



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: Разработка серверных частей интернет-ресурсов

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Серверная часть веб-приложения «Каршеринг»

Студент: Сидоров Станислав Дмитриевич

Группа: ИКБО-20-21

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

Руководитель: старший преподаватель Сеницын Анатолий Васильевич

Работа допущена к защите \_\_\_\_\_ (дата) \_\_\_\_\_ /Сеницын А.В./

Оценка по итогам защиты: \_\_\_\_\_

\_\_\_\_\_ / \_\_\_\_\_ /

\_\_\_\_\_ / \_\_\_\_\_ /

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

М. РТУ МИРЭА. 2023 г.

УДК 004.4

Сидоров С.Д. Курсовая работа направления подготовки «Программная инженерия» на тему **«серверная часть веб-приложения «Каршеринг»»**: М. 2023 г., МИРЭА – Российский технологический университет (РТУ МИРЭА), Институт информационных технологий (ИИТ), кафедра инструментального и прикладного программного обеспечения (ИиППО) – 30 стр., 17 рис., 1 табл., 22 источн.

Ключевые слова: веб-приложение каршеринг, чистая архитектура, JWT авторизация, масштабируемая архитектура, разделение сервисов

Целью работы является создание серверной части веб приложения на тему «Каршеринг». В рамках работы осуществлен краткий анализ аналогов веб-приложения по выбранной тематике, разработано приложение с использованием фреймворка Spring, произведено тестирование приложения и проверка на антиплагиат.

Ivanov I.I., Final qualification work for the educational program "Software development and information system design" of the training program "Software engineering" on the topic **"server side of web-app "Carsharing"**: M. 2023, MIREA - Russian Technological University (RTU MIREA), Institute of Information Technologies (IIT), Department of the Tool and Applied Software (Department of TAS) - 30 p., 17 ill., 1 tabl., 22 ref.

Keywords: car-sharing web application, clean architecture, JWT authentication, scalable architecture, service separation

The aim of the work is to create the server-side of a web application on the topic of "Carsharing." 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.

РТУ МИРЭА: 119454, Москва, пр-т Вернадского, д. 78

кафедра инструментального и прикладного программного обеспечения (ИиППО)

Тираж: 1 экз. (на правах рукописи)

Файл: «ПЗ\_РСЧИР\_ИКБО-20-21\_СидоровСД.pdf», исполнитель Сидоров С.Д.

© С.Д. Сидоров

## АННОТАЦИЯ

В курсовой работе описывалось создание интернет-ресурса, на тему «Каршеринг». 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 30 листах, содержит 17 рисунков, 22 использованных источников, 3 листинга, 1 таблица.

The course work described the creation of an internet resource on the topic of "Carsharing". The work contains an analysis of the subject area of the Internet resource being developed, the creation of web pages of the Internet resource using Spring Framework technologies and testing of the developed application.

The introduction substantiates the relevance of the chosen topic, defines the purpose of the work and the tasks to be solved to achieve it, describes the object and subject of the study, the methods used and the information base of the study, and briefly describes the structure of the CD by sections.

The main part contains the material necessary to achieve the goal of the CD. The main part includes general information (in particular, the name of the internet resource, an enumeration of the application software necessary for the development and operation of the Internet resource, as well as the names of the languages and technologies with which the Internet resource is implemented), a description of the functional purpose of the Internet resource and its logical structure, a description of the development and the functions of the software application, testing the operation of the application. In conclusion, the theoretical conclusions that were formulated as a result of this course work are consistently presented.

Term paper on 30 sheets, contains 17 drawings, 22 used sources, 3 listings, 1 table.

## СОДЕРЖАНИЕ

ГЛОССАРИЙ.....	7
ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ .....	8
ВВЕДЕНИЕ.....	9
1. СБОР И АНАЛИЗ ТРЕБОВАНИЙ К ВЕБ-ПРИЛОЖЕНИЮ.....	10
1.1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ .....	10
2. ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ.....	12
2.1 Выбор архитектуры.....	12
2.2 Выбор технологий разработки.....	14
2.3 Прикладное программное обеспечение, необходимое для разработки и функционирования приложения.....	14
3 РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА .....	15
4 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА .....	19
4.1 Разработка слоя моделей .....	19
4.2 Разработка слоя контроллеров.....	20
4.3 Разработка слоя представления .....	21
4.4 Разработка микросервиса авторизации.....	22
5 ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ	23
ЗАКЛЮЧЕНИЕ .....	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	29

## ГЛОССАРИЙ

1. **JWT** — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON.
2. **Model-View-Controller (MVC)** — это схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер.
3. **Clean Architecture** — это архитектурный шаблон разработки программного обеспечения, предложенная Робертом Мартином, нацеленная на создание модульных, гибких и легко поддерживаемых приложений.

