Pizza Service project

Developer: IVAN ZAMULA

Pizza Service Project is a REST API application. The keynote of business logic: this is an application which helps users to go through all cafes and pizzas in database, get their data and characteristics, search cafe by name and address, get all pizzas for cafe, search pizzas by name and so on. The admin users can add cafe and pizza, change the values of their data as well.

The Application has back-end and front-end parts. The back-end part is developed with Java and Java Spring Boot. It includes REST API endpoints, controllers and entities, h2-in-memory database. The front-end part is developed as a REACT application, that uses REST API endpoints from back-end and forms user interface.

Main functionality of the project

This project is a back-end solution for pizza-cafe management.

It would include the following main objects:

1. Users - there must be two types of users in app: admin, ordinary user (user)

Admin user has got all permissions in app: the main functionality would include the ability to create, read, update, and delete (CRUD) records for both pizzas and cafes. The functionality would be secured by requiring a username of "admin" to access these features. User has got only ability to READ cafe and pizza

2. Cafe - CRUD: Users with the "admin" username would be able to create new cafe records by providing information such as the cafe's name, location, and phone. They would also be able to view, update, and delete existing cafe records.

3. Pizza - CRUD: Users with the "admin" username would be able to create new pizza records by providing information such as the pizza's name, size, key_ingredients, cafe_id. They would also be able to view, update, and delete existing pizza records.

2. Requests

POST, DELETE and PUT requests: Users with the "admin" username would be able to create new resources by sending a POST request, delete existing resources by sending a DELETE request, and update existing resources by sending a PUT request. GET is working for all.

3. DB description

Cafe to Pizza has bidirectional @OneToMany relationship. This means that one cafe can have multiple pizzas, but each pizza can only belong to one cafe.

Add programmatically some pizzas and cafes. For this create class InitDataBase.java and add some data.

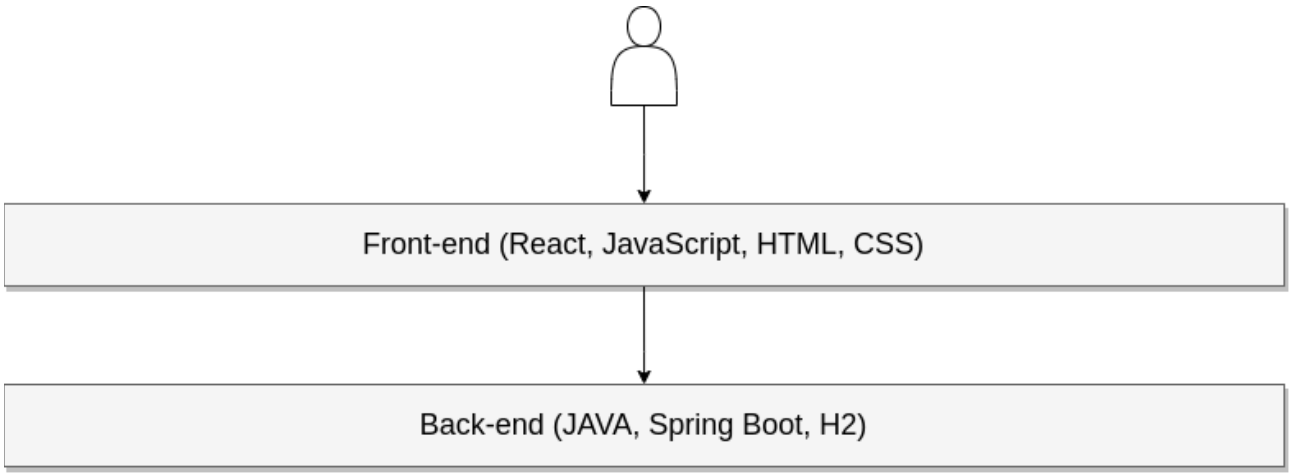
4. REST API endpoints

The application would also have the ability to retrieve all pizzas, retrieve a specific pizza, retrieve all cafes, retrieve a specific cafe and so on. For more information, see REST API description.

