LU06b - Login & Authentication

Login

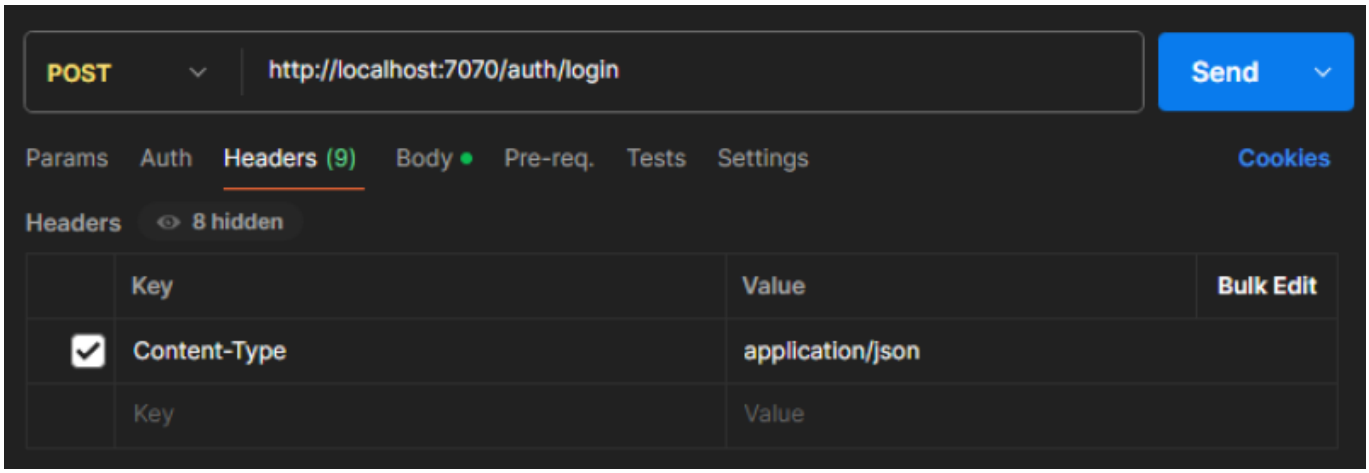
Anforderung 6: Ein Benutzer in der Datenbank soll sich via `/auth/login` anmelden können.

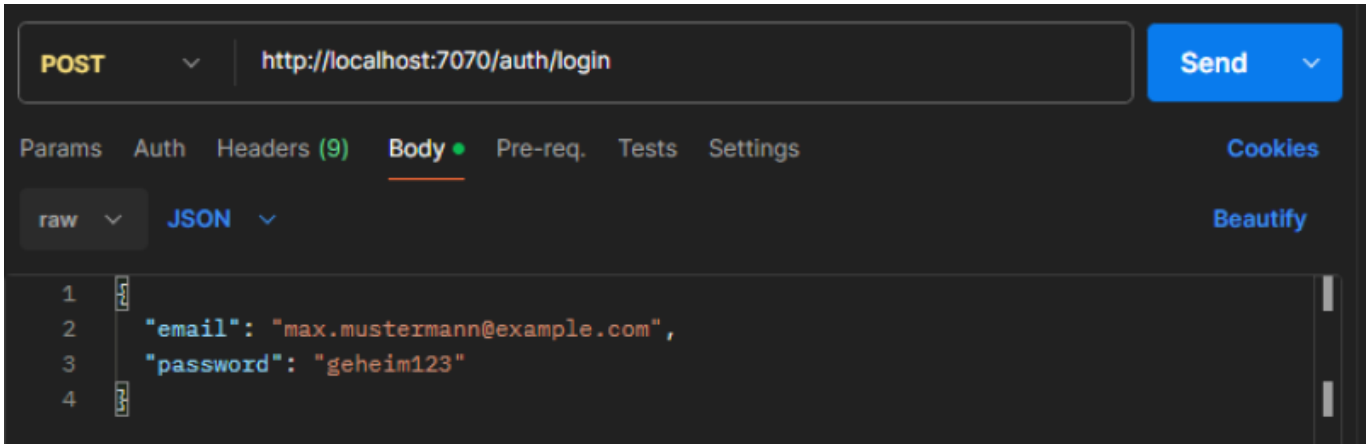
Sie erhalten die JWT-Funktionalitäten mit dem Commit `07b12a2`. Implementieren Sie eine POST-API, welche als JSON-Body die E-Mailadresse („email“) und das Klartextpasswort („password“) akzeptiert.

