FullCart WebShop

Katona, Áron

Team

Fodor, Zsófia Katona, Áron

January 4, 2021

Abstract

The project is a proof-of-concept application for using session types for communication between micro-services in a distributed back-end.

The application implements a webshop. It provides services for creating, viewing and managing user accounts, products and orders.

Another important factor is the list of technologies we used, and their purpose:

1. Scala
2. Spring Boot: webserver, database access (JPA)
3. Maven: project, modules
4. Docker: Containers for the services and for the databases
5. Docker Compose: setting up all the containers at once
6. Scribble, Scribble-Java [1]: Session type generation
7. REST+HATEOAS: communication protocl with the end-user

The project is divided into modules, one for each micro-service. The back-end is split into 3 microservices:

- Webshop(proxy) service: receives HTTP requests and responds to them
- Product service: manages the products
- User service: manages the users
- Buying service: manages the orders placed by the user

The proxy provides the REST endpoints and calls the other micro-services via the session types, in order to fulfill the requests.

My responsibility was the implementation of the:

- Buying Service
- Webshop Service
- BuyingSession protocol
- Maven project configuration
- Containerization

Contents

1 Design

1.1 Use Case Diagram

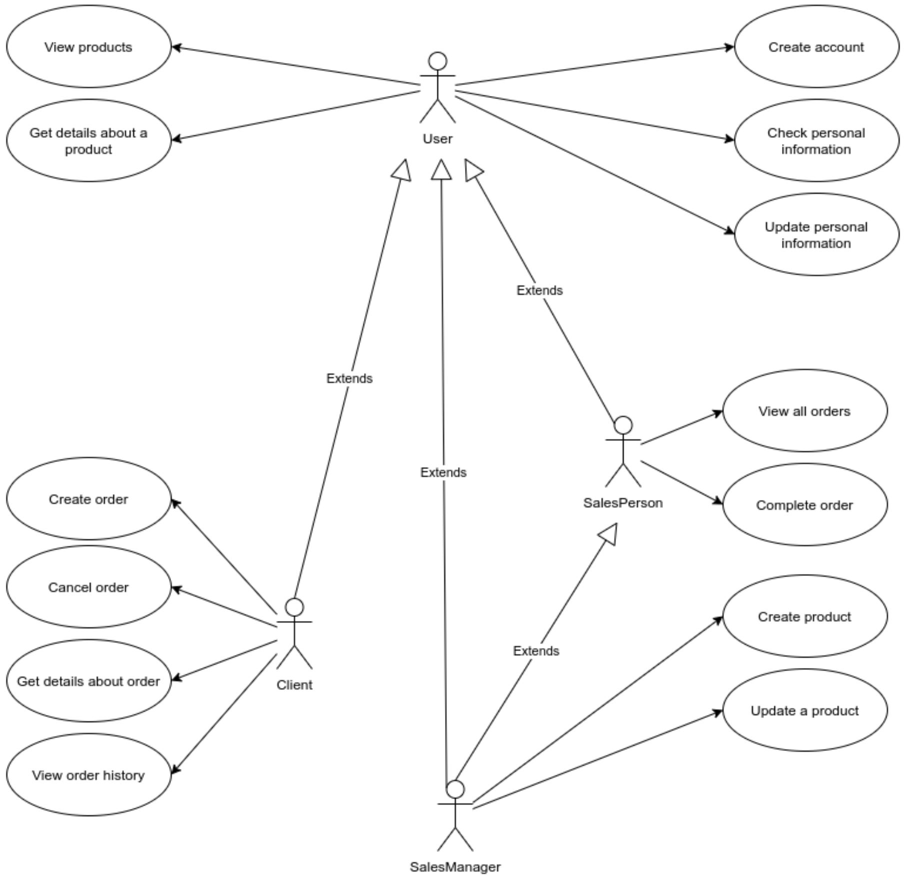

Figure 1: Use Case diagram

1.2 Models

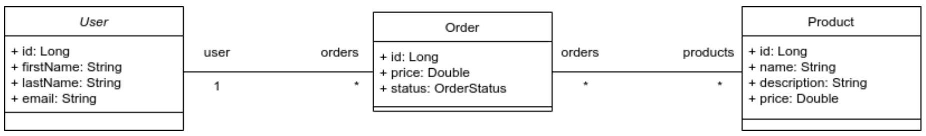

Figure 2: Models of the application

1.3 Protocols

1.3.1 Protocol specification

Listing 1 contains the protocol specification in the Scribble language. The following protocols were defined:

- **ProductSession:** protocol for communication with the product service
- **UserSession:** protocol for communication with the user service
- **BuyingSession:** protocol for communication with the buying service, which also communicates with the User and Product service.

