SYSTEM SPECIFICATION

PalletPals – Phase 2

Lecturers	Lukas Frey Prof. Bradley Richards
Authors	Tibor Haller Marco Kaufmann Daniel Locher
Place and date	Brugg-Windisch, 02.06.2022

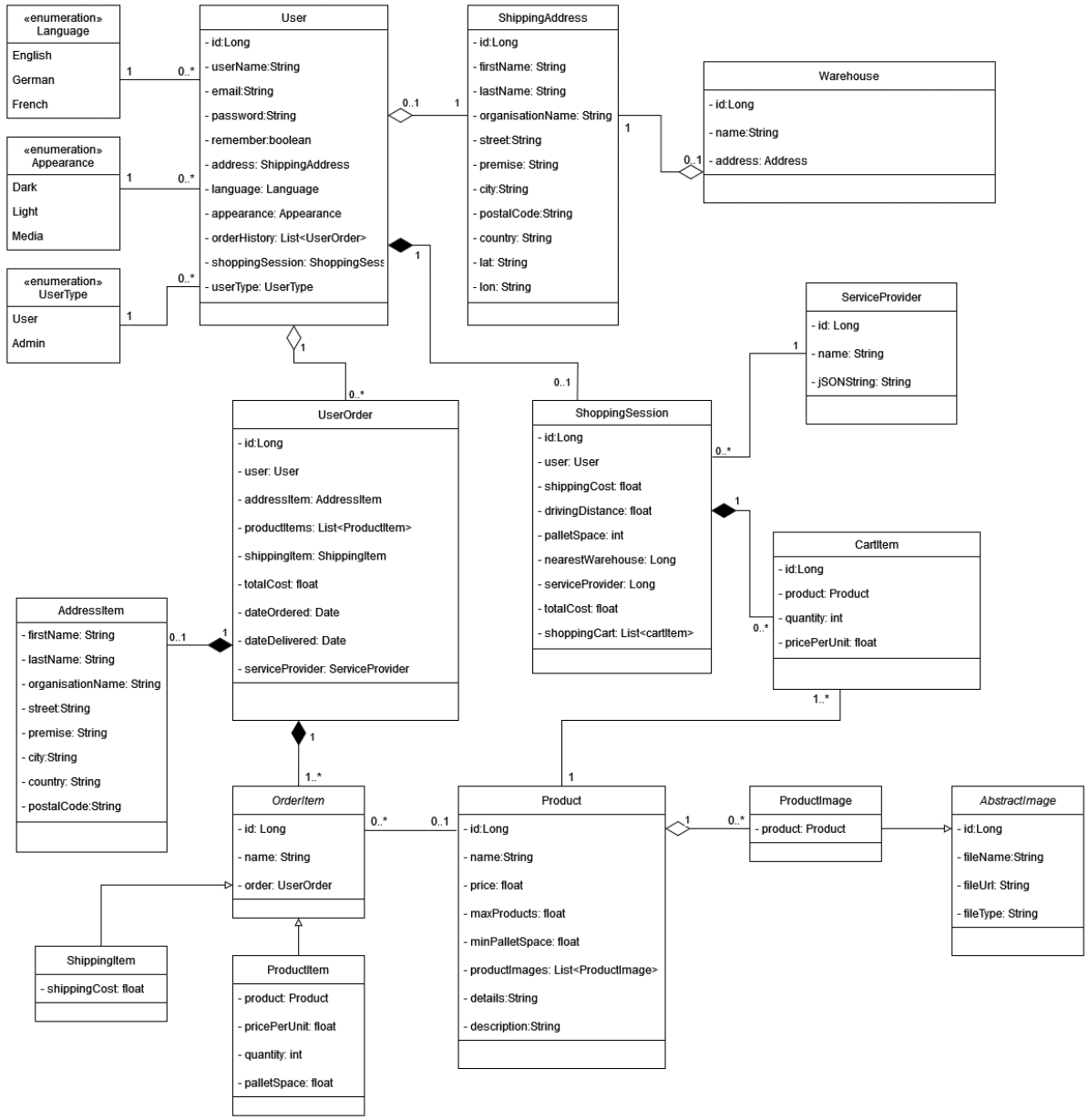
Introduction

This document describes the system specification of the PalletPals application, including the database representation in the form of an entity-relationship diagram, the logical backend design in form of a class diagram, and the definition of the communication protocol using sequence diagrams. Furthermore, the responsibilities of the team members is described.

