Павел_МельникМетоды АПИ Заказы + DaData

Чеклист:

- Убедитесь что в разделе “Требования по интеграции с DaData“ есть мэппинг для входных данных
- Убедитесь что в разделе “Требования по интеграции с DaData“ есть мэппинг для выходных данных
- Убедитесь что в требованиях к бэку есть вызов метода DaData для стандартизации адреса

Работа с заказами

1. Создать заказ

Метод	POST /orders/
Описание	Создает новый заказ с проверкой и стандартизацией адреса через сервис DaData.
Входные данные	Объект Order, содержащий информацию о заказе, включая адрес.
Выходные данные	Возвращает объект Order, включая стандартизированный адрес и код проверки.
Коды ответов	• 201 Created – заказ успешно создан и адрес стандартизирован. • 400 Bad Request – неверные данные для создания заказа. • 500 Internal Server Error – ошибка сервера.
Требования по реализации	• Метод должен проверять и стандартизировать адрес через API DaData перед созданием заказа. • В случае возникновения ошибки при проверке адреса сервер должен вернуть соответствующий код ошибки и описание ошибки.

2. Получить данные о заказе

Метод	GET /orders/{order_id}
Описание	Возвращает данные конкретного заказа по его ID, включая проверенный и стандартизированный адрес.
Входные данные	нет
Выходные данные	Объект Order

Коды ответов	• 200 OK – Успешный ответ содержит данные заказа, включая стандартизированный адрес. • 404 Not Found – Заказ не найден. • 500 Internal Server Error – Ошибка сервера.
Требования по реализации	• Метод должен возвращать данные конкретного заказа, включая проверенный и стандартизированный адрес, по его ID из базы данных. • В случае ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

3. Обновить заказ

Метод	PUT /orders/{order_id}
Описание	Обновляет информацию о существующем заказе, включая повторную проверку и стандартизацию адреса через API DaData.
Входные данные	Объект Order
Выходные данные	Обновленный объект Order
Коды ответов	• 200 OK – Заказ успешно обновлен и адрес повторно стандартизирован. • 400 Bad Request – Неверные данные для обновления заказа. • 404 Not Found – Заказ не найден. • 500 Internal Server Error – Ошибка сервера.
Требования по реализации	• Метод должен обновлять информацию о существующем заказе, включая повторную проверку и стандартизацию адреса через API DaData. • В случае возникновения ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

4. Удалить заказ

Метод	DELETE /orders/{order_id}
Описание	Удаляет указанный заказ.
Входные данные	нет
Выходные данные	нет
Коды ответов	• 204 No Content – Заказ успешно удален. • 404 Not Found – Заказ не найден. • 500 Internal Server Error – Ошибка сервера.

Требования по реализации	• Метод должен удалять запись о заказе в базе данных. • В случае ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

Работа с товарами в заказе

1. Получить данные о товаре в заказе

Метод	GET /orders/{order_id}/items/{item_id}
Описание	Возвращает данные о конкретном товаре в заказе по его ID.
Входные данные	• `order_id` (int) – идентификатор заказа. • `item_id` (int) – идентификатор товара в заказе.
Выходные данные	Объект `OrderItem`, содержащий данные о товаре в заказе, включая название, количество, цену и т.д.
Коды ответов	• `200 OK` – Успешный ответ содержит данные о товаре в заказе. • `404 Not Found` – Товар в заказе не найден. • `500 Internal Server Error` – Ошибка сервера.
Требования по реализации	• Метод должен возвращать данные о конкретном товаре в заказе по его ID из базы данных. • В случае ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

2. Получить список товаров в заказе

Метод	GET /orders/{order_id}/items
Описание	Возвращает список всех товаров в заказе.
Входные данные	`order_id` (int) – идентификатор заказа.
Выходные данные	Массив объектов `OrderItem`, каждый из которых содержит информацию о товаре в заказе.
Коды ответов	• `200 OK` – Список товаров в заказе успешно возвращен. • `500 Internal Server Error` – Ошибка сервера.

