



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Шаблоны программных платформ языка Джава

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Приложение «Каршеринговая компания»

Студент: Сидоров Станислав Дмитриевич

Группа: ИКБО-20-21

Работа представлена к защите _____ (дата) _____ /Сидоров С.Д./

Руководитель: старший преподаватель Зорина Наталья Валентиновна

Работа допущена к защите _____ (дата) _____ /Зорина Н.В./

Оценка по итогам защиты: _____

_____ / _____ /

_____ / _____ /

(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших
защиту)

М. РТУ МИРЭА. 2022 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ на выполнение курсовой работы

по дисциплине: Шаблоны программных платформ языка Джава
по профилю: Разработка программных продуктов и проектирование информационных систем

Студент: Сидоров Станислав Дмитриевич

Группа: ИКБО-20-21

Срок представления к защите: 17.05.2023

Руководитель: старший преподаватель Зорина Наталья Валентиновна

Тема Приложение «Каршеринговая компания»

Исходные данные: индивидуальное задание на разработку; документация по Spring Framework и JEE, документация по языку Java (версия не ниже 8); инструменты и технологии: JDK (не ниже 8), gitHub, IntelliJIDEA. Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

1. Провести анализ предметной области и формирование основных требований к приложению, 2. Обосновать выбор средств ведения разработки. 3. Разработать приложение с использованием фреймворка Spring, выбранной технологии и инструментария. 4. Провести тестирование приложения. 5. Оформить пояснительную записку по курсовой работе 6. Провести анализ текста на антиплагиат 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: [подпись] / Болбаков Р. Г. / , « 28 » 02 2023 г.

Задание на КР выдал: [подпись] / Зорина Н. В. / , « 28 » февраля 2023 г.

Задание на КР получил: [подпись] / Сидоров С.Д. / , « 28 » февраля 2023 г.

УДК 4.4

Сидоров С.Д. Приложение «Каршеринговая компания»/ **Курсовая работа** по дисциплине «Шаблоны программных платформ языка Джава» профиля «Разработка программных продуктов и проектирование информационных систем» направления профессиональной подготовки бакалавриата 09.03.04 «Программная инженерия» (2-ый семестр) / руководитель старший преподаватель Н.В. Зорина / кафедра ИиППО Института ИТ МИРЭА – с. 37, табл. 0, ист. 7

Целью работы является создание серверного программного приложения на тему «Каршеринговая компания».

В рамках работы осуществлен краткий анализ аналогов веб-приложения по выбранной тематики, разработано приложение с использованием фреймворка Spring, произведено тестирование приложения и проверка на антиплагиат.

Sidorov S.D. Application "Car-sharing company"/ **Coursework** on the subject of "Java Platform Programming Patterns" in the profile of "Software Product Development and Information Systems Design" of the Bachelor's degree program in "Software Engineering" (2nd semester). The paper is 37 pages long, contains 0 tables, and 7 sources, and was supervised by Senior Lecturer N.V. Zorina from the Department of Computer Science and Engineering at the Institute of Information Technology at MIEM.

The purpose of the work is to create a server-side software application on the topic of "Car-sharing company".

The work includes a brief analysis of web application analogs on the chosen topic, the development of an application using the Spring framework, testing of the application, and checking for plagiarism.

М. МИРЭА. Ин-т ИИ. Каф. ИиППО. 2023 г. Сидоров С.Д.

Аннотация

В курсовой работе описывалось создание интернет-ресурса, на тему «Каршеринговая компания». Работа содержит анализ предметной области разрабатываемого интернет-ресурса, создание веб-страниц интернет-ресурса с использованием технологий Spring Framework и тестирование разработанного приложения.

В введении обосновывается актуальность выбранной темы, определяется цель работы и задачи, подлежащие решению для её достижения, описываются объект и предмет исследования используемые методы и информационная база исследования, а также кратко характеризуется структура КР по разделам.

В основной части содержится материал, необходимый для достижения цели КР. Основная часть включает в себя общие сведения (в частности, наименование интернет-ресурса, перечисление прикладного программного обеспечения, необходимого для разработки и функционирования интернет-ресурса, а также названия языков и технологий, с помощью которых реализован интернет-ресурс), описание функционального назначения интернет-ресурса и его логической структуры, описание разработки и функций программного приложения, тестирование работы приложения. В заключении последовательно излагаются теоретические выводы, которые были сформулированы в результате выполнения данной курсовой работы.

Курсовая работа на 37 листах, содержит 14 рисунков, 7 использованных источников, 6 листингов.

The term paper describes the creation of an internet resource on the topic of "Car-sharing company". The work includes an analysis of the subject area of the developed internet resource, the creation of web pages of the internet resource using Spring Framework technologies, and testing of the developed application.

The introduction justifies the relevance of the chosen topic, defines the goal of the work and the tasks that need to be solved to achieve it, describes the object and subject of the research, the methods and information base of the research, and briefly characterizes the structure of the course paper by sections.

The main part contains the material necessary to achieve the goal of the course paper. The main part includes general information (in particular, the name of the internet resource, a list of application software necessary for the development and operation of the internet resource, as well as the names of languages and technologies used to implement the internet resource), a description of the functional purpose of the internet resource and its logical structure, a description of the development and functions of the software application, and testing of the application's operation.

The conclusion presents the theoretical conclusions that were formulated as a result of the completion of this course paper.

The course paper is 37 pages long and contains 14 figures, 7 used sources, and 6 listings.

