Password Safe

Inhalt

[Konzept 1](#_Toc75869851)

[Account 1](#_Toc75869852)

[Erstellung 1](#_Toc75869853)

[Login / Logout 1](#_Toc75869854)

[Dashboard 1](#_Toc75869855)

[Passwort Tabelle 1](#_Toc75869856)

[Umsetzung 3](#_Toc75869857)

[Account Sicherheit 3](#_Toc75869858)

[Passwort Sicherheit 3](#_Toc75869859)

# Konzept

## Account

### Erstellung

Wir wollen eine Applikation machen, bei der man sich mit einem Username und Passwort registrieren kann. Wenn man sich registriert hat, besitzt man einen Account und der Username kann nicht mehr vergeben werden. Der User bekommt einen JSON Web Token mitwelchem er sich für die anderen Endpoints autorisieren kann.

### Login / Logout

Wenn der Benutzer sich ausloggen will, kann er auf Logout klicken. Der JSON Web Token wird anschliessend aus dem localstorage gelöscht. Nachdem er sich ausgeloggt hat, kann er sich beim Login mit seinen zuvor gewählten Logindaten wieder einloggen. Nachdem er sich eingeloggt hat, wird im wieder ein JSON Web Token zugestellt mitwelchem er sich für die anderen Endpoints autorisieren kann.

## Dashboard

Nach dem registrieren oder login wird er auf das Dashboard weitergeleitet. Auf dem Dashboard sieht der Benutzer all seine zuvor erstellten Passwörter. Falls er keine Passwörter erstellt hat, wird in der Tabelle «No Data» angezeigt.

### Passwort Tabelle

Unterhalb der Tabelle befindet sich ein Create Knopf. Dieser öffnet einen Dialog, auf welchem man einen Passwortnamen und ein Passwort eingeben kann. Dieses wird danach mithilfe von einem Backend verschlüsselt gespeichert. Sobald der User ein Passwort erstellt hat, wird dies nun in der Tabelle angezeigt. Jedes Passwort welches in der Tabelle angezeigt wird, hat eine ID, ein Namen, ein Passwort und einen Lösch Button. Sobald die Tabelle wieder, keine Daten hat, wird die «No Data» Schaltfläche angezeigt.

# Umsetzung

## Account Sicherheit

Die Accountsicherheit gewährleisten wir mit einem Login und JWT gesicherte Endpoints. Mit diesem Login stellen wir dem User einen JWT gesigntem Token. Wir benutzen den JWT Service von NestJs.

Die Passwörter hashen wir mit Bcrypt und 10 rounds. Falls sich der user einloggen will benutzen wir die Bcrypt compare Funktion.

## Passwort Sicherheit

Die Passwortsicherheit gewährleisten wir indem wir die Passwörter vor dem abspeichern verschlüsseln mithilfe einer Symmetrischen Verschlüsselung. Für diese Verschlüsselung benutzen wir CryptoJS. Der Key der Verschlüsselung ist, die UserId und der Username. Somit können wir einen Unique key sicherstellen. Ebenfalls hat ein user ohne den JWT Token des Users kein Access auf den Endpoint, da der Key aus dem JWT Token generiert wird.

# Reflexion

Es ist uns leicht gefallen, das Frontend zu implementieren. Der Anfang vom Backend war auch super. Wir haben mit NestJS gearbeitet und das ist super toll. Jedoch bei der ganzen JWT Autorisierung hatten wir ein wenig Probleme, nicht weil es schwer war. Sondern weil unser Konzept so sicher war, dass es ziemlich komplex wurde.

Beim Verschlüsseln hatten wir ebenfalls Probleme, da es zuerst lange nicht funktioniert hat, weil wir nicht bcrypt.compare verwendet haben, sondern einen hash neu generiert haben und diese verglichen haben.