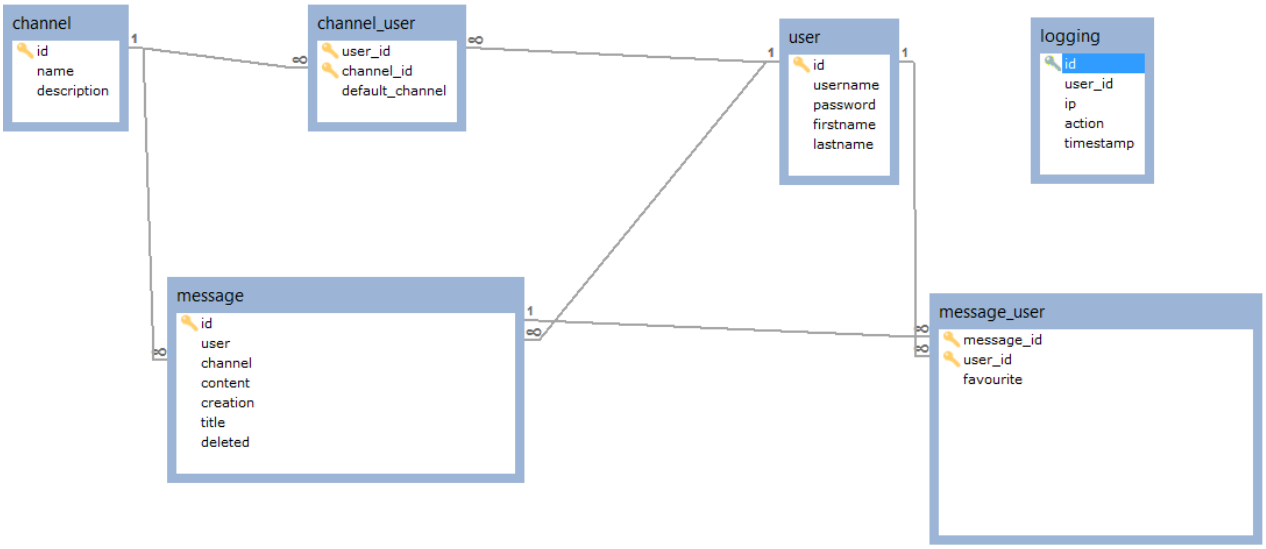
Datenbank:



Channel:

In dieser Tabelle werden alle Channels gespeichert.

Message:

In dieser Tabelle werden alle verfassten Nachrichten gespeichert. Die Spalte User speichert dabei die UserID des verfassers. Alle anderen Spalten erklären sich von selbst.

User:

Speichert alle notwendigen Daten eines Users. Das Passwort wird als Hash gespeichert.

Channel_User:

Speichert die Zuordnung, welcher User welche Channels lesen und schreiben kann.

Message_User:

Wird einem User eine Message zum ersten Mal angezeigt, wird in dieser Tabelle ein Eintrag erfasst. Exisitiert also für User/Nachricht ein Eintrag, bekommt er diese Nachricht als gelesen angezeigt. Markiert er eine Nachricht als Favourite, so wird zusätzlich dieses Feld gesetzt.

Php:

Die Applikation wurde vom Prinzip als MVC aufgebaut. Die Grundlage dafür, haben wir im Unterricht, bei unserem Beispiel erworben. Diesen Grundstock habe ich deshalb auch wiederverwendet.

Die Struktur habe ich so aufgebaut das alle Controller in /controller , alle datenzugriffs funktionen in /dataManagers, alle Entitäten in /models, alle Views in /views liegen.

Da Im Unterricht empfohlen wurde, für die Views eine Template-Engine wie zum Beispiel Twig zu verwenden, habe ich diese benutzt. Die Templates liegen hierbei in /templates.
Es wurde ein BasisTemplate entwicklet, von welchem jedes weiter ableitet.

Um das Laden eines Templates (anzeigen) etwas zu vereinfachen, habe ich mir eine Hilfsklasse erstellt (/util/TemplateLoader.php). Jeder Controller erhält eine Methode Show welche dann diesen TemplateLoader verwendet. In der View kann controller->show() aufgerufen werden, um ein Template zu rendern.

Im TemplateLoader werden immer gewisse Basis variablen mit übergeben. Beispielsweise wenn ein Error passiert ist, so wird dieser mitübergeben, damit er dem User angezeigt werden kann.

In den Controllern, werden sowohl die Daten ausgelesen, validiert, gespeichert usw. Daher sind die Controller auch alle Singeltons. Die Klasse ViewController habe ich vom Unterricht übernommen. Diese kümmert sich darum, die Aktionen an den richtigen Controller weiter zu leiten.

Als Datenbankzugriff wird mysqli verwendet. In einer Basis Klasse DataManager werden alle Grundfunktionalitäten zur verfügung gestellt. Alle anderen leiten von dieser ab, und erweitern Sie um Funktionalitäten. Z.B. MessageDataManager stellt dann Funktionen wie Message aus der Datenbank auslesen dar.

Da Messages immer aktuell angezeigt werden sollen, werden diese per AJAX long polling geholt. Auf der Seite befindet sich also ein JavaScript welches in sekundenabständen den Server um neue Nachrichten befragt, und anzeigt. Der zuständige Controller liefert dabei nach anfrage ein JSON Array zurück, welches die neuen Nachrichten beinhält. Diese werden dann sogleich dem Client angezeigt.

Fehlermeldungen werden beim Auftreten einfach an den SessionController übergeben mittels einer Funktion addErrorMessage(). Wird danach dem Client die Seite angezeigt, so bekommt er einen Alert mit der gespeicherten Fehlermeldung angezeigt. Sobald eine Fehlermeldung einmal angezeigt wurde, wird sie danach gelöscht.

Um nachvollziehen zu können, was welcher User gemacht hat, wird jede Aktion in die Tabelle loggings geschrieben.

Allgemeines:

Unter sources slack-light, befindet sich der PHP-code, das Apache vezeichniss sollte also auf diesen Ordner zeigen.

Unter sources/database befindet sich ein file structure.sql welches lediglich die Struktur der Datenbank enthält.
Die Datei databasedump.sql enthält sowohl Struktur der Datenbank als auch jene Daten, wie sie in den Testfällen erstellt wurden.

Die Eingefügten Benutzer sind (Benutzername/Passwort) wolfgang/wolfgang, max/max, scm4/scm4