Software Architectures
Microservices

Term project **eSHOP**
Documentation

Jan Hlaváč, Martin Hruška, Petr Švagr

B4M36SWA

2021

1. Introduction

The project **eSHOP** is a microservice application for a simple e-shop website backend. It is split into **4** separate microservices that deal with the process of buying products off the shop.

1.1. Used technologies

Java 11 with **SpringBoot** for base app source code
Docker for running app in separate containers
Eureka for service discovery
Swagger OpenAPI for API endpoints documentation, provided by SpringDoc dependency
JUnit, **Mockito** for testing the REST API

1.2. Microservices and related features

The detailed architecture is shown in the next chapter. Each service has its own defined SwaggerUI API endpoints, accessible from URL `<microservice IP>/swagger-ui.html`

1.2.1. User service

User service deals with User accounts. It describes the User and his identificators (address, full name, email) to process the eshop Order.

1.2.2. Basket service

Basket service consists of items the User is currently shopping for but has yet to pay for.

1.2.3. Order service

Order service consists of items from the Basket the User was shopping for. The Order is created when User pays for the items from the Basket. Orders can be used for archiving of previous shopping activities as well.

1.2.4. Catalog service

Catalog service consists of three parts, making up the whole eshop catalogue.

- **Catalog service** – describes specific catalogues of items/categories.
  (e.g. Electronics, Hobby, Furniture, Kitchen Appliances, Garden, etc.)
- **Category service** – describes specific category of items within said catalog
  (e.g. Laptops, Desktop PC, Mobile phones, Cameras, Hardware, etc.)
- **Item service** – describes a specific item within a category
  (e.g. Lenovo ThinkPad E15, Samsung Galaxy A51 etc.)

1.2.5. Discovery service

Discovery service server is a separately running small application that registers all eshop services as discovery clients.

1.2.6. Health Monitoring

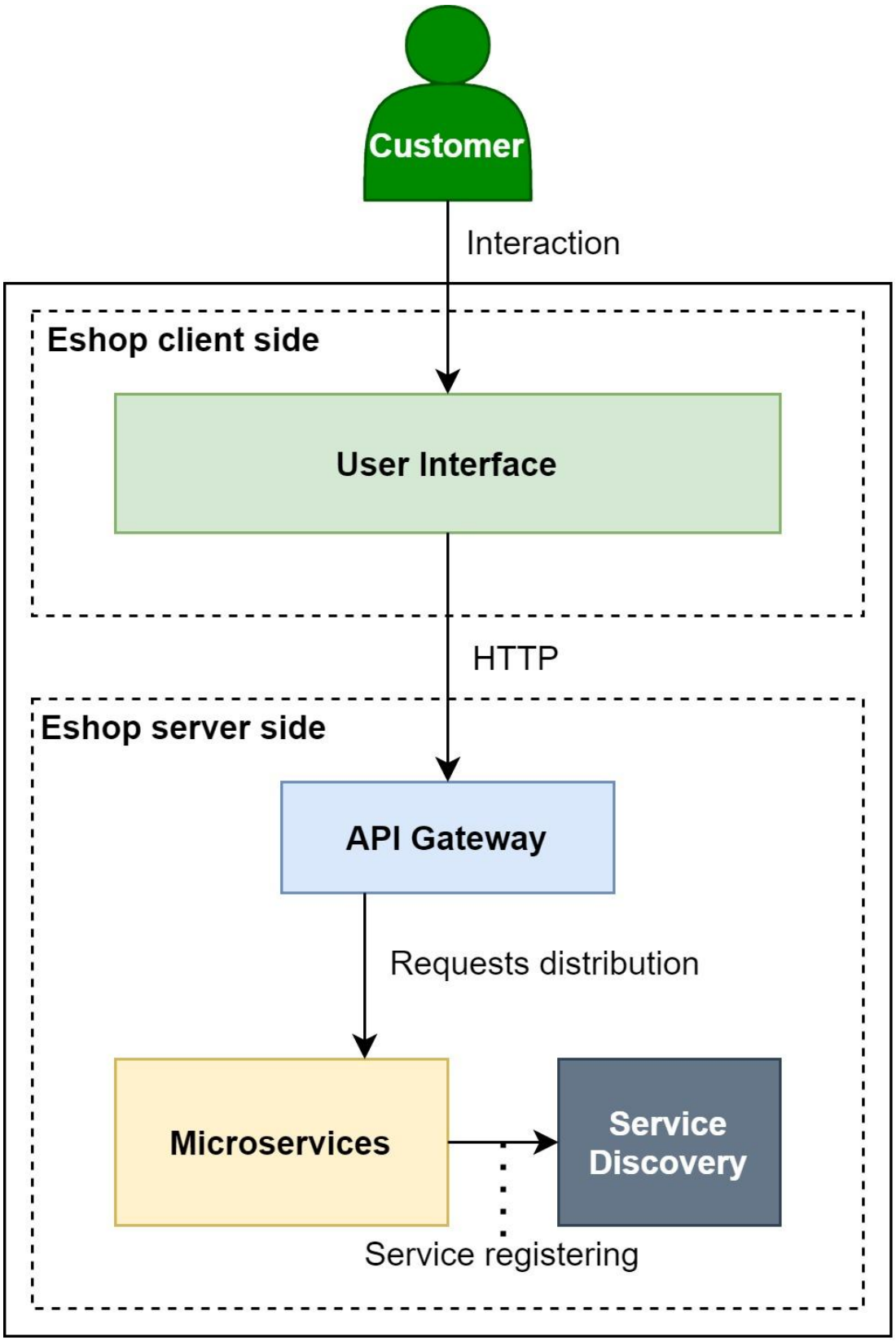
The system also provides health check actuators to verify microservice is up and running. It is accessible either by displaying Registered microservices inside Discovery service, or alternatively in each microservice under `<microservice IP>/actuator/health`

1.2.7. Tests

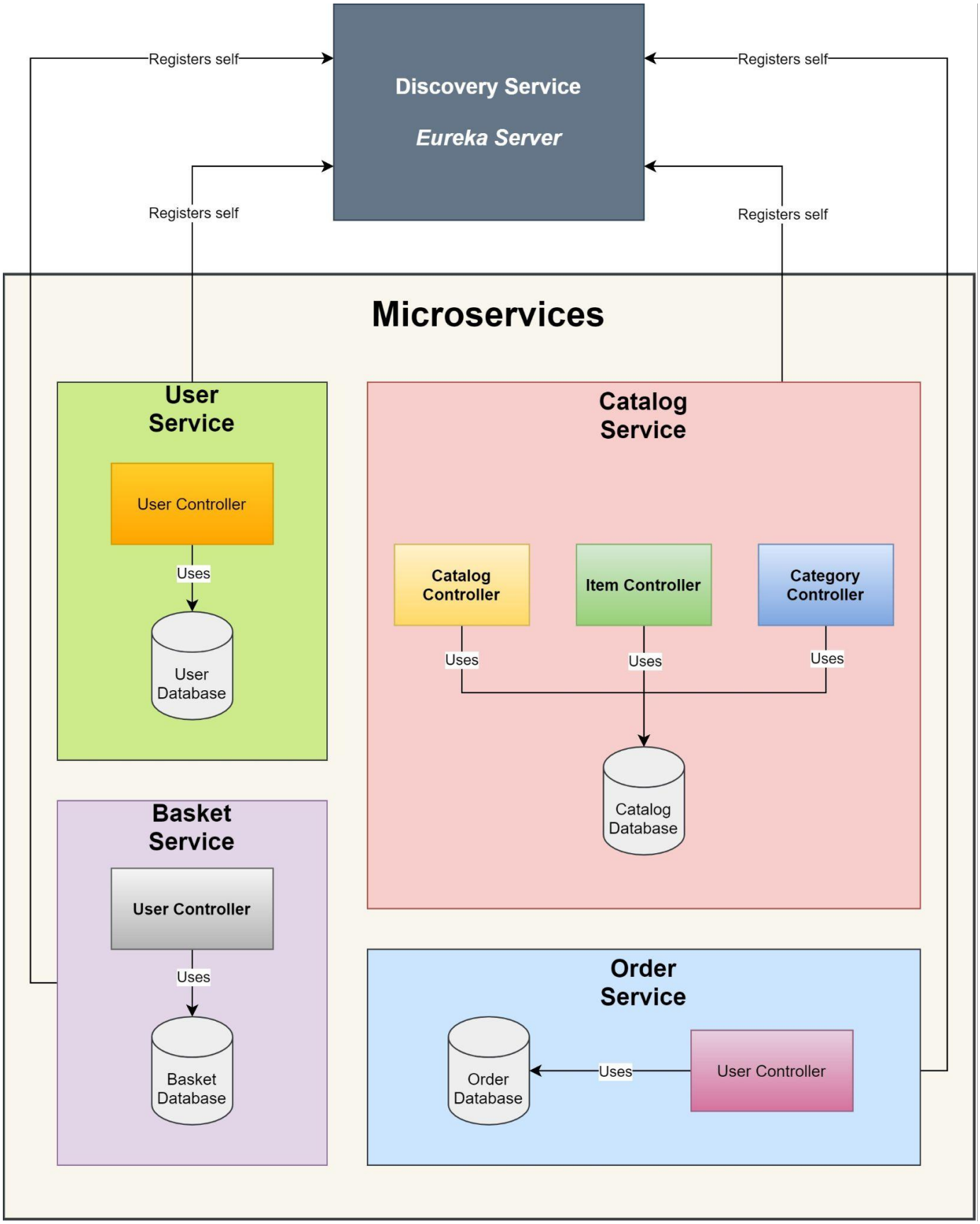
Each microservice has been tested with simple CRUD mock tests to verify functionality of REST API.

2. Architecture diagrams

2.1. System context

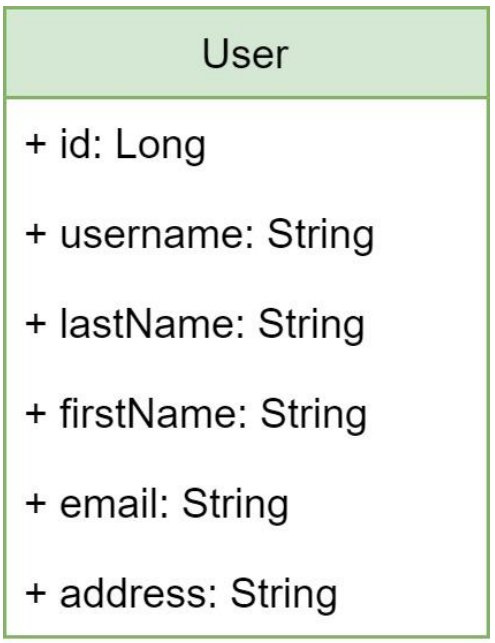


2.2. Microservices in detail

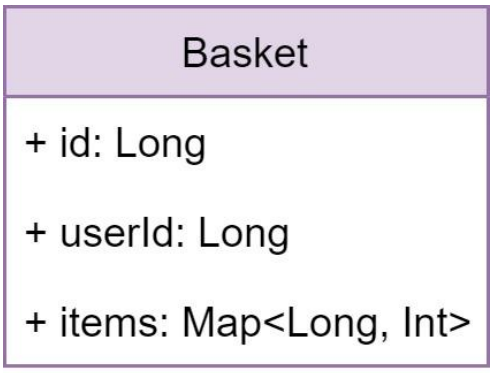


2.3. Class diagrams

2.3.1. User microservice



2.3.2. Basket microservice



2.3.3. Order microservice

2.3.4. Catalog microservice