

# **Projektdefinition sqrr1**

**Web-Applikation zum Üben und automatisierten Korrigieren von SQL**

Alexander-Michael Kühnle      Matthias Michael Döpmann  
Mark Umnus

19. Juni 2019

# Inhaltsverzeichnis

<b>1</b>	<b>Planung</b>	<b>4</b>
1.1	Anforderungen . . . . .	4
1.1.1	Funktionale Anforderungen . . . . .	4
1.1.2	Nichtfunktionale Anforderungen . . . . .	4
1.2	Termine . . . . .	4
<b>2</b>	<b>Umsetzung</b>	<b>6</b>
2.1	Verwendete Technologien . . . . .	6
2.2	Liefergegenstände . . . . .	6
2.3	Datenmodell . . . . .	6
<b>A</b>	<b>Protokolle</b>	<b>9</b>

## **Erläuterungen**

Fragen an Betreuer

Fragen an uns

## **Einführung**

Dieses Dokument dient der Planung und Durchführung des im Titel genannten Projektes im Rahmen des Moduls „*Softwareentwicklungsprojekt I*“. Im Kapitel 1.1 werden die Anforderungen aufgezählt, die an dieses Projekt gestellt werden. Kapitel 1.2 zeigt den initial erstellten Zeitplan.

# 1 Planung

## 1.1 Anforderungen

### 1.1.1 Funktionale Anforderungen

- Kontenverwaltung von Studierenden und Dozierenden
- Dozierende können Aufgaben erstellen
- Studierende können Aufgaben bearbeiten
  - Aufgabenstellung in natürlicher Sprache
  - Entgegennahme von SQL-Anfragen
  - Anzeige des Ergebnisses der Anfrage (auf zufälliger/ Beispieltabelle)
  - Erkennung der Korrektheit der Anfrage bezüglich der gestellten Frage
  - bei Unsicherheit → Ersteller informieren
- freies Üben
- Studierende sollen virtuelle Abzeichen erwerben können → von Dozenten erstellt
- Adminaccount zur Benutzerverwaltung? → Nein; wird von Dozenten miterledigt

### 1.1.2 Nichtfunktionale Anforderungen

**Einfachheit** Das Programm soll durch Benutzer ohne das Lesen einer Anleitung bedienbar sein.

**Wartbarkeit** Das Programm soll auch für projektexterne Entwickler verständlich, wartbar und erweiterbar sein.

**Qualität** Das Programm soll durch Tests grundlegenden Qualitätsansprüchen genügen.

**Freiheit** Das Programm soll unter Linux funktionieren und möglichst nur auf freie/offene Software zurückgreifen.

## 1.2 Termine

## 1 Planung

Tabelle 1.1: Zeitplan





Datum	Ziele
01.05.2019	Technologien bestimmt
08.05.2019	Technologien installiert und lauffähig; Zeit- und Architekturplan erstellt
15.05.2019	Datenmodell ausgearbeitet; GUI auf Papier designed
22.05.2019	Rails Sicherheitsguide und OWASP überflogen; obligatorischen Teil des Datenmodells umgesetzt; Verbindung zwischen Front- und Backend hergestellt
29.05.2019	Benutzerregistrierung ermöglicht; Prototypen erstellt; Vortragsfolien erstellt; Konzept für automatische Kontrolle erarbeitet
05.06.2019	Dozentenvue; Aufgabenstellungen erstellbar; automatische Aufgabenkontrolle getestet
12.06.2019	Studentenvue; Aufgaben lösbar; automatische Aufgabenkontrolle implementiert
19.06.2019	Einstellungsmenü; freies Üben ermöglicht; Tests
26.06.2019	Optionales (wie Leaderboard, Achievements) implementiert
03.07.2019	Puffer; Kernprojekt fertig; kleine Verbesserungen (z. B. Dokumentation)
10.07.2019	Abschlussvortragsfolien erstellt

## 2 Umsetzung

### 2.1 Verwendete Technologien

- Ruby 2.6.3
- Ruby on Rails 5.2.3
- Apache HTTP Server **Version?**
- PostgreSQL **Version?**
- Semantic UI **Version?**
- React **Version?**
- Codemirror

### 2.2 Liefergegenstände

In Tabelle 2.1 auf Seite 7 sind all jene Liefergegenstände aufgeführt, die vom Betreuer bestätigt wurden. Die IDs in dieser Tabelle gehören zu entsprechenden Git-Branches. In der Tabelle sind zudem Status aufgezeigt, die dem Format – »noch nicht gestartet«, – »in Bearbeitung«, – »abgeschlossen« und – »hier liegt ein Fehler vor« folgen.

### 2.3 Datenmodell

Tabelle 2.1: Liefergegenstände

<b>A Benutzerverwaltung</b>			
ID	Liefergegenstand	Erläuterung	Status
A.1	Allgemeines		■
A.1.1	Registrierung		●
A.1.2	Nutzername		●
A.1.3	Passwort		●
A.1.4	E-Mail-Adresse		●
A.1.5	Einloggen		●
A.1.6	Ausloggen		●
A.1.7	Account löschen		■
A.1.8	Passwortwiederherstellung	per E-Mail	□
A.2	Studierendenaccounts		■
A.2.1	Aufgabenliste	Markierung für Bearbeitungsstand	■
A.2.2	Abzeichenliste		□
A.3	Dozentenaccounts		■
A.3.1	Aufgabenverwaltung	CRUD-Operationen	■
A.3.2	Benutzerverwaltung	CRUD-Operationen, Rechtezuweisung	■
A.3.3	Abzeichenverwaltung	CRUD-Operationen	□
<b>B Interface</b>			
ID	Liefergegenstand	Erläuterung	Status
B.1	Einstellungen	Privatsphäre, Benachrichtigungen	□
B.2	Aufgabenübersicht	nach Kategorien geordnet	●
B.2.1	Aufgabenname		●
B.2.2	Statusindikator	neu, begonnen, abgeschlossen	●
B.3	Spielwiese		□
B.4	Leaderboard		□
<b>C Aufgabenbearbeitung</b>			
ID	Liefergegenstand	Erläuterung	Status
C.1	Aufgabenstellung durch Dozenten	in natürlicher Sprache	●
C.1.1	Hinterlegen mind. einer Musterlösung		●
C.2	Eingeben einer Query durch Studenten		■
C.3	Prüfen der Query durch System	mit Timeout	■
C.3.1	Ausarbeiten eines Verfahrens		●
C.3.2	Testen des Verfahrens		■
C.4	Anwenden der Query durch System		■
C.5	Anzeige der Ergebnisse durch System		■
C.6	Meldung an Dozenten durch System	im Fehlerfall	□
C.7	Vergabe von Belohnungen durch System	Punkte, Abzeichen	□

## 2 Umsetzung

Tabelle 2.2: Benutzer

Attribut	Erläuterung
<u>userid</u>	



# A Protokolle

## 01. Mai 2019

### Ziele

- ✓ zu verwendende Technologien bestimmt
- ✓ grundlegenden Zeitplan entworfen

## 08. Mai 2019

### Ziele

- ✓ Programmbibliotheken installiert und eingerichtet
- ✓ Liefergegenstände spezifiziert
- ✓ Zeitplan daran angepasst
- ✓ Zuständigkeiten geklärt

### Verlauf

- Dozenzen = Admins
- Spielwiese approved
- Timeout bei Anfrage setzen
- nach Einloggen Aufgabenliste (möglicherweise nach Kategorien geordnet) anzeigen  
→ mit Häkchen dran
- Leaderboard
- Programm als 1-Page-Anwendung

## 15. Mai 2019

### Ziele

- ✓ Datenmodell ausgearbeitet
- ✓ GUI-Skizze auf Papier erstellt

### Verlauf

- Menü oben oder an der Seite des GUIs
- Speichern der letzten Query pro Aufgabe, aber nur explizit mit einem Knopf

- Kein „Sind Sie sich sicher, dass Sie die Seite ohne Speichern verlassen wollen?“-Pop-up
- Ergebnis von eingereichten Lösungen  $\in \{\text{richtig, falsch, unsicher}\}$
- Einsehen der Ergebnisse durch Betreuer, der bei **unsicher** selbst entscheiden kann
- Hinterlegen der DB-Schemen TPC-H und Uni mit Daten; Ausführen der Anfragen darauf
- Hinterlegen eines „versteckten“ Datensatzes zur zusätzlichen Validierung der Querys
- Abstufen der Priorität von Scoreboard, Abzeichen und Rücksetzen des Passworts
- Speichern von mehreren Musterquerys pro Aufgabe als Dozent
- Überprüfen der Querys z. B. mit Postgres **EXPLAIN**
- Berechnen der Punkte eines Benutzers (nicht speichern) für bessere Konsistenz
- Rechtemanagement auf Anwendungsebene statt DB-Ebene

## **22. Mai 2019**

### **Ziele**

- Prototypen erstellt
  - Aufgabe auswählen
  - Query eintragen
  - Ergebnis wird angezeigt zusammen mit "Richtig!"
- Vortrag ausgearbeitet
  - Aufgabenstellung
  - Vorgehen/Vision
  - Stand
  - Zukunftspläne
  - Vortragsfolien erstellt