DISYS-Project Fuel Station Documentation

About the project

This project was created by Rafin Ifta Alam, Lukas Huber, Dominik Pierdiluca und Jan Schreiber as a team for our fuel company for electric vehicles. The goal of this project is the implementation of a Fuel Station App. Customers can generate invoices for their vehicle charging activities.

We used a simple JavaFX UI for the frontend and for our APIs we built it in the SpringBoot framework. Additionally, we used RabbitMQ to align our messages between the services and the databases, which are basically PostgreSQL instances. For the development of this project, the team used GitHub for version control. Here is the link to our GitHub-repository: https://github.com/LukasHuber2001/Disys. Furthermore, did the team use Docker-containers for the databases and the RabbitMQ instances.

System Architecture:

The following UML diagram reflects the system architecture:

UML einfügen

Set-Up and Installation of Fuel Station App:

First, you need to clone this whole project from the GitHub repository. To archieve this, you can either do this with for example IntelliJ's built-in tool or run this command in your preferred command line interpreter within the directory you wish to save the project in:

```
--- Bash

Cd “path/to/your/desired/folder”

Git clone https://github.com/LukasHuber2001/Disys
```

Launch application modules and services

Then follow these actions:

1. Start your Docker Desktop application and then open your preferred command line interpreter and run the commands below. This will run the docker-compose file and create the databases and the RabbitMQ queue in a containerized environment.

```
--- Bash
Cd "./Db"
Docker compose up
```

2. Once the containers are up and running you need to open this project in IntelliJ.
   i) Then start these parts of applications in the following order:
      (a) RPC-Client
      (b) MessagingQueue
      (c) FuelStationPdfCreator
      (d) DataCollectorDispatcher
      (e) DataCollectorReceiver
      (f) StationDataCollector
      (g) SpringBoot
      (h) JavaFX

When you have completed all the parts, you can now use the application.

<u>Lessons learned</u>

Throughout the project, the team gained following key lessons:

1) Effective collaboration with GitHub version control: Our team collaboration and use of GitHub, a version control tool, enhanced our code management. We could work simultaneously on various project components, track modifications, merge code, and resolve conflicts.
2) More efficient development environments with Docker: Docker containers significantly improved our development process by allowing us to encapsulate databases and RabbitMQ instances. This ensured consistency across various team members' environments, making development faster and more efficient.
3) Learning curve: While integrating technologies such as JavaFX, Spring Boot, and RabbitMQ into our project, we faced a learning curve. However, we leveraged documentation and online resources to acquire new skills and apply them effectively. Specifically, for RabbitMQ, understanding message transmission and reception, along with establishing a consistent pattern, was crucial.
4) Transparent communication & coordination: Successful project outcomes relied on effective communication and coordination among team members. Regular meetings, clear task assignments, and ongoing updates ensured alignment across the team.

5) Understanding system architecture & parts integration: Creating a distributed system necessitated a comprehensive grasp of the system's overall architecture and the interactions among its various components. Consequently, it was crucial for each team member to comprehend the underlying design choices and associated theory.
6) Trouble shooting: During the project, we recognized the significance of robust error handling and effective debugging for promptly identifying and resolving issues.

Despite facing time constraints due to concurrent projects, our team acknowledges the need for improving component modularity and reusability. We also recognize the importance of flexibility in technological choices, considering varying skill sets and management demands. Overall, the experience gained will prove valuable for future endeavors.