**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……………………………………………………………………….7

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

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

2.1 Сущность Customer……………………………………………………………………………9

2.2 Сущность Car…………………………………………………………………….…………..10

2.3 Сущность Sale………………………………………..………………..11

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

3.1 Клиенты (Customer)………………………………………………………………………...13

3.2 Автомобили (Car)………………………………………...…………...15

3.3 Продажи (Sale)………………………………………………………...17

4. Разработка REST-сервиса…………………………………………………………………………....21

4.1 Структура проекта…………………………………………………….21

4.2 Запуск проекта……………………………………………………………………………23

Заключение………………………………………………………………...25

Список использованных источников…..………………………………...26

**ВВЕДЕНИЕ**

В современном автомобильном рынке важным элементом становится цифровизация бизнеса автосалонов. Создание REST API для автосалона представляет собой ключевой шаг в направлении улучшения онлайн-сервисов и повышения доступности информации для клиентов. Данный курсовой проект нацелен на разработку и внедрение REST API для автосалона, обеспечивающего клиентам и сотрудникам удобный доступ к информации о наличии автомобилей, их технических характеристиках, ценах и условиях покупки. Целью проекта является создание эффективного интерфейса, который позволит клиентам быстро получать актуальную информацию о доступных автомобилях, а также обеспечит сотрудникам автосалона инструменты для управления продуктами и заказами. Разработка REST API для автосалона способствует повышению конкурентоспособности бизнеса, улучшению клиентского опыта и оптимизации процессов продажи автомобилей через онлайн-каналы.

**1. Описание предметной области**

1.1 Назначение программного продукта

Цель разработки REST API для автосалона - обеспечить удобный и эффективный способ взаимодействия между базой данных автосалона и внешними системами или клиентскими приложениями. API позволяет автоматизировать процессы управления данными, обеспечивает простой доступ к информации о доступных автомобилях, продажах, клиентах и прочих данных автосалона. Это помогает улучшить сервис, оптимизировать процессы продаж, анализировать данные и управлять бизнесом более эффективно. REST API также упрощает интеграцию с другими системами, что позволяет автосалону быть более конкурентоспособным и эффективным на рынке.

1.2 Задачи программного продукта

В рамках данного проекта необходимо разработать важные функции для работы приложения. Были выделены основные из них:

* Предоставление доступа к базе данных автосалона для получения информации о доступных автомобилях, их характеристиках, ценах и прочих данных.
* Возможность добавления, редактирования и удаления данных об автомобилях, заказах, клиентах и других объектах автосалона.

Программное решение будет способствовать повышению эффективности работы автосалона, улучшению обслуживания клиентов и расширению возможностей для развития бизнеса.

1.3. Источники входной информации

Для разработки информационной системы автосалон были использованы следующие источники входной информации:

* Справочная литература: В процессе подготовки курсовой работы были изучена бизнес модель автосалона, а также литература по технологиям разработки программного обеспечения и базам данных. Эти источники позволили получить представление о требованиях к функциональности и управлению автосалона.
* Анализ конкурентов и существующих решений. Был проведен анализ конкурентов и изучены существующие информационные системы для автосалона.

Эти источники информации стали основой для определения функциональных требований к разрабатываемой информационной системе и обеспечили соответствие сервиса потребностям пользователей и бизнес-целям проекта.

1.4. Выходная информация

Информационная система автосалон предоставляет разнообразную выходную информацию, необходимую для эффективного управления и принятия стратегических решений. Вот основные категории выходной информации:

Отчеты и статистика:

Статистические данные о продажах: отчеты о количестве заказанных автомобилей определённой марки и модели, обороте и популярности определенных марок автомобилей.

Управление персоналом: возможность добавления новых сотрудников, изменения их ролей и доступа к системе.

**2. Описание маппинга**

Приложение для автосалона разрабатывается с использованием технологии Node.js для серверной части и PostgreSQL в качестве системы управления базами данных. Целью приложения является автоматизация управления продажами, отслеживание популярных марок и моделей автомобилей и взаимодействием с клиентами, обеспечивая эффективное функционирование автосалона и привлечения новых клиентов.