## **ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ**

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. 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 примеров таких сервисов является каршеринг, позволяя пользователям арендовать различные транспортные средства за оплату, зависящую от времени пользования, что позволяет людям совершать повседневные поездки по цене такси, но с комфортом личного автомобиля.

Целью данной курсовой работы является разработка серверной части веб-приложения “Каршеринг” с использованием Spring Framework, JDK и IntelliJIDEA. Для выполнения поставленной цели процесс разработки был поделён на несколько частей:

- Анализ предметной области разрабатываемого веб-приложения;
- Выбор технологий разработки;
- Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования;
- Реализовать слой серверной логики веб-приложения с применением выбранной технологии;
- 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 "Каршеринг".

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 и представлены в таблице 1 различные функции, присущие какому-либо из выше представленных сервисов или не присутствующие вовсе.

Таблица 1 – Сравнение приложений – аналогов

	Яндекс Драйв	Делимобиль	BelkaCar
Наличие списка автомобилей	+	-	+
Доступ к профилю пользователя	+	-	-
Информация об аренде	+	+	-

На основе выше представленной информации можно заметить, что часть сервисов не предоставляют возможность пользователю взаимодействовать с компанией через веб-приложение, а также были определены ключевые функции, которые должны быть включены в приложение “Каршеринг”:

- регистрация пользователя;
- просмотр информации об проведенных арендах и данных профиля;
- выбор автомобиля для аренды;
- начало и окончание аренды автомобилей;
- изменение, добавление и удаление информации об автомобилях и пользователях.

## **2. ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ**

### **2.1 Выбор архитектуры**

Первым этапом выбора технологий является выбор архитектуры приложения. Далее будут рассмотрены несколько из наиболее используемых.

Монолитная архитектура [1] – это классический подход, при котором все компоненты приложения находятся в одной единственной программе или кодовой базе. Монолит позволяет быстро и просто разворачивать приложения, однако может быть сложным для масштабирования и поддержки при увеличении размера и сложности проекта.

Микросервисная архитектура [2] – это подход, основанный на разделении приложения на составные части, называемые микросервисами. Каждый микросервис является небольшим, автономным модулем, который выполняет определенную функцию приложения. Микросервисы могут взаимодействовать друг с другом посредством API. Эта архитектура облегчает масштабирование, гибкость разработки и обновления отдельных компонентов, но может быть сложной для управления и требовать дополнительных усилий в области координации и управления межсервисными взаимодействиями.

Микросервисный монолит [3] – это гибридный подход, сочетающий черты монолитной и микросервисной архитектур. В этой модели весь функционал приложения разделен на изолированные модули, но эти модули все еще развертываются и работают внутри одной кодовой базы. Это позволяет извлечь выгоды микросервисной архитектуры, не готовя новые инфраструктурные компоненты для каждого сервиса.

В рамках приложения “Каршеринг” наиболее подходящей является микросервисная архитектура, так как она позволяет распределить нагрузку на

разные сервисы, предоставляет дополнительные возможности к дальнейшему масштабированию,

Далее будут рассмотрены несколько паттернов проектирования, используемых внутри микросервисов.

Чистая архитектура (Clean Architecture) [4] – это архитектурный шаблон разработки программного обеспечения, предложенная Робертом Мартином. Она нацелена на создание модульных, гибких и легко поддерживаемых приложений. Архитектура должна:

- быть тестируемой;
- не зависеть от UI;
- не зависеть от БД, внешних фреймворков и библиотек.

Это достигается путем разделения системы на слои, и следования принципам SOLID [4].

MVC [5] является распространённым паттерном проектирования, используемым для проектирования веб-приложений. Он позволяет разделить логику приложения на три основных компонента: модель (Model), представление (View) и контроллер (Controller).

Модель представляет собой слой, отвечающий за работу с данными – обработку бизнес-логики.

Представление отвечает за отображение данных в удобном пользователю формате, а также за получение информации от пользователя.

Контроллер является посредником между моделью и представлением, принимая запросы и распределяя их по необходимым действиям.

