

SocialFunnel

Test Plan

Version <1.1>

Revision History

Date	Version	Description	Author
10.05.2015	1.0	Erster Entwurf	Simon Brückl
28.05.2015	1.1	Metrics added	Simon Brückl

Table of Contents

- 1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
- 2. Evaluation Mission and Test Motivation
 - 2.1 Background
 - 2.2 Evaluation Mission
- 3. Test Approach
- 4. Deliverables
- 5. Testing Workflow
- 6. Metrics

1. Introduction

1.1 Purpose

Dieser Test Plan enthält wichtige Informationen über den Aufbau unseres Testablaufes, der benötigten Entwicklungsumgebung und den von uns integrierten Testarten.

Die o.g. Informationen werden in diesem Dokument detailliert beschrieben.

1.2 Scope

Die von uns verwendeten Testarten sind Function Tests, Junit-Tests und Datenbanken Mocks. Diese wurden auch in unserem automatischen Testabläufen eingebaut.

2. Evaluation Mission and Test Motivation

Wir möchten ein funktionierendes Produkt liefern mit einer möglichst positiven User Experience. Dazu ist es notwendig die Webseite möglichst fehlerfrei zu betreiben. Auch ist eine fehlerfreie Implementation notwendig, um mit sensiblen Personen Daten sorgfältig umzugehen.

2.1 Background

Da unser Team aus mehreren Programmieren besteht, kann es vorkommen, dass beim Verändern, Löschen, und Hinzufügen von Code, anderen schon bestehenden Code negativ beeinflusst wird.

Das frühzeitige und automatische Testen soll diese Fehler aufdecken und den Programmierern deshalb ein Hilfsmittel sein.

2.2 Evaluation Mission

Wie oben erwähnt ist unser Ziel ein stabiles und qualitativ hochwertiges Spiel zu programmieren. Es sollen möglichst alle Fehler gefunden und behoben werden.

3. Test Approach

Dieses Kapitel zeigt, wie die Tests realisiert werden. Zunächst werden nach jedem commit automatisch die Tests mithilfe von Travis ausgeführt.



Anschließend werden die Testergebnisse an coveralls.io übertragen und ausgewertet.



Als letztes werden diese Informationen öffentlich auf GitHub im Projekt-Readme angezeigt.

SocialFunnel

build passing coverage 0%

3.1 Testing Techniques and Types

3.1.1 Data and Database Integrity Testing

Da nutzerbezogene Daten gespeichert werden, ist es sehr wichtig, sowohl auf Datensicherheit als auch auf Korrektheit zu testen,

Technique Objective:	Es soll geprüft werden, ob Daten korrekt in die Datenbank persistiert werden und ob diese korrekt wieder ausgelesen werden.
Technique:	Es werden Datenbankmocks verwendet um die CRUD-Methoden auf Korrektheit zu testen. Testdaten werden verwendet um die Datenbank zu testen.
Required Tools:	- Entwicklungsumgebung - Datenbank - Datenbankmanager
Success Criteria:	Werden Daten korrekt ausgelesen und gespeichert, sind die Tests erfolgreich.
Special Considerations:	Passwörter werden als Hash-Wert abgespeichert.

3.1.2 Function Testing

Das Ziel von Function testing ist, dass ein Produkt nach den funktionellen Anforderung geprüft wird. Hier wird also geprüft, ob sich das Programm nach den Anforderungen der Use Cases verhält.

Technique Objective:	Es soll geprüft werden, ob die Anforderungen der Use Cases richtig umgesetzt wurden.
Technique:	Es wird für jeden Use Case eine Schrittfolge beschrieben.

Oracles:	Dafür wird für jeden Use Case ein Feature File erstellt.
Required Tools:	<ul style="list-style-type: none"> • GitHub • Cucumber • Eclipse
Success Criteria:	

Die dafür verwendeten feature files sind in den jeweiligen Use-Case Dokumenten zu sehen. Diese sind auf unserem Blog unter www.ladasifunnel.wordpress.com zu finden.

3.1.3 Unit Testing

Unit Tests werden angewendet, um die funktionalen Module von Programmen zu testen. D.h. sie werden auf korrekte Funktionalität überprüft.

Dazu verwenden wir JUnit-Tests.

JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (Klassen oder Methoden) geeignet ist.

Es wird

Technique Objective:	Die Funktionalität des Programmcodes soll auf Korrektheit überprüft werden.
Technique:	Für jedes überprüfbare Modul werden Tests geschrieben, welche die erwartete Ausgabe enthalten.
Oracles:	Für einzelne Module werden JUnit-Klassen erstellt.
Required Tools:	<ul style="list-style-type: none"> • GitHub • JUnit • Eclipse
Success Criteria:	Wenn die erwartete Ausgabe gleich der wirklichen Ausgabe ist, wurde der Test erfolgreich durchgeführt.

4. Deliverables

4.1 Test Evaluation Summaries

Die Informationen über die Tests werden auf coveralls.io gesammelt.

Dort sind die Informationen aufgeschlüsselt und gut einsehbar.

FILES

SEARCH:

ALL 42

CHANGED 9

SOURCE CHANGED 1

COVERAGE CHANGED 8

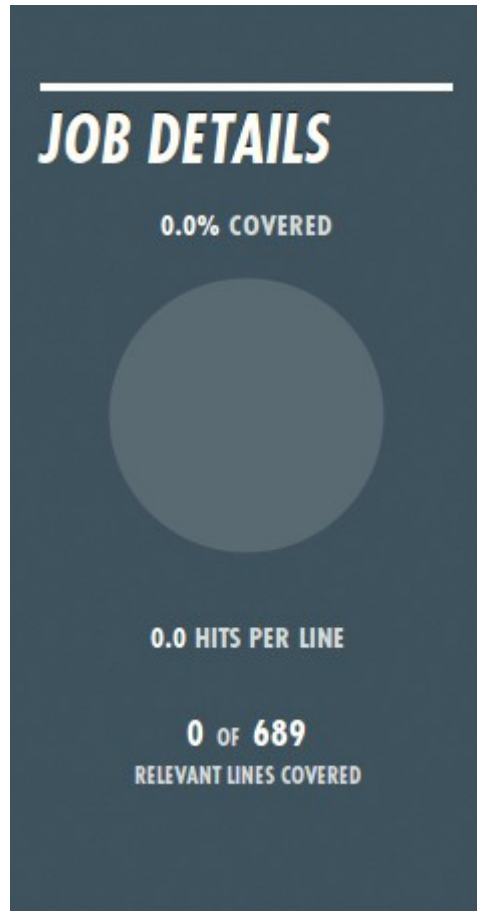
SHOW 10

ENTRIES

COVERAGE	FILE	LINES	RELEVANT	COVERED	MISSED	HITS/LINE
<div><div></div>0.0</div>	.../de/dhbw/socialfunnel/security/RootController.java	32	6	0	6	0.0
<div><div></div>0.0</div>	...hbw/socialfunnel/security/SocialConfiguration.java	85	12	0	13	0.0
<div><div></div>0.0</div>	.../de/dhbw/socialfunnel/security/SocialFunnelUI.java	78	28	0	30	0.0
<div><div></div>0.0</div>	...main/java/de/dhbw/socialfunnel/view/LoginView.java	69	20	0	22	0.0
<div><div></div>0.0</div>	...a/de/dhbw/socialfunnel/view/NetworkChangeView.java	59	10	0	19	0.0
<div><div></div>0.0</div>	...de/dhbw/socialfunnel/view/component/LoginForm.java	115	34	0	46	0.0
<div><div></div>0.0</div>	...e/dhbw/socialfunnel/view/component/SC_MenuBar.java	41	11	0	20	0.0
<div><div></div>0.0</div>	...dhbw/socialfunnel/view/component/RegisterForm.java	161	75	0	88	0.0
<div><div></div>0.0</div>	...dhbw/socialfunnel/view/component/LogoutWindow.java	53	23	0	25	0.0

SHOWING 1 TO 9 OF 9 ENTRIES

4.2 Reporting on Test Coverage



5. Testing Workflow

Unter 3.Test Approach ist der Workflow bereits aufgeführt.

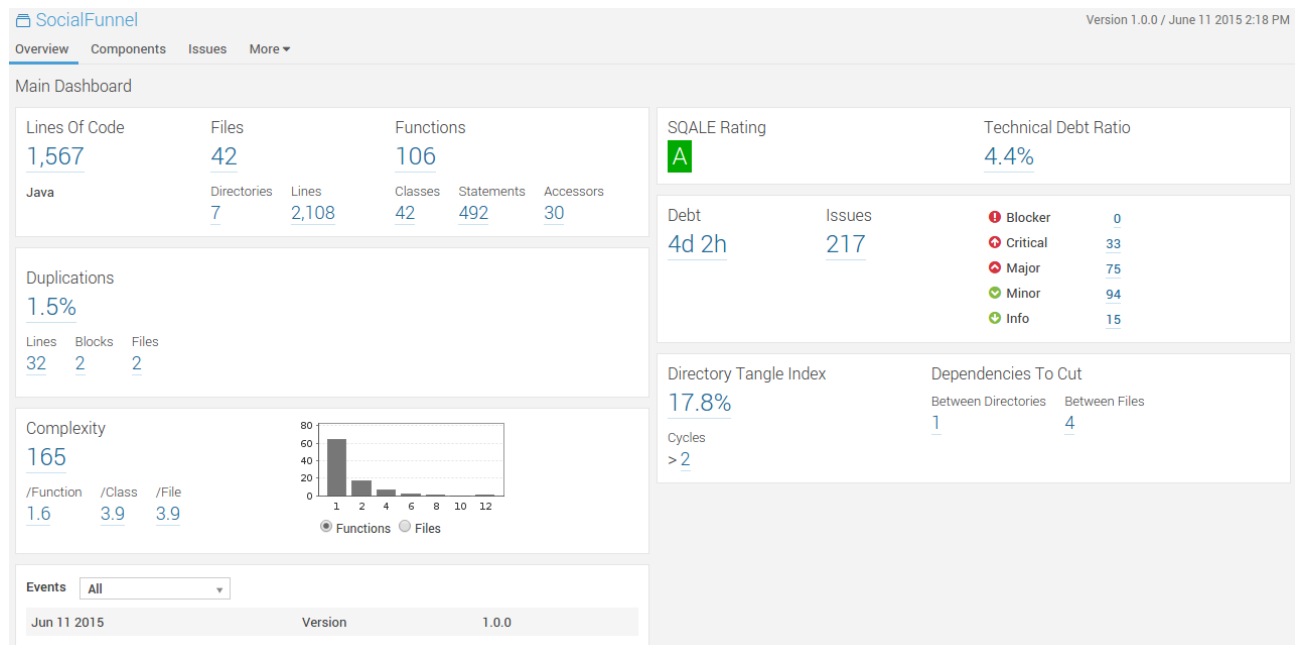
6. Metrics

6.1 Metrics Reporting

Für die Analyse wird SonarQube verwendet.

```
[INFO] [14:19:14.756] Analysis reports sent to server in 630ms
[INFO] [14:19:14.756] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/
dashboard/index/de.dhbw.se:SocialFunnel
[INFO] [14:19:14.756] Note that you will be able to access the updated dashboard
once the server has processed the submitted analysis report.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:16 min
[INFO] Finished at: 2015-06-11T14:19:14+02:00
[INFO] Final Memory: 31M/361M
[INFO] -----
```

Ergebnisseite:



6.2 Integration of Metrics Analysis in Workflow

Der SonarQube Report wird nach jedem Build auf dem lokalen Rechner ausgeführt.

6.3 Definition of 2 Metrics

1. Duplicate Code:

Hier werden Code-Duplikate identifiziert.

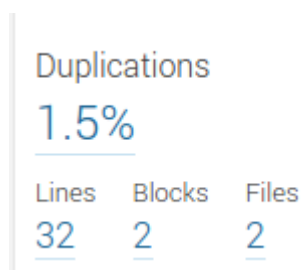
2. Complexity:

Beschreibt die Komplexität des Codes. Um diese Maßzahl zu bestimmen werden Schlüsselwörter, wie IF, FOR, CATCH, THROW, RETURN, WHILE, && und || je Klasse oder Methode gezählt.

Hierbei wird eine kleine Zahl als positiv angesehen, da es bei erhöhter Komplexität für Außenstehende schwerer zu verstehen ist.

6.4 Refactoring

Before (Duplicates):

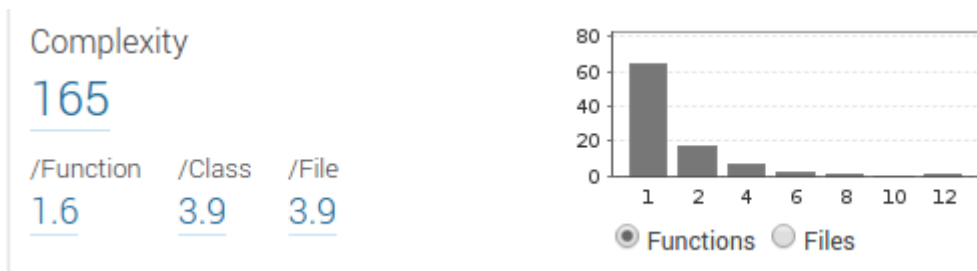


```

16      public final static String VIEW_NAME = "error";
17
18      @PostConstruct
19      void init() {
20          setSizeFull();
21          setMargin(true);
22          setSpacing(true);
23
24          addComponent(new Label("Error"));
25      }
26
27      @Override
28      public void enter(ViewChangeEvent event) {
29          // TODO Auto-generated method stub
30
31      }

```

Before (Complexity):



```

22 dbet...      @Override
23 simo...      public UserDetails loadUserByUsername(String username)
24              throws UsernameNotFoundException {
25
26 dbet...          List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
27 dbet...          authorities.add(new SimpleGrantedAuthority("CLIENT"));
28 simo...          // return new User("guest", "password", true, true, false, false,
29 dbet...          // authorities);

```

This block of commented-out lines of code should be removed. [...](#) 3 hours ago L29 [...](#)

Major Open Not assigned Not planned 5min debt misra, unused

```

30 dbet...          // fetch user from e.g. DB
31 dbet...          // hier frage an Datenbank nach user
32          // wenn kein user, exception schmeissen
33
34 simo...          String[] userinfo = new String[] { null, null };
35          de.dhbw.socialfunnel.model.User temp = userDao.findByEmail(username);
36          userinfo[0] = temp.getEmail();
37          userinfo[1] = temp.getPassword();
38 dbet...          // userinfo = new DBHelper().getUserByName(username);

```

After Refactoring:

