользователь вводит логин и пароль.

Проверка данных выполняется через метод getUserByUsernameAndPassword() класса DbHelepr2.

В случае успеха система загружает данные пользователя в SharedPreferences.

2.4. Алгоритм отображения информации об автомобилях

В классе MainActivity2 реализован метод displayOrderInfo(), который:

Выполняет SQL-запрос к таблице Vehicles для получения всех автомобилей.

Формирует список ArrayList<HashMap<String, Any>> для отображения.

Отображает данные с помощью адаптера VehicleAdapter.

Пример получения данных об автомобилях:

```
val cursor = mDb.rawQuery("SELECT * FROM Vehicles", null)
cursor.use {
    while (it.moveToNext()) {
        val vehicle = HashMap<String, Any>()
        vehicle["Model"] = it.getString(it.getColumnIndex("Model"))
        vehicle["LicensePlate"] = it.getString(it.getColumnIndex("LicensePlate"))
        vehicle["StatusAuto"] = it.getString(it.getColumnIndex("StatusAuto"))
    }
}
```

```
orders.add(vehicle)
```

```
} }
```

2.5. Алгоритм работы с заказами

Обработка заказов реализована в классе DatabaseHelper. Основные методы:

addOrder(order: Order): добавление нового заказа.

getAllOrders(): получение списка всех заказов.

getUserBookings(userId: Int): получение заказов конкретного пользователя.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 11 |

3. Руководство программиста

Приложение включает в себя работу с базой данных, аутентификацию и регистрацию пользователей, а также функциональность для работы с бронированиями и профилями пользователей. Руководство охватывает основные моменты работы с каждым компонентом приложения, начиная с создания базы данных и заканчивая обработкой пользовательского интерфейса.

Основные компоненты приложения:

MainActivity — главная активность, с кнопками для навигации по различным разделам (машины, водители, заказы, профиль).

AuthActivity — экран авторизации пользователя.

RegistrationActivity — экран регистрации нового пользователя.

BookingsActivity — экран для отображения бронирований.

ProfileActivity — экран для отображения профиля пользователя.

DbHelepr2 — класс для работы с базой данных SQLite.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 12 |

3.1 MainActivity: Главная активность

MainActivity отвечает за отображение главного экрана приложения и навигацию по остальным экранам, связанным с автомобилями, водителями, заказами и профилем пользователя.

Описание методов:

onCreate():

Это основной метод для инициализации UI. В нем настраиваются кнопки и их обработчики. Внутри мы находим все кнопки по их ID и задаем для них обработчики нажатий, чтобы при клике происходил переход на соответствующие экраны.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    // Инициализация кнопок  
    val buttonAuto = findViewById<Button>(R.id.carsButton)  
    val buttoDrivers = findViewById<Button>(R.id.driversButton)  
    val buttoOrders = findViewById<Button>(R.id.ordersButton)  
    val buttonRegistration = findViewById<Button>(R.id.loginButton)  
    val buttonAuth = findViewById<Button>(R.id.authButton)  
    val bookingsButton = findViewById<Button>(R.id.bookingsButton)  
    val profileButton = findViewById<Button>(R.id.profileButton)  
    // Установка обработчиков для кнопок  
    buttonAuto.setOnClickListener {  
        val intent = Intent(this@MainActivity, MainActivity2::class.java)  
        startActivity(intent) // Переход на экран с автомобилями  
    }  
    buttoDrivers.setOnClickListener {  
        val intent = Intent(this@MainActivity, DriverActivity::class.java)  
        startActivity(intent) // Переход на экран с водителями  
    }  
}
```

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 13 |

```

buttonOrders.setOnClickListener {
    val intent = Intent(this@MainActivity, OrdersActivity::class.java)
    startActivity(intent) // Переход на экран с заказами
}
buttonRegistration.setOnClickListener {
    val intent = Intent(this@MainActivity, RegistrationActivity::class.java)
    startActivity(intent) // Переход на экран регистрации
}
buttonAuth.setOnClickListener {
    val intent = Intent(this@MainActivity, AuthActivity::class.java)
    startActivity(intent) // Переход на экран авторизации
}
bookingsButton.setOnClickListener {
    val intent = Intent(this@MainActivity, BookingsActivity::class.java)
    startActivity(intent) // Переход на экран бронирований
}
profileButton.setOnClickListener {
    val intent = Intent(this@MainActivity, ProfileActivity::class.java)
    startActivity(intent) // Переход на экран профиля
}
}

```

3.2 AuthActivity: Экран авторизации

AuthActivity предназначена для ввода логина и пароля пользователем, а также для проверки этих данных с базой данных.

Описание методов:

onCreate():

Этот метод инициализирует UI: находит элементы для ввода логина и пароля, а также кнопку для входа.

При нажатии на кнопку "Войти", программа проверяет данные пользователя в базе данных с помощью метода getUserByUsernameAndPassword() из DbHelepr2.

Если данные верны, сохраняется ID пользователя в SharedPreferences и происходит переход на главный экран. В случае неверных данных показывается ошибка.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_auth)  
    dbHelper = DbHelepr2(this)  
    val username = findViewById<EditText>(R.id.username)  
    val password = findViewById<EditText>(R.id.password)  
    val loginButton = findViewById<Button>(R.id.loginButton)  
    loginButton.setOnClickListener {  
        val userNameText = username.text.toString()  
        val passwordText = password.text.toString()  
        // Проверка, чтобы поля не были пустыми  
        if (userNameText.isNotEmpty() && passwordText.isNotEmpty()) {  
            // Проверка данных в базе  
            val isValidUser = dbHelper.getUserByUsernameAndPassword(userNameText, passwordText)
```

```

        // Если пользователь найден, сохраняем его ID и переходим на
главный экран
        if (isValidUser) {
            val userId = dbHelper.getUserIdByUsername(userNameText)
            val sharedPreferences = getSharedPreferences("TaxiParkPrefs",
MODE_PRIVATE)
            sharedPreferences.edit().putInt("LoggedInUserId", userId).apply()
// Сохранение ID пользователя
            Toast.makeText(this, "Вы успешно авторизовались",
Toast.LENGTH_SHORT).show()
            finish() // Закрытие экрана авторизации и переход на главный
экран
        } else {
            // Если данные неверные
            Toast.makeText(this, "Неправильный логин или пароль",
Toast.LENGTH_SHORT).show()
        }
    } else {
        // Если поля пустые
        Toast.makeText(this, "Пожалуйста, заполните все поля",
Toast.LENGTH_SHORT).show()
    }
}
}
}

