**СОДЕРЖАНИЕ**

Введение…………………………………………………………………….6

1.Описание предметной области……………………………………. …....7

1.1 Назначение программного продукта…………………………………..7

1.2 Основные цели и задачи программного продукта…………………….7

1.3 Основные функциональные возможности…………………………….8

1.4 Преимущества использования программного продукта……………..8

2. Описание маппинга……………………………………………………...9

2.1 Сущность Client…………………………………………………………9

2.2 Сущность Parts ………………………………………………………..10

2.3 Сущность Order………………………………………………………..11

3. Описание точек доступа………………………………………………..13

3.1 Клиенты (Clients)……………………………………………………...13

3.2 Запчасти (Parts)………………………………………………………...15

3.3 Заказы (Orders)………………………………………………………...19

4. Разработка REST-сервиса……………………………………………...23

4.1 Структура проекта…………………………………………………….23

4.2 Запуск проекта…………………………………………………………25

Заключение………………………………………………………………...27

Список использованных источников…..………………………………...28

**ВВЕДЕНИЕ**

Автомобили - незаменимая часть нашей повседневной жизни, обеспечивая мобильность и комфорт в современном мире. Вместе с ростом числа автовладельцев возрастает и спрос на автозапчасти, необходимые для обслуживания и ремонта автомобилей. Магазины автозапчастей играют важную роль в этой цепочке предоставления услуг автомобильного сервиса, обеспечивая широкий выбор запчастей, необходимых для поддержания технического состояния автомобиля. Цель данной курсовой работы заключается в исследовании особенностей функционирования магазинов автозапчастей, их роли на авторынке, а также оценке влияния внешних и внутренних факторов на их деятельность. В рамках работы будет проанализировано строение и принципы работы магазинов автозапчастей, особенности управления запасами и ценообразования, а также влияние конкуренции на бизнес данного сегмента. Понимание особенностей функционирования магазинов автозапчастей является важным шагом для оптимизации и развития данного сектора автомобильной индустрии. В результате исследования можно будет выявить основные тенденции развития данного рынка и предложить рекомендации по улучшению эффективности деятельности магазинов автозапчастей.

**1. Описание предметной области**

1.1 Назначение программного продукта

Программный продукт, разработанный в рамках данной курсовой работы, представляет собой систему магазина автозапчастей. Основное назначение системы – автоматизация процессов, связанных с заказом автомобильных запчастей, увеличение скорости и обработки, заказов и качества обслуживания клиентов и повышение эффективности работы сотрудников магазина. Данный программный продукт разработан для того, чтобы обеспечить удобное и эффективное управление заказами, учет запчастей для их автомобилей, а также контроль за доставкой.

1.2 Основные цели и задачи программного продукта

1. Автоматизация процессов обслуживания:

* Обеспечение автоматического создания и ведения заказов.
* Сокращение времени на оформление документов и поиск информации.
* Уменьшение вероятности ошибок при ведении учета.

2. Улучшение взаимодействия с клиентами:

* Ведение базы данных клиентов, включающей контактную информацию и историю заказов.
* Учет запчастей, принадлежащих клиентам, и истории их заказов.
* Возможность отправки уведомлений клиентам о статусе их заказа.

3. Оптимизация заказов:

* Управление датой заказа и статусом доставки.
* Учет выполненных работ и использование запасных частей.

4. Контроль и отчетность:

* Генерация отчетов по выполненным работам, использованным материалам и другим параметрам.
* Анализ данных для принятия управленческих решений.
* Контроль за финансовыми потоками и учет затрат.

1.3 Основные функциональные возможности

Управление заказами. Ведение базы данных клиентов и их заказов, поиск и фильтрация по различным параметрам, добавление и редактирование информации о клиентах и их заказах.

Управление заказами. Учет заказов, находящихся в статусе обработки хранение информации о марке, модели, VIN-коде и прочих характеристиках.

Создание и ведение заказов. Оформление заказов на доставку, добавление информации о необходимых комплектующих, контроль за статусом выполнения заказа.