СОДЕРЖАНИЕ

| | |
|---|----|
| Обозначения и сокращения | 7 |
| Введение | 8 |
| 1. СБОР И АНАЛИЗ ТРЕБОВАНИЙ К ВЕБ-ПРИЛОЖЕНИЮ | 9 |
| 1.1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ | 9 |
| ВЫВОДЫ К РАЗДЕЛУ 1 | 9 |
| 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА | 11 |
| 2.1 Проектирование веб-приложения | 11 |
| 2.2 Выбор средств и технологии ведения разработки | 14 |
| 2.3 Структура программного приложения | 15 |
| 2.4 Описание классов программного приложения | 18 |
| Выводы к разделу 2 | 28 |
| 3 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ | 29 |
| 3.1 Тестирование регистрации и авторизации | 29 |
| 3.2 Тестирование пользовательского функционала | 30 |
| 3.3 Тестирование функций администратора | 32 |
| Выводы к разделу 3 | 35 |
| ЗАКЛЮЧЕНИЕ | 36 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 37 |

Обозначения и сокращения

БД – база данных;

СУБД – система управления базами данных;

URL – унифицированный указатель ресурса;

http – протокол передачи данных.

Введение

В настоящее время темп жизни среднестатистического человека значительно увеличился по сравнению с предыдущими столетиями. Люди стали все больше ценить своё личное время и комфорт. Данный факт повлиял на обширное распространение различных сервисов, существование которых ранее невозможно было даже представить. В наше время стали доступны различные приложения для доставки продуктов и товаров, для аренды квартир и техники, а также множество других онлайн сервисов улучшающих повседневную жизнь и позволяющих тратить все меньше времени на закрытие бытовых потребностей. Одним из них является каршеринг. Каршеринг позволяет пользователям арендовать различные транспортные средства за поминутную оплату, что позволяет людям совершать повседневные поездки по цене такси с комфортом личного автомобиля.

Целью данной курсовой работы является разработка приложения “Каршеринговая компания” с использованием Spring Framework, JDK и IntelliJIDEA. Для упрощения разработки процесс был поделён на несколько частей:

1. Анализ предметной области разрабатываемого веб-приложения;
2. Выбор средств ведения разработки;
3. Разработать веб-приложение с использованием IntelliJ IDEA, GitHub, JDK, Spring Framework, JEE;
4. Провести тестирование приложения.

В результате приложение должно обладать функционалом, необходимым для управления элементами каршеринговой компании, а также логикой функционирования, соответствующей современным стандартам разработки приложений.

1. СБОР И АНАЛИЗ ТРЕБОВАНИЙ К ВЕБ-ПРИЛОЖЕНИЮ

1.1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной курсовой работе предметной областью является исследование веб-приложений на тему "Каршеринговая компания".

Каршеринговая компания позволяет пользователям получать доступ к автомобилю на короткий срок за определенную плату, что позволяет людям, не имеющим личный автотранспорт пользоваться удобным средством перемещения, при этом забирая расходы на обслуживание, ремонт, обновление и до оснащения автопарка на себя.

На данный момент существует большое количество различных каршеринговых компаний, каждая из которых обладает своими собственными особенностями в реализации базового функционала.

В качестве исследуемых аналогов были выбраны одни из самых популярных приложений, предоставляющих услуги каршеринга, такие как: "Яндекс Драйв", "Делимобиль", "BelkaCar". Данные приложения были проанализированы на предмет представляемого функционала. В ходе анализа были выделены определённые функции, присущие каждому из выше представленных сервисов, представляющие собой основу приложения "Каршеринговая компания", которые будут реализованы при дальнейшей разработке. Из них был составлен список услуг необходимы для предоставления услуг каршеринговой компании:

1. Регистрация новых пользователей;
2. Аренда транспортных средств;
3. Просмотр информации о совершенных ранее арендах;

ВЫВОДЫ К РАЗДЕЛУ 1

Исходя из вышеперечисленного можно сделать следующий вывод. Веб-приложение должно иметь регистрацию пользователей, старт и окончание аренды, добавление, изменение и удаление информации о транспортных

средствах, изменение данных о пользователях, а также предоставлять возможность работникам просматривать список аренд, изменять системные данные, тарифные планы, параметры доступа пользователей и категории транспортных средств.

2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

2.1 Проектирование веб-приложения

Для удобного хранения данных была использована СУБД PostgreSQL. Далее была разработана БД (Рисунок 2.1). В БД входит 10 таблиц:

1. **users** (таблица пользователей). Таблица хранит в себе персональные данные пользователя, роль, указатель на уровень и пароль. Колонка `snpassport` нужна для уникального идентифицирования пользователя, колонка `full_name` – полное имя пользователя, колонка `date_of_birth` – дата рождения пользователя, колонка `password` – зашифрованный пароль пользователя. Колонка `ID_level` хранит `id` текущего уровня пользователя.
2. **userlevel** (таблица уровней). Таблица, предоставляющая список уровней, которыми может обладать пользователь. Колонка `ID_level` является уникальным идентификатором записи, `level_name` содержит название уровня.
3. **grouplevel** (таблица групп). Таблица, предоставляющая список групп, к которым могут быть отнесены различные транспортные средства. Колонка `id_group` является уникальным идентификатором для каждой записи, `group_name` содержит название группы.
4. **Permission** (таблица разрешений) Таблица, содержащая информацию о уровне пользователя необходимом для управления данной группой транспортных средств. Колонка `id_permission` является уникальным идентификатором для каждой записи, `id_group` – `id` группы транспортных средств, `id_level` – `id` уровня пользователя необходимого для управления данной группой.
5. **Vehicle_model** (таблица моделей) Таблица, содержащая информацию о представленных в автопарке моделях транспортных средств. Колонка `id_model` – уникальный идентификатор каждой модели, колонка `model_name` содержит название модели.

