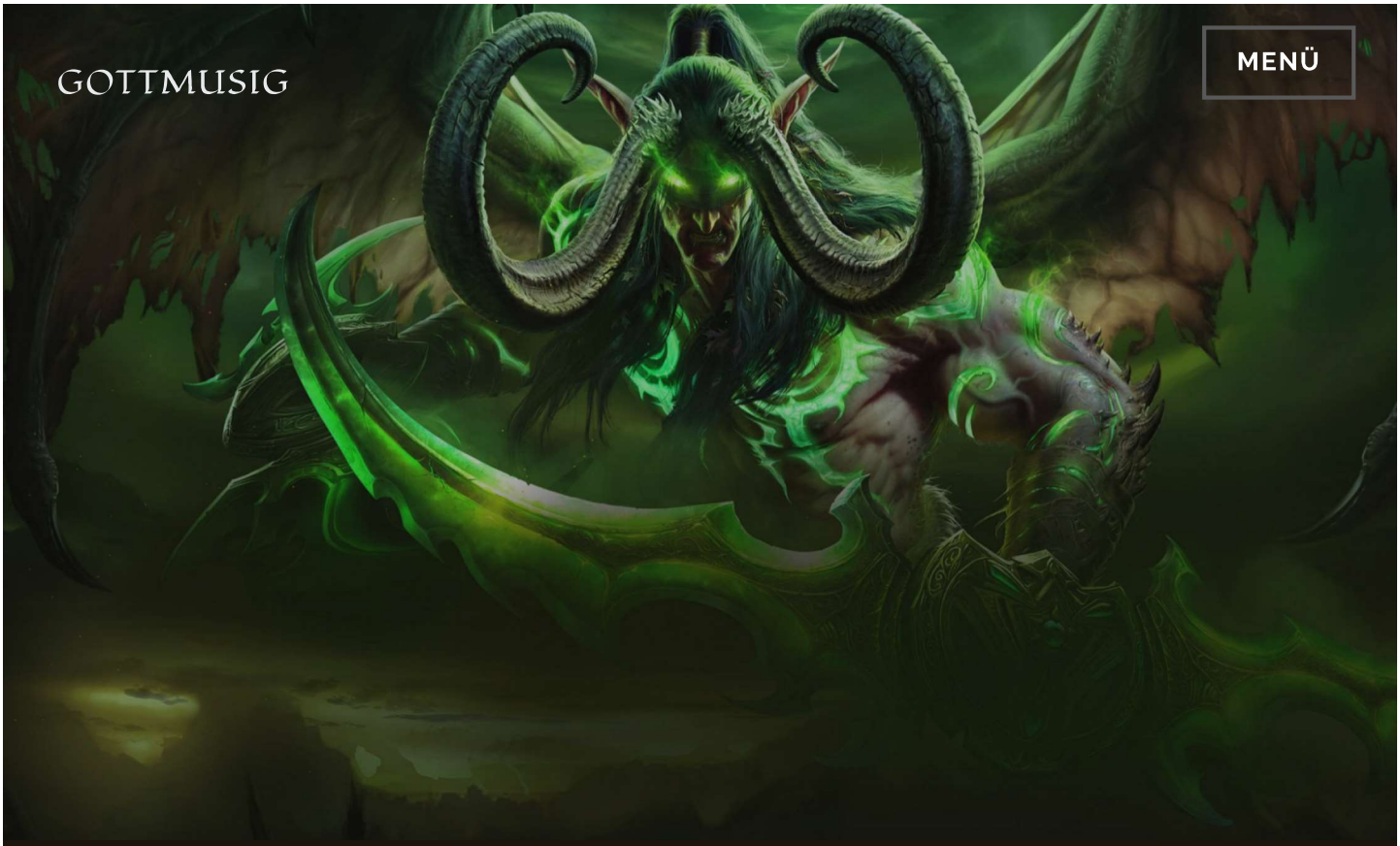


Development

Final Handin

4. JULI 2017

GOTTMUSIG

HINTERLASSE EINEN KOMMENTAR

Hey guys,

we officially finish our Software Engineering project GottMusIg with this blog post, but we haven't finished working on it, so look out for more posts on this blog!

