Praxisarbeit

Modul 133

kxi – Jetski Services

Dokumentation Technik: IPERKA

Autor: Özgür Isbert

E-Mail: [oezguer.isbert@student.ibz.ch](mailto:oezguer.isbert@student.ibz.ch)

Dozent: Lukas Müller

Table of Contents

[Vorwort 3](#_Toc58921666)

[GitHub 3](#_Toc58921667)

[Voraussetzungen 3](#_Toc58921668)

[IPERKA 4](#_Toc58921669)

[Information 4](#_Toc58921670)

[Planung 4](#_Toc58921671)

[Entscheidung 4](#_Toc58921672)

[Realisierung 5](#_Toc58921673)

[Kontrolle 7](#_Toc58921674)

[Auswertung 7](#_Toc58921675)

# Vorwort

Dies ist eine Fortführung des Moduls 307 und beendet das Modul 133. In diesem Modul geht es um „Web-Applikationen mit Session-Handling realisieren“. Ich habe mir die Aufgabe angeschaut und habe dies via mein MVC-Pattern eigens erstellt und implementiert.

## GitHub

Das Projekt ist auf GitHub unter

<https://github.com/oezguerisbert/modul-133/tree/main/Praxisarbeit>

[](https://github.com/0xFr0zen/modul-133/tree/main/Praxisarbeit)

aufzufinden.   
Nach dem Runterladen der Dateien kann man direkt die Seite starten.

Das Projekt ist automatisiert, das heißt sämtliche Skripte werden im Vorfeld geladen und in der Datenbank abgelegt. Sie müssen beim Start des Skriptes die Datenbankkonfiguration eingeben, welche den Rest dann abklärt.

## Voraussetzungen

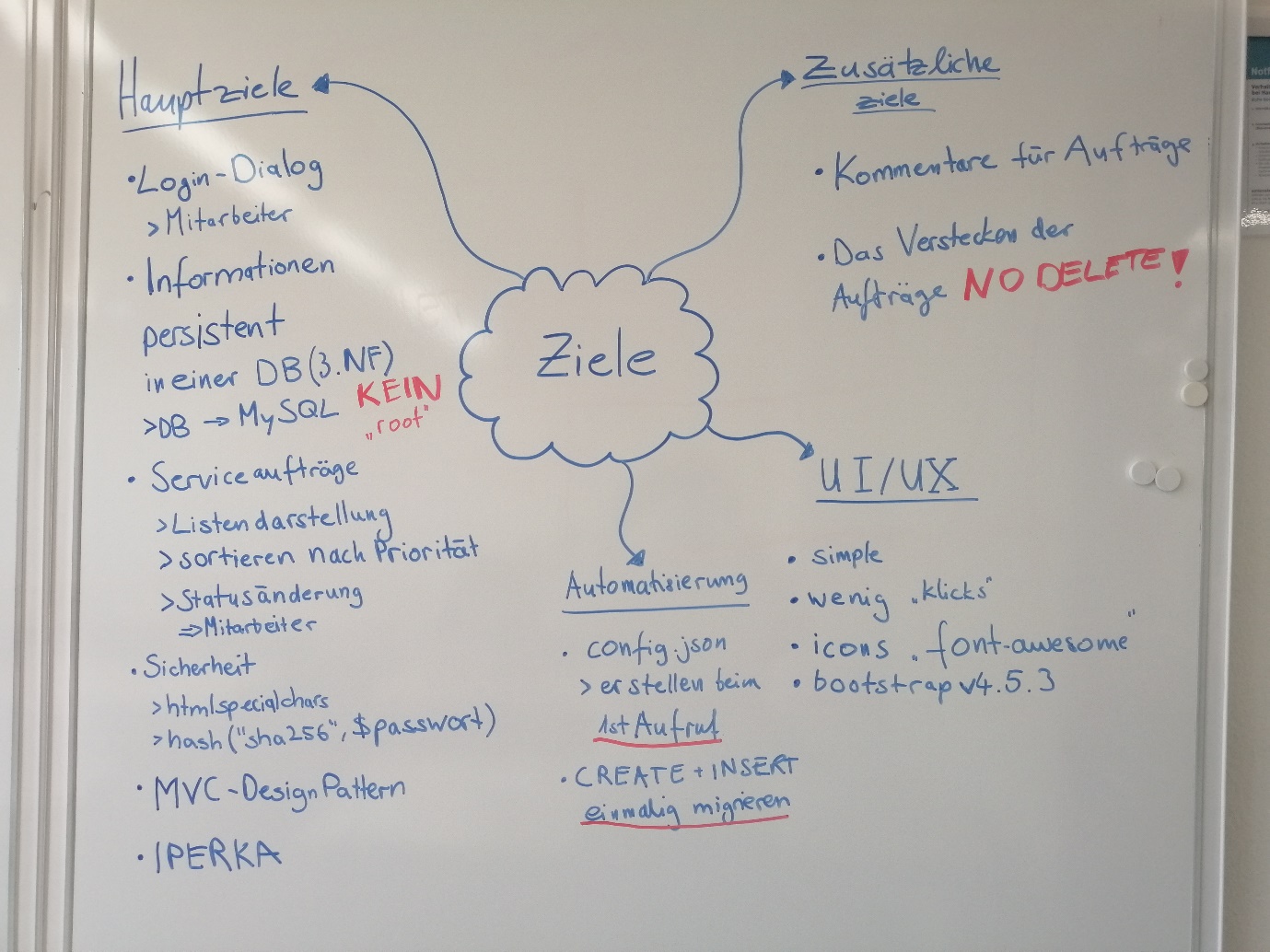
Um dieses Projekt laufen zu lassen müssen bestimme Vorkehrungen vorgenommen werden.

