МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

**«Мобильное приложение покупки/продажи авто»**

Выполнил студент Бич Павел Валерьевич

(Ф.И.О.)

Руководитель проекта асс. Нистюк О. А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

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

Консультант: асс. Нистюк О. А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс. Нистюк О. А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2023

**Содержание**

ВВЕДЕНИЕ 3

1. Постановка задачи и аналитический обзор литературы 4

1.1 Приложение «av.by» 4

1.2 Приложение «Drom.ru» 5

1.3 Выводы по главе «Обзор аналогов приложений» 7

2. Проектирование информационной системы 8

2.1 Локальная и глобальная база данных 8

2.2 Проектирование мобильного приложения 14

2.3 Проектирование веб-сервера 15

2.4 Вывод по главе «Проектирование информационной системы» 16

3. Разработка информационной системы 17

3.1 Разработка базы данных 17

3.2 Разработка мобильного приложения 18

3.3 Разработка веб-приложения 19

3.4 Вывод по главе «Разработка информационной системы» 21

4. Обеспечение безопасности информационной системы 22

Заключение 23

Список использованных источников 24

Приложение А 25

Приложение B 26

# ВВЕДЕНИЕ

В настоящее время мобильные приложения играют огромную роль в повседневной жизни людей. Они упрощают нашу жизнь, делают ее более комфортной и эффективной. Одной из областей, которая активно использует мобильные приложения, является автомобильный рынок. Создание мобильного приложения для продажи и покупки автомобилей является актуальной темой в связи с растущим числом людей, которые предпочитают покупать и продавать автомобили через мобильные устройства.

В данной курсовой работе мы рассмотрим процесс создания мобильного приложения для продажи и покупки автомобилей, используя серверную часть, написанную на Node.js, Android-приложение, созданное на Java, и базу данных MySQL. Мы также изучим основные принципы взаимодействия между сервером и клиентским приложением и рассмотрим способы обеспечения безопасности данных в приложении. Кроме того, мы рассмотрим способы взаимодействия приложения с базой данных MySQL, включая создание таблиц, запросы к базе данных и работу с данными на сервере. Изучим основные принципы безопасности при работе с базой данных и применим их в нашем приложении.

Мы также изучим примеры успешных приложений для продажи и покупки автомобилей и проанализируем их особенности. В результате нашей работы мы получим практические рекомендации по созданию мобильного приложения для продажи и покупки автомобилей, которые помогут нам создать полноценное и удобное приложение для пользователей.

# Постановка задачи и аналитический обзор литературы

## 1.1 Приложение «av.by»

Важным этапом создания различных проектов является ознакомление с разных литературных источников и аналитический разбор.

Были проанализированы несколько разных программных средств связанных с моей тематикой.

Av.by - это популярный интернет-ресурс, предоставляющий услуги по продаже и покупке автомобилей в Беларуси. Сайт содержит огромную базу объявлений, которые можно фильтровать по различным параметрам, таким как марка, модель, год выпуска, тип топлива, цена и другие. Пользователи могут создавать личные кабинеты для сохранения своих предпочтений и отслеживания интересующих объявлений.

Кроме того, на сайте можно получить информацию об автосалонах, кредитовании, страховании, оформлении документов и других вопросах, связанных с автомобилями. Av.by имеет мобильные приложения для iOS и Android, что делает поиск и покупку автомобилей еще более удобными.

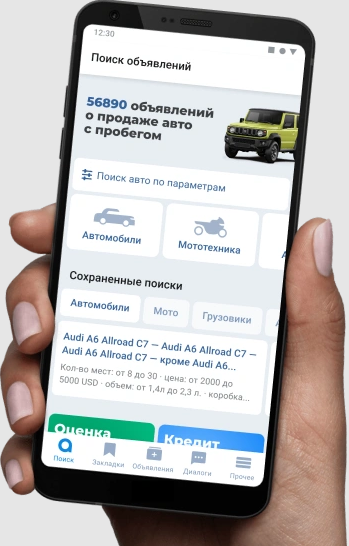


Рис. 1.1 – Мобильное приложение.

Данное приложение так же имеет и десктопную версию, которая показана на рисунке ниже (рисунок 1.2)

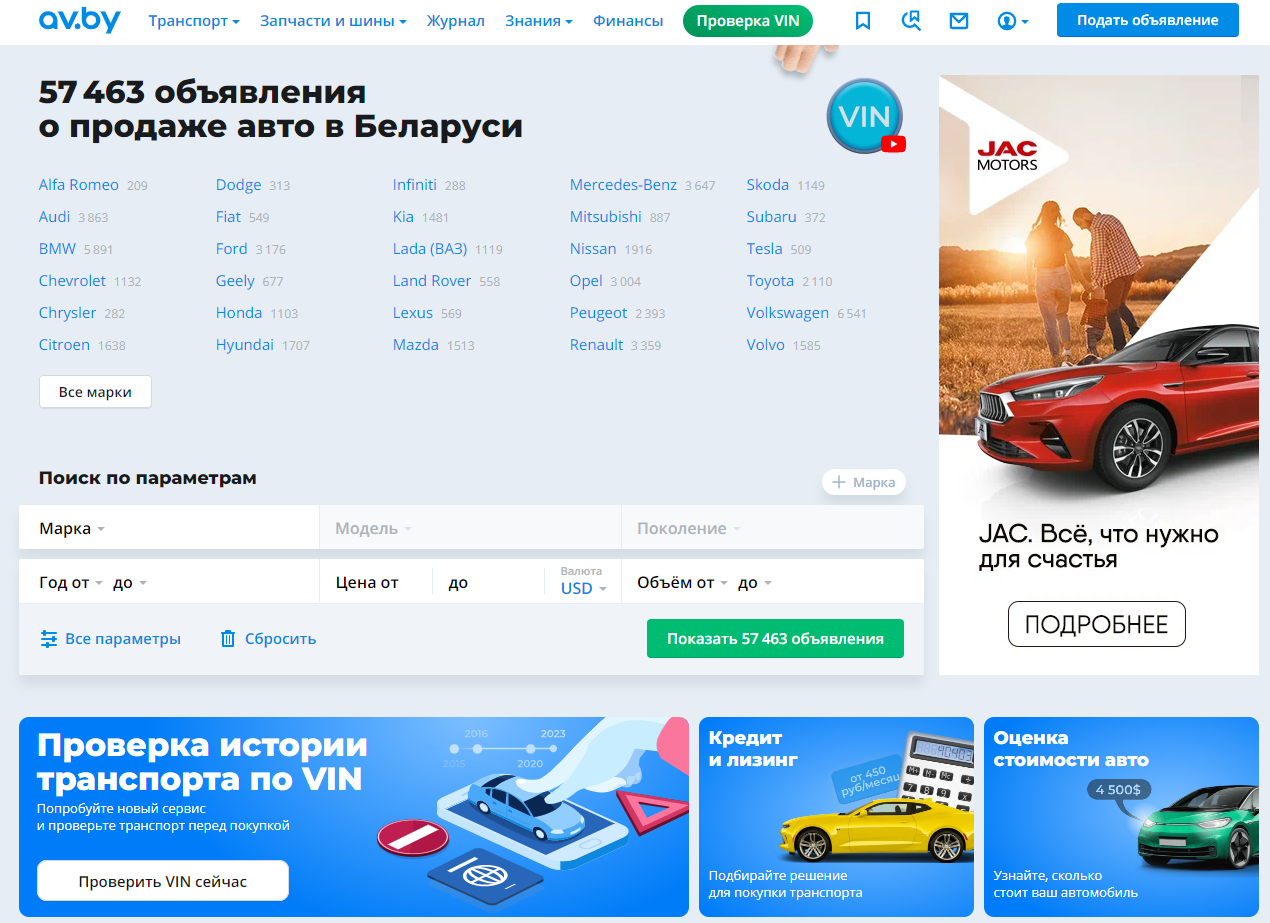


Рис. 1.2 – Версия для компьютеров.

Дизайн выполнен аналогично мобильной версии, в той же цветовой палитре, однако в десктопной версии присутствует реклама. Именно с этого приложения пришло вдохновение на создание своего.

## 1.2 Приложение «Drom.ru»

Приложение «Drom».

Drom.ru - это один из крупнейших автомобильных порталов в России, который позволяет пользователям покупать и продавать автомобили как новые, так и б/у, а также заказывать услуги по ремонту, тюнингу и обслуживанию автомобилей. На сайте можно найти широкий выбор автомобилей от разных производителей и в разных ценовых категориях, а также прочитать новости и обзоры автомобильного мира. Сайт предлагает удобный поиск по различным параметрам, фильтрацию по географическому местоположению и функции сравнения автомобилей. Кроме того, на Drom.ru можно найти информацию о технических характеристиках и истории эксплуатации конкретного автомобиля, а также связаться с продавцом напрямую через сервис.

Далее будут рассмотрены некоторые страницы данного приложения (Рис. 1.3 и 1.4)

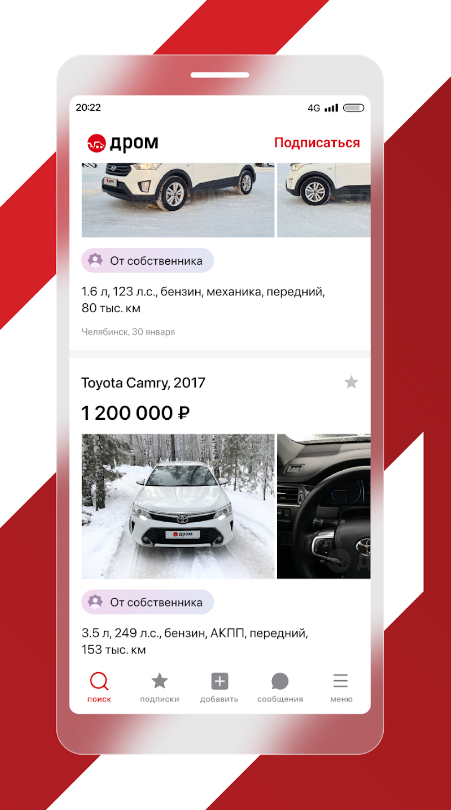


Рис. 1.3 – Drom – дизайн под телефон.

Как можно заметить десктопная версия значительно уступает мобильной – сайт хоть и соответсвует мобильной версии по цветовой палитре, но сделан достаточно скучно.

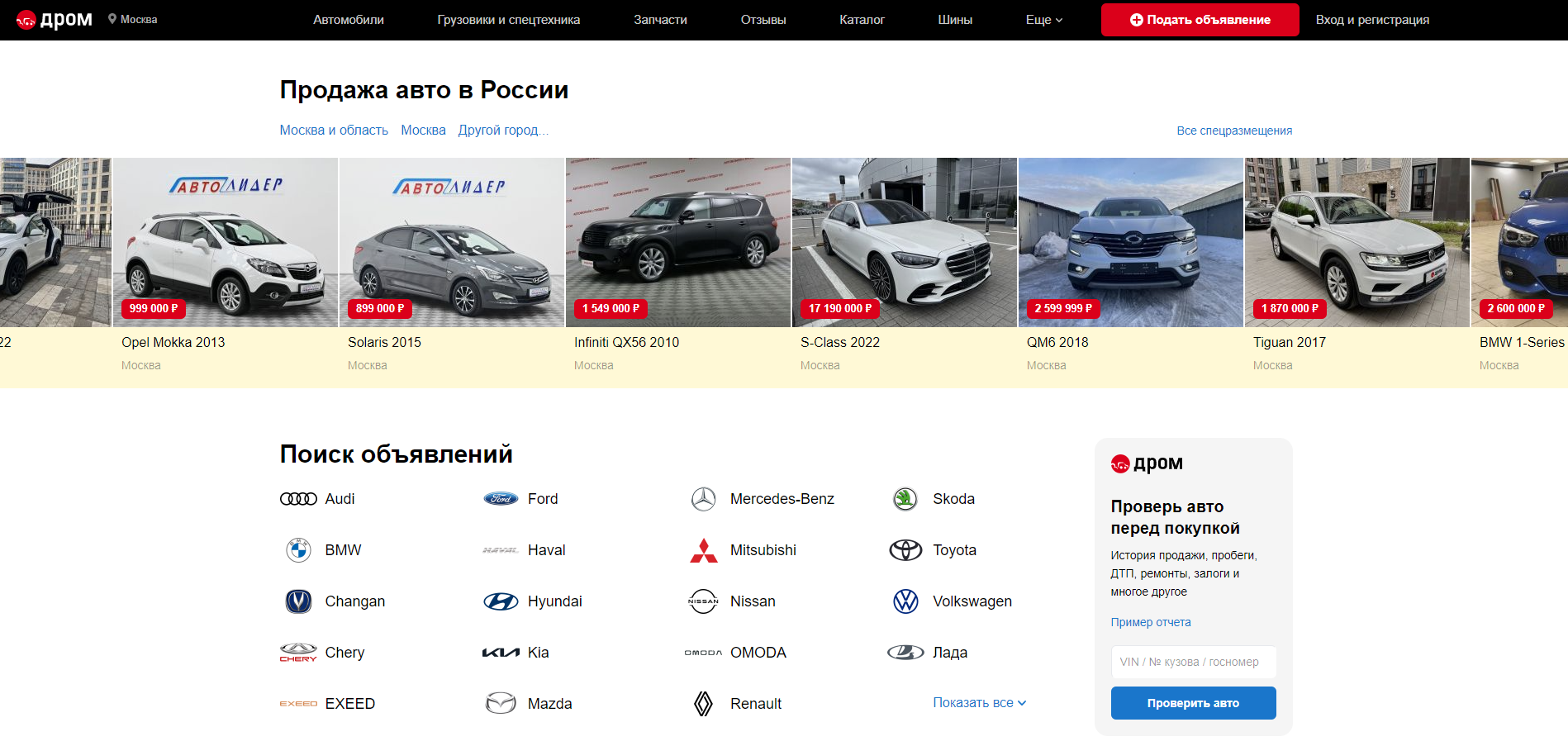


Рис. 1.4 – Версия Drom для компьютеров.

Мое же приложение будет ориентировано на белорусский авто-рынок, чем будет похоже на «av.by» и отличаться от «Drom»

## Выводы по главе «Обзор аналогов приложений»

В данной главе были показаны два популярных приложения, которые являются аналогами приложения «Carvitta»: «av.by» и «Drom.ru». В результате подытожим сравнения.

Av.by позволяет с легкостью смотреть объявления и выбирать автомобили, легкая навигация позволяет быстро связаться с владельцем авто на белорусском авто-рынке. Drom.ru же направлен на российский авто-рынок. Несмотря на особенности каждого приложения, оба из них предлагают высокое качество обслуживания. Выбор между ними зависит только от предпочтений самого пользователя, а также страны проживания.

В течение анализа были выявлены слабые и сильные стороны обоих приложений, реализация некоторых решений будет представлена также и в версии Carvitta.

# Проектирование информационной системы

## 2.1 База данных

Задачей курсового проекта создать приложение, которое будет иметь заданный функционал:

* регистрация\авторизация пользователей
* просмотр всех существующих авто
* добавление объявления о продаже
* просмотр подробной информации объявления
* возможность убрать объявление о продаже из поиска
* личный кабинет пользователя с краткой информацией

В базе данных данного приложения созданы для использования 2 таблицы. Скрипт создания БД и таблиц находится в Приложении А. Ниже представлена их характеристика.

Схема базы данных представлена в таблице 2.1 – 2.2.

**Таблица 2.1- схема таблицы *users.***

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Типы данных** | **Назначение** |
| id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| login | Text | Логин пользователя |
| name | Text | Имя пользователя |
| password | Text | Пароль |
| isAuth | Integer | Переменная, отвечающая за проверку: авторизован ли юзер |

В таблице «users» содержится информация для авторизации пользователей. Столбцы данной таблицы:

* id – уникальный идентификатор(автоинкрементный), целочисленный тип данных, является первичным ключом;
* login – строковый тип данных;
* name – строковый тип данных
* password – строковый тип данных, хешируется на сервере и на мобильном приложении с помощью SHA-256
* isAuth –целочисленный тип данных, нужен чтобы приложение понимало, что пользователь уже производил авторизацию и сразу пускало на главную страницу.

**Таблица 2.2 – схема таблицы *cars*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| usid | Integer | Уникальный идентификатор владельца объявления |
| marka | Text | Марка авто |
| model | Text | Модель авто |
| price | Text | Цена |
| year | Text | Год выпуска |
| mehauto | Text | Коробка передач (Механика/автомат) |
| volume | Text | Объем двигателя |
| typefuel | Text | Тип топлива |
| shape | Text | Кузов |
| distance | Text | Пробег |
| img | Text | Первая картинка |
| img2 | Text | Вторая картинка |
| phone | Text | Моб. телефон |
| issell | Tinyint(1) | Продано (1 – да / 0 -нет ) |

В таблице «***cars»*** содержатся объявления. Столбцы данной таблицы:

* id – первичный ключ, который автоматически генерируется для каждой записи;
* usid – целочисленный тип, хранящий идентификатор владельца объявления;
* marka – строковый тип данных, хранящий марку автомобиля;
* model – строковый тип данных, который хранит модель авто;
* price – строковый тип данных для хранения цены авто
* year – строковый тип данных для хранения года выпуска авто
* mehauto – строковый тип данных, хранящий тип трансмиссии
* volume – строковый тип данных, используется для хранения объема двигателя
* typefuel – строковый тип данных, используется для хранения типа топлива
* shape – строковый тип данных, используется для хранения типа кузова
* distance – строковый тип данных, используется для хранения пробега автомобиля
* img – строковый тип данных, используется для хранения путя к первой картинке
* img2 – строковый тип данных, используется для хранения путя ко второй картинке
* phone – строковый тип данных, используется для хранения мобильного номера владельца авто
* issell – целочисленный тип данных, используется для определения: продано авто или нет

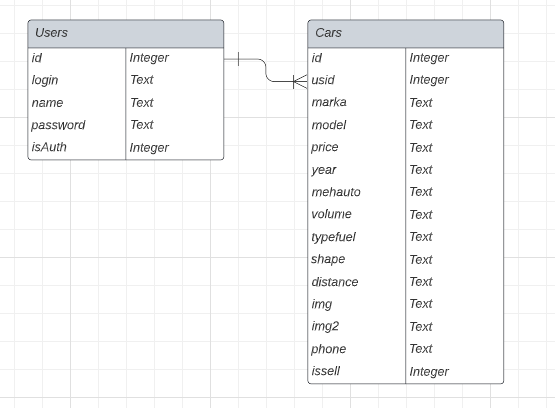
****

Рис. 2.4 – структура базы данных

В таблице users будут все данные о зарегистрированных пользователях: их логин, имя, пароль, авторизован ли.

В таблице cars содержатся объявления и информация о них.

## 2.2 Проектирование мобильного приложения

В данной курсовой работе необходимо было разработать приложение, позволяющее пользователям просматривать объявления о продажах авто, выбирать понравившееся и добавлять свои. Изображение 2.5 показывает страницу загрузки приложения, которая встречает каждого пользователя при входе в приложение.

****

Рис. 2.5 – Страница загрузки

После этого пользователя перекидывает на страницу авторизации, где представлена форма для ввода данных (Рис. 2.6). Здесь пользователь может ввести логин и пароль. Также предусмотрена валидация данных например: нельзя оставить поля пустыми (Рис 2.7), идет проверка на существующий логин (Рис. 2.8), проверка на соответствие логина и пароля(Рис. 2.9). Если же все было введено верно, то приложение перебросит юзера на главную страницу с поиском авто(Рис. 2.11).

