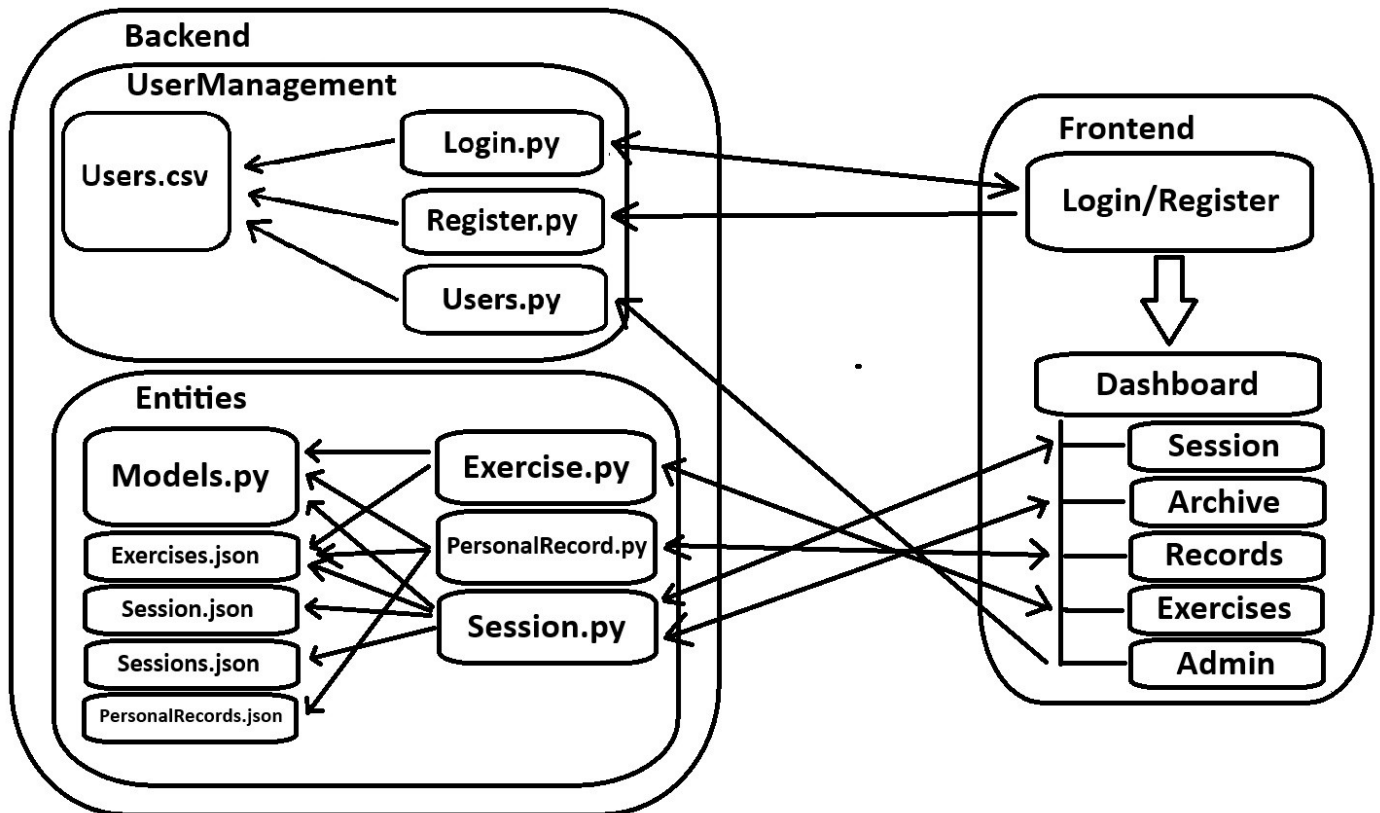


# Dokumentation: Workout-Tracker

## 1. Datenmodell



## 2. Rollen-Konzept

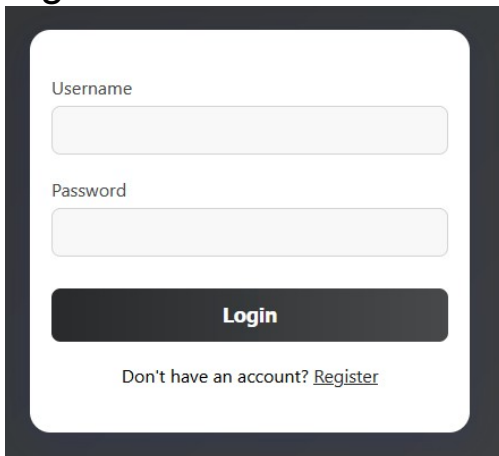
Rollen werden über Secure Session Cookies authentifiziert. Zum Login-Zeitpunkt werden die Rollen des Nutzers aus `Users.csv` gelesen und im Cookie gespeichert. Als Sicherheitsmaßnahme hat jeder Cookie eine begrenzte Lebenszeit von 3600 Sekunden (also einer Stunde) und wird zusätzlich beim Logout invalidiert. Der Cookie wird nie an den Client übergeben, sondern ist im Server (InMemory) gespeichert und wird auch nur dort geprüft. Siehe Screenshot:

```
roles_list = json.loads(user["Roles"])
session_token = secrets.token_urlsafe(32)
session_store[session_token] = {
    "username": request.username,
    "roles": roles_list
}

response = JSONResponse({"message": "Login successful"})
response.set_cookie(
    key="session_token",
    value=session_token,
    httponly=True,
    secure=True,
    samesite="strict",
    max_age=3600
)
```