* PHP-Versionen > 7
* MySQL Version >= 5
* MySQL Datenbank mit dem Namen „modul133“.

# IPERKA

## Information

Informationen wurden aus der Aufgabe entnommen. Hier ist ein Mindmap zu den Aufgaben, die erfüllt werden müssen.



## Planung

Für die Planung habe ich das CI-Prinzip (Continuous Integration) übernommen. Hierbei werden die Jeweiligen Funktionen und Implementationen weiterbauend hinzugefügt. Es wird nicht auf eine Version festgelegt, aber dennoch versioniert gearbeitet.

Als Datenstruktur wurde das MVC Pattern übernommen und implementiert. Die Models wurden in dem Unterordner “classes” abgespeichert.

## Entscheidung

Aus den 10 Pflichtoptionen mussten wir 2 auswählen. Ich habe mich für “Kommentare für Aufträge” und “Das Verstecken der Aufträge !NO\_DELETE” entschieden. Begründet mit simplen Änderungen in der Datenbank und Zweckvolles Nutzen der Oberfläche, wurde dieser Entscheid abgehalten.

Zusätzlich habe ich mir die Mühe gemacht und als weiteren Aufwand die Automatisierung der Datenbanksysteme und Konfiguration vorgenommen. Sobald das Projekt gestartet wird auf einem Server und die Seite zum ersten Mal aufgerufen wurde, muss man den Server einstellen. Dies kann entweder direkt über die Oberfläche oder per POST-Methode mit den diversen Einstellungen mitgeliefert werden und das System wird vorkonfiguriert. Die Datenbank soll im Vorfeld migriert werden. Sämtliche SQL-Datenbanktabellen und Fremdschlüssel sollen eingerichtet werden und nicht mehr aktualisiert werden. Bei einem *wipe* d.h ein Reset der Datenbank, muss man nur die „migration“-Option auf true stellen und die Datenbank wird wieder neu aufgesetzt.

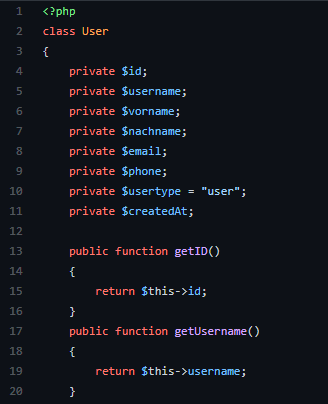
Als weitere Entscheidung habe ich die Icon Sets von der Schriftart „FontAwesome“ übernommen. Diese sind einfache Grafiken, welche die Oberfläche einfacher gestalten. Somit wird dann der Punkt „einfach halten“ eingehalten.

## Realisierung

Ich habe zuvor das Projekt aus dem Modul 307 übernommen, so wie es aufzufinden ist. Dieses kann man unter [GitHub (Modul 307)](https://github.com/oezguerisbert/modul-307/tree/main/Praxisarbeit) finden.

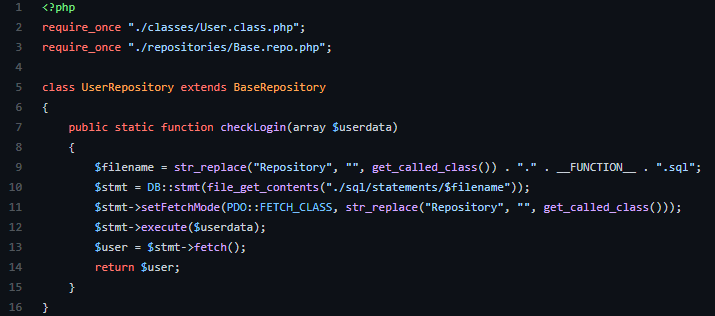
Ich habe für die Umsetzung von MVC (Model-View-Controller) die Ordnerstruktur erneuert und die Ordner „classes“, „controllers“, „repositories“, „sql“ und „views“ erstellt. In diesen Ordnern sind die jeweiligen korrespondierenden Code-Snippets vorhanden.

Beispiel: Die Klasse User



Die Werte werden später über die Repositories beladen. Repositories sind die Schnittstellen von den Klassen zu der Datenbank.

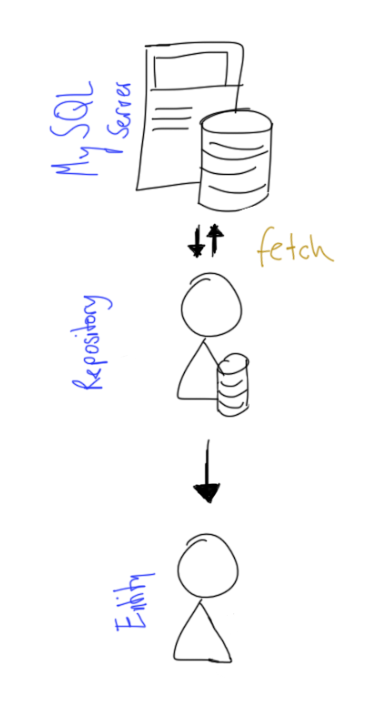
Beispiel: Die Repository UserRepository



Die BaseRepository:

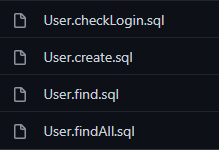


Der Sinn hinter diesem Code ist das ich die SQL-Queries in seperate Dateien schreiben kann und die Logik vom Command trennen kann. Die Methode findAll() ist im Grunde automatisiert. Die Klasse und die SQL-Dateien, die diese Methode aufrufen tut, werden dazu benötigt.



Dabei habe ich die Namenskonvention: **<Klasse>.<Methode>.sql** genommen.

Hier ein Ausschnitt der Dateien im „sql“-Ordner:



## Kontrolle

## Auswertung