Our accumulated work over the 2 Semesters:

Requirements:

Software Requirements Specification
– OUCD

UC-1: DPS Difference

UC-2: Choose Character

UC-3: Show Character

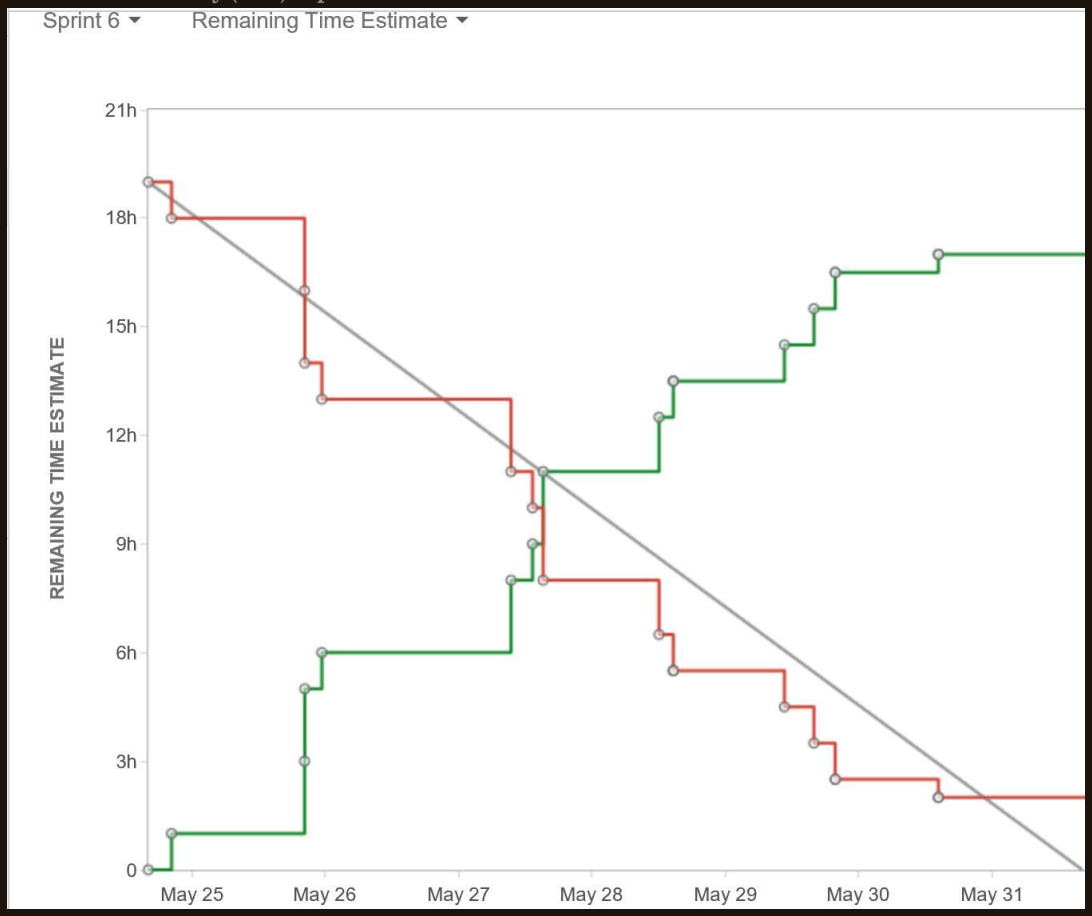
UC-4: Update Database

UC-5: Update Character

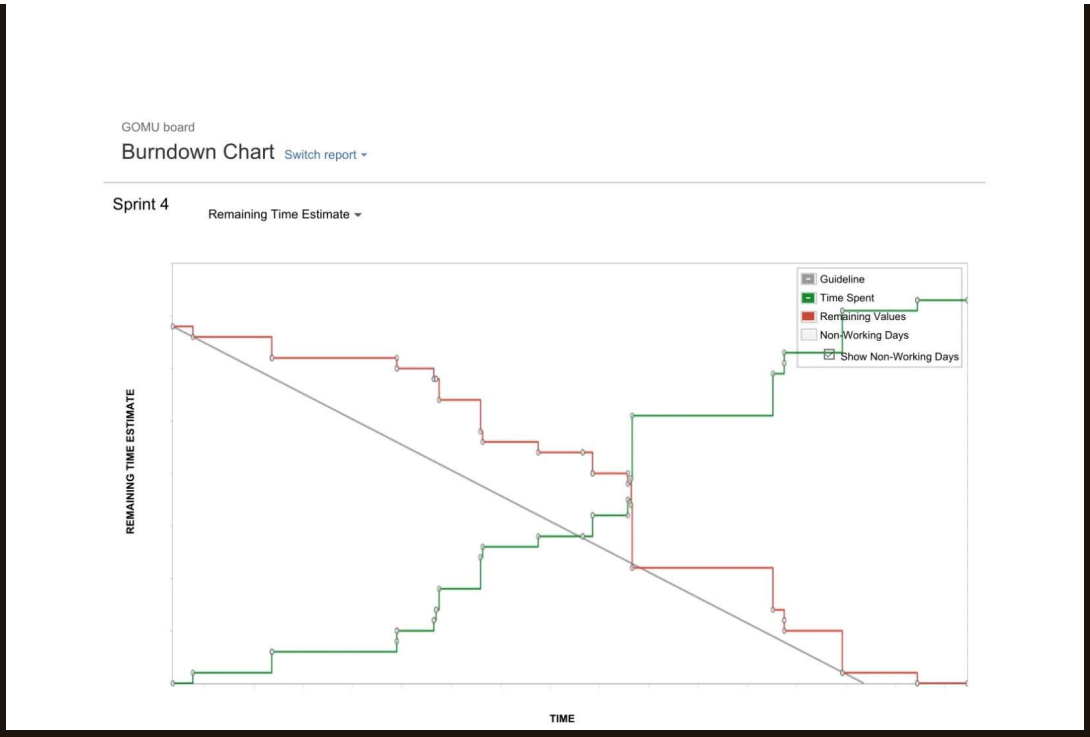
Test Cases:

- Feature Files (blog post)
- Feature Files (github) and click
- also see Test Plan
- Test Report (Jenkins)

Project Management

- Gantt Chart
- Scrum with Jira
- Here are 2 healthy(ish) Sprints:
  - 

  - Sprint 6 –> link

- 

Sprint 4 –> link

- Function Points (FP)

Code

- GitHub organization
  - branch graph (DatabaseService)
- Automatic deployment on Maven Repository via Jenkins
- we showed the working application during our final presentation

Quality

- Architecture
  - Software Architecture Document (SAD)
  - including metrics and patterns
  - relevant blog Posts
    - Class Diagrams
    - Software Pattern
    - Metrics
    - Refactoring
- Change Management
  - Test Plan
  - relevant blog Posts
    - Metrics –> SonarQube

- New Scope and Risk Management –> regularly updated risk plan
- CI and Installation –> automatically created test report
- Software Pattern

Presentations

- Midterm Presentation
  - Handout
- Final Presentation
  - Handout

Overview of blog posts

Archive

- Software
  - DatabaseService
  - Simulation
  - Frontend

All blogpost at a glance

What	Where
W1: Blog and Vision	read
W2: Team/Roles/Technology	read
W3: SRS	read
W4: UC+Prototype	read
W5: Scrum	read
W6: Gherkin feature files	read
W7: Class diagram	read
W8: MVC Tool	read
W9: MS Project Gantt	read
W11: Midterm Presentation	read
W2: Risk Plan/hours spent/new UC	read

W3: FP	read
W 4: Unit Testing	read
W 5: Refactoring	read
W 6: Pattern	read
W 7: Metrics	read
W 8: Test Coverage	read
W 9: Deployment CI	read
W10: Installation	read

