Shop system technical specification

Strzechowski Konrad

June 2024

version 1.0

Contents

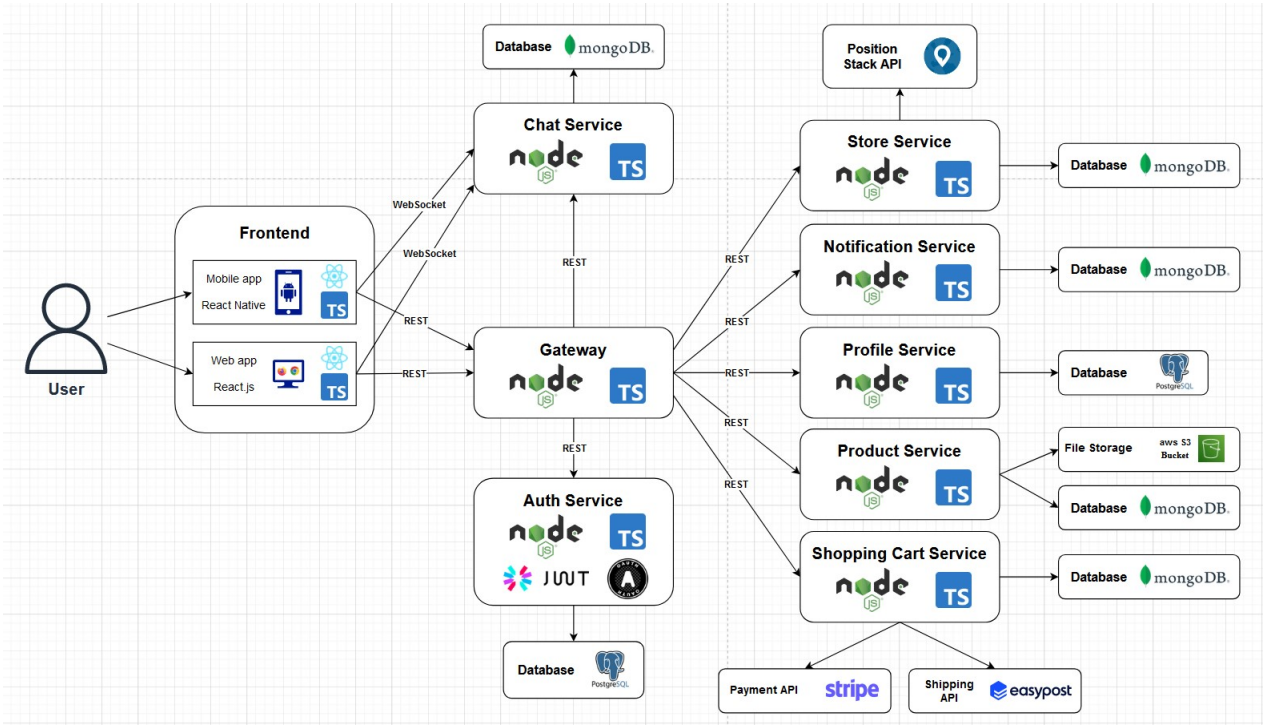
1 Abstract

This document contains the technical specification of a large-scale enterprise store system. The specification consists of a detailed discussion of the system architecture, highlighting its layers, nodes, modules, and main components. Each system module is thoroughly presented, including a description of its tasks and the technological requirements for implementation. Besides, the document presents the dependencies occurring between the modules and precisely specifies the ways and types of communication occurring in the system. Additionally, specification contains details about external interfaces. Finally, this document provides an approximation of the planned appearance of the application through detailed mockups of both the mobile and web versions. These mockups offer a visual representation of the user interface and user experience, illustrating the design, layout, and functionality that users can expect.