MVC обеспечивает разделение ответственности между компонентами, предоставляет возможности к масштабированию, а также упрощает разработку и тестирование приложений.

Для разработки серверной части веб-приложения “Каршеринг” был выбран паттерн MVC, что позволит в дальнейшем масштабировать приложение и вносить новые изменения.

## **2.2 Выбор технологий разработки**

В качестве языка программирования для разработки серверной части веб-приложения был выбран объектно-ориентированный язык программирования Java [6], который отлично себя зарекомендовал в данном виде разработки. Критериями выбора данного языка стали:

- Java обладает обширным списком библиотек и фреймворков;
- Java обеспечивает кросс-платформенный язык программирования, что в дальнейшем позволит переносить и масштабировать приложение;
- Java предлагает множество встроенных функций по улучшению безопасности и эффективной работы с памятью.

В качестве базы данных была выбрана PostgreSQL [8]. PostgreSQL – это мощная объектно-реляционная система управления базами данных (СУБД). Она является одной из наиболее распространенных СУБД с открытым исходным кодом и обладает активным сообществом разработчиков.

Для взаимодействия между серверной и клиентской частями приложения был использован протокол HTTP, а также формат передачи данных JSON. JSON является универсальным форматом, который поддерживается большинством языков программирования и позволяет передавать данные в удобном для обработки виде.

## **2.3 Прикладное программное обеспечение, необходимое для разработки и функционирования приложения**

В качестве среды разработки, предназначенного для работы с Java был выбран редактор с функциями IDE – IntelliJ IDEA. IntelliJ IDEA предоставляет широкий набор инструментов для удобства разработки.

### **3 РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА**

В качестве основы была выбрана микросервисная архитектура. В следствии чего приложение было разделено на два различных микросервиса:

- сервис авторизации;
- сервис аренды.

Сервис авторизации предоставляет генерацию токенов на основании введенных пользователем данных. Сервис аренды в свою очередь выполняет необходимые действия по работе с базами данных, передачи данных между пользователем и приложением.

Сервис авторизации был вынесен в отдельный микросервис для возможности подключения к нему других микросервисов, которые появятся при масштабировании веб-приложения, в следствии чего пользователь сможет использовать один профиль в различных сервисах.

В предыдущей главе для разработки приложения был выбран паттерн MVC.

Следуя разграничениям заданными этим паттерном, структура приложения была разделена на несколько частей:

- представление;
- контроллер;
- модель.

Слой представления отвечает за отображения данных системы пользователю.

Слой контроллера является связующим звеном между представлением и другими компонентами.

Слой модели представляет собой оператора бизнес-логики, который отвечает за взаимодействие данных между собой, правильную их обработку и хранение.

Также в ходе разработки серверной-части веб-приложения, была спроектирована база данных. В результате была сформирована диаграмма ERD [16], изображенная на рисунке 1, отображающая сущности присутствующие в базе данных.

