Softwarestruktur (und Bestandteile?)

Das Vokabeln5 Backend Programm ist in mehrere Klassen aufgeteilt, wobei jede Klasse ein eigenes Themenfeld adressiert.

Der Zentrale Ausgangspunkt ist dabei die Klasse Startup. Sie verfügt über die einzige main-Methode, mit der auch das Programm gestartet wird. Hier werden alle initialen Schritte ausgeführt. Dazu gehört:
-das Festlegen aller Server und Datenbankparameter über die Klasse confManager
-das starten des embedded Jetty Webservers
-die Festlegung aller Spark Routen und Filter
-die Bereitstellung der Website (HTML)
-die Bereitstellung einer Methode mit der eine Datenbankverbindung aufgebaut wird

Die Klasse confManager definiert die Variablen Parameter, entweder über die parameter der Kommandozeile, oder eine Datei, oder mithilfe der Standardparameter.

Die Klasse Configs definiert lediglich die Standardparameter und lässt es deshalb nicht zu, dass von ihr Instanzen erstellt werden.

Die einzelnen Routen und Filter starten Dienste die Thematisch in den jeweiligen Klassen zusammengefasst sind.

Die Klasse UserManager ist für alle Login & Registrierungen betreffenden Themen zuständig. Dafür werden auf die Hilfsklassen User und Crypto zurückgegriffen. Zudem wird der Passwort Brute Force Schutz hier durchgeführt.

Die Klasse vocManager verwaltet alle die Vokabeln betreffenden Operationen. Dafür wird gebrauch von der Hilfsklasse Vocabulary gemacht.

Die Klasse User erlaubt es Instanzen eines Users zu erstellen, bestehend aus Email und verschlüsseltem Passwort.

Die Klasse Vocabulary erstellt eine vollständige Vokabel bestehend aus ID, Frage, Antwort, Sprache und Phase.

Die Klasse Crypto verwaltet alle JWT Operationen, sodass in Zusammenarbeit mit UserManager ein Login auch über JWT möglich ist.

Im Ordner Resources/www befinden sich die drei HTML Websites.
-index.html wird bei einem GET auf / gesendet.
-register.html wird bei GET auf /app/register gesendet (auch über index.html erreichbar).
-404.html wird bei allen nicht definierten URLs gesendet.

Verwendete Tools, Frameworks und Klassenbibliotheken

Spark Java
Spark ist ein micro Framework zum entwickeln von Webapplikationen für Kotlin und Java 8. Es erlaubt über die einfache Angabe von Lambda Expressions sehr schnell einen Webserver aufzubauen, der HTTP-Anfragen verarbeitet. Dazu sind Request und Response manipulierbare Objekte die innerhalb des Threads, welcher für jede Anfrage erstellt, weitergegeben werden, sodass alle Filter und Routen auf die gleichen Objekte zugreifen. Dadurch wird Spark zu einem überaus schnell zu programmierendem, aber dennoch mächtigem Framework.

GSON
GSON ist Googles JavaScript Object Notation Java-Library um Java Objekte in JSON zu verwandeln und JSON Strings in das passende Java Objekt. Genau dafür haben wir die Library verwendet.

PostgreSQL
PostgreSQL bezeichnet sich selbst als “The World's Most Advanced Open Source Relational Database”. In unserem Projekt haben wir eine PostgreSQL Datenbank verwendet. Dies ist insbesondere wichtig zu wissen, da PostgreSQL sich zwar größtenteils an den SQL-Standard SQL:2016 hält, es aber auch viele PostgreSQL spezifische Funktionalitäten gibt.

JDBC
Java Database Connectivity ist eine Datenbankschnittstelle der Java-Plattform um einheitlich mit Datenbanken von verschiedenen Herstellern zu kommunizieren. Wir haben dementsprechend den PostgreSQL JDBC Driver 42.2.13 verwendet, der es uns ermöglicht eine Verbindung zur Datenbank aufzubauen und Prepared-Statements zu generieren, auszuführen und uns das ResultSet anzusehen.

Gradle
Gradle ist das Build-Management-Automatisierungs-Tool mit dem wir unser Projekt entwickelt haben. Es überzeugt, durch die einfache gestaltete möglichkeit Plugins und Librarys einzubinden.

Logbag/SLF4J
Logback, der Nachfolger von Simple Logging Facade for Java, ist eine abstraktion für viele Logging Frameworks wie java.util.logging, welches wir verwendet haben. Es ermöglicht uns während der Entwicklung Live Feedback vom Server zu bekommen, welche Anfragen gestellt werden, beziehungsweise was genau auf dem Server passiert, und welche Fehler generiert werden.

Spring Security Crypto Module
Das Spring Security Crypto Module, als unabhängiger (keine Dependencies im Code) Teil des Spring Security beziehungsweise des Spring Frameworks, bietet unterstützung für symmetrische Verschlüsselung, Schlüsselgenerierung und Passwortverschlüsselung.

Letzteres haben wir mit dem BCryptPasswordEncoder benutzt um die Passwörter der User verschlüsselt in der Datenbank zu speichern und das zugesandte Passwort beim Login zu überprüfen.

Commons CLI
Die Apache Commons CLI Library bietet eine API zum parsing von Befehlszeilenoptionen. Somit konnten wir beim Programmstart per Command-line (Kommandozeile) dem Programm Variable Parameter abweichend von den Standardwerten übergeben, oder eine Hilfe-Übersicht innerhalb der Kommandozeile zur Verfügung stellen.

JJWT
JSON Web Token for Java and Android ist eine Library zum Erstellen und Verifizieren von JWTs. Mit dieser Library war es uns möglich dem User einen drei Monate gültigen, nicht fälschbaren JWT auszuhändigen, mit dem er sich anmelden kann, ohne jedesmal Email und Passwort schicken zu müssen.

JAXB
Die Java Architecture for XML Binding stellt eine API zur Verfügung, die Java Developern wie uns ein automatisches mapping zwischen XML Dokumenten und Java Objekten ermöglicht. Wir verwenden die API um den Secret Key als Base64 in ein Bytearray für den HS256 Signing-Algorithm des JWTs zu konvertieren.