2 System architecture

The system consists of a backend built with a microservices architecture and two client applications: web and mobile. The backend is developed using the Node.js runtime and TypeScript language, but thanks to the microservices architecture, it is possible to use different languages and runtimes for future development as well. The web application is created using the React library, while the mobile app is implemented with React Native. This approach leverages the similarities between React and React Native, simplifying the development process and ensuring a consistent user experience across both platforms. All communication between services, except for chat, is implemented using standard REST APIs. Chats utilize WebSockets for real-time communication. Microservices provide the ability to easily scale and develop the application in the future, ensuring flexibility and adaptability as requirements evolve.

Figure 1: Overview diagram of the Store system architecture



2.1 Gateway

The Gateway is a single point of access for all clients, ensuring proper communication with each service and concealing implementation specifics from the frontend. It acts as a bridge between services, performing the necessary operations to efficiently fulfill requests. The Gateway API can easily combine data from different services to create the appropriate result needed by the user. It also provides security and performs the necessary authorization through close communication with the Auth service. It should also be noted that the gateway has no direct connection to any database and can be easily duplicated and used with a load balancer under heavy load.

2.2 Auth Service

The Auth API ensures a secure authentication and authorization process by utilizing OAuth 2.0 and JWT tokens. This enables an easy and secure connection between clients and other backend services. The Auth service is fully responsible for issuing, validating, and renewing tokens. It also connects to PostgreSQL datbase, which is essential for the proper management of user credentials.

2.3 Store Service

The Store service manages local stores and their locations by communicating with the external API, Position Stack. Position Stack provides a fast and straightforward geocoding solution, allowing us to map each address to precise coordinates, which enables users to view store locations on a map. Given the variability in location data, the Store service uses a non-relational MongoDB database. This choice allows for flexible data

management, accommodating changes and documents with different fields within a single collection.

2.4 Notification Service

The Notification Service is responsible for sending notifications to users. Its primary function is to distribute new marketing ads to email addresses. Additionally, it sends receipts or invoices after each purchase. The service also integrates closely with the mobile app through the gateway to deliver push notifications in real-time to users who have installed the app. To accommodate the evolving requirements, we have again chosen a MongoDB database for its flexibility and scalability.

2.5 Profile Service

The Profile service manages all user details, including data for customers, employees, and administrators. The Profile API utilizes a PostgreSQL database, as the clearly specified user account requirements make a relational database the best fit for ensuring data integrity and efficient querying.

2.6 Product Service

The Product service is currently the largest and most widely used API, responsible for managing all product data down to the smallest details. Due to the variability in product information, it utilizes a MongoDB database. The service employs text indexes and standard compound indexes to offer fast and accurate search and sorting capabilities, ensuring an efficient and responsive user experience. Additionally, the Product API leverages AWS S3 buckets to store and manage images for each product, ensuring reliable and scalable media storage.

2.7 Shopping Cart Service

The Shopping Cart Service is responsible for processing both online and local purchases. It works closely with the external Payment API, Stripe, and the Shipping API, EasyPost, to provide customers with various purchasing and shipping methods. Due to the differing details of each purchase, the Shopping Cart Service uses a MongoDB database to handle it efficiently.

2.8 Chat Service

The Chat service is currently final planned API. It provides users with ability to start a chat with automated bots, but also real human assistants. It also saves the entire chat history, allowing for data analysis to further optimize app and user experience.

3 Service Communication

This section details how the various services within the application communicate with each other, focusing on the use of RESTful APIs. It includes examples of REST

endpoints for different services, illustrating the methods and data formats used for inter-service communication. Please note that gateway endpoints are omitted, as they are very similar to the endpoints used by the services themselves.