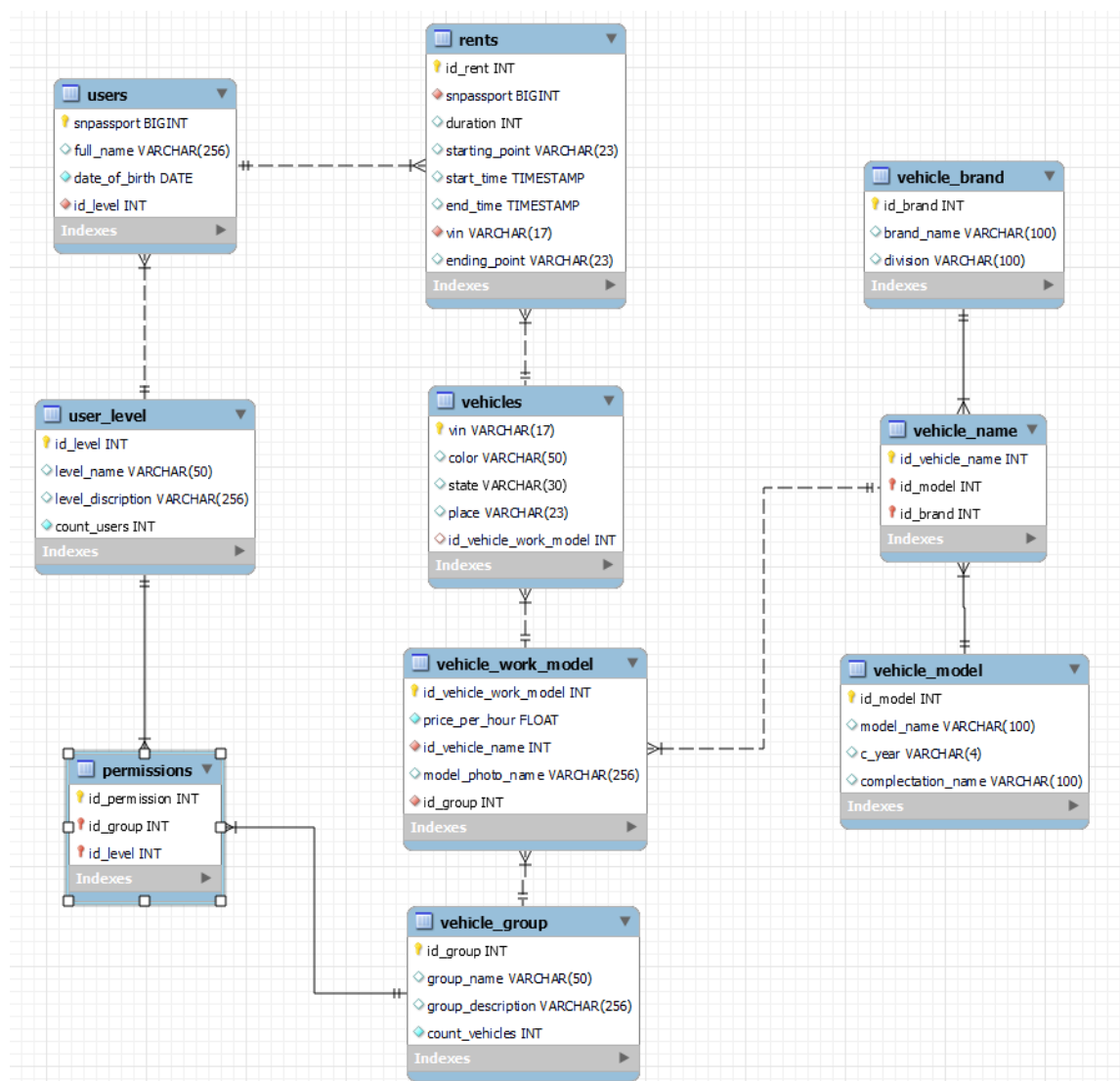


Рисунок 1 – ERD-диаграмма базы данных сервиса аренды

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 идентификации 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.

## 4 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА

### 4.1 Разработка слоя моделей

В процессе разработки были созданы модели данных, которые описывают основные сущности с их полями. На рисунке 2 представлен список разработанных моделей.

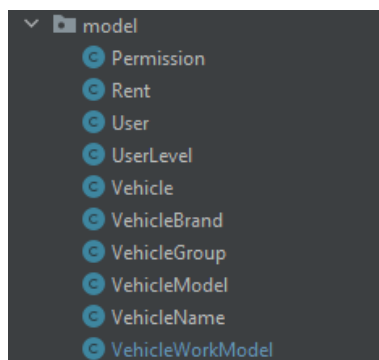


Рисунок 2 – Список разработанных моделей

В качестве примера, в листинге 1 представлен код основной модели приложения – Rent.

Листинг 1 – Код модели Rent

```
@Table(name = "rents")
public class Rent {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_rent")
    private Integer idRent;

    @JoinColumn(name = "snpassport")
    private Long snpassport;
    @JoinColumn(name = "vin")
    private String vin;
    @Column(name = "duration")
    private Integer duration;
    @Column(name = "starting_point")
    private String startingPoint;
    @Column(name = "ending_point")
    private String endingPoint;
    @Column(name = "start_time")
    private LocalDateTime startTime;
    @Column(name = "end_time")
    private LocalDateTime endTime;
}
```

Для работы с бизнес-логикой, обработки различных данных были созданы, помеченные аннотацией `@Service`, классы – сервисы, которые содержат методы, позволяющие работать с базой данных и выполнять какие-либо действия, например начинать или завершать аренду. На рисунке 3 представлен список разработанных сервисов.