```


3.3 RegistrationActivity: Экран регистрации

RegistrationActivity позволяет пользователю создать новый аккаунт в приложении.

Описание методов:

onCreate():

В этом методе инициализируются все элементы интерфейса для ввода данных пользователя: логин, email, телефон и пароль. При нажатии на кнопку "Зарегистрироваться" выполняется проверка на пустые поля. После успешной проверки данные добавляются в базу данных с помощью SQL-запроса и compileStatement().

Данные также сохраняются в SharedPreferences.

По завершению регистрации, выводится сообщение об успехе, и экран регистрации закрывается.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_registration)  
    dbHelper = DbHelper2(this)  
    val username = findViewById<EditText>(R.id.username)  
    val email = findViewById<EditText>(R.id.email)  
    val phoneNumber = findViewById<EditText>(R.id.phoneNumber)  
    val password = findViewById<EditText>(R.id.password)  
    val registerButton = findViewById<Button>(R.id.registerButton)  
  
    registerButton.setOnClickListener {  
        val userNameText = username.text.toString()  
        val emailText = email.text.toString()  
        val phoneNumberText = phoneNumber.text.toString()  
        val passwordText = password.text.toString()  
  
        // Проверка, чтобы все поля были заполнены  
        if (userNameText.isNotEmpty() && emailText.isNotEmpty() &&  
passwordText.isNotEmpty()) {  
            // Вставка данных пользователя в базу данных  
            val db = dbHelper.writableDatabase
```

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВЛГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 17 |

```

        val query = "INSERT INTO Users (Username, Email, PhoneNumber,
Password) VALUES (?, ?, ?, ?)"
        val stmt = db.compileStatement(query)
        stmt.bindString(1, userNameText)
        stmt.bindString(2, emailText)
        stmt.bindString(3, phoneNumberText)
        stmt.bindString(4, passwordText)
        stmt.executeInsert()

        // Сохранение данных в SharedPreferences
        val sharedPreferences = getSharedPreferences("TaxiParkPrefs",
MODE_PRIVATE)
        with(sharedPreferences.edit()) {
            putString("Логин", userNameText)
            putString("Почта", emailText)
            putString("Номер телефона", phoneNumberText)
            apply()
        }
        Toast.makeText(this, "Вы успешно зарегистрировались",
Toast.LENGTH_SHORT).show()
        finish() // Закрытие экрана регистрации
    } else {
        // Если какие-то поля пустые
        Toast.makeText(this, "Пожалуйста, заполните все поля",
Toast.LENGTH_SHORT).show()
    }
}
}
}

```

Работа с базой данных:

DbHelepr2 — это класс для работы с базой данных SQLite. Он выполняет несколько ключевых функций:

Создание и копирование базы данных. База данных копируется из ассетов в файловую систему устройства при первом запуске приложения.

Обновление базы данных. Если версия базы данных изменяется, происходит ее обновление.

Запросы к базе данных. Класс выполняет SQL-запросы для работы с данными пользователей и бронированиями.

Основные методы:

openDataBase(): Открывает базу данных для чтения и записи.

getUserByUsernameAndPassword(): Проверяет, существует ли пользователь с указанным логином и паролем.

getUserIdByUsername(): Возвращает ID пользователя по логину.

createBooking(): Создает запись о новом бронировании.

getUserBookings(): Возвращает список бронирований пользователя.

checkTableExists(): Проверяет существование таблицы в базе данных.

Основные задачи класса:

1. Инициализация и создание базы данных.
2. Копирование базы данных из активов, если база еще не существует.
3. Обновление базы данных при изменении версии.
4. Предоставление методов для выполнения SQL-запросов к базе данных, таких как создание записей и извлечение информации.

Методы класса

1. init (Конструктор)

