

SEM 13a - DelftBlue assignment 3

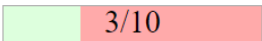
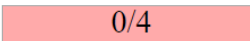
Michiel Bakker - Bink Boëtius - Ionut Ciobanu - Ivans Kravcevs - Nils Mikk

Automated Mutation Testing

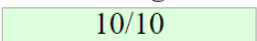
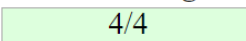
NodeRepositoryService (Merge request [!58](#))

NodeRepositoryService class in Resources-microservice was not tested properly. Tests were added to kill all mutants in the methods of this class.

Before:

Name	Line Coverage	Mutation Coverage
NodeRepositoryService.java	30%  3/10	0%  0/4

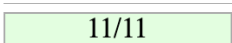
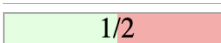
After:

Name	Line Coverage	Mutation Coverage
NodeRepositoryService.java	100%  10/10	100%  4/4

ScheduleService (Merge request [!61](#))

When running PITest on the schedule microservice, it is shown that the mutation coverage of ScheduleService was only 50%. So this class was not properly tested. Tests were added to kill all the mutants PITest introduced.

Before:

ScheduleService.java	100%  11/11	50%  1/2
--------------------------------------	--	---

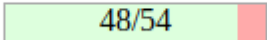
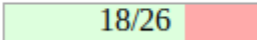
After:

ScheduleService.java	100%  11/11	100%  2/2
--------------------------------------	--	--

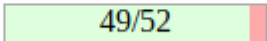
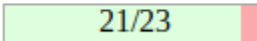
WaitingListController (Merge request [!57](#))

The mutation coverage of WaitingListController was improved from 69% to 91%. A detailed description of what was changed can be found in the merge request.

Before:

Name	Line Coverage	Mutation Coverage
WaitingListController.java	89%  48/54	69%  18/26

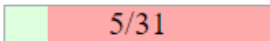
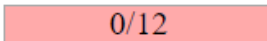
After:

Name	Line Coverage	Mutation Coverage
WaitingListController.java	94%  49/52	91%  21/23

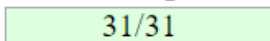
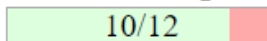
ResourcesController (Merge request [!62](#))

The ResourcesController was not tested properly and because of that, no mutations were killed, resulting in a 0% mutation coverage. Now there are tests added in the ResourcesControllerTest class that kill almost all mutations, resulting in a 83% mutation coverage.

Before:

Name	Line Coverage	Mutation Coverage
ResourcesController.java	16%  5/31	0%  0/12

After:

Name	Line Coverage	Mutation Coverage
ResourcesController.java	100%  31/31	83%  10/12

Manual Mutation Testing

Info about the core domain

AddNewRequestService (Merge request [!59](#))

AddNewRequestService is an essential service in the User microservice. It was created after refactoring the addRequest method of the controller for assignment 2. It handles the logic of creating a request, sending it to the waiting list, and saving the status to the database.

It was found out, during the manual mutation testing of this class, that `contactWaitingList` method (that handles sending requests to the waiting list and saving status to the database functionality) is not tested for the case of not proper response (without exception throwing) from `WaitingList`. It was found out by injecting null into one of the return statements. An additional test case was added to kill this mutant. (Full description and screenshots in merge request).

FacultyAdminController (Merge request [!64](#))

FacultyAdminController is an essential class in the application, because this class is needed for the faculty admins to accept and reject requests. If this class did not exist, it would not be possible for admins to choose between the requests, which would be a big problem for the functionality of the application.

During the manual mutation testing of the essential class, it was found that it was possible to inject a bug that was not killed by the tests of the class. The bug was the following: in the catch part of the method, a `ResponseStatusException` "BAD-REQUEST" should be thrown, but if the `HttpStatus.BAD_REQUEST` were changed to `HttpStatus.OK`, this would not throw the right exception. An additional test case was added to ensure that the exception that was being thrown was a `BAD_REQUEST` exception. After adding this test, the manual mutation was caught by the tests of the class.

SingleTableWaitingList (Merge request [!60](#))

The class `SingleTableWaitingList` is critical to our application since it deals with adding requests, removing them, and getting requests from the pending requests database. The method `addRequest` is used for adding requests to the waiting list, so it is important that it works properly.

We injected a bug in `addRequest` that was not found by our test suite. On line 44, `savedRequest.getId()` was changed to return `1L`. The return value of the method is used by the user microservice for storing request statuses. The ID is also given to the user, so that they can check their request. A test case was added that would catch this bug.

RequestValidationService (Merge request [!65](#))

The `RequestValidationService` class is a class in the `Schedule` microservice. It is called by the `scheduleRequest()` method in the `ScheduleController` class to check if the requests that are going to be scheduled are actually valid. Since the DelftBlue supercomputer will use the schedule stored in this microservice while processing the requests, we do not want any invalid requests in this schedule and therefore this class is critical to our application.

The method in this class where a bug was injected manually is the `valid()` method. This method checks whether the request can be scheduled on the preferred date. Requests can only be scheduled on days after a request was received to schedule a request. There is a special case, however, that a request can only be scheduled for the next day if the request to schedule itself is received before (and including) 23:55. However, when injecting a bug, so that the time to check is reset to 00:00 instead of 23:55, the test suite still passes, while this would mean a request could never be scheduled for the next day, which is not the case.

By adding a new test that checks if a request, received at 23:55, to schedule a request for the next day, this mutation is killed.