- Blog
  - PDF

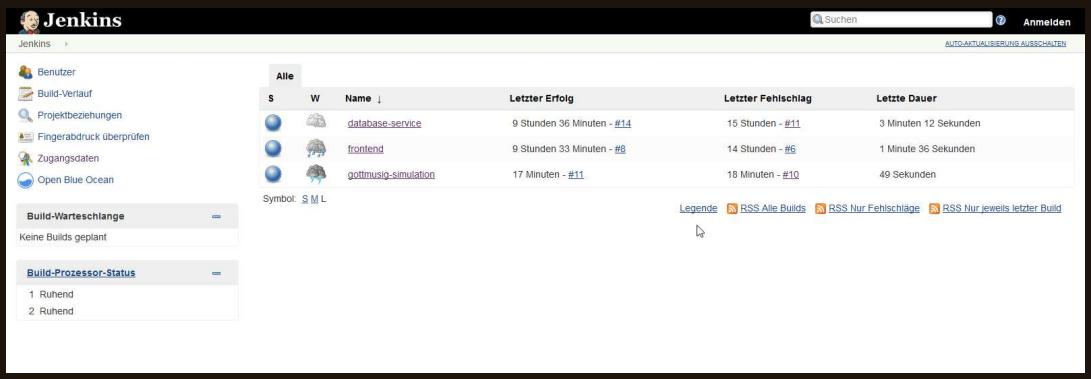
CI and Installation

8. JUNI 2017

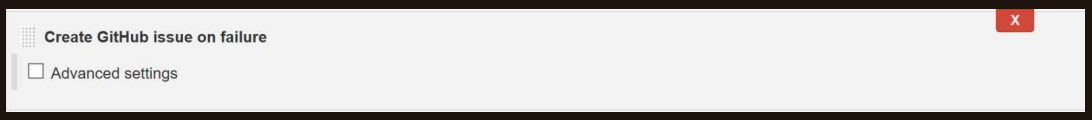
GOTTMUSIG

2 KOMMENTARE

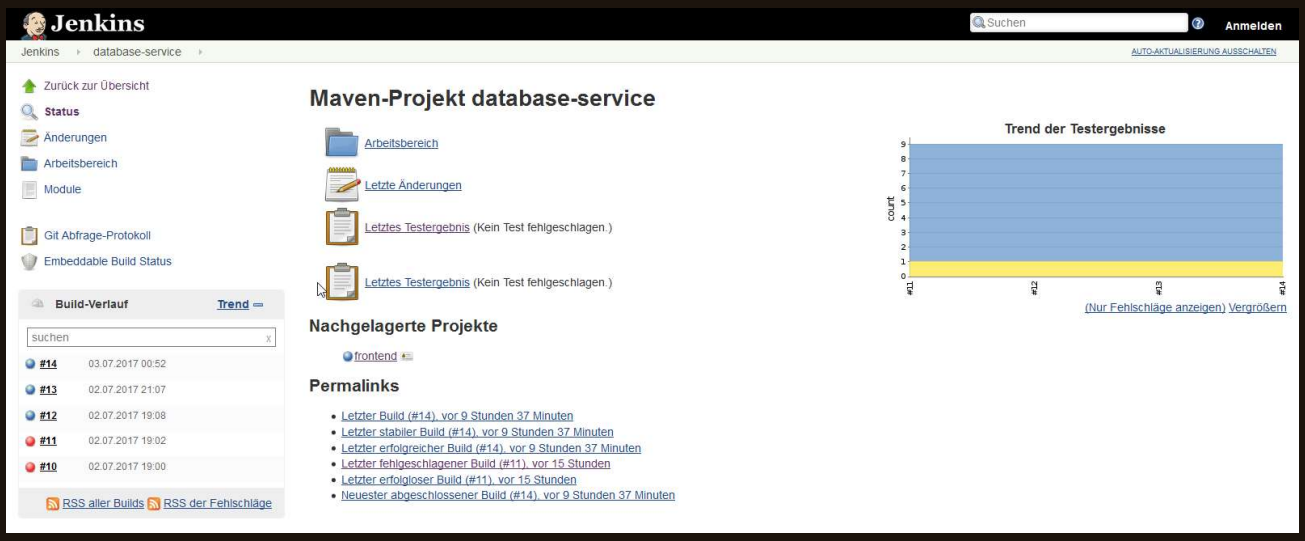
Hey guys,
today we want to show you our continuous integration process. Our projects are build and tested by Jenkins every day and on every change in our master branch.



