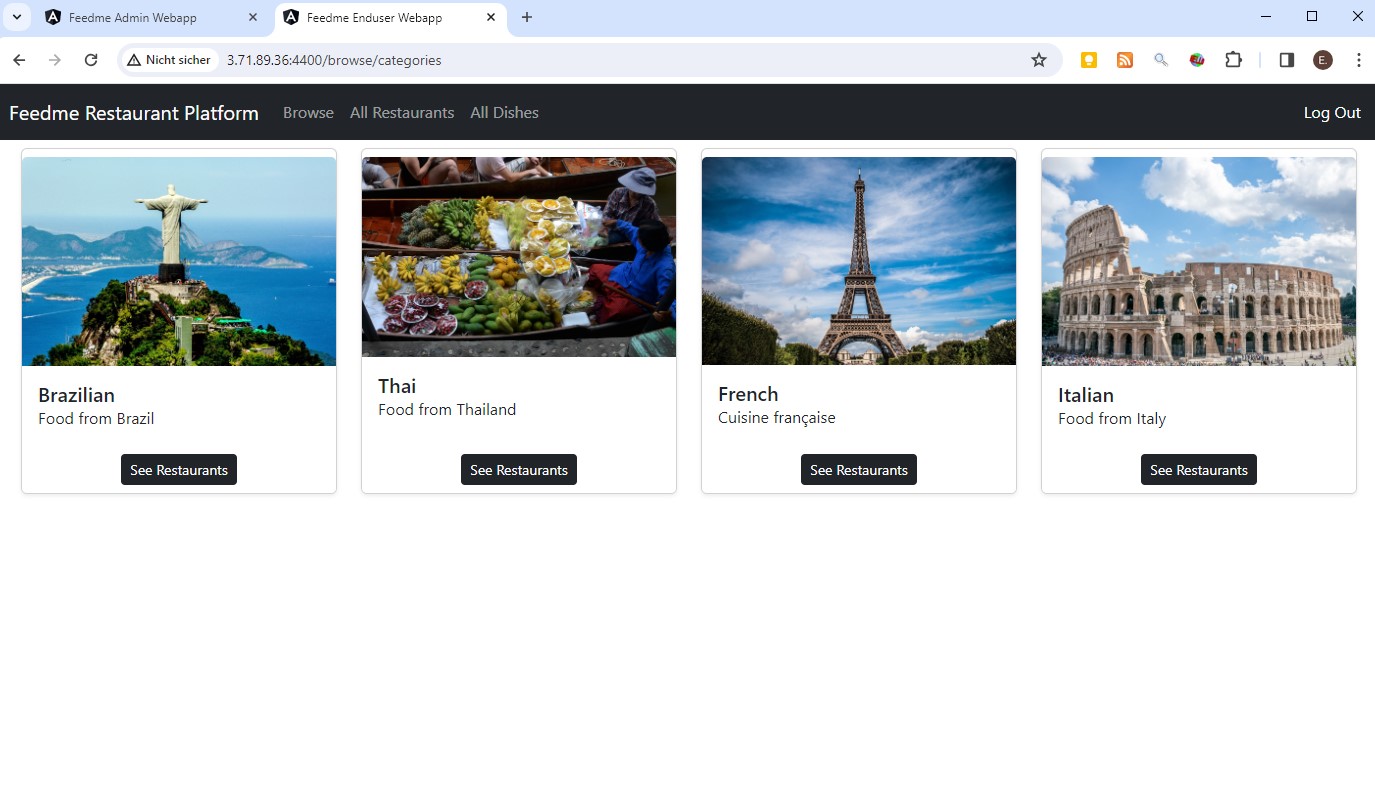
**Feedme - Online Restaurant Platform**

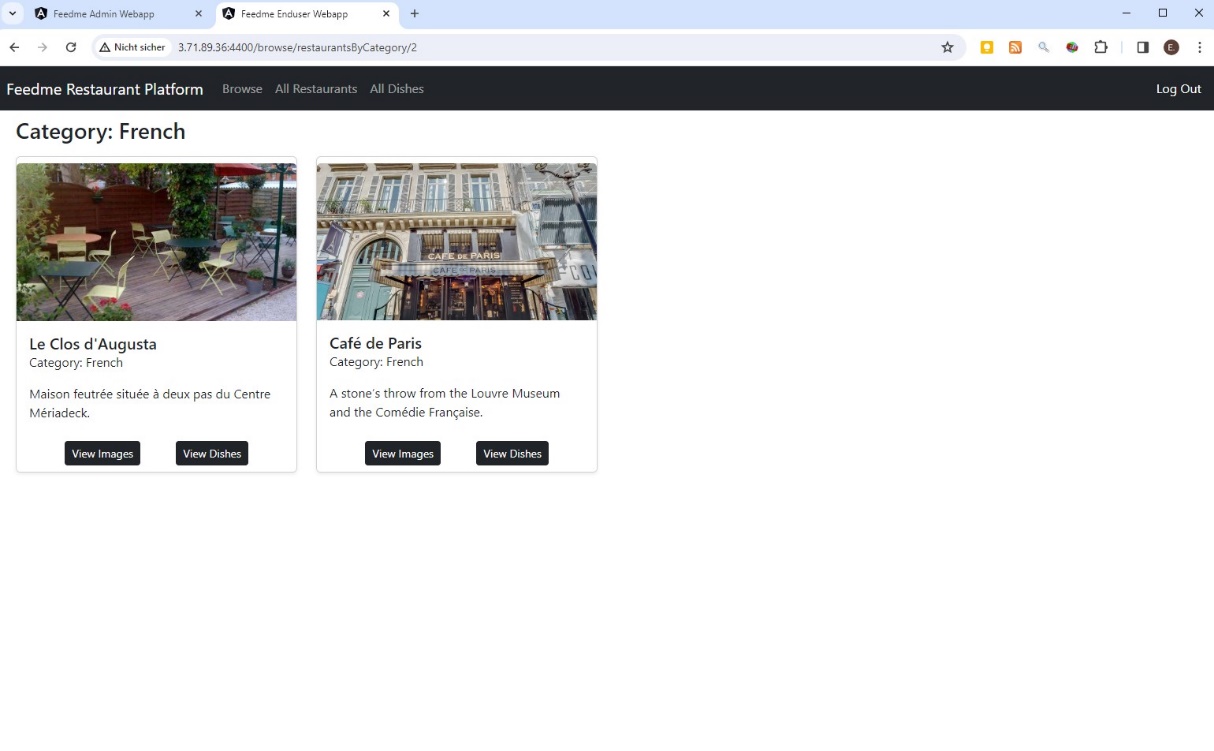
Program Capstone Project for the Course

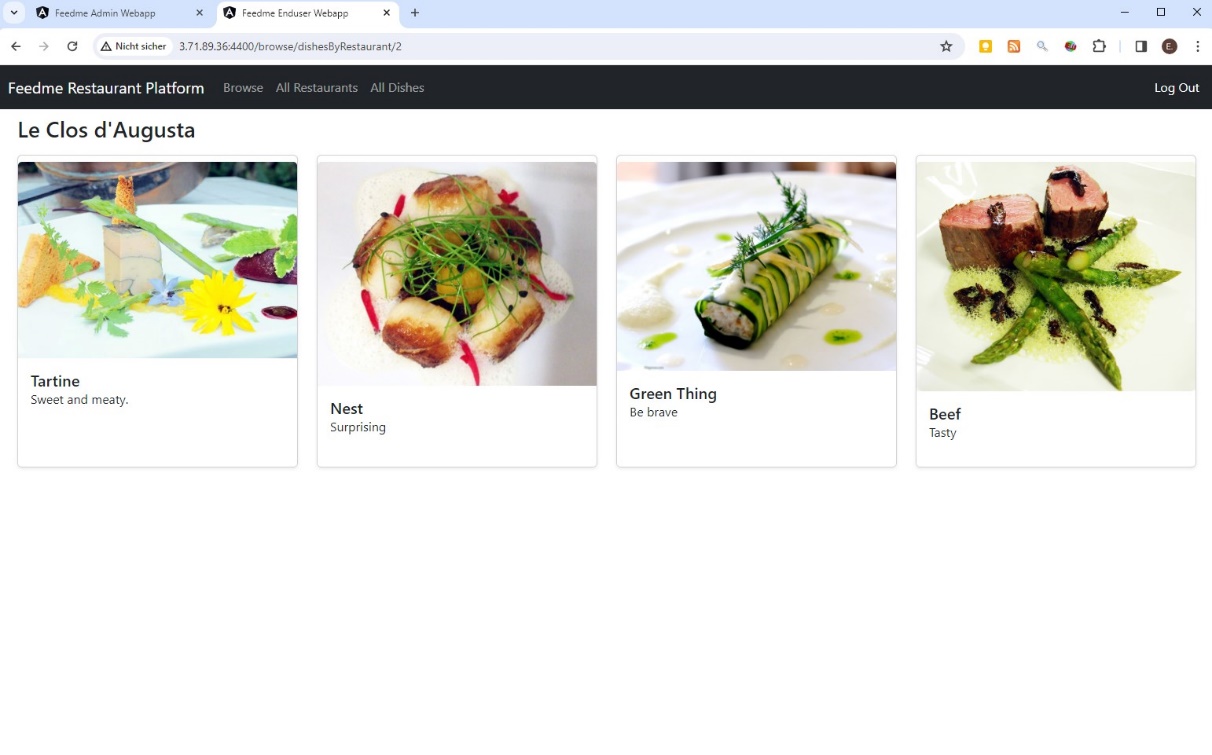
Caltech Full Stack Developer, Cohort February 2023

Author: Elia Colombo

Date: 04.01.2024







Contents

Seite

1 Heading 3

1.1 Project description 3

1.2 Tech stack 3

2 Architecture 4

3 SCRUM plan 5

3.1 Epic: Develop and Test Backend Webservice with Springboot and MySQL 5

3.2 Epic: Develop Frontend Admin Webapp with Angular and Bootstrap 5

3.3 Epic: Develop Frontend Enduser Webapp with Angular and Bootstrap 5

3.4 Epic: Setup infrastructure 6

3.5 Epic: Create Container running MySQL for persistence oft he application 6

3.6 Epic: Configure webservice Spring Boot application to be built and run in docker container 6

3.7 Epic: Configure enduser/admin Angular applications to be built and hosted in docker containers 6

3.8 Organisation 7

4 Project Git Repositories 7

5 How to run the project 7

6 Screenshots 8

6.1 Enduser Webapp running on EC2 instance 8

6.1.1 Dashboard 8

6.1.2 Authentication: Register / Login 9

6.1.3 Browse Categories 9

6.1.4 Browse Restaurants 10

6.1.5 Browse Dishes 12

6.2 Admin Webapp running on EC2 instance 13

6.2.1 Authentication: Register / Login 13

6.2.2 Manage Categories 13

6.2.3 Manage Restaurants 14

6.2.4 Manage Dishes 15

6.2.5 Manage Users 16

6.3 Development 17

6.3.1 Spring initializr 17

6.3.2 Backend Project Structure 17

6.3.3 Enduser Webapp Structure 18

6.3.4 Admin Webapp Structure 19

6.4 CI/CD 20

6.4.1 Jenkins pipelines on Jenkins running on the AWS EC2 instance: 20

6.4.2 Docker images running on the AWS EC2 instance: 20

6.4.3 Databases created by the Spring Boot Web Service in the mysql container: 20

# Heading

## Project description

The application “Feedme” developed for this capstone project is a web application where restaurants and their dishes can be comfortably browsed in the enduser webapp by registered end users (login required). The dishes can be either browsed filtered by category and restaurant, or consulted in lists. Information is accompanied by illustrative images. A dashboard gives a summary of the available information. Information and images can be comfortably entered by team of registered admins using the admin webapp (login required). The admin webapp also allows to manage the users.

