Hello Alexander.

In this document I intend to explain to you the overall solution I implemented to solve the task so far.

I started off defining all the clases, jpa repositories, services and controllers. Then I started to try to figure out how to make all the calculations involved in the process, including:

* How much time each order would take to be delivered
* How long would workers and cooks be busy
* How and when would I add or remove items availability, etc etc

The main logic is inside the file class “businessLogic”. I’ve been watching tutorials of the MVC Framework but I still don’t have certainty of where to put such a file, so I placed it inside the services folder.

What I did was to create two functions that estimate how much time it would take for both workers and cooks to look for all the ingredients missing and cook the dishes, respectively.

Then I also created a couple of functions, very similar to the estimation functions, that schedule the ingredients to be looked for for each worker and the recipes to be cooked for each cook.

I also played a bit with threads execution, to set an order from done=false to done= true or for increasing the number of ingredients once a worker has delivered the ingredient in the given time.

I still have several things to fix/test, such as the CRUD for every class and some other tasks that I have listed in a TODO file.

This is the first MVP of it and I intend to work on the front end in the next couple weeks.

You can take a look at what’s loaded on startup to the database in the file:

curryWurst\src\main\resources\data.sql

The list of things yet to be done is in:

curryWurst\src\main\resources\TODO.txt

Most likely the list will grow once I start testing every endpoint.

I also leave you here the correct format for sending a create order request (I omitted a couple fields)

{

  "done": **true**,

  "expectedDeliveryTime": "15:40",

  "id": 0,

  "orderItems": {

    "Nudel-Wurst-Käse-Salat": 1,

    "Kraut-Wurst-Pfanne": 2,

    "cocacola": 2

  },

  "price": 0

}

It is also listed in the swagger interface at:

<http://localhost:8080/swagger-ui/>

The database endpoint is:

<http://localhost:8080/h2/>

Kind regards.

Carlos.