**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 Car…………………………………………………………..10

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

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

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

3.2 Автомобили (Vehicles)………………………………………………...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**

В современном мире автомобиль стал неотъемлемой частью повседневной жизни. Обладатели автомобилей полагаются на свои транспортные средства для выполнения множества задач, начиная от ежедневных поездок на работу и заканчивая долгими путешествиями. В связи с этим, обеспечение надежности и безопасности автомобиля является важнейшей задачей для каждого автовладельца. Здесь на помощь приходят станции технического обслуживания (СТО), которые предоставляют широкий спектр услуг по ремонту и обслуживанию автомобилей.

СТО играют ключевую роль в поддержании работоспособности транспортных средств. Их деятельность охватывает разнообразные аспекты, начиная от простых процедур, таких как замена масла и заканчивая сложными ремонтами двигателей.

Целью данной курсовой работы является разработка информационной системы для управления деятельностью станции технического обслуживания. Основными задачами проекта являются автоматизация процессов записи клиентов на обслуживание, ведение учета выполненных работ и заказанных запчастей, а также управление базой данных клиентов и их транспортных средств. В рамках работы будут рассмотрены методы и средства, используемые для создания системы, начиная от моделирования структуры базы данных и заканчивая реализацией интерфейсов для взаимодействия с пользователями.

Создание такой системы позволит значительно повысить эффективность работы СТО, улучшить качество обслуживания и снизить вероятность ошибок. Автоматизация процессов позволит сотрудникам станции сосредоточиться на выполнении непосредственных задач по ремонту и обслуживанию автомобилей, что в конечном итоге приведет к повышению уровня удовлетворенности клиентов.

Введение информационных технологий в работу станции технического обслуживания открывает новые возможности для оптимизации процессов, повышения прозрачности и улучшения взаимодействия с клиентами. Данная курсовая работа направлена на изучение этих аспектов и разработку практических решений, которые могут быть применены в реальной деятельности СТО.

**1. Описание предметной области**

1.1 Назначение программного продукта

Программный продукт, разработанный в рамках данной курсовой работы, представляет собой систему управления станцией технического обслуживания (СТО). Основное назначение системы – автоматизация процессов, связанных с обслуживанием автомобилей, улучшение качества обслуживания клиентов и повышение эффективности работы сотрудников СТО. Данный программный продукт разработан для того, чтобы обеспечить удобное и эффективное управление заказами на обслуживание, учет клиентов и их автомобилей, а также контроль за выполнением ремонтных работ.

1.2 Основные цели и задачи программного продукта

1. Автоматизация процессов обслуживания:

* Обеспечение автоматического создания и ведения заказов на обслуживание.
* 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 обращений.
* Учет автомобилей, принадлежащих клиентам, и истории их обслуживания.
* Возможность отправки уведомлений клиентам о статусе их заказа и необходимости планового обслуживания.

3. Оптимизация работы сотрудников:

* Управление расписанием и загрузкой сотрудников.
* Учет выполненных работ и использование запасных частей.
* Обеспечение доступа к необходимой информации для выполнения качественного ремонта.

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 клиентами. Ведение базы данных клиентов, поиск и фильтрация по различным параметрам, добавление и редактирование информации о клиентах.

Управление автомобилями. Учет автомобилей, принадлежащих клиентам, хранение информации о марке, модели, VIN-коде и прочих характеристиках.

Создание и ведение заказов. Оформление заказов на обслуживание, добавление информации о необходимых работах и запасных частях, контроль за статусом выполнения заказа.

1.4 Преимущества использования программного продукта

Повышение эффективности работы СТО. Автоматизация рутинных задач позволяет сократить время на их выполнение и уменьшить количество ошибок.

Улучшение качества обслуживания. Полная информация о клиенте и его автомобиле позволяет персонализировать обслуживание и своевременно информировать клиента о необходимости планового обслуживания.

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

Программный продукт для управления станцией технического обслуживания является комплексным решением, направленным на улучшение всех аспектов работы СТО, 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 система управления станцией технического обслуживания (СТО) 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 Order.

2.1 Сущность Client

Сущность Client представляет клиентов СТО и включает основную информацию о них.

package by.vstu.car.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 = "last\_name", nullable = false)

private String lastName;

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

private String phone;

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

private String email;

@Column(name = "address")

private String address;

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

private List<Car> cars;

}

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.

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

2.2 Сущность Car

Сущность Car представляет автомобили, принадлежащие клиентам СТО.

package by.vstu.car.model;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

@Table(name = "cars")

public class Car {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

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

private String make;

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

private String model;

@Column(name = "vin", nullable = false, unique = true)

private String vin;

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

private int year;

@ManyToOne

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

private Client client;

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

private List<Order> 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 = "car", cascade = CascadeType.ALL, orphanRemoval = true): устанавливает связь "один ко многим" с сущностью Order.

2.3 Сущность Order

Сущность Order представляет заказы на обслуживание автомобилей.

package by.vstu.car.model;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

@Table(name = "orders")

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

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

private LocalDate orderDate;

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

private String status;

@Column(name = "description")

private String description;

@ManyToOne

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

private Car car;

}

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, Car 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 станции технического обслуживания. 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 (Clients)

1. Получить всех клиентов

URL: /api/clients

Метод: GET

Описание: Возвращает список всех клиентов.

Ответ:

[

{

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

...

]

2.Получить клиента по ID

URL: /api/clients/{id}

Метод: GET

Описание: Возвращает клиента по его ID.

