# 

TEST DOCUMENT TITLE

Quality Assurance of the solution

[Document information](#h.qwydk7x1nscw)

[Document version](#h.qgvwgi1jjpq)

[Approval List](#h.6r7j8dh65x32)

[Confidentiality Rating](#h.hes98nyepwis)

[General](#h.1agiq4l899wt)

[Stress Performance Test](#h.hk3acqe9xl2n)

[User Acceptance Test](#h.mz9eoe1a5djk)

[Other Tests?](#h.e4pd5f1cycku)

# Document information

## Document version

|  |  |  |
| --- | --- | --- |
| Version | Author e-mail | Description |
| 1.0 | trlb@kea.dk | Initial draft |
|  |  |  |

## Approval List

|  |  |  |
| --- | --- | --- |
| Who | Function | E-mail |
| Nikolaj B. Hemmeshøj | Head of Enterprise Architecture | nibh@kea.dk |
| Jarl Tuxen | Chief Information Security Officer | jart@kea.dk |

## Confidentiality Rating

|  |  |
| --- | --- |
| Rating |  |
| Company Confidential | X |
| Non Confidential |  |

# **General**

This document contains information about our testing process.

We need our performance testing to ensure that our program is fast and error free under any user load. And to ensure that the application will work as intended when all software units and methods are executed together.