6. Vehicle_brand (таблица брендов) Таблица, содержащая информацию о представленных в автопарке брендах транспортных средств. Колонка id_model – уникальный идентификатор каждой модели, колонка model_name содержит название модели.
7. Vehicle_name (таблица имен) Таблица, содержащая информацию о комбинациях моделей и брендов у транспортных средств присутствующих в автопарке. Колонка id_vehicle_name – уникальный идентификатор каждой комбинации, колонка id_brand – id бренда транспортного средства, id_model – id модели транспортного средства.
8. Vehicle_work_model (таблица рабочих моделей) Таблица, содержащая информацию о группах транспортных средств одной модели. Колонка id_vehicle_work_model – уникальный идентификатор каждой рабочей модели, колонка price_per_hour – цена транспортного средства в час, колонка model_photo_name – название отображаемой фотографии на странице поиска, колонка id_vehicle_name – id имени транспортного средства, колонка id_group – id группы, к которой принадлежит транспортное средство.
9. Vehicle (таблица транспортных средств) Таблица, содержащая информацию о каждом транспортном средстве автопарка. Колонка VIN – уникальный идентификатор каждого транспортного средства, колонки color, state, place содержат информацию о цвете, статусе и местоположении транспортного средства, колонка id_vehicle_work_model – id рабочей модели транспортного средства.
10. Rent (таблица аренд) Таблица, содержащая информацию о каждой аренде проведенной пользователями. Колонка id_rent содержит уникальный идентификатор каждой аренды. Колонки duration, starting_point, end_point, start_time, end_time содержат основную информацию об аренде такую, как длительность, местоположение старта, местоположение конца, время начала и время конца аренды.

Колонки VIN и snpassport, содержат уникальные идентификаторы транспортного средства и пользователя соответственно.

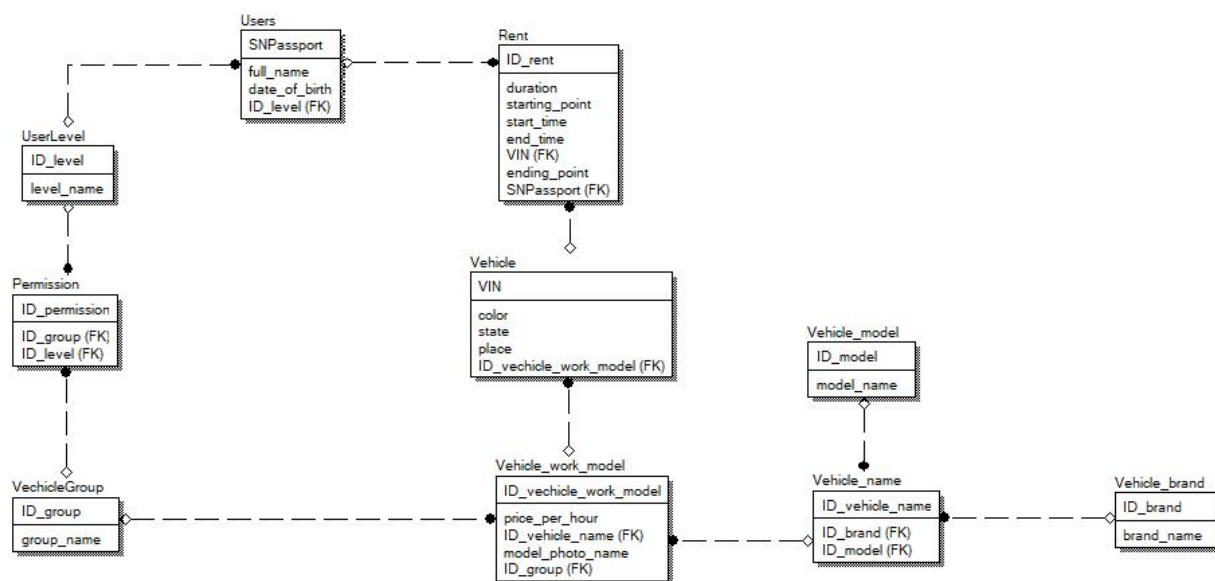


Рисунок 2.1 – Схема БД

2.2 Выбор средств и технологии ведения разработки

Для разработки в качестве IDE была выбрана IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java и разработана JetBrains. В качестве основного браузера использовался Opera. Для тестирования http запросов был использован Postman — платформа для совместной разработки API. Для работы с БД был использована СУБД PostgreSQL.

Основным языком программирования был выбран Java. Также были использованы фреймворки Spring Framework, Spring Security, Spring Boot. Для общения с БД была выбрана библиотека Hibernate и для удобной работы библиотека Lombok — java-библиотека, которая автоматически подключается к вашему редактору и инструментам сборки. Был использован Bootstrap для оформления сайта. А также использована система сборки Maven - фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML. Приложение работает на основе MVC-шаблона (Model, View, Controller). Это схема разделения данных приложения, UI и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

2.3 Структура программного приложения

Корневым пакетом программного приложения является `ru.mirea.SidorovSD` (Рисунок 2.2) и содержит следующие пакеты:

1. `controllers` – содержит классы контроллеров, предоставляющие данные пользователю.
2. `Models` – пакет, содержащий классы взаимодействующие с БД.
3. `Services` – содержит классы сервисов, реализующих внутреннюю бизнес-логику.
4. `Repos` – содержит интерфейсы организующие взаимодействие с БД.
5. `DTO` – содержит классы, представляющие данные для отправки или получения из вне.