5. Security

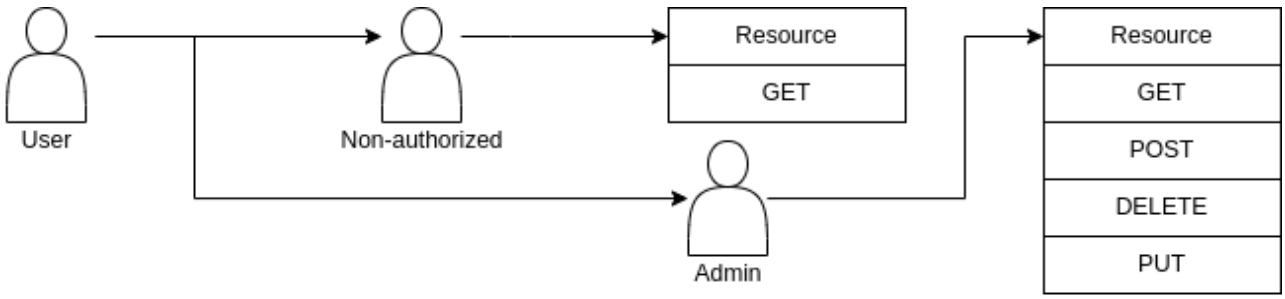
The application would be secured by requiring a username of "admin" to access the pizza and cafe CRUD functionality, as well as the POST, DELETE and PUT requests.

This could be implemented using Spring Security, which would check the user's credentials and restrict access to certain parts of the application based on their role.



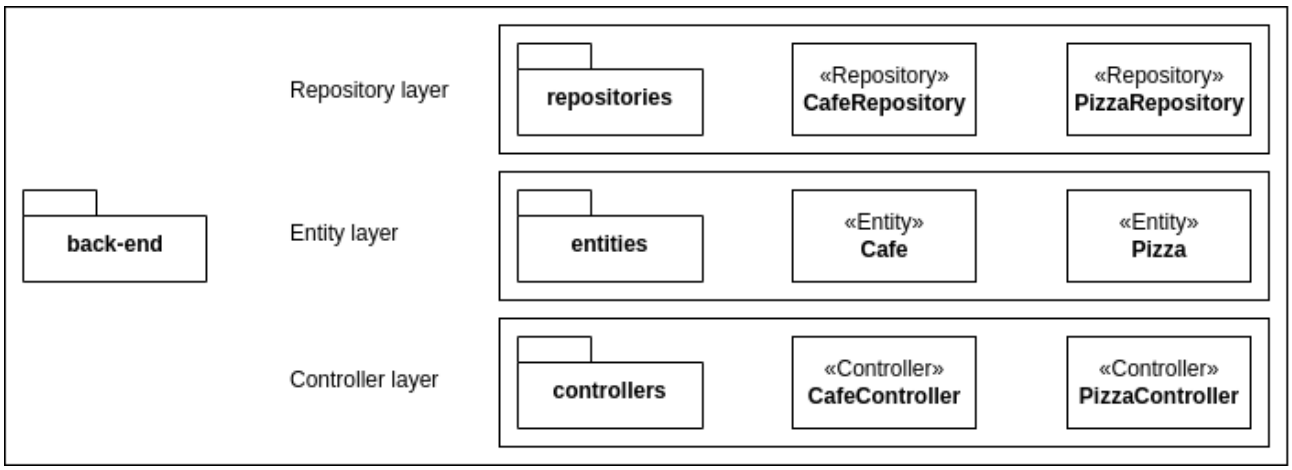
Users security

There are two user types in the Application: admin user and not-authorized user. The Application restrict access: not-authorized user is able to get responses only from REST API with GET methods. Admin user has the whole access to all endpoints.