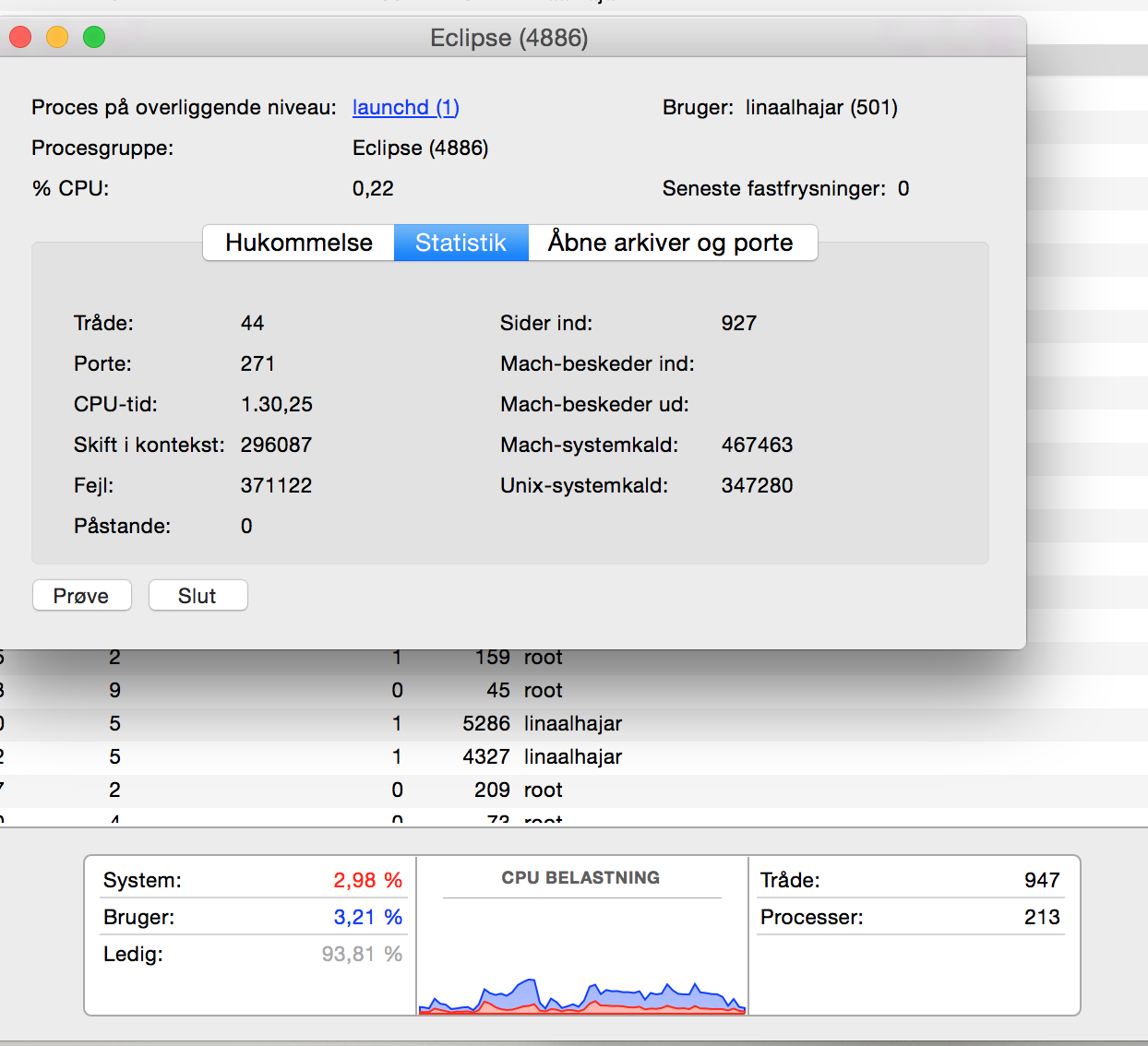
# **Stress Performance Test**

**Load test:**

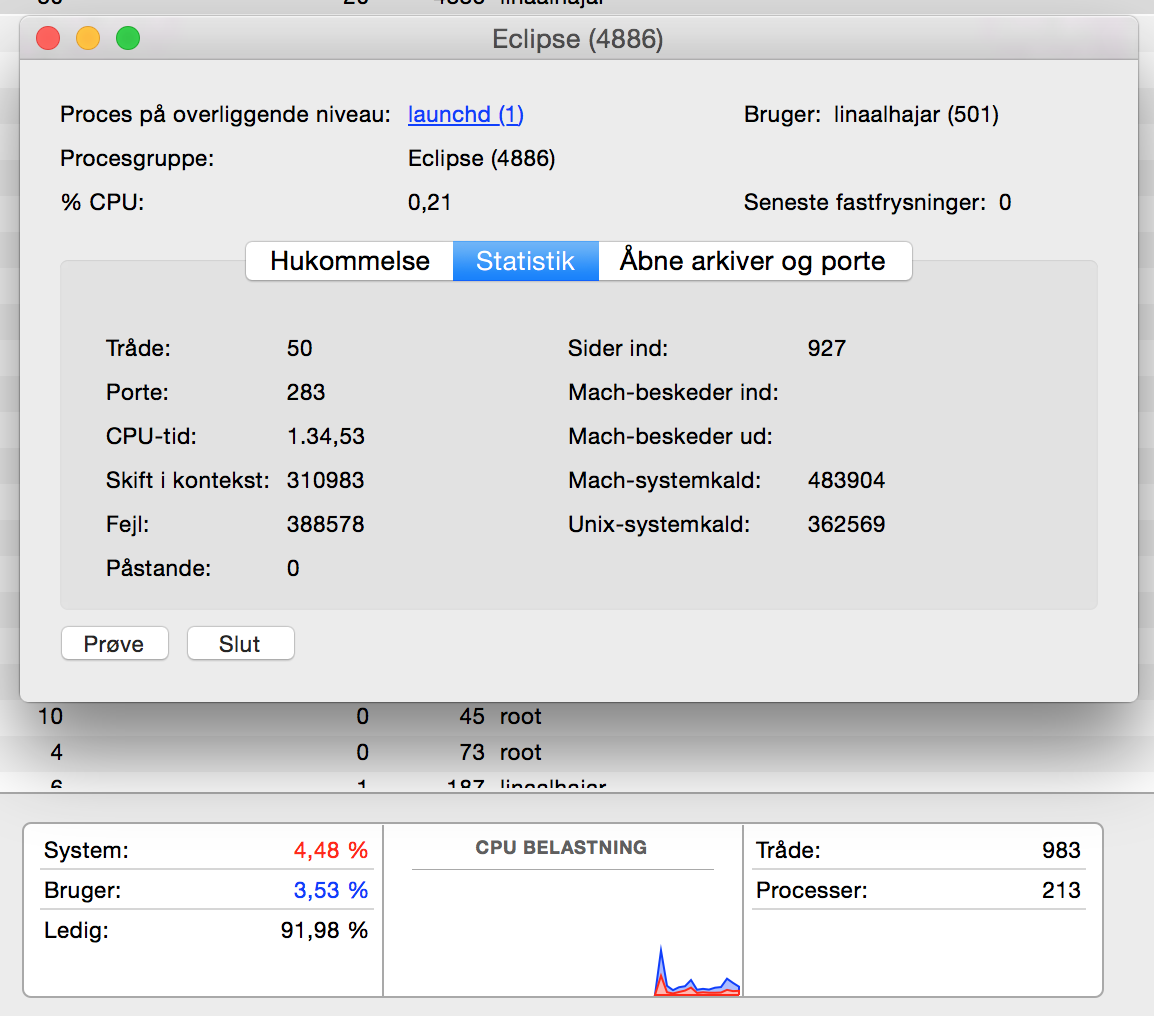
Load test case: User Log in:

**Stress Test**

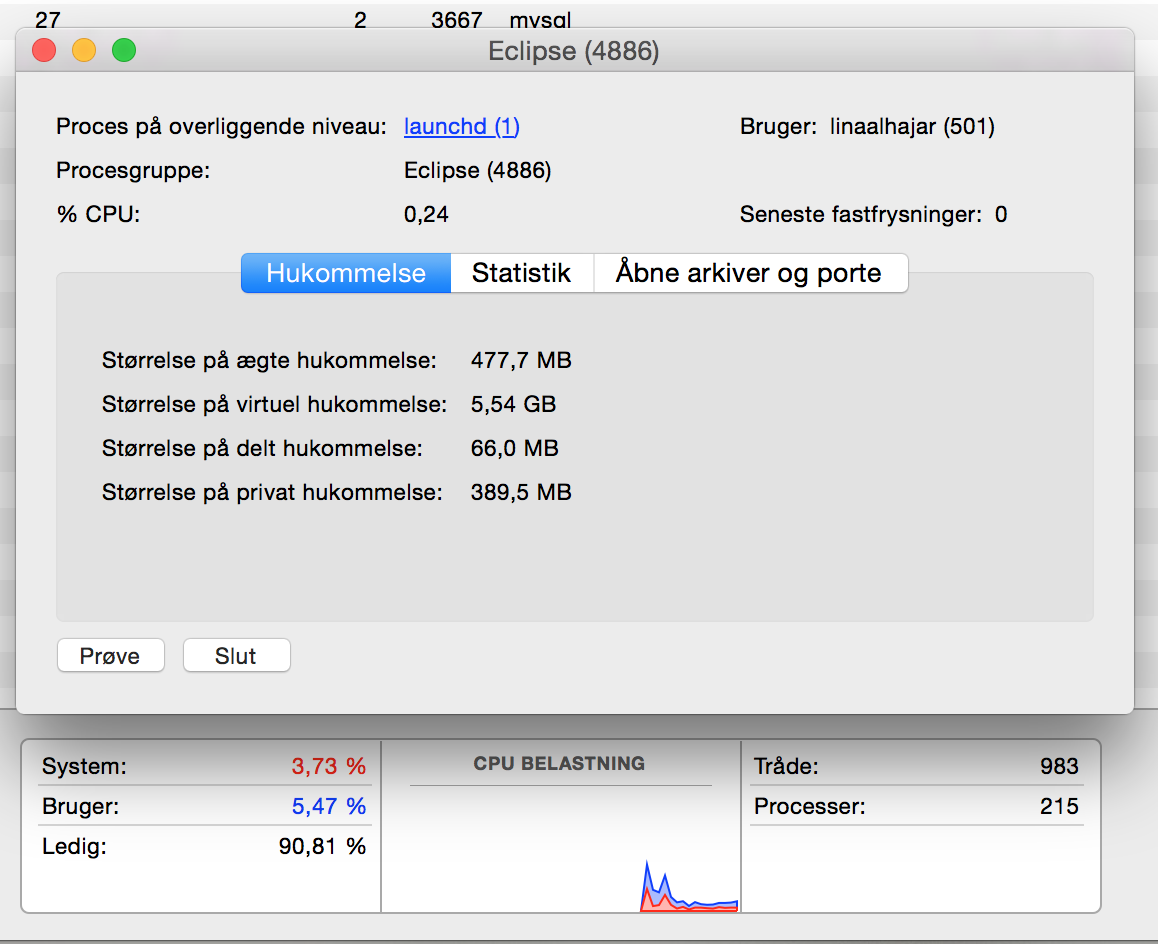
**Screenshots of memory usage:**

1.Lunching the Application

2. Starting the Server



3. Starting the Client



**Spike Test** (how does the system recover from a load burst)

to verify a system's stability during bursts of concurrent user and or system activity to varying degrees of load over varying time periods.

Spike testing should also verify that an application recovers between periods of spike activity.

**Soak Test** (no system degradation over time - e.g. memory utilization)

# 

# **User Acceptance Test**

The Client or the user can sign out of the application at anytime.

