

ASSIGNMENT 3

Group 31b

*Alexandra-Ioana Neagu, Bozhidar Andonov, Ferhan Yildiz,
Jannick Weitzel, Luuk van de Laar, Tudor-George Popica*

1. The classes we decided to test

During the project we already gained pretty good mutation coverage, mainly in the *reservations* microservice. Therefore, we didn't have too many classes to choose from. The classes we went with were:

- **UserAPI**
- **JwtRequestFilter**
- **FilteredAuthenticationService**
- **AuthenticationAPI**

As you can see, these classes are mainly from the *authentication* microservice, since this microservice was the least tested one, during the project.

2. The problem and solution

Before we could improve coverage, the first step was to identify where the classes lacked it. Then we added tests to those parts of the code so the mutation score could go up.

2.1 UserAPI

In the **UserAPI**, in the method **deleteUser**, the call to the method **deleteUser** in the **UserService** had not been tested properly yet. Meaning if mutation testing set the line that called that method to null, the current tests would not catch that change yet.

The solution to the problem was to create a test for the **deleteUser** method in the **UserAPI**, which is what we did. After creating a test for the method, **deleteUser** had indeed been called, like expected. We resolved the lacking coverage and brought mutation coverage from 50% to 75% - an improvement of **25%** for this class.

Name	Line Coverage		Mutation Coverage	
UserAPI.java	67%	<div><div>10/15</div></div>	50%	<div><div>2/4</div></div>



UserAPI.java	87%	<div><div>13/15</div></div>	75%	<div><div>3/4</div></div>
------------------------------	-----	-----------------------------	-----	---------------------------

2.2 JwtRequestFilter

In the `JwtRequestFilter` class, the `doFilterInternal` method had not been tested at all, resulting in a low mutation coverage for the class.

To fix this, we added a test that verifies that the `doFilter` method is called correctly inside the `doFilterInternal` method by the `FilterChain`, and it also checks that the `HttpServletRequest`, `HttpServletResponse`, and `FilterChain` are instantiated and wired correctly.

Verifying these has increased the mutation coverage by **22%** (from 22% -> 44%).

Name	Line Coverage		Mutation Coverage	
JwtRequestFilter.java	40%	<div><div>8/20</div></div>	22%	<div><div>2/9</div></div>

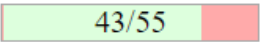
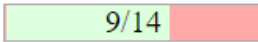


JwtRequestFilter.java	40%	<div><div>8/20</div></div>	44%	<div><div>4/9</div></div>
---------------------------------------	-----	----------------------------	-----	---------------------------

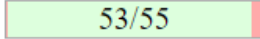
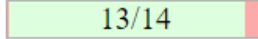
2.3 FilteredAuthenticationService

In the `FilteredAuthenticationService`, in the method `changePasswordUser`, the interaction with the database had not been tested properly yet. Meaning if mutation testing set the lines that updated the database to null, the current tests would not catch that change yet.

The solution to the problem was pretty straightforward. By mocking the service handling the database (`myUserDetailService`) and verifying whether the methods `deleteUserCredential` and `addUserCredential` had indeed been called, like expected. We resolved the lacking coverage and brought mutation coverage from 64% to 93%- an improvement of **29%** for this class.

Name	Line Coverage	Mutation Coverage
FilteredAuthenticationService.java	78% 	64% 



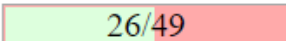
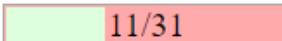
FilteredAuthenticationService.java	96% 	93% 
--	--	---

2.4 AuthenticationAPI

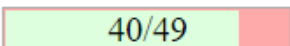
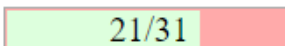
There were some methods in `AuthenticationAPI` which had not been tested properly yet. We decided on testing 2 of them:

- `adminRegistration` - this method had none of its branches covered which greatly reduced both mutation and branch coverage by a lot. That was why it was a perfect fit for testing this class.
- `changePassword` - this method also did not have any mutation coverage so we also tested it.

We created 8 new tests and did not modify any existing tests to increase the mutation score of this class. The mutation coverage went from 35% to 68% - an improvement of **33%** for this class.

Name	Line Coverage	Mutation Coverage
AuthenticationAPI.java	53% 	35% 



AuthenticationAPI.java	82% 	68% 
--	---	---