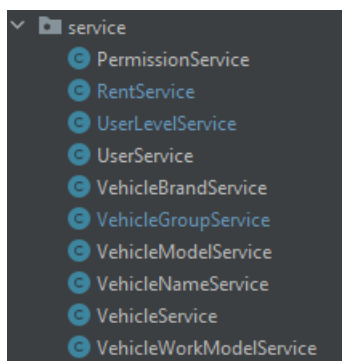


Рисунок 3 – Список разработанных сервисов.

## 4.2 Разработка слоя контроллеров

После создание слоя моделей, был создан слой контроллеров, представляющий собой набор различных обработчиков запросов, которые направляют данные в определенные сервисы для их дальнейшей обработки. Фрагмент кода `RentController` представлен в листинге 2.

Листинг 2 – Фрагмент кода контроллера `RentController`

```
@PostMapping("/start")
public ResponseEntity<ExecutionResult<Rent>>
startRent(@RequestBody Rent rent){
    ExecutionResult<Rent> result = rentService.startRent(rent);
    if (result.getErrorMessage() != null)
        return ResponseEntity.badRequest().body(result);
    return ResponseEntity.ok(result);
}
@PostMapping("/close/{id}")
public ResponseEntity<ExecutionResult<Rent>>
startRent(@PathVariable Integer id, @RequestBody Rent rent){
    ExecutionResult<Rent> result = rentService.closeRent(id,rent);
    if (result.getErrorMessage() != null)
        return ResponseEntity.badRequest().body(result);
    return ResponseEntity.ok(result);
}
```

На рисунке 4 изображены созданные контроллеры.

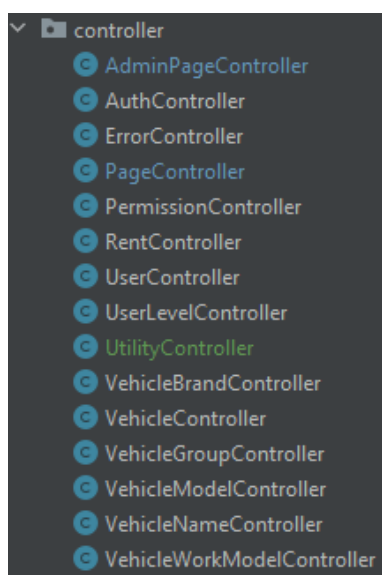


Рисунок 4 – Список разработанных контроллеров

### 4.3 Разработка слоя представления

В процессе разработки слоя представления в контроллеры была добавлена часть, позволяющая направлять пользователю подготовленные страницы с данными. Для этого использовалась библиотека Thymeleaf, что позволило уменьшить время загрузки страниц, не используя запросы для получения данных со стороны пользователя. В результате были созданы страницы для пользователя и администратора с использованием HTML и CSS, а для отправления запросов использовался JavaScript.

На рисунке 5 изображен список разработанных страниц.

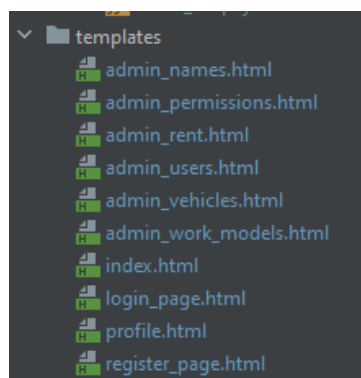


Рисунок 5 – Список разработанных страниц

## 4.4 Разработка микросервиса авторизации

В процессе разработки микросервиса авторизации, было создано отдельное Java Spring приложение, к которому обращается микросервис аренды внутри фильтрации обращения с помощью Java Spring Security.

Микросервис авторизации в свою очередь принимая запрос, обрабатывает данные и на основе имеющихся записей в базе данных генерирует JWT токен, содержащий информацию о пользователе.

В листинге 3 содержится фрагмент кода обработчика login запроса.

```
@PostMapping(path = "login")
public LoginResult login(@RequestBody LoginRequest loginRequest)
{
    UserDetails userDetails;
    try {
        userDetails =
userDetailsService.loadUserByUsername(loginRequest.getUsername()
);
    } catch (UsernameNotFoundException e) {
        throw new
ResponseStatusException(HttpStatus.UNAUTHORIZED, "User not
found");
    }
    if
(loginRequest.getPassword().equals(userDetails.getPassword())){
        Map<String, String> claims = new HashMap<>();
        claims.put("username", loginRequest.getUsername());
        claims.put("password", loginRequest.getPassword());
        String authorities =
userDetails.getAuthorities().stream()
            .map(Object::toString)
            .collect(Collectors.joining(","));
        claims.put("authorities", authorities);
        claims.put("userId", "1");
        String jwt =
jwtHelper.createJwtForClaims(loginRequest.getUsername(),
claims);
        return new LoginResult(jwt);
    }
    throw new ResponseStatusException(HttpStatus.UNAUTHORIZED,
"User not authenticated");
}
```

## 5 ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

Тестирование программного продукта является важной частью разработки, поскольку оно позволяет выявить ошибки и дефекты, а также гарантирует работоспособность обещанного функционала.

