Sprint Retrospective, Iteration #4

Project: Corona-proof room scheduling
Group: OP29-SEM57

User story #	Task #	Assigned to	Estimated effort	Real effort	Done	Notes
Luca Becheanu	Testing identity service	Luca Becheanu	Medium	Large	yes	Added tests for validating the jwt token.
	Add jwt validation to course and calendar services	Luca Becheanu	Large	Large	yes	A service will send a request to the identity service and as response it will receive the role and netid extracted from the JWT token.
	Fix Post Methods	Luca Becheanu	Medium	Medium	yes	Some post methods had parameters sent through the url instead of being sent through the request body in JSON format.
	Fix failures in Course Service	Luca Becheanu	Large	Large	yes	More information about the errors can be found in problems encountered.
Matthijs de Goede	Testing Calendar Microservice	Can Parlar, Matthijs de Goede Alex Bobe	Large	Too large	Partially	The controllers of the calendar microservice appeared less testable than we thought. It was really hard to create tests for them and a huge number of mocks had to be configured.
	Creating Postman API requests and documentation	Matthijs de Goede	Medium	Medium	yes	I created a Postman project with requests to all our endpoints to check their responses and generated an interactive documentation page for them.

	Refactoring Course Management System	Merdan Durmus, Matthijs de Goede	Large	Large	yes	We refactored the Course management system so that the endpoints are in line with each other, documentation and tests have been added and authentication is in place.
Timen Zandbergen	Set up replacement databases	Timen Zandbergen	Medium	Medium	yes	The databases are now hosted on a VPS
	Fix issue with LocalDate on endpoints	Timen Zandbergen	Small	Small	yes	There is a bug within Spring or JPA
	Change every endpoint to take in and return JSON	Timen Zandbergen	Large	Large	yes	All post requests take in JSON, and all endpoints return JSON
Alexandru Bobe	Testing Calendar Microservice	Can Parlar, Matthijs de Goede Alex Bobe	Large	Too large	Partially	The controllers of the calendar microservice appeared less testable than we thought. It was really hard to create tests for them and a huge number of mocks had to be configured.
	Managed the sprint	Alexandru Bobe	Small	Small	yes	As the sprint master, I had to take care of everything during this sprint, from tasks to retrospective
	Fixed bugs in endpoint methods	Alexandru Bobe, Matthijs de Goede	Medium	Medium	Yes	Some endpoints were not working properly and we had to change some things in the code
Can Parlar	Testing Calendar Microservice	Can Parlar, Matthijs de	Large	Too large	Partially	The controllers of the calendar microservice appeared less testable than

		Goede Alex Bobe				we thought. It was really hard to create tests for them and a huge number of mocks had to be configured.
	Testing Restrictions Microservice	Can Parlar	Medium	Large	Partially	Figuring out how to mock the validation methods was a challenge, but the actual methods are not tested this way. But, I achieved 100% test coverage in the controller.
	Update Readme file	Can Parlar	Small	Small	Yes	I have updated the readme file to include essential information about our project. Running, testing, and database connection.
	Communication methods between services	Can Parlar	Medium	Medium	Yes	I have created communication methods between rooms to restrictions and restrictions to calendar service.
	General testing	Can Parlar	Small	Small	Yes	I have added small tests to applications and added lombok files to exclude generated methods from jacoco.
	Added validation to restriction methods	Can Parlar	Medium	Medium	Yes	I have added validation methods to restrictions service to allow teachers to change restrictions.
Merdan Durmus	Test Course Management Service	Merdan Durmus	Medium	Hard	Partially	We don't know how to mock the HTTP request and responses. We will look into this next sprint and see if we can change the client
	Fix schedule retrieval for teachers	Merdan Durmus and Luca Becheanu	Medium	Hard	Yes	The start was a challenge since we needed to verify the role of the user. This part was done by Luca

Main problems encountered

Problem 1

The bean 'dataSource', defined in BeanDefinition defined in class path resource [org/springframework/boot/autoconfigure/jdbc/DataSourceConfiguration$H, errors creating bean with name 'xRepository', errors no identifier specified for entity.

Reaction

There’s been a lot of fixing to do, and the solutions found online were not that helpful since the first error had to do with the gradle build, some queries were failing due to the wrong naming of the tables in the databases (all had to be lowercase) and other queries failed due to the naming of the methods in the repositories, meaning that a lot of manual debugging had to be made.

