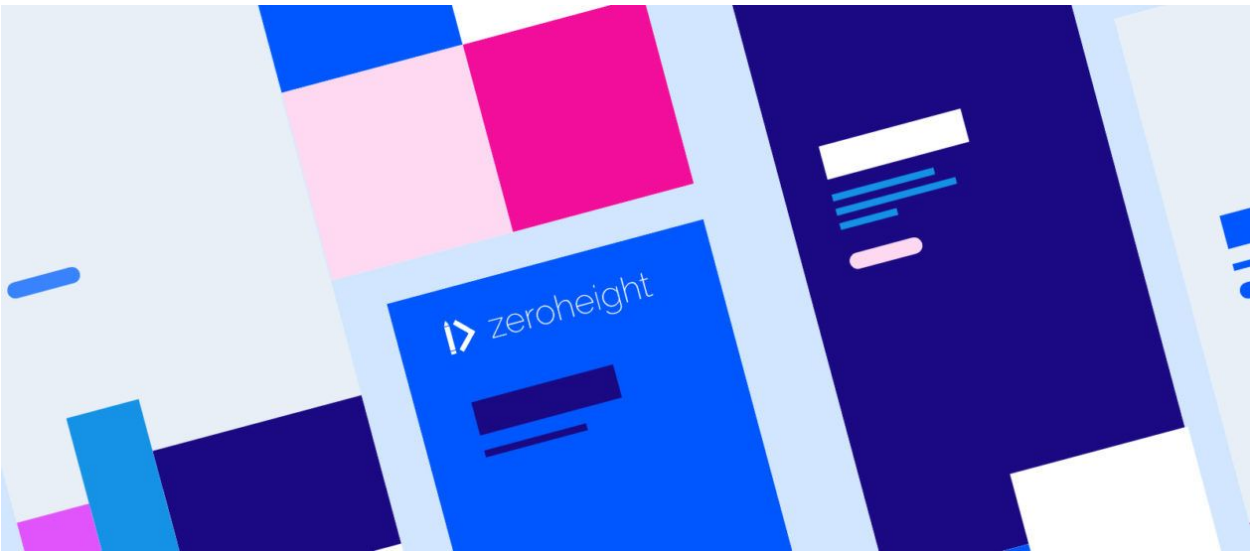
Project EWA System Documentation



Naam: Sam Toxopeus & Jerôme Tesselaar

Team: spaghetteria-4

Date: 4-1-2021

Product owner: M. Groen

Coach: M. Fuckner

1) Introduction **3**

2) Product vision **4**

3) Epic stories **6**

4) Layered Architecture Package Diagram **9**

5) Navigable Class Diagram **11**

6) Deployment Diagram **12**

7) Alternative options **13**

8) Analytical reflection **14**

1) Introduction

This document contains information about the TO-DO website developed for a chain of pasta bars named Spaghetteria. We developed this website with the Typescript Framework Angular for the front-end and the Java Framework Spring Boot to make the backend. This document will contain our interpretation of the vision that we had while working on the project, and also an explanation of the features and what direction we would like to see the project go in next.

Also the most epic user stories that we have developed will be explained in this document, defined by the acceptance criteria and quality guidelines. And finally we will be adding different types of diagrams, a Package Diagram with an explanation of the choices that we made, and challenges we beat along the way.

And a Class diagram that will show an overview of the classes of our project, one that shows the structure and dependencies and another that shows the use of the external interface. All will be clarified with relevant text.

The document also contains the design choices we made along the way, not the way the website was designed but more themes along the way of security, performance, storage etc. And will also contain a justification why we made the choices that we made.

The last diagram that this document will contain is a Deployment Diagram, and will contain the choices that we made about the physical deployment of the application that we made. Also with a justification of our choices.

And finally we write an analytical reflection of the design we made, and it will conclude with the way we see that further improvement can be made to the program.

2) Product vision

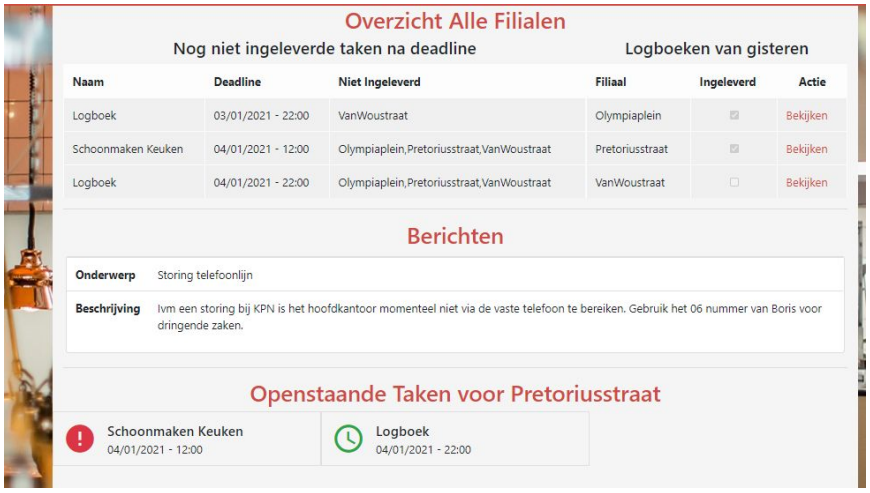
The product that we've made is a website that in core is a TO-DO list application for the restaurant that we designed it for. It is a place where tasks can be made by the managers of the restaurant branches, or important contacts can be seen in an overview. Top management can also post messages with important information, and retains insight if the messages and tasks have been read and if the proper actions are taken.

The tasks are the most important part of this website, and at the homepage you can see an overview of all the tasks that need to be done. The tasks come with different types of icons, so that it's easy to see if a task is overdue, nearly at its deadline or is a recurring or recently made task.

Example of the tasks: red means that the task is past it's deadline.
The green icon means that the task is recurring.

In addition to the tasks we have a log module feature, where the branch manager can write all the important data of that day. Examples of that are the amount of deliveries that are done that day, or the daily revenue. This is something that the branch managers will be doing daily, and gives the top manager a good idea about the going on of the different restaurants.

In addition to that the home screen of the administrator, also shows the data of all the tasks that have been over the deadline and the log modules that have been written the previous day. We made sure to show all the important information at the first screen that they see after logging in:



The home screen for the administrator shows all the important information: Tasks that are over the deadline, log modules written the day before, Important messages and tasks that are still not finished.

These are just a couple of examples of our most important features we have on the website, that we've made for our target audience. That target audience not only contains the top managers, and branch managers of the restaurant. But also the employees that need to fulfill those tasks, and make sure that the correct photos or messages will be uploaded to finish the task correctly.

The way we see the direction to take for a bigger growth in the future is that more branches will be added to the website, since this communication system is only made for one company in mind. So as long as the company grows and adds more branches to their collective, our website can also expand by adding more branches with additional features to the website.

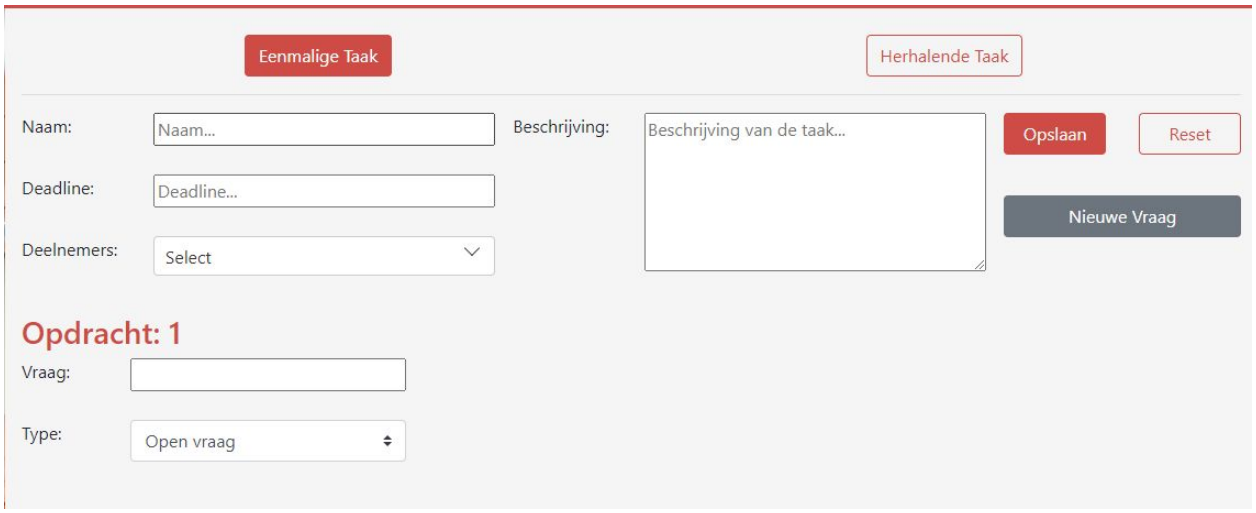
3) Epic stories

One of the most important epic stories that has been realized is the login system, because it was very important top managers could login there as an administrator. And the different branches of the restaurant each had their own individual login accounts, all with their own data (tasks and important contacts for example).