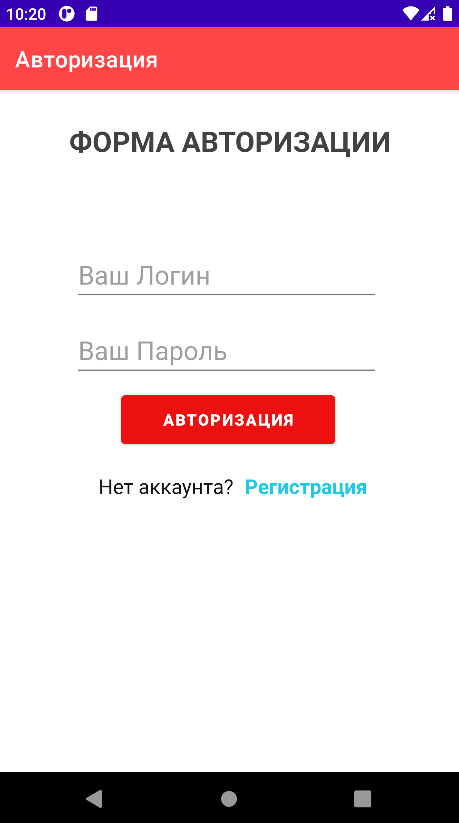


Рис. 2.6. – Страница авторизации

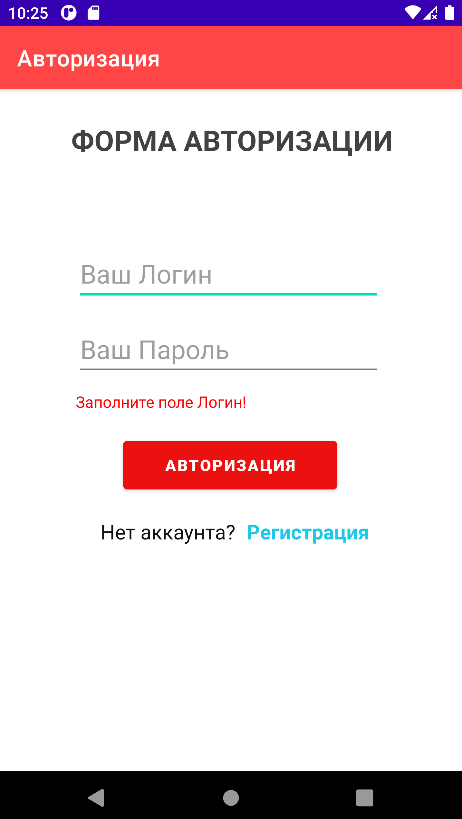


Рис 2.7 – Валидация пустых полей



Рис. 2.8 – Валидация несуществующего логина

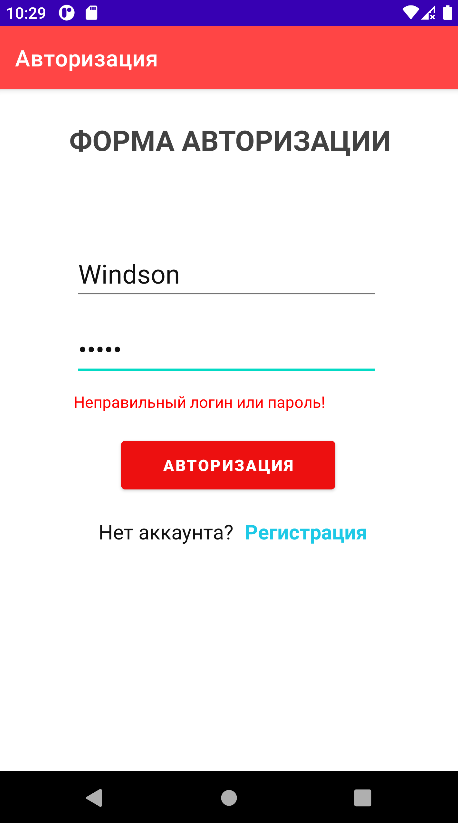


Рис 2.9 – Валидация неверных логина и пароля

Если юзер не зарегистрирован в приложении, то существует форма регистрации (Рис. 2.10), на которую можно попасть со страницы авторизации, щелкнув на кнопку «Регистрация» под кнопкой «Авторизация».

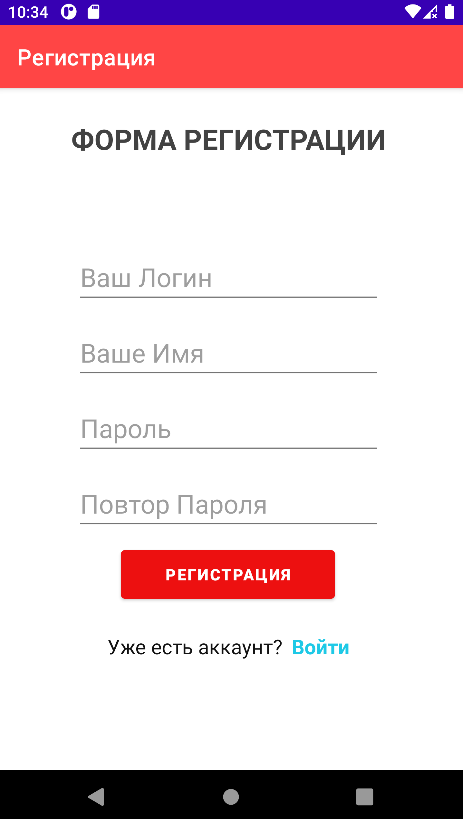


Рис 2.10 – Страница создания аккаунта

Пользователь необходимо ввести логин, имя и пароль с повторением, который должен быть не менее 6 символов. При введении пользователем некорректной информации будут выданы соответствующие сообщения. После успешной пройденной регистрации пользователь переходит на страницу авторизации (Рис. 2.6)

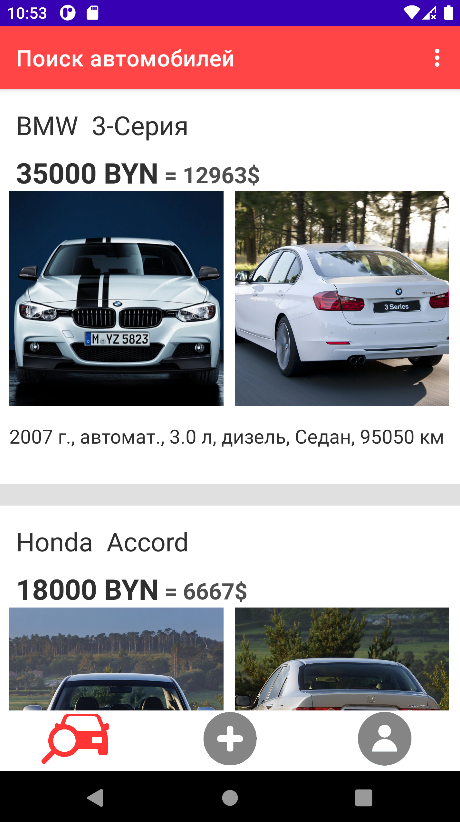


Рис 2.11 – Страница после авторизации

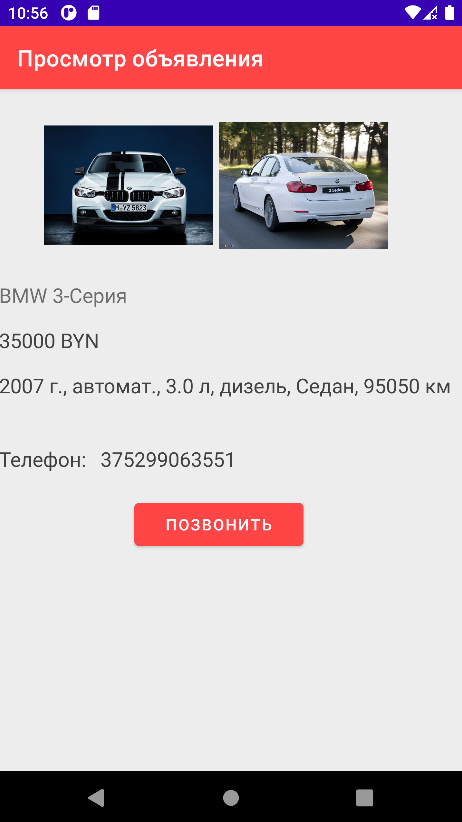


Рис. 2.12 – Страница объявления

Пользователь, нажав на иконку, перейдет на страницу объявления(Рис. 2.12). На странице объявления пользователь просмотреть подробную информацию о машине и перейти в телефонный справочник для звонка по номеру по кнопке «ПОЗВОНИТЬ». Нажав в навигационном меню на иконку человека, пользователь переходит на страницу юзера(Рис. 2.13), в которой может просмотреть свой логин, имя и пароль, а также перейти на страницу со своими объявлениями из аккаунта. Зажав на своем объявлении появляется всплывающее меню, после нажатия на кнопку продано, объявление принимает немного другой вид(Рис. 2.14) и из общего поиска пропадает, оставаясь только в объявлениях пользователя.

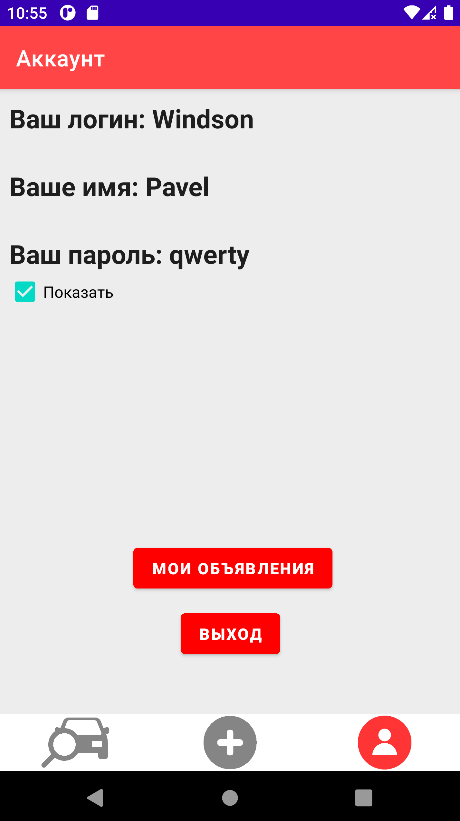


Рис 2.13 – Страница аккаунта юзера

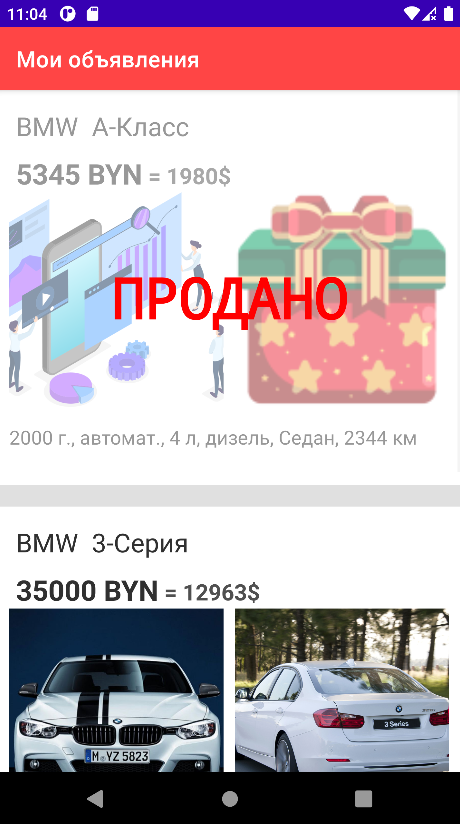


Рис 2.14. – Страница мои объявления

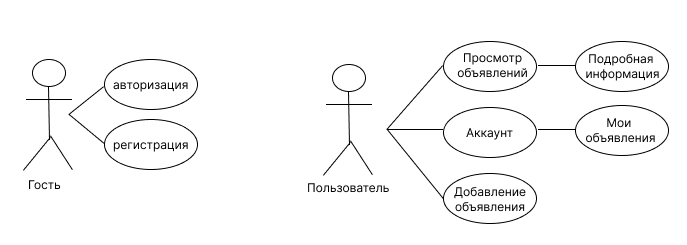


Рис. 2.15 – диаграмма использования мобильного приложения для пользователей

## 2.3 Проектирование веб-сервера

Веб-приложение - клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

Веб приложение состоит из двух компонентов: веб сервера и клиента. Веб. Веб сервер имеет REST архитектуру и осуществляет взаимодействие с базой данных. Сервер содержит в себе реализованные сервисы для взаимодействия с каждой сущностью базы данных.

При обращении к серверу по заданному маршруту и определенному HTTP методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

При помощи HTTP метода GET сервер будет возвращать данные клиенту, ссылаясь на аргументы в запросе, как на условие для выборки данных. Метод POST в свою очередь позволяет передать данные на сервер для последующей обработки.

Веб-приложение состоит их двух частей: клиент и сервер. Клиент в свою очередь выступает в роле интерфейса веб-приложения.

## Вывод по главе «Проектирование информационной системы»

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения и описаны варианты использования программного средства.

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

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между сервером и клиентом.

# Разработка информационной системы

Для разработки приложений курсового проекта были использованы следующие технические средства:

MySQL - реляционная база данных.

SQL – декларативный язык программирования, применяемый для создания, модификации, и управления данными в РБД.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Visual Studio Code – редактор исходного кода.

Java –язык программирования.

Node.js — это платформа с открытым исходным кодом для работы с языком JavaScript.

SmarterASP.net – хостинг базы данных

## Разработка базы данных

Для хранения данных, поступающих из приложения, использована MySQL.

Также были созданы хранимые процедуры, для взаимодействия с таблицами базы данных.

/addCar

BEGIN

INSERT INTO cars (

usid,

marka,

model,

price,

year,

mehauto,

volume,

typefuel,

shape,

distance,

img,

img2,

phone,

issell

)

VALUES (

usid,

marka,

model,

price,

year,

mehauto,

volume,

typefuel,

shape,

distance,

img,

img2,

phone,

issell

);

END

/addUser

BEGIN

INSERT INTO users(login, name, password, isAuth) VALUES (login, name, password, isAuth);

END

/getAllCars

BEGIN

SELECT \* FROM cars WHERE issell = p\_issell;

END

/getAllUsers

BEGIN

SELECT \* FROM users;

END

/auth

BEGIN

SELECT id, login, name, password, isAuth FROM users WHERE login = p\_login AND password = p\_password;

END

/exit0

BEGIN

UPDATE users SET isAuth = 0 WHERE login = p\_login;

END

Листинг 1 – примеры хранимых процедур

## 3.2 Разработка мобильного приложения

Мобильное приложение было реализовано в среде разработки – Android Studio. Для написания приложения был использован язык программирования Java.

Графическая структура проекта представлена на рисунке 3.

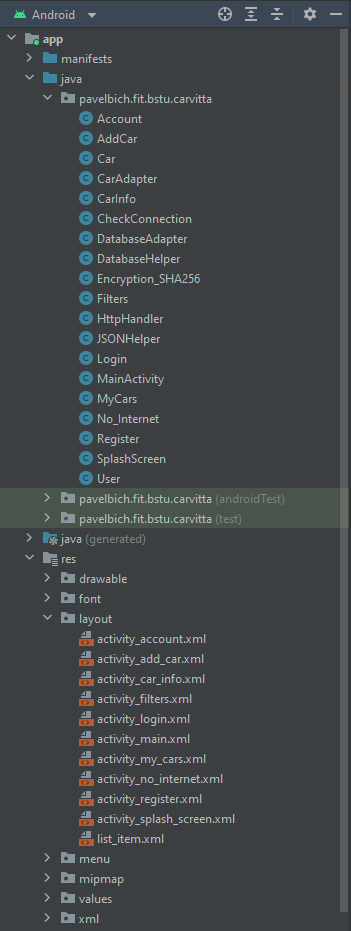


Рис. 3 – Графическая структура проекта

С помощю пакет http приложение асинхронно отправляет запросы к серверу (листинг 2)

new Thread(new Runnable() {

@Override

public void run() {

try {

httpHandler.post("http://10.0.2.2:5559/addCar",json.toString());

synchronized (lock){

lock.notify();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}).start();

synchronized (lock){

try{

lock.wait();

}catch(InterruptedException e){

e.printStackTrace();

}

}

Листинг 2 – обращение к серверу со стороны клиента

В Android Studio,в приложении, работающее на языке Java, нужно выполнять запросы к серверу в отдельных потоках, если не следовать данному правилу, то запрос будет блокировать основной поток и приложение крашнется. Чтобы этого избежать ставим код запроса в отдельный поток, затем основной поток Java ставим в режим ожидания и ждем, когда выполнится запрос в отдельном потоке.

Сам запрос происходит с помощью обращения к объекту httpHandler (листинг 3), в нем определен метод post, принимающий 2 параметра (url – адрес к серверу и тело запроса)

public class HttpHandler {

public static String post(String url, String postData) {

HttpURLConnection urlConnection = null;

BufferedReader reader = null;

String result = null;

try {

URL requestUrl = new URL(url);

urlConnection = (HttpURLConnection) requestUrl.openConnection();

urlConnection.setRequestMethod("POST");

urlConnection.setDoOutput(true);

urlConnection.setRequestProperty("Content-Type", "application/json");

OutputStream outputStream = urlConnection.getOutputStream();

outputStream.write(postData.getBytes());

outputStream.flush();

InputStream inputStream = urlConnection.getInputStream();

StringBuilder buffer = new StringBuilder();

if (inputStream == null) {

return null;

}

reader = new BufferedReader(new InputStreamReader(inputStream));

String line;

while ((line = reader.readLine()) != null) {

buffer.append(line + "\n");

}

if (buffer.length() == 0) {

return null;

}

result = buffer.toString();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (urlConnection != null) {

urlConnection.disconnect();

}

if (reader != null) {

try {

reader.close();

} catch (final IOException e) {

e.printStackTrace();

}

}

}

return result;

}

}

Листинг 3 – класс httpHandler

## 3.3 Разработка веб-приложения

Веб-сервер разработан с использование языка программирование JavaScript и среды исполнения NodeJS. NodeJS — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Веб сервер разработан с использованием архитектуры REST.(Рис. 3.1)



Рис. 3.1 – структура веб-сервера

В общем случае обращение к бд выглядит вызов конкретной процедуры и передача параметров, если они нужны (Листинг 4)

case "/addCar":

if(request.method === "POST" && request.url === "/addCar"){

let body1 ='';

request.on('data',data1 =>{

body1+=data1.toString();

});

request.on('end',()=>{

const params = JSON.parse(body1);

var usid = params.p\_usid;

var marka = params.p\_marka;

var model = params.p\_model;

var price = params.p\_price;

var year = params.p\_year;

var mehauto = params.p\_mehauto;

var volume = params.p\_volume;

var typefuel = params.p\_typefuel;

var shape = params.p\_shape;

var distance = params.p\_distance;

var img1 = params.p\_img;

var img2 = params.p\_img2;

var phone = params.p\_phone;

var issell = params.p\_issell;

console.log(marka);

const paramssss = [usid,marka,model,price,year,mehauto,volume,typefuel,shape,distance,img1,img2,phone,issell]

connection.query('CALL addCar(?,?,?,?,?,?,?,?,?,?,?,?,?,?)',paramssss, (error,results) =>{

if (error) {

console.error('Error calling stored procedure: ', error);

return;

}

console.log('coonection', results);

})

response.writeHead(200,{'Content-Type':'text/html; charset=utf-8'});

response.end();

})

}

break

Листинг 4 - пример обработки запроса к серверу

## Вывод по главе «Разработка информационной системы»

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента.

Третьим этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрены некоторые классы и их методы, в частности.

Четвертый этап – это разработка мобильного приложения, где была представлена графическая структура веб-сервера и клиента, рассмотрены некоторые классы и их методы.

# 4. Обеспечение безопасности информационной системы

Для обеспечения защиты персональных данных пользователей применялись методы хеширования и криптографии. Пароли пользователей информационной системы хранятся в базе данных в хешированном виде. Хеширование – это процесс преобразования входной строки переменной длины в выходную строку фиксированной длины, называемой хеш-кодом или дайджестом сообщения. Процесс хеширования реализуется путем использования криптографической хеш-функции (Приложение B). Хеширование паролей не позволяет злоумышленнику получить доступ к аккаунту пользователя информационной системы.

В данном приложении реализована функция хеширование пароля с помощью SHA-256. Приложение сравнивает два разных хеша и смотрит на совпадение. Если хеши совпадают, значит пароль введен правильно.

SHA-256 является более безопасным и надежным алгоритмом хеширования по сравнению с алгоритмом MD5. Основные отличия между этими алгоритмами заключаются в следующем: MD5 использует хеш-функцию с длиной хеша 128 бит, в то время как SHA-256 использует длину хеша в 256 бит. Это означает, что хеш-код, полученный с помощью SHA-256, намного более устойчив к коллизиям (когда два разных сообщения дают одинаковый хеш-код) и к взлому. SHA-256 использует более сложный алгоритм хеширования, который включает в себя большее количество раундов и более сложные математические операции, чем MD5. Это делает SHA-256 более устойчивым к различным видам атак, таким как атаки с использованием словаря, атаки с перебором и т.д.

MD5 является более быстрым алгоритмом по сравнению с SHA-256. Однако, скорость работы не всегда является ключевым критерием при выборе алгоритма хеширования, потому что он должен быть безопасным и защищенным от взлома.

В целом, SHA-256 является более безопасным алгоритмом хеширования и рекомендуется использовать для целей безопасности данных, таких как хранение паролей, проверка целостности данных, создание цифровых подписей и т.д.

# Заключение

Целью моего курсового проекта было предоставить пользователям возможность легко и удобно смотреть объявления и добавлять свои для покупки/продажи авто.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 2 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В приложении был определен основной функционал, также была разработана диаграмма вариантов использования. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных и HTTP методы для работы с серверной частью.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: PostgreSQL, сред разработки Android Studio и Visual Studio Code, языков программирования Java и JS, и программной платформы node.js.В качестве хостинга – SmarterAsp.net.

Была реализована База Данных.

При решении данной задачи была достигнута поставленная цель, и был создан проект «Carvitta», который включает в себя android-клиент, веб-клиент и веб-сервер.

# Список использованных источников

1. Шифрование SHA-256 [Электронный ресурс] – question-it.com – Режим доступа: https://question-it.com/questions/4526555/kak-heshirovat-stroku-s-pomoschju-sha256-v-java
2. Создание сервера на Node Js - metanit.com/web/nodejs/3.1.php
3. Создание хранимых процедур в БД https://bestprogrammer.ru/izuchenie/sozdanie-hranimyh-protsedur-v-mysql
4. Java, Android Studio - https://metanit.com/java/android/1.2.php

# 

# Приложение А

CREATE TABLE `cars` (

`id` int NOT NULL,

`usid` int NOT NULL,

`marka` text NOT NULL,

`model` text NOT NULL,

`price` text NOT NULL,

`year` text NOT NULL,

`mehauto` text NOT NULL,

`volume` text NOT NULL,

`typefuel` text NOT NULL,

`shape` text NOT NULL,

`distance` text NOT NULL,

`img` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`img2` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`phone` text NOT NULL,

`issell` tinyint(1) NOT NULL DEFAULT '0') ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `users` (

`id` int NOT NULL,

`login` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`name` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`password` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`isAuth` int NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

# 

# Приложение B

# public class Encryption\_SHA256 {

# public static String encrypt(String input) {

# try {

# // Создаем экземпляр объекта MessageDigest с использованием алгоритма SHA-256

# MessageDigest digest = MessageDigest.getInstance("SHA-256");

# // Преобразуем входную строку в массив байтов

# byte[] encodedHash = digest.digest(input.getBytes(StandardCharsets.UTF\_8));

# // Преобразуем хеш в строку в шестнадцатеричном формате

# StringBuilder hexString = new StringBuilder();

# for (byte b : encodedHash) {

# String hex = Integer.toHexString(0xff & b);

# if (hex.length() == 1) {

# hexString.append('0');

# }

# hexString.append(hex);

# }

# // Возвращаем полученную строку

# return hexString.toString();

# } catch (NoSuchAlgorithmException e) {

# e.printStackTrace();

# }

# return null;

# }

# }