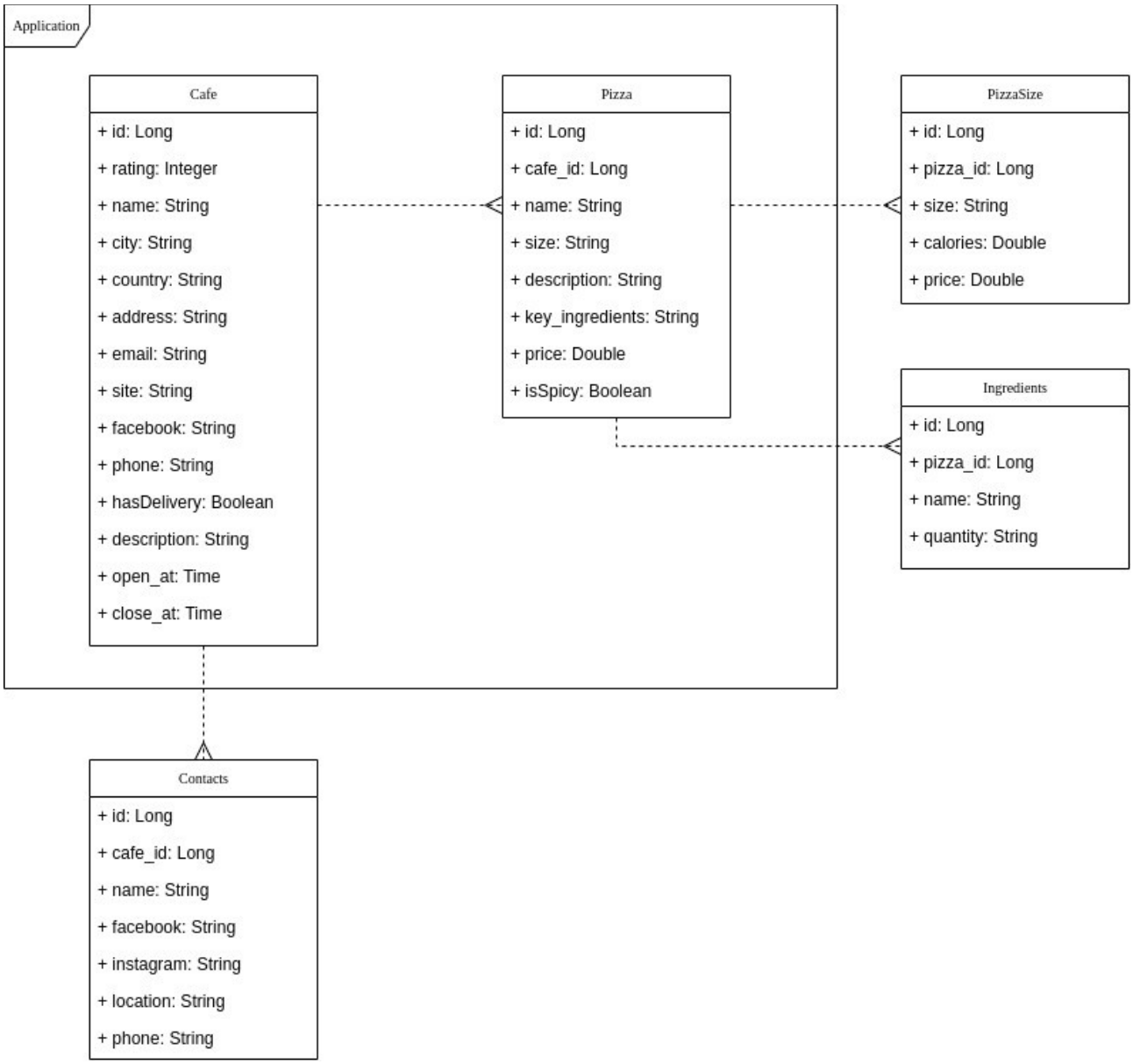


Back-end layers

The back-end part includes entities, repositories, controllers, H2-in-memory database, import.sql data loader, application.properties as settings file and so on. As a data storage is used a H2-in-memory database. The main advantage of this database is keeping all data in memory, that helps to give a quick access to data. H2 Database is very easy to use. Lightweight and Fast − H2 database is very lightweight and being in memory, it is very fast. Switch configurations − Using profiles, you can easily switch between production level database and in-memory database.