This application is a proof of concept as demanded by the capstone project definition. But the solid Angular-Bootstrap interface backed by a Java-Springboot backend connected to a MySQL database for data persistence can be easily extended with additional information, properties, filter, search functions, security checks and be transformed into a fully functional ordering platform.

The Nginx webserver hosting the frontend enduser and admin webapp, the Java/Springboot backend webservice with RESTfull API and the MySQL database are executed within dedicated Docker containers that communicate via a docker bridge network. The containers are run on an Amazon AWS Elastic Cloud (EC2) instance.

The database feedme\_db is generated when its container is first run. All further table management and data transfer is then handled by the hibernate libraries of the spring boot application. Building and deployment off all other components is automated through Jenkins pipelines so that changes to the source code in the git repositories automatically triggers delivery and deployment of the updated code.

## Tech stack

Frontend:

* Angular CLI 16.2.2
* Bootstrap 5.3

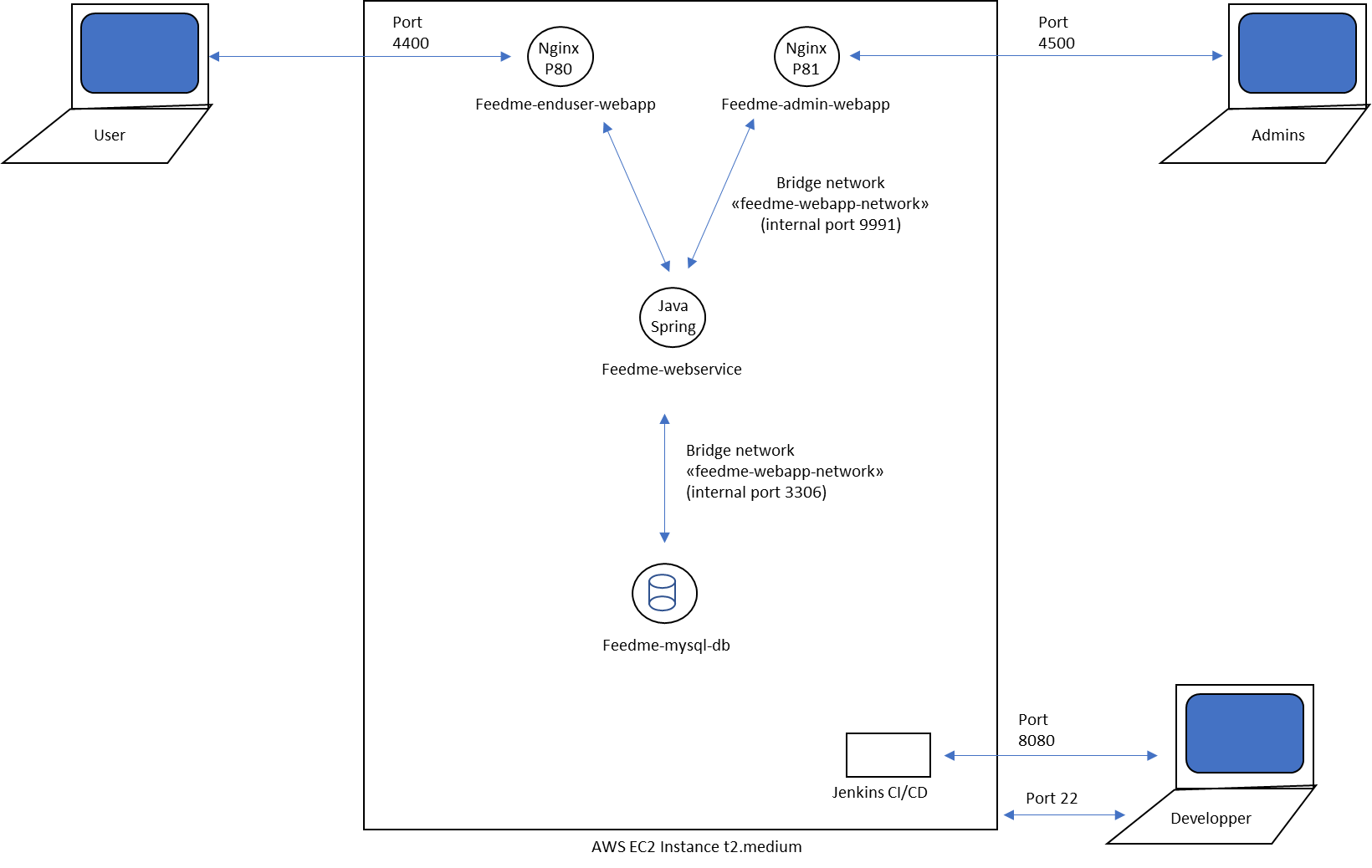
Backend:

* Springboot 3.1.4
* Java 17
* MySQL

Infrastructure:

* AWS EC2 t2.medium instance
* Ubuntu 22.04.3 LTS
* Docker 24.0.7
* Jenkins 2.414.3
* Maven 17.0.8.1

# Architecture



Inbound ports in use on the EC2 instance:

* Port 22 SSH to directly interact with the EC2 instance from my IP
* Port 8080 for Jenkins  
  (running directly on the EC2 instance)
* Port 9991 for the Feedme Webservice  
  (exposes internal port 9991 of the webservice in the Feedme-webservice container)
* Port 4400 for the Feedme Enduser Webapp   
  (exposes internal port 80 of the Nginx webserver running the webapp in the feedme-enduser-container)
* Port 4500 for the Feedme Admin Webapp  
  (exposes internal port 81 of the Nginx webserver running the webapp in the feedme-enduser-container)

# SCRUM plan

A backlog is elaborated based on the following epics. The tasks are assigned to Elia Colombo. The Sprints are built based on the backlog and scheduled over different weekends in December 2023.

## Epic: Develop and Test Backend Webservice with Springboot and MySQL

The Webservice stores data in entities (admins, users, restaurants, restaurant categories, dishes).

**Issues:**

* **implement entities, repositories, services, controller, dto and exceptions**  
  \*Description\*: Backend shall make available CRUD operations on all entities, communication shall occur via a RESTful API using HTTP, pagination shall be provided to reduce data transfer, data is saved in MySQL. The information is relational (e.g. dish 🡨 🡪 restaurant) so the key relationships have to be configured.
* \*Acceptance Criteria\*: All data and operations needed for Frontend are implemented
* \*Validations\*: Do Junit Tests to verify implementation (app starts, context is created, data is retrieved and transferred as expected, http status messages are sent, etc.).

## Epic: Develop Frontend Admin Webapp with Angular and Bootstrap

The Admin Frontend Application is to be used by registered admins of the platform (login required) to comfortably add and manage information and images of restaurant categories, restaurants, dishes, users. New admins and endusers can be registered.

**Issues:**

* **implement dahboard, services, components for manage & edit all entities**\*Description\* : implement clear, appealing and fast web application  
  Consume data RESTfully delivered by backend, use lazy loading by modularizing the components to manage the different components, use reactive forms in modal windows for data editing. Use AuthGuard authentication for the admin users pages.  
  \*Acceptance Criteria\*: information and images can be entered, user profiles can be registered and modified, parameters like name, description, etc. can be edited. New admin user can be registered.  
  \*Validations\*: Test and document the webapp with sample data

## Epic: Develop Frontend Enduser Webapp with Angular and Bootstrap

The Enduser Frontend Application is to be used by registered endusers (login required) of the platform to comfortably browse the dishes filtered by category and restaurant, or consulted in comprehensive lists. Information is accompanied by illustrative images. New enduser can register.

**Issues:**

* **implement dahboard, services, components for manage & edit all entities**\*Description\* : implement clear, appealing and fast web application. Consume data RESTfully delivered by backend, use lazy loading by modularizing the components to manage the different components, use reactive forms in modal windows for data editing. Use AuthGuard authentication for the user pages.  
  \*Acceptance Criteria\*: A dashboard page gives an overview of the stored information. Information can be browsed, restaurants can be viewed by category, dishes can be viewed by restaurant. Restaurants and dishes can be viewed in comprehensive lists. Images can be scrolled and viewed. New user can be registered.  
  \*Validations\*: Test and document the webapp with sample data

## Epic: Setup infrastructure

Jenkins and build tools shall run on AWS EC2 instance. The application components shall run in docker containers on the same instance

**Issues:**

* Configure Ubuntu instance and start it, define security group parameters for inbound traffic
* Install Java, Maven, Jenkins, Docker via ssh

## Epic: Create Container running MySQL for persistence oft he application

Data is loaded and saved from/to a mysql database running in its own docker container connected to the webservice component container via a docker bridge network and standard port 3306

**Issues:**

* Create MySQL Docker container from mysql base image, run it with adequate parameters to create application database (cabmanager\_db) and user

## Epic: Configure webservice Spring Boot application to be built and run in docker container

The webservice shall run in a container and receive and send data to mysql and handle inbound requests via port 9991. The application shall be built and deployed automatically when new code is committed to its github repository.

**Issues:**

* Create “production” profile to connect webservice to the container mySQL in the Docker network  
  spring.datasource.url = jdbc:mysql://**feedme-mysql-db**:3306/feedme\_db
* Create Jenkins pipeline triggered by changes in the github repository to build the Spring Boot Application with the “production” profile, create docker container and deploy it
* Adapt Dockerfile to generate docker container running the webservice, exposing port 9991 and connect webservice container to Docker bridge network

## Epic: Configure enduser/admin Angular applications to be built and hosted in docker containers

The enduser and admin webapps shall run on nginx webservers exposed to the Internet on port 4400 and 4500 respectively in separate containers and receive and send data to the webservice through the Docker network on port 9991. The webapps shall be built and deployed automatically when new code is committed to their github repository.

