

Online-Verwaltungstool für Mehrfamilienhaus

Aufgabe B



Projektarbeit Web-Engineering

Publikation: 2017

Autoren Kevin Trottmann, Matej Djuranec, Mikayil Bekooglu

Fachhochschule Nordwestschweiz

**Inhaltsverzeichnis**

[1 Aufgabe 4](#_Toc438671349)

[1.1 Ausgangslage 4](#_Toc438671350)

[1.2 Ziele 4](#_Toc438671351)

[2 Projektmanagement 5](#_Toc438671352)

[2.1 Planung (Gantt) 5](#_Toc438671353)

[2.2 Organisation 6](#_Toc438671354)

[2.2.1 Team 6](#_Toc438671355)

[2.2.2 Kommunikation 6](#_Toc438671356)

[2.2.3 Tools 6](#_Toc438671357)

[3 Anforderungsanalyse 7](#_Toc438671358)

[3.1 Use Cases 7](#_Toc438671359)

[3.1.1 Benutzer Verwaltung 7](#_Toc438671360)

[3.1.2 Räume anzeigen 8](#_Toc438671361)

[3.1.3 Räume buchen 8](#_Toc438671362)

[3.2 Datenbankdiagramm 9](#_Toc438671363)

[3.3 PHP Funktionen 9](#_Toc438671364)

[3.3.1 Api functions 9](#_Toc438671365)

[3.3.2 Datenbank 10](#_Toc438671366)

[3.3.3 Benutzer 10](#_Toc438671367)

[3.3.4 Reporting 12](#_Toc438671368)

[3.3.5 Räume Buchen 12](#_Toc438671369)

[4 Aufbau und Design 14](#_Toc438671370)

[4.1 Ansichten 14](#_Toc438671371)

[4.1.1 RoomList 14](#_Toc438671372)

[4.1.2 PlanView 14](#_Toc438671373)

[4.1.3 About 14](#_Toc438671374)

[4.1.4 RoomView 14](#_Toc438671375)

[4.1.5 Profile (nur eingeloggte Benutzer) 15](#_Toc438671376)

[4.1.6 Bookings (nur eingeloggte Benutzer) 15](#_Toc438671377)

[4.1.7 Administration (nur eingeloggte Benutzer mit Administratorenrechten) 15](#_Toc438671378)

[4.1.8 Reports (nur eingeloggte Benutzer mit Administratorenrechten) 15](#_Toc438671379)

[5 Technische Umsetzung 16](#_Toc438671380)

[5.1 Kommunikation 16](#_Toc438671381)

[5.2 Nicht funktionale Anforderungen 16](#_Toc438671382)

[5.3 Zugangsdaten 16](#_Toc438671383)

[5.4 Testing 17](#_Toc438671384)

[6 Fazit 18](#_Toc438671385)

# Aufgabe

## Ausgangslage

Da ist ein Besitzer eines 12-Familienhauses, der die Verwaltung selbständig führen will. Wir wollen diesem Verwalter ein praktisches Onlinetool erstellen, welches er einfach über den Webbrowser bedienen kann.

## Ziele

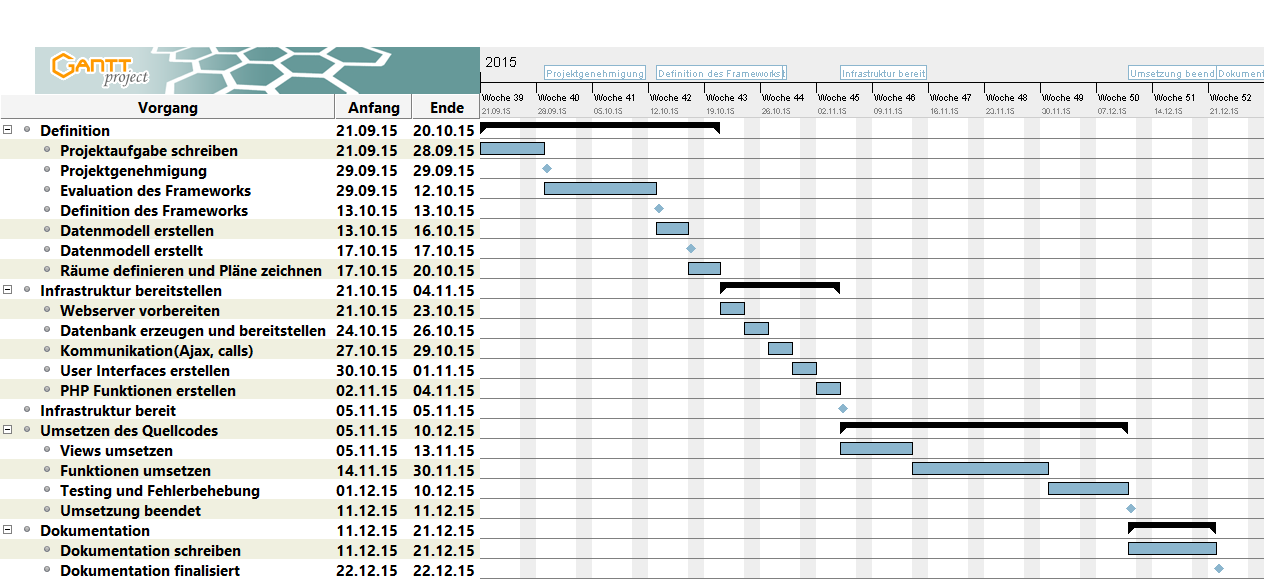
Folgendes wollen Sie jederzeit online abrufen können:

* Mieterspiegel (Adressen und Mietzinse)
* Erfasste Rechnungen (Reparatur-, Oel-, Wasser-, Strom-, Hauswartsrechnungen)
* Weitere Rechnungen sollen jederzeit erfasst werden können.
* Mietzinseingänge sollen einfach erfasst werden können.
* Ende Jahr soll man eine Abrechnung ausdrucken können.
* Die Heizkostenabrechnung und die Nebenkosten sollen ausgewiesen werden.
* Über Formulare können Daten eingegeben, mutiert oder gelöscht werden.

Erstellen Sie diese Online-Verwaltungshilfe für den Hausverwalter. Benutzen Sie Webformulare für die Dateneingabe und lösen Sie die Datenausgabe mit PHP oder Perl. Sie können dazu auch die Datenbank MySQL verwenden. Auch alle weiteren Sprachelement, die Sie kennen, können eingesetzt werden (JavaScript, CSS, ASP usw.)

# Projektmanagement

## Planung (Gantt)



## Organisation

### Team

Kevin Trottmann

Matej Djuranec

Mikayil Bekooglu

### Kommunikation

Wöchentliche Skype Termine, Dropbox Ordner für den Datenaustausch und eine To-do-Liste, in welcher laufend die anstehenden Pendenzen zugeteilt wurden, waren hauptsächlich unsere Kommunikationskanäle. Für dringendere Abklärungen nutzten wir den Gruppenchat in WhatsApp. Des Weiteren trafen wir uns regelmässig in der Schule, um unseren Wissensstand abzugleichen.

### Tools

Dropbox: Dokumentation laufend geführt, To-do-Liste

Webserver: Der Code wurde direkt auf dem Webserver programmiert. Jedes Projektmitglied hatte ein eigenes Verzeichnis und konnte seine Funktionen programmieren und testen. Nach erfolgreichem Test wurde der Code in das Hauptverzeichnis kopiert.

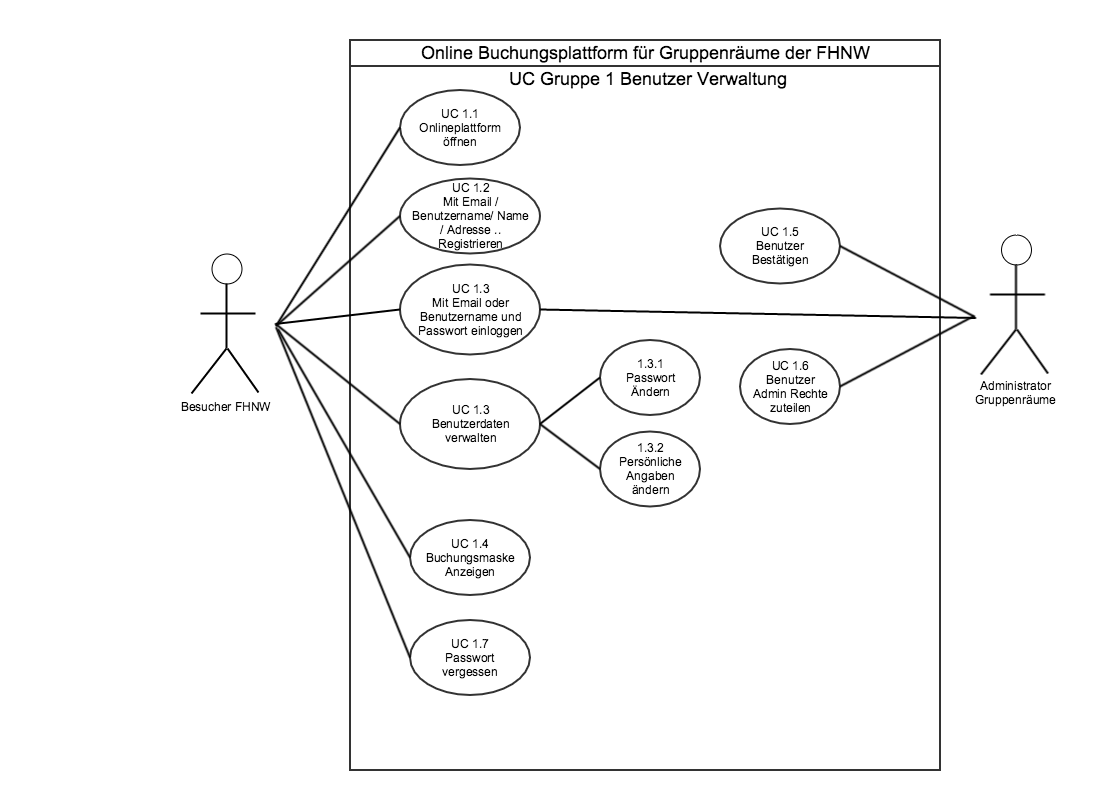
Editor: Jedes Projektmitglied arbeitete mit unterschiedlichen Tools, da jedes Projektmitglied seine bevorzugte Entwicklungsumgebung hat.

# Anforderungsanalyse

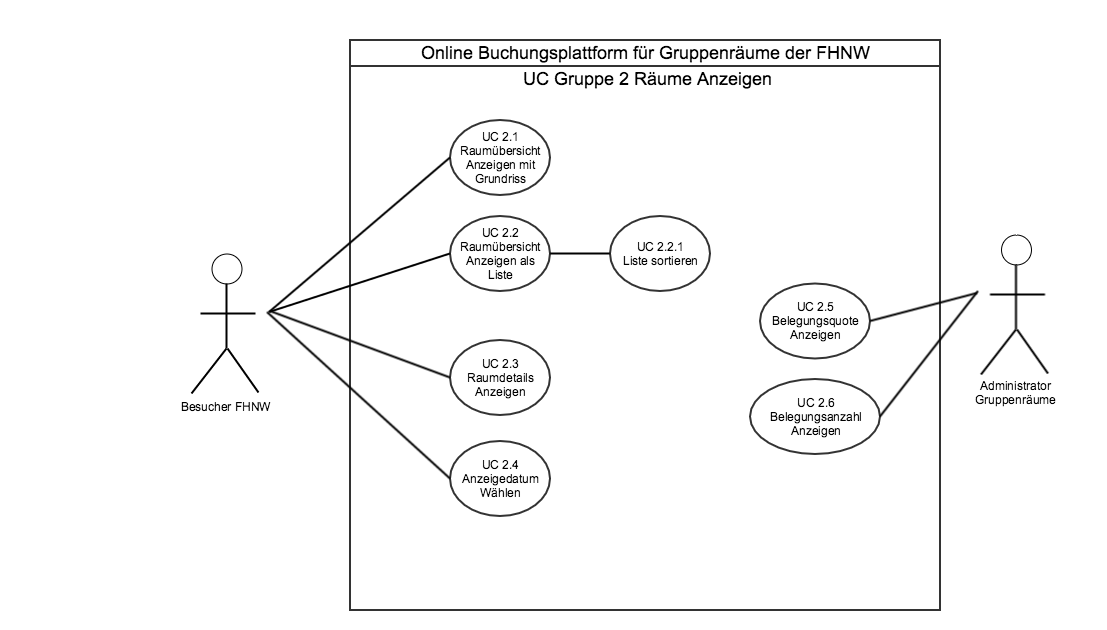
## Use Cases

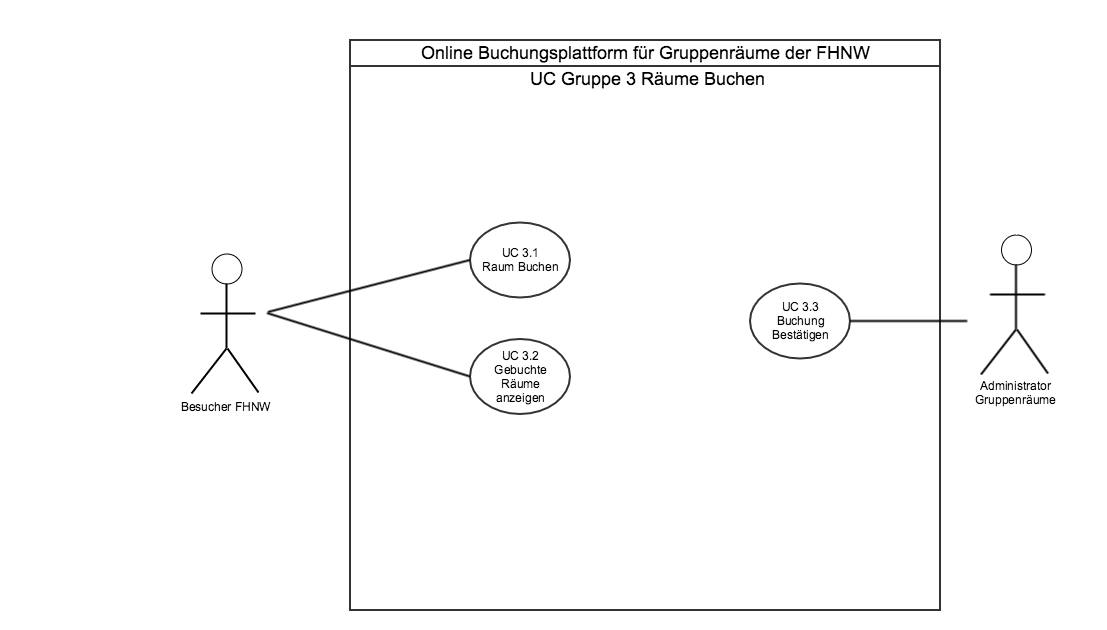
Um die Anforderungen genauer aufzuzeigen, wurden Use Case Diagramme erstellt. Die Use Case Diagramme sind in drei Gruppen aufgeteilt:

### Benutzer Verwaltung



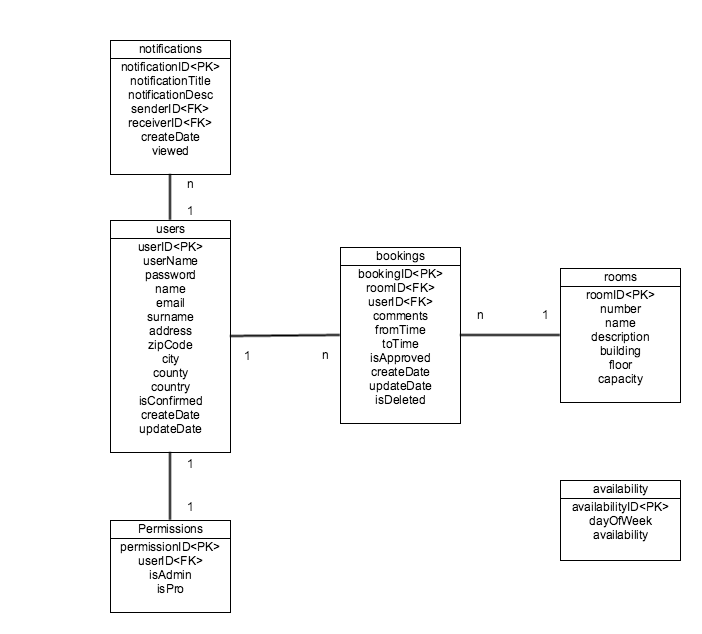
### Räume anzeigen





### Räume buchen

## Datenbankdiagramm



## PHP Funktionen

Die Funktionen wurden in fünf Dateien unterteilt, um die Kapselung und Übersichtlichkeit sicherzustellen.

### Api functions

In der Datei Api befinden sich einfachere Funktionen, wie zum Beispiel die checkEmailReal()-Funktion, welche überprüft, ob der übergebende String eine valide E-Mailadresse ist.

#### callback()

Überprüft die übergebene Methode auf invalide Strings.

#### checkEmail()

Überprüft in der Datenbank, ob bereits ein User mit der eingegebenen E-Mailadresse vorhanden ist.

#### checkEmailReal()

Diese Funktion überprüft, ob es sich um eine gültige E-Mailadresse handelt und gibt einen Boolean zurück.

#### getUserID()

Diese Funktion gibt die User ID für einen übergebenen Benutzernamen oder eine E-Mailadresse zurück.

#### sendEmail()

Der Funktion werden Zieladresse, Betreff und Nachricht übergeben. Anschliessend wird mittels PHPMailer eine E-Mail versendet. Zurückgegeben wird ein Boolean.

#### sendInviteEmail()

Diese Funktion ist vom Prinzip her gleich wie die sendEmail()-Funktion, jedoch wird hier eine Einladung versendet. Der Rückgabewert ist ein Boolean.

### Datenbank

#### connect()

Baut eine Verbindung zur Datenbank auf.

### Benutzer

#### getLogin()

Gibt den aktuellen Benutzer zurück.

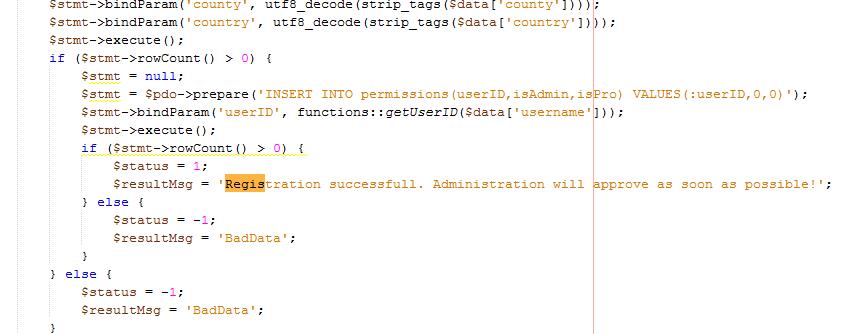
#### checkLogin()

Überprüft das Login des Benutzers.

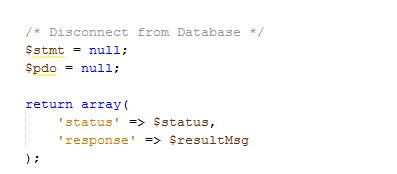
#### logout()

Meldet einen Benutzer vom System ab.

#### registerUser()

Ermöglicht einem neuen Benutzer sich zu registrieren. Um eine Registrierung vornehmen zu können, muss der User folgende Daten angeben: E-Mail, Benutzername, Passwort, Name, Vorname, Strasse, Postleitzahl und Wohnort. Wird eine Angabe nicht übergeben, wird die Registrierung nicht ausgeführt. Im Prepared-Statement *($stmt = $pdo->prepare())* wird eine SQL-Query vorbereitet. Nach der ersten IF-Abfrage werden mittels BindParam-Statements, Parameter auf die angegebenen Variablennamen übergeben.

Nach der Übergabe der Parameter wird das vorbereitete Statement mit dem Befehl *execute()* ausgeführt. Wurde die Query ausgeführt, wechselt der Status auf 1, dies bedeutet, dass die Registrierung erfolgreich abgeschlossen wurde. Ist dies nicht der Fall, gibt der Status eine negative Zahl zurück. Ein negativer Status besagt, dass ein Fehler vorliegt.

Zum Schluss wird die Datenbankverbindung geschlossen und ein JSON-Array mit dem Status und der dazu gehörigen Message zurückgegeben.

#### changeAdmin()

Vergibt oder entzieht einem Benutzer die Administrationsrechte.

#### changePro()

Vergibt oder entzieht einem Benutzer die Pro-Rechte.

#### deleteUser()

Löscht einen User aus der Datenbank.

#### approveUser()

Bevor die Registration eines Users abgeschlossen werden kann, muss er vom Administrator bestätigt werden. Nach der Bestätigung wird dem neuen User eine Willkommens-E-Mail gesendet.

#### getUsers()

Gibt alle Benutzer aus.

#### updateUser()

Mithilfe dieser Funktion können die eingeloggten Benutzer ihre Profildaten ändern. Der Username kann nicht geändert werden. Alle anderen Daten kann man updaten. Die Änderungen werden in der Datenbank gespeichert. Der Rückgabewert ist ein Boolean.

#### changePassword()

Um das Passwort zu ändern, wird der Funktion changePassword() das bisherige und das neue Passwort übergeben. Das bisherige Passwort wird mit der Datenbank verglichen und bei Übereinstimmung mit dem neuen ersetzt. Der Rückgabewert ist ein Boolean.

#### forgotPassword()

Wurde das Passwort vergessen, kann man sich ein neues per E-Mail zusenden lassen. Mit der PHP-Funktion unique(rand(), false) wird ein neues Passwort generiert. Mit dem MD5-Hash wird es verschlüsselt und in der Datenbank aktualisiert. Das unverschlüsselte Passwort wird an die eingegebe E-Mailadresse versendet, mit der Aufforderung das Passwort im Profil zu ändern.

### Reporting

#### getReport()

Erzeugt die Daten für den Report und gibt für jeden Raum aus, ob er frei, reserviert oder noch bestätigt werden muss.

#### getBookingByRoom()

Gibt alle Buchungen eines Raumes aus.

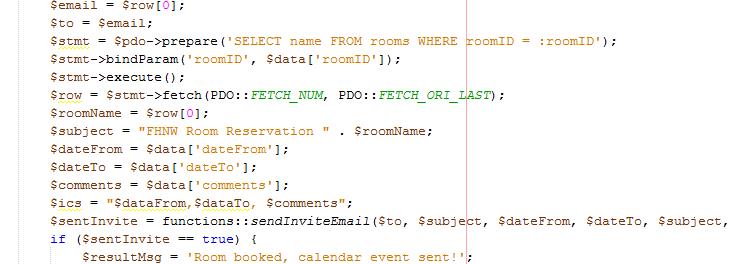
#### getAvailability()

Gibt ein Boolean zurück mit der Information, ob der Raum zu diesem Zeitpunkt schon reserviert ist.

### Räume Buchen

#### C:\Users\Diego\Documents\PrintScreen Files\ScreenShot045.jpgaddBooking()

Reserviert einen Raum für einen bestimmten Zeitraum. Mit *if(isset($\_SESSION[‘userID‘]))* wird überprüft, ob der User eingeloggt ist. Weiter wird überprüft, ob der zu buchende Raum zur gewünschten Zeit schon gebucht ist und ob der User in der gewünschten Zeit bereits einen Raum reserviert hat. Als nächstes wird überprüft, ob es sich um einen Pro-User handelt. Anschliessend wird die SQL-Query für die Raumreservation vorbereitet.

Wurde die Query erfolgreich ausgeführt und handelt es sich um einen Pro-User, holt sich die Funktion aus der Datenbank den Namen und die E-Mailadresse des eingeloggten Users. Für das Versenden der Einladungs-E-Mail ist die Funktion *sendInviteEmail()* zuständig. Ihr werden E-Mailadresse, Betreff, Anfangsdatum und –zeit, Enddatum und –zeit sowie die Kommentare als Parameter übergeben. Wie bei der Funktion *registerUser()* ist auch hier die Rückgabe ein JSON-Array.

#### hasRoomBookingConflict()

Gibt ein Boolean zurück mit der Information, ob dieser Raum zu diesem Zeitpunkt bereits gebucht ist.

#### hasUserBookingConflict()

Gibt ein Boolean zurück mit der Information, ob der User zu diesem Zeitpunkt bereits einen Raum gebucht hat.

#### getBookings()

Gibt alle Buchungen zurück.

#### getBookingsByRoom()

Gibt eine Liste mit allen Buchungen eines Raumes zurück.

#### approveBooking()

Mit dieser Funktion kann der Administrator eine Buchung bestätigen. Anschliessend wird ein E-Mail an den Benutzer versendet mit der Buchungsbestätigung. Als letztes wird eine Meldung ausgegeben mit der Information, ob die Buchung bestätigt wurde oder nicht. Fehlen die Administrationsrechte wird eine Meldung ausgegeben.

#### deleteBooking()

Mit dieser Funktion kann ein Benutzer seine eigenen und ein Administrator sämtliche Buchungen löschen. Als Rückgabe erhält man die Information, ob das Löschen erfolgreich war oder nicht.

#### getRooms()

Gibt alle Räume mit der Information ihrer Verfügbarkeit zurück.

# Aufbau und Design

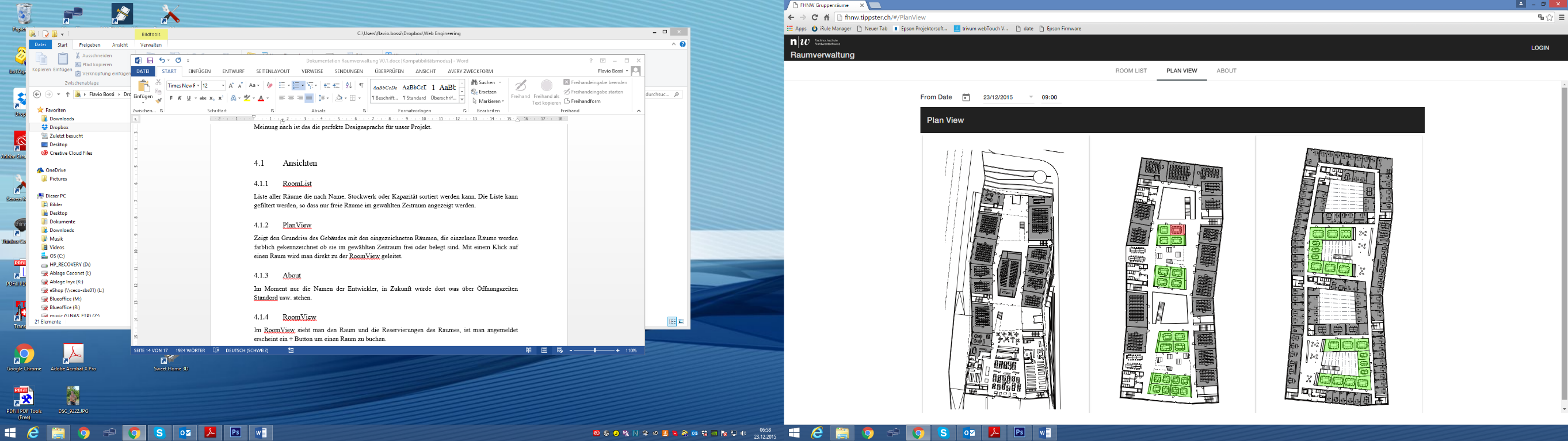
Beim Design haben wir uns soweit wie möglich an das Material Design gehalten, welches von Google entwickelt wurde. Die Stärke dieses Designs liegt in der intuitiven Bedienung und der optimalen Darstellung auf den unterschiedlichsten Geräten.

## Ansichten

### RoomList

Liste aller Räume die nach Name, Stockwerk oder Kapazität sortiert werden kann. Die Liste kann gefiltert werden, sodass nur freie Räume im gewählten Zeitraum angezeigt werden.

### PlanView

Zeigt den Grundriss des Gebäudes mit den eingezeichneten Räumen. Dabei sind die einzelnen Räume mit verschiedenen Farben gekennzeichnet, um zu visualisieren, welche Räume zum gewählten Zeitraum verfügbar sind. Mit einem Klick auf einen Raum wird man direkt zu der RoomView geleitet.

### About

Im Moment stehen auf dieser Seite nur die Namen der Entwickler. In Zukunft könnte dort Öffnungszeit, Standort usw. abgebildet werden.

### RoomView

Im RoomView sieht man einen Raum und dessen Reservierungen. Ist ein Benutzer angemeldet, erscheint ein + Button um einen Raum zu buchen.

### Profile (nur eingeloggte Benutzer)

Zeigt die Profildaten des angemeldeten Benutzers an. Zudem können die persönlichen Daten und das Passwort bearbeitet werden.

### Bookings (nur eingeloggte Benutzer)

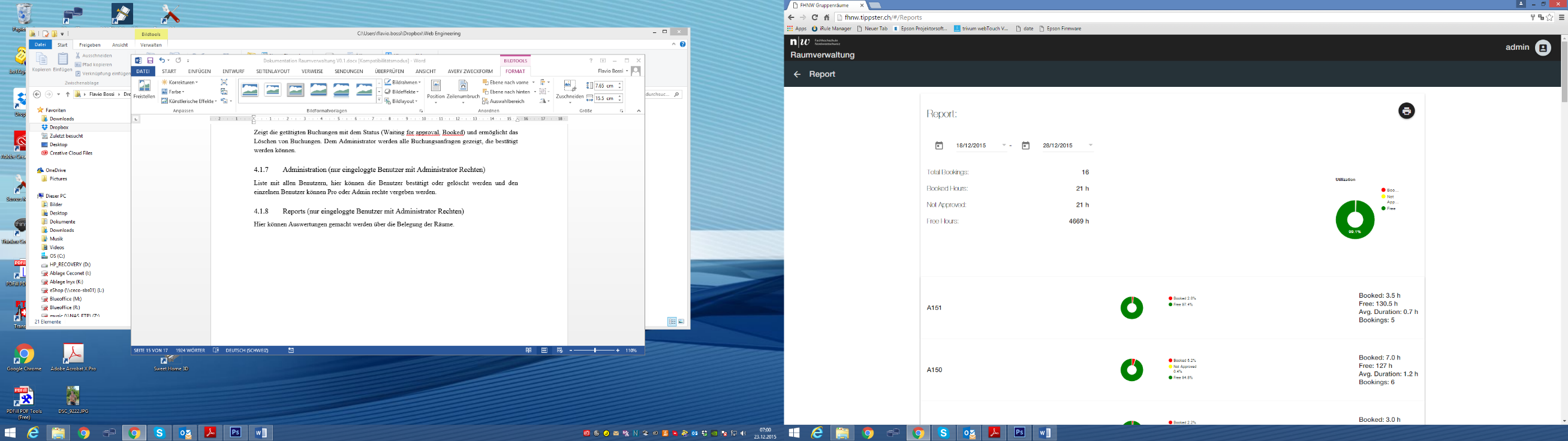
Zeigt die getätigten Buchungen und deren Status (Waiting for approval, Booked) und ermöglicht das Löschen von Buchungen. Dem Administrator werden alle Buchungsanfragen gezeigt, welche zu bestätigen sind.

### Administration (nur eingeloggte Benutzer mit Administratorenrechten)

Enthält eine Liste mit allen Benutzern. Der Administrator kann Benutzer bestätigen, löschen und einzelnen Benutzern Pro- oder Administratorenrechte vergeben.

### Reports (nur eingeloggte Benutzer mit Administratorenrechten)

Hier können Auswertungen über die Belegung der Räume gemacht werden.



# Technische Umsetzung

## Kommunikation

Die Server Client Kommunikation wird über AJAX Calls realisiert, welche im Javascript Code aufgerufen werden. Die PHP Funktionen geben jeweils einen String im JSON Format zurück.

Alle PHP Funktionen, welche über AJAX Calls aufgerufen werden, geben mindestens zwei Parameter („status“ und „response“) zurück.

Status wird initial bei einer Funktion immer auf 0 gesetzt und während der Verarbeitung auf 1 oder einen negativen Wert gesetzt. Ein negativer Wert bedeutet, dass ein Fehler aufgetreten ist und Status 1 zeigt die erfolgreiche Verarbeitung auf. Der Response kann zusätzlich ein Fehler oder eine Erfolgsmeldung mitgegeben werden.

Zwischen Client und Server werden keine unverschlüsselten Passwörter versendet. Sie werden entweder Client- oder Serverseitig mit MD5 verschlüsselt. Nur bei der Funktion forgotPassword() wird das Passwort unverschlüsselt per E-Mail übermittelt. Hier wird der User jedoch aufgefordert sein Passwort umgehend zu ändern.

## Nicht funktionale Anforderungen

Wir haben uns bei der Entwicklung auf die aktuellen Versionen der grössten Hersteller konzentriert. In diesem Umfeld kann davon ausgegangen werden, dass niemand mehr mit IE8 usw. arbeitet. Wichtig ist aus unserer Sicht das responsive Design, die Studenten wollen schnell und einfach über das Smartphone die Verfügbarkeit prüfen und sofort einen Raum buchen können. Der grosse Mehrwert von unserem Tool ist die Einfachheit und Geschwindigkeit, weshalb der Fokus nicht auf ein ausgefallenes Design, sondern auf eine intuitive Bedienung und schnelle Ladezeit gelegt wurde.

## Zugangsdaten

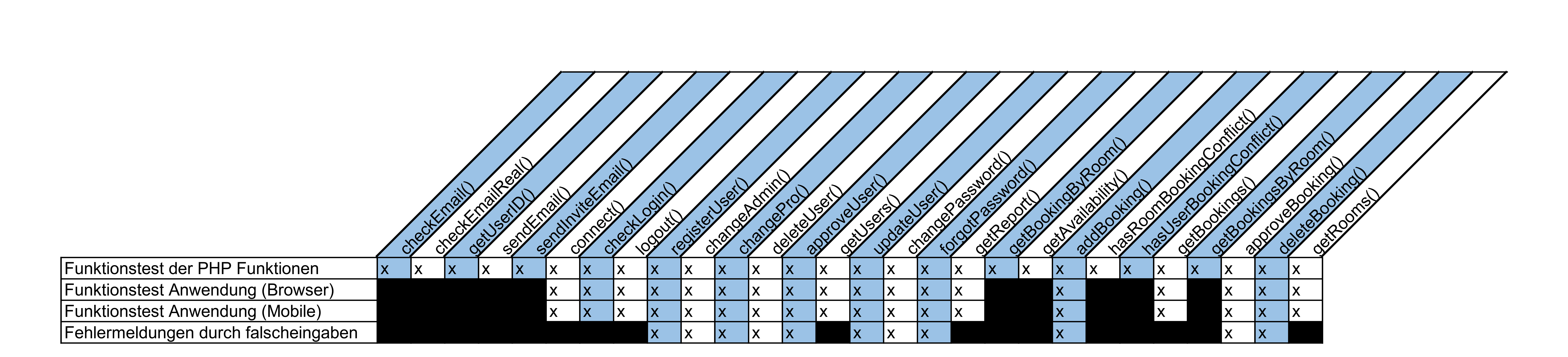
URL: <http://fhnw.tippster.ch>

Admin: admin/admin

Pro-User: test/test

## Testing

Nach der Programmierung jeder PHP Funktion wurden verschiedene Tests durchgeführt. Anschliessend an diese Tests wurde das System auf verschiedenen Browsern, auf Mobile- und Desktopgeräten aus Benutzersicht getestet.



# Fazit

Beim Kaffee in der Pause der ersten Vorlesung im Modul Web-Engineering wurde uns klar, dass wir nun die Möglichkeit bekommen, unser lang ersehntes Tool für die Raumverwaltung umzusetzen. Nach den ersten Abklärungen mussten wir feststellen, dass von den Mitarbeitern der Raumverwaltung der FHNW, dieses Tool als nicht notwendig erachtet wird.

Je länger wir uns mit dem Thema beschäftigten und nach unzähligen Gesprächen mit Dozierenden und Studenten war eines klar. Die Anwender wollen dieses Tool.

Einerseits durften wir während dieser Projektarbeit viel Neues lernen und anderseits haben wir ein Tool auf das wir stolz sein können.

Wir hoffen die Belegschaft der Raumverwaltung der FHNW mit dem Ergebnis überzeugen zu können und dass unser Tool in Zukunft zur Anwendung kommt.