Sprint Retrospective, Iteration #5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Story # | Task # | Task Assigned To | Estimated Effort per Task  *(in hours)* | Actual Effort per Task *(in hours)* | Done  *(yes / no)* | Notes |
| *User creates an account. Then he creates a house so his housemates can join.* | Add update methods for the User, House, Request controllers | Ina | 1.5 | 1.5 | Yes |  |
| Test the update methods for the User, House, Request controllers | Ina | 2 | 2 | Yes | Manual testing - completed  Automatically tested - completed  Covered all the possible branches of the update methods: ok, not\_found and internal\_server\_error |
| Test all the Request microservice endpoints manually | Fabian  Ina | 2 | 3 | Yes |  |
| Add method for a user to create a House | Fabian | 0.5 | 0.5 | Yes |  |
| Add method for a user to leave a House | Ina | 1 | 1 | Yes | When there is only one user left in the household, and the user wants to leave as well, the house will be deleted. |
| Add method to get a list of users (their usernames) from a household | Fabian | 1 | 1 | Yes |  |
| Add method for a user to join a household once the members accepted their request | Ina | 1 | 1 | Yes |  |
| Add method for members of a household to accept requests of a user to enter their household | Ina | 2 | 2 | Yes |  |
| Manually and automatically test the userJoiningHouse, user LeavingHouse, membersAcceptingRequest | Ina | 2 | 3 | Yes | For the JUnit tests, all the possible branches were covered |
| Fixed recursion and add/update methods in the Request microservice | Ina | 5 | 10 | Yes | Had this problem for the whole last week. Tried many options, multiple times. In the end, I deleted the tables in the database and started again -> eventually worked |
| Try to fix recursive serialization | Oskar | 3 | 4 | Yes  (will not be merged) | Ina’s solution was considered superior to mine, thus my branch will not be merged |
| Add more tests for the entities to achieve 100% line coverage | Ina | 1 | 1 | Yes | Successfully achieved 100% line coverage for each entity of the Request Microservice |
| Add more tests for the Request Controller to achieve 100% line coverage | Ina | 1 | 2 | Yes | Successfully achieved 100% line coverage for the Request Controller of the Request Microservice |
| Add more tests for the House Controller to achieve 100% line coverage | Not known yet | 1 | - | No | Not yet at 100%  The methods that still need work on the tests:  - splitCreditsWhenExpired  - getUsernamesByHouse  - deleteHouse |
| Add more tests for the User Controller to achieve 100% line coverage | Not known yet | 1 | - | No | Not yet at 100%  The methods that still need work on the tests:  - editUserCredits  - splitUserCredits |
| Create tests for TransactionsSplitCredits entity | Fabian | 1 | 1 | Yes |  |
| Change return types of the methods in controllers to Response Entity | Atanas  Fabian | 5 | 5 | Yes | It required quite some testing and refactorization in tests. |
| Fix functionality of some controllers in Requests Microservice | Fabian | 1 | 1 | Yes |  |
| Create tests for transaction controller | Oskar | 3 | 5 | Yes | Due to the change of return values by methods tests had to be done again thus it made the process longer than anticipated |
| *User adds products to the fridge. Then, after there are products in the fridge, users are able to make use of products, by adding transactions in order to keep track of how the products are used* | Update Product controller with new methods in order to cover all the required functionalities | Kendra | 1.5 | 2 | Yes | I have added a couple of basic methods which felt necessary for a better implementation of the product controller. However there still is much improvement needed on them. |
| Migrating the database schema to DigitalOcean, because the TU Delft database is unreliable | Atanas, Oskar, Fabian | 1.5 | 3+ | No | We had a lot of problems with the auto-creation of the database schema in MySQL 8. In the end it did work for the transactions microservice, but we decided we do not have enough time to change the databases.  We will continue only if we have more problems with the TU Delft database. |
| Added a @Username annotation that returns the username of the user making a request for the transactions microservice. | Atanas | 1 | 2 | Yes |  |
| Connect the authentication and requests microservices | Atanas | 1 | 2 | Yes | Connected the authentication and requests microservices to add a new user to the databases of both microservices when a user registers. |
| Create more tests for the Product controller to reflect the changes | Kendra | 1 | 1.5 | Yes | I have created a lot of tests for the product controller in order to achieve a high test coverage as it was really low. We have also tried doing mock tests as they seemed more efficient to make. |
|  | Extract all common code to a separate package to avoid some code duplication | Atanas | 1 | 1.5+ | No | Decided it is not worth it, after the first lecture of the week, so this task was abandoned. |
| Changed the datasource config for the TUDelft MySQL database | Atanas | 0.5 | 1 | Not merged yet | Restricted how many connections to the database the datasource is allowed to open. Also updated the connection timeout parameters and the maximum pool size.  Couldn’t reliably test it, but can confirm that there are less open connections on the database by checking the running processes on the database. |
| Make all microservices pass PMD. | Oskar | 3 | 5 | Yes | A lot of classes were annotated to skip pmd check. This made us not spotting a lot of bugs that I fixed. There are still one or two classes that has to be checked |
|  | Manual test controllers | Oskar, Fabian | 3 | 3 | Yes |  |
|  | Make presentation | Kendra, Atanas, Fabian, Oskar, Ina, Stoyan | 12 | 12 | Yes | During the last week we had to make a presentation about the process, where we all filmed videos individually. After that we merged all of them into our final video which was submitted to the Brightspace. |
| A user will automatically get his credits increased when he adds a product to the fridge. Also the credits of the user will be changed when he eats a product and the amount will be subtracted from his balance. | Create communication between the microservices: Transactions and requests | Stoyan | 2 | 3 | Yes | In order to change the credits of a user we have to establish a communication between the microservices Transactions and Requests, where the credits of a particular user will be changed. This required a bit of research and working as a group in order to make it work. |
| Create method for automatically adding credits to a user when he buys a product and adds it to the fridge | Stoyan | 2 | 2+ | Yes | A function to change the credits of a user when he adds a product. It gets the price of the product, converts it to credits, sends it to the request microservice and adds the credits to the user. |
| Added an option to the method for adding credits, to also delete credits. The method is renamed to changing credits and now it is also able to subtract credits from a user | Stoyan | 1 | 1 | Yes | Now we can add and subtract credits when a user gets/adds a product. The credits of the user will be changed automatically by sending the credits to the request microservices. |
|  | Added method for splitting credits when eating together | Stoyan | 2 | 2 | Yes | Added a method that ensures that the credits will be split among the users when they are eating together. |
|  | Creating a method to get the products by house number | Stoyan | 3 | 3 | Yes | This method ensures that we have a separation of the fridges from different houses. It is useful to just check all of the products in the fridge in your house. |
|  | Added method to set product as expired. This way the price of the left portions of this product will be divided and subtracted evenly from each user of the household | Stoyan | 3 | 3 | Yes | It was hard to implement this method. It again includes communication between the microservices. When a product is set to expired, its price for the left portions is divided to the number of users and automatically subtracted from each user of the household the fridge is. |
|  | Edit videos + merge  (presentation) | Oskar | 2 | 2 | Yes |  |
|  | Add documents to the repository in order to have everything more centralized | Kendra | 0.5 | 0.5 | Yes | We needed to add all the sprint plans and meeting documents in the repository in order to make sure we have all our processes properly documented. |
|  | Beautify code | Oskar | 0.5 | 0.5 | Yes |  |

Project: Software Engineering Methods - Student House Food Management

Group: 51

Main Problems Encountered

**Process related problems**

**Problem 1**

Description: We had a lot of time management issues. We did not properly judge how much time we would need to implement the full functionality, and did not test our microservices in time. Also, connecting the microservices proved to be a lot more difficult than expected.

Reaction: We split the remaining work, and focused on only the most pressing issues. We decided that we will not have time to implement all features in time, but we focused on this project and spent a lot of time trying and improving.

**Problem 2**

Description: Time management issue. Up to week five we mostly worked from Sundays to Wednesdays (this can be seen in GitLab analysis). Working just before the meetings made the whole process more stressful

Reaction: During last week, as we realised that it is the problem we decided to spread the work more evenly throughout the week. This made the work less stressful and more effective.

**Code related problems**

**Problem 1**

Description: We have a problem with database connection. Currently we have 3 possible options:

\*locally hosted mySQL database

\*Digital Ocean Cloud database (MySQL 8)

\*TU Delft hosted database (MySQL 5)

We have problems when trying to use DOC solution, since the JDBC controller cannot set up the schema of the database. Thus we decide to use TU Delft solution right now, but this solution is not perfect, since we often get “too many connections error”. This solution is not permanent.

We are working on solving the issue with DOC and we hope to deliver it until the deadline.

//Update 18.12.20

Atanas managed to set up the Digital Ocean Cloud so we stick with it because of its consistency.

**Problem 2**

Description: The connection between microservices is not yet manually tested. We should focus on going through the code and checking whether everything is working ok.

**Problem 3**

We have a lot of code duplication across the microservices (mainly in the configuration classes) that can be resolved by creating a common dependency that all microservices use.

However, I had some problems with the configuration, so I decided we can do that later if we have time.

**Problem 4**

Description: We have unfortunately been busy with making everything work properly and haven’t got the chance to implement a reset functionality for our application.

Reaction: We will try doing that in the days to come in order to implement this option on time, before the deadline.

**Problem 5**

Description: We had some problems with the communication of the microservices and with some basic functionalities, where we have to fix the issues. We have tried different approaches to fix the functionalities of some methods and to send information from the Transaction microservice to the Request microservice.

Adjustments for the next Sprint Plan

*Finish all the requirements by Friday and make sure they are fully functional, especially in terms of communication between the microservices.*

*Solve problems connected with Digital Ocean Cloud database connection*

*Make more tests for the controllers*