3.1 Auth Service

POST /register
Register new account

Request:
Content:
email Valid available email address
firstName Valid first name
lastName Valid last name
password Valid secure password

```
{
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "password": "SecurePassword1!"
}
```

Response:
200: *Successful registration*
Content:
id User id
email Email address used for registration
firstName First name used for registration
lastName Last name used for registration

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
}
```

400: *Missing fields*
400: *Invalid email format*
409: *Email is already used*

POST /register/employee
Register new employee account

Request:

Headers:

Access-Token Valid access token

Content:

email Valid available email address
firstName Valid first name
lastName Valid last name
password Valid secure password
accountType Valid account type

```
{
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "password": "SecurePassword1!",
  "accountType": "ADMIN"
}
```

Response:

200: *Successful registration*

Content:

id User id
email Email address used for registration
firstName First name used for registration
lastName Last name used for registration
accountType Selected account type

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "accountType": "ADMIN"
}
```

400: *Missing fields*
400: *Invalid email format*
409: *Email is already used*
401: *Unauthorized*
403: *Forbidden*

POST /login

Login to account

Request:

Content:

email Valid available email address

password Valid secure password

```
{
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "password": "SecurePassword1!"
}
```

Response:

200: *Successful login*

Content:

id User id

email User email address

firstName User first name

lastName User last name

accountType User account type

accessToken JWT access token

refreshAccessToken JWT refresh access token

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "accountType": "USER",
  "accessToken": "eyJhbGciOs...",
  "refreshAccessToken": "eyJhbGciOi..."
}
```

400: *Missing fields*

401: *Invalid credentials*

403: *Account not activated*

429: *Too many login attempts*

POST /logout
Log out of account

Request:
Headers:
Refresh-Token Valid refresh access token

Response:
200: *Successful logout*
401: *Unauthorized*

DELETE /account
Delete user account

Request:
Headers:
Access-Token Valid access token

Response:
200: *Successfully deleted user account*
403: *Forbidden*

3.2 Store Service

GET /**stores**

Get list of stores

Request:

Headers:

totalCount Number of all elements

pageSize Page size

currentPage Number of current page

totalPages Number of all pages

firstItem First item identifier used for back pagination

lastItem Last item identifier used for next pagination

Response:

200: *Stores list*

Content:

sites [] Sites

id Site id

type Site type

location Site location

x Coordinate

y Coordinate

address Site address

country Site country

city Site city

zipCode Site zip code

street Site street

streetNumber Site street number

```
{
  "sites": [
    "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    "type": "Store",
    "location": {
      x: 52.222620,
      y: 20.982963
    }
    "address": {
      country: "Poland",
      city: "Warsaw",
      zipCode: "02-304",
      street: "Aleje␣Jerozolimskie",
      streetNumber: "107"
    }
  ]
}
```

GET /sites

Get list of sites (stores + magazines etc.), used by employees

---

Request:

Headers:

Access-Token Valid access token

totalCount Number of all elements

pageSize Page size

currentPage Number of current page

totalPages Number of all pages

firstItem First item identifier used for back pagination

lastItem Last item identifier used for next pagination

Response:

200: *Sites list*

Content:

sites [] Sites

id Site id

type Site type

location Site location

x Coordinate

y Coordinate

address Site address

country Site country

city Site city

zipCode Site zip code

street Site street

streetNumber Site street number

```
{
  "sites": [
    "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    "type": "Store",
    "location": {
      x: 52.222620,
      y: 20.982963
    }
    "address": {
      country: "Poland",
      city: "Warsaw",
      zipCode: "02-304",
      street: "Aleje␣Jerozolimskie",
      streetNumber: "107"
    }
  ]
}
```

403: *Forbidden*

GET /sites/{id}
Get site details

Request:
Headers:
Access-Token Valid access token

Response:
200: *Site details*
Content:
id Site id
type Site type
location Site location
x Coordinate
y Coordinate
address Site address
country Site country
city Site city
zipCode Site zip code
street Site street
streetNumber Site street number

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "type": "Store",
  "location": {
    x: 52.222620,
    y: 20.982963
  }
  "address": {
    country: "Poland",
    city: "Warsaw",
    zipCode: "02-304",
    street: "Aleje␣Jerozolimskie",
    streetNumber: "107"
  }
}
```

