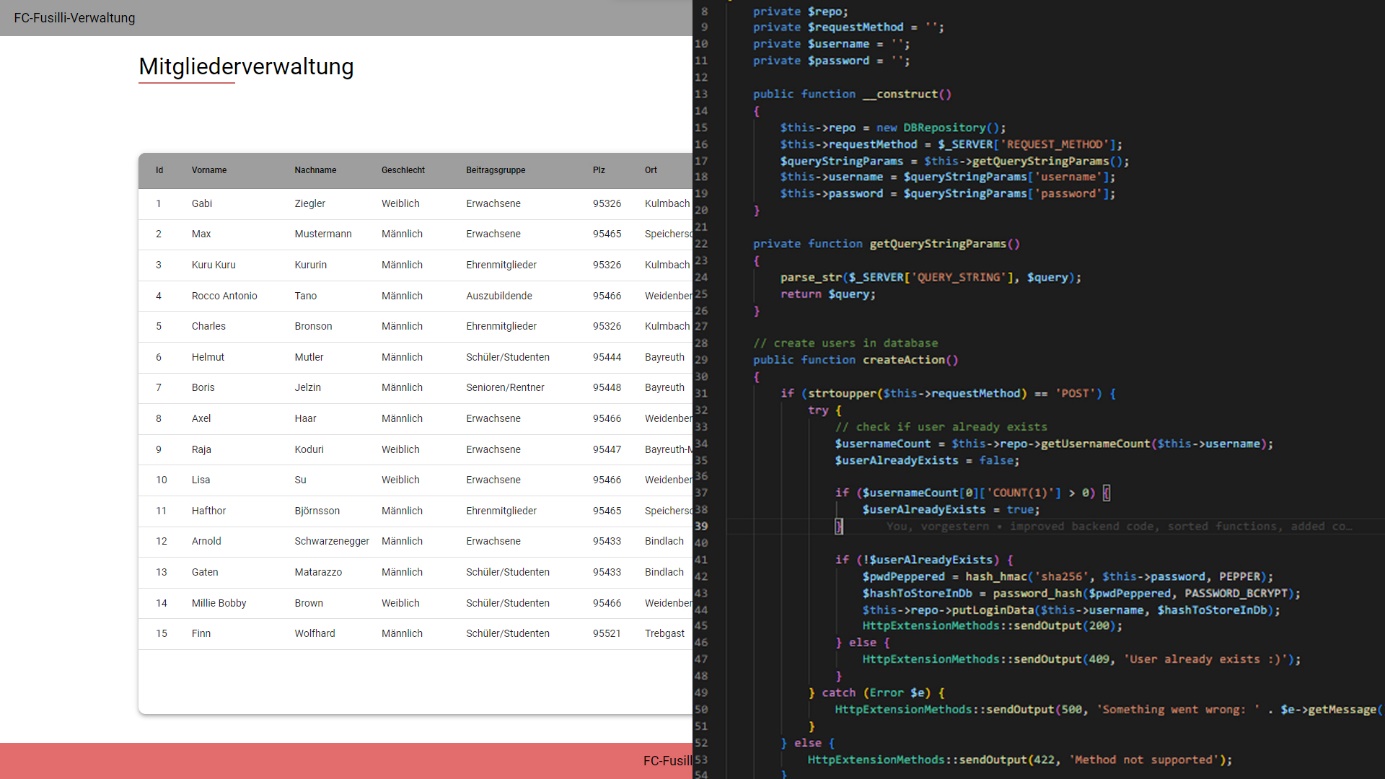
Juli - 2022

WEBBASIERTER-DATENBANKMANAGER



Gruppe 4 – Fusilli

|  |  |
| --- | --- |
| Thomas S. | @Noodle693 |
| Marco H. | @GodlesZ95 |
| Samuel S. | @Gunzli02 |
| Michael W. | @miwied |

Inhaltsverzeichnis

**Docker**

Docker ist eine Open-Source Software zur Isolierung von Anwendungen mit Hilfe von Containervirtualisierung.

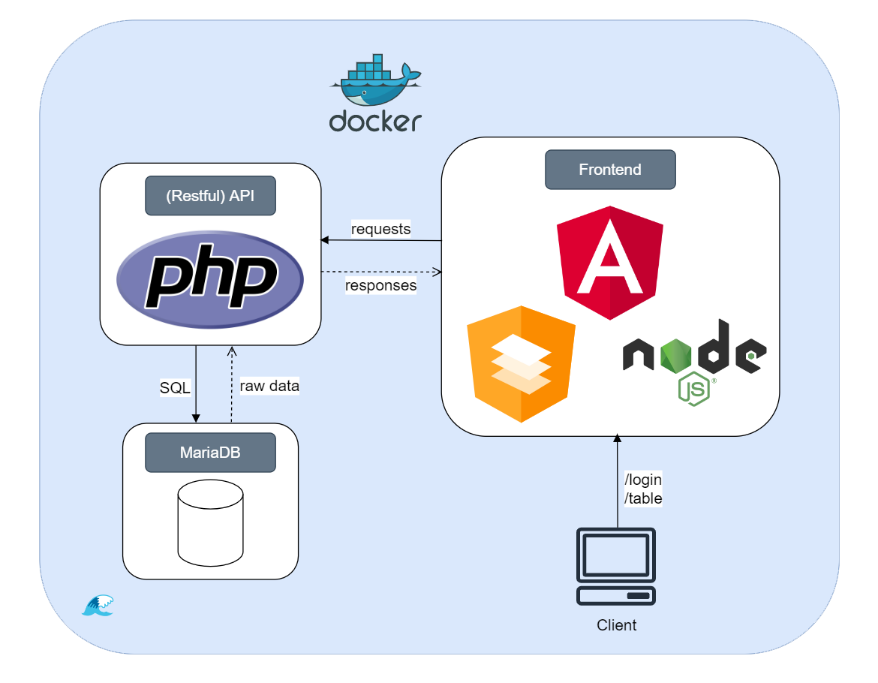
Docker vereinfacht die Bereitstellung von Anwendungen, weil sich Container, die alle nötigen Pakete enthalten, leicht als Dateien transportieren und installieren lassen.

Container gewährleisten die Trennung und Verwaltung der auf einem Rechner genutzten Ressourcen.

In unserem Projekt nutzen wir Docker, um die Anwendung nicht nur später für den Endverbraucher einfach installierbar zu machen, sondern auch um die Entwicklung effektiver und effizienter zu gestalten.

Mithilfe von Docker hat jeder Entwickler die gleichen Voraussetzungen.

Dinge wie PHP Webserver, Maria DB- und Angular-Hosting werden mithilfe von Docker bereitgestellt und müssen somit nicht über externe Programme / Lösungen zur Verfügung gestellt werden.



Unsere Infrastruktur wird somit in Docker ausgeführt und stellt sich ausfolgenden Containern zusammen:

**Angular-App**

Zum Bauen der Angular App, mithilfe von node.js

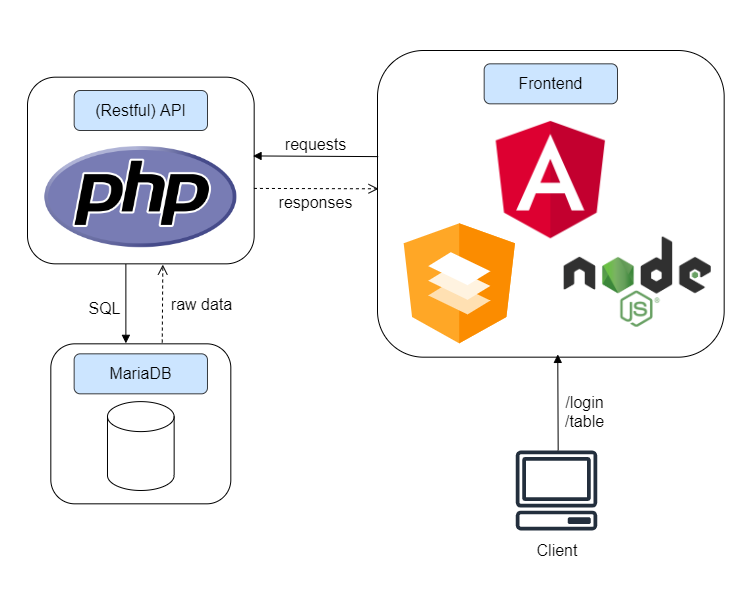
**MariaDB**

Datenbank für Sportverein- und Login-Daten

**PHP-Apache**

Webserver für das Backend

**Projekt Architektur**

****

Das **Frontend** besteht aus einer Angular Applikation bestehend aus Komponenten des Material-Designs. Node.js beinhaltet den Node-Package-Manager (NPM), welcher die notwendigen Pakete für die Applikation installiert und bereitstellt.

Das **Backend** (implementiert in php) bezieht die Daten direkt aus der Datenbank (MariaDB) welche mithilfe von PDO (Hilfslibrary für Datenbankzugriff) in strukturierte Objekte aufgeteilt werden, die jeweiligen Objekte werden dann in das JSON-Format kodiert und bei einer entsprechenden Anfrage mittels einer HTTP-API an das Frontend kommuniziert.

**Login (Backend-Seitig)**

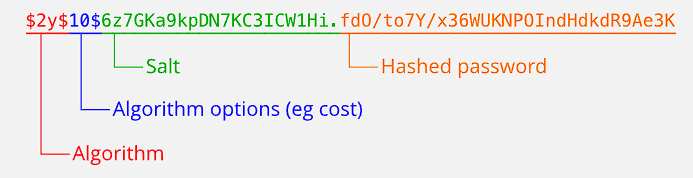
**Registrierung**

Bevor man sich anmelden kann, muss erstmal ein Nutzer angelegt bzw. registriert werden. Über das Frontend wird ein POST-Request mit dem Nutzernamen und dem Passwort als Parameter ausgeführt, hierzu wird folgende Route verwendet:

http://localhost/login.php/createUser?username=USERNAME&password=PASSWORD

Das Passwort wird dann mit der Funktion „hash\_mac()“ und einem Konfigurierbaren „pepper“ mit dem „sha256“ Algorithmus gehashed, anschließend wird dann das gehashte + gepperte Passwort mit der „password\_hash()“ Funktion nochmals gehashed und „gesalted“. Das Resultat, welches in der Datenbank gespeichert wird, sieht wie folgt aus:

S



**Algorithmus**

**Algorithmus Einstellungen**

**„Salt“**

**Doppelt gehashtes und gepeppertes Passwort**

**Anmeldung**

Will sich ein Nutzer anmelden muss dieser über das Frontend seinen Nutzernamen und das Passwort eingeben, das Frontend sendet nun einen POST-Request mit dem eingegebenen Nutzernamen und dem Passwort an das Backend und nutzt dafür diese Route:

http://localhost/login.php/getToken?username=USERNAME&password=PASSWORD

Das eingegebene Klartext-Passwort wird nun wieder gehashed + geppered und anschließend mit dem Passwort-Hash aus der Datenbank mithilfe der „password\_verify()“ Funktion verglichen, bei Richtigkeit wird ein JSON-Webtoken erstellt und der HTTP-Response übergeben.

**Schutz durch pepper und salt**

Sollte ein Angreifer zugriff zur Datenbank bekommen, wird dieser keinerlei Möglichkeiten haben die Passwörter wieder in Klartext zurückzuführen.

Aufgrund der Kombination aus gepperten und gesalteten Passwörter sind Vergleiche aus einer „Rainbow Tabelle“ nutzlos, auch Brute-Force- oder Dictionary-Attacken sind nahezu unbrauchbar bei dieser Passwortspeicherungs-Methode.