Event-driven & Process-oriented Architectures
Exercise 3 & Exercise 4

Karim Ibrahim, Jonas Länzlinger, Kai Schultz

March 19, 2024

For the upcoming hand-ins we will have two different GitHub repositories.

1. EDPO-Group1 - This repository contains all hand-in relevant documentation files.
2. EDPO-Group1-Project - This repository contains our source code of the implementation.

This on Canvas uploaded PDF-file serves the purpose to refer to each individual Exercise and the corresponding important documents / files.

See README.md for the demo!

1 Exercise 3 & 4 - Documentation of our Progress

1.1 Infrastructure

We have decided to create an application for working with the low-level data streams of the `Fischertechnik Factory`. Therefore, we preliminarely decided to make use of the following technical specifics for our project:

1. Spring: building our individual services
2. Docker: quick and easy deployment
3. Camunda: modelling the processes
4. MQTT: listening on the data stream from the factory
5. Kafka: streaming the data that we collected with MQTT to the respective topics (i.e. stations)

1.2 Processes

For this iteration we have modelled a very simple business process:

1. Order Process
2. Warehouse Process

`Textual`

Placing an order constitutes the start of the Order process. Right after process initialization the order gets dispatched to the Warehouse process (the Warehouse station in the factory) where a user needs to perform a user task, i.e. checking a checkbox in order to continue in the flow. After this user task, the Warehouse process is concluded and the flow moves back to the Order process, where it gets passed through the stations Grabber and Delivery. When the service task in Delivery is finished, the order is ready for pickup, which is modelled with another user task. When someone checks the checkbox for this user task, the Order process is complete.

For now, we only outlines an idea of an error scenario, but from this point on it should not be a big issue to realize this error scenario.

1.3 Services

In general we have focused on the infrastructure and technical setup. There is little to no business logic in place yet.

1. FactorySimulator: This service simulates the behavior of the factory. Therefore, we have provided a data.txt file that contains live data collected from the factory. This data.txt file read one line each second and publishes an MQTT message on the "factory" topic. This service will be omitted once we can connect to the real factory.

2. FactoryListener: In the real scenario this is the connection between our application and the factory. We implemented an MQTT client that collects all messages from the "factory" topic. We decided to wrap the MQTT messages in this service, because then we don't need to implement MQTT for all other services. After parsing the MQTT messages, this service emits events on Kafka to the corresponding topics of the stations (for now, only to Warehouse and Grabber).

3. Order: This is the Order service that is owner of the Order Camunda process. He acts as an orchestrator in our architecture between the different station services.

4. Warehouse: This is the Warehouse service that is owner of the Warehouse Camunda process. The Warehouse service gets started via an external task message from the Order process. The Warehouse Caumnda end event emits an external task message to the Order process. Additionally the Warehouse service is listening on the Kafka "HBW_1" topic, to simulate how he could incorporate acting with information obtained from the factory.

5. Grabber: A toy Grabber service, that is performing an external task obtained by the Order process. For now, there is no business logic; he just prints some logs. Additionally the Grabber service is listening on the Kafka "VGR_1" topic, to simulate how he could incorporate acting with information obtained from the factory.

6. Delivery: A toy Delivery service, that is performing an external task obtained by the Order process. For now, there is no business logic; he just prints some logs.

1.4 Ports & Endpoints

1. `localhost:8080` - Camunda platform for Order process (demo/demo)
2. `localhost:8080/order.html` - endpoint to ordering website
3. `localhost:8081` - Camunda platform for Warehouse process (demo/demo)
4. `localhost:8082` - Grabber service
5. `localhost:8083` - Delivery service
6. `localhost:8084/send` - start redirecting MQTT messages from the factory to the respective station services
7. `localhost:8085/send` - start emitting simulated factory messages via MQTT on "factory" topic

2 Group Contributions

The contributions of each team member are equally distributed. Most of the implementations have been done in pair-programming.