403: *Forbidden*

POST /sites/{id}
Create new site

Request:
Headers:
Access-Token Valid access token
Content:
type Site type
location Site location
x Coordinate
y Coordinate
address Site address
country Site country
city Site city
zipCode Site zip code
street Site street
streetNumber Site street number

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "type": "Store",
  "location": {
    x: 52.222620,
    y: 20.982963
  }
  "address": {
    country: "Poland",
    city: "Warsaw",
    zipCode: "02-304",
    street: "Aleje␣Jerozolimskie",
    streetNumber: "107"
  }
}
```

Response:
200: *Successfully created site*
403: *Forbidden*

PUT /sites/{id}
Edit site

Request:
Headers:
Access-Token Valid access token
Content:
location Site location
x Coordinate
y Coordinate

```
{
  "location": {
    x: 52.222620,
    y: 20.982963
  }
}
```

Response:
200: *Successfully updated site*
Content:
id Site id
type Site type
location Site location
x Coordinate
y Coordinate
address Site address
country Site country
city Site city
zipCode Site zip code
street Site street
streetNumber Site street number

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "type": "Store",
  "location": {
    x: 52.222620,
    y: 20.982963
  }
  "address": {
    country: "Poland",
    city: "Warsaw",
    zipCode: "02-304",
    street: "Aleje␣Jerozolimskie",
    streetNumber: "107"
  }
}
```

403: *Forbidden*

DELETE /account
Delete site

Request:
Headers:
Access-Token Valid access token

Response:
200: *Successfully deleted site*
403: *Forbidden*

3.3 Notification Service

GET /consents
Get user consents for notifications

Request:
Headers:
Access-Token Valid access token

Response:
200: *User notification consents*
Content:
id User id
consents [] User notification consents
{type} Consent type

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "consents": [
    "email": true,
    "sms": true,
  ]
}
```

403: *Forbidden*

POST /account

Create user account (used by gateway after successful registration)

Request:

Headers:

Access-Token Valid access token

Content:

id User id (assigned by Auth Service)

email User email address

firstName User first name

lastName User last name

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
}
```

Response:

200: *Successfully created user account*

Content:

id User id

email User email address

firstName User first name

lastName User last name

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith"
}
```

400: *Missing fields*

403: *Forbidden*

PUT /consents

Update user consents for notifications

Request:

Headers:

Access-Token Valid access token

Content:

consents [] User notification consents

{type} Consent type

```
{
  "consents": [
    "email": true,
    "sms": true,
  ]
}
```

Response:

200: *Updated user consents for notifications*

Content:

id User id

consents [] User notification consents

{type} Consent type

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "consents": [
    "email": true,
    "sms": true,
  ]
}
```

403: *Forbidden*

PUT /account

Update user account details (dependent on changes made to the Profile service)

Request:

Headers:

Access-Token Valid access token

Content:

firstName User first name

lastName User last name

age User age

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": "24"
}
```

Response:

200: *Updated user details*

Content:

id User id

email User email address

firstName User first name

lastName User last name

age User age

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "age": "24"
}
```

403: *Forbidden*

DELETE /account

Delete user account

Request:

Headers:

Access-Token Valid access token

Response:

200: *Successfully deleted user account*

403: *Forbidden*

3.4 Profile Service

GET /account
Get user own account details

Request:
Headers:
Access-Token Valid access token

Response:
200: *User details*
Content:
id User id
email User email address
firstName User first name
lastName User last name
age User age

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "age": "24"
}
```

403: *Forbidden*

GET /account/{email}

Get user account details (for employees)

Request:

Headers:

Access-Token Valid access token

Response:

200: *User details*

Content:

id User id

email User email address

firstName User first name

lastName User last name

age User age

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "age": "24"
}
```

403: *Forbidden*

POST /account

Create user account (used by gateway after successful registration)

Request:

Headers:

Access-Token Valid access token

Content:

id User id (assigned by Auth Service)

email User email address

firstName User first name

lastName User last name

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
}
```

Response:

200: *Successfully created user account*

Content:

id User id

email User email address

firstName User first name

lastName User last name

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith"
}
```

