Software Systems Architecture

Asynchronous Communication

Goal	In distributed systems, in particular Microservices, the parts need to exchange information to support the functionalities for the users. There are many approaches to that, but one specially important for Microservices is using Asynchronous Communication style. The goal of this lab is to put two Microservices to communicate using RabbitMQ, a message broker that follows asynchronous style.

Task	**Domain:** The goal is to create a Website for visualizing culinary incidents of a learning cook: what have been the problems with the attempts of recipes? **General Vision:** The following are the responsibilities of each Microservice in the architecture: ● Creator: a Web server to handle the registration of new culinary incidents. ● Querier: a Web server dedicated to supplying data for readers, such as external clients or user interfaces. Although we focused on culinary incidents, the domain is generic. It is as follows: - An **item** has a name, and represents something to be listed [in this case, a recipe]; - A **category** is a grouping criterion [here, a problem that happened]; - An **item** have a single **category,** but a **category** can have many **items;** For example, the time I tried to do canederli [**item**] but forgot to add eggs [problem = *missing ingredient,* **category**]; and also, once I made pancakes [**item**] and forgot the sugar [problem = *missing ingredient,* **category**]. So the two recipes suffered from the same problem. **Usage of LLM and IA tools:** you can use <u>exclusively</u> to <u>**assist in bug fixing**</u>, under the condition that no code is directly copied-and-pasted into your code; other uses are not allowed.

Spec	**Microservices (Web servers)**: use Java + SpringBoot as production code **Databases**: both are PostGreSQL, but there's nothing to do there – Spring will handle all of that for us.

Message Broker: is a RabbitMQ server, but there's little-to-nothing to do there – we'll configure it using Spring.

Infrastructure: to avoid worrying about installations, the base code uses Docker and Docker-Compose to spin a container for each database, and for the RabbitMQ server. This time, its usage **is mandatory**.

Delivery	• Put your code in a zip file with everything inside • In the root folder of the zip file add a file called "readme.txt" saying how much you did (if the task is incomplete, you must state it) The regular assignment worth 3 lab points.

Rules	**You cannot do:** Copy any part of the code created by another student; Look at the code of another student before finishing; generate code using AI or LLM tools. **You can do:** Ask help for debugging; help another student to debug after finishing your own. Use AI tools to assist in bug fixing.