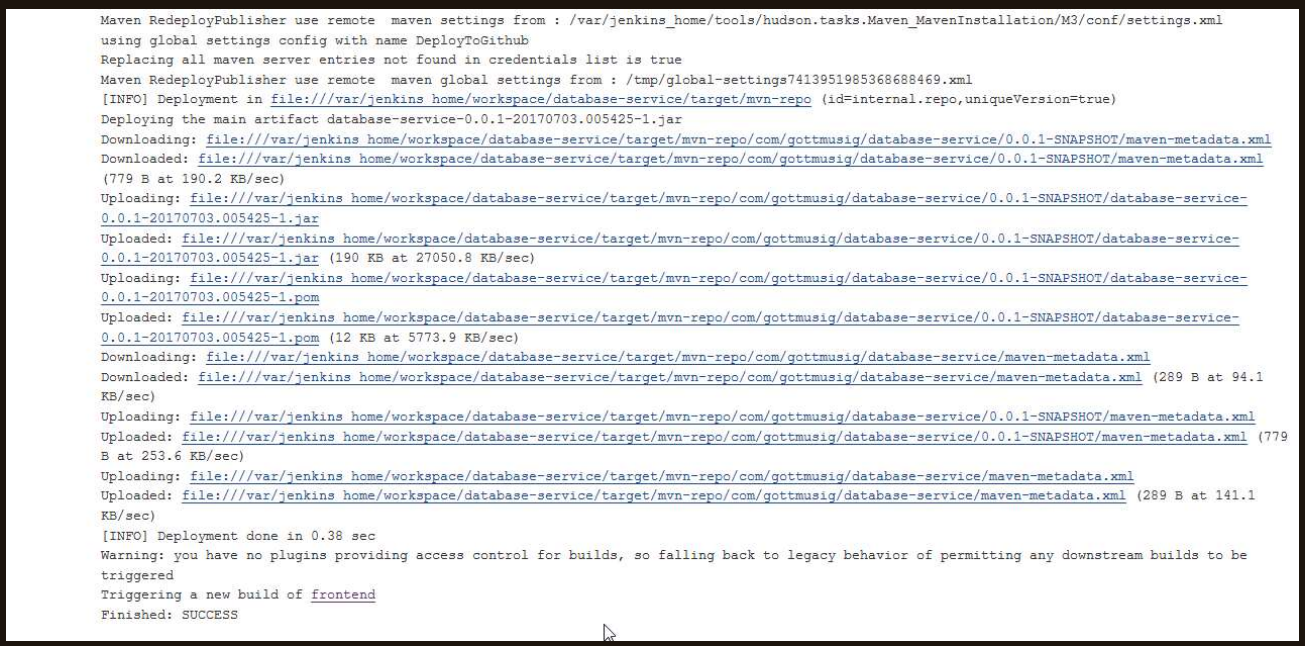
The generated metrics will be uploaded to Sonar . If there are failures in our tests the team will be notified by email, an issue on our GitHub repository will be opended automatically

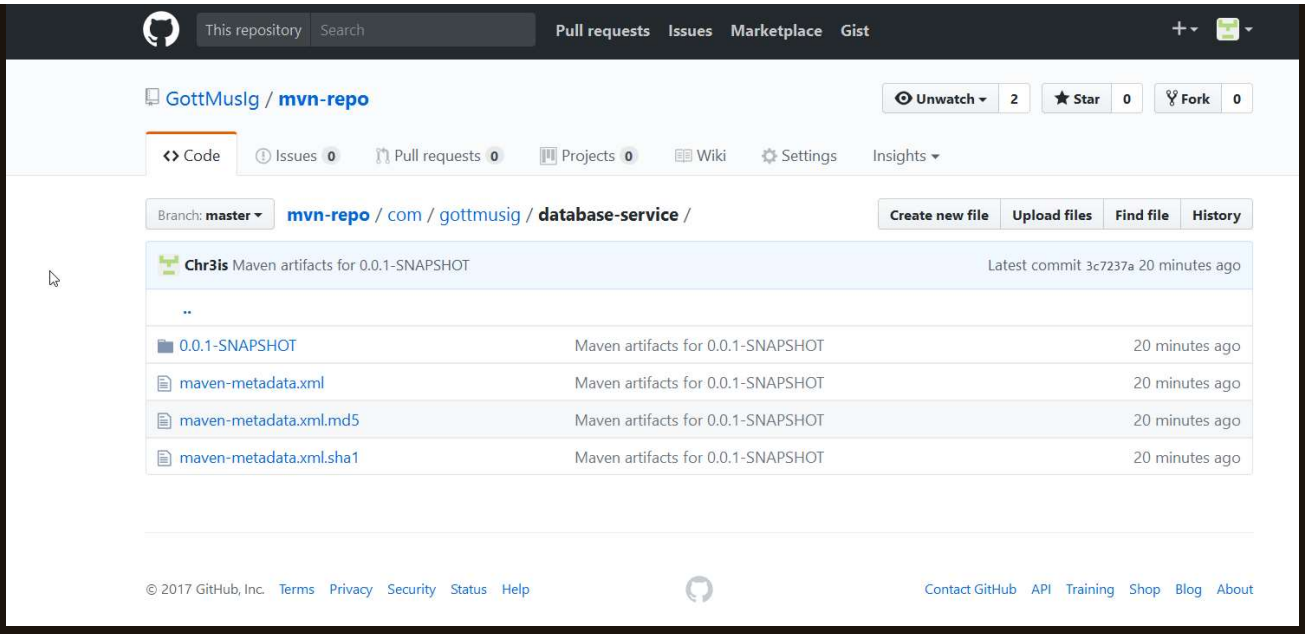


and there won't be a deployment to our artifact repository.