1.4 Преимущества использования программного продукта

Повышение эффективности работы Магазина автозапчастей. Автоматизация заказов позволяет сократить время на их выполнение и уменьшить количество ошибок.

Улучшение качества обслуживания. Полная информация о клиенте и его заказе, позволяет отслеживать статус заказал и статус доставки к клиенту.

Контроль и прозрачность. Возможность генерации отчетов и анализа данных позволяет руководству Магазина принимать обоснованные управленческие решения.

Программный продукт магазин автозапчастей является комплексным решением, направленным на улучшение всех аспектов работы Магазина, начиная от взаимодействия с клиентами и заканчивая внутренней организацией процессов. Это позволяет не только повысить качество обслуживания клиентов, но и значительно улучшить управляемость и контроль за деятельностью станции.

**2. Описание маппинга**

В данной курсовой работе Магазин автозапчастей включает несколько ключевых сущностей, каждая из которых связана с базой данных посредством маппинга. Маппинг сущностей позволяет эффективно управлять данными и осуществлять взаимодействие между объектами системы и записями в базе данных. В данном разделе будет описан маппинг основных сущностей: Client, Car и Parts.

2.1 Сущность Client

Сущность Client представляет клиентов Магазина автозапчастей и включает основную информацию о них.

package CourseworkWorkAutoParts.model;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

@Table(name = "clients")

public class Client {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "first\_name", nullable = false)

private String firstName;

@Column(name = "phone", nullable = false)

private String phone;

@Column(name = "email", nullable = false)

private String email;

@Column(name = "delivery\_address")

private String delivery\_address;

}

Маппинг полей Client:

@Entity и @Table(name = "clients"): обозначает класс как сущность JPA и связывает его с таблицей clients.

@Id и @GeneratedValue(strategy = GenerationType.IDENTITY): обозначает первичный ключ и стратегию генерации значения.

@Column: определяет соответствие полей класса столбцам таблицы.

2.2 Сущность Parts

Сущность Parts представляет автозапчасти, которые клиенты заказали.

package CourseworkWorkAutoParts.model;

import jakarta.persistence.\*;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.AllArgsConstructor;

import java.util.List;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

@Table(name = "parts")

public class Parts {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "client\_id", nullable = false)

private Client client;

@Column(name = "type\_parts", nullable = false)

private String type\_parts;

@Column(name = "type\_cars", nullable = false)

private String type\_cars;

@Column(name = "year\_cars", nullable = false)

private int year\_cars;

@Column(name = "type\_engine", nullable = false)

private String type\_engine;

@OneToMany(mappedBy = "parts", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Parts> orders;

}

Маппинг полей Car:

@Entity и @Table(name = "cars"): обозначает класс как сущность JPA и связывает его с таблицей cars.

@Id и @GeneratedValue(strategy = GenerationType.IDENTITY): обозначает первичный ключ и стратегию генерации значения.

@Column: определяет соответствие полей класса столбцам таблицы.

@ManyToOne и @JoinColumn(name = "client\_id", nullable = false): устанавливает связь "многие к одному" с сущностью Client.

@OneToMany(mappedBy = "parts", cascade = CascadeType.ALL, orphanRemoval = true): устанавливает связь "один ко многим" с сущностью Order.

2.3 Сущность Order

Сущность Order представляет заказы на автозапчасти.

package CourseworkWorkAutoParts.model;

import jakarta.persistence.\*;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.AllArgsConstructor;

import java.util.Date;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

@Table(name = "orders")

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "client\_id", nullable = false)

private Client client;

@ManyToOne

@JoinColumn(name = "parts\_id", nullable = false)

private Parts parts;

@Column(name = "order\_date", nullable = false)

private Date orderDate;

@Column(name = "status", nullable = false)

private String status;

@Column(name = "services\_and\_parts", columnDefinition = "TEXT")

private String servicesAndParts;

}

Маппинг полей Order:

@Entity и @Table(name = "orders"): обозначает класс как сущность JPA и связывает его с таблицей orders.

@Id и @GeneratedValue(strategy = GenerationType.IDENTITY): обозначает первичный ключ и стратегию генерации значения.

@Column: определяет соответствие полей класса столбцам таблицы.

@ManyToOne и @JoinColumn(name = "car\_id", nullable = false): устанавливает связь "многие к одному" с сущностью Car.

Маппинг сущностей Client, Parts и Order к соответствующим таблицам в базе данных позволяет эффективно управлять данными и выполнять операции CRUD (создание, чтение, обновление, удаление). Эти сущности взаимосвязаны, что обеспечивает целостность данных и удобство работы с системой. Правильная конфигурация маппинга является ключевым элементом в построении надежной и масштабируемой системы управления станцией технического обслуживания.

**3. Описание точек доступа**

В данном разделе описаны точки доступа (endpoints) для взаимодействия с информационной системой магазина автозапчастей. Каждая точка доступа имеет уникальное название, метод запроса (GET, POST, PUT, DELETE), URL-адрес и, при необходимости, параметры запроса и тело запроса.

3.1 Клиенты (Client)

1. Получить всех клиентов

URL: /api/client

Метод: GET

Описание: Возвращает список всех клиентов.

Ответ:

[

{

"id": 1,

"firstName": "Антон",

"phone": "+375296789471",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

...

]

2.Получить клиента по ID

URL: /api/client/{id}

Метод: GET

Описание: Возвращает клиента по его ID.

Ответ:

{

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

3.Создать нового клиента

URL: /api/client

Метод: POST

Описание: Создает нового клиента.

Запрос:

{

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

Ответ:

{

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

4.Обновить информацию о клиенте

URL: /api/client/{id}

Метод: PUT

Описание: Обновляет информацию о существующем клиенте.

Запрос:

{

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

Ответ:

{

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

5.Удалить клиента

URL: /api/client/{id}

Метод: DELETE

Описание: Удаляет клиента по его ID.

Ответ: HTTP 204 No Content

3.2 Автозапчасти (Parts)

1.Получить данные об автозапчастях

URL: /api/parts

Метод: GET

Описание: Возвращает список всех автозапчастей.

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

...

]

2.Получить автозапчасть по ID

URL: /api/ parts /{id}

Метод: GET

Описание: Возвращает автозапчасть по его ID.

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

}

3.Создать новый запчастей

URL: /api/ parts

Метод: POST

Описание: Создает новый заказ на автозапчасть.

Запрос:

{

"client": {

"id": 1

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

}

4.Обновить информацию об автозапчастях

URL: /api/ parts /{id}

Метод: PUT

Описание: Обновляет информацию о существующем запчасти.

Запрос:

{

"client": {

"id": 1

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

}

5.Удалить запчасть

URL: /api/ parts /{id}

Метод: DELETE

Описание: Удаляет заказ по его ID.

Ответ: HTTP 204 No Content

6. Получить все заказы, принадлежащие клиенту по ID клиента

URL: /api/ parts /client/{clientId}

Метод: GET

Описание: Возвращает список всех запчастей, принадлежащих клиенту по его ID.

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

...

]

7.Получить все заказы, принадлежащие клиенту по ID клиента

URL: /api/ parts /client /{clientId}

Метод: GET

Описание: Возвращает список всех запчастей, принадлежащих клиенту по его ID.

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

...

]

3.3 Заказы (Orders)

1.Получить все заказы

URL: /api/orders

Метод: GET

Описание: Возвращает список всех заказов.

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"parts": {

"id": 1,

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

"orderDate": "2024-05-23",

"status": "Отправлен",

"servicesAndParts": "Тормозные диски (Комплект, 1 шт) - 70 руб"

},

...

]

2.Получить заказ по ID

URL: /api/orders/{id}

Метод: GET

Описание: Возвращает заказ по его ID.

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"parts": {

"id": 1,

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

"orderDate": "2024-05-23",

"status": "Отправлен",

"servicesAndParts": "Тормозные диски (Комплект, 1 шт) - 70 руб"

}

3.Создать новый заказ

URL: /api/orders

Метод: POST

Описание: Создает новый заказ.

Запрос:

{

"client": {

"id": 1

},

"parts": {

"id": 1

},

"orderDate": "2024-05-23",

"status": "Отправлен",

"servicesAndParts": "Тормозные диски (Комплект, 1 шт) - 70 руб"

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"vehicle": {

"id": 1,

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

"orderDate": "2024-05-23",

"status": "Отправлен",

"servicesAndParts": "Тормозные диски (Комплект, 1 шт) - 70 руб"

}

4.Обновить информацию о заказе

URL: /api/orders/{id}

Метод: PUT

Описание: Обновляет информацию о существующем заказе.

Запрос:

{

"client": {

"id": 1

},

"vehicle": {

"id": 1

},

"orderDate": "2024-05-23",

"status": "Доставлен",

"servicesAndParts": "Тормозные диски (Комплект, 1 шт) - 70 руб"

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Антон",

"phone": "+375291234567",

"email": "petrov@gmail.com",

" deliveryAddress": ". г.Витебск ул.Фрунзе дом 102"

},

"vehicle": {

"id": 1,

"typeParts": "Тормозные\_диски",

"typeCars": "BMW",

"yearCars": 2000,

"typeEngine": 1.8tdi,

},

"orderDate": "2024-05-23",

"status": "Доставлен",

"servicesAndParts": "Тормозные диски (Комплект, 1 шт) - 70 руб"

}

5.Удалить заказ

URL: /api/orders/{id}

Метод: DELETE

Описание: Удаляет заказ по его ID.

Ответ: HTTP 204 No Content

Эти точки доступа обеспечивают полное управление данными клиентов, запчастей и заказов в автосервисе.

**4. Разработка REST-сервиса**

4.1 Структура проекта

При разработке интернет магазина автозапчастей на базе Spring Boot и JPA важно придерживаться организованной структуры проекта. Это способствует удобству разработки, поддержки и масштабируемости системы.

1. Главный класс - AutoPartsMain.java

Главный класс, который запускает Spring Boot приложение. Этот класс содержит метод main, который инициализирует и запускает встроенный веб-сервер.

2. Контроллеры (Controller)

Пакет controller содержит REST-контроллеры, которые обрабатывают HTTP-запросы, поступающие от клиентов (например, из веб-браузера или других приложений).

ClientController.java и ClientControllerImpl.java

Интерфейс и реализация контроллера для управления клиентами. Эти классы отвечают за обработку запросов, связанных с созданием, обновлением, удалением и получением данных о клиентах.

PartsController.java и PartsControllerImpl.java

Интерфейс и реализация контроллера для управления запчастями. Эти классы обрабатывают запросы, связанные с запчастями, такие как заказ новых запчастей, обновление информации о них, удаление и получение данных.

OrderController.java и OrderControllerImpl.java

Интерфейс и реализация контроллера для управления заказами. Эти классы отвечают за обработку запросов, связанных с созданием, обновлением, удалением и получением информации о заказах.

3. Объекты передачи данных (DTO)

Пакет dto содержит классы, используемые для передачи данных между различными слоями приложения (Data Transfer Objects).

Client.java

DTO для клиента. Этот класс используется для передачи данных о клиентах между слоем представления и бизнес-логикой.

Parts.java

DTO для запчастей. Этот класс служит для передачи данных о запчастях между различными компонентами системы.

Order.java

DTO для заказа. Этот класс используется для передачи данных о заказах между слоем представления и бизнес-логикой.

4. Сущности (Entities)

Пакет model содержит JPA-сущности, которые представляют таблицы в базе данных.

Client.java

Сущность клиента. Этот класс маппит данные клиента на соответствующую таблицу в базе данных.

Parts.java

Сущность запчасти. Этот класс маппит данные запчастей на таблицу в базе данных.

Order.java

Сущность заказа. Этот класс маппит данные заказа на таблицу в базе данных.

5. Репозитории (Repositories)

Пакет repository содержит интерфейсы репозиториев, которые предоставляют методы для выполнения операций с базой данных.

ClientRepository.java

Репозиторий для клиентов. Этот интерфейс наследуется от JpaRepository и предоставляет методы для работы с данными клиентов в базе данных.

PartsRepository.java

Репозиторий для запчастей. Этот интерфейс предоставляет методы для выполнения CRUD операций с данными запчастей.

OrderRepository.java

Репозиторий для заказов. Этот интерфейс предоставляет методы для выполнения CRUD операций с данными заказов.

6. Конфигурация (Configuration)

Пакет config содержит конфигурационные классы, необходимые для настройки веб-приложения.

7. Ресурсы (Resources)

Папка resources содержит конфигурационные файлы и скрипты для инициализации базы данных.

application.properties

Файл конфигурации Spring Boot. Содержит настройки для подключения к базе данных, настройки логирования и другие параметры конфигурации приложения.

4.2 Запуск проекта

Запуск проекта Spring Boot для магазина автозапчастей включает в себя несколько шагов, которые необходимо выполнить для корректного развертывания и запуска приложения. В данном разделе описаны все необходимые шаги для подготовки и запуска проекта.

1. Установка необходимых инструментов

Для запуска проекта вам потребуются следующие инструменты:

* Java Development Kit (JDK) 17 или выше
* Maven. Инструмент для управления зависимостями и сборки проекта.
* PostgreSQL База данных, используемая в проекте.

2. Настройка базы данных

Создание базы данных: Создайте новую базу данных для проекта в PostgreSQL.

Настройка application.properties. Необходимо убедится, что файл application.properties содержит правильные настройки для подключения к базе данных.

3. Сборка и запуск проекта

Клонирование репозитория: Клонируйте репозиторий проекта на ваш локальный компьютер.

Сборка проекта с помощью Maven: Выполните команду для сборки проекта и загрузки всех необходимых зависимостей.

mvn clean install

Запуск приложения: Запустите приложение с помощью Maven или непосредственно через ваш IDE (например, IntelliJ IDEA или Eclipse).

mvn spring-boot:run

Либо запустите PartsApplication.java как Java-приложение из вашей IDE.

4. Доступ к API

После успешного запуска приложения, API будет доступен по адресу http://localhost:8080. Вы можете использовать инструменты для тестирования API, такие как Postman или curl, чтобы взаимодействовать с вашим приложением.

5. Завершение работы

Для остановки приложения выполните команду Ctrl + C в терминале, где оно запущено, либо завершите процесс через вашу IDE.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной курсовой работы была разработана информационная система магазин автозапчастей с использованием технологий Spring Boot, Spring Data JPA и PostgreSQL. Данная система позволяет эффективно управлять клиентами, запчастями и их заказами, предоставляя удобный интерфейс для взаимодействия с данными через RESTful API.

Основные достижения курсовой работы включают:

1. Анализ предметной области и постановка задачи. Были изучены основные процессы, заказа атозапчастей , определены ключевые требования к системе и сформулирована цель разработки.

2. Проектирование базы данных: Разработана и реализована структура базы данных, включающая таблицы для хранения информации о клиентах, запчастях и заказах. Были использованы связи между таблицами для обеспечения целостности данных.

3. Разработка RESTful API: Созданы контроллеры, сервисы и репозитории, обеспечивающие выполнение CRUD-операций (создание, чтение, обновление, удаление) для основных сущностей системы. API предоставляет возможность взаимодействия с системой через стандартные HTTP-запросы.

Разработанная система является надежным и эффективным инструментом для управления станцией технического обслуживания и может служить основой для дальнейшего развития и модернизации.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Документация Spring Boot: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
2. Документация Spring Data JPA: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
3. Документация Hibernate: https://hibernate.org/orm/documentation/
4. Документация PostgreSQL: https://www.postgresql.org/docs/
5. Документация Maven: https://maven.apache.org/guides/
6. Документация RESTful Web Services: https://restfulapi.net/