Back-end layersThere are two entities in the Application: Cafe and Pizza. The following schema gives a description of the entities and fields. There are also a few entities that give us a variant of possible evolution of the Application: PizzaSize, Ingredients, Contacts.



REST API description

All Application functionality must give the whole CRUD interface with REST API endpoints. The following table consist on description of all REST API endpoints, methods, access restriction and expected responses.

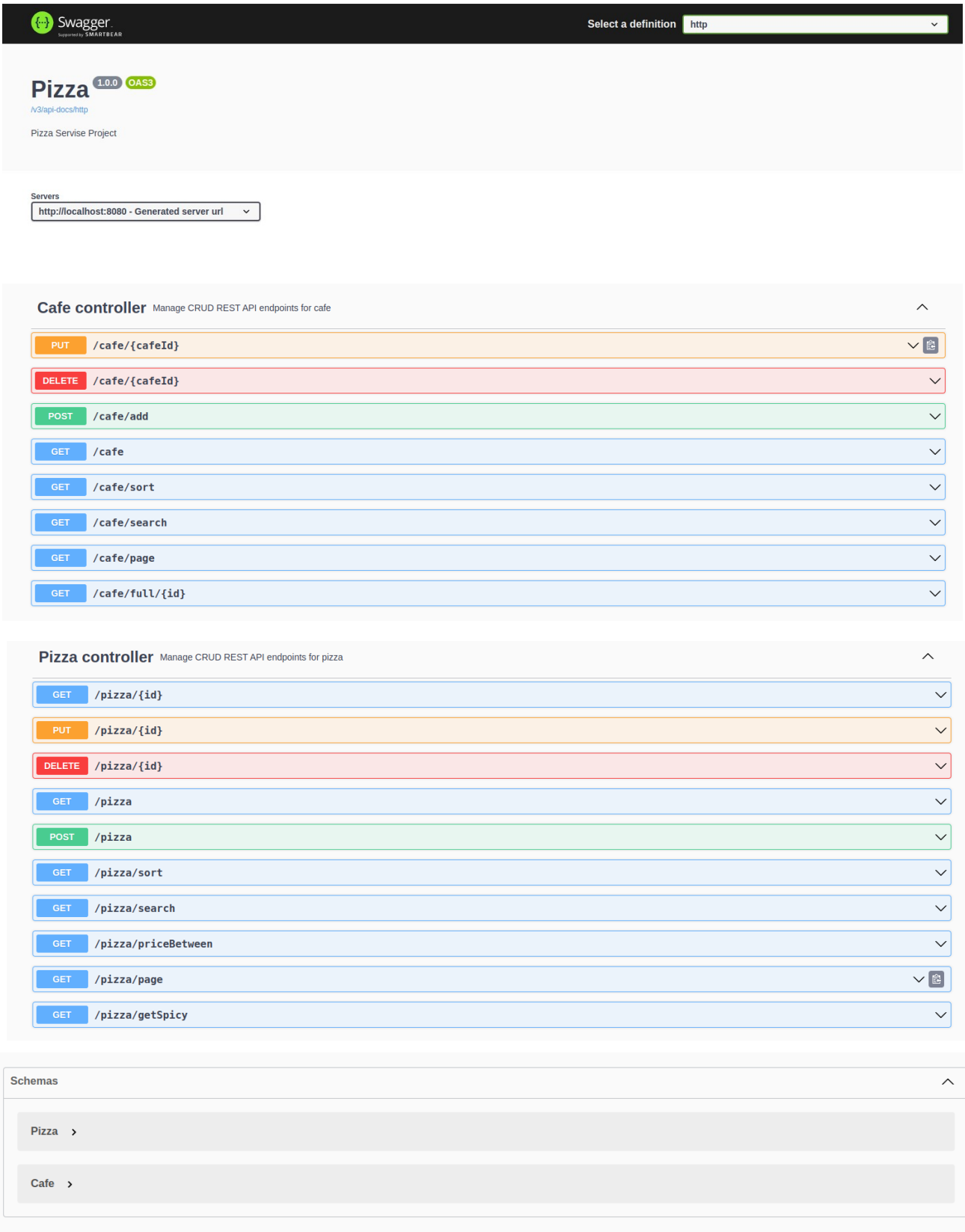
#	Operation	Method	Resource Path	Access	Result example (example with Curl)
1	List all cafes	GET	/cafes	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/cafes
2	Add a new cafe	POST	/cafe	Admin	(with auth) curl -v -H "Content-Type: application/json" -X POST -u "admin:admin" -d '{"name":"Pizza Point","city":"Berlin","address":"Haupstrase 1","email":"pizzapoint@gmail.com","phone":"0034 231 234","open_at":"09:00:00","close_at":"21:00:00"}' http://localhost:8080/cafe (without auth expected to fail)curl -v -H "Content-Type: application/json" -X POST -d http://localhost:8080/cafe
3	Get cafe by id with all pizza details listed	GET	/cafe/full/{id}	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/cafe/full/1
4	Update cafe details (identified by id)	PUT	/cafe/{id}	Admin	curl -v -H "Content-Type: application/json" -X PUT -u "admin:admin" -d '{"name":"Update","city":"Dresden","address":"Haupstrase 1","email":"pizzapoint@gmail.com","phone":"0034 231 234","open_at":"09:00:00","close_at":"21:00:00"}' http://localhost:8080/cafe/1 (without auth expected to fail)curl -v -H "Content-Type: application/json" -X PUT -u http://localhost:8080/cafe/1
5	Delete cafe by id	DELETE	/cafe/{id}	Admin	curl -v -H "Content-Type: application/json" -X DELETE -u "admin:admin" http://localhost:8080/cafe/1 (without auth expected to fail)curl -v -H "Content-Type: application/json" -X DELETE http://localhost:8080/cafe/1
6	Basic search by cafe address (should return	GET	/cafes?address={cafe address}	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/cafes?address=BERLIN