Требования по реализации	1. Метод должен возвращать список всех товаров в заказе по указанному идентификатору заказа. 2. В случае ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

3. Добавить товар в заказ

Метод	POST /orders/{order_id}/items
Описание	Добавляет новый товар в заказ.
Входные данные	Объект `OrderItem`, содержащий данные о товаре (например, идентификатор товара, количество, цена).
Выходные данные	Возвращает объект `OrderItem`, включающий данные о добавленном товаре в заказ.
Коды ответов	• `201 Created` – Товар успешно добавлен в заказ. • `404 Not Found` – Заказ или товар не найдены в системе. • `500 Internal Server Error` – Ошибка сервера.
Требования по реализации	• Метод должен добавлять новый товар в заказ, обновляя информацию в базе данных. • В случае ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

4. Обновть данные о товаре в заказе

Метод	PUT /orders/{order_id}/items/{item_id}
Описание	Обновляет информацию о существующем товаре в заказе.
Входные данные	Объект `OrderItem` с обновленными данными (например, количество, цена).
Выходные данные	Обновленный объект `OrderItem`.
Коды ответов	• `200 OK` – Товар в заказе успешно обновлен. • `400 Bad Request` – Неверные данные для обновления товара в заказе. • `404 Not Found` – Товар в заказе не найден. • `500 Internal Server Error` – Ошибка сервера.

Требования по реализации	• Метод должен обновлять информацию о существующем товаре в заказе в базе данных. • В случае возникновения ошибки метод должен вернуть соответствующий код ошибки и описание ошибки.

5. Удалить товар из заказа

Метод	DELETE /orders/{order_id}/items/{item_id}
Описание	Удаляет указанный товар из заказа.
Входные данные	1. `order_id` (int) – идентификатор заказа. 2. `item_id` (int) – идентификатор товара в заказе.
Выходные данные	нет
Коды ответов	• `204 No Content` – Товар успешно удален из заказа. • `404 Not Found` – Товар в заказе не найден. • `500 Internal Server Error` – Ошибка сервера.
Требования по реализации	• Метод должен удалять запись о товаре в заказе в базе данных. • В случае ошибки сервер должен вернуть соответствующий код ошибки и описание ошибки.

Приложение 1. Мэппинг данных объектов JSON на модель БД

JSON Поле	Описание	Поле в БД	Тип данных
`order_id`	Идентификатор заказа	`id`	`INT`
`customer_name`	Имя клиента	`customer_name`	`VARCHAR`
`customer_phone`	Телефон клиента	`customer_phone`	`VARCHAR`
`delivery_address`	Адрес доставки	`delivery_address`	`TEXT`
`status`	Статус заказа	`status`	`VARCHAR`
`total_price`	Общая сумма заказа	`total_price`	`DECIMAL`
`created_at`	Дата создания заказа	`created_at`	`TIMESTAMP`

Приложение 2. Требования по интеграции с DaData

Заполнение входных данных

JSON Поле	Описание	Пример значения

`query`	Адрес, который нужно стандартизировать	`"ул. Ленина, д. 1"`

Обработка выходных данных

JSON Поле	Описание	Поле в БД	Тип данных
`result`	Стандартизованный адрес	`delivery_address`	`TEXT`
`qc`	Код проверки адреса	`address_quality_code`	`VARCHAR`
`unparsed_parts`	Необработанные части адреса	`unparsed_address_parts`	`TEXT`
`postal_code`	Почтовый индекс	`postal_code`	`VARCHAR`
`country`	Страна	`country`	`VARCHAR`
`region_with_type`	Регион с типом (область, край и т.д.)	`region_with_type`	`VARCHAR`
`city_with_type`	Город с типом (город, село и т.д.)	`city_with_type`	`VARCHAR`
`street_with_type`	Улица с типом (улица, проспект и т.д.)	`street_with_type`	`VARCHAR`
`house`	Номер дома	`house`	`VARCHAR`
`block`	Корпус, строение и т.д.	`block`	`VARCHAR`

Стандартизация адреса

Используемый метод	`POST https://cleaner.dadata.ru/api/v1/clean/address`
Краткое описание метода	Метод выполняет проверку и стандартизацию введенного адреса, возвращая его в структурированном виде.
Заполнение входных данных	Входные данные представляют собой JSON-объект, содержащий строку с адресом, введенным клиентом.
Обработка выходных данных	Выходные данные включают в себя стандартизированный адрес, а также код проверки, указывающий на успешность или наличие ошибок в адресе. Эти данные сохраняются в базе данных и возвращаются в ответе API при создании или обновлении заказа.
Документация	API: стандартизация адресов