Technische Universität Braunschweig

ISF

Das Grosse SQL-Spiel

The SQL-Alchemist

Software-Entwicklungspraktikum (SEP)
Sommersemester 2015

Testprotokolle

Auftraggeber:
Technische Universität Braunschweig
Institut für Informationssysteme
Prof. Dr. Wolf-Tilo Balke
Mühlenpfordtstraße 23, 2.OG
D-38106 Braunschweig

Betreuer: Jan-Christoph Kalo

Auftragnehmer:

Name	E-Mail-Adresse
Gabriel Ahlers	g.ahlers@tu-braunschweig.de
Majid Dashtiepielehroud	m.dashtiepielehroud@tu-braunschweig.de
Ronja Friebe	r.friebe@tu-braunschweig.de
Stefan Hanisch	stefan.hanisch@tu-braunschweig.de
Fabio Luigi Mazzone	f.mazzone@tu-braunschweig.de
Nicole Naczk	n.naczk@tu-braunschweig.de
Denis Nagel	denis.nagel@tu-braunschweig.de
Luca Porcello	l.porcello@tu-braunschweig.de
Christian Reineke	c.reineke@tu-braunschweig.de
Christian Sander	christian.sander@tu-braunschweig.de
Carl Schiller	c.schiller@tu-braunschweig.de
Levent Muzaffer Üner	l.uener@tu-braunschweig.de
Sören van der Wall	s.van-der-wall@tu-braunschweig.de
Daniel Wolfram	d.wolfram@tu-braunschweig.de

Braunschweig, 16. Juli 2015

Inhaltsverzeichnis

1 Testdurchführung (2015-07-10)

Art des Tests: Integrationstest
Testfall: T700
Abgedeckte Funktionen: **F20** Beteiligte Tester: Denis Nagel

1.1 Testumgebung

Die Funktion wurde unter Windows 8.1 auf einem Webserver getestet. Es wurde eine deutsche Systemumgebung verwendet. Die Applikation wurde über Firefox 39.0 gestartet.

1.2 Testprotokoll

Funktion	***F20***
Tester	*Denis Nagel*
Eingaben	*Username: Testnutzer, E-Mail-Adresse: testnutzer@local.de, Passwort: password1234*
Soll - Reaktion	*Die eingegebenen Daten werden erkannt und der Nutzer wird angemeldet, sowie zum Hauptmenü weitergeleitet.*
Ist – Reaktion	*Der Nutzer wurde erfolgreich angemeldet und ins Hauptmenü weitergeleitet.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F20***
Tester	*Denis Nagel*
Eingaben	*Username: FalscherNutzer, E-Mail-Adresse: testnutzer@local.de, Passwort: password1234*
Soll - Reaktion	*Es wird dem Nutzer mitgeteilt, dass die eingegebenen Daten nicht korrekt sind.*
Ist – Reaktion	*Die korrekte Meldung wurde ausgegeben.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F20***
Tester	*Denis Nagel*
Eingaben	*Username: Testnutzer, E-Mail-Adresse: falschernutzer@local.de, Passwort: password1234*
Soll - Reaktion	*Es wird dem Nutzer mitgeteilt, dass die eingegebenen Daten nicht korrekt sind.*
Ist – Reaktion	*Die korrekte Meldung wurde ausgegeben.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F20***
Tester	*Denis Nagel*
Eingaben	*Username: Testnutzer, E-Mail-Adresse: testnutzer@local.de, Passwort: wrongpassword*
Soll - Reaktion	*Es wird dem Nutzer mitgeteilt, dass die eingegebenen Daten nicht korrekt sind.*
Ist – Reaktion	*Die korrekte Meldung wurde ausgegeben.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

1.3 Zusammenfassung

Der Testlauf hat gezeigt, dass die Anmeldung von Nutzern wie gewünscht funktioniert.

2 Testdurchführung (2015-07-13)

Art des Tests: Integrationstest
Testfall: T700 Abgedeckte Funktionen: **F30** Beteiligte Tester: Denis Nagel

2.1 Testumgebung

Die Funktion wurde unter Windows 8.1 auf einem Webserver getestet. Es wurde eine deutsche Systemumgebung verwendet. Die Applikation wurde über Firefox 39.0 gestartet.