	all cafes whose name contains search term)				
7	List all pizzas of specific cafe	GET	/pizzas?cafe_id={id}	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/pizzas?cafe_id=1
8	Add new pizza to specific cafe	POST	/pizza	Admin	curl -v -H "Content-Type: application/json" -X POST -u "admin:admin" -d '{"name":"Gawaii", "size":"XL", "key_ingredients":"pineapple", "price":25, "cafe_id":4}' http://localhost:8080/pizza (without auth expected to fail)curl -v -H "Content-Type: application/json" -X POST -u "admin:admin" -d http://localhost:8080/pizza
9	Get specific pizza details	GET	/pizza/{id}	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/pizza/1
10	Update pizza details (by pizza id)	PUT	/pizza/{id}	Admin	curl -v -H "Content-Type: application/json" -X PUT -u "admin:admin" -d '{"name":"Updated", "size":"XL", "key_ingredients":"pineapple", "price":12, "cafe_id":4}' http://localhost:8080/pizza/1 (without auth expected to fail)curl -v -H "Content-Type: application/json" -X PUT -d http://localhost:8080/pizza/1
11	Delete specific pizza	DELETE	/pizza/{id}	Admin	curl -v -H "Content-Type: application/json" -X DELETE -u "admin:admin" http://localhost:8080/pizza/1 (without auth expected to fail)curl -v -H "Content-Type: application/json" -X DELETE -u "d:d" http://localhost:8080/pizza/1
12	List all pizzas from database	GET	/pizzas	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/pizzas
13	Basic search by pizza name (should return all pizzas whose name contains search term)	GET	/pizzas?name={pizza name}	Any	curl -v -H "Content-Type: application/json" -X GET http://localhost:8080/pizzas?name=MARGARITA

Swagger

Swagger is installed to look through all REST API endpoints and it gives the information about REST API endpoints that have been developed.

Swagger is a set of open-source tools built around the OpenAPI Specification that can help you design, build, document and consume REST APIs. The major Swagger tools include: Swagger Editor – browser-based editor where you can write OpenAPI definitions.



Front-end pages

The front-end part is developed with REACT framework. It used JavaScript templates to create pages and includes the main pages for list cafes, pizzas, search results, add and update cafe, add and update pizza.