Все файлы конфигурации располагаются в основном пакете.

Также папка `resources` вынесенная за основной пакет, содержит конфигурационный файл `application.properties`, а также папки `static` и `templates` необходимые для функционирования клиентской части.

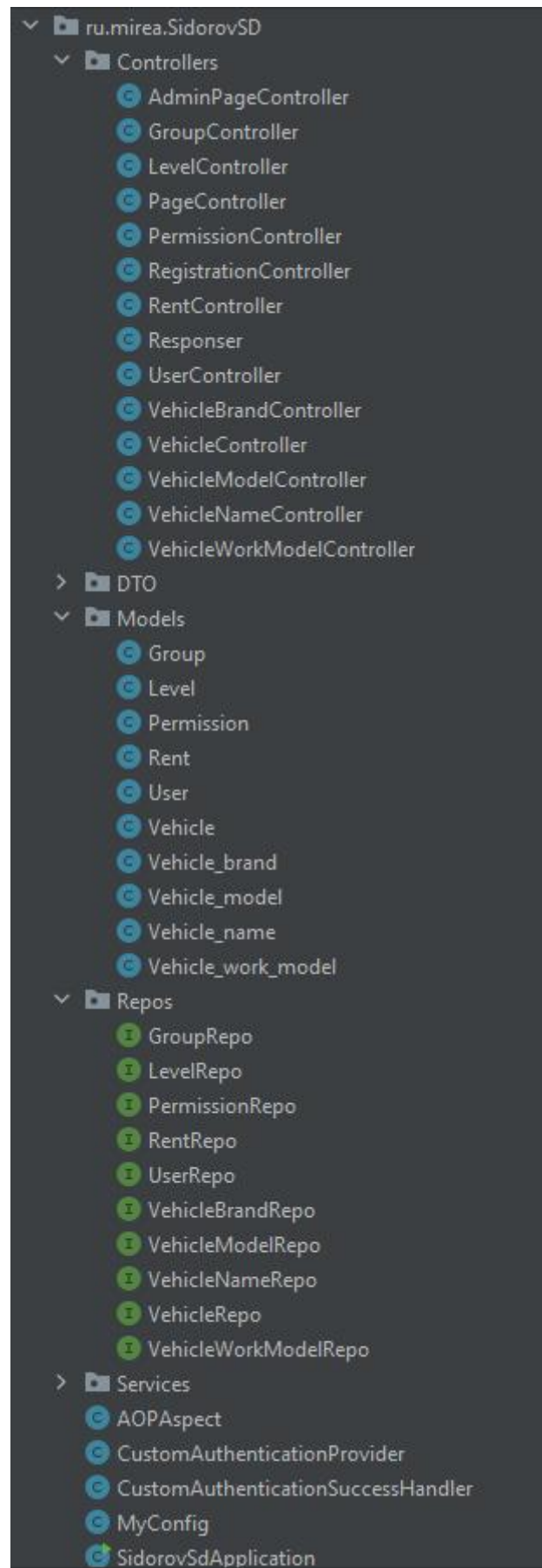


Рисунок 2.2 – Содержимое корневого пакета

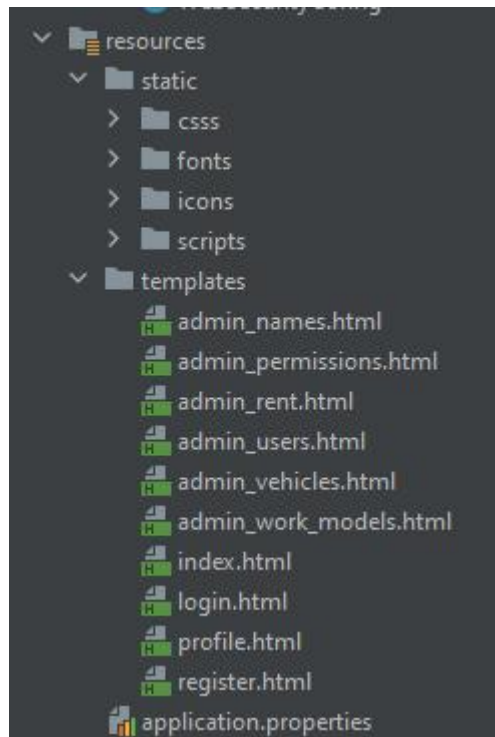


Рисунок 2.3 – Содержимое папка resources

2.4 Описание классов программного приложения

Изначально был сгенерирован пустой проект с помощью Spring Initializr, где были выбраны следующие библиотеки:

1. Spring Web;
2. Spring Data JPA;
3. Lombok;
4. Spring Security;
5. PostgreSQL Driver;
6. Thymeleaf;
7. Spring Boot.

Далее было решено начать с настройки взаимодействия между приложением и БД. Для достижения данной цели были созданы классы пакета Models. Каждый класс был помечен аннотацией `@Entity` для указания того, что конкретный класс является сущностью. Для связывания сущности с определенной таблицей в БД использовалась аннотация `@Table` с параметром `name` равным названию таблицы в БД. Также для автоматической генерации геттеров и сеттеров использовалась аннотация `@Data` фреймворка Lombok. Атрибуты класса были помечены аннотацией `@Column`, задав параметру `name` название колонки в БД, что позволило Hibernate присвоить нужное значение атрибуту. В качестве примера сущности был представлен класс User. (Листинг 2.1)

