Sprint Retrospective, Iteration #1

User story #	Task #	Assigned to	Estimated effort	Real effort	Done	Notes
Luca Becheanu	Create Database Schema	Luca Becheanu	Medium effort	Medium effort	Yes	Created main database table which was then divided into 5, having a schema for each service
	Create Eureka Registry for Microservice Discovery	Luca Becheanu	Small effort	Medium effort	Partially, only the identity service is currently registered	Created a base to connect micro services through one server
Matthijs de Goede	Domain Driven Design	Matthijs de Goede	Small effort	Small effort	Yes	Came up with a subdivision into microservices that we enhanced together
	Design of the scheduling algorithm	Alexandru Bobe and Matthijs de Goede	Medium effort	Medium effort	Yes	We wrote the pseudocode. However, we were uncertain about what percentage of course participants to aim for when allocating a room.
	Implementation of the scheduling algorithm	Alexandru Bobe and Matthijs de Goede	Large effort	Large effort	Partially	The implementation is heavily WIP. We were heavily dependent on other services so we started mocking their implementations
Shared	Requirement design	All	Large effort	Large effort	Yes	We followed the template provided in Brightspace to make a clear division into categories of importance. We later added a permission table and user stories.

	API Definition design	All	Medium effort	Medium effort	Subject to change	We came up with the function signatures of the API endpoints we need for every microservice in order to meet the requirements.
	Service interaction graph	All	Medium effort	Medium effort	Yes	We visualized the interactions in the form of directed requests between all microservices.
Timen Zandbergen	Create templates		Medium effort	Medium effort	Yes	Checkstyle still has a problem with the naming
Alexandru Bobe	Design of the scheduling algorithm	Alexandru Bobe and Matthijs de Goede	Medium effort	Medium effort	Yes	We wrote the pseudocode. However, we were uncertain about what percentage of course participants to aim for when allocating a room.
	Implementation of the scheduling algorithm	Alexandru Bobe and Matthijs de Goede	Large effort	Large effort	Partially	The implementation is heavily WIP. We were heavily dependent on other services so we started mocking their implementations
Can Parlar	Create Issue and Merge Request Template	Can Parlar	Small Effort	Small Effort	Yes	Created a detailed issue template, in order to have a common way of creating issues. Did the same for merge requests.
	Started working on the restrictions and rooms microservices	Can Parlar	Medium Effort	Medium Effort	Yes	Researched API endpoints and microservices. Connected the database to the program and started creating entities for the service.
Merdan Durmus	implementation of databases for each microservice	Merdan Durmus	Medium Effort	Medium Effort	Yes	There was a duplicate table which wwe decided to remove

	Implementation of course management microservice	Merdan Durmus	Medium Effort	Medium Effort	Partially	Created different methods for this microservice.

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

Main problems encountered

Problem 1 - Eureka complaining about timezones

Description

- When running the first discovered microservice (the rest are not yet connected), launching the application will yield an exception: Failed to obtain JDBC Connection; nested exception is java.sql.SQLException: The server time zone value 'CEST' is unrecognized or represents more than one time zone. You must configure either the server or JDBC driver (via the 'serverTimezone' configuration property) to use a more specific time zone value if you want to utilize time zone support.

Reaction

- Spent hours on stackoverflow and various other resources to find a fix, yet nothing came up that would solve the problem.

Problem 2 - Being dependent on other microservices

Description

- Alexandru and Matthijs noticed that they heavily relied upon the implementation of other microservices and that they had to make assumptions about the interface between the different microservices. This made it hard to start with the implementation of the scheduling algorithm.

Reaction

- Specific issues for Sprint 2 have been created to request a specific implementation or data influx from other micro services.
- Communicator classes that encapsulate the API calls to other microservices and return dummy values till they're implemented have been created.

Problem 3 - Checkstyle configuration error

Description

- When trying to set up the checkstyle plugin (Settings->Tools->Checkstyle->Add configuration file) we encountered multiple errors such as: com.puppycrawl.tools.checkstyle.api.CheckstyleException: cannot initialize module SuppressionFilter - Unable to parse (CheckstyleMain/CheckstyleTest do work when running gradle ChekstyleMain/ gradle checkstyleTest so it is not a big issue)
- We encountered a problem with the subproject naming. Intelij threw an error when using kebab-case in the package names, whereas Checkstyle threw an error with when using snake_case. We have to decide what to use still.

Reaction

- We asked our TA to take a look and maybe help with the issue.

Problem 4 - Deciding what rooms to consider to schedule a particular lecture

Description

- Alexandru and Matthijs weren't entirely sure on the exact scheduling goal of the scheduling algorithm. In the naïve implementation a room would be chosen that fits all the course participants of a particular course. However, this is infeasible in corona times as all rooms have a maximum capacity of 30%.

Reaction

- First an implementation based on a lower bound in the form of a percentage of the course participants that should be able to physically be present in a lecture was created. However, this complicates the scheduling as we would have to try the scheduling for different higher percentages before accepting the lower bound. The short term solution might be to first assign the biggest room to the biggest course, taking the risk that we cannot guarantee the the once per two week requirement.

Adjustments for the next sprint plan

- Create gitlab issues for everything you do, at the start of the sprint
- Merge everything into a dedicated development branch first before merging it to master to protect master even better
- Subdivide bigger tasks into smaller tasks so that multiple people can work on them
- Make sure that the workload is more evenly balanced