We have several projects and some depend on others. We created a maven repository on GitHub for the project which needs to be included as dependency in the other projects. If the build is successful the artifact will be deployed to our GitHub repository and every project gets the latest changes through the included dependency. So there's no need to check the project manually out and build it local.





Greets,
Team GottMusIg

Metrics

8. JUNI 2017
GOTTMUSIG
3 KOMMENTARE

Hi,

we are now using Metrics to help us develop better software.

The first Metric tool we use is sonarcloud.

Sonarcloud is fully integrated in our Continuous Integration process, it is part of our deployment process on our Jenkins Server.

Sonar has shown us some issues which we tried to improve and are on our way to a reliable software.

Now we're gonna take a look at a specific Problem SonarQube made us aware of:

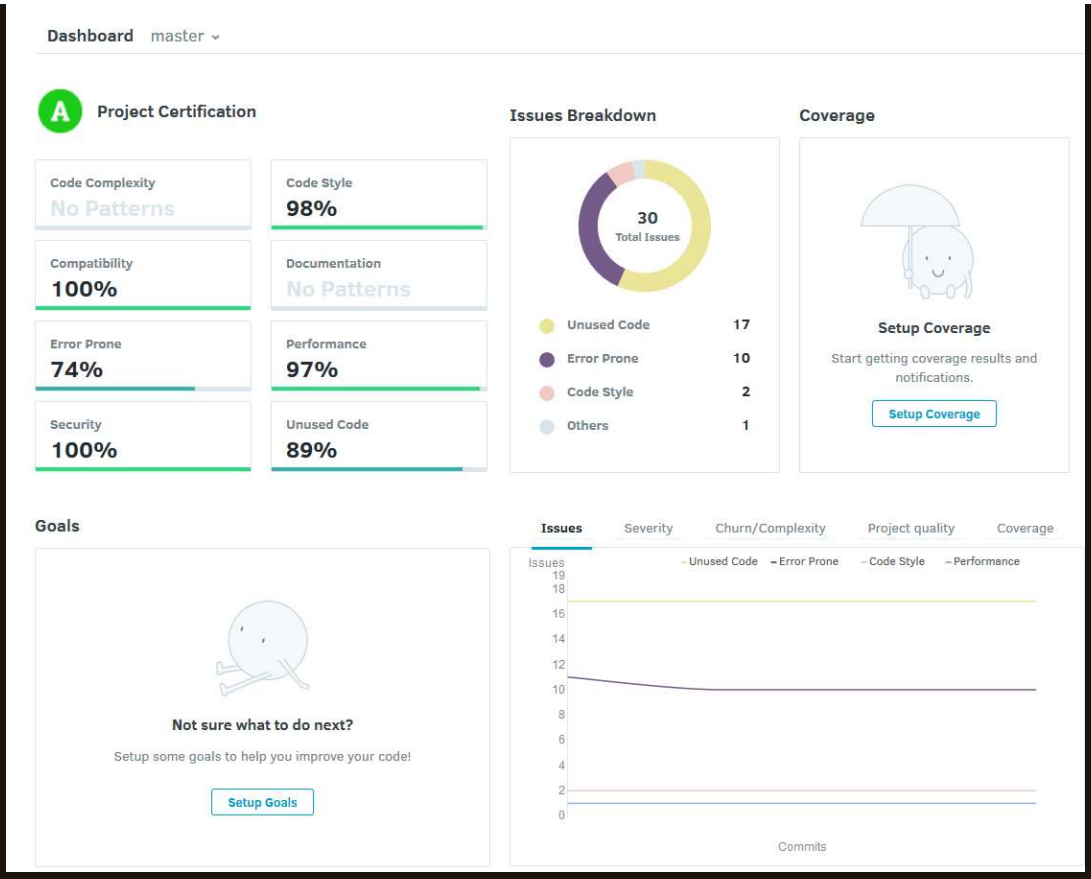
When we used the string „class" several times, sonar suggested to make an own field for this string to avoid many repetitions.

Also sonar warns us if the classes have too many parents:

But we could not fix this problem because of the frameworks we use.

We also use codacy as our second metric tool. It shows us our Code Style, Security, Unused Code and alot of other stuff.

Here is a picture of our Dashboard:



Greetings,

GottMusIg

Software Pattern

27. MAI 2017

GOTTMUSIG

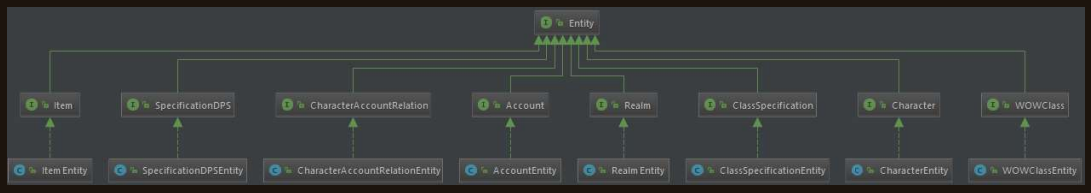
2 KOMMENTARE

Hey guys,

last week we looked at some common Software Patterns and how we can implement them into GottMusIg!

We decided to try out the dependency inversion pattern, which wants high-Level modules to depend on low-level modules and both should depend on abstractions.

The abstractions are seen in the picture below:



GitHub

Other modules of our Software use the abstraction „Entity“ to get the information from the several implementations. So other parts of our software don’t even know that there is an implementation, all they have is the interace where they can interact with the Entity below.

Greets,
Leon

Refactoring

15. MAI 2017

GOTTMUSIG

3 KOMMENTARE

Hey guys,

last week we did a bit of refactoring based on the book „Fowler's Refactoring, Improving the Design of Existing Code" by the Computer Science legend Martin Fowler.

Here you can see our exercises:

- Chris' Repository
- Kamils Repository
- Leons Repository

We hope to be able to write cleaner and better Code while programming GottMusIg based on the principles and proposals of the book.

See you next week,
Leon

Test Plan / Test Coverage

10. MAI 2017
GOTTMUSIG
2 KOMMENTARE

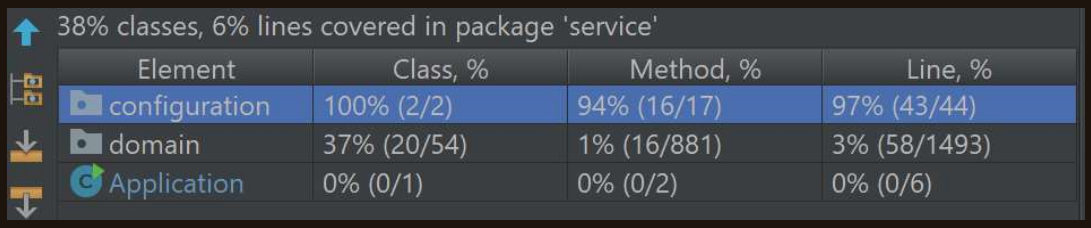
Hi guys,

this week we created a Test Plan for our project.

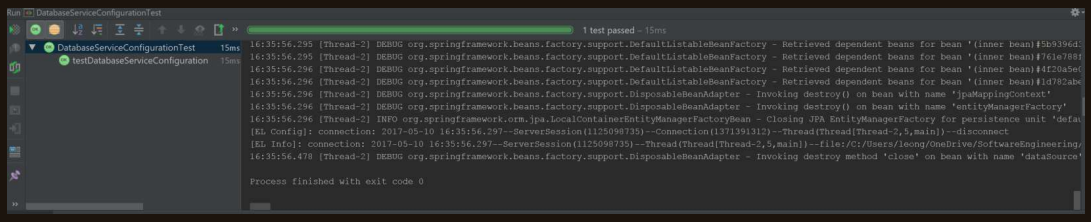
We use JUnit and Maven for automated testing.

We already wrote some Unit Tests for our database service as shown here, and added testing in our maven build process.

We are able to start the Unit test via our IDE and have an integrated Coverage tool as shown in the pictures below. (The coverage will go up soon, i promise 😉) .



Integrated Coverage Tool



Test successfully run in IDE

Our test dependencies are listed below.

```
<!-- TESTING -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <version>${spring-boot.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-all</artifactId>
    <version>1.10.19</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.193</version>
</dependency>
```

EDIT:

Here the update for week 8 of our Software Engineering Project.

We use Coveralls and SonarQube in our automated test workflow within our Jenkins Server. As of now we have 21% test coverage on our database-service. We also added all the badges! Check them out at our Project repository.

More info on our testing Tools and workflow in our Test Plan

Greets,

Leon

Function Point Estimation

23. APRIL 2017
GOTTMUSIG
2 KOMMENTARE

Hey guys,

another week passed and of course we were not idle.
This week we estimated our Use Cases with function points,
illustrated here.

You will hear from us next week.

Greets,

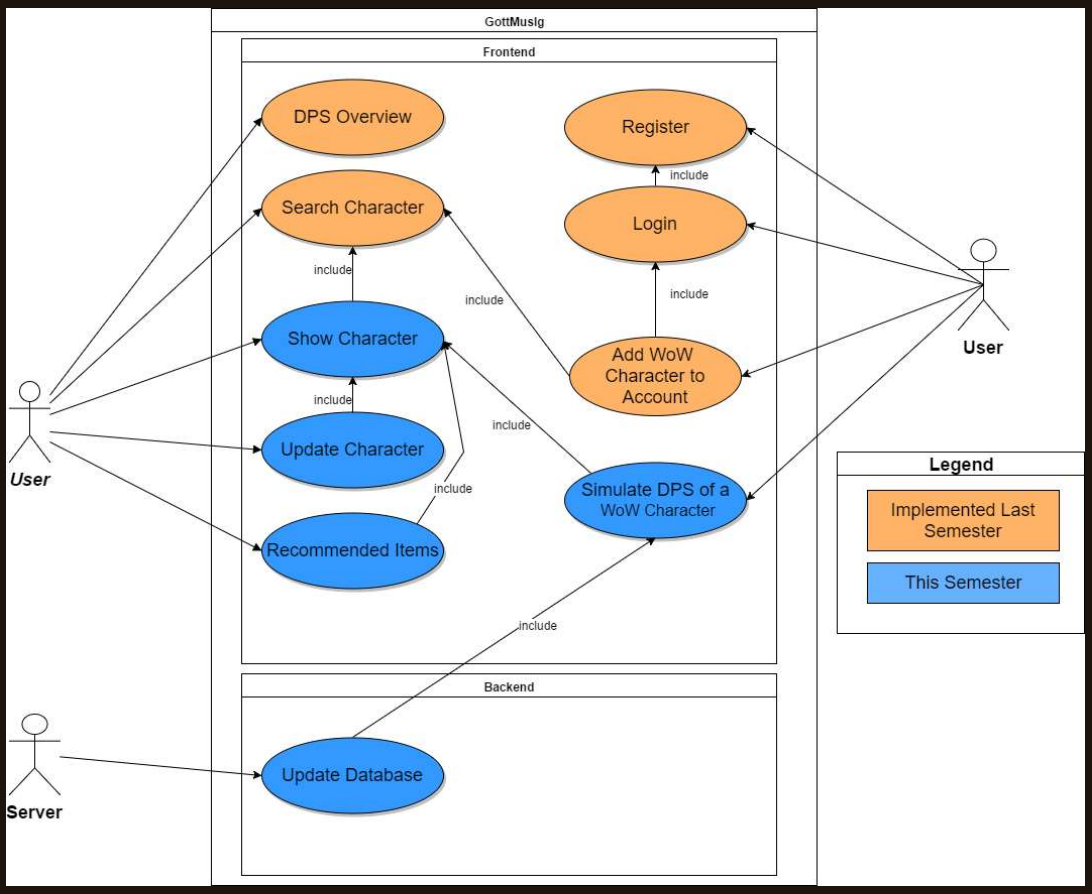
Leon

New Scope and Risk Management

16. APRIL 2017
GOTTMUSIG
3 KOMMENTARE

Hey guys,

as promised we are continuing our work on GottMusIg.
We redefined our scope and came up with 5 new use cases we want to implement this semester. We are confident we will accomplish our (ambitious) goals to present you an awesome website.



We also came up with some risks that could endanger our project and how to mitigate them. Check it out!

Risk-Management

More updates on GottMusIg coming next week!

Greets,
Leon

GottMusIg is back from the Holidays

10. APRIL 2017
GOTTMUSIG
HINTERLASSE EINEN KOMMENTAR

Hey guys,

we are back on track with our GottMusIg Project and are eager to make progress.

We won't drastically change our technologies nor architecture, but we do have a little bit of refactoring to do and we still have to implement a lot of features.

More info regarding our project will be blogged as always so stay tuned.

Greetings,
Leon

Midterm Handin

30. DEZEMBER 2016
GOTTMUSIG
1 KOMMENTAR

SRS

Software Requirements Specification

SAD

Software Architecture Document

UCD

Choose Character

DPS Difference

Feature Files

DPS Difference Feature

Choose Character Feature

Project Management

Jira

YouTrack (we will switch from Jira to YouTrack next Semester)

Gantt Chart

Gantt Chart

Code

all our projects are on our GitHub community GottMusIg.

It's worth mentioning that the GottMusIg project implements the DPS Calculation with SimulationCraft.

Presentation Slides

Slides