The most important acceptance criteria for this story was the security aspect, we made sure that all the logins made by either admins or branch managers all had their own token to identify themselves with. The tokens need to be inserted in the Authentication portion of the header, so the login attempt can be properly identified.

This is all according to the Security norm of the ISO 25010 where the Authenticity part says that the source of the login needs to be verified, with a security like signature (in our case this is the token). The token also has an expiration time of 12 hours, and once those 12 hours are over a new login token needs to be collected from the server or the user can't login.

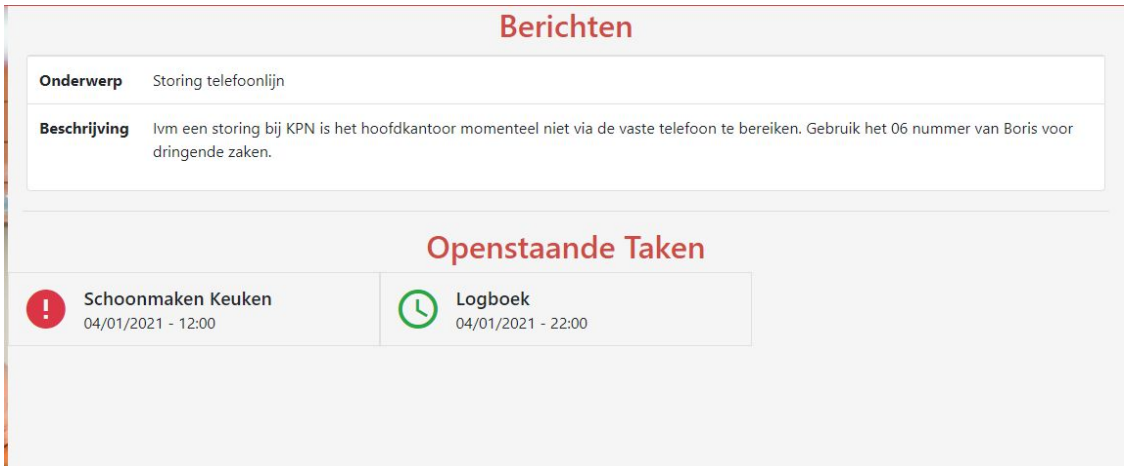
The second important user story is the tasks, the most important acceptance criteria was that only tasks where shown that belonged to the specific branch (chosen by the administrator or according to the login name). We also wanted the tasks to have a big amount of flexibility, with the amount of questions that could be asked or the amount of dropdown menus with different types of choices in it.



Example of the adjustability of creating a task.

We lay the focus on the Usability side of the ISO 25010 system, mainly the operability of the system. We lay the focus on making the tasks really easy to adjust and make it easy to add different types of questions or features. This also adds to the Maintainability chapter of the ISO 25010 system. Mainly the modifiability, because of the ease that tasks can be made or changed it's less likely that a user makes a mistake.