```
init {  
    copyDataBase() // Копирует базу данных, если она еще не существует  
    readableDatabase // Открывает базу данных в режиме чтения  
}
```

В конструкторе класса вызывается метод `copyDataBase()`, который проверяет, существует ли база данных. Если она не существует, метод копирует ее из ассетов.

`readableDatabase` иницирует создание базы данных, если она еще не была создана, и затем закрывает ее.

2. `copyDataBase()`

```
private fun copyDataBase() {  
    if (!checkDataBase()) { // Проверяем, существует ли база данных  
        readableDatabase // Создает базу данных  
        close() // Закрывает соединение с базой данных  
        try {  
            copyDBFile() // Копирует файл базы данных из ассетов  
        } catch (mIOException: IOException) {  
            throw Error("ErrorCopyingDataBase") // Если произошла ошибка  
копирования  
        }  
    }  
}
```

Метод проверяет, существует ли уже база данных с помощью `checkDataBase()`. Если базы данных нет, он вызывает `readableDatabase`, чтобы создать пустую базу данных. Затем вызывается `copyDBFile()`, который копирует файл базы данных из папки `assets` в нужное место.

3. `copyDBFile()`

`@Throws(IOException::class)`

```
private fun copyDBFile() {  
    val mInput: InputStream = mContext.assets.open(DB_NAME) //  
Открываем файл базы данных из assets  
    val mOutput: OutputStream = FileOutputStream(DB_PATH + DB_NAME)  
    // Открываем поток для записи в директорию базы данных  
    val mBuffer = ByteArray(1024) // Буфер для копирования данных
```

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 20 |

```

var mLength: Int
while (mInput.read(mBuffer).also { mLength = it } > 0) {
    mOutput.write(mBuffer, 0, mLength) // Копируем данные
}
mOutput.flush()
mOutput.close()
mInput.close()
}

```

В этом методе происходит копирование файла базы данных из ассетов в файловую систему устройства. Для этого используются потоки ввода и вывода. Буфер используется для того, чтобы копировать данные блоками по 1024 байта.

4. checkDataBase()

```

private fun checkDataBase(): Boolean {
    val dbFile = File(DB_PATH + DB_NAME) // Путь к базе данных
    return dbFile.exists() // Проверяем, существует ли файл базы данных
}

```

Метод проверяет, существует ли файл базы данных по указанному пути. Если файл существует, возвращается true, в противном случае — false.

5. openDataBase()

```
@Throws(SQLException::class)
```

```

fun openDataBase(): Boolean {
    mDataBase = SQLiteDatabase.openDatabase(DB_PATH + DB_NAME,
    null, SQLiteDatabase.CREATE_IF_NECESSARY)
    return mDataBase != null // Проверяем, удалось ли открыть базу данных
}

```

Метод открывает базу данных для чтения и записи. Если база данных не существует, она будет создана. Возвращает true, если база данных открыта успешно, и false в случае ошибки.

6. onCreate() и onUpgrade()

```
override fun onCreate(db: SQLiteDatabase) {
```

// Этот метод вызывается, когда база данных создается впервые. В данном случае он не используется, так как база уже будет копироваться из ассетов.

```
}
```

```
override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
```

```
    if (newVersion > oldVersion) {
```

mNeedUpdate = true // Устанавливаем флаг необходимости обновления базы данных

```
    }
```

```
}
```

onCreate(): Этот метод вызывается при создании базы данных, но в данном случае он не используется, так как база данных копируется из ассетов, а не создается с нуля.

onUpgrade(): Этот метод вызывается при обновлении версии базы данных. Если новая версия базы данных больше текущей, устанавливается флаг mNeedUpdate = true, что сигнализирует о необходимости обновления.

7. getUserByUsernameAndPassword()

```
fun getUserByUsernameAndPassword(username: String, password: String): Boolean {
```

```
    val db = this.readableDatabase // Открываем базу данных для чтения
```

```
    val cursor = db.rawQuery(
```

```
        "SELECT * FROM Users WHERE Username = ? AND Password = ?",  
        arrayOf(username, password)
```

```
    )
```

```
    return cursor.count > 0 // Если есть результат — пользователь найден
```

```
}
```

Метод выполняет SQL-запрос для поиска пользователя по имени пользователя (username) и паролю (password). Возвращает true, если пользователь найден (т.е. количество строк в курсоре больше нуля), иначе false.

8. createBooking()

```
fun createBooking(orderId: Int, pickupLocation: String, dropoffLocation:
String, status: String, userId: Int): Long {
    val db = writableDatabase // Открываем базу данных для записи
    val contentValues = ContentValues().apply {
        put("OrderID", orderId)
        put("BookingDate", System.currentTimeMillis()) // Сохраняем текущую
дату и время
        put("PickupLocation", pickupLocation)
        put("DropoffLocation", dropoffLocation)
        put("Status", status)
    }
    return db.insert("Booking", null, contentValues) // Вставляем новое
бронирование в таблицу Booking
}
```

Метод добавляет новое бронирование в таблицу Booking. Используется объект ContentValues, который позволяет добавлять значения в базу данных.

Возвращает ID вставленной записи (или -1 в случае ошибки).

9. getUserIdByUsername()

