syntaxive

Version 0.1

Backend Zusatzleistungen

Berlin, 11.04.2021

1. Einleitung

In diesem Dokument werde ich potentielle Zusatzleistungen auflisten, die nicht spezifisch angefragt wurden aber aufgrund der Funktionalität hinzugefügt wurden und eventuell eine Zusatzleistung begründen könnten.

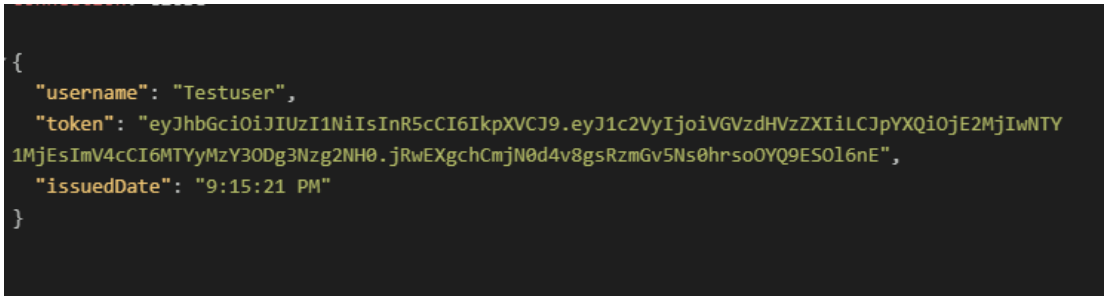
2. Refresh Token und logout

Da der JWT-Token der erzeugt wird nicht ewig gültig sein soll wird er zuerst mit einem expiration-date besehen:

const expirationTime = process.env.TIMEOUT
const expiresAt = issuedAt + (expirationTime * 1000)

gleichzeitig wird auch ein Refreshtoken erstellt mit dem man sich nicht authentifizieren kann aber verwendet werden kann, während man eingeloggt ist, um seinen JWT Token aufzufrischen durch aufrufen des Endpunktes /token der wiederum einen neuen AuthenticationToken zurückliefert. Nach verwenden des Endpunktes /logout wird dann dieser Refreshtoken von dem User gelöscht der im AuthentifizierungsToken hinterlegt ist.

Beispiel:
#Login als User mit BasicAuth
POST https://localhost:8080/authenticate/loginBasic
Authorization: Basic VGVzdHVzZXI6cGFzc3dvcmQ=



Dieser User hat dann gleichzeitig auch einen RefreshToken bekommen wie im Bild zu sehen:


```
{
  "isAdministrator": false,
  "_id": "60ae6d42df57a8545c21f093",
  "username": "Testuser",
  "password": "$2b$10$Do4W.0yj1hna6ClSFc1Ppe.by8xqM/wq4.5RhxmgX6DUE11ROqXBC",
  "email": "Testest@testmail.com",
  "preference": "Python, Javascript, Java",
  "__v": 0,
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiVGVzdHVzZXIiLCJpYXQiO
jE2MjIwNTY1MjF9.TpTeuHday68G5AXnvdmWwRnc8dOT7K7vnX-w04hMNBs"
},
```


Bevor der Token nun abläuft kann der User folgenden Endpunkt ansprechen:
POST https://localhost:8080/authenticate/token
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiVGVzdHVzZXIiLCJpYX
QiOjE2MjIwNTY1MjEsImV4cCI6MTYyMzY3ODg3Nzg2NH0.jRwEXgchCmjN0
d4v8gsRzmGv5Ns0hrsoOYQ9ESOI6nE

und bekommt somit einen aufgefrischten Token zurück

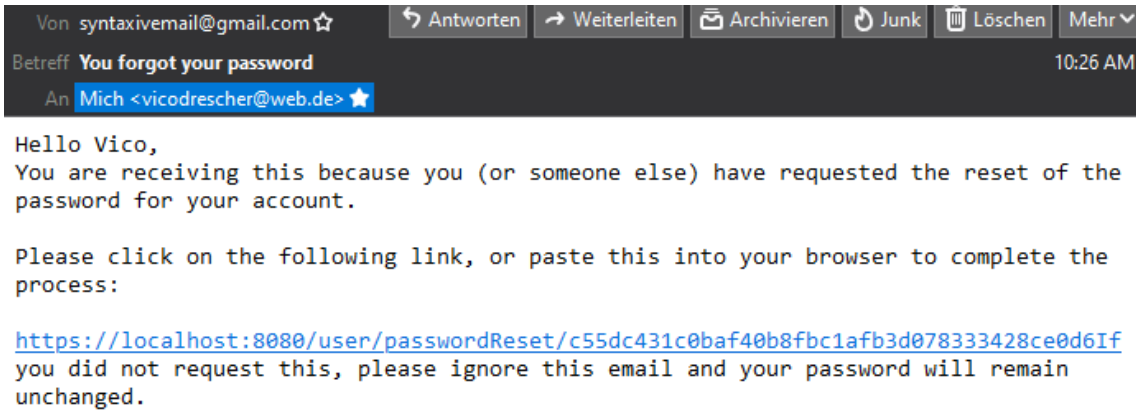

```
"refreshedToken: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2MjIwNTY1ODgsImV4cCI
6MTYyMzY3ODk0NTQyNn0.QTTqxfWvFdphj6R-H0U3uLaiaYlT2EGhrQ0miQqMqok"
```


3. Passwort ändern

Zum ändern des Passwortes bekommt der User durch triggern des Endpunktes /user/forgot eine Email wie folgt:
POST https://localhost:8080/user/forgot
Content-Type: application/json

```
{
   "email": "redacted"
}
```

daraufhin wird in userService.js die function forgotPassword getriggered die einen resetToken per crypto generiert und dem User hinzufügt mit einem Expiration date. Desweiteren wird ein Link gebaut der dann angesprochen werden kann und den resetToken beinhaltet. Der User wird dann gesucht und eine Email wird zu dem User geschickt die den Link beinhaltet



Nach aufrufen des Endpunktes passwordReset wie folgt:
POST
https://localhost:8080/user/passwordReset/4bed1d889df208a525be31110badd7a3b4218faf
Content-Type: application/json

```
{
   "password" : "hund123"
}
```

wird das Passwort zu hund123 geändert und durch die pre save method per bcrypt gehashed und dann abgespeichert.