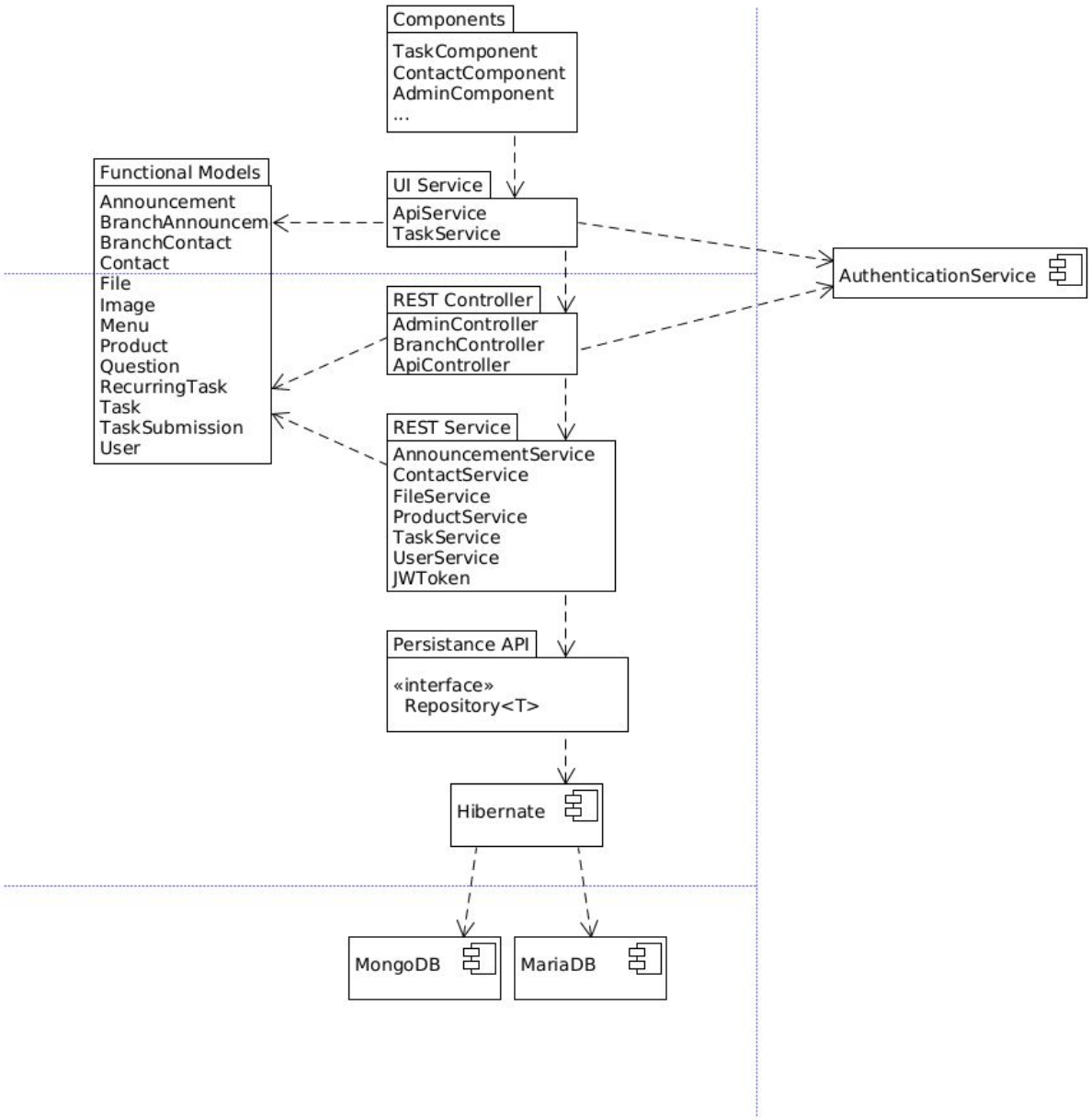
Also the icons that were added makes it easy to see when a task is done, or a task is recurring. This all comes together in the home screen of our application, something we added in the later stage of the development. It's a nice overview of all the important features that we added to the website, and is different depending on the person that's logged in. The addition that an administrator get's are the most important tasks, messages and log modules of ALL the branches. While the branch that's logged in only gets an overview of the tasks and announcements that are important for them.



Home screen when a branch is logged in.

Home screen when the administrator is logged in.

4) Layered Architecture Package Diagram

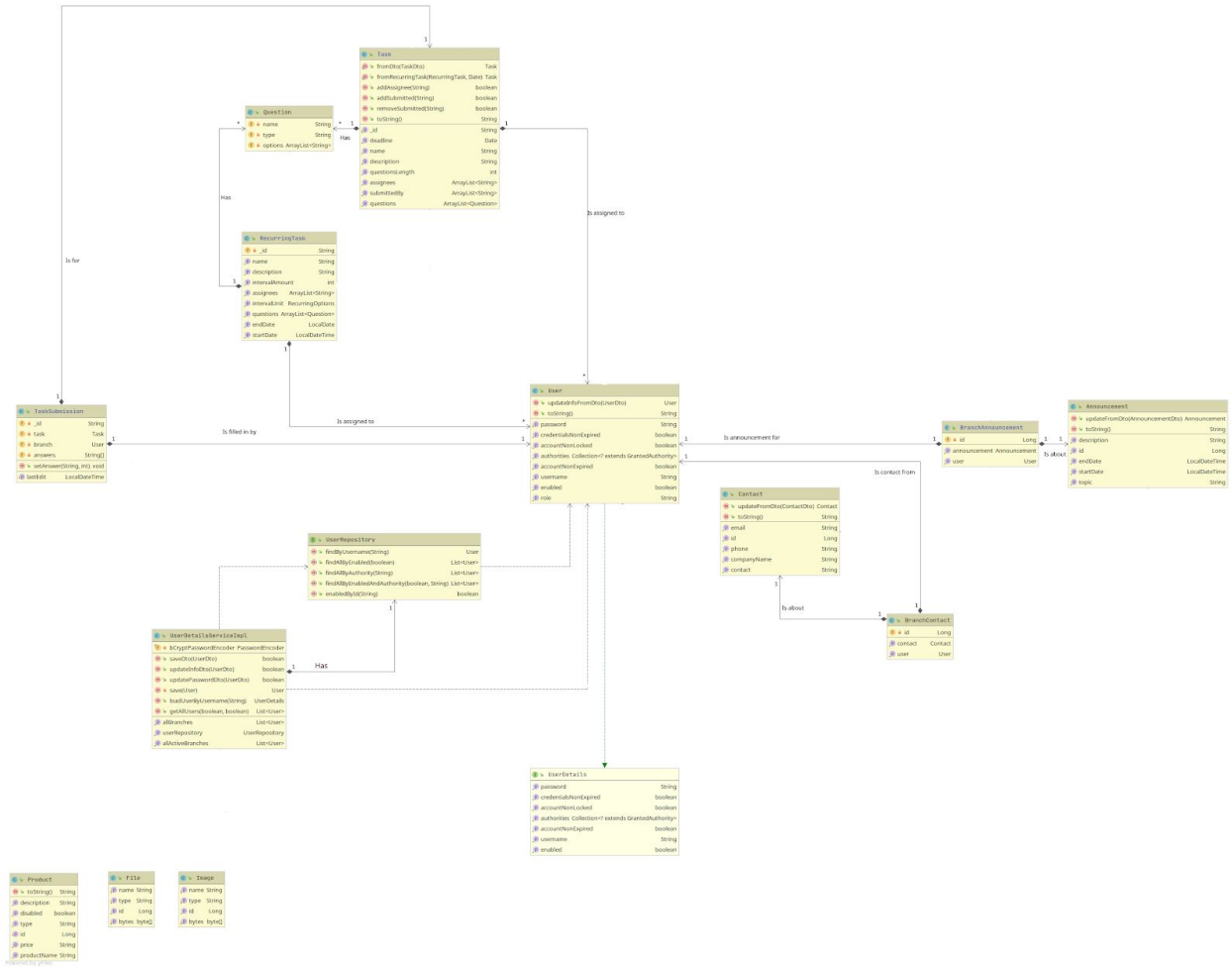


We only have 2 services in our frontend, this is because we thought we needed to keep all calls to our API in the same service. We later realised this created clutter and started working with a separate service for some of the task features.

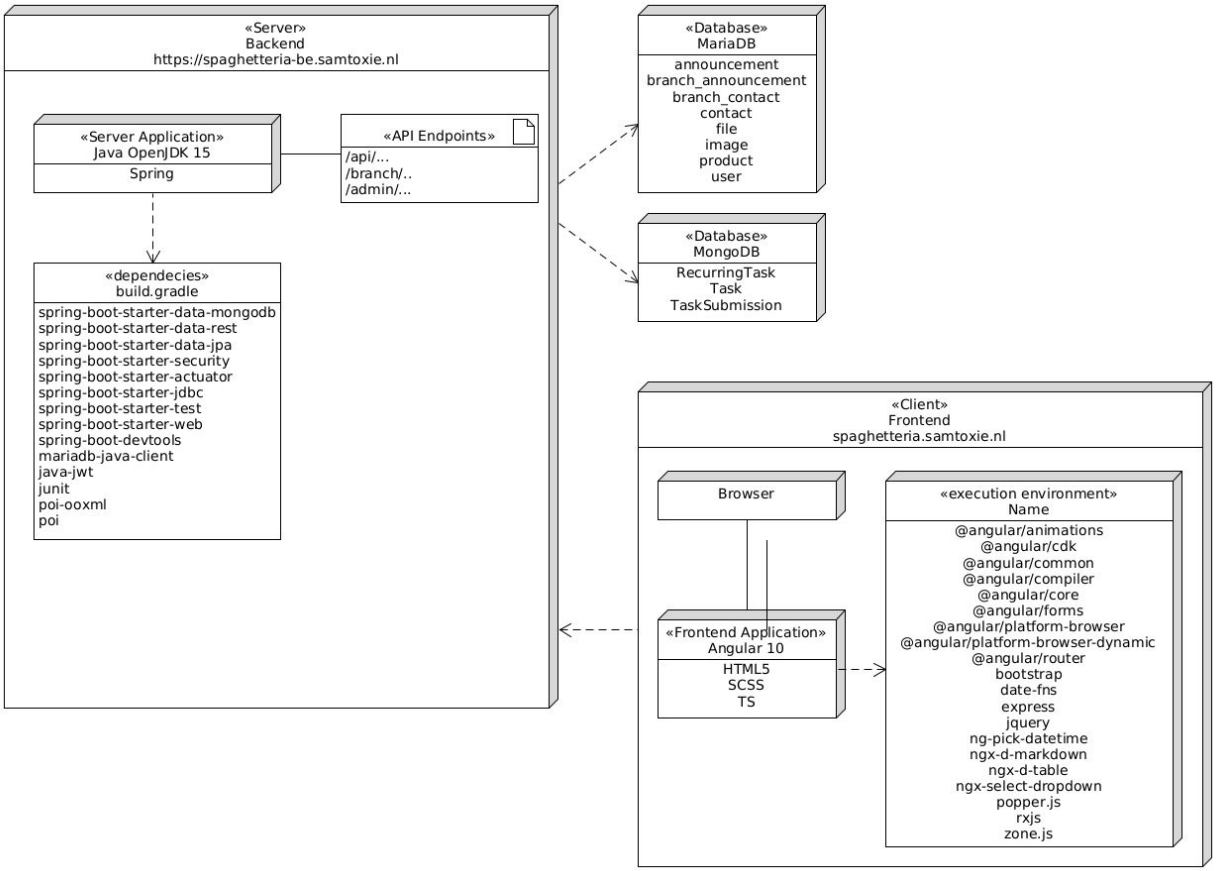
In our backend we have three controllers, ApiController requires no authentication, BranchController requires an account with the USER role or higher and lastly AdminController requires an account with the ADMIN role.

We use a combination of MongoDB and MariaDB storage. For the tasks we wanted flexibility in how we store the data, we realised we wouldn’t achieve this in MariaDB with a heavy boilerplate.

5) Navigable Class Diagram



6) Deployment Diagram



7) Alternative options

For our solution we used a combination of MariaDB and MongoDB, as this is what we had previous experience with. We went with this choice as we needed ordered arrays to be stored, and deemed using several tables to achieve this would become too complicated to expand and maintain.

This is not ideal and could be replaced by a solution using PostgreSQL, or by storing the arrays as binary objects in MariaDB. We didn’t know this in advance, hence our choice. Moving to PostgreSQL would be the preferred choice, as it allows the data to be stored in the same database without needing to save certain parts as a binary object. This removes several obstacles, like keeping data in sync.

As for our authentication system, we currently use a JWT implementation, this is then stored in the SessionStorage. Due to this choice, opening a new tab requires you to start a new session, and to solve this an implementation using LocalStorage could be used.

8) Analytical reflection

The website we made has a lot of different types of features, but there is always room to add more features and improve upon them. And there were tasks that we left open because we didn't have the resources to finish them. The calendar with an overview of tasks, still needs to get an overview based on month. And a yearly overview of the tasks for the branches will also be a nice addition.

We also had plans to implement a language feature, where the language could be switched from Dutch to English accordingly. This would also be a great feature to implement, because the pasta bar also has employees from different types of countries. It would also be nice to have an addition to this language setting by adding a way to switch the language to Italian. Some of the cooks of the restaurant don't understand English, so a way to make them understand the website would make a great addition.

Also an overview of the suppliers with additional orders, was something that we wanted to see implement but didn't have the appropriate means for. So that could be something useful to improve the website with.

We also had ideas where there would be a live chat system implemented in the website, where branch managers could talk to top managers in real time. And in addition to that would be notified when they had a message waiting for them on login. This is something that would make the communication between the different branches and top managers even smoother, than the announcement service that we have already.

As briefly mentioned a notification system is something that would make a really nice addition to the website, where the administrator or top manager will be notified in real time when a task is over the deadline. Or when an important message is sent that needs to be read with haste. This would also improve the communication. Not only between the users and the application itself, but will also make the communication between the branches a bit quicker.

As for our technical design, we use two separate databases, looking back it would have been better to use another solution like we previously discussed.

Also our user implementation is a bit flawed as it uses the username as ID, which leads to some complications for example when wanting to edit this username. This becomes even more flawed when using our solution with two databases. To prevent issues due to this we don't allow users to be renamed.