400: *Missing fields*

403: *Forbidden*

PUT /account
Update user account details

Request:
Headers:
Access-Token Valid access token
Content:
firstName User first name
lastName User last name
age User age

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": "24"
}
```

Response:
200: *Updated user details*
Content:
id User id
email User email address
firstName User first name
lastName User last name
age User age

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "email": "example@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "age": "24"
}
```

403: *Forbidden*

DELETE /account
Delete user account

Request:
Headers:
Access-Token Valid access token

Response:
200: *Successfully deleted user account*
403: *Forbidden*

3.5 Product Service

GET /product/{category}/{subctegory}?maxPrice=300&sort=high-price
Get list of products

Request:
Headers:
totalCount Number of all elements
pageSize Page size
currentPage Number of current page
totalPages Number of all pages
firstItem First item identifier used for back pagination
lastItem Last item identifier used for next pagination

Response:
200: *Products list*
Content:
chats [] Chat history array
id Product id
name Product name
price Product price
priceCurrency Price Currency
image Product image
url Image url
category Main category of product
type Category type
subcategory Subcategory of product
type Subcategory type


```
{
  "products": [
    {
      "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
      "name": "shoes1X",
      "price": "200",
      "priceCurrency": "pln",
      "image": {
        "url": "https://eu-central-1.console.aws.amazon.com//665d9d6..."
      }
      "category": {
        "type": "Shoes",
        "subcategory": {
          "type": "Running"
        }
      }
    }
  ]
}
```

GET /product/{id}
Get product details

Response:
200: *Product details*
Content:
id Product id
name Product name
description Product description
price Product price
priceCurrency Price Currency
images [] Product images array
url Image url
category Main category of product
type Category type
subcategory Subcategory of product
type Subcategory type


```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "name": "shoes1X",
  "description": "Comfortable shoes for running",
  "price": "200",
  "priceCurrency": "pln",
  "images": [
    {
      "url": "https://eu-central-1.console.aws.amazon.com//665d9d6..."
    }
  ]
  "category": {
    "type": "Shoes",
    "subcategory": {
        "type": "Running"
    }
  }
}
```

POST /product

Create product

Request:

Headers:

Access-Token Valid access token

Content:

name Product name

description Product description

price Product price

priceCurrency Price Currency

category Main category of product

type Category type

subcategory Subcategory of product

type Subcategory type

```
{
  "name": "shoes1X",
  "description": "Comfortable shoes for running",
  "price": "200",
  "priceCurrency": "pln",
  "category": {
    "type": "Shoes",
    "subcategory": {
        "type": "Running"
    }
  }
}
```

Response:

200: *Successfully created product*

id Product id

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
}
```

400: *Missing fields*

403: *Forbidden*

PUT /product/{id}
Update product details

Request:

Headers:

Access-Token Valid access token

Content:

cost Product cost

```
{
  "cost": "200"
}
```

Response:

200: *Updated user details*

Content:

id Product id

name Product name

description Product description

price Product price

priceCurrency Price Currency

category Main category of product

type Category type

subcategory Subcategory of product

type Subcategory type

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
  "name": "shoes1X",
  "description": "Comfortable shoes for running",
  "price": "200",
  "priceCurrency": "pln",
  "category": {
    "type": "Shoes",
    "subcategory": {
        "type": "Running"
    }
  }
}
```

403: *Forbidden*

DELETE /product

Delete product

Request:

Headers:

Access-Token Valid access token

Response:

200: *Successfully deleted product*

403: *Forbidden*

POST /order

Create an order and block selected products

Request:

Content:

id Order id

products [] Selected products

id Product id

quantity Amount of selected products

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
  "products": [
    {
      "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
      "quantity": 1
    }
  ]
}
```

Response:

200: *Successfully created an order*

410: *Products are not available*

POST /order/cancel

Cancel the order and unblock selected products

Request:

Content:

id Order id

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
}
```

Response:

200: *Successfully cancelled an order*

404: *Order does not exist*

POST /order/finish

Finalize the order and permanently remove the selected products from the Product Service

Request:

Content:

id Order id

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
}
```

Response:

200: *Successfully finalized an order*

404: *Order does not exist*

3.6 Shopping Cart Service

GET /order
Get list of orders

Request:
Headers:
Access-Token Valid access token

Response:
200: *Orders list*
Content:
orders [] Orders list
id Order id
price Total order price
date Order creation date

```
{
  "orders": [
    {
      "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
      "price": 200,
      "date": "2024-07-25T15:30:00Z"
    }
  ]
}
```

403: *Forbidden*

GET /order/{id}
Get order details

Request:
Headers:
Access-Token Valid access token

Response:
200: *Order details*
Content:
id Order id
products [] Selected products
id Product id
quantity Amount of selected products
price Product price
discount Product discount
discount Total order discount
userDetails Customer details
deliveryAddress Delivery address
price Total order price


```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
  "products": [
    {
      "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
      "quantity": 1,
      "price": 200,
      "discount": 10,
    }
  ]
  "price": 200,
  "discount": 10,
}
```


404: *Order does not exist*

POST /order
Create an order

Request:
Content:
id Order id
products [] Selected products
id Product id
quantity Amount of selected products
price Product price
discount Product discount
discount Total order discount
date Order creation date
userDetails Customer details
deliveryAddress Delivery address
price Total order price

```
{
  "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
  "products": [
    {
      "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
      "quantity": 1
      "price": 200,
      "discount": 10,
    }
  ]
  "price": 200,
  "discount": 10,
  "date": "2024-07-25T15:30:00Z"
}
```

Response:
200: *Successfully created an order*

POST /order/{id}/cancel
Cancel the order

Request:
Headers:
Access-Token Valid access token

Response:
200: *Successfully cancelled an order*
404: *Order does not exist*

PUT /order/{id}
Update order status

Request:
Headers:
Access-Token Valid access token
Content:
status 'in-delivery'

```
{
  "status": "in-delivery"
}
```

Response:
200: *Successfully updated order*
404: *Order does not exist*

3.7 Chat Service

GET /history?page={number}&currentPage={number}&item={date}
Get chat history

Request:
Access-Token Valid access token

Response:
200: *Chat history*
Headers:
Access-Token Valid access token
totalCount Number of all elements
pageSize Page size
currentPage Number of current page
totalPages Number of all pages
firstItem First item identifier used for back pagination
lastItem Last item identifier used for next pagination
Content:
chats [] Chat history array
id Chat id
firstMessage First message
creationDate Start date of chat

```
{
  "chats": [
    {
      "id": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
      "firstMessage": "Hello, I have a problem with last order",
      "creationDate": "2024-07-25T15:30:00Z",
    }
  ]
}
```

403: *Forbidden*

GET /history/{id}
Get chat messages

Response:

200: *Chat history*

Headers:

Access-Token Valid access token
totalCount Number of all elements
pageSize Page size
currentPage Number of current page
totalPages Number of all pages
firstItem First item identifier used for back pagination
lastItem Last item identifier used for next pagination

Content:

messages [] Chat messages array
content Message content
date Message date
senderId Id of message sender
senderName Name of message sender

```
{
  "messages": [
    {
        "content": "Hello, I have a problem with last order",
        "date": "2024-07-25T15:30:00Z",
        "senderId": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11",
        "senderName": "John Smith",
    }
  ]
}
```

403: *Forbidden*

DELETE /history/
Delete whole user chat history

Request:

Headers:

Access-Token Valid access token

Response:

200: *Successfully deleted chat history*
403: *Forbidden*

DELETE /history/{id}
Delete chat

Request:
Headers:
Access-Token Valid access token

Response:
200: *Successfully deleted chat*
403: *Forbidden*