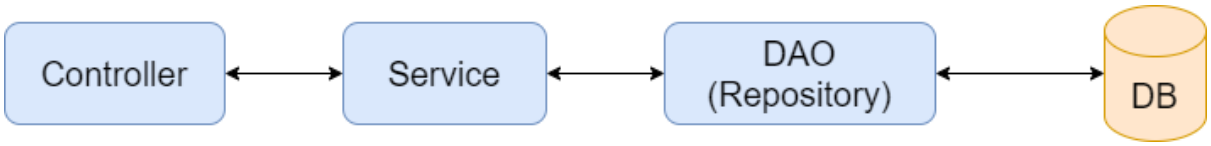
Dokumentation zu der Spring Boot Software

Sehr geehrte Damen und Herren,

die Software, die bei ihnen angekommen ist, basiert auf den Aufbau eines MVC Design Pattern in dem Spring Boot Framework, welches ein Framework von Java ist.

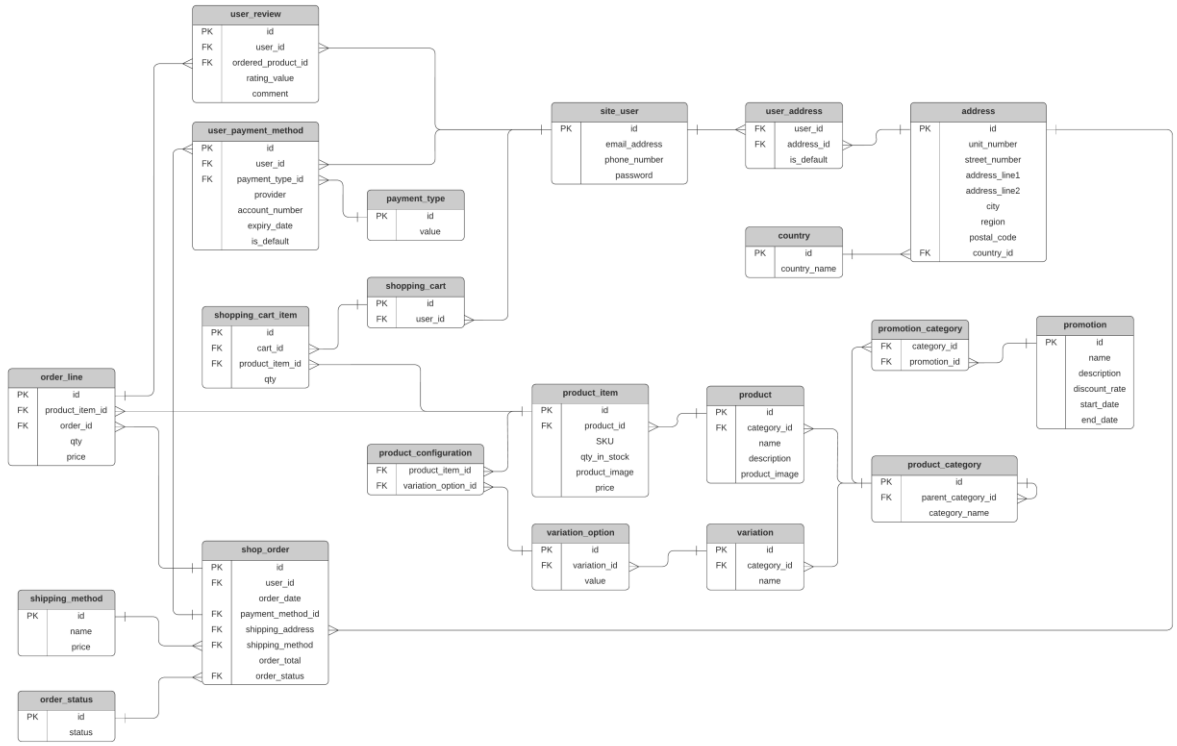
MVC wurde ausgewählt, weil es Ziel ist eine kleine CRUD-Anwendung aufzubauen, welches, das Steuern der Datenbank als Hauptziel im System hat.