```
fun getUserIdByUsername(username: String): Int {
    val db = readableDatabase // Открываем базу данных для чтения
    val cursor = db.rawQuery("SELECT UserID FROM Users WHERE
Username = ?", arrayOf(username))
    var userId = -1
    if (cursor.moveToFirst()) {
        userId = cursor.getInt(0) // Извлекаем UserID
    }
}
```

```

    }
    cursor.close()
    return userId
}

```

Метод ищет пользователя по имени пользователя и возвращает его ID (UserID).

Если пользователь не найден, возвращает -1.

10. checkTableExists()

```

fun checkTableExists(tableName: String): Boolean {
    val db = this.readableDatabase // Открываем базу данных для чтения
    val cursor = db.rawQuery(
        "SELECT name FROM sqlite_master WHERE type='table' AND
name=?",
        arrayOf(tableName)
    )
    val exists = cursor.count > 0 // Если таблица существует
    cursor.close()
    return exists
}

```

Метод проверяет, существует ли таблица в базе данных. Выполняется запрос к системной таблице sqlite_master, которая хранит информацию о всех таблицах в базе данных.

Возвращает true, если таблица существует, и false, если нет.

Экран авторизации (AuthActivity):

В этой активности пользователю предлагается ввести логин и пароль для авторизации.

Основные шаги:

Пользователь вводит логин и пароль в соответствующие поля.

При нажатии на кнопку "Войти" выполняется проверка данных с использованием метода `getUserByUsernameAndPassword` из класса `DbHelepr2`. Если данные правильные, сохраняются настройки пользователя в `SharedPreferences` и происходит переход к основному экрану. В случае неверных данных выводится сообщение об ошибке.