**Issues:**

* Create “production” profile to setup the connection to the webservice on the EC2 instance
* Create Jenkins pipeline triggered by changes in the github repositories to build the Angular web app with the “production” profile, create docker container with the application running it on a nginx webserver and deploy the container
* Adapt Dockerfiles to generate separate docker containers running the web apps on nginx web servers
* Communicate RESTfully with web service container via docker network port 9991
* Expose nginx port 80 and 81 to the internet on inbound port 4400 and 4500 respectively of the elastic cloud instance

## Organisation

**Backlog:** All of Above

**Tasks assigned to:** Elia Colombo

**Sprint:** December 2023

# Project Git Repositories

Backend Webservice: <https://github.com/ecolombo/feedme-webservice>

Frontend Admin Webapp: <https://github.com/ecolombo/feedme-admin-webapp>

Frontend Enduser Webapp: <https://github.com/ecolombo/feedme-enduser-webapp>

# How to run the project

Create AWS EC2 Ubuntu instance

Open inbound traffic on ports 4400, 4500, 8080 and 9991

Via SSH, install Java (>= v17), Maven, Jenkins, Docker.

Create docker bridge network feedme-webapp-network:

sudo docker network create feedme-webapp-network

Create and run mysql container and database:

sudo docker run --name feedme-mysql-db --network feedme-webapp-network -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=feedme\_db -d mysql

Enter the EC2 instance db address into the environment.ts file of the enduser and admin webapp

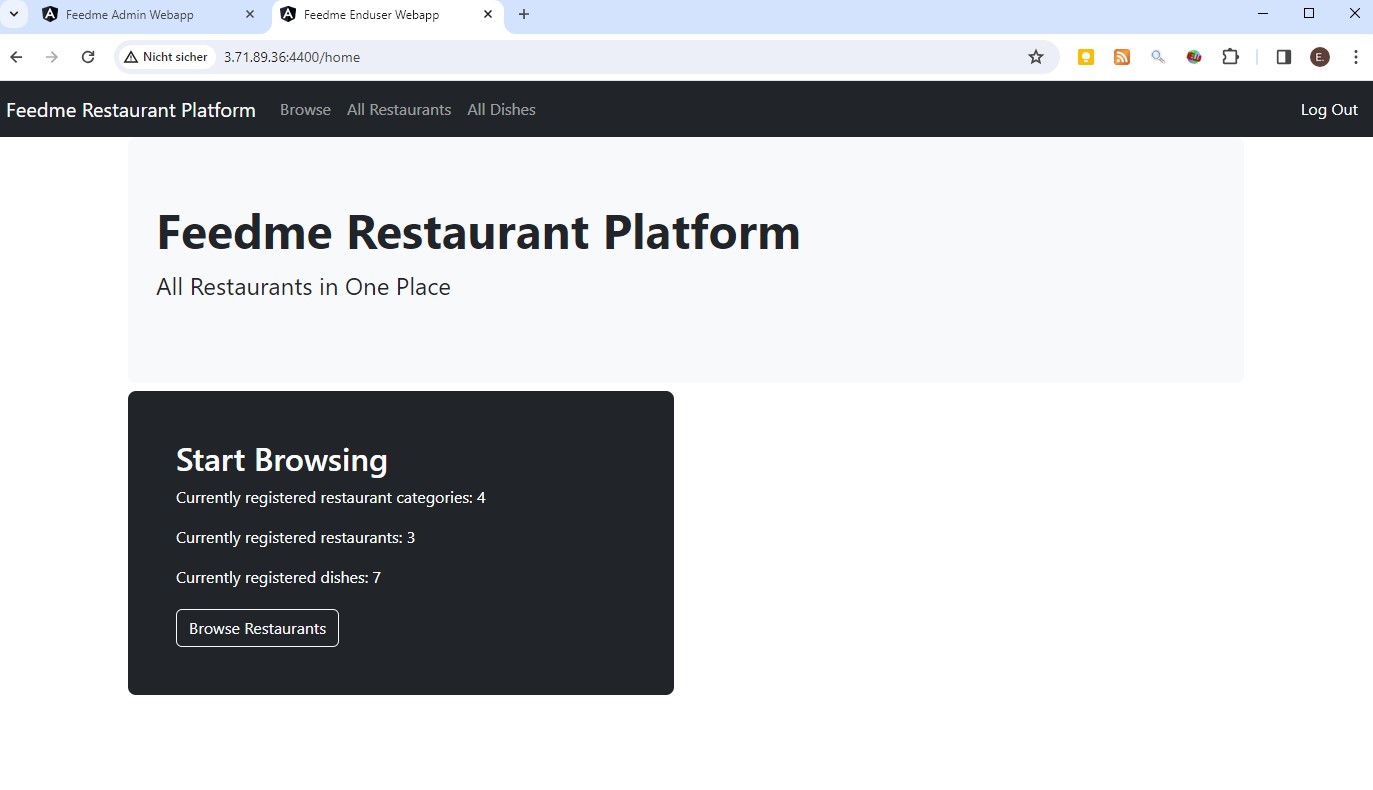
Create Jenkins pipelines based on the github repositories and execute them

Done. Access the application with your web browser via the instance hostname or IP and port 4400 (enduser) and 4500 (admins).

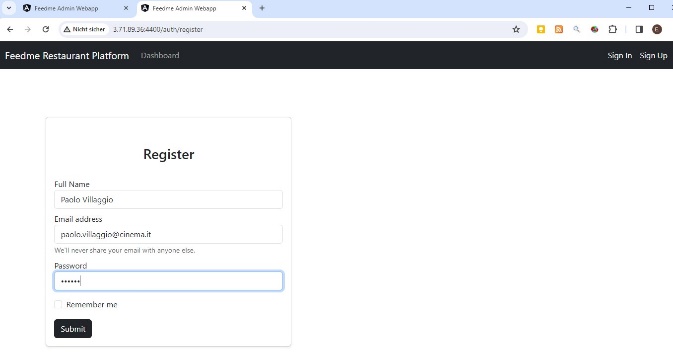
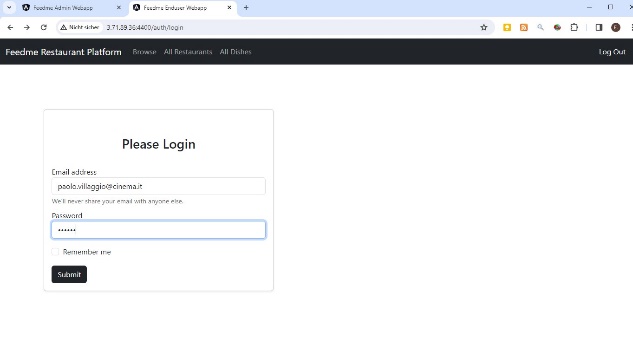
# Screenshots

## Enduser Webapp running on EC2 instance

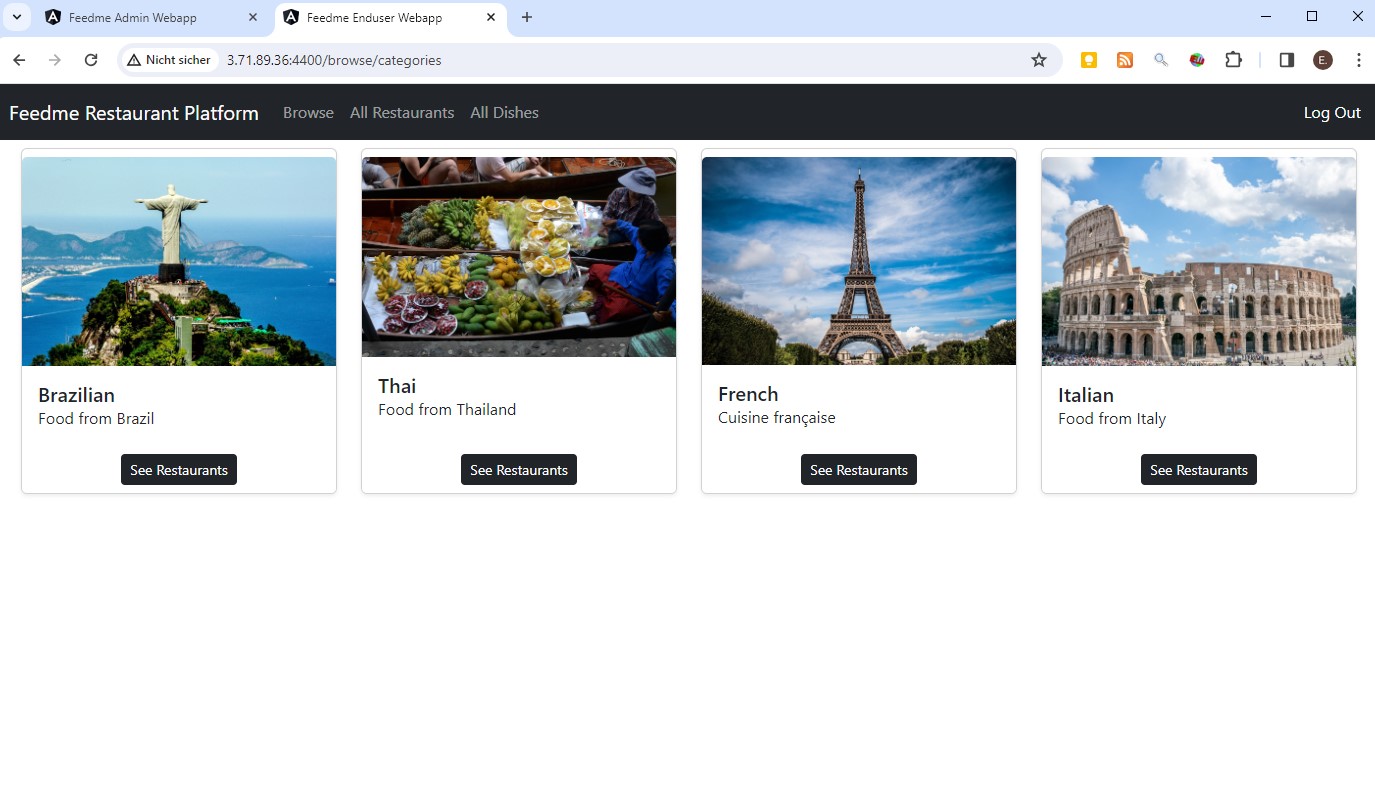
### Dashboard



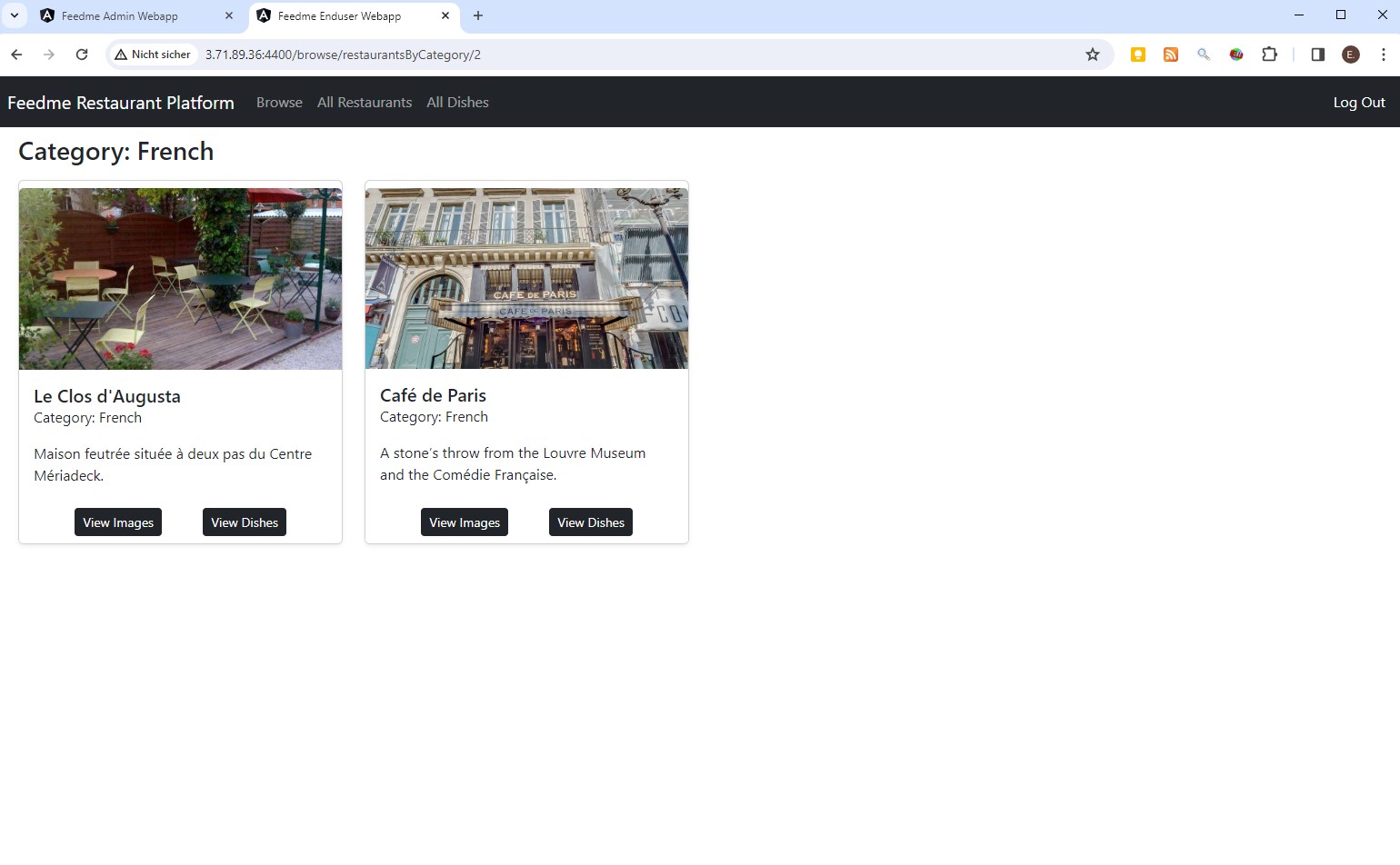
### Authentication: Register / Login

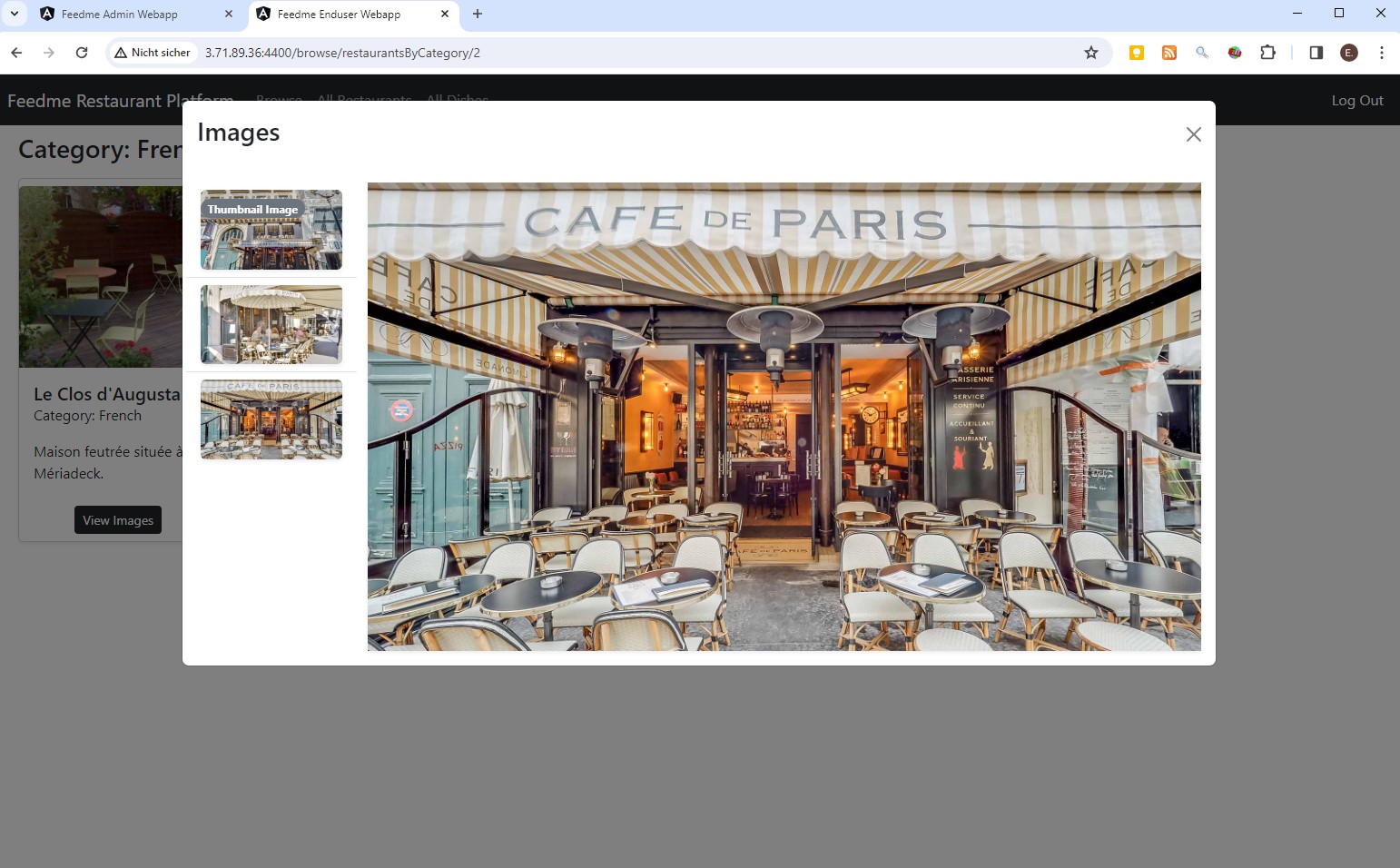
 

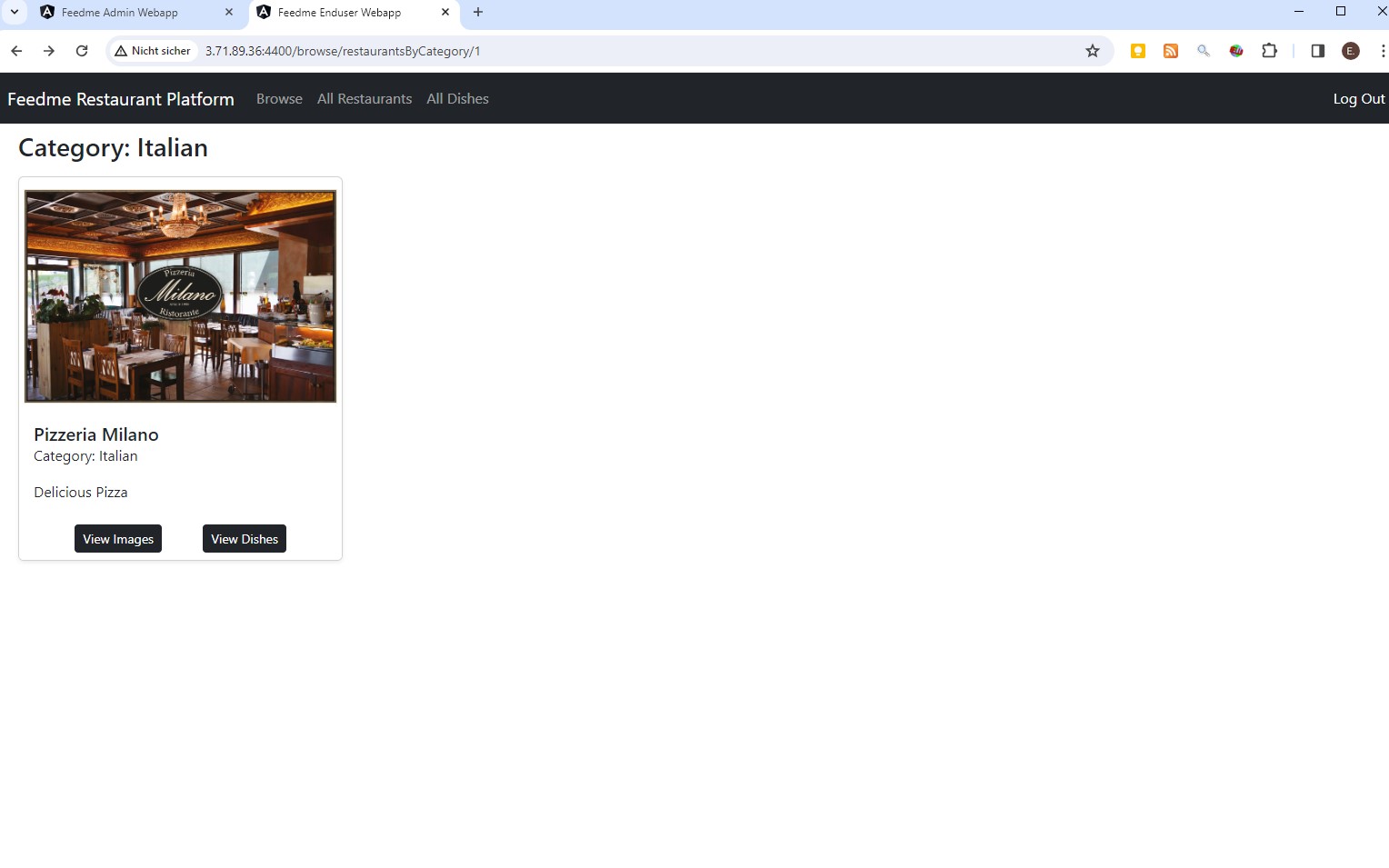
### Browse Categories

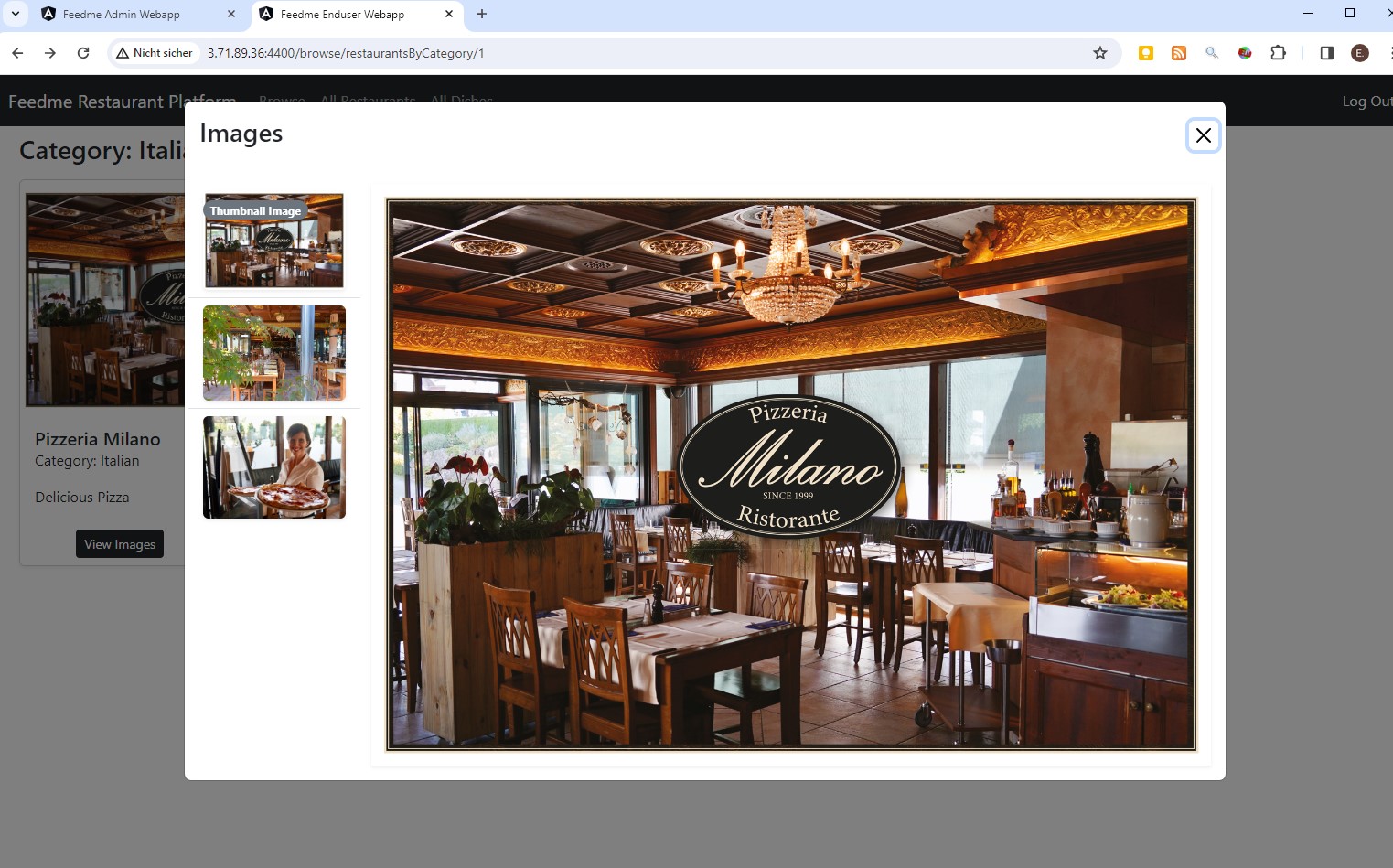


### Browse Restaurants

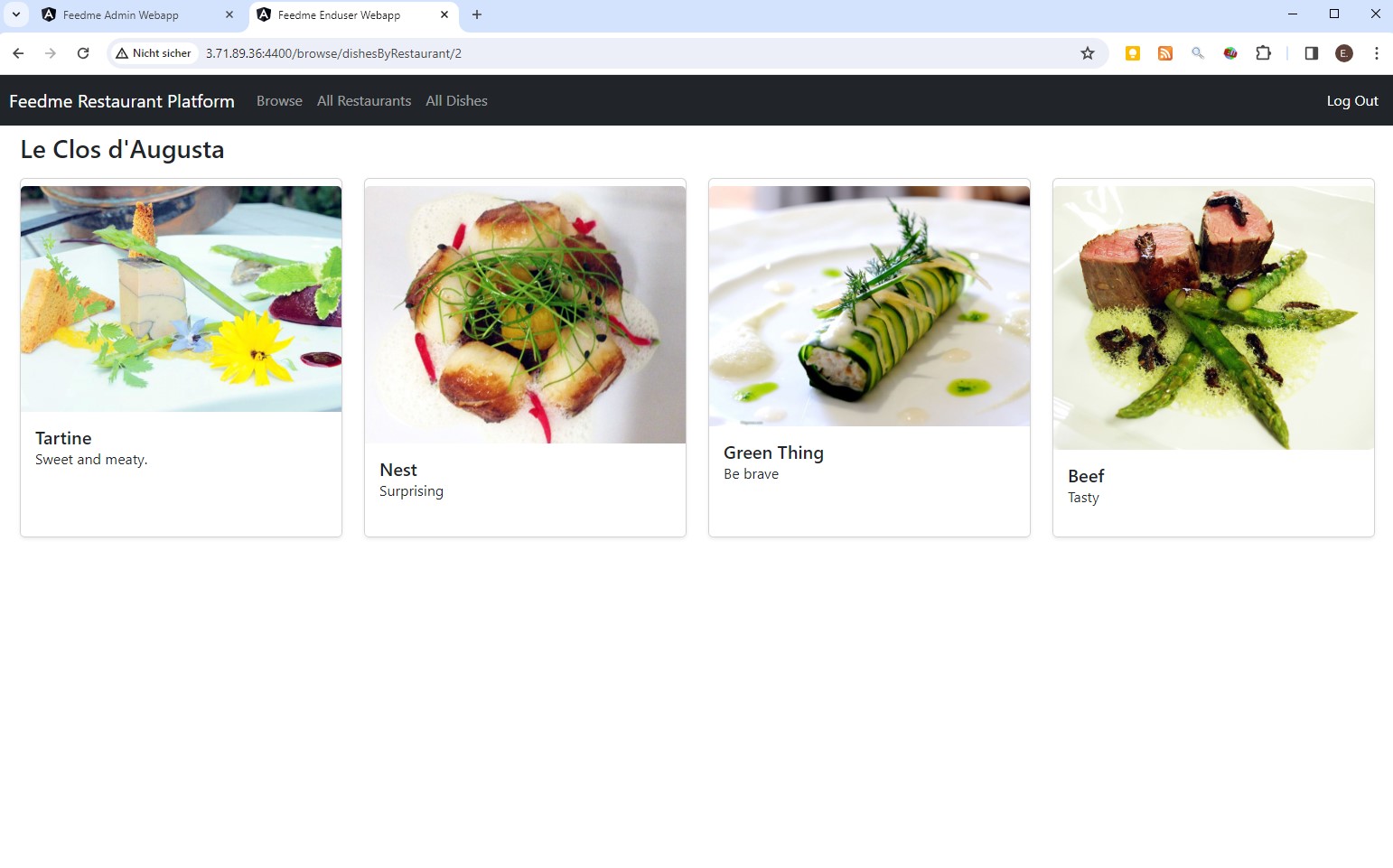


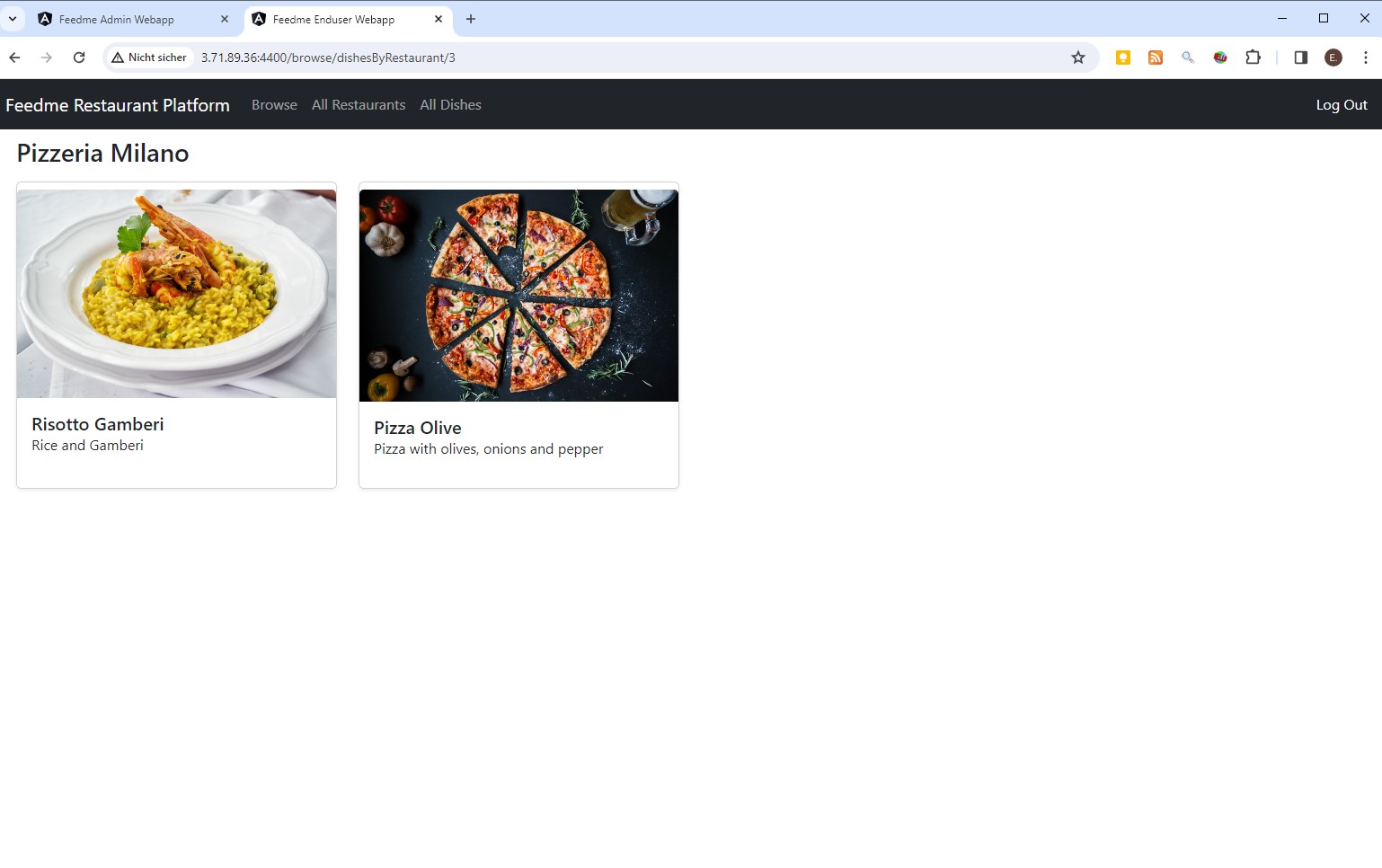






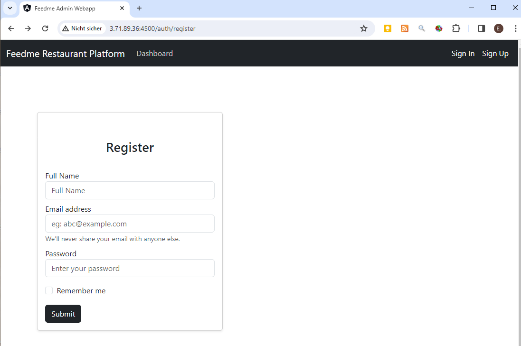
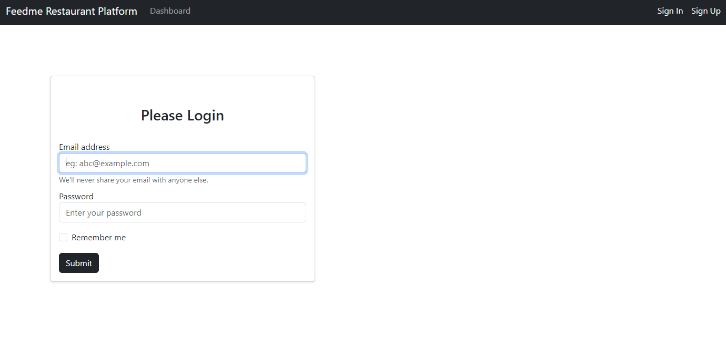
### Browse Dishes



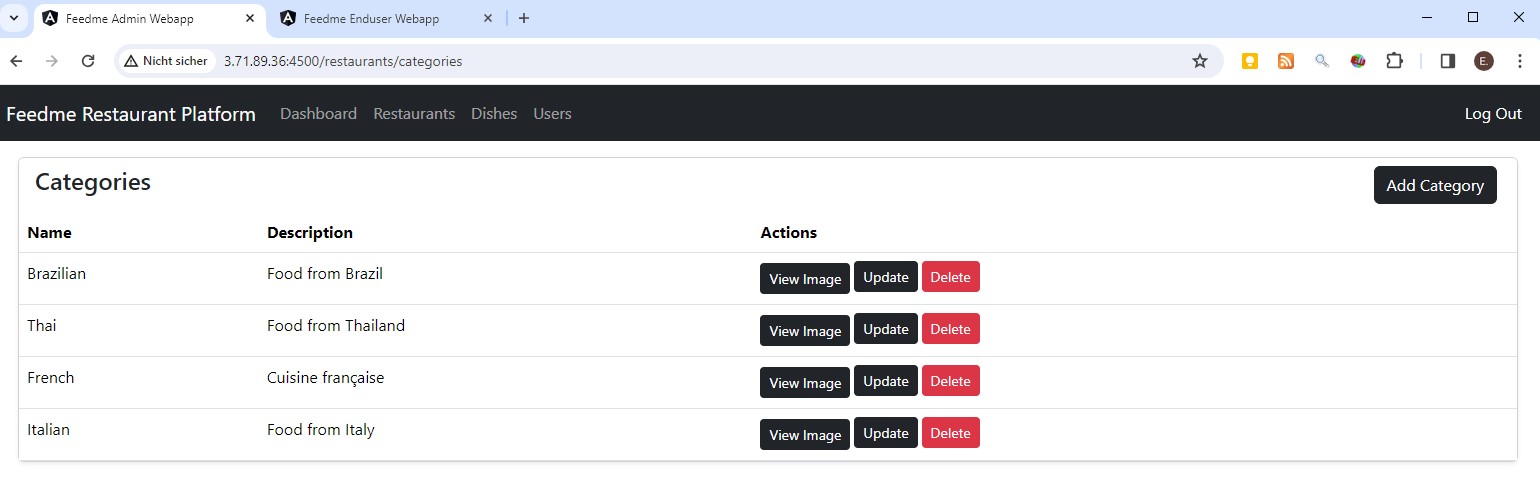


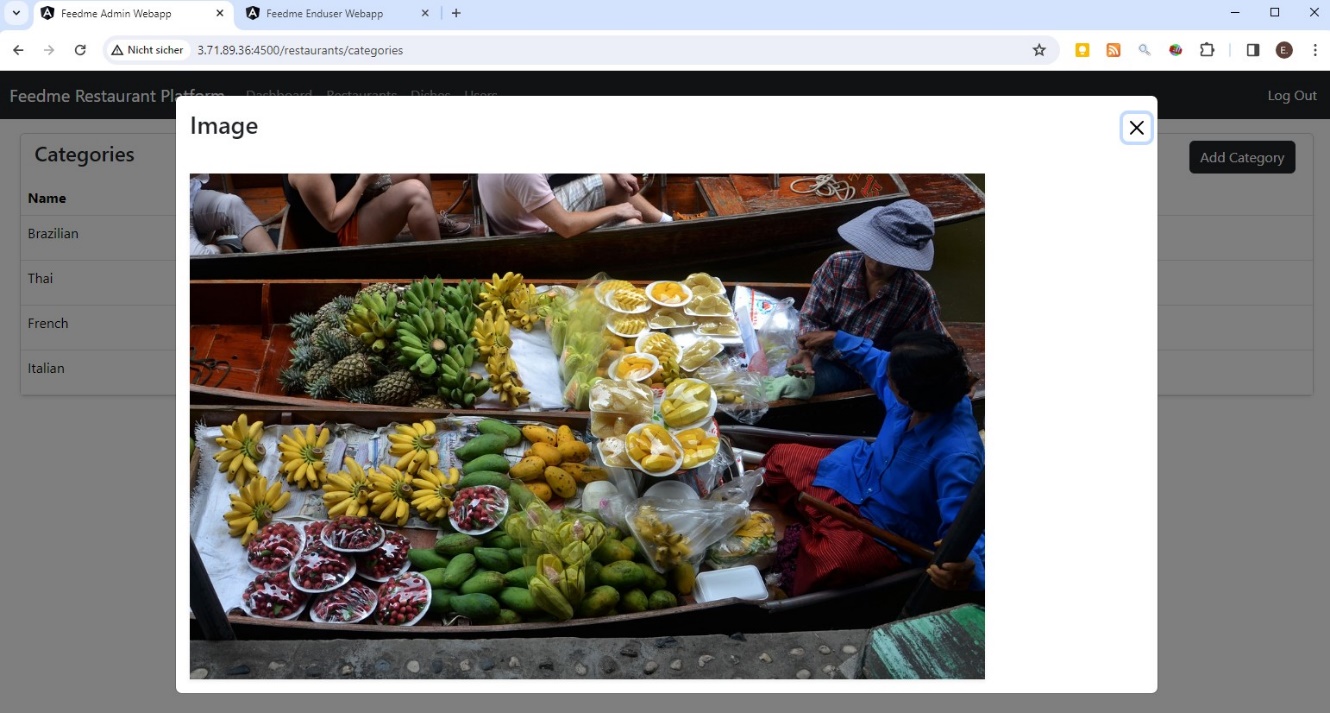
## Admin Webapp running on EC2 instance

### Authentication: Register / Login

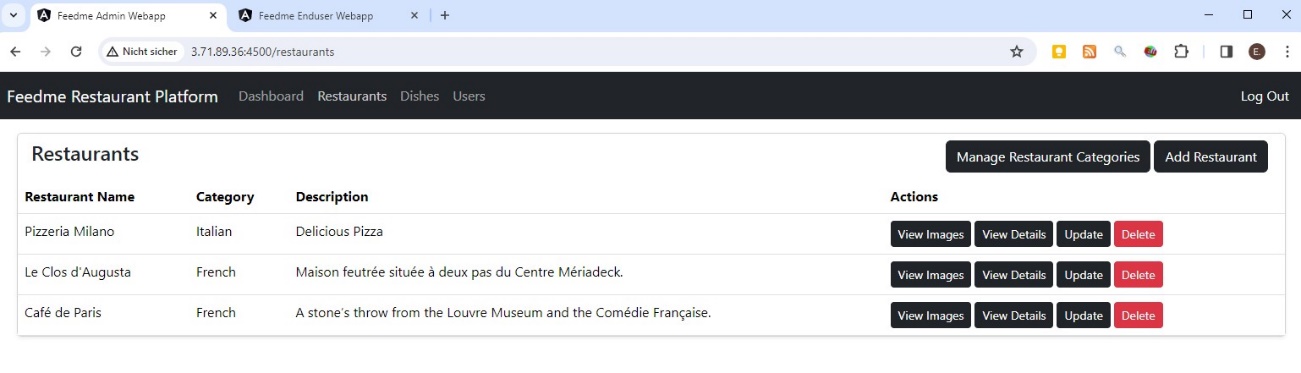
 

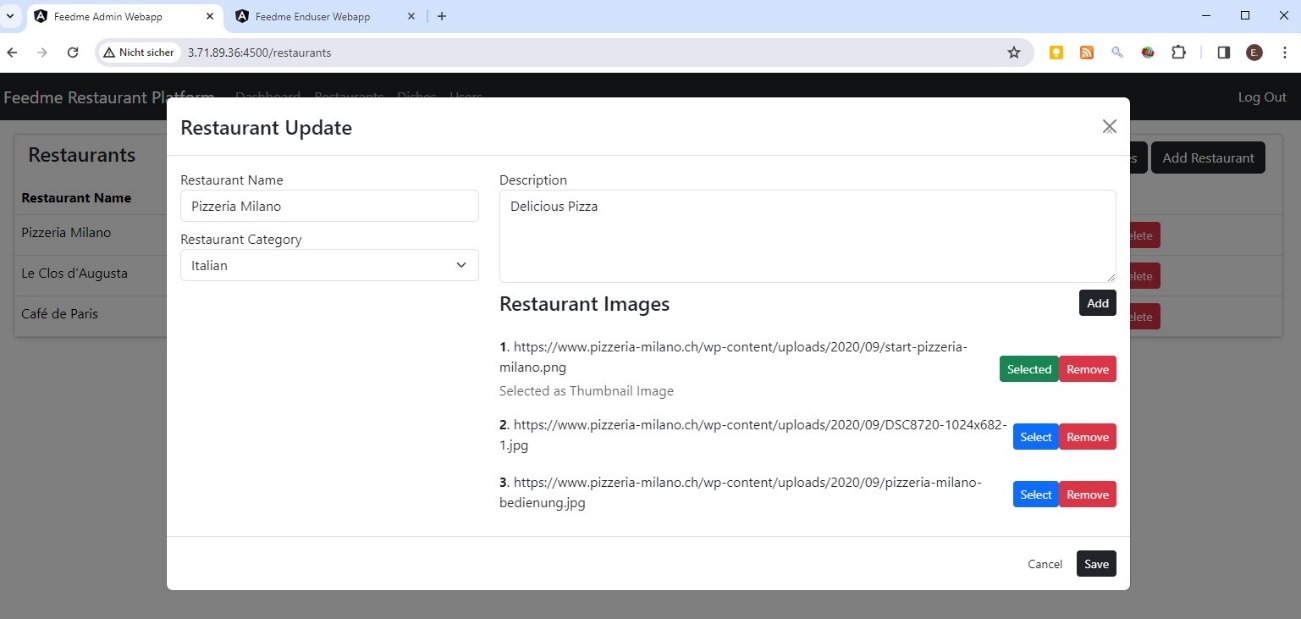
### Manage Categories

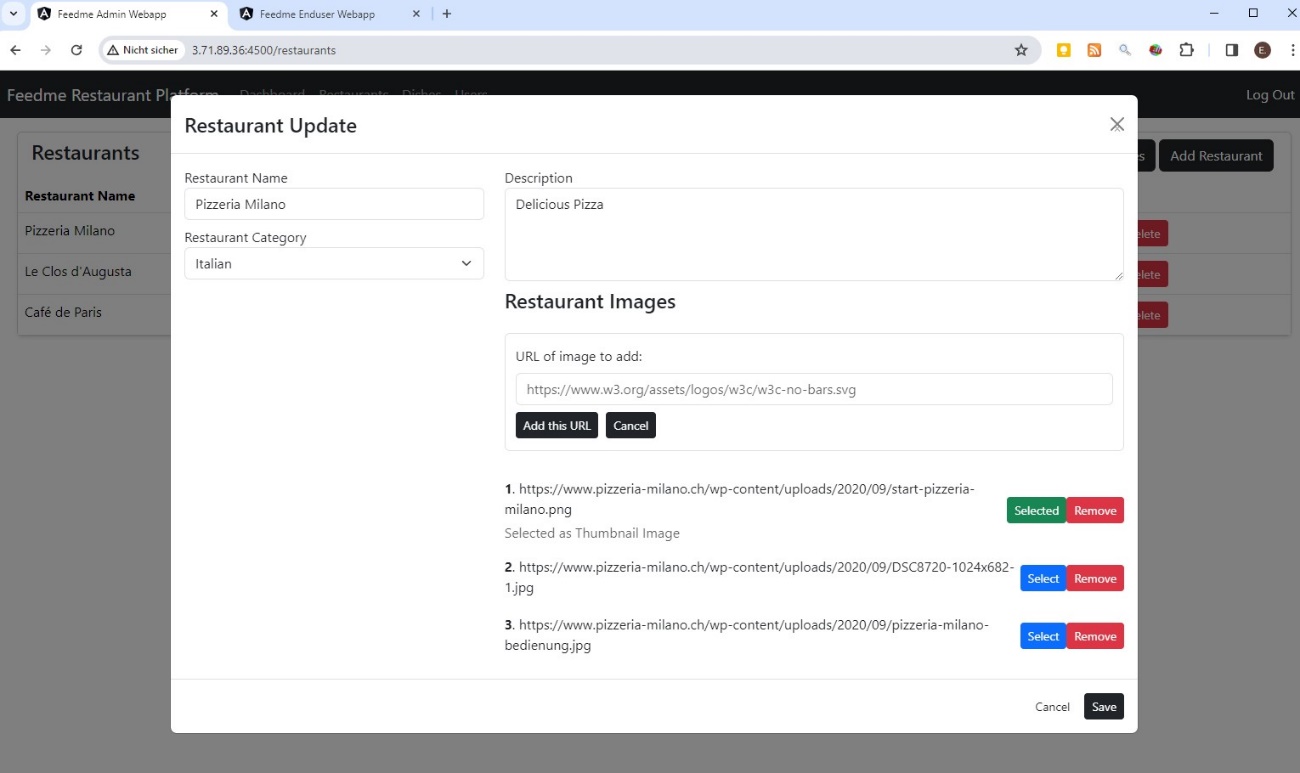




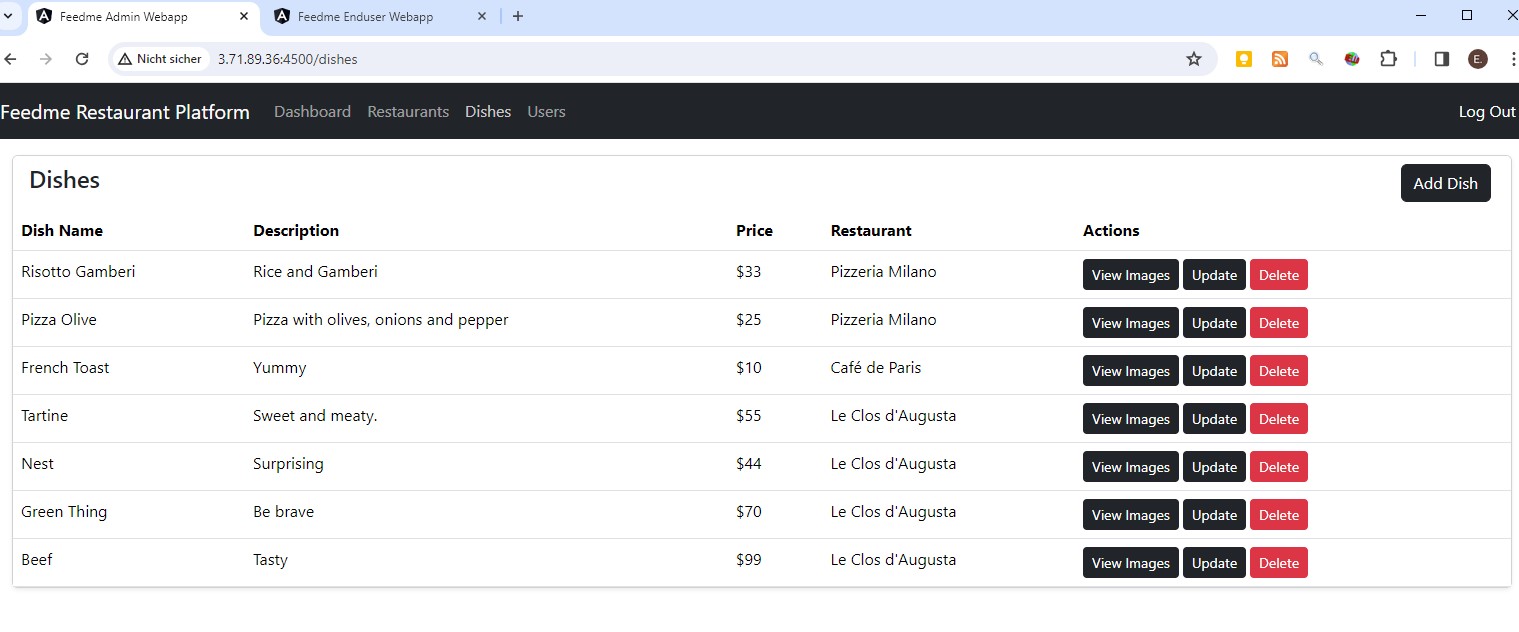
### Manage Restaurants

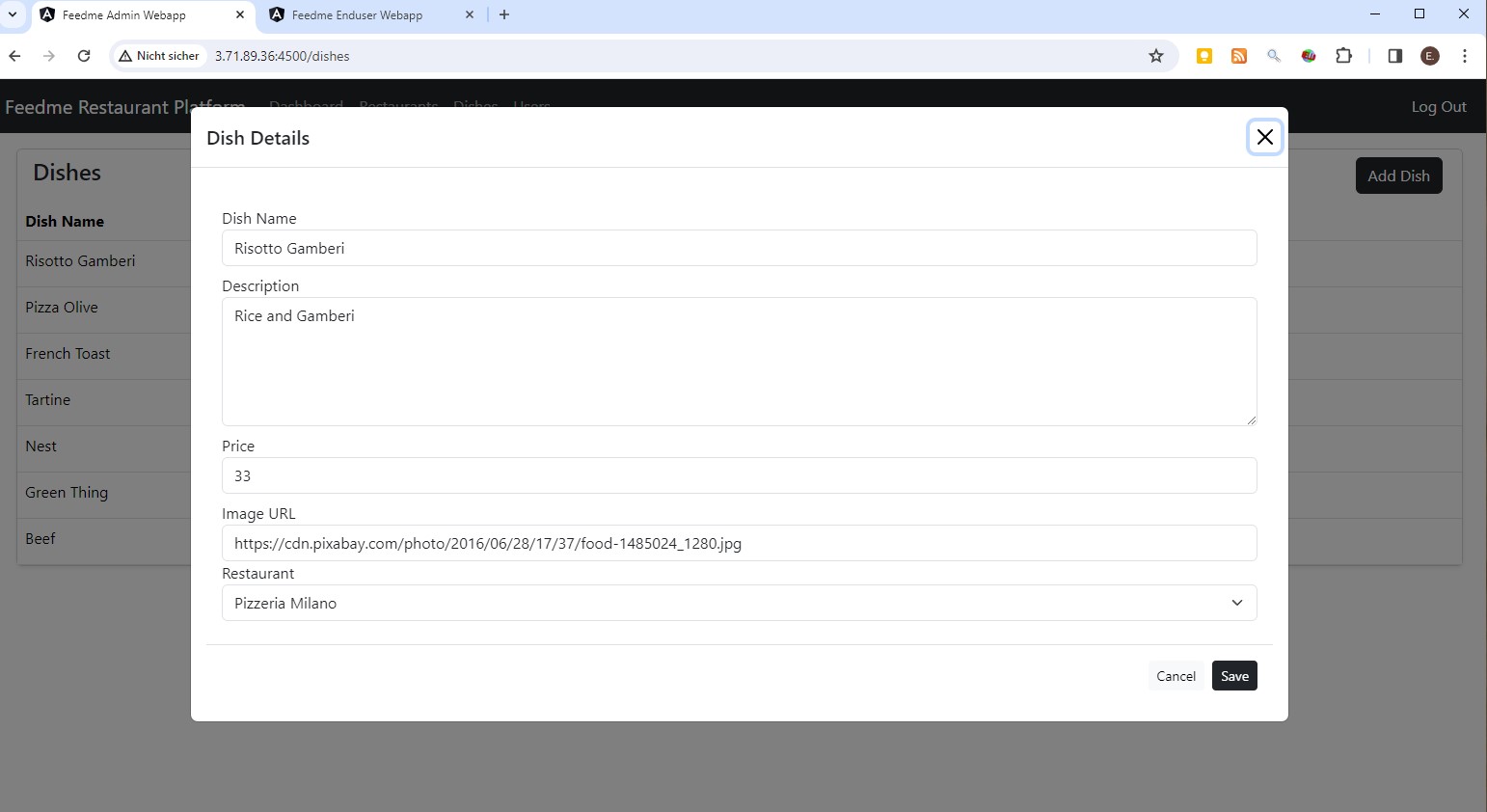




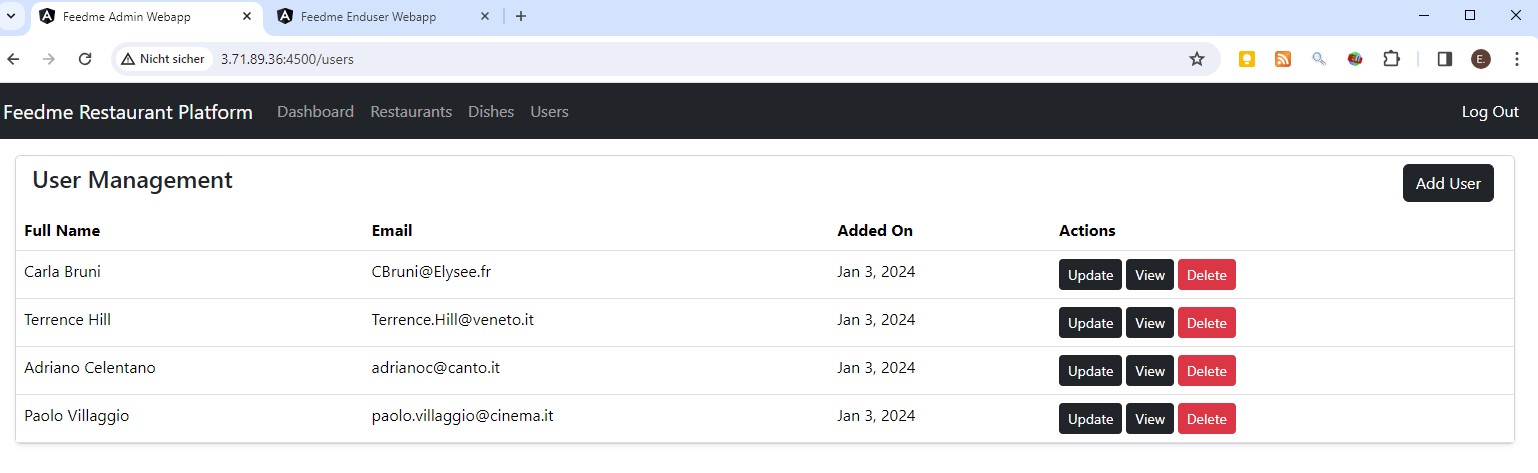


### Manage Dishes



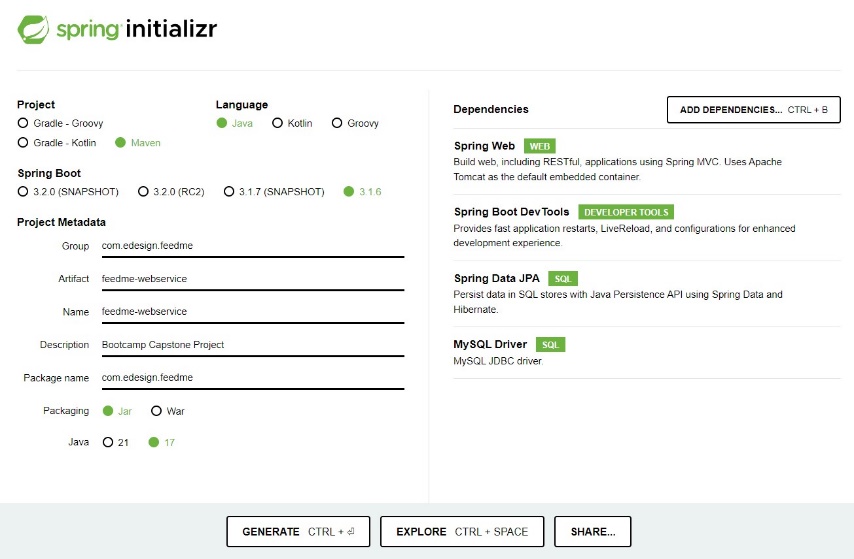


### Manage Users

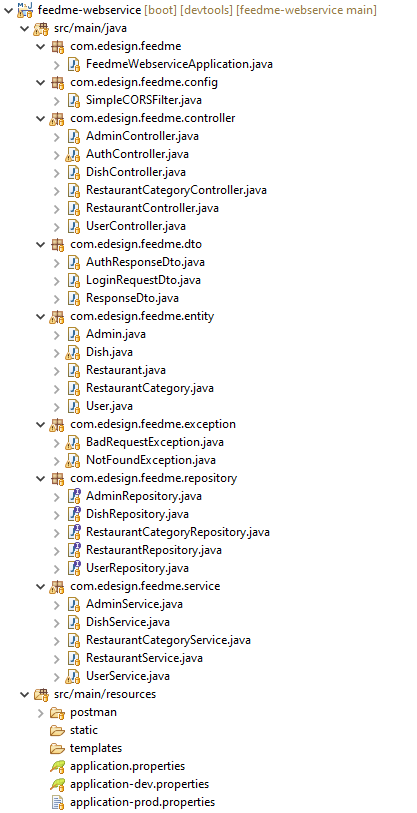


## Development

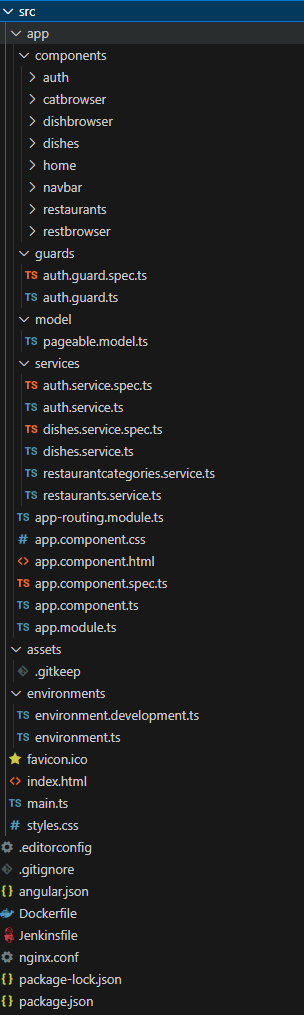
### Spring initializr



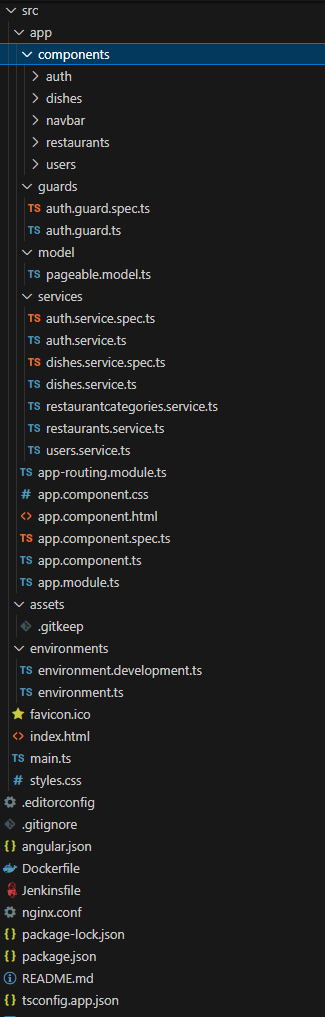
### Backend Project Structure



### Enduser Webapp Structure

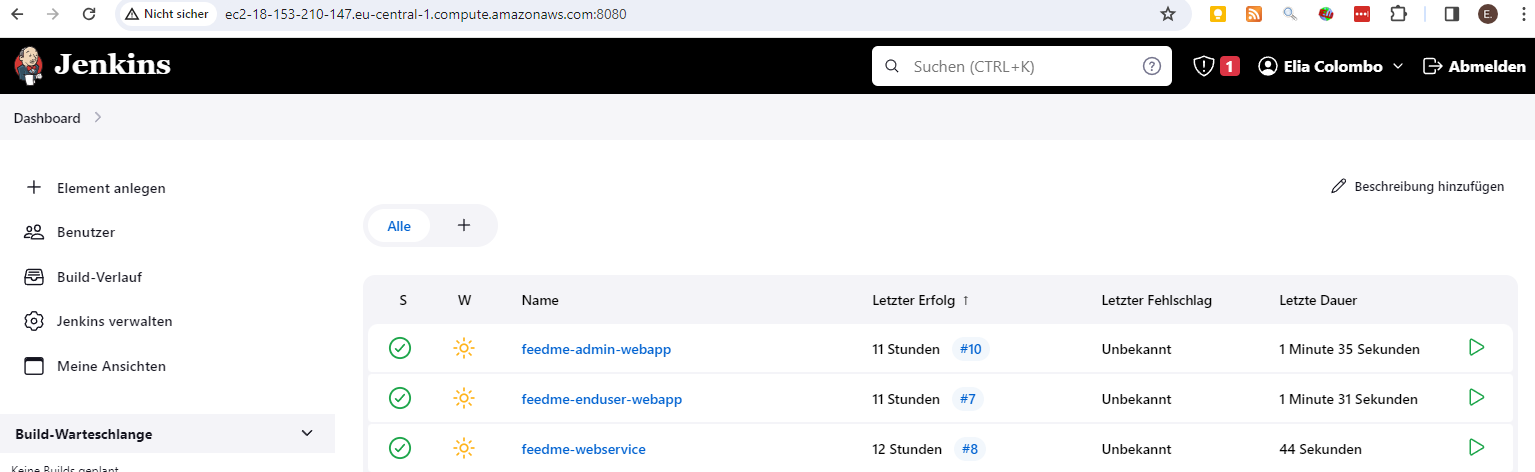


### Admin Webapp Structure

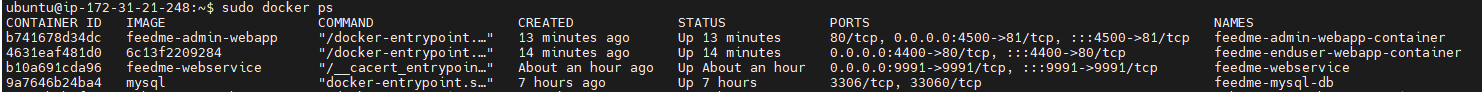


## CI/CD

### Jenkins pipelines on Jenkins running on the AWS EC2 instance:



### Docker images running on the AWS EC2 instance:



### Databases created by the Spring Boot Web Service in the mysql container:

