Sprint Retrospective, Iteration #3

Microservice	Issue	Assignee	Effort	Done?	Notes
User/Auth	The system must not only authenticate users, but also assign them roles which determine which actions they can perform.	Alan Kuźnicki	4h	Yes	The base infrastructure for authorization is done and some methods are annotated with relevant @PreAuthorize annotations. However, not all that should be, are.
Cluster	All strategies in the cluster microservice should be well tested with a focus on good branch coverage	Alan Kuźnicki	3h	Yes	-
	A system admin should be able to see the complete schedule from the cluster	Alan Kuźnicki	15 min	Yes	This was mostly already implemented, it just needed a relevant annotation to limit it to users with the SYSADMIN role.
	A system admin should be able to see the current capacity of the cluster per node	Alan Kuźnicki	15 min	Yes	As above.
	A system admin should be able to see the reserved and available resources for each day and faculty	Alan Kuźnicki	2h	No	Required some tinkering with the endpoints and is still not done. Extensive refactoring of the services seems in order. A new service might need to be added.

	A user must be able to delete a node that he has contributed to the cluster (second task of the issue - deletion after midnight)	Ariel Potolski Eilat	5h	Yes	The issue is done, however I encountered problems during testing. The implementation is correct though, as I tested it manually through Postman.
	Increase the branch testing coverage of the node and job classes.	Ariel Potolski Eilat	15 min	Yes	The branch coverage for both classes now is 100%.
User/Cluster	A user must be notified about their job's status by the cluster	Ariel Podolski Eilat		No	This issue is not done yet. I have already implemented a service sendNotification and a Notification class in the cluster microservice. The sendNotification sends the given notification to the user microservice. I also have implemented the first case where we send a notification, which is right after we schedule the job. I still need to add the other cases where we send a notification and test everything. None of these changes have been merged into main or dev yet.
User	A user can get notifications in a given timeframe	Bart Coster	6h	Yes	There is an endpoint to retrieve notifications, where you can give an timeframe optional timeframe from when to when you want the notifications.
	A user can delete notifications	Bart Coster	5h	Yes	Only the user can delete notifications; there should be an option for the system/Admin to also do it. That can be done in a next sprint

	Endpoint to retrieve all faculties	Kasper van Duijne	1h	Yes	Went well.
	Refactoring Notification	Kasper van Duijne	1h	Yes	Small troubles with Spring errors.
	Testing User service	Kasper van Duijne	3h	No	Problems with Spring, partly solved, but tests could be expanded.
	Change to repository for notifications	Kasper van Duijne	3h	Yes	Went well, some troubles with dates not being the same before stored in the database versus after retrieved.
Request	Increase the branch and line coverage of the Job Controller class	Alexander Andonov / Vladimir Pavlov	4h	Yes	Added tests for the time specific requirements to the sending of requests. We still need to test some of the branches and specifications though.
	Increase the branch and line coverage of the Request Allocation class	Alexander Andonov / Vladimir Pavlov	5h	Yes	The sending of valid and invalid requests to the cluster is tested. Also, requesting if enough resources are available is tested and works. We still need to test the methods that ask the cluster if there are enough resources for the specific job, because we weren't sure how to mock that interaction between the microservices.
	6 hours before the following day, a request should be automatically approved as long as there are free resources available for the faculty of the requesting user.	Alexander Andonov	1h	No	Focused mainly on testing, since we figure coverage is more important and this issue can be quickly added in.
	5 minutes before the following day, no request	Vladimir Pavlov	3h	Yes	The additional requirement to the functionality of sending requests works

	should be accepted for that day anymore.				without previous implementation.

Problems encountered

1. Some classes or methods are exceedingly difficult, or outright impossible to test.
    1.1. Description: the removal after midnight for the node. We could not create an integration test that tested the correct behavior of our implementation. Furthermore, some of the functionality of the Request allocation class requires back and forth interaction between microservices, which poses a difficulty as well.
    1.2. Reaction: After spending a lot of time trying to figure out how to write the implementation test for the removal after midnight, we decided to only test it manually using Postman. In the future, if we encounter similar “untestable” functionalities that cannot be refactored to become testable, we will simply test them manually.

2. It is sometimes difficult to decide in which service to put specific functionality and how to divide those.
    2.1. Description: some functionality, like extracting certain information from the repository, is difficult to assign to a specific class with 100% certainty. This is especially visible in the cluster, with a multitude of services and conflicting ideas on how to separate functionality between them and how granular they should be.
    2.2. Reaction: for now, we have put the functionality where we thought it fit best. We made several separate services as well, since we believe that potentially merging them will be easier that having to untangle and separate them. Extensive refactoring is due and will be conducted in the next sprint.

3. The repository for Notification changes the Date it was sent.

3.1. Description: when a notification is sent to the repository it changes the Date. it will not change the day it will add some things and the equals does no longer work.

3.2. Reaction: the date does not matter for the moment so it is not yet fixed but we will try to tackle it next sprint. In the test where the equals was needed it took another approach. Where it looked at the amount of notifications in the system instead of the exact notification.

4. Testing the service classes in the cluster microservice

4.1. Description: one of the issues we were going to tackle on this sprint in the cluster microservice was to test the service classes. However, this proved to be very challenging due to all the dependencies we have in our code.

4.2. Reaction: We decided to first refactor the code, and try to make it more testable. After that, we will work on the testing of the service classes.

5. Implementing the Notification Database in the User Microservice

5.1. Description: the java.util Date gave some problems. A notification's date format changed when storing it in the repository database. So a notification retrieved from the database was different, and thus tests were not passing.

5.2. Reaction: There were multiple solutions. Either try to find some Spring annotations in order to fix it, change the date to a string in the database, or reimplement the date's equals method, so comparing 2 dates in different formats will work correctly, if the dates are the same. We chose the latter, and it worked as expected.

Adjustments for the next sprint

1. We will open more merge requests earlier before the end of the sprint to avoid congestion.
2. We will focus on testing interaction between microservices and potentially improve our system design, overall, now that we have the functionality.
3. We will refactor and comment our code to make sure it’s understandable, follows proper software engineering methods, and adheres to the requirements and our UML diagram.
4. We will make sure that every issue is either included in our final project or discarded with a proper justification.