Листинг 2.1 – Фрагмент кода сущности User

```
@Entity
@Data
@Table(name = "users")
public class User {
    @Id
    @Column(name = "SNPassport")
    private String snpassport;
    @Column(name = "full_name")
    private String fullname;
    @Column(name = "date_of_birth")
    private String date_of_birth;
    @Column(name = "password")
    private String password;
    @Column(name = "username")
    private String username;
    @Column(name = "role")
    private String role;
    @JoinColumn(name = "id_level", referencedColumnName = "id_level")
    private int idLevel;}}
```

Для того чтобы получить данные из БД, необходимо создать интерфейс, расширяющий `JpaRepository`. Это позволяет вызывать нужный метод без необходимости постоянно прописывать SQL запросы. Интерфейсы были размещены в пакете `Repos` и помечены аннотацией `@Repository` для обнаружения Hibernate. Пример интерфейса представлен в Листинге 2.2.

Листинг 2.2 – Фрагмент кода UserRepository

```
@Repository
public interface UserRepo extends JpaRepository<User, Integer> {
    User findBySnpassport(String username);
    List<User> findByIdLevel(int idLevel);
    List<User> findByRole(String role);
}
```

Далее был создан сервисный слой для реализации бизнес-логики и взаимодействия с данными из БД. Файлы сервисов были помещены в пакет `Services` и были помечены аннотацией `@Service`. Пример сервиса изображен в Листинге 2.3.

Листинг 2.3 – Фрагмент кода `UserService`

```
@Service
@Slf4j
public class UserService {
    @Autowired
    private final UserRepo userRepo;
    @Autowired
    private final LevelRepo levelRepo;
    @Autowired
    private final RentRepo rentRepo;
    public UserService(UserRepo userRepo, LevelRepo levelRepo, RentRepo rentRepo) {
        this.userRepo = userRepo;
        this.levelRepo = levelRepo;
        this.rentRepo = rentRepo;
    }
    public List<User> getAll(){
        return userRepo.findAll();
    }
    public List<User> getAllByLevel(int idLevel){
        return userRepo.findByIdLevel(idLevel);
    }

    public List<User> getAllByRole(String role){
        return userRepo.findByRole(role);
    }
}
```

Продолжение Листинга 2.3

```
public User findBySnpassport(String snpassport) { return
userRepo.findBySnpassport(snpassport);}

public void saveUser(User user){
    if (isUserExist(user.getSnpassport()))
        return null;
    String encodedPassword = new BCryptPasswordEncoder().encode(user.getPassword());
    user.setPassword(encodedPassword);
    user.setRole("USER");
    user.setIdLevel(1);
    userRepo.save(user);}

public String changeUserInfo(String snpassport, String fullname, String dateOfBirth, String
password, String username, String role, int idLevel){
    User user = userRepo.findBySnpassport(snpassport);
    if (user == null) return "User doesn't exist";
    if (!fullname.equals("-")) user.setFullname(fullname);
    if (!dateOfBirth.equals("-")) user.setDate_of_birth(dateOfBirth);
    if (!password.equals("-")) user.setPassword(password);
    if (!username.equals("-")) user.setUsername(username);
    if (!role.equals("-")) user.setRole(role);
    if (idLevel != -1){
        if (levelRepo.findByIdLevel(idLevel) != null)
            user.setIdLevel(idLevel);
        else return "Level dosen't exist";}
    userRepo.save(user);
    return "OK";}

public String deleteUser(String snpassport){
    User user = userRepo.findBySnpassport(snpassport);
    if (user == null) return "User doesn't exist";
    if (!rentRepo.findBySnpassport(snpassport).isEmpty())
        return "User in use";
    userRepo.delete(user);
    return "OK";
}}
```

После разработки логики взаимодействия с данными был создан механизм взаимодействия между приложением и пользователем. Этот

процесс осуществляется через классы-контроллеры, которые реализуют логику приложения в ответ на URL-запрос из вне.

Для создания контроллера класс был помечен аннотацией `@Controller`. Методы контроллера аннотируются с помощью `@RequestMapping`, при которой устанавливается URL-адрес и тип HTTP-метода (GET, POST, DELETE, PATCH). Для удобства, для каждого типа HTTP-метода существует своя аннотация, например, `@GetMapping("/URL")`, которая соответствует `@RequestMapping(value="/URL", method = RequestMethod.GET)`. Методы также принимают параметры запросов, например с помощью `@RequestBody` или `@RequestParam`. В методах вызываются соответствующий функционал сервисов для выдачи данных пользователю.

Данные пользователю могут быть возвращены, как в виде html-страницы, так и в виде текста в формате JSON. В случае если возвращается html-страница, то для этого создаётся объект класса `ModelAndView`, куда помещаются необходимая страница и отображаемые данные в качестве атрибута для выдачи их пользователю с помощью Thymeleaf. В случае, если данные должны быть возвращены в формате JSON строки, данные с начала конвертируются в формат DTO или в формат ответа, содержащего http-код и строку `response` – результат выполнения. Пример контроллера для работы с данными изображен в Листинге 2.4. Пример контроллера для отображения страниц приведен в Листинге 2.5.