Responsibilities

The backend and database integration is developed by Tibor Haller and Daniel Locher whereas the frontend is developed by Marco Kaufmann. However, contributions are reviewed by all team members and deviations from this separation are possible.

Class Diagram

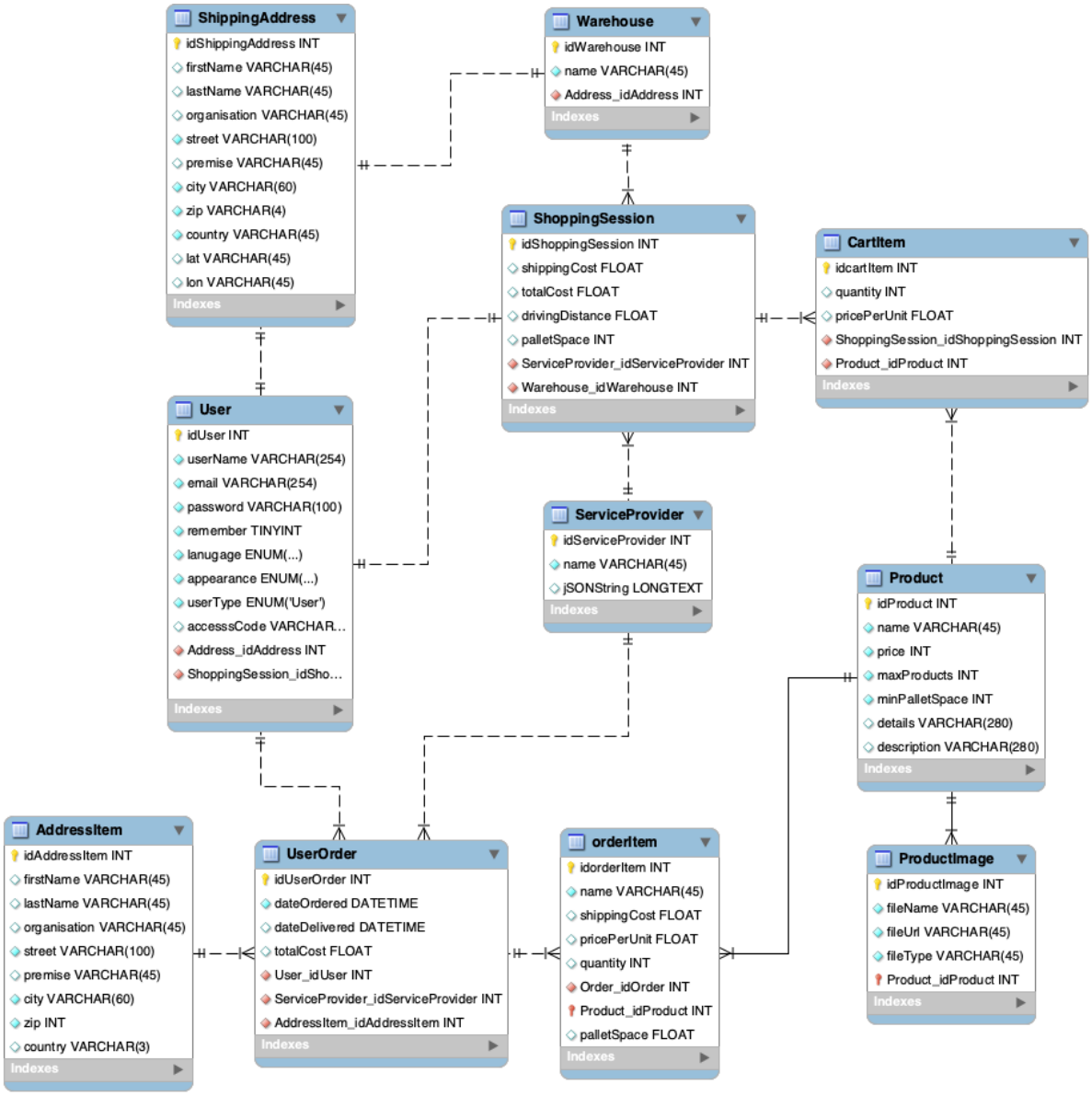


The Class Diagram shall give the reader of this document an overall perspective about the class structure of our application. Mostly, this diagram aims to be self-explanatory, nevertheless we like to point out a few things.

As our goal is to exceed the basic requirements, we plan to develop a more holistic online shop experience. One key issue was to ensure that orders reflect a snapshot of the relevant data, which is important when for example the user changes his address after an order was finalized, or that for example historic order data are preserved correctly. We have roughly split our classes into three overall data groups:

- Static Data
  This group includes all regular data classes, such as classes with product and user related data. A user should be able to change his personal data, as well as product data can change over time.
- Session Data
  We pursue to offer a real online shop feeling. With this in mind, we introduced session data to preserve a user's shopping basket over a certain period. Even if the website will be left, the shopping cart of a user is still saved in our database. Therefore, if this user revisits our website, he or she can flawlessly continue his or her purchase. Session data includes the classes ShoppingSession and CartItem. Once a user submits his order, the session data is used to create an order with the correct information, such as quantities or shipping address.
- Processed Data
  After a user has confirmed the order, the data gathered in his or hers shopping session will be changed to processed data, using the order class. According to the user's shopping session new objects of the classes UserOrder, AddressItem, ShippingItem, and ProductItem will be created and referenced by the new order object. AddressItem is only relevant in the context of UserAddress and only used to store address attributes as part of the class. The logic behind this process is, that the user shall have an history of his or her orders. These objects are important to ensure the data integrity as simply working with references incurs the risk of faulty order when some data is updated at a later point. One example that must be handled is that the user updates his shipping address after finalizing an order. Furthermore, the ShoppingSession object with its related CartItems get deleted as soon as the order has been saved to the database, meaning that the user's shopping cart is emptied.

Database Design



The database represents nearly the same logic as the class diagram. However, we like to elaborate on how our two abstract classes OrderItem and ProductImage are getting handled. We chose to pursue the single table strategy, where the interrelated subclasses are saved into the same table. This has the consequence that those tables will contain null values, but in terms of querying the data, we expect to reduce the complexity. More information can be found at https://en.wikibooks.org/wiki/Java_Persistence/Inheritance#Single_Table_Inheritance.

Since we will build the backend logic with Spring Boot, we will make use of Spring JPA which will handle the setup of the database. According to this Entity Relationship Diagram we will create the Java classes with the according annotations so the system can operate smoothly.

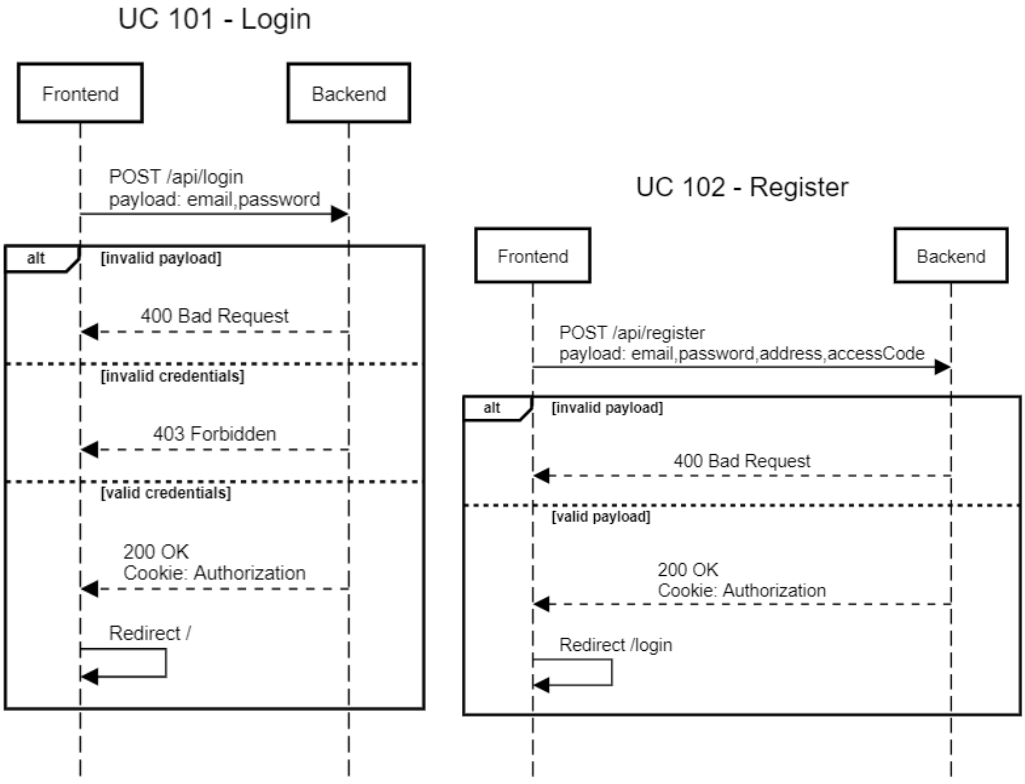
Communications Protocol

The communication between the frontend and the backend is based on the client-server design pattern where the frontend serves as the client and the backend serves as the server. This results in the frontend making requests to the backend using the HTTP protocol and the backend returning HTTP responses. These requests are made using the Fetch API[1].

Respective HTTP methods such as GET, POST, or DELETE are used to send requests with a potential payload that has a content type of text/json. Alternatively, image or similar data is sent using the multipart/form-data content type.

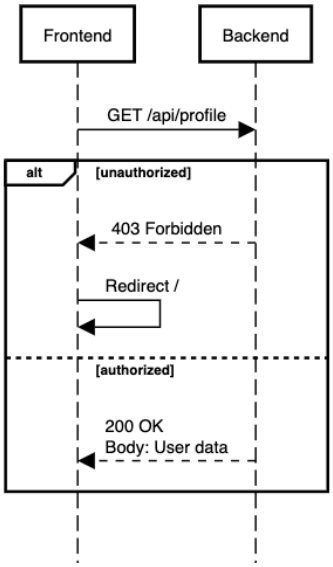
Authentication and authorization are verified on each request by the backend. When a login request is successful, the backend sets a HttpOnly cookie containing a JSON Web Token (JWT) of the logged in user. This authorization cookie is read for every request to verify both authentication and authorization. Furthermore, a HEAD request is made by the frontend when first loaded to verify the status of the user.

To prevent cross site request forgery attacks, a XSRF-TOKEN is set as cookie on every response by the backend. This cookie is read and verified by the backend on every request. The following sequence diagrams describe the requests and responses that occur for each use case.

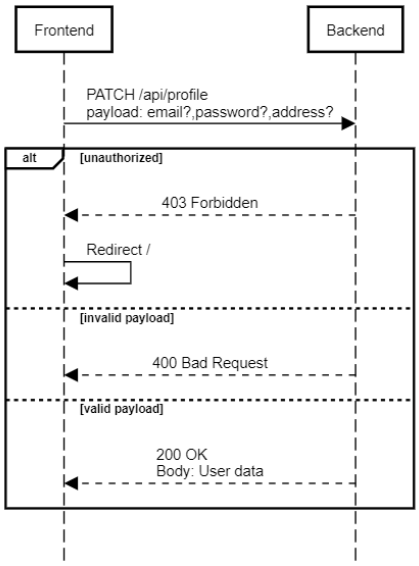


[1] https://fetch.spec.whatwg.org/

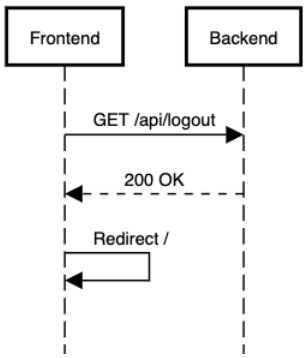
UC 104 - View user data



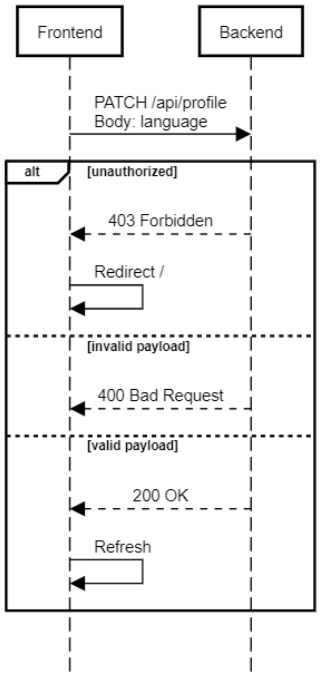
UC 105 - Edit user data



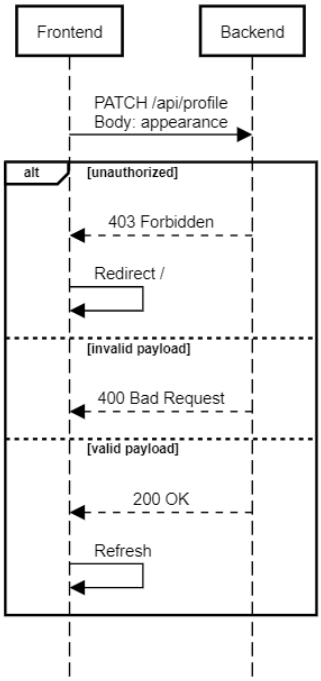
UC 103 - Logout



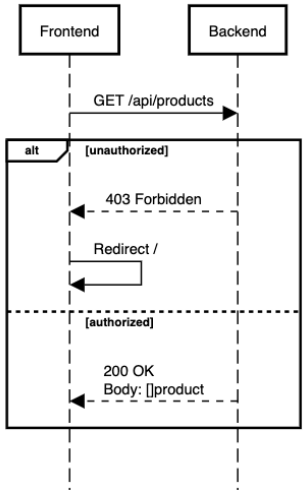
UC 106 - Change language



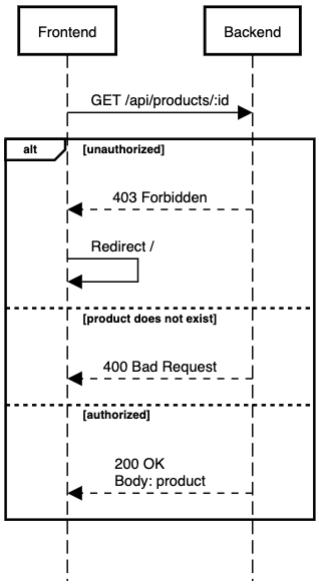
UC 107 - Change appearance



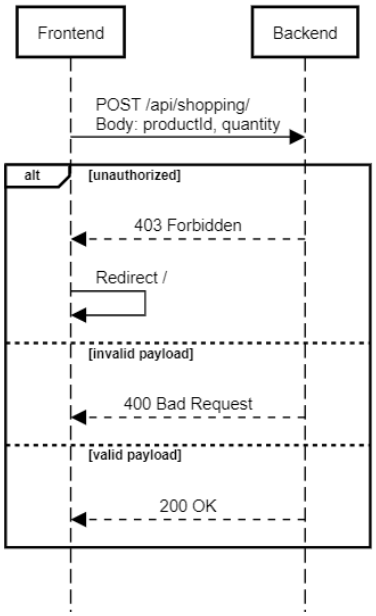
UC 201 - View product overview



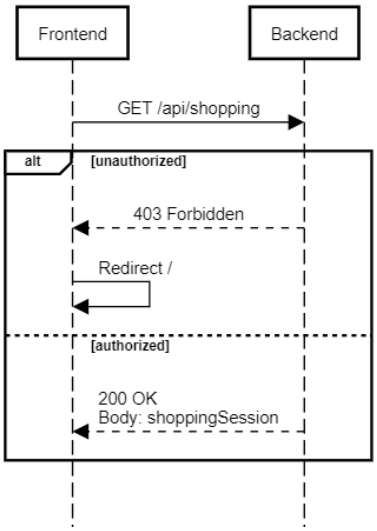
UC 202 - View product detail view



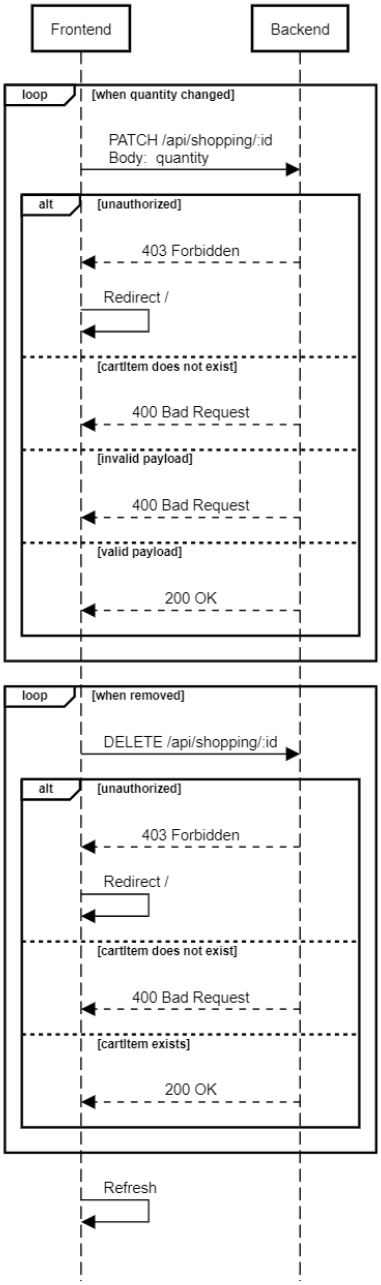
IC 301 - Add product to shopping ca



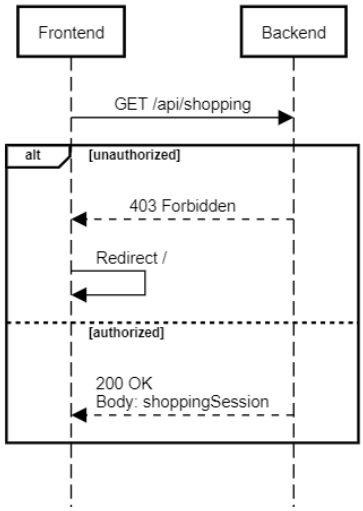
UC 302 - Preview shopping cart



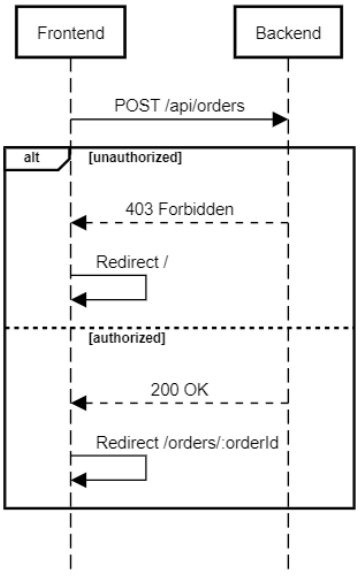
UC 304 - Edit shopping cart



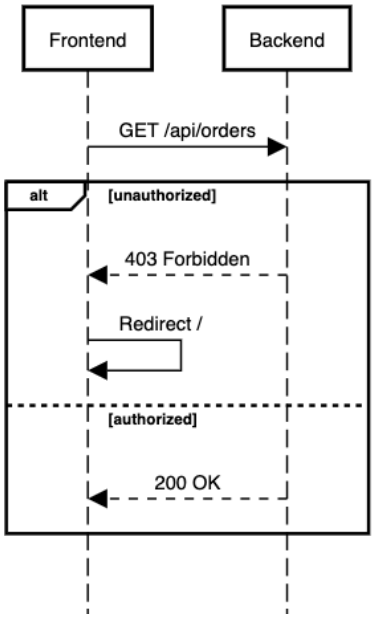
UC 303 - View shopping cart



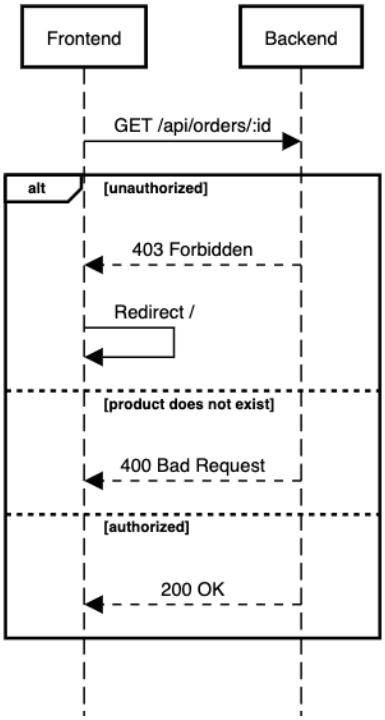
UC 401 - Submit order



UC 402 - View order history



UC 403 - View order details



UC 404 - Re-order past order