2.1 Сущность Customer

Сущность Customer представляет клиентов автосалона и включает основную информацию о них.

package by.vstu.avtosalon.model;  
  
import jakarta.persistence.\*;  
import lombok.Data;  
  
@Data  
@Entity  
@Table(name = "customers")  
public class Customer {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long customerId;  
  
 @Column(nullable = false)  
 private String firstName;  
  
 @Column(nullable = false)  
 private String lastName;  
  
 @Column  
 private String phone;  
  
 @Column  
 private String email;  
  
 @Column  
 private String address;  
}

Маппинг полей Client:

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

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

@Column: определяет соответствие полей класса столбцам таблицы.

2.2 Сущность Car

Сущность Car представляет автомобили, для продажи клиентам автосалона.

package by.vstu.avtosalon.model;  
  
import jakarta.persistence.\*;  
  
import lombok.Data;  
@Data  
@Entity  
@Table(name = "cars")  
public class Car {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long carId;  
  
 @Column(nullable = false)  
 private String make;  
  
 @Column(nullable = false)  
 private String model;  
  
 @Column(nullable = false)  
 private int year;  
  
 @Column(nullable = false, unique = true)  
 private String vin;  
  
 @Column(nullable = false)  
 private double price;  
  
 @Column(nullable = false)  
 private String color;  
  
 @Column(nullable = false)  
 private String status;  
}

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.

2.3 Сущность Sale

Сущность Sale представляет продажи автомобилей клиенту.

package by.vstu.avtosalon.model;  
  
import jakarta.persistence.\*;  
import lombok.Data;  
import java.time.LocalDate;  
  
@Data  
@Entity  
@Table(name = "sales")  
public class Sale {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long saleId;  
  
 @ManyToOne  
 @JoinColumn(name = "carId", nullable = false)  
 private Car car;  
  
 @ManyToOne  
 @JoinColumn(name = "customerId", nullable = false)  
 private Customer customer;  
  
 @Column(nullable = false)  
 private LocalDate saleDate;  
  
 @Column(nullable = false)  
 private double salePrice;  
}

Маппинг полей Order:

@Entity и @Table(name = "sale"): 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 = "carId", nullable = false): устанавливает связь "многие к одному" с сущностью Car.

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

Маппинг сущностей Customer, Car и Sale 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 автосалона. 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 (Customer)

1. Получить всех клиентов

URL: /api/customer

Метод: GET

Описание: Возвращает список всех клиентов.

Ответ:

[

{

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

},

...

]

2.Получить клиента по ID

URL: /api/customer/{id}

Метод: GET

Описание: Возвращает клиента по его ID.

Ответ:

{

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

}

3.Создать нового клиента

URL: /api/customer

Метод: POST

Описание: Создает нового клиента.

Запрос:

{

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

}

Ответ:

{

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

}

4.Обновить информацию о клиенте

URL: /api/customer/{id}

Метод: PUT

Описание: Обновляет информацию о существующем клиенте.

Запрос:

{

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

}

Ответ:

{

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

}

5.Удалить клиента

URL: /api/customer/{id}

Метод: DELETE

Описание: Удаляет клиента по его ID.

Ответ: HTTP 204 No Content

3.2 Автомобили (Car)

1.Получить все автомобили

URL: /api/car

Метод: GET

Описание: Возвращает список всех автомобилей.

Ответ:

{

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

},

2.Получить автомобиль по ID

URL: /api/car/{id}

Метод: GET

Описание: Возвращает автомобиль по его ID.

Ответ:

{

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

}

3.Создать новый автомобиль

URL: /api/car

Метод: POST

Описание: Создает новый автомобиль.

Запрос:

{

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

}

Ответ:

{

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

}

4.Обновить информацию об автомобиле

URL: /api/car/{id}

Метод: PUT

Описание: Обновляет информацию о существующем автомобиле.

Запрос:

{

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

}

Ответ:

{

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

}

5.Удалить автомобиль

URL: /api/car/{id}

Метод: DELETE

Описание: Удаляет автомобиль по его ID.

Ответ: HTTP 204 No Content

3.3 Продажи (Sales)

1.Получить все продажи

URL: /api/sales

Метод: GET

Описание: Возвращает список всех продаж.