Листинг 2.4 – Фрагмент кода класса UserController

```
@RestController
@RequestMapping("/api/user")
public class UserController {
    private final ModelMapper modelMapper;
    private final UserService userService;
    private final Responser responser = new Responser();
    public UserController(ModelMapper modelMapper, UserService userService) {
        this.modelMapper = modelMapper;
        this.userService = userService;
    }
    @GetMapping("/all")
    public List<UserDTO> getAll(){
        return userService.getAll().stream().map(this::convertToUserDTO).toList();
    }
    @GetMapping("/level")
    public List<UserDTO> getAllByLevel(@RequestParam int idLevel){
        return
userService.getAllByLevel(idLevel).stream().map(this::convertToUserDTO).toList();
    }
    @GetMapping("/role")
    public List<UserDTO> getAllByRole(@RequestParam String role){
        return userService.getAllByRole(role).stream().map(this::convertToUserDTO).toList();
    }
    @GetMapping("/info")
    public UserDTO getUserInfo(@RequestParam String snpassport){
        return convertToUserDTO(userService.findBySnpassport(snpassport));
    }
}
```

Продолжение листинга 2.4

```
@PostMapping("/change")
public ResponseEntity<String> changeUserInfo(@RequestParam String snpassport,
@RequestParam String fullname, @RequestParam String dateOfBirth, @RequestParam String
password, @RequestParam String username, @RequestParam String role, @RequestParam int
idLevel){
```

```

        return responder.createResponse(userService.changeUserInfo(snpassport, fullname,
dateOfBirth, password, username, role, idLevel));
    }

    @DeleteMapping()
    public ResponseEntity<String> deleteUser(@RequestParam String snpassport){
        return responder.createResponse(userService.deleteUser(snpassport));
    }

    public UserDTO convertToUserDTO( User user){
        return modelMapper.map(user, UserDTO.class);
    }

    public User convertToUser( UserDTO userDTO) { return modelMapper.map(userDTO,
User.class); }
}

```


Листинг 2.5 – Фрагмент кода класса PageController

```
@Controller
@Slf4j
public class PageController {

    private final LevelService levelService;
    private final GroupService groupService;
    private final PermissionService permissionService;
    private final RentService rentService;
    private final UserService userService;
    private final VehicleBrandService vehicleBrandService;
    private final VehicleService vehicleService;
    private final VehicleModelService vehicleModelService;
    private final VehicleNameService vehicleNameService;
    private final VehicleWorkModelService vehicleWorkModelService;

    private final ModelMapper modelMapper;

    public PageController(...) {}

    @RequestMapping("/start")
    public ModelAndView getMainPage(@AuthenticationPrincipal UserDetails user){}

    @RequestMapping("/profile")
    public ModelAndView getProfilePage(@AuthenticationPrincipal UserDetails user){
        ModelAndView modelAndView = new ModelAndView("profile");
        User muser = userService.findBySnpassport(user.getUsername());
        Rent currentRent = rentService.getCurrentRent(muser.getSnpassport());
        List<Rent> rents = rentService.getAllByPass(muser.getSnpassport());
        if (currentRent != null){
            Vehicle_work_model curVehicleModel =
vehicleWorkModelService.getWorkModel(vehicleService.getVehicle(currentRent.getVin()).getIdVehicleWorkModel());
```

Продолжение листинга 2.5

```
String curVehicleName =
vehicleNameService.getVehicleName(curVehicleModel.getIdVehicleName());

modelAndView.addObject("vehicle_model", curVehicleModel);
```

```

        modelAndView.addObject("vehicle_name", vehicleName);
    }
    modelAndView.addObject("rents", rents);
    modelAndView.addObject("muser", muser);
    modelAndView.addObject("level", levelService.findById(muser.getIdLevel()));
    modelAndView.addObject("rent", currentRent);
    return modelAndView;
}

public VehicleDTO convertToVehicleDTO(Vehicle vehicle){
    return modelMapper.map(vehicle, VehicleDTO.class);
}

public Vehicle convertToVehicle(VehicleDTO vehicleDTO){
    return modelMapper.map(vehicleDTO, Vehicle.class);
}
}

```

После того, как взаимодействие с пользователем и с БД было реализовано, необходимо стало защитить различные слои информации от обычного пользователя. Для этого был использован Spring Security и класс `WebSecurityConfig` (Листинг 2.6) позволяющий ограничивать пользователей в доступе в зависимости от их роли или наличия авторизации.

Данный класс был помечен аннотациями `@Configuration` и `@EnableWebSecurity` для правильного распознавания системой. В нем был создан метод, помеченный аннотацией `@Bean`, содержащий в себе фильтрацию доступа к различным паттернам запросов, позволяющий выдать доступ на использование админ-панели только пользователям с ролью “ADMIN”, а также запрещающий отправку запросов не авторизованным пользователям.