На данном этапе развития приложения было проведено функциональное тестирование с использованием разработанного интерфейса, результаты которого приведены ниже.

На рисунках 6 – 8 изображено тестирование функционала авторизации.

Регистрация

[Вход](#)

Серия номер паспорта:

1718598304

ФИО пользователя:

Сидоров Станислав Дмитриевич

Дата рождения:

2004-06-03

Имя пользователя:

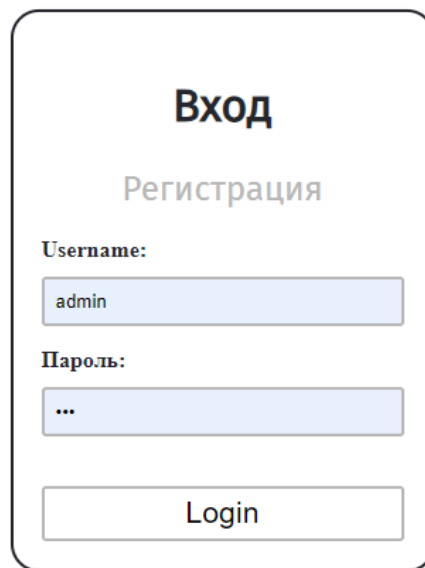
admin

Пароль:

...

Register

Рисунок 6 – Заполнение формы регистрации



**Вход**

Регистрация

Username:

admin

Пароль:

...

Login

Рисунок 7 – Заполнение формы входа

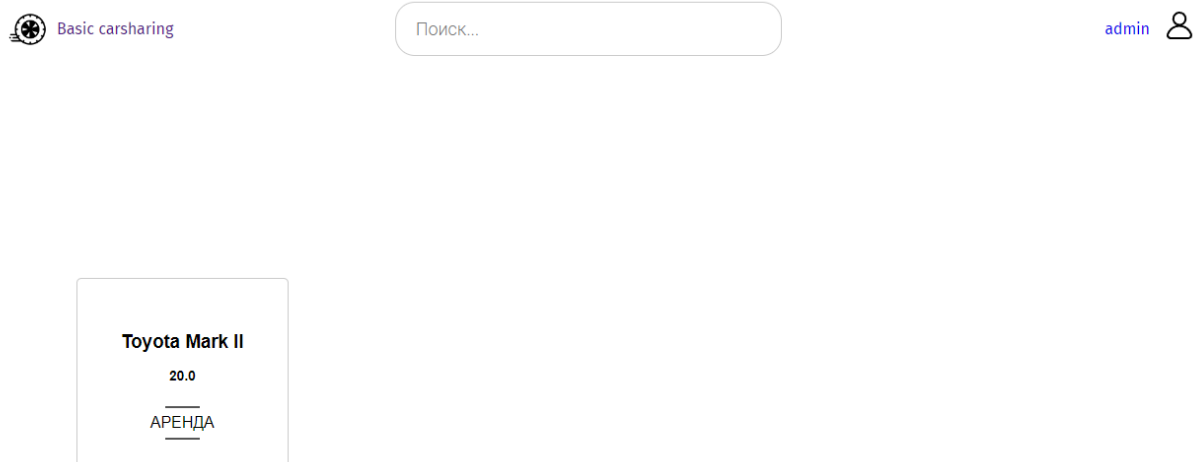


Рисунок 8 – Результат успешной авторизации

На рисунках 9 – 10 изображены результаты тестирования функционала взаимодействия с арендой.

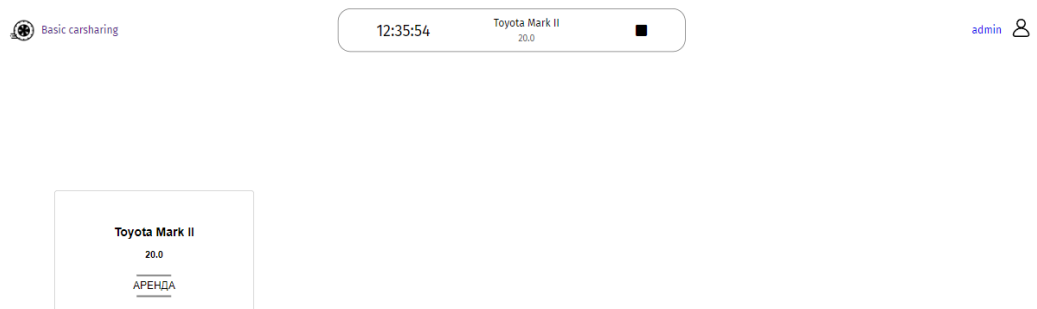


Рисунок 9 – Результат успешного начала аренды

## ИСТОРИЯ ПОЕЗДОК

RENT ID	VIN	DURATION	START POINT	END POINT	START TIME	END TIME
1	ABCDEFGHIJKLMNO0146		test	56kycX	08/12/2023 12:35:54	08/12/2023 12:35:54

Рисунок 10 – Результат успешного завершения аренды

Также веб-приложение обладает страницами, позволяющими администраторам взаимодействовать с сущностями системы.

На рисунках 11 – 16 продемонстрировано тестирование одного из разделов панели администратора.

### Table of work models

ID	Model photo name	Price per hour	Vehicle name ID	Group ID
1	toymarkii2000	20.0	1	1
2	mere2002001	25.5	2	2
3	honcrv2004	30.0	3	3

Рисунок 11 – Отображение панели существующих записей в базе

## Add new work model

Model photo name:

Price per hour:

Id vehicle name:

Group id:

Рисунок 12 – Заполнение формы добавления новой записи



Table of work models

ID	Model photo name	Price per hour	Vehicle name ID	Group ID
1	toymarkii2000	20.0	1	1
2	mere2002001	25.5	2	2
3	honcrv2004	30.0	3	3
4	car_tmp_photo_name	600.0	3	1

Рисунок 13 – Результат добавления новой записи

## Change work model

Work Model ID:

Model photo name:

Price per hour:

Group id:

Id vehicle name:

CHANGE

Рисунок 14 – Заполнения формы на изменение записи

Table of work models

ID	Model photo name	Price per hour	Vehicle name ID	Group ID
1	toymarkii2000	20.0	1	1
2	mere2002001	25.5	2	2
3	honcrv2004	30.0	3	3
4	test	23.2	2	2


Рисунок 15 – Результат изменения записи в базе

## Table of work models


ID	Model photo name	Price per hour	Vehicle name ID	Group ID
1	toymarkii2000	20.0	1	1
2	mere2002001	25.5	2	2
3	honcrv2004	30.0	3	3


### Рисунок 16 – Результат удаления записи из базы

Также любой пользователь веб-приложения должен иметь возможность посмотреть свой профиль, тестирование данной возможности изображено на рисунке 17.

 Basic carsharing

Поиск...

admin 

  
1718598304

Фамилия Имя Отчество  
Сидоров Станислав Дмитриевич

Текущий уровень  
First level

ИСТОРИЯ ПОЕЗДОК

RENT ID	VIN	DURATION	START POINT	END POINT	START TIME	END TIME
1	ABCDEFGHIJKLMNO0146	test	56kycX	08/12/2023 12:35:54	08/12/2023 12:35:54	

### Рисунок 17 – Отображение профиля пользователя

## ЗАКЛЮЧЕНИЕ

Для достижения поставленной цели было необходимо решить следующие задачи:

- провести анализ предметной области разрабатываемого веб-приложения;
- обосновать выбор технологий разработки веб-приложения;
- разработать архитектуру веб-приложения на основе выбранного паттерна проектирования;
- реализовать слой серверной логики веб-приложения с применением выбранной технологии;
- реализовать слой логики базы данных;
- разработать слой клиентского представления веб-приложения;

В результате выполнения данной курсовой работы были выполнены все поставленные задачи, разработано приложение по теме «Каршеринг» с использованием паттерна проектирования MVC, произведено функциональное тестирование с помощью веб-интерфейса.

Отчет сформирован согласно инструкции по организации и проведению курсового проектирования и ГОСТ 7.32. Так же был подготовлен демонстрационный материал в виде презентации. Вся проделанная работа была проверена с помощью системы антиплагиат.

Исходный код серверной части приложения доступен по ссылке – <https://github.com/MShizikU/BasicCarSharing>[22]

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ньюмен С. От монолита к микросервисам: Пер. с англ. — СПб.: БХВ-Петербург, 2021. — 272 с.: ил.
2. Создание микросервисов. 2-е изд. — СПб.: Питер, 2023. — 624 с.: ил. — (Серия «Бестселлеры O'Reilly») ISBN 978-5-9775-6723-7.
3. Ричардсон, С. Шаблоны микросервисов: с примерами на Java. - 1-е издание. - Manning Publications, 2020. - 526 с. - ISBN 978-1617294549.
4. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — СПб. : Питер, 2021. — 352 с.
5. Тузовский, А. Ф. Проектирование и разработка web-приложений: учебное пособие для вузов / А. Ф. Тузовский. — Москва: Издательство Юрайт, 2021. — 218 с. — (Высшее образование). — ISBN 978-5-534-00515-8. — Текст : электронный // Образовательная платформа Юрайт [Электронный ресурс]. — URL: <https://urait.ru/bcode/469982> (дата обращения: 1.10.2023).
6. Сьерра Кэти, Бэйтс Берт Head First Java / Сьерра Кэти, Бэйтс Берт — СПб. : Эксмо, 2023. — 720 с. — ISBN 978-5-699-54574-2.
7. Раджпут Д. Spring. Все паттерны проектирования. – СПб.: Питер, 2019.
8. PostgreSQL: About [Электронный ресурс] – URL: <https://www.postgresql.org/about/> (дата обращения 10.10.2023).
9. Introduction to ORM with Spring [Электронный ресурс] – URL: (дата обращения 11.10.2023).
10. Hibernate Getting Started Guide [Электронный ресурс]. – URL: [https://docs.jboss.org/hibernate/orm/6.1/quickstart/html\\_single](https://docs.jboss.org/hibernate/orm/6.1/quickstart/html_single) (дата обращения 11.10.2023).
11. Project Lombok: Clean, Concise Java Code [Электронный ресурс] – URL: <https://www.oracle.com/corporate/features/project-lombok.html> (дата обращения 3.11.2023).
12. JAVA Development Kit (JDK) - GeeksforGeeks [Электронный ресурс]

– URL: <https://www.geeksforgeeks.org/jdk-in-java/> (дата обращения 1.10.2023).

13. What is Gradle? [Электронный ресурс] – URL: [https://docs.gradle.org/8.1.1/userguide/what\\_is\\_gradle.html](https://docs.gradle.org/8.1.1/userguide/what_is_gradle.html) (дата обращения 1.11.2023).

14. Модель C4 (C4 model) для визуализации архитектуры программного обеспечения [Электронный ресурс] – URL: <https://infostart.ru/pm/1540208/> (дата обращения: 12.10.2023).

15. Фаулер М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ Плюс, 2004. – 192 с. – ил. – ISBN 5-93286-060-X.

16. Кара-Ушанов Владимир Юрьевич МОДЕЛЬ «СУЩНОСТЬ – СВЯЗЬ» [Электронный ресурс]: Учебное пособие. – Екатеринбург: УрФУ, 2017 – URL: <https://study.urfu.ru/Aid/Publication/13604/1/Kara-Ushanov.pdf> (дата обращения 12.10.2023).

17. Spring Data JPA [Электронный ресурс] – URL: <https://spring.io/projects/spring-data-jpa> (дата обращения 20.10.2023).

18. JPA Criteria Queries | Baeldung [Электронный ресурс] – URL: <https://www.baeldung.com/hibernate-criteria-queries> (дата обращения 1.03.2023)

19. Hands-On RESTful API Design Patterns and Best Practices / Harihara Subramanian, Pethuru. — Raj Packt Publishing Ltd, 2019. — 378 с. — ISBN 978-1788992664.

20. Меджуи М., Уайлд Э., Митра Р., Амундсен М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. — СПб.: Питер, 2020.

21. Хоффман Эндрю X85 Безопасность веб-приложений. — СПб.: Питер, 2021. — 336 с.: ил. — (Серия «Бестселлеры O'Reilly»).

22. Сидоров С. | BasicCarSharing [ Исходный код ] — URL: <https://github.com/MShizikU/BasicCarSharing> (08.12.2023)