Des Weiteren habe ich mich an diesem Diagramm, orientiert



Einen Frontend Framework habe ich nicht eingesetzt, die Software wurde mit Postman durch HTTP Request getestet.

Die Datenbank zu der Software.



https://dbshostedfiles.s3.us-west-2.amazonaws.com/dsa/erd_ecommerce_database.png

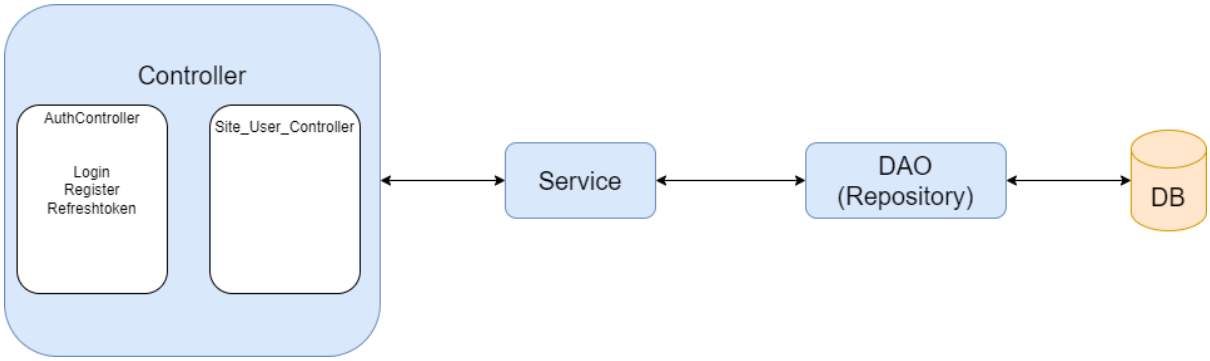
Die Datenbank auf dem Diagramm wurde in einer PostgreSQL Umgebung mit JPA in Spring Boot erstellt.

Das Projekt hat das Ziel eine E-Commerce-Software, zustande zu bringen, dabei ist es mein Ziel, den Backend mit dem Spring Boot Framework zu erstellen.

Aktuell ist sie noch nicht fertig, aber in einem Zustand, wo sie sehen können, wie die Architektur einer Web-Anwendung aussieht.

Der jetzige Zustand demonstriert nur diesen Zustand.

Auf dem unteren Diagramm kann man zwei von den vier Controllern erkennen, die ich angewendet habe.



Einer dieser Controller ist der Authentications Controller, welches durch das Login oder registrieren, einen JWT-Token erstellt und dem Header des Frontends hinzugefügt werden soll.

POST http://localhost:8088/auth/register

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "id":1,
    "vorname":"sinan",
    "nachname":"yaman",
    "email":"sinan@gmx.de",
    "phone_number":"123",
    "password":"test"
}
```

ody Cookies Headers (14) Test Results (0/1)

Pretty Raw Preview Visualize JSON


