Tampereen yliopisto

COMP.CS.510
Web Development 2 – Architecting

Group work

University lecturer
Antti Sand, Ph. D.

Content

- Group forming
- Assignment overview
  - Backend
  - Frontend
  - Optional
- Requirements
  - Summary for backend and frontend
  - Other requirements
- Grading

Group forming

- You should be in groups by 10th March
- If you wish to do the assignment solo or in a pair, that is also possible, but we recommend sticking to the recommended group size

- The group assignment is mandatory

Overview

- A simple application for ordering a sandwich
- Very simple functionality, the focus is on distributed components and communication
- Front end – back end
  - Frontend using, for example, Vue.js, React.js, others also possible
  - Backend distributed to servers A & B
  - Server A and Server B using Node.js
- Asynchronous communication using a message broker
- Docker containerization

Overview – Make a sandwich

- User chooses the sandwich they want to order from the frontend
  - You can make a predefined selection of these, better solution is to enable custom orders
- Frontend sends a HTTP message containing the sandwich order to server A
- Server A saves the order
  - Preferably into a database (e.g., SQL, mongoDB)
- Server A requests server B to handle the order through the message broker
- Server B receives the message, waits for a few seconds and then sends a message back to server A through the message broker, informing that the order is now ready
- Server A updates the order data for itself
- Frontend is updated (up to you how, just not by refreshing the entire page) to show that the sandwich order is now ready (e.g., websockets, polling, …)

- Be sure to check the full documentation for all the little details!

Backend – Server A

- Functionality: Handle messages from frontend, access data storage with sandwich orders, send orders to server B through the message broker
- Contains information of the available sandwiches the user can order
- Implements the Swagger API, that has been provided for you
- You will need to:
  - Dockerize the server A correctly
  - Fill in the missing functionality for the HTTP methods
  - Choose and implement a database of your choosing to save the sandwich orders
    - Fill it with premade sandwich types

Backend – Server B and RabbitMQ

- Server B is pretty simple compared with server A. Just listen to the message broker for new sandwich orders, and once received, wait a few seconds and sent them back as ‘ready’
  - Just modify index.js appropriately
- RabbitMQ will have two message queues: one for Server A – Server B, and one vice versa
- You will need to:
  - Dockerize Server B and RabbitMQ correctly
    - You’ll want RabbitMQ to start *before* the other backend components. Make sure to add a delay and/or exception handling somewhere to ensure that
  - Fill in the missing functionality for Server B
  - Connect Server A and Server B to the message broker

Frontend

- We recommend creating it with React or Vue, but it is up to you if you want to use something else. As long as it works.
- Must contain (at bare minimum) functionality to send new sandwich orders, check all the previous orders and look up an order with a specific order ID
- Good-looking UI is a plus, but not mandatory for this course
- You will need to:
  - Create the UI
  - Connect the frontend components to the Server A’s API
  - For better solution, allow for custom orders

Optional

- If you check the Swagger API for Server A, you'll notice there are some optional methods left to implement, namely for user login and creating new types of sandwiches
  - These are optional, but if you desire a high grade, implementing either of them will be viewed favorably in the grading
- You may choose to use some other technology than React or Vue to create your frontend
  - We might not be able to offer any advice to any problems you might encounter

Requirements summary

- Backend is dockerized correctly and executing a single, documented command-line starts the entire backend without error
  - Do not use any scripts on the host machine to run the backend, i.e., have all the dependencies inside the container(s)
  - Frontend can also be dockerized, but can also be started separately, as long as you have documented how to do it in few, simple steps
- Frontend has the required functionality to send and view sandwich orders
- Backend handles the orders correctly, communication between the servers is done through the message broker
- Any additional functionality you decide to implement will be viewed favorably in the grading

Mid-project check

- 7.4.2024 will be a mid-project check
  - Aim is to check that every group got started OK
- What is checked:
  - You have used Git's issues to partition the group assignment to reasonable chunks, and distributed them between the group members
  - There should be at least some issues and division of work shown
- Completing this awards the group 2 points

Other requirements

- Your final version is in your Git repository's master or main branch as the latest commit
- You have documented your work and made a pdf file to the root folder in your Git
- You have used Git's issues to partition the group assignment to reasonable chunks, and distributed them between the group members
  - These will be reviewed during the mid-project check-up, so start early and keep the issues up-to-date!

To wrap this up

- Read the assignment document in the Moodle
- Install the needed tools
  - Text editor or IDE (e.g., Visual Studio Code)
  - Git
  - Docker, Docker Compose
- If you need help, ask in the Moodle forum, so others with similar issues can also get help
- Remember, the course TA’s are available for support!