Die Implementation könnte grob so aussehen:

```
var json = ctx.bodyValidator(Map.class)
    .check(m -> m.containsKey("email"), "email is required")
    .check(m -> m.containsKey("password"), "password is required")
    .get();

String inputEmail = (String) json.get("email");
String inputPassword = (String) json.get("password");

... // TODO: find user in database with given email

byte[] storedSalt = Base64.getDecoder().decode(user.getPasswordSalt());
byte[] storedHash = Base64.getDecoder().decode(user.getPasswordHash());

if (PasswordHandler.verifyPassword(inputPassword, storedHash, storedSalt)) {
    String jwt = JwtHandler.createJwt(inputEmail, user.getId());
    ctx.json(Map.of("token", jwt));
    return;
}

...

// TODO: Use the same error message if the user is not found and if the
password is wrong
ctx.status(401).json(Map.of("error", "Invalid email or password"));
```

Mit Postman können Sie Ihre API testen.



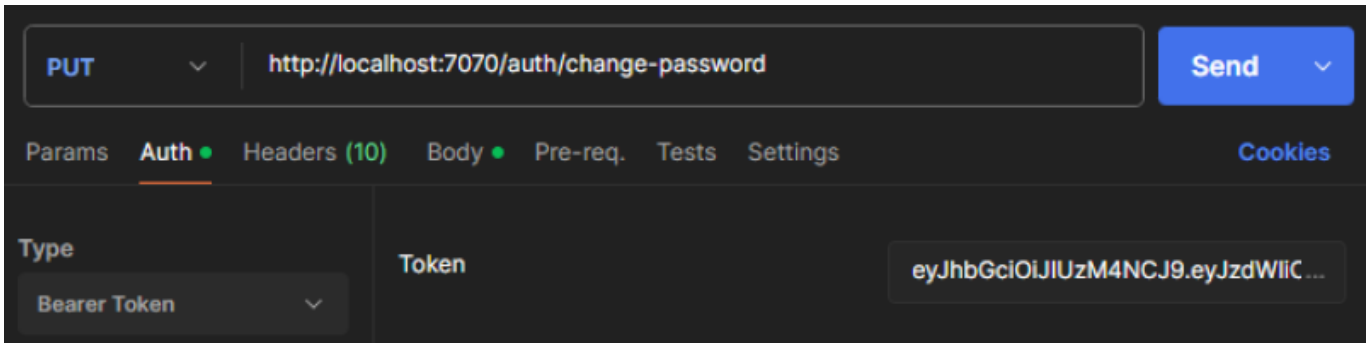


Analysieren Sie das zurückgelieferte JWT auf https://www.jwt.io/

Authentication

Anforderung 7: Ein Benutzer mit einem gültigen JWT soll sein Passwort und via `/auth/change-password` ändern können können.

Implementieren Sie eine PUT-API, welche als JSON-Body das alte Passwort („oldPassword“) und das neue Passwort („newPassword“) akzeptiert. Bei der Authorization wird ein zuvor generiertes JWT mitgegeben. Der „Content-Type“-Header ist gleich wie bei der POST-API.



Die Implementation könnte grob so aussehen:

```
String authHeader = ctx.header("Authorization");
```

```
...
String token = authHeader.substring("Bearer ".length());
...
userId = JwtHandler.getUserIdFromJwt(token); // This also includes the
verification
...

var json = ctx.bodyValidator(Map.class)
    .check(m -> m.containsKey("oldPassword"), "oldPassword is required")
    .check(m -> m.containsKey("newPassword"), "newPassword is required")
    .get();

String oldPassword = (String) json.get("oldPassword");
String newPassword = (String) json.get("newPassword");

... // TODO: find user by id

byte[] storedSalt = Base64.getDecoder().decode(user.getPasswordSalt());
byte[] storedHash = Base64.getDecoder().decode(user.getPasswordHash());
...
if (PasswordHandler.verifyPassword(oldPassword, storedHash, storedSalt)) {
    byte[] newHash = PasswordHandler.hashPassword(newPassword, storedSalt);
    user.setPasswordHash(Base64.getEncoder().encodeToString(newHash));

    userPersistor.save(user);
    ctx.json(Map.of("message", "Password changed successfully"));
}
...
```

Testen Sie die Implementation manuell mit Postman.

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/ffit/3-jahr/java/learningunits/lu06/b?rev=1758583621

Last update: **2025/09/23 01:27**