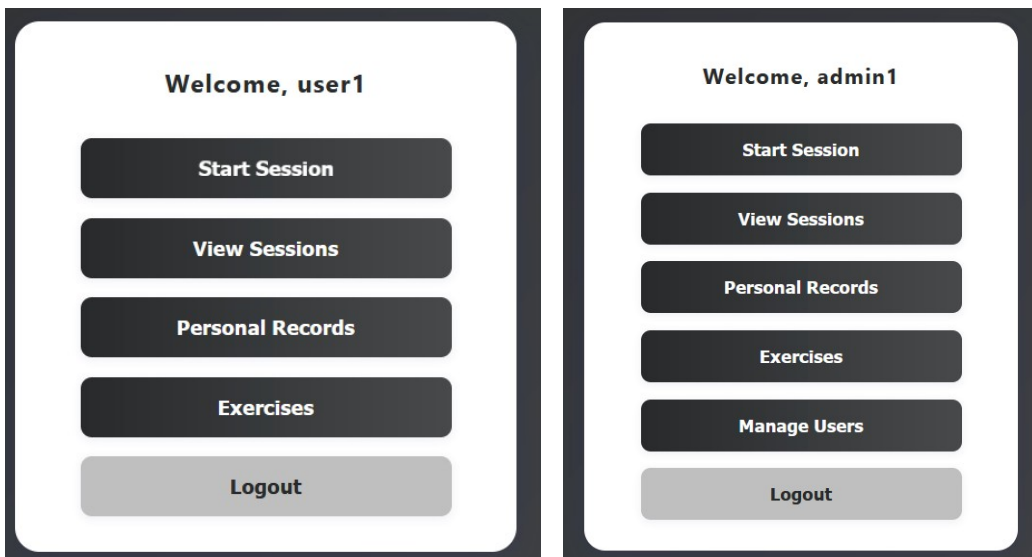
### 3. Screenshots

Login:



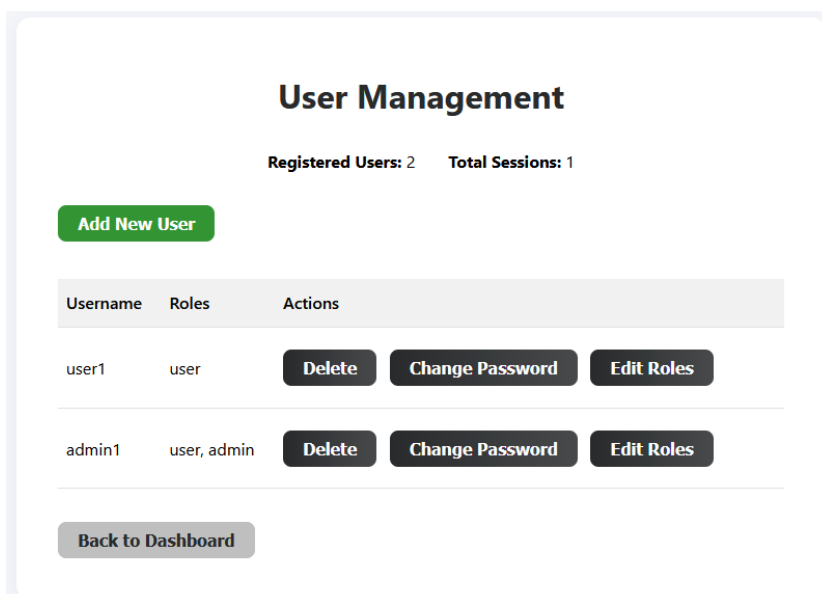
A login form with a dark background. It contains two input fields: 'Username' and 'Password'. Below the fields is a dark 'Login' button. At the bottom, there is a link that says 'Don't have an account? [Register](#)'.

Dashboard:



Two dashboard screenshots side-by-side. The left one is for 'user1' and the right one is for 'admin1'. Both show a 'Welcome' message and a list of buttons: 'Start Session', 'View Sessions', 'Personal Records', 'Exercises', and 'Logout'. The 'Logout' button is disabled (greyed out).

Admin Menü:



A screenshot of the 'User Management' admin menu. It features a title 'User Management', statistics 'Registered Users: 2' and 'Total Sessions: 1', and a green 'Add New User' button. Below is a table with columns 'Username', 'Roles', and 'Actions'. The table lists two users: 'user1' (role: user) and 'admin1' (role: user, admin). Each user has three action buttons: 'Delete', 'Change Password', and 'Edit Roles'. At the bottom is a 'Back to Dashboard' button.

Username	Roles	Actions
user1	user	<a href="#">Delete</a> <a href="#">Change Password</a> <a href="#">Edit Roles</a>
admin1	user, admin	<a href="#">Delete</a> <a href="#">Change Password</a> <a href="#">Edit Roles</a>

## 4. RegEx-Validierung

Sämtlicher User-Input wird im Frontend Javascript und im Backend statt, um gleichermaßen für Sicherheit und User-Experience zu sorgen. Die Expressions wurden sinnvoll gewählt, sodass z.B. Satzzeichen in der Beschreibung einer Übung erlaubt sind, aber nicht im Name. Zusätzlich werden Typen validiert, sodass die Anzahl der Wiederholungen z.B. einen Integer fordert.

Hier einige Beispiele:

- Username

```
function isValidUsername(str) {  
    return /^[A-Za-z0-9]+$/.test(str);  
}
```

- Beschreibung der Übung

```
field_pattern = re.compile(r"^[A-Za-z0-9\s.,!?:'\\"()\\-\\[\\]]+$")
```

## 5. Bekannte