

Communication Tool “Slack light”

Erstellen Sie eine Webanwendung für ein Communication-Tool (wie bspw.: <http://www.slack.com>). In Ihrer Anwendung können registrierte und angemeldete Benutzer miteinander kommunizieren, dazu können die Benutzer unterschiedliche Themen / Channels nutzen.

1 Funktionale Anforderungen

1.1 Posting

Die Postings werden zumindest mit folgenden Attributen erfasst:

- Channel / Thema (1 : n)
- Titel
- Beschreibung (Inhalt)
- Verfasser (Benutzer)
-

Nach erfolgreichem Login werden dem Benutzer seine Nachrichten in chronologischer Form, geordnet / kategorisiert nach Channels angezeigt. Sobald ein Benutzer eine Nachricht gelesen hat (= angezeigt bekommen hat), wird diese als gelesen markiert. Achten Sie auch darauf, ungelesene Nachrichten zu markieren. Zusätzlich soll es möglich sein, dass ein Benutzer gewisse Posts als “wichtig” markiert - diese erscheinen in der Auflistung gesondert markiert und werden an erster Stelle gereiht. Die Nachrichten können je nach Channel gefiltert werden, sodass die Kommunikation auch nur innerhalb eines Channels stattfinden kann.

Benutzer können nur eigene Nachrichten bearbeiten und löschen, aber nur dann, wenn noch keine Antworten bzw. nachfolgenden Posts auf den jeweiligen Eintrag existieren.

WICHTIG: innerhalb eines Channels / Themas können mehrere Benutzer miteinander kommunizieren - verschiedene Stati der Postings (gelesen / ungelesen / wichtig) gelten also je eingeloggtem Benutzer!

Sie können für Ihre Anwendung vordefinierte Channels nutzen, eine eigene Channel-Verwaltung ist nicht notwendig - es sollten jedoch zumindest 2 Channels existieren. Bei der Neuregistrierung kann der Benutzer auswählen, welche Channels er nutzen möchte.

1.2 Benutzer / Rollen

Standardmäßig existieren folgende Rollen:

1. Anonyme Besucher

Können sich für die Anwendung durch Vergabe eines Benutzernamens und Passwort neu registrieren und einen oder mehreren Channels beitreten.

2. Registrierte Nutzer

Nach erfolgter Registrierung kann die Kommunikation in den ausgewählten Channels stattfinden.

2 Non-funktionale Anforderungen

2.1 Programmierung Server

Umsetzung mittels PHP > 5.4 (OOP) – achten Sie auf einen sauberen Aufbau und eine Trennung der einzelnen Anwendungsschichten. Für die Anbindung der Datenquelle benutzen Sie die Bibliotheken *mysqli* oder *PDO*. Sichern Sie Ihre Anwendung gegen SQL-Injection und XSS-Angriffe durch serverseitige Prüfung der Eingaben und Parameter.

Bei Benutzeraktionen werden zusätzlich Daten zur Identifizierung mitgespeichert werden (bspw. IP-Adresse des Benutzers)

2.2 Datenbank

Die Datenverwaltung erfolgt mittels MySQL 5 (InnoDB). Achten Sie auf die Grundregeln des Datenbank-Designs (Datentypen, Normalisierung, sinnvolle Constraints,...). Hinweis: Datensätze, die von den Usern gelöscht wurden, sollten nicht direkt aus der DB gelöscht werden, sondern lediglich mit einem entsprechenden Flag gekennzeichnet werden.

2.3 Client / Frontend

Die Verwendung von Javascript (auch Frameworks, aber lediglich UI Frameworks wie bspw. JQuery UI) ist erlaubt. Der HTML-Code sollte je nach DOCTYPE (XHTML oder HTML5) valide sein, sämtliche Formatierungen via CSS umgesetzt werden. Die Benutzeroberfläche sollte intuitiv bedienbar sein und bei neuen Nutzern der Anwendung keinerlei Erklärungsbedarf erfordern. Achten Sie insbesondere auf Validierung von Eingaben sowie die Fehlerausgabe. Auch eine sinnvolle Verwendung von AJAX ist möglich, sofern diese für Sie sinnvoll erscheint (ist jedoch nicht zwingend Voraussetzung).

Hinweis: das „Design“ sollte einfach, aber funktional sein – wichtig ist die Bedienbarkeit und die logische Führung des Benutzers durch die Website (Navigationselemente, Feedback bei Benutzeraktionen, Formular-Validierungen & -Layout,...). Achten Sie insbesondere auf eine übersichtliche Darstellung, wenn die Threads länger werden.

3 Form der Abgabe

3.1 Dokumentation

Erstellen Sie für Ihr Datenbank-Modell ein ER- oder UML-Diagramm, welches die Entitäten und Beziehungen visualisiert. Für die Anwendung erstellen Sie bitte eine textuelle Beschreibung und / oder eine dokumentierte Skizze der Architektur.

3.2 Testfälle

Führen Sie ausführliche Tests Ihrer Anwendung durch und dokumentieren Sie diese. Testen Sie auch fehlerhafte Eingaben! Ihre Testfälle sollten auf jeden Fall folgende Zustände testen:

- Benutzerregistrierung (neu)
- Benutzerlogin (inkl. Fehlerüberprüfung)
- Darstellung des Nachrichten-Feeds inkl. Status & Favoriten
- Benutzer erfasst neuen Eintrag
- Benutzer editiert / löscht bestehenden Eintrag (nur, wenn noch keine Antwort)
- Benutzer markiert Beiträge als Favoriten

- Zweiter Benutzer kommuniziert mit erstem Benutzer und führt ebenfalls obige Schritte durch

Erstellen Sie die Dokumentation und die Testfälle in Form eines PDF-Dokuments.

3.3 Anwendung

- **Datenbank**

SQL – Dump inkl. Testdaten und CREATE DATABASE statement

Datenbankname: **fh_2015_scm4_[IHRE_MATRIKELNUMMER]**

Benutzername: **fh_2015_scm4**

Passwort: **fh_2015_scm4**

- **Code**

Zip-Archiv, Startdokument = index.php, muss auf Standard – XAMPP Installation lauffähig sein. Die Dokumentation speichern Sie bitte im Unterverzeichnis „/doc“.

3.4 Deadline

Laden Sie Ihre Anwendung als **eine Datei im ZIP-Format** in der entsprechenden Abgabe via Moodle hoch. Den konkreten Termin entnehmen Sie bitte dem Abgabe-Modul in Moodle.

3.5 Beurteilung

Die Beurteilung der Übung erfolgt im Rahmen einer Präsentation vor der eigenen Gruppe. Bei dieser Präsentation stellen Sie bitte Ihre Lösung vor, demonstrieren die Testfälle und stehen für technische und inhaltliche Fragen bzw. Code-Reviews zur Verfügung.

Der Termin wird, wie in der LVA bereits angesprochen, von Ihrem Studiengangsprecher mit der Administration abgeklärt und findet am Beginn des WS 2015 statt. Der genaue Termin wird noch kommuniziert.

Übung 3

1 Kommunikationstool 'Slack Light'

1.1 Datenbank

Folgende Abbildung zeigt das ER Diagramm des Datenmodells für das Kommunikationstool 'Slack Light'.

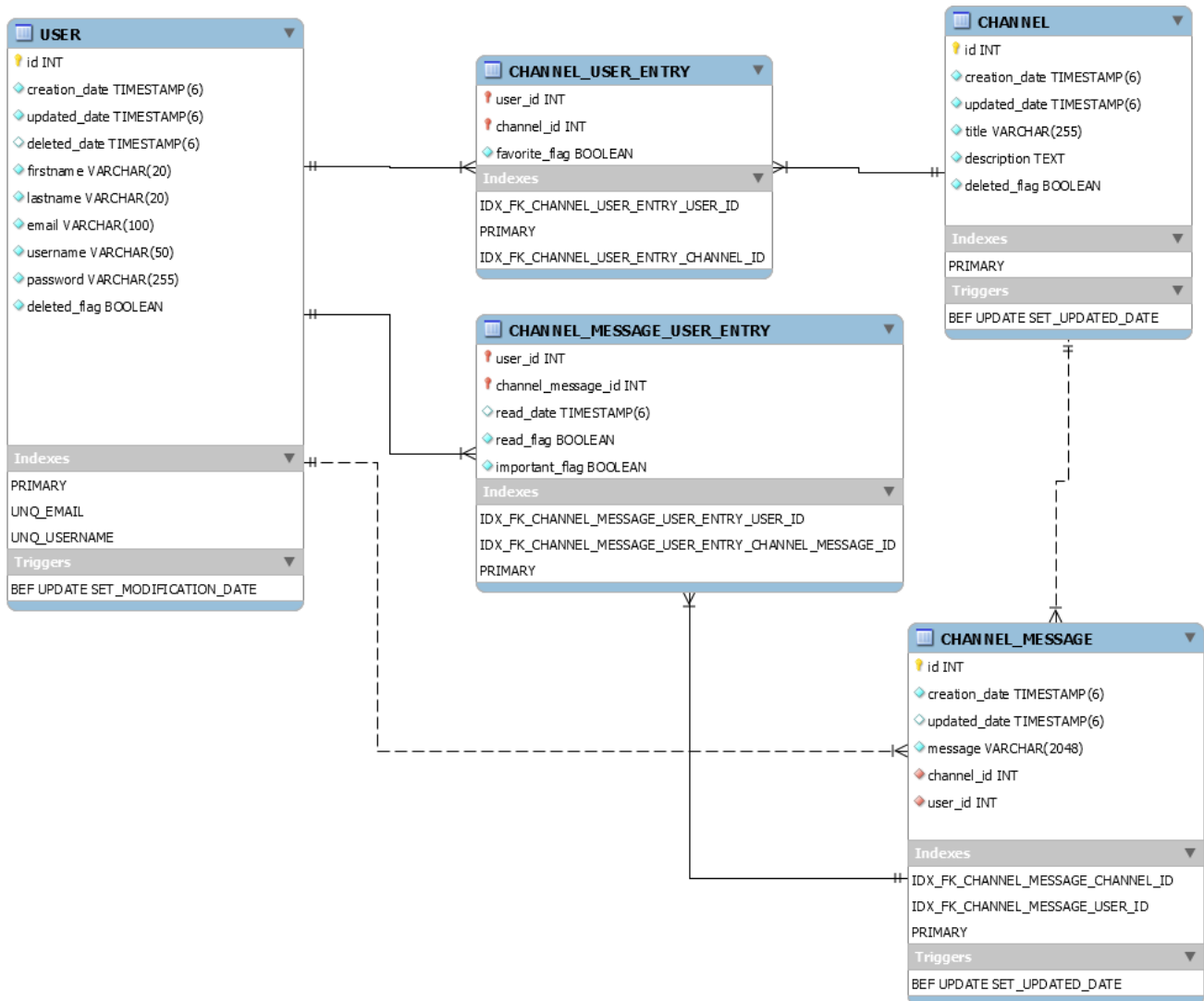


Abbildung 1: ER-Modell

1.2 Applikationsarchitektur

Folgender Abschnitt beschreibt die Architektur der Applikation.

Die Applikation ist in zwei Haupt PHP Dateien aufgeteilt über die alle Request gehandelt werden:

1. **index.php:**

All offenen Ressourcen wie Login und Registrierung

2. **start.php:**

Alle geschützten Ressourcen, die einen Login erfordern.

Diese Dateien sowie alle *.css, *.js Dateien sowie Images sind in einem public Verzeichnis zusammengefasst und können ohne Zugriffskontrolle abgerufen werden.

Alle anderen angezeigten Seiten werden über das Templating Tool **Twig** generiert und werden über die beiden PHP Dateien index.php oder start.php and den Client übermittelt. Dadurch ist kein direkter Zugriff auf diese Dateien möglich.

Folgende Abbildung illustriert die Architektur der Applikation beziehungsweise den Ablauf eines Requests.

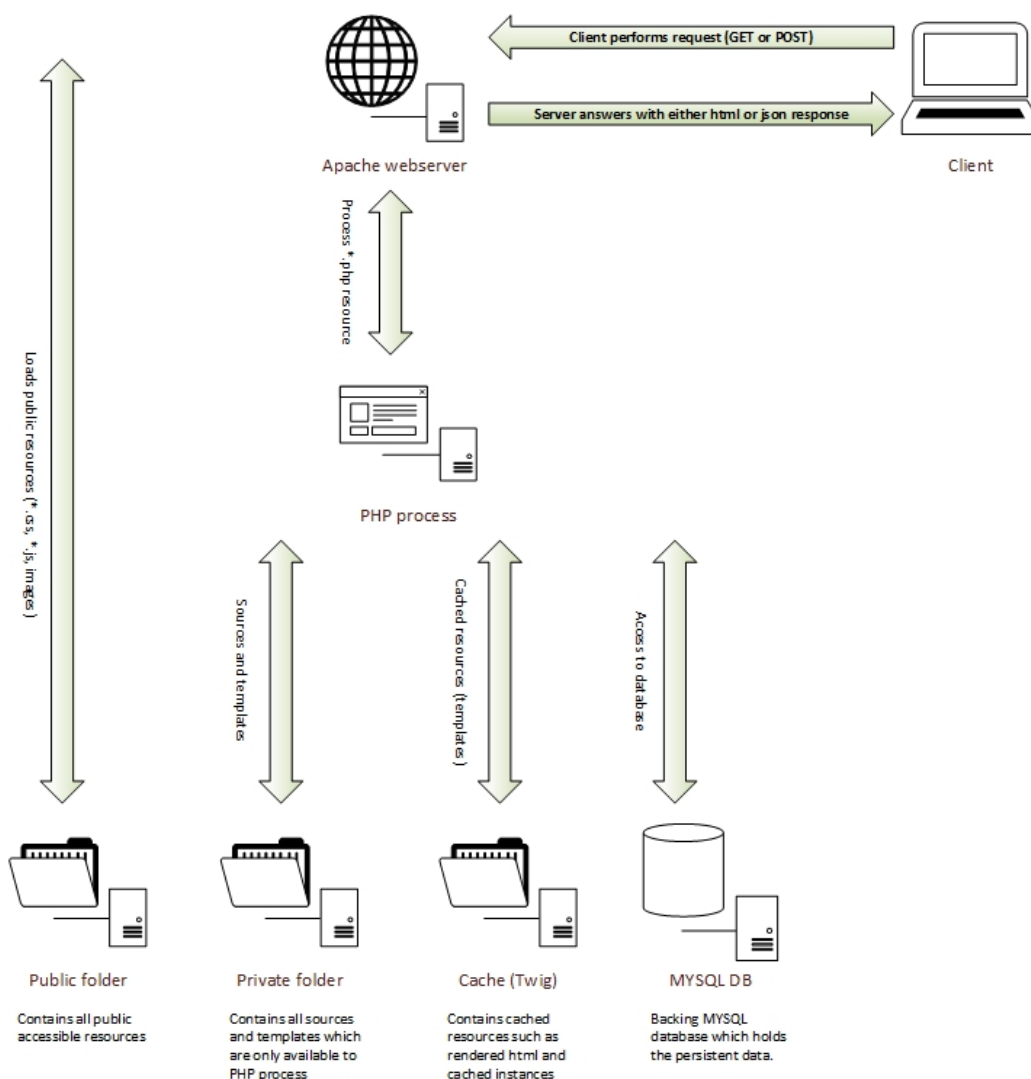


Abbildung 2: Applikationsarchitektur

Übung 3

Die Applikation wurde in 3 Ebenen unterteilt, wobei nicht alle Ebenen nach außen frei zugänglich sind.

1. **public/***

Enthält alle öffentlich und ohne Einschränkung zugängliche Ressourcen. Diese Ressourcen sind direkt über den Webserver zugänglich und müssen nicht von einem PHP Prozess provided werden. Diese sind z.B.: Javascript-, CSS-, Image-Dateien oder die beiden Hauptseiten index.php und start.php (erfordert eingeloggten Benutzer).

2. **source/***

Alle geschützten Ressourcen wie die PHP Sources und die Twig Templates. Sie sind nur über den PHP Prozess und Source in index.php und start.php erreichbar. Dadurch sind die Sources von außen abgeschirmt, wobei der Webserver diese Verzeichnisse von einem externen zugriff zu schützen hat.

3. **cache/***

Enthält die kompilierten Twig Templates sowie die gerenderten Html-Dateien, somit müssen diese nicht bei jeder Verarbeitung neu kompiliert oder gerendert werden. Der Code entscheidet hierbei ob ein erneutes Rendern erforderlich ist oder nicht.

Es werden zwei externe Libraries verwendet, die zwar über composer geladen wurden aber über einen eigenen Autoloader innerhalb des PHP Prozess geladen werden.

1. **Twig:**

Hierbei handelt es sich um eine Template Library, die dazu verwendet wird um die Html-Dateien für den Client aufzubereiten. Hierzu wird ein Assoziativen Array mit Parametern befüllt, welche innerhalb der Twig-Template-Engine verarbeiten werden um das Html zu produzieren. Dabei ist kein PHP Source in den Html-Dateien enthalten, was meiner Meinung nach die Übersichtlichkeit erhöht, den der dynamische Content wird innerhalb des PHP Codes aufbereitet und nur die Ausgabeparameter an die Twig-Template-Engine weitergereicht.

2. **Stash:**

Hierbei handelt es sich um eine Caching Library, welche dazu verwendet wird um die gerenderten Html-Dateien zu cachen, damit diese nicht immer neu verarbeitet werden müssen. Im PHP Code wird entschieden ob eine Html-Datei neu gerendert werden soll oder ob diese aus den Cache genommen werden soll.

Übung 3

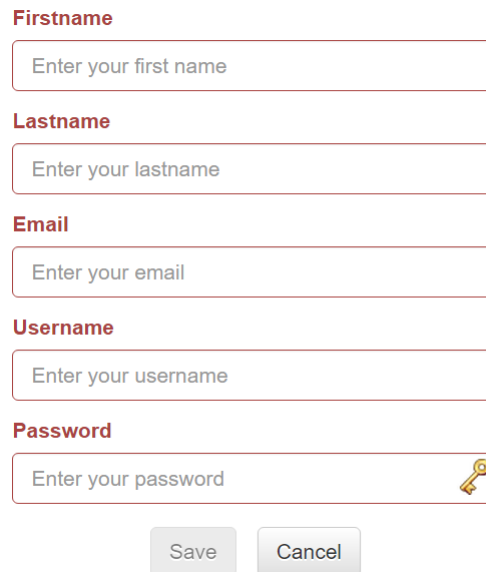
1.3 Tests

Folgend sind die Tests für die Applikation angeführt.

1.3.1 Benutzerregistrierung

Folgend sind die Tests für die Benutzerregistrierung angeführt.

Folgender Test zeigt die angewendete Client Validierung über Bootstrap-Form-Validation Plugin.



A registration form with five input fields, each with a red border indicating a validation error. The fields are labeled 'Firstname', 'Lastname', 'Email', 'Username', and 'Password'. The 'Password' field has a key icon on the right. Below the fields are 'Save' and 'Cancel' buttons.

Firstname
Enter your first name

Lastname
Enter your lastname

Email
Enter your email

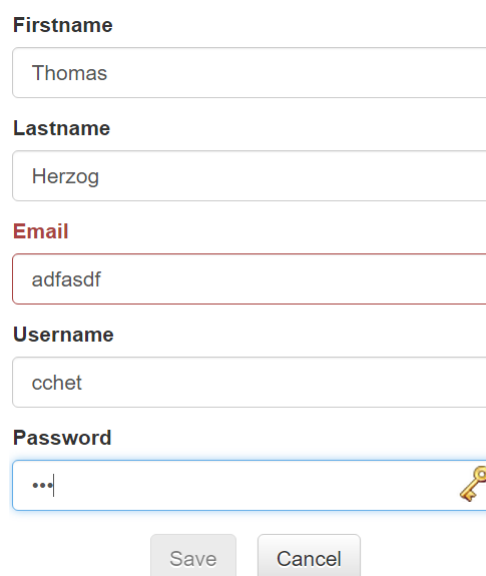
Username
Enter your username

Password
Enter your password

Save Cancel

Abbildung 3: Bootstrap Form Validation

Neben der Required Validerung erfolgt auch eine Validierung auf eine syntaktisch gültige E-Mail Adresse.



A registration form with five input fields. The 'Firstname' field contains 'Thomas', 'Lastname' contains 'Herzog', 'Email' contains 'adfasdf', and 'Username' contains 'cchet'. The 'Password' field has a key icon on the right. Below the fields are 'Save' and 'Cancel' buttons.

Firstname
Thomas

Lastname
Herzog

Email
adfasdf

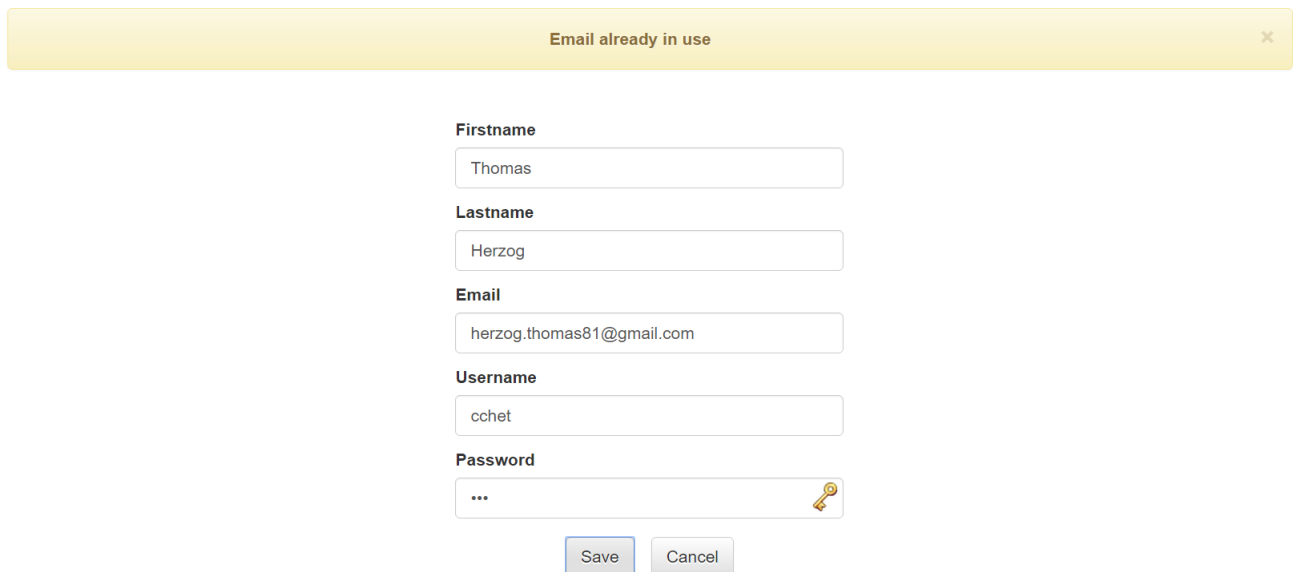
Username
cchet

Password
...

Save Cancel

Abbildung 4: Bootstrap Form Validation E-Mail

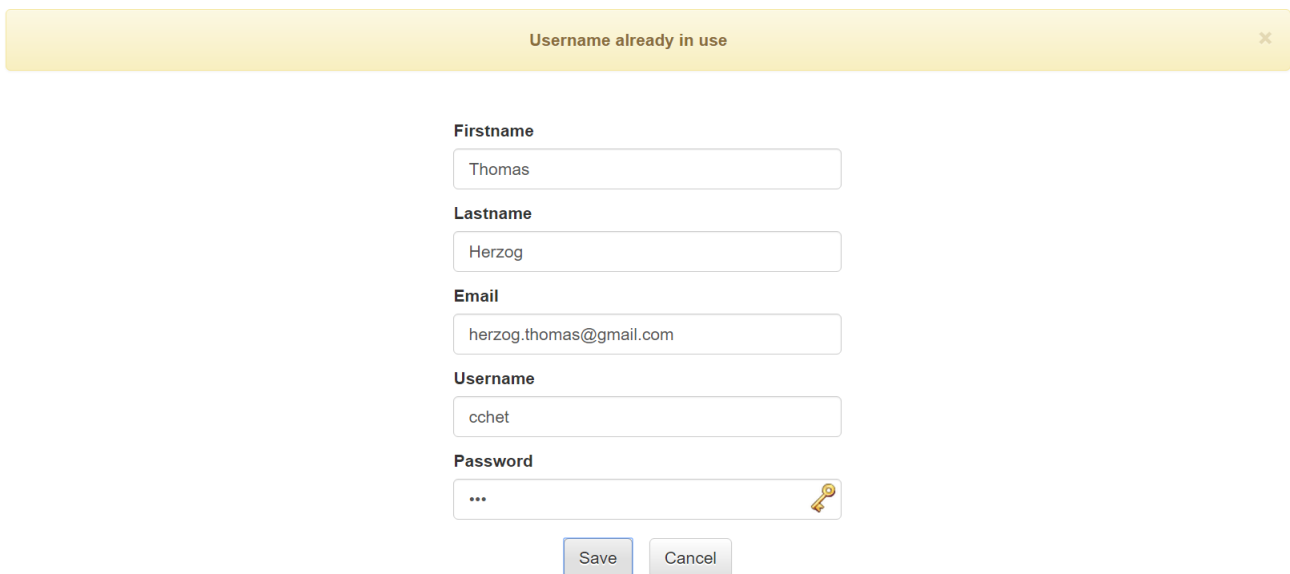
Wenn eine E-Mail bereits von einem Benutzer verwendet wird, dann wird dies serverseitig Validiert und an eine Fehlermeldung an den Client übermittelt.



The screenshot shows a registration form with a yellow error banner at the top that reads "Email already in use" with a close button (X). The form fields are: Firstname (Thomas), Lastname (Herzog), Email (herzog.thomas81@gmail.com), Username (cchet), and Password (masked with three dots and a key icon). At the bottom are "Save" and "Cancel" buttons.

Abbildung 5: Serverseitige E-Mail Validierung

Wenn ein Benutzername bereits von einem aktiven Benutzer verwendet wird, dann wird dies serverseitig validiert und eine Fehlermeldung an den Client übermittelt.



The screenshot shows the same registration form as in Abbildung 5, but the yellow error banner at the top reads "Username already in use" with a close button (X). The form fields are: Firstname (Thomas), Lastname (Herzog), Email (herzog.thomas@gmail.com), Username (cchet), and Password (masked with three dots and a key icon). At the bottom are "Save" and "Cancel" buttons.

Abbildung 6: Serverseitige Benutzernamen Validierung

Übung 3

Nach der erfolgreichen Registrierung eines Benutzers wird eine Erfolgsmeldung angezeigt, wobei nach einigen Sekunden automatisch auf die Login Seite weitergeleitet wird, obwohl ebenfalls eine Button zur Verfügung steht, der auf die Login Seite wechselt sollte das automatische Weiterleiten nicht funktionieren.

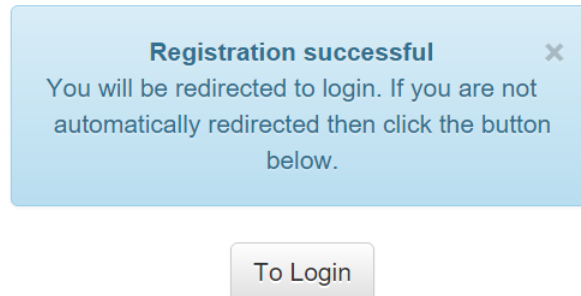


Abbildung 7: Registrierung erfolgreich