```
loginButton.setOnClickListener {  
    val userNameText = username.text.toString()  
    val passwordText = password.text.toString()  
    if (userNameText.isNotEmpty() && passwordText.isNotEmpty()) {  
        val                                     isValidUser                                     =  
dbHelper.getUserByUsernameAndPassword(userNameText, passwordText)  
        if (isValidUser) {  
            val userId = dbHelper.getUserIdByUsername(userNameText)  
            val  sharedPreferences  =  getSharedPreferences("TaxiParkPrefs",  
MODE_PRIVATE)  
            sharedPreferences.edit().putInt("LoggedInUserId", userId).apply()  
            Toast.makeText(this, "Вы успешно авторизовались",  
Toast.LENGTH_SHORT).show()  
            finish() // Переход к основному экрану  
        } else {  
            Toast.makeText(this, "Неправильный логин или пароль",  
Toast.LENGTH_SHORT).show()  
        }  
    } else {Toast.makeText(this, "Пожалуйста, заполните все поля",  
Toast.LENGTH_SHORT).show()}}
```

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 25 |

4. Руководство пользователя

Руководство пользователя для мобильного приложения

Это руководство поможет вам освоиться с основными функциями и страницами нашего Android-приложения. В нем описаны ключевые экраны и действия, которые доступны пользователю.

Главная страница (MainActivity)

На главной странице приложения вы увидите:

Логотип приложения — расположен в верхней части экрана. Это изображение представляет бренд и является визуальным элементом интерфейса.

Кнопки навигации:

Водители: Откроет раздел, посвященный водителям.

Автомобили: Откроет раздел с информацией о доступных автомобилях.

Мои заказы: Перейдет к списку ваших заказов.

Бронирования: Откроет раздел с вашими активными бронированиями.

Мой профиль: Перейдет к вашему личному профилю.

Страница авторизации:

Поле для ввода логина — введите ваш логин для входа в систему.

Поле для ввода пароля — введите ваш пароль.

Кнопка "Войти" — после ввода логина и пароля, нажмите эту кнопку, чтобы войти в приложение.

Если вы еще не зарегистрированы, можете создать новый аккаунт, перейдя по кнопке регистрации на главной странице.

Страница выбора водителя и местоположений:

Выпадающее окно для выбора водителя — используйте выпадающий список для выбора водителя. Поле для ввода местоположения введите адрес, с которого вы хотите, чтобы водитель забрал вас. Поле для ввода места назначения введите адрес, куда вы хотите поехать.

Кнопка "Сохранить заказ" — после ввода всех данных нажмите эту кнопку, чтобы сохранить ваш заказ.

Страница автомобилей:

В разделе автомобилей вы найдете список доступных машин с подробной информацией. Каждый элемент списка включает название автомобиля и фото автомобиля.

Основные действия в приложении

Регистрация и авторизация: Чтобы использовать приложение, вам необходимо зарегистрироваться или войти в систему. Для этого нажмите на кнопку Регистрация или Авторизация на главной странице.

Создание заказа: Чтобы заказать автомобиль, выберите водителя и введите нужные местоположения. Затем нажмите Сохранить заказ, чтобы оформить заявку.

Просмотр автомобилей: Перейдите в раздел Автомобили, чтобы ознакомиться с доступными транспортными средствами. Вы можете увидеть фотографии автомобилей и прочитать описание.

Просмотр и управление профилем: На странице профиля вы можете увидеть свою информацию, а также выйти из приложения.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 27 |

Заключение

Ниже описаны основные задачи, реализованные в данной курсовой работе:

Основные этапы разработки включают:

Проектирование интерфейса: В процессе создания приложения была разработана простая и интуитивно понятная структура, обеспечивающая комфортное взаимодействие пользователя с функционалом. Страница входа и регистрации помогает новым пользователям быстро создать аккаунт или войти в систему. Главная страница предоставляет быстрый доступ ко всем основным разделам, таким как управление автомобилями, водителями, заказами и профилем.

Реализация ключевых функций: Важнейшими функциями приложения являются бронирование такси, просмотр профиля, а также просмотр и добавление заказов. Пользователи могут легко забронировать такси, выбрать водителя, а также уточнить места посадки и высадки. В приложении предусмотрены кнопки для быстрого перехода к необходимым разделам, таким как "Мои заказы", "Бронирования", и "Мой профиль".

Удобство для пользователей: Приложение предлагает простоту использования, с возможностью быстрого выполнения всех действий, таких как создание заказа, просмотр списка автомобилей, изменение профиля или выход из системы.

Разработанное приложение для таксопарка позволяет эффективно управлять процессом бронирования такси, а также предоставляет пользователям удобный интерфейс для взаимодействия с системой. Учитывая удобство интерфейса, безопасность данных и функциональные возможности, приложение может стать надежным инструментом как для пользователей, так и для владельцев таксопарков, обеспечивая удобство, безопасность и эффективную работу бизнеса.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 28 |

Список литературы:

1. Введение в разработку приложений для смартфонов на ОС Android / А.Семакова Национальный открытый Университет «ИНТУИТ», 2016
2. Колисниченко Д.Н. Программирование для Android 5. Самоучитель. — СПб.: БХВ-Петербург, 2015. — 303 с.
3. 2016. Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. — СПб.: Питер, Гриффитс Дэвид, Гриффитс Дон Head First. Программирование для Android. 2-е изд. — СПб.: Питер, 2018. — 912 с.

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 29 |

Приложение

Приложение 1

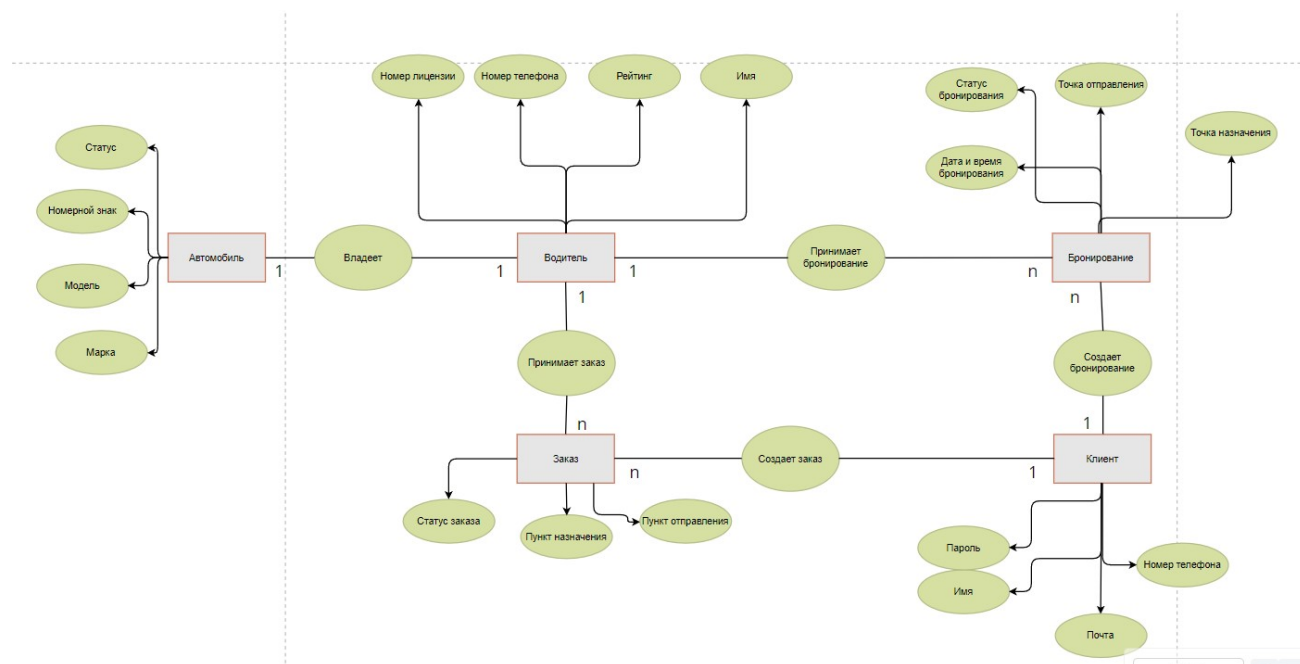


Рисунок 1 - Концептуальная модель данных

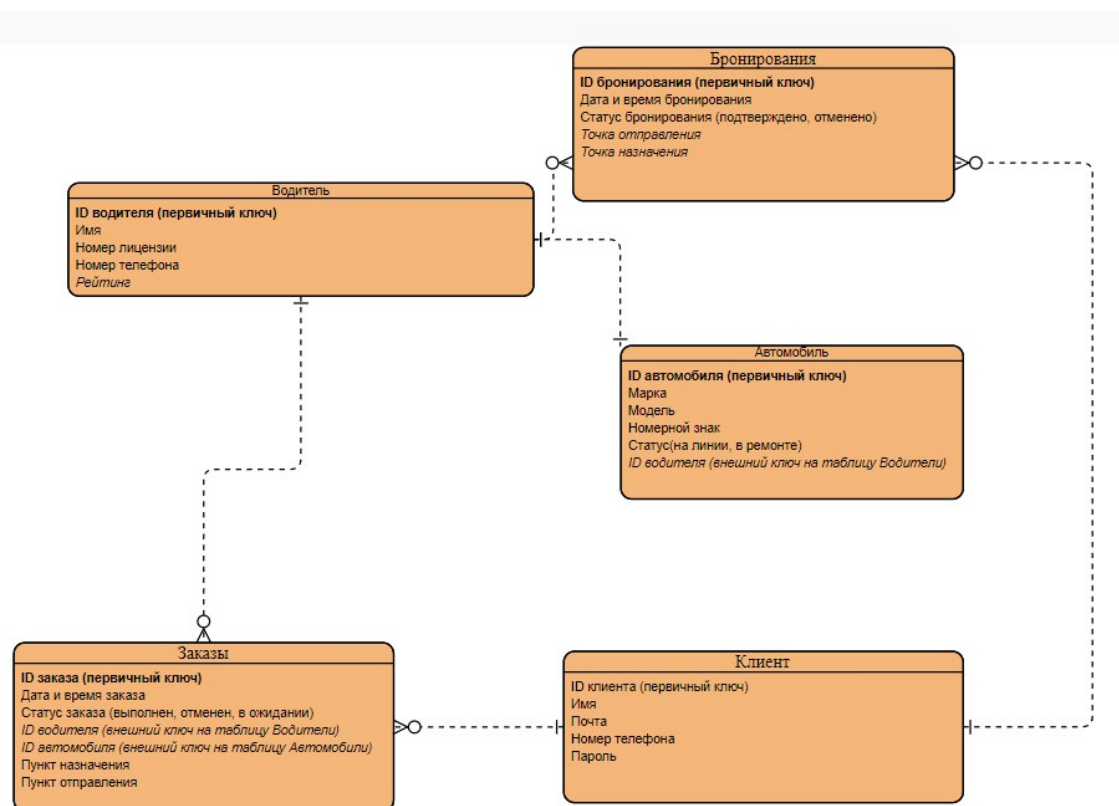


Рисунок 2 – Логическая модель данных

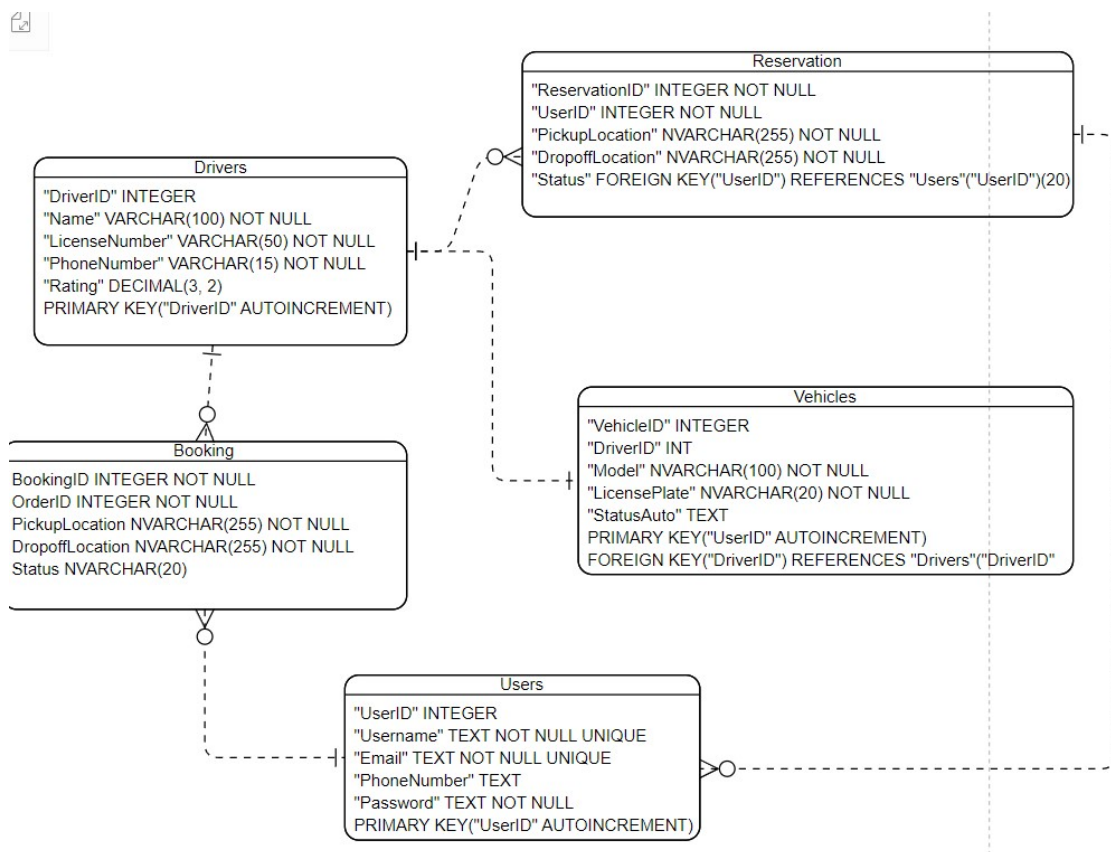


Рисунок 3 – Физическая модель данных

Приложение 2 . Программный код продукта

В процессе разработки программного продукта работы таксопарка были проведены тесты, которые подтвердили соответствие всех функциональных требований. Выявленные ошибки были устранены, что повысило качество программы. В целом, программа успешно прошла тестирование и готова к использованию клиентами таксопарка.