Listing 1: `Webshop.scr`

```
module com.fullcart.session.Webshop;

data <java> "java.lang.Long" from "rt.jar" as Long;
data <java> "java.lang.Iterable" from "rt.jar" as List;
data <java> "java.lang.String" from "rt.jar" as String;
data <java> "com.fullcart.dto.ProductDTO" from "rt.jar" as Product;
data <java> "com.fullcart.dto.UserDTO" from "rt.jar" as User;
data <java> "com.fullcart.dto.OrderDTO" from "rt.jar" as Order;


/**
 * C = Client
 * P = Product service
 */
global protocol ProductSession(role C, role P) {
    choice at C {
        GetAll() from C to P;
        Ok(List) from P to C;
        do ProductSession(C, P);
    } or {
        GetOne(Long) from C to P;
        choice at P {
            Ok(Product) from P to C;
        } or {
            NotFound() from P to C;
        }
        do ProductSession(C, P);
    } or {
        Create(Product) from C to P;
        choice at P {
            Ok(Product) from P to C;
        } or {
            Err(String) from P to C;
        }
        do ProductSession(C, P);
    } or {
        Update(Long, Product) from C to P;
        choice at P {
            Ok(Product) from P to C;
        } or {
            NotFound() from P to C;
        } or {
            Err(String) from P to C;
        }
        do ProductSession(C, P);
    } or {
        Replace(Long, Product) from C to P;
        choice at P {
            Ok(Product) from P to C;
        } or {
            Created(Product) from P to C;
        } or {
            Err(String) from P to C;
        }
        do ProductSession(C, P);
    } or {
        Delete(Long) from C to P;
        choice at P {
            Ok() from P to C;
        } or {
            NotFound() from P to C;
        }
        do ProductSession(C, P);
    } or {
        Bye() from C to P;
    }
}

/**
 * C = Client
 * U = User service
 */
global protocol UserSession(role C, role U) {
    choice at C {
        GetAll() from C to U;
        Ok(List) from U to C;
        do UserSession(C, U);
    } or {
        GetOne(Long) from C to U;
        choice at U {
            Ok(User) from U to C;
        } or {
            NotFound() from U to C;
        }
        do UserSession(C, U);
    } or {
        Create(User) from C to U;
        choice at U {
            Ok(User) from U to C;
        } or {
            Err(String) from U to C;
        }
        do UserSession(C, U);
    } or {
        Update(Long, User) from C to U;
        choice at U {
            Ok(User) from U to C;
        } or {
            NotFound() from U to C;
        } or {
            Err(String) from U to C;
        }
        do UserSession(C, U);
    } or {
        Replace(Long, User) from C to U;
        choice at U {
            Ok(User) from U to C;
        } or {
            Created(User) from U to C;
        } or {
            Err(String) from U to C;
        }
        do UserSession(C, U);
    } or {
        Delete(Long) from C to U;
        choice at U {
            Ok() from U to C;
        } or {
            NotFound() from U to C;
        }
        do UserSession(C, U);
    } or {
        Bye() from C to U;
    }
}

/**
 * C = Client
 * P = Product service
 * U = User service
 * B = buying service
 */
global protocol BuyingSession (role C, role P, role U, role B)
 {
```

```
choice at C {
   GetAll() from C to B;
   Ok(List) from B to C;
   do BuyingSession(C, P, U, B);
} or {
   GetOne(Long) from C to B;
   choice at B {
      Ok(Order) from B to C;
   } or {
      NotFound() from B to C;
   }
   do BuyingSession(C, P, U, B);
} or {
    Create(Order) from C to B;
    choice at B {
       GetOne(Long) from B to U;
       choice at U {
          Ok(User) from U to B;
          GetAll(List) from B to P;
          choice at P {
             Ok(List) from P to B;
             Ok(Order) from B to C;
          } or {
             NotFound(Long) from P to B;
             ProductNotFound(Long) from B to C;
          }
       } or {
          NotFound(Long) from U to B;
          UserNotFound(Long) from B to C;
       }
    } or {
      Err(String) from B to C;
    }
    do BuyingSession(C, P, U, B);
} or {
   Cancel(Long) from C to B;
   choice at B {
      Ok(Order) from B to C;
   } or {
      NotFound() from B to C;
   } or {
      NotAllowed() from B to C;
   }
   do BuyingSession(C, P, U, B);
} or {
   Complete(Long) from C to B;
   choice at B {
      Ok(Order) from B to C;
   } or {
      NotFound() from B to C;
```

```
        } or {
            NotAllowed() from B to C;
        }
        do BuyingSession(C, P, U, B);
    } or {
        Bye() from C to B;
        Bye() from B to P;
        Bye() from B to U;
    }
}
```

1.3.2 Endpoint finite-state-machines from the client's point of view

[Next 3 pages]

63: __com_fullcart_session_Webshop_UserSession__C_U__
U!GetAll()
U?Ok(List)
U!GetOne(Long)
U?Ok(User)
U?NotFound()
U!Create(User)
U?Ok(User)
U?Err(String)
U!Update(Long, User)
U?Ok(User)
U?NotFound()
U?Err(String)
U!Replace(Long, User)
U?Ok(User)
U?Created(User)
U?Err(String)
U!Delete(Long)
U?Ok()
U?NotFound()
U!Bye()
65:
66:
67:
68:
69:
70:
64:

15: __com_fullcart_session_Webshop_ProductSession__C_P__
P!GetAll()
P?Ok(List)
P!GetOne(Long)
P?Ok(Product)
P?NotFound()
P!Create(Product)
P?Ok(Product)
P?Err(String)
P!Update(Long, Product)
P?Ok(Product)
P?NotFound()
P?Err(String)
P!Replace(Long, Product)
P?Ok(Product)
P?Created(Product)
P?Err(String)
P!Delete(Long)
P?Ok()
P?NotFound()
P!Bye()
17:
18:
19:
20:
21:
22:
16:

109: __com_fullcart_session_Webshop_BuyingSession__C_P_U_B__
B!GetAll()
B?Ok(List)
B!GetOne(Long)
B?Ok(Order)
B?NotFound()
B!Create(Order)
B?Ok(Order)
B?ProductNotFound(Long)
B?UserNotFound(Long)
B?Err(String)
B!Cancel(Long)
B?Ok(Order)
B?NotFound()
B?NotAllowed()
B!Complete(Long)
B?Ok(Order)
B?NotFound()
B?NotAllowed()
B!Bye()
111:
112:
113:
114:
115:
110:

1.4 Services

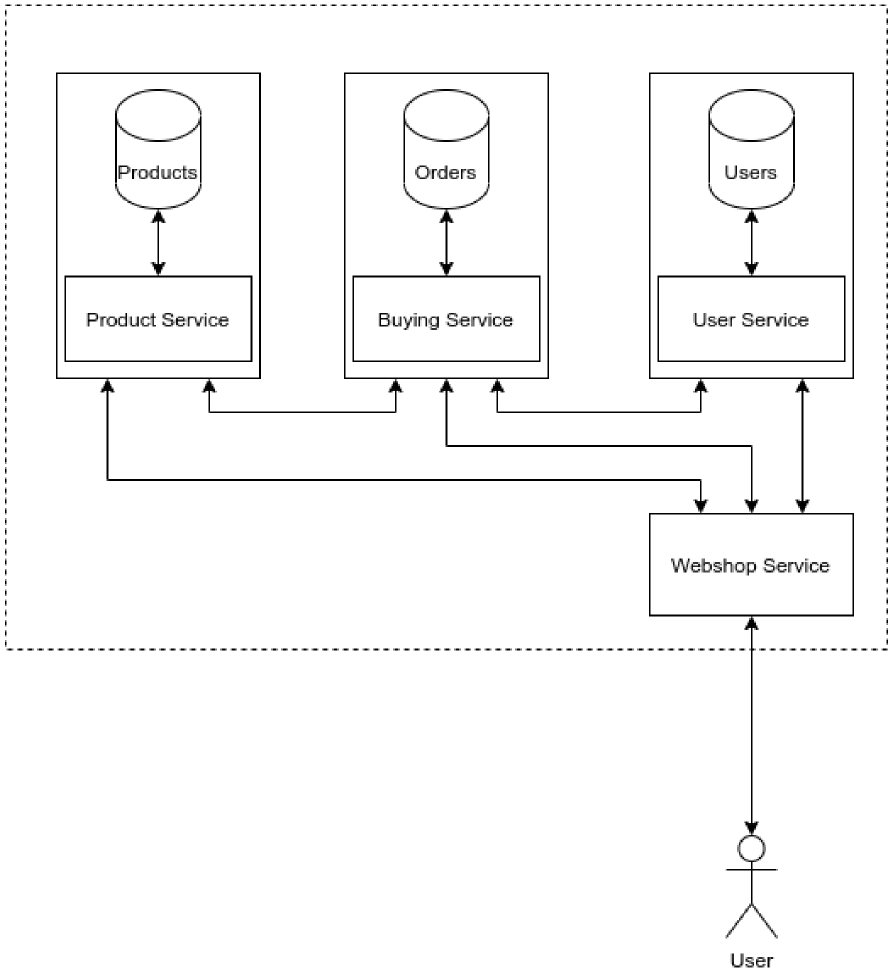

Figure 3: Overview of the micro-services

1.5 Deployment

The application is deployed in Docker containers. Each container represents an independent server computer.

For each micro-service a separate docker container was created. For those services that need a database, a separate MySQL container was associated. These databases are accessible to the service through the specified port, but inaccessible to any other containers.

The services communicate with each other through a specified port, using the protocols defined in the session types.

1.5.1 Deployment Diagram

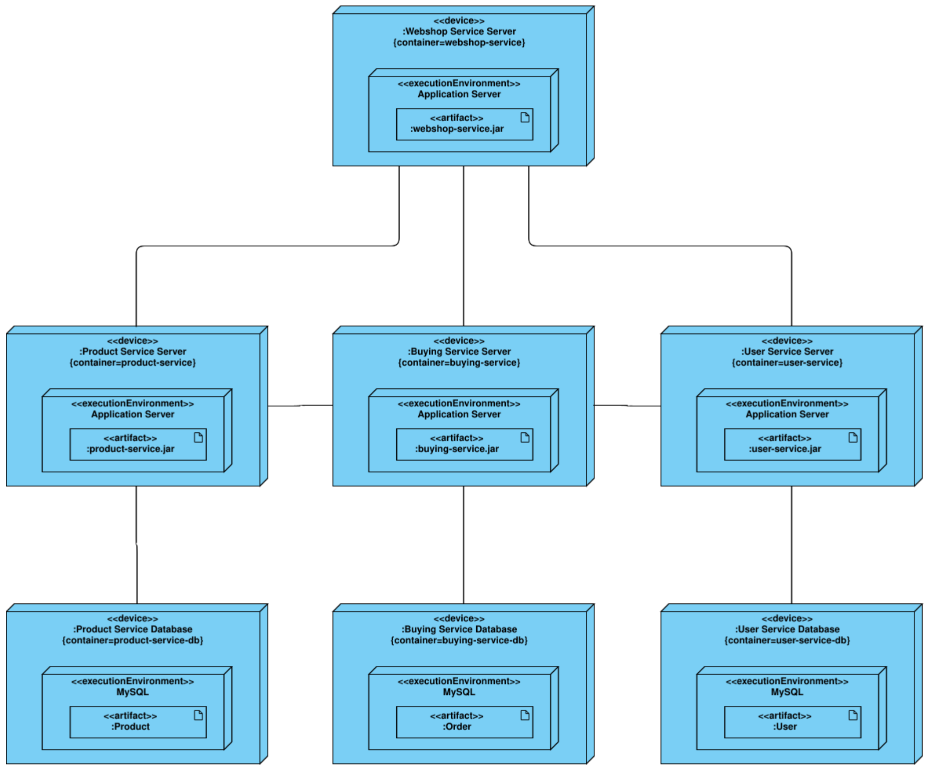

Figure 4: Overview of the micro-services

1.6 Sample .env file

This file must be placed next to ‘the docker-compose.yml‘ file.

Listing 2: `.env`

```
DATABASE_USER=user
DATABASE_PASSWORD=1234
DATABASE_NAME=webshop
DATABASE_PORT=3306
DATABASE_ROOT_PASSWORD=1234
```

References

[1] R. Hu and N. Yoshida, “Hybrid Session Verification through Endpoint API Generation,” in *19th International Conference on Fundamental Approaches to Software Engineering*, vol. 9633 of *LNCS*, pp. 401–418, Springer, 2016.