Problem 2

The dates and times were malformed when exported to the database. For some reason, every day became one later and every hour advanced by 60 minutes too. In the entire dataflow in the application itself, the dates and times were correct. It took some time to figure out that it is a database related issue.

Reaction

To account for this we change the dates and times just before we export them to the database.

Problem 3

We started too late with proper testing of the Calendar service, which results in low overall test coverage. We also couldn't really find a way to test the Communicator classes in an automated way.

Reaction

To compensate for the automated testing issues, we created a Postman project that includes ALL the endpoints and example queries so that you can verify that they work as expected. They serve as system tests now.

Problem 4

Mocking the jwt validation classes was very hard and it took a lot of time to figure out how to get it to work. This resulted in methods using jwt validation not being testable and a lot of last minute work to be done.

Solution

We persevered and collaborated until we found a solution. Then we made last minute changes to implement the found tactique in different parts of the system, so that previously untestable methods can still be tested.

Requirement analysis

Functional

All in all we believe we did a great job when it comes to the functional requirements. All of the must haves are there and we even managed to get the should haves fully implemented. However, we haven't been able to also integrate any of the could haves, but this shows that our prioritization is on point as we have all the must haves in place.

So to evaluate all functional requirements. In our current system:

- A user, which has a name and is either a student or a teacher is able to log in using NetID and password
- A teacher is able to request a lecture for course x of duration y in minutes to be scheduled on day z
- A lecture schedule can be created once every quarter by making a call to the Calendar Service, which makes sure to schedule all requests received up to that point and for dates in the future.

- Each lecture is scheduled in a particular room for a timeslot in such a way that rooms with the highest capacity shall be assigned to the courses with the most attending students, and smaller rooms to smaller lectures.
- Both students and teachers are able to retrieve their own schedule for a specific day, course or just the one with all of the upcoming lectures. Information about the timeslot, room and in the case of the student an indication of whether he or she is selected to attend the lecture on campus is displayed.
- Students can attend at least one lecture in person every 2 weeks.
- Restrictions about the allowed capacities for rooms, time gaps between lectures, starting times and ending times can easily be adjusted on the Restrictions Service.
- The above restrictions are taken into account when making the Corona Proof schedule.
- Students can indicate that they refrain from attending the class in person, and lecturers can request a list of all students that will attend for a specific lecture.
- A teacher can create new courses, with their name, course code and a list of attending students.

Non Functional

When it comes to the non-functional requirements,most of them are there, but we failed to reach at least 85% meaningful branch coverage. Our test coverage lies around 65%. This could indicate wrong prioritization of the number of features covered over properly tested features. However, we found a great alternative by performing system tests with Postman, which show that the endpoints are working as expected.

So to evaluate all non functional requirements. The current system:

- Is compatible with Windows, Linux and MAC
- Uses a REST api for interfacing
- Organizes individual components following the microservices paradigm
- Is implemented in Java 14
- Uses Spring security to handle security
- Has a meaningful branch test coverage of around 65% from jUnit tests and all the endpoint methods have been tested using Postman
- Is able to cope with at least 300 and at most 500-600 students per course
- Assume that the requirements given to the system are applicable

General Evaluation

At the end of the project we are proud we managed to develop an entire system in just a few weeks time. Even though the initial deadline was extended, that's still an achievement.

We enjoyed working in the group and the meetings were generally very productive as the strategy for the Sprint was determined and the work divided accordingly. However, additional meetings were hard to schedule, which made it hard to communicate the progress within the sprints and get everybody on the same page. Although we felt we spent a lot of time on getting the requirements done, specifying endpoint definitions and designing the interactions between microservices in the first two weeks and hence felt that we were lacking behind, this helped us a lot further in the project as it was really clear what was left to be done to match the requirements.

After a tip from the TA to make Issue based branches, we focussed a lot on using issues, tags and templates, which facilitated the tracking of the project. The TA complimented us for this having such a nice issue board. In terms of merge requests, they often were too big to work for efficient merging, which led to some hard to solve merge conflicts and made it hard to work on the basis of small issues.

There was also the issue of a difference in the number of commits per user during the project. The TA reminded us to get these numbers closer together and we really tried to do that. However, we believe that the number of commits, although indicating activity, is not necessarily a valid measure for the contribution of each group member. The main issue with this method is that people commit at varying rates and certain logical units of change can take significantly more effort than others.