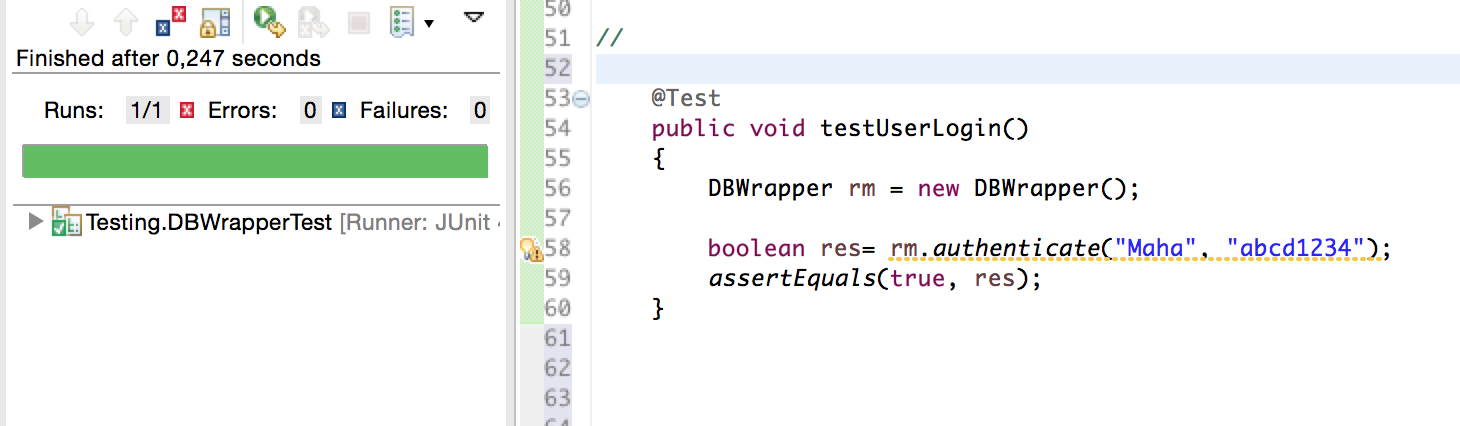
**JUnit Test**

We have used JUnit test to test the functionality required for certain methods of our project to ensure that the code is implemented, and that there are no breaking changes that have not been taken into consideration.

We have tested the following cases:

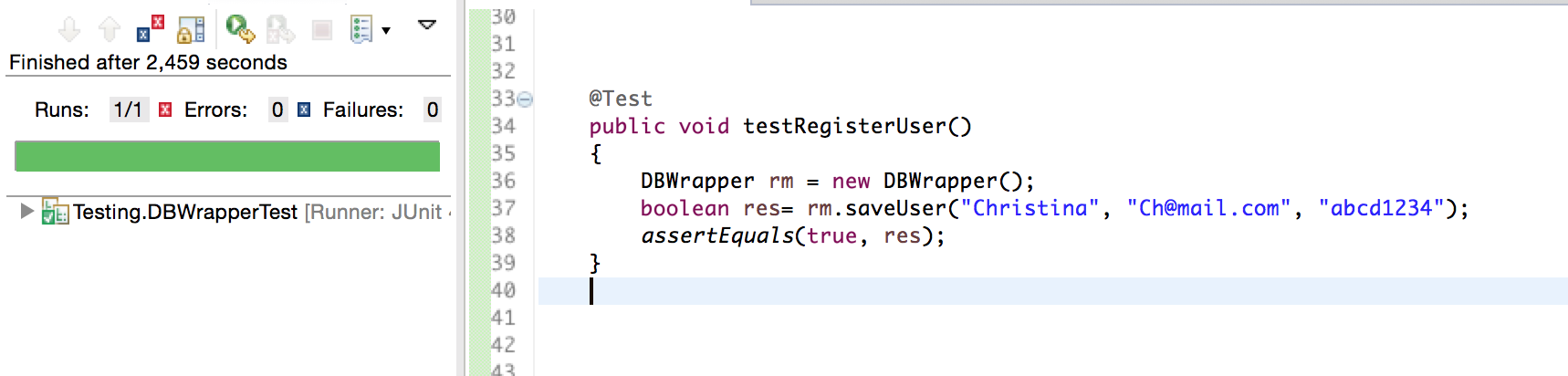
* User log in.

This test is testing if the user name and password is existing in the Database.