```
{
    "message": "User successfully registered.",
    "userId": 33,
    "accessToken": "Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIzMyIsImlhdCI6MTcxMzQ1MjUwMiwiZXhwIjoxNzEzNDUzMTA3fQ.
        oLRwyzNCTZhtgo6MuJlP73Ne2zSNrIWmwirY_doXau-L2IehcqBYTsMcU8TNX2yJOQ1gYebPUu3g-gbMfe-vXw",
    "refreshToken": null,
    "status": 0,
    "email": null,
    "data": null
}
```

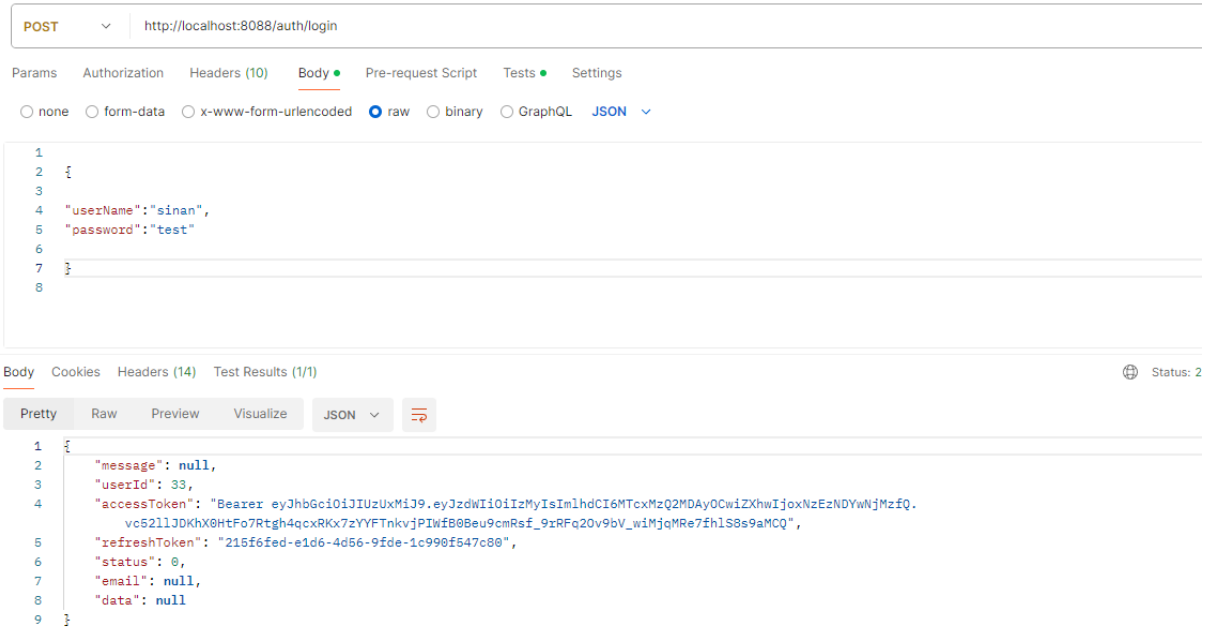
```java
@PostMapping("/register")
public ResponseEntity<AuthResponse> register(@RequestBody RegisterRequest registerRequest) {
    AuthResponse authResponse = new AuthResponse();

    if(authService.getOneUserByUserName(registerRequest.getVorname()) != null) {
        authResponse.setMessage("Username already in use.");
        return new ResponseEntity<>(authResponse, HttpStatus.BAD_REQUEST);
    }


    Site_User user = new Site_User();
    user.setVorname(registerRequest.getVorname());
    user.setNachname(registerRequest.getNachname());
    user.setEmail(registerRequest.getEmail());
    user.setPhone_number(registerRequest.getPhone_number());
    user.setPassword(passwordEncoder.encode(registerRequest.getPassword()));

    authService.saveOneUser(user);
    UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(registerRequest.getVorname(), registerRequest.getPassword());
    Authentication auth = this.authenticationManager.authenticate(authToken);

    SecurityContextHolder.getContext().setAuthentication(auth);
    String jwtToken = jwtTokenProvider.generateJwtToken(auth);
    authResponse.setMessage("User successfully registered.");
    authResponse.setAccessToken("Bearer " + jwtToken);
    authResponse.setUserId(user.getId());

    return new ResponseEntity<>(authResponse, HttpStatus.CREATED);
}
```

src/main/java/com/project/ecommerce/controllers/Authentication_Controller.java

Da ich einen großen Wert auf die Verwendung neusten Technologien lege, habe ich in diesem Projekt mit grundlegenden Themen viel Zeit verbracht.

Dazu gehören Spring Security, JWT-Token, Refresh Token und alles Notwendige, welches in der Spring Boot Umgebung notwendig ist.

Der Spring Security funktioniert nur grundlegend und ist nicht sehr weit fortgeschritten.



Beim Login erhalten wir einen refreshToken welches für das Frontend gedacht ist und bei jeder Aktion des Webservices ausgelöst werden soll, damit sich der AccesToken erneuert, andernfalls würde der Access Token ab einer bestimmten Zeit ihre Funktion verlieren.

Der Site_User_Controller bezieht sich über den Service auf die Datenbankaktivitäten der Site_User Tabelle.

Ich würde mich freuen, wenn sie weitere Ratschläge geben würden, was ich noch am Backend verändern kann und mich weiter verbessern kann