Software Engineering 2021/2022

Assignment 6 – Modularity using APIs

Teacher: Tom Pepels – tom.pepels@maastrichtuniversity.nl
Lab date: **9 May 2022** - Due date: **30 May 2022**

This assignment awards **30** points toward the total 100 assignment points.

Important: *For this course, you work on all assignments in* ***teams*** *of* ***two****. Before submitting the assignment, you must join one of the* ***prepared groups*** *in Canvas (click on* ***People*** *in the course menu).*

Do not make your own group in Canvas, nor submit the assignment without joining a group first!

In this assignment, you will write an **API** with your fellow students. The goal is to get an understanding of **RESTful API's** but also to get a sense of the difficulties you encounter when releasing an API into production.

You will probably encounter many obstacles getting your server to work with other people's clients and your client to work with other peoples' servers. Do not worry, relax, this is a *feature* of the assignment and a learning experience. Do not think that you must write the perfect API today rather than learning as much as you can from trying to go from **requirements** to **specification** to **implementation**.

The planning for this assignment is as follows:

1. Read this assignment thoroughly.
2. Choose which language you would like to use to implement the API. I have posted a video tutorial and written tutorials explaining how to implement REST API's in **Java**, **Python** and **Node.js** (**JavaScript**)
3. You and your partner implement a server that can use the API according to the requirements listed below, using the **endpoints** provided
4. Test your server application against **the provided API client written in Java.**
5. Make a ***screen recording*** where you show the following:
    a. Your **source files**,
    b. You **start** the API server from your IDE,
    c. Show that the client connects to the server and runs **all commands** listed in the requirements.

    There are several free tools available that you can use to record your screen.
6. You may speak when recording to explain what we are seeing, if you do not want to talk during the recording make sure we can see what you intend to show. For instance, by miming, using sign language, Morse code, medieval alliterate poetry, smoke signals, or carefully choreographed interpretive dance routine. (I prefer poetry and dance)
7. Hand in ***your group's*** **source code** and **your screen recording** on canvas. We will use the screen recording for grading the assignment. The source code is checked for plagiarism.

Submission Guidelines

Submit deliverables as a zip wile with filename: SE_ASS6_***GroupN_IDStudent1_IDStudent2***.zip substituting ***GroupN*** with your group number, and ***IDStudent1***, and ***IDStudent2*** with your student id's.

Make sure that you also write/mention your:

- group name,
- student names,
- student Id's,

in **all submitted** files (write a comment at the top of all source code files)! Failure to do this may result in ***NG*** (no grade) for the assignment.

Grading

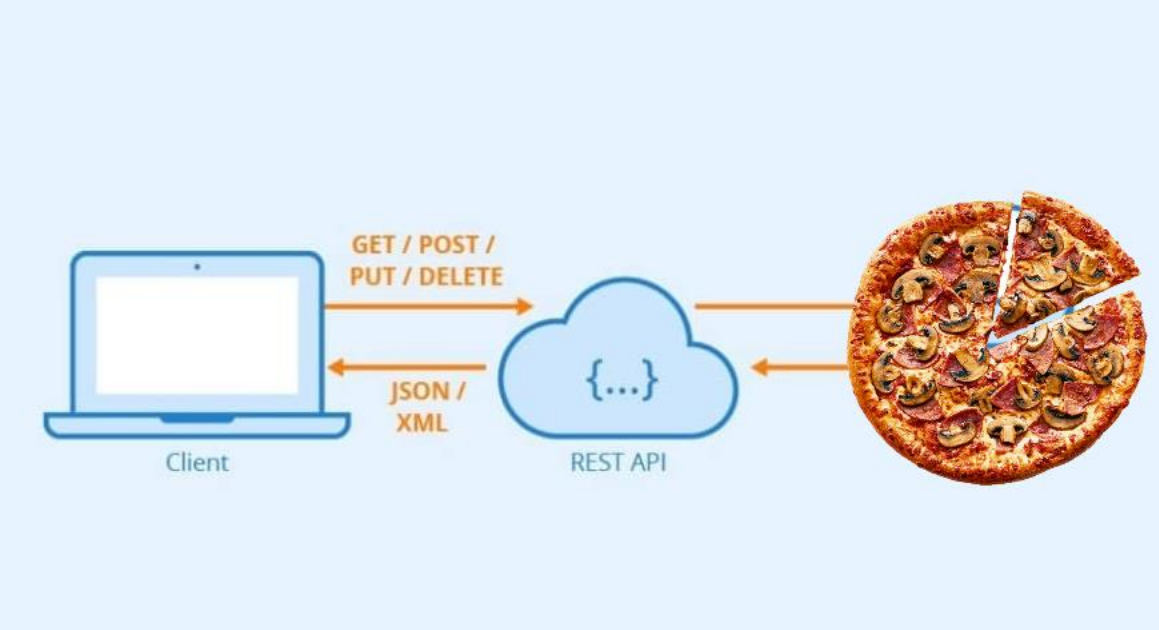
A grading form with feedback is included with the assignment.

The Pizza Conundrum

We have entered a post-Uber world. European politicians have decided that centralized food delivery networks are a main cause for the population's obesity. Therefore, neither Thuisbezorgd, Uber nor Deliveroo can operate under EU law anymore.

You have two options, leave the EU, or write a decentralized API for ordering Pizza's. Being the lazy programmers that you are, you of course opt for the latter. The idea is that to avoid legislation, each Pizza restaurant needs its own API.

This means that every Pizza restaurant in Europe will have to operate its own Pizza ordering service. Pizza restaurants are notoriously good in software architecture design, as we all know. Therefore, they decide that they want to separate their models and controllers from their views. After a year of deliberation (they all speak different languages and no English and still had to make Pizza's for their customers in between meetings), this is the architecture that they came up with the following design:



They are all very proud of this and would like you to implement their controller and API for them.

The end goal is that any application can connect with any Pizza restaurant's API to list and order pizzas. Therefore, your code should be able to provide this service to websites, mobile apps and desktop applications.

Software Requirements

You are tasked to write a REST service that can provide the following services:

- List all the pizzas available for ordering at the restaurant.
- For a given pizza return the toppings, price and whether it is vegetarian or not.
- Allow the placement of orders of pizzas.
    - When a client orders a pizza, he/she should be sent a confirmation with an order-id the pizzas ordered (there can be more than 1) and the estimated delivery time.
- Given an order-id, a client should be able to request the current estimated delivery time, and the status of the order (cancelled, in process, out for delivery).
- A client should be able to cancel their order if it was placed no more than 5 minutes ago. The order status should in this case be set to *cancel*.

Given that, there is no need to write a ***model*** (they asked another company to do that for them). You may hard-code your pizzas with ingredients and prices or store them in a text file. The same goes for orders, you may send fake, static information to the client. If you want to do more, you may also store pizzas and orders in a database or csv file. However, this is not required (in this course ;).

REST API Endpoints

Of course, for all restaurants to be able to use the same REST API, we need to define our REST endpoints and their methods. You can find all endpoints required for this exercise attached to the assignment in Canvas.

Make sure to implement all endpoints as presented in the pizza_api_endpoints pdf file. If your endpoints or replies do not match, the client will not be able to connect.