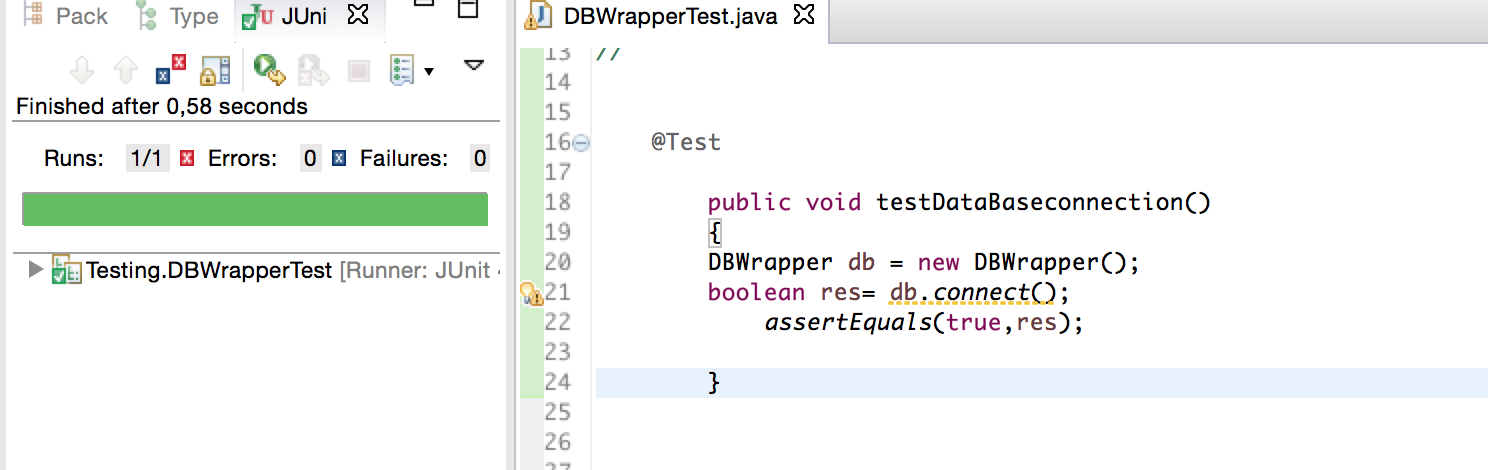
* Save User (Register User )

Testing if the saveUser method is saving the user with the given data.



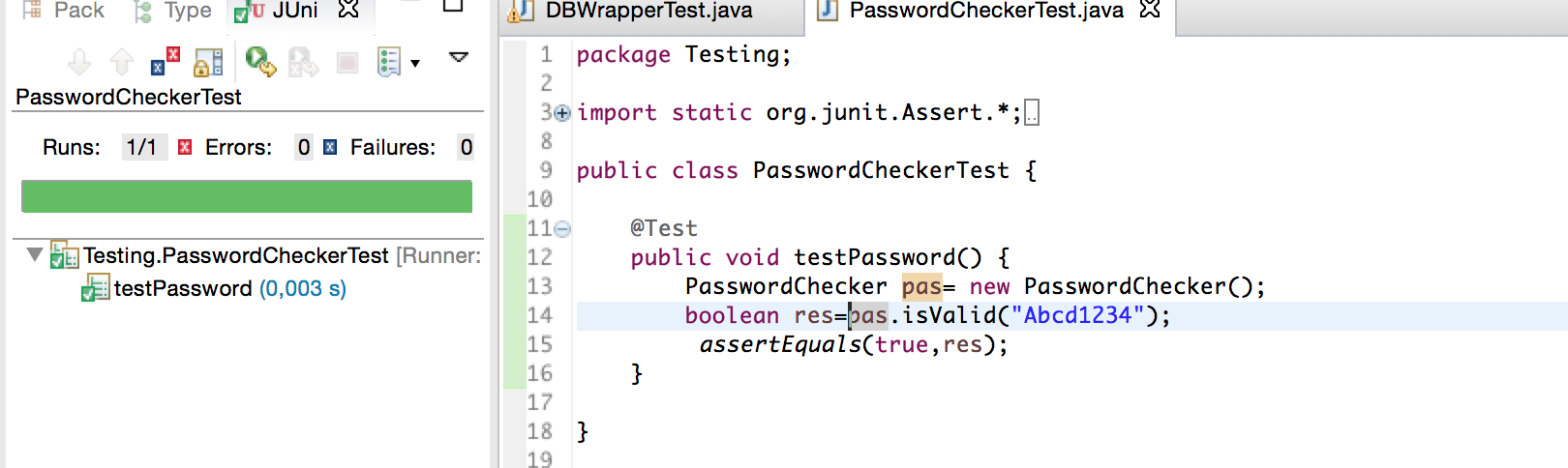
* Database connection.

Testing if the Database connection is established.



* Password validation.

Testing if the password is valid according on the requirements



* Email Validation

We have tested to cases the first one testEmailValidation() where the email is written right and valid, and the second one testEmailValidation2() is where the email is not valid.