Листинг 2.6 – Фрагмент кода класса WebSecurityConfig

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {

    @Autowired
    private CustomAuthenticationProvider authProvider;

    @Autowired
    private CustomAuthenticationSuccessHandler customAuthenticationSuccessHandler;

    @Autowired
    private DataSource dataSource;

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf().disable().cors().disable()
            .authorizeHttpRequests()
            .requestMatchers("/profile", "start").authenticated()
            .requestMatchers("/crm/*").hasRole("ADMIN")
            .requestMatchers("/login", "/logout", "/register").permitAll()
            .anyRequest().permitAll()
            .and()
            .formLogin()
            .usernameParameter("snpassport")
            .loginPage("/login")
            .loginProcessingUrl("/login")
            .successHandler(customAuthenticationSuccessHandler)
            .failureForwardUrl("/login?error");

        return httpSecurity.build();
    }
}
```

Продолжение листинга 2.6

```
@Autowired
protected void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
```

```

    auth
        .jdbcAuthentication()
        .dataSource(dataSource)
        .usersByUsernameQuery("SELECT snpassport, password, true FROM users WHERE
snpassport = ?")
        .authoritiesByUsernameQuery("SELECT snpassport, role FROM users WHERE
snpassport = ?");
    }
    @Bean
    public AuthenticationManager authManager(HttpSecurity http) throws Exception {
        AuthenticationManagerBuilder authenticationManagerBuilder =
            http.getSharedObject(AuthenticationManagerBuilder.class);
        authenticationManagerBuilder.authenticationProvider(authProvider);
        return authenticationManagerBuilder.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}

```

Выводы к разделу 2

Основываясь на проведенном анализе, рассмотренном в первом разделе, спроектирована база данных, разработана структура приложения, определен инструменты, шаблоны и технологии разработки. В результате были реализованы соответствующие классы.

3 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

3.1 Тестирование регистрации и авторизации

Для регистрации необходимо перейти в “/register” или нажать кнопку “Регистрация” на форме входа. Были введены регистрационные данные для нового пользователя (Рисунок 3.1).

Регистрация
Вход

Серия номер паспорта:

1718598307

ФИО пользователя:

Сидоров Станислав Дмитриевич

Дата рождения:

06.02.2003

Имя пользователя:

SidorovSD

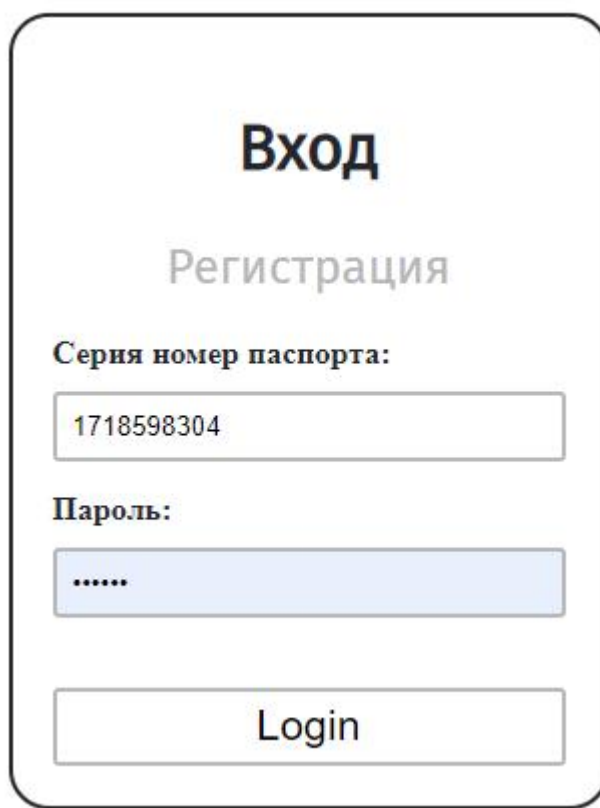
Пароль:

.....

Login

Рисунок 3.1 – Заполненная форма регистрации

Далее был выполнен вход под пользователем с правами администратора (Рисунок 3.2).



Вход

Регистрация

Серия номер паспорта:

1718598304

Пароль:

.....

Login

Рисунок 3.2 – Заполненная форма под пользователя

В результате в открывшейся админ панели можно наблюдать запись (Рисунок 3.3) с только что созданным пользователем.

| | | | | |
|------------|------------------------------------|------------|--|--------|
| 1718598307 | Сидоров Станислав Дмитриевич | 06.02.2003 | \$2a\$10\$fnyqXM.TDPmA3uJGkoRmOxFc4aObYKXheO42E6whOxf/A.fUlBMW | Sidoro |
|------------|------------------------------------|------------|--|--------|

Рисунок 3.3 – Запись о новом пользователе

3.2 Тестирование пользовательского функционала

Для тестирования пользовательского функционала был выполнен вход с аккаунта с ролью “USER”, в следствии чего была выдана страница “/start” – главная страница приложения (Рисунок 3.4), где уже можно увидеть карточку автомобиля доступного для аренды.

Быстро

Дешево

Спорт

На вечер

Премиум

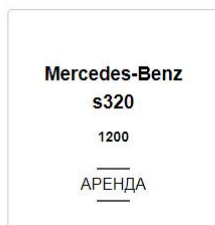



Рисунок 3.4 – Главная страница

При нажатии на кнопку “Аренда”, в строке поиска появляется панель, показывающая информацию об аренде и позволяющая её остановить (Рисунок 3.5).



Рисунок 3.5 – Информационная панель аренды

Также перейдя в профиль пользователя, можно увидеть основную информацию о нем, а также только что начатую аренду в таблице (Рисунок 3.6).


1718598307

Фамилия Имя Отчество

Сидоров Станислав Дмитриевич

Текущий уровень

Basic

ИСТОРИЯ ПОЕЗДОК

| RENT ID | VIN | DURATION | START POINT | END POINT | START TIME | END TIME |
|---------|-----------|----------|---------------------|-----------|---------------------|----------|
| 20 | 123abc100 | 0 | MoscowVernadskogo76 | none | 16.05.2023 19:32:33 | none |

Рисунок 3.6 – Профиль пользователя

При нажатии на кнопку остановки аренды, панель в строке пользователя пропадает, а запись в таблице изменяется (Рисунок 3.7).

| RENT ID | VIN | DURATION | START POINT | END POINT | START TIME | END TIME |
|---------|-----------|----------|---------------------|---------------|---------------------|---------------------|
| 20 | 123abc100 | 645 | MoscowVernadskogo76 | MoscowHove263 | 16.05.2023 19:32:33 | 16.05.2023 19:43:18 |

Рисунок 3.7 – Профиль пользователя после окончания аренды.

3.3 Тестирование функций администратора

Были протестированы все функции доступные администратору через админ-панель, но продемонстрирована будет лишь часть из них. Для примера была выбрана основная страница, содержащая информацию об арендах (Рисунок 3.8).

| ID rent | SNPassport | VIN | Duration | Start point | End point | Start time | End time |
|---------|------------|-----------|----------|---------------------|---------------|---------------------|---------------------|
| 14 | 1718598305 | 123abc100 | 120 | Moscow | Voronej | 09.05.2023 19:34:00 | 09.05.2023 19:36:00 |
| 15 | 1718598305 | 123abc124 | 871 | MoscowVernadskogo76 | MoscowHove263 | 10.05.2023 21:41:25 | 10.05.2023 21:55:56 |
| 20 | 1718598307 | 123abc100 | 645 | MoscowVernadskogo76 | MoscowHove263 | 16.05.2023 19:32:33 | 16.05.2023 19:43:18 |

Рисунок 3.8 – Страница admin_rent

На данной странице расположена таблица со всеми созданными арендами, также существуют формы для добавления (Рисунок 3.9), изменения (Рисунок 3.10) и удаления (Рисунок 3.11) записи.

Add new rent

SNPassport:

1718598307

VIN:

123abc125

Start point:

Moscow

End point:

Vladimir

Start time:

16.05.2023 18:19:20

End time:

16.05.2023 20:03:20

ADD

Рисунок 3.9 – Форма добавления аренды

Change rent

ID:

SNPassport:

VIN:

Start point:

End point:

Start time:

End time:

Рисунок 3.10 – Форма для изменения аренды

Delete rent rent

Rent id:

DELETE

Рисунок 3.11 – Форма удаления аренды

Все использованные формы отработали успешно, что можно увидеть на рисунке 3.12.

Table of rents

| ID rent | SNPassport | VIN | Duration | Start point | End point | Start time | End time |
|---------|------------|-----------|----------|---------------------|---------------|---------------------|---------------------|
| 15 | 1718598305 | 123abc124 | 871 | MoscowVernadskogo76 | MoscowHove263 | 10.05.2023 21:41:25 | 10.05.2023 21:55:56 |
| 20 | 1718598307 | 123abc100 | 645 | MoscowVernadskogo76 | MoscowHove263 | 16.05.2023 19:32:33 | 16.05.2023 19:43:18 |
| 21 | 1718598305 | 123abc123 | 6240 | Vladivostok | Vladimir | 16.05.2023 18:19:20 | 16.05.2023 20:03:20 |

Рисунок 3.12 – Результат работы форм

Выводы к разделу 3

С помощью тестирования можно корректировать работу приложения. В данном разделе было продемонстрировано тестирование основного функционала приложения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была освоена компетенция «Шаблоны программных платформ языка Джава» в степени соответствующей рабочей программной дисциплины. Как результат – выполненное приложение на тему «Каршеринговая компания».

Во время разработки курсовой работы было создано 10 таблиц БД, 10 сущностей, 13 контроллеров, 11 сервисов, 10 репозитория и 10 HTML страниц. Весь код был написан с помощью таких технологий как JDK, Spring Framework, Spring Security, Spring Boot, Lombok, Postman, Bootstrap. Также в ходе создания проекта была полностью изучена новая среда разработки IntelliJIDEA.

Был проведен анализ предметной области разрабатываемого приложения. Все созданные веб-страницы содержат подобранный в ходе разработки текстовый и визуальный контент. Также они имеют одинаковую стилизацию основной структуры, соответствующую современным стандартам веб-разработки. Благодаря проработанной идеи размещения контента на веб-страницах, весь предоставленный пользователю материал легкодоступен и понятен. Все пункты, поставленные в задании на курсовую работу, были выполнены. Исходя из этого, выполнение курсовой работы можно считать успешным.

Ссылка на курсовую работу: <https://github.com/MShizikU/BasicCarSharing>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Справочник по Spring Framework / [Электронный ресурс] // Baeldung: [сайт]. — URL: <https://www.baeldung.com/> (дата обращения: 01.05.2023).
2. Официальный сайт Spring Framework / [Электронный ресурс] // Spring: [сайт]. — URL: <https://spring.io/> (дата обращения: 01.05.2023).
3. Официальный веб-сайт среды разработки IntelliJ idea / [Электронный ресурс] // URL: <https://www.jetbrains.com/ru-ru/idea/> (дата обращения: 01.05.2023).
4. Официальный веб-сайт PostgreSQL / [Электронный ресурс] // PostgreSQL: [сайт]. URL: <https://www.postgresql.org> (дата обращения: 01.05.2023).
5. Документация по Bootstrap 4 / [Электронный ресурс] // URL: <https://bootstrap-4.ru/> (дата обращения: 01.05.2023).