Ссылка на репозиторий GitHub

<https://github.com/MaxaDromka/TaxiPark>

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 32 |

Приложение 3. Снимки окон программы

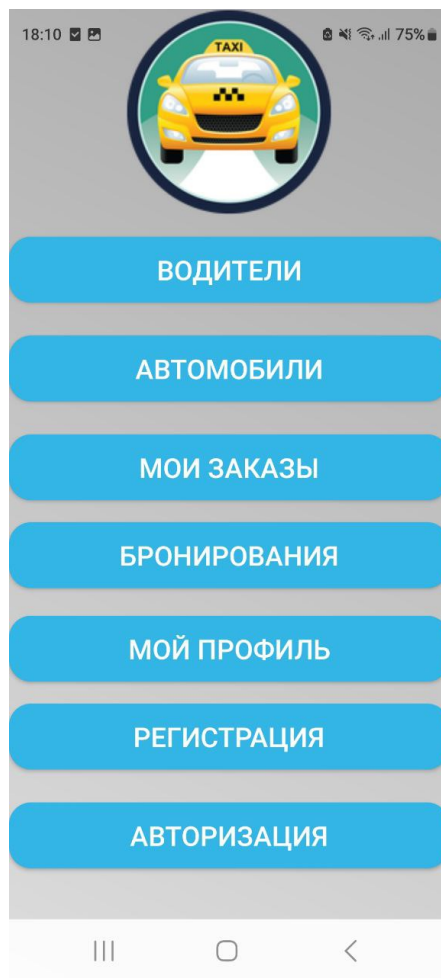


Рисунок 1 – Основное окно приложения

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 33 |



Рисунок 2 – Просмотр сведений о водителях

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 34 |

Список заказов

СОЗДАТЬ ЗАКАЗ



Водитель: yaroslav
Место забора: ул. Первая
Место высадки: ул. Советская
Статус заказа: Отменен





Водитель: Driver8
Место забора: ул. Ленина д 9
Место высадки: ул.Гагарина д 4
Статус заказа: В ожидании

Водитель: max
Место забора: ggg
Место высадки: bgv
Статус заказа: В ожидании



Рисунок 3 – Список заказов клиента

18:13

 75%

Создание заказа

Выберите водителя

max

Пункт отправления

Пункт назначения

СОХРАНИТЬ ЗАКАЗ

|||

○

<

Рисунок 4 – страница для создания заказа

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 36 |

Список бронирований

НОВОЕ БРОНИРОВАНИЕ

Номер бронирования: 2
Точка отправления:
Санкт-Петербург, Невский
проспект
Точка прибытия:
Санкт-Петербург, Эрмитаж
Статус: Confirmed

Номер бронирования: 6
Точка отправления: ул.
Гагарина д 3
Точка прибытия: ул.Ленина д
7
Статус: Подтвержденный



Рисунок 5 – Список бронирований клиента

Страница бронирований

Пункт отправления

Пункт назначения

513AM

6:14PM

715

СОЗДАТЬ БРОНИРОВАНИЕ

|||

○

<

Рисунок 7 – страница для добавления бронирования

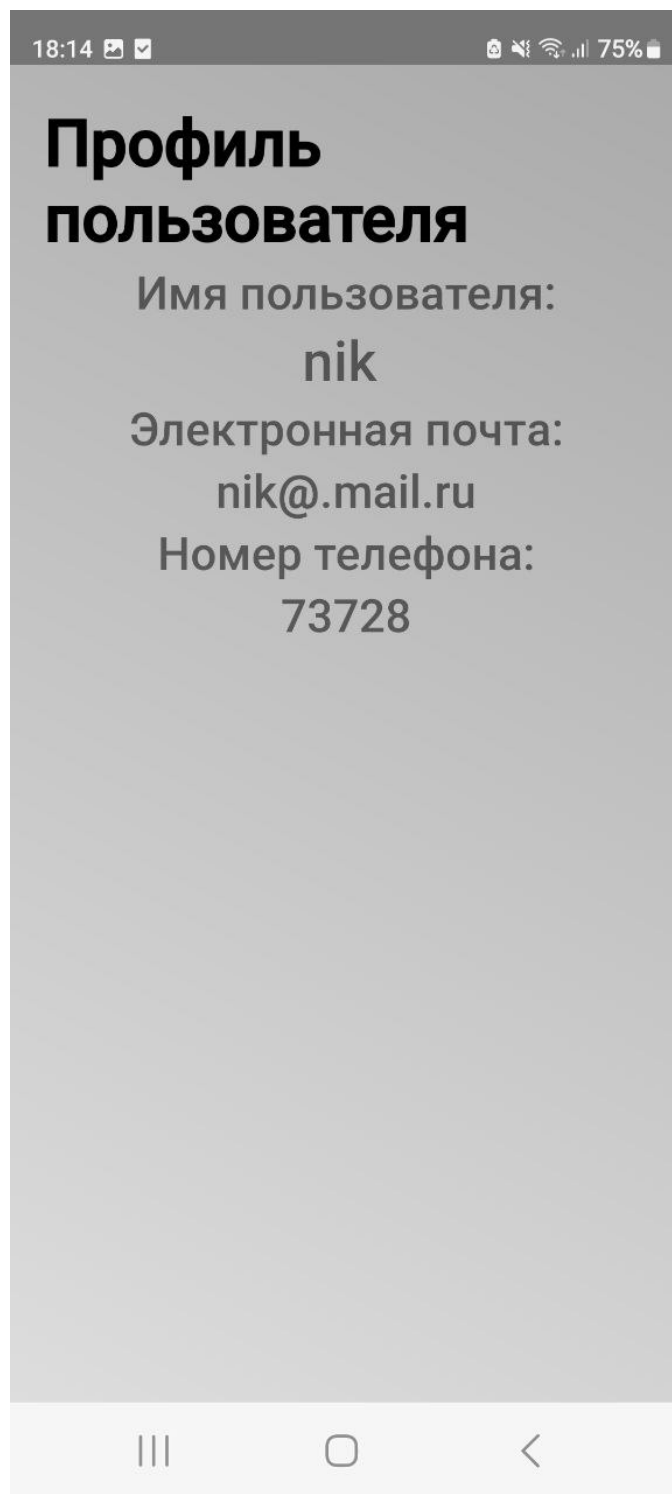


Рисунок 8 – просмотр данных пользователя

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 39 |

18:15 75%

Регистрация

Логин

Почта

Номер телефона

Пароль

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 9 – страница регистрации

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 40 |

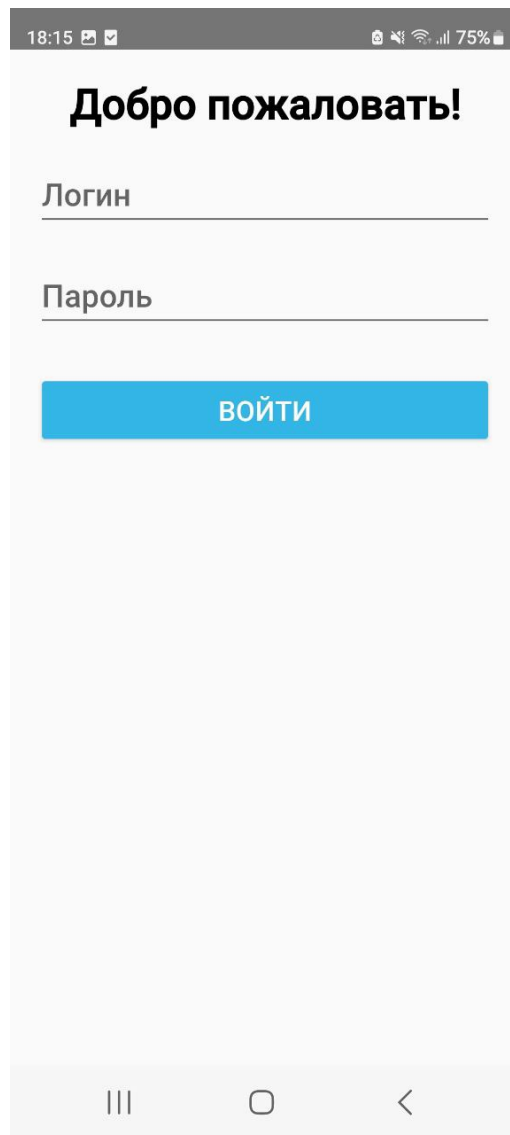


Рисунок 10 – страница авторизации

| | | | | | | |
|------|-----|----------|---------|-----|-------------------------|-----|
| | | | | | МИВлГУ 09.03.04 – 0.015 | Лис |
| Изм. | Лис | № докум. | Подпись | Дат | | 41 |