Ответ:

[

{

"id": 2,

"customers": {

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

},

"cars": {

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

},

"SaleDate": "2005-05-01",

"SalePrice": "65000.00",

},

...

]

2.Получить продажи по ID

URL: /api/sales/{id}

Метод: GET

Описание: Возвращает продажи по его ID.

Ответ:

{

"id": 2,

"customers": {

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

},

"cars": {

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

},

"SaleDate": "2005-05-01",

"SalePrice": "65000.00",

},

3.Создать новую продажу

URL: /api/sales

Метод: POST

Описание: Создает новую продажу.

Запрос:

{

"customers": {

"id": 1

},

"cars": {

"id": 1

},

"SaleDate": "2008-05-01",

"SalePrice": "65000.00",

}

Ответ:

{

"id": 2,

"customers": {

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

},

"cars": {

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

},

"SaleDate": "2008-05-01",

"SalePrice": "65000.00",

}

4.Обновить информацию о продажи

URL: /api/sales/{id}

Метод: PUT

Описание: Обновляет информацию о существующей продаже.

Запрос:

{

"customers": {

"id": 1

},

"cars": {

"id": 1

},

"SaleDate": "2008-05-01",

"SalePrice": "65000.00",

}

Ответ:

{

"id": 2,

"customers": {

"id": 2,

"firstName": "Пётр",

"lastName": "Петров",

"phone": "+375442134456",

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

"address": "ул. Слоленская, д. 8к2"

},

"cars": {

"id": 1,

"make": "Nissan",

"model": "Primera p11",

"year": 2001,

"vin": "VSS7KZ12399832145",

"price": "65000.00"

"color": "синий"

"status": "в наличие"

},

"SaleDate": "2008-05-01",

"SalePrice": "65000.00",

}

5.Удалить продажу

URL: /api/sales/{id}

Метод: DELETE

Описание: Удаляет продажу по его ID.

Ответ: HTTP 204 No Content

Эти точки доступа обеспечивают полное управление данными клиентов, автомобилей и продаж в автосалоне.

**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 - AutoSalonApplication.java

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

2. Контроллеры (Controller)

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

CustomerController.java и CustomerControllerImpl.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.

SaleController.java и SaleControllerImpl.java

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

3. Объекты передачи данных (DTO)

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

Customer.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.

Sale.java

DTO для заказа. Этот класс используется для передачи данных о продажах 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.

Sale.java

Сущность продажи. 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.

CustomerRepository.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.

SaleRepository.java

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

6. Сервисы (Services)

Пакет service содержит интерфейсы и реализации бизнес-логики приложения.

CustomerService.java и CustomerServiceImpl.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.

SaleService.java и SaleServiceImpl.java

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

7. Конфигурация (Configuration)

Пакет JpaConfig 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 автосалона 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 AutoSalonApplication.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**

Целью проекта было создание эффективного и удобного инструмента для управления бизнес-процессами автосалона, а также обеспечение удобства для клиентов при выборе и покупке автомобилей. В результате выполнения проекта были достигнуты следующие результаты:

1. Разработана база данных, содержащая информацию о доступных автомобилях, их характеристиках, ценах, а также информацию о клиентах автосалона.

2. Реализован функционал, позволяющий сотрудникам автосалона осуществлять поиск и фильтрацию автомобилей по различным параметрам, проводить продажу автомобилей, а также вести учет клиентов и сделок.

3. Создан интерфейс для клиентов, позволяющий им просматривать доступные автомобили, сравнивать их характеристики, оставлять заявки на покупку и получать информацию о своих сделках. Информационная система автосалона предоставляет ряд преимуществ для его владельцев и клиентов. Она позволяет оптимизировать процессы продажи автомобилей, улучшить обслуживание клиентов и повысить эффективность работы сотрудников. Клиенты могут легко найти и сравнить автомобили, получить всю необходимую информацию о них и совершить покупку в удобное для них время. Таким образом, разработанная информационная система является полезным инструментом для автосалона, способствующим улучшению качества обслуживания клиентов и повышению эффективности бизнес-процессов.

**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/