Ответ:

{

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

}

3.Создать нового клиента

URL: /api/clients

Метод: POST

Описание: Создает нового клиента.

Запрос:

{

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

}

Ответ:

{

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

}

4.Обновить информацию о клиенте

URL: /api/clients/{id}

Метод: PUT

Описание: Обновляет информацию о существующем клиенте.

Запрос:

{

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

}

Ответ:

{

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

}

5.Удалить клиента

URL: /api/clients/{id}

Метод: DELETE

Описание: Удаляет клиента по его ID.

Ответ: HTTP 204 No Content

3.2 Автомобили (Vehicles)

1.Получить все автомобили

URL: /api/vehicles

Метод: GET

Описание: Возвращает список всех автомобилей.

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

...

]

2.Получить автомобиль по ID

URL: /api/vehicles/{id}

Метод: GET

Описание: Возвращает автомобиль по его ID.

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

}

3.Создать новый автомобиль

URL: /api/vehicles

Метод: POST

Описание: Создает новый автомобиль.

Запрос:

{

"client": {

"id": 1

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

}

4.Обновить информацию об автомобиле

URL: /api/vehicles/{id}

Метод: PUT

Описание: Обновляет информацию о существующем автомобиле.

Запрос:

{

"client": {

"id": 1

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

}

5.Удалить автомобиль

URL: /api/vehicles/{id}

Метод: DELETE

Описание: Удаляет автомобиль по его ID.

Ответ: HTTP 204 No Content

6. Получить все автомобили, принадлежащие клиенту по ID клиента

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

Метод: GET

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

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

...

]

7.Получить все автомобили, принадлежащие клиенту по ID клиента

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

Метод: GET

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

Ответ:

[

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

...

]

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": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"vehicle": {

"id": 1,

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

"orderDate": "2024-05-01",

"status": "Выполнен",

"servicesAndParts": "Замена масла (Масляный фильтр, 1 шт) - 1500 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": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"vehicle": {

"id": 1,

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

"orderDate": "2024-05-01",

"status": "Выполнен",

"servicesAndParts": "Замена масла (Масляный фильтр, 1 шт) - 1500 руб"

}

3.Создать новый заказ

URL: /api/orders

Метод: POST

Описание: Создает новый заказ.

Запрос:

{

"client": {

"id": 1

},

"vehicle": {

"id": 1

},

"orderDate": "2024-05-01",

"status": "В процессе",

"servicesAndParts": "Замена масла (Масляный фильтр, 1 шт) - 1500 руб"

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"vehicle": {

"id": 1,

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

"orderDate": "2024-05-01",

"status": "В процессе",

"servicesAndParts": "Замена масла (Масляный фильтр, 1 шт) - 1500 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": "2024-05-01",

"status": "Выполнен",

"servicesAndParts": "Замена масла (Масляный фильтр, 1 шт) - 1500 руб"

}

Ответ:

{

"id": 1,

"client": {

"id": 1,

"firstName": "Иван",

"lastName": "Иванов",

"phone": "+79161234567",

"email": "ivanov@mail.com",

"address": "ул. Ленина, д. 1"

},

"vehicle": {

"id": 1,

"make": "Toyota",

"model": "Camry",

"year": 2015,

"vin": "JTDBE32K500123456",

"licensePlate": "А123ВЕ77"

},

"orderDate": "2024-05-01",

"status": "Выполнен",

"servicesAndParts": "Замена масла (Масляный фильтр, 1 шт) - 1500 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, автомобилей и заказов в автосервисе.

**4. Разработка REST-сервиса**

4.1 Структура проекта

При разработке системы управления станцией технического обслуживания (СТО) на базе Spring Boot и JPA важно придерживаться организованной структуры проекта. Это способствует удобству разработки, поддержки и масштабируемости системы.

1. Главный класс - CarApplication.java

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

2. Контроллеры (Controllers)

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.

CarController.java и CarControllerImpl.java

Интерфейс и реализация контроллера для управления автомобилями. Эти классы обрабатывают запросы, связанные с автомобилями, такие как добавление новых автомобилей, 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.

Car.java

DTO для автомобиля. Этот класс служит для передачи данных об автомобилях 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.

Car.java

Сущность автомобиля. Этот класс маппит данные автомобиля 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.

CarRepository.java

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

OrderRepository.java

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

6. Сервисы (Services)

Пакет service содержит интерфейсы и реализации бизнес-логики приложения.

ClientService.java и ClientServiceImpl.java

Интерфейс и реализация сервиса для управления клиентами. Эти классы содержат бизнес-логику для операций над клиентами, таких как добавление, обновление, удаление и получение данных.

CarService.java и CarServiceImpl.java

Интерфейс и реализация сервиса для управления автомобилями. Эти классы содержат бизнес-логику для операций над автомобилями.

OrderService.java и OrderServiceImpl.java

Интерфейс и реализация сервиса для управления заказами. Эти классы содержат бизнес-логику для операций над заказами.

7. Конфигурация (Configuration)

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

8. 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 системы управления станцией технического обслуживания (СТО) 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 CarApplication.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 для управления станцией технического обслуживания (СТО) с использованием технологий Spring Boot, Spring Data JPA и PostgreSQL. Данная система позволяет эффективно управлять клиентами, автомобилями и 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, происходящие на станции технического обслуживания, 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, автомобилях 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 Lombok: https://projectlombok.org/
6. Документация Maven: https://maven.apache.org/guides/
7. Документация RESTful Web Services: https://restfulapi.net/