2.2 Testprotokoll

Funktion	***F20***
Tester	*Denis Nagel*
Eingaben	*Der User klickt auf den Button "Logout".*
Soll - Reaktion	*Die Applikation beendet die UserSession und der Nutzer wird ausgeloggt, wobei er zum Login-Bildschirm weitergeleitet wird.*
Ist – Reaktion	*Der Nutzer wurde erfolgreich abgemeldet und zum Login-Bildschirm weitergeleitet.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

2.3 Zusammenfassung

Der Testlauf hat gezeigt, dass die Abmeldung von Nutzern wie gewünscht funktioniert.

3 Testdurchführung (2015-07-13)

Art des Tests: Integrationstest
Testfall: T800
Abgedeckte Funktionen: **F130** Beteiligte Tester: Fabio Mazzone & Gabriel Ahlers

3.1 Testumgebung

Die Funktion wurde unter Ubuntu 14.04 LTS auf einem Webserver getestet. Es wurde eine deutsche Systemumgebung verwendet. Die Applikation wurde im Chromium Web Browser (Version: 43.0.2357.81) gestartet.

3.2 Testprotokoll

Funktion	***F130***
Tester	*Fabio Mazzone & Gabriel Ahlers*
Eingaben	*Im SQL-Trainer (Funktionsweise in allen Modi gleich) gibt der Benutzer ein korrektes Statement als Antwort auf eine Aufgabe*
Soll - Reaktion	*Dem Front-End soll mitgeteilt werden, dass die Lösung korrekt ist. Außerdem soll im Profil gespeichert werden, dass die Aufgabe gelöst wurde*
Ist – Reaktion	*Es wurde übergeben, dass das die Lösung korrekt ist. Die Markierung wurde ebenfalls angelegt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F130***
Tester	*Fabio Mazzone & Gabriel Ahlers*
Eingaben	*Im SQL-Trainer (Funktionsweise in allen Modi gleich) gibt der Benutzer ein leeres Statement als Antwort auf eine Aufgabe*
Soll - Reaktion	*Dem Front-End soll mitgeteilt werden, dass die Lösung nicht korrekt ist.*
Ist – Reaktion	*Es wurde übergeben, dass die Lösung nicht korrekt ist.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F130***
Tester	*Fabio Mazzone & Gabriel Ahlers*
Eingaben	*Im SQL-Trainer (Funktionsweise in allen Modi gleich) gibt der Benutzer ein leeres Statement als Antwort auf eine Aufgabe*
Soll - Reaktion	*Der Server soll die Anfrage sofort verwerfen und dies dem Front-End mitteilen.*
Ist – Reaktion	*Das Front-End überprüft bereits, ob die Antwort leer ist, somit erreicht die Anfrage den Server nicht und die Funktion wird nicht aufgerufen.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

3.3 Zusammenfassung

Der Testlauf hat gezeigt, dass die Funktionen des SQL-Trainers wie gewünscht funktionieren.

4 Testdurchführung (2015-07-07)

Art des Tests: Integrationstest
Testfall: T900
Abgedeckte Funktionen: **F160** Beteiligte Tester: Denis Nagel

4.1 Testumgebung

Die Funktion wurde unter Windows 8.1 auf einem Webserver getestet. Es wurde eine deutsche Systemumgebung verwendet. Die Applikation wurde über Firefox 39.0 gestartet.

4.2 Testprotokoll

Funktion	***F160***
Tester	*Denis Nagel*
Eingaben	*Man wählt den Button "Punkteïm Abschnitt "Rankingsän.*
Soll - Reaktion	*Es wird eine absteigend sortierte Liste der 10 Spieler mit den meisten erspielten Punkten, sowie die Position des aktuellen Nutzers angezeigt.*
Ist – Reaktion	*Es wurde eine absteigend sortierte Liste der 10 Spieler mit den meisten erspielten Punkten, sowie die Position des aktuellen Nutzers angezeigt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F160***
Tester	*Denis Nagel*
Eingaben	*Man wählt den Button "Lofi-Coinsïm Abschnitt "Rankingsän.*
Soll - Reaktion	*Es wird eine absteigend sortierte Liste der 10 Spieler mit den meisten eingesammelten Lofi-Coins, sowie die Position des aktuellen Nutzers angezeigt.*
Ist – Reaktion	*Es wurde eine absteigend sortierte Liste der 10 Spieler mit den meisten eingesammelten Lofi-Coins, sowie die Position des aktuellen Nutzers angezeigt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F160***
Tester	*Denis Nagel*
Eingaben	*Man wählt den Button SZeitïm Abschnitt "Rankingsän.*
Soll - Reaktion	*Es wird eine absteigend sortierte Liste der 10 Spieler mit der längsten Spielzeit, sowie die Position des aktuellen Nutzers angezeigt.*
Ist – Reaktion	*Es wurde eine absteigend sortierte Liste der 10 Spieler mit der längsten Spielzeit, sowie die Position des aktuellen Nutzers angezeigt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F160***
Tester	*Denis Nagel*
Eingaben	*Man wählt den Button "Durchläufeïm Abschnitt "Rankingsän.*
Soll - Reaktion	*Es wird eine aufsteigend sortierte Liste der 10 Spieler mit der geringsten Anzahl an Durchläufen, sowie die Position des aktuellen Nutzers angezeigt.*
Ist – Reaktion	*Es wurde eine aufsteigend sortierte Liste der 10 Spieler mit den wenigsten Durchläufen, sowie die Position des aktuellen Nutzers angezeigt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F160***
Tester	*Denis Nagel*
Eingaben	*Man wählt den Button SSQL-Statementsïm Abschnitt "Rankings-än.*
Soll - Reaktion	*Es wird eine absteigend sortierte Liste der 10 Spieler mit den meisten gelösten SQL-Statements, sowie die Position des aktuellen Nutzers angezeigt.*
Ist – Reaktion	*Es wurde eine absteigend sortierte Liste der 10 Spieler mit den meisten gelösten SQL-Statements, sowie die Position des aktuellen Nutzers angezeigt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Funktion	***F160***
Tester	*Denis Nagel*
Eingaben	*Man wählt den Button Ërfolgsquoteïm Abschnitt "Rankingsän.*
Soll - Reaktion	*Es wird eine absteigend sortierte Liste der 10 Spieler mit der höchsten Erfolgsquote bei der Bearbeitung der SQL-Statements, sowie die Position des aktuellen Nutzers angezeigt.*
Ist – Reaktion	*Es wurde eine absteigend sortierte Liste der 10 Spieler mit der höchsten Erfolgsquote bei der Bearbeitung der SQL-Statements, sowie die Position des aktuellen Nutzers angezeigt.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

4.3 Zusammenfassung

Der Testlauf hat gezeigt, dass alle Funktionen des Ranking-Screens wie gewünscht funktionieren.

5 Testdurchführung (2015-07-10)

Art des Tests: Integrationstest
Testfall: T1100
Abgedeckte Funktionen: **F40** Beteiligte Tester: Stefan Hanisch

5.1 Testumgebung

Die Funktion wurde unter Windows 8.1 auf einem Webserver getestet. Es wurde eine deutsche Systemumgebung verwendet. Die Applikation wurde über Firefox 39.0 gestartet.

5.2 Testprotokoll

Tester	*Stefan Hanisch*
Eingaben	*Der Benutzer ist angemeldet und befindet sich im Hauptmenü und klickt mit der linken Maustaste auf den Button "Profile".*
Soll - Reaktion	*Dem User wird sein Profil mit seiner ID, seinem Benutzernamen, den gesammelten Münzen, der Anzahl der Spielläufe, der gespielten Zeit, den gelösten Statements und der Erfolgsrate angezeigt.).*
Ist – Reaktion	*Die gewünschten Daten des aktuellen Benutzers werden korrekt angezeigt.*
Ergebnis	*Der Test wurde erfolgreich durchgeführt.*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

5.3 Zusammenfassung

Der Testlauf hat gezeigt, dass dem User alle PRofildaten ordnungsgemäß angezeigt werden.

6 Testdurchführung (2015-07-09)

Art des Tests: Unit-Test
Testfall: T1100
Beteiligte Tester: Denis Nagel

6.1 Testumgebung

Die Methoden der Klasse wurden unter Windows 8.1, über IntelliJ und mittels JUnit getestet. Es wurde eine deutsche Systemumgebung verwendet. Die Applikation wurde über Firefox 39.0 gestartet.

6.2 Testprotokoll

Tester	*Denis Nagel*
Eingaben	*@Test* *public void testByScore(){* *boolean testSuccessful = true;* *try {* *Profile profile = Profile.getById("gültige ID");* *List<Profile> testList = Profile.sortByScore(profile);* *Profile testProfile;* *int lowerScore = 0;* *int higherScore = -1;* *int correctOrder;* *for (Object profileObject : list) {* *correctOrder = 0;* *testProfile = (Profile) profileObject;* *if (higherScore == -1) {* *higherScore = testProfile.getTotalScore();* *continue;* *} else {* *lowerScore = testProfile.getTotalScore();* *if (lowerScore <= higherScore) {* *correctOrder = 1;* *} else {* *correctOrder = 0;* *}* *higherScore = lowerScore;* *}* *assertEquals(1, correctOrder);* *}* *} catch (Throwable e) {* *testSuccessful = false;* *}* *assertTrue(testSuccessful);* *Logger.info(SShopItemGetAvatarList successfull");* *}*
Soll - Reaktion	*Die Liste soll absteigend nach Gesamtpunktzahl sortiert werden.*

Ist – Reaktion	*Die Methode hat die Liste der Profile absteigend nach Gesamtpunktzahl sortiert.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Tester	*Denis Nagel*
Eingaben	*@Test* *public void testByTime(){* *boolean testSuccessful = true;* *try {* *Profile profile = Profile.getById("gültige ID");* *List<Profile> testList = Profile.sortByTime(profile);* *Profile testProfile;* *int lessTime = 0;* *int moreTime = -1;* *int correctOrder;* *for (Object profileObject : list) {* *correctOrder = 0;* *testProfile = (Profile) profileObject;* *if (moreTime == -1) {* *moreTime = testProfile.getTime();* *continue;* *} else {* *lessTime = testProfile.getTime();* *if (lessTime <= moreTime) {* *correctOrder = 1;* *} else {* *correctOrder = 0;* *}* *moreTime = lessTime;* *}* *assertEquals(1, correctOrder);* *}* *} catch (Throwable e) {* *testSuccessful = false;* *}* *assertTrue(testSuccessful);* *Logger.info(SShopItemGetAvatarList successfull");* *}*
Soll - Reaktion	*Die Liste soll absteigend nach gespielter Zeit sortiert werden.*
Ist – Reaktion	*Die Methode hat die Liste der Profile absteigend nach gespielter Zeit sortiert.*

Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Tester	*Denis Nagel*
Eingaben	*@Test* *public void testByRuns(){* *boolean testSuccessful = true;* *try {* *Profile profile = Profile.getById("gültige ID");* *List<Profile> testList = Profile.sortByRuns(profile);* *Profile testProfile;* *int lessRuns = 0;* *int moreRuns = -1;* *int correctOrder;* *for (Object profileObject : list) {* *correctOrder = 0;* *testProfile = (Profile) profileObject;* *if (moreRuns == -1) {* *moreRuns = testProfile.getRuns();* *continue;* *} else {* *lessRuns = testProfile.getRuns();* *if (lessRuns <= moreRuns) {* *correctOrder = 1;* *} else {* *correctOrder = 0;* *}* *moreRuns = lessRuns;* *}* *assertEquals(1, correctOrder);* *}* *} catch (Throwable e) {* *testSuccessful = false;* *}* *assertTrue(testSuccessful);* *Logger.info(SShopItemGetAvatarList successfull");* *}*
Soll - Reaktion	*Die Liste soll absteigend nach gepielten Durchläufen sortiert werden.*

Ist – Reaktion	*Die Methode hat die Liste der Profile absteigend nach gespielten Durchläufen sortiert.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Tester	*Denis Nagel*
Eingaben	*@Test* *public void testBySQL(){* *boolean testSuccessful = true;* *try {* *Profile profile = Profile.getById("gültige ID");* *List<Profile> testList = Profile.sortByScore(profile);* *Profile testProfile;* *int lessSQL = 0;* *int moreSQL = -1;* *int correctOrder;* *for (Object profileObject : list) {* *correctOrder = 0;* *testProfile = (Profile) profileObject;* *if (moreSQL == -1) {* *moreSQL = testProfile.getSQL();* *continue;* *} else {* *lessSQL = testProfile.getSQL();* *if (lessSQL <= moreSQL) {* *correctOrder = 1;* *} else {* *correctOrder = 0;* *}* *moreSQL = lessSQL;* *}* *assertEquals(1, correctOrder);* *}* *} catch (Throwable e) {* *testSuccessful = false;* *}* *assertTrue(testSuccessful);* *Logger.info(SShopItemGetAvatarList successfull");* *}*
Soll - Reaktion	*Die Liste soll absteigend nach gelösten SQL-Statements sortiert werden.*

Ist – Reaktion	*Die Methode hat die Liste der Profile absteigend nach gelösten SQL-Statements sortiert.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Tester	*Denis Nagel*
Eingaben	*@Test* *public void testByRate(){* *boolean testSuccessful = true;* *try {* *Profile profile = Profile.getById("gültige ID");* *List<Profile> testList = Profile.sortByScore(profile);* *Profile testProfile;* *int lowerRate = 0;* *int higherRate = -1;* *int correctOrder;* *for (Object profileObject : list) {* *correctOrder = 0;* *testProfile = (Profile) profileObject;* *if (higherRate == -1) {* *higherRate = testProfile.getRate();* *continue;* *} else {* *lowerRate = testProfile.getRate();* *if (lowerRate <= higherRate) {* *correctOrder = 1;* *} else {* *correctOrder = 0;* *}* *higherRate = lowerRate;* *}* *assertEquals(1, correctOrder);* *}* *} catch (Throwable e) {* *testSuccessful = false;* *}* *assertTrue(testSuccessful);* *Logger.info(SShopItemGetAvatarList successfull");* *}*
Soll - Reaktion	*Die Liste soll absteigend nach der Erfolgsquote beim Lösen der SQL-Statements sortiert werden.*

Ist – Reaktion	*Die Methode hat die Liste der Profile absteigend nach der Erfolgsquote beim Lösen der SQL-Statements sortiert.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

Tester	*Denis Nagel*
Eingaben	*@Test* *public void testByCoins(){* *boolean testSuccessful = true;* *try {* *Profile profile = Profile.getById("gültige ID");* *List<Profile> testList = Profile.sortByCoins(profile);* *Profile testProfile;* *int lessCoins = 0;* *int moreCoins = -1;* *int correctOrder;* *for (Object profileObject : list) {* *correctOrder = 0;* *testProfile = (Profile) profileObject;* *if (moreCoins == -1) {* *moreCoins = testProfile.getCoins();* *continue;* *} else {* *lessCoins = testProfile.getCoins();* *if (lessCoins <= moreCoins) {* *correctOrder = 1;* *} else {* *correctOrder = 0;* *}* *moreCoins = lessCoins;* *}* *assertEquals(1, correctOrder);* *}* *} catch (Throwable e) {* *testSuccessful = false;* *}* *assertTrue(testSuccessful);* *Logger.info(SShopItemGetAvatarList successfull");* *}*
Soll - Reaktion	*Die Liste soll absteigend nach der Anzahl der gesammelten Lofi-Coins sortiert werden.*

Ist – Reaktion	*Die Methode hat die Liste der Profile absteigend nach der Anzahl der gesammelten Lofi-Coins sortiert.*
Ergebnis	*Der Testlauf ist erfolgreich abgeschlossen worden.*
Unvorhergesehene Ereignisse	
Nacharbeiten	

6.3 Zusammenfassung

Der Testlauf hat gezeigt, dass die Ranking-Methoden der Klasse "Profile"wie gewünscht funktionieren.

7 Testdurchführung (2015-07-12)

Art des Tests: Integrationstest
Testfall: T1100
Beteiligte Tester: Fabio Mazzone & Gabriel Ahlers

Folgende Methode wird zum initializieren der Testumgebung vor jedem der Tests durchgeführt:

```
@BeforeClass
public static void app(){
  FakeApplication fakeApplication = fakeApplication(inMemoryDatabase("test"));
  start(fakeApplication);
  Avatar.init();¸
  ShopItem.init();
}
```

7.1 Tesprotokoll

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testUserCreate(){
	boolean testSuccessful= true;
	try {
	User.create(“UserCreate“, “UserCreate@local.de“, “empty“);
	}
	catch (UsernameTakenException | EmailTakenException e) {
	testSuccessful = false;
	}
	
	assertTrue(testSuccessful);
	Logger.info(“UserCreate successfull“);
	}

Soll - Reaktion	*Konsole zeigt Meldung: “UserCreate successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “UserCreate successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test(expected = UsernameTakenException.class)
	public void testUserCreateNameTaken() throws EmailTakenException, UsernameTakenException {
	try {
	User.create(“UserTaken“, “UserTaken1@local.de“, “empty“);
	User.create(“UserTaken“, “UserTaken2@local.de“, “empty“);
	} catch (UsernameTakenException | EmailTakenException e) {
	Logger.info(“UserNameTaken successfull“);
	throw e;
	}
	
	}

Soll - Reaktion	*Konsole zeigt Meldung: “UserNameTaken successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “UserNameTaken successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test(expected = EmailTakenException.class)
	public void testUserCreateEmailTaken() throws EmailTakenException, UsernameTakenException {
	try {
	User.create(“EmailTaken1“, “EmailTaken@local.de“, “empty“);
	User.create(“EmailTaken2“, “EmailTaken@local.de“, “empty“);
	} catch (UsernameTakenException | EmailTakenException e) {
	Logger.info(“EmailNameTaken successfull“);
	throw e;
	}
	}

Soll - Reaktion	*Konsole zeigt Meldung: “EmailTaken successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “EmailTaken successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

7.2 Zusammenfassung

Der Testlauf ergab keine Fehler.

8 Testdurchführung (2015-07-12)

Art des Tests: Integrationstest
Testfall: T1100
Beteiligte Tester: Fabio Mazzone & Gabriel Ahlers

Folgende Methode wird zum initialisieren der Testumgebung vor jedem der Tests durchgeführt:

```
@BeforeClass
public static void app(){
  FakeApplication fakeApplication = fakeApplication(inMemoryDatabase(“test“));
  start(fakeApplication);
}
```

8.1 Testprotokoll

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testPotionCreate(){
	Potion potion = Potion.create(“TestPotion“,1 , 3, 4);
	
	assertTrue(potion != null);
	Logger.info("PotionCreate successfull");
	}

Soll - Reaktion	*Konsole zeigt Meldung: “PotionCreate successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “PotionCreate successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testPotionCreateDouble() {
	Potion Potion1 = Potion.create(“TestPotion1“,1 , 3, 4);
	Potion Potion2 = Potion.create(“TestPotion1“,1 , 3, 4);
	
	assertTrue(Potion1 == Potion2);
	Logger.info(“PotionCreateDouble successfull“);
	}

Soll - Reaktion	*Konsole zeigt Meldung: “PotionCreateDouble successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “PotionCreateDouble successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

8.2 Zusammenfassung

Der Testlauf ergab keine Fehler.

9 Testdurchführung (2015-07-12)

Art des Tests: Integrationstest
Testfall: T1100
Beteiligte Tester: Fabio Mazzone & Gabriel Ahlers

Folgende Methode wird zum initializieren der Testumgebung vor jedem der Tests durchgeführt:

```
@BeforeClass
public static void app(){
  FakeApplication fakeApplication = fakeApplication(inMemoryDatabase("test"));
  start(fakeApplication);
  Avatar.init();¸
  ShopItem.init();
}
```

9.1 Testprotokoll

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testMapCreate(){
	Map map = Map.create(2, “path“, true);
	
	assertTrue(map != null);
	Logger.info("MapCreate successfull");
	}

Soll - Reaktion	*Konsole zeigt Meldung: “MapCreate successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “MapCreate successfull“*
Ergebnis	*Test erfolgreich durchgeführt*

Unvorhergesehene Ereignisse	–
Nacharbeiten	–

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testMapCreateDouble() {
	Map map1 = Map.create(3, “path“, true);
	Map map2 = Map.create(3, “path“, true);
	
	assertTrue(map1 == map2);
	Logger.info(“MapCreateDouble successfull“);
	}

Soll - Reaktion	*Konsole zeigt Meldung: “MapCreateDouble successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “MapCreateDouble successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

9.2 Zusammenfassung

Der Testlauf ergab keine Fehler.

10 Testdurchführung (2015-07-12)

Art des Tests: Integrationstest
Testfall: T1100
Beteiligte Tester: Fabio Mazzone & Gabriel Ahlers

Folgende Methode wird zum initialisieren der Testumgebung vor jedem der Tests durchgeführt:

```
@BeforeClass
public static void app(){
  FakeApplication fakeApplication = fakeApplication(inMemoryDatabase("test"));
  start(fakeApplication);
  Potion.init();
}
```

10.1 Testprotokoll

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testInventoryCreate(){
	Profile profile = null;
	try {
	Profile profile = User.create(ÜserCreate", “UserCreate@local.de“, “empty“).profile;
	} catch (UsernameTakenException | EmailTakenException e) {
	e.printStackTrace();
	}
	Potion potion = Potion.getById(1L);
	Inventory inv = Inventory.create(profile, potion);
	assertTrue(inv != null);
	Logger.info(“InventoryCreate successfull“);
	}

Soll - Reaktion	*Konsole zeigt Meldung: “InventoryCreate successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “InventoryCreate successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

10.2 Zusammenfassung

Der Testlauf ergab keine Fehler.

11 Testdurchführung (2015-07-12)

Art des Tests: Integrationstest
Testfall: T1100
Beteiligte Tester: Fabio Mazzone & Gabriel Ahlers

Folgende Methode wird zum initializieren der Testumgebung vor jedem der Tests durchgeführt:

```
@BeforeClass
public static void app(){
  FakeApplication fakeApplication = fakeApplication(inMemoryDatabase(“test“));
  start(fakeApplication);
}
```

11.1 Testprotokoll

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testAvatarCreate(){
	Avatar avatar = Avatar.create(“Test Avatar“, “desc“, “filename“, “soundurl“, false, 2, 3, 3, 4, 5);
	
	assertTrue(avatar != null);
	
	Log-
	ger.info(ÄvatarCreate successfull");
	}

Soll - Reaktion	*Konsole zeigt Meldung: “AvatarCreate successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “AvatarCreate successfull“*
Ergebnis	*Test erfolgreich durchgeführt*

Unvorhergesehene Ereignisse	–
Nacharbeiten	–

Tester	*Gabriel Ahlers & Fabio Mazzone*

Eingaben	@Test
	public void testAvatarCreateDouble() {
	Avatar avatar1 = Avatar.create(“Test Avatar“, “desc“, “filename“, “soundurl“, false, 2, 3, 3, 4, 5);
	Avatar avatar2 = Avatar.create(“Test Avatar“, “desc“, “filename“, “soundurl“, false, 2, 3, 3, 4, 5);
	
	assertTrue(avatar1 == avatar2);
	Logger.info(“AvatarCreateDouble successfull“);
	}

Soll - Reaktion	*Konsole zeigt Meldung: “AvatarCreateDouble successfull“*
Ist – Reaktion	*Konsole zeigt Meldung: “AvatarCreateDouble successfull“*
Ergebnis	*Test erfolgreich durchgeführt*
Unvorhergesehene Ereignisse	–
Nacharbeiten	–

11.2 Zusammenfassung

Der Testlauf ergab keine Fehler.