# Login auf Backendseite

Beim Login werden Username und Passwort im JSON-Format vom Client erwartet. Der Server holt sich das gehashte Passwort und den Salt aus der Datenbank. Dem übergebenen Passwort wird der Salt hinzugefügt und die Gesamtzeichenfolge gehasht. Stimmt der berechnete Hash mit dem Hash in der Datenbank überein, so war der Login erfolgreich und der Server antwortet mit dem Statuscode 200 (OK) und einem Token, mit dem er sich in zukünftigen Anfragen authentifizieren kann. Sollte der Login fehlerhaft sein, antwortet der Server mit dem Statuscode 401 (Unauthorized).

# Registrierung auf Backendseite

Die Registrierung von neuen Usern läuft ähnlich wie der Login. Der Server erwartet einen Usernamen und ein Passwort im JSON-Format vom Client. Zuerst prüft der Server ob in der Datenbank bereits ein User mit dem gleichen Usernamen vorhanden ist. Sollte es bereits einen User geben, antwortet der Server mit dem Statuscode 409 (Conflict) und einem custom Errorcode und Erklärung im JSON-Format. Sollte der Username noch nicht vergeben sein, generiert der Server eine zufällige Zeichenfolge (Salt) und fügt sie dem Passwort an. Die neue Zeichenfolge wird mittels SHA-256 gehasht und zusammen mit Usernamen und Salt in der Datenbank gespeichert. Der Server antwortet dann mit dem Statuscode 200 (OK) und sendet dem Client einen Token, mit dem er sich in zukünftigen Anfragen authentifizieren kann.

# Tokenauthentication

Für die Tokenauthentication wurde JSON Web Tokens verwendet. Ein Token besteht aus drei Parts, die im JSON-Format geschrieben werden.

1. Der Header enthält allgemeine Daten wie den verwendeten Verschlüsselungsalgorithmus und den Typ des Tokens.
2. Die Payload enthält die wichtigen Daten, die für die Authentifizierung benötigt werden oder anders nützlich sein können.
3. Bei der Verifizierungssignatur wird der Base64 encodierte Header und die Base64 encodierte Payload mit einem Punkt getrennt aneinandergeheftet und dann mit dem Secret Key verschlüsselt und anschließen gehasht.

Der Token hat dann das Format <base64 Header>.<base64 Payload>.<base64 Verifizierungssignatur>

Ein gültiger Token könnte dann z.B. so aussehen: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1lIjoiRiIsImlzcyI6Imh0dHA6Ly9sb2NhbGhvc3Q6NDQzMTYvIiwiYXVkIjoiaHR0cDovL2xvY2FsaG9zdDo0NDMxNi8ifQ.xBHxCzGO\_MSsVmJlE6XSGNw1s-tImRKXf5LjnrJPchA

In der Payload unseres Tokens wird der Name des Users, der Aussteller (in diesem Fall <http://localhost:44316/>) und die Audience (auch <http://localhost:44316/>) gespeichert.

# Reflexion

Meine erste Aufgabe war die Erstellung des Accountservers. Da ich mich mit c# und den asp.net Technologien bereits gut auskannte, nutzte ich die Möglichkeit mich mehr mit Dependency Injection(DI) und Best-Practises auseinanderzusetzten um so eine skalierbare Grundlage für den Server zu schaffen. Das Salten und Hashen von Passwörtern war mir im Vorfeld schon bekannt und erforderte kaum neues Wissen. In der Tokenauthentifizierung kannte ich mich noch nicht aus und hatte so Zeit mich mit ein Paar verschiedenen Verfahren auseinanderzusetzen.

Das Projektumfeld fand ich nicht sonderlich berauschend. In den umliegenden Räumlichkeiten wurde mehrmals die Woche lautstark gebohrt.  
Die Arbeit in der Gruppe war im Großen und Ganzen oberes Mittelmaß. Es ist des Öfteren vorgekommen, dass Gruppenmitglieder über eine längere Zeit nicht am Projekt gearbeitet haben.

Aus dem Projekt nehme ich mit, dass ich meine Motivation etwas zu Tun weniger von der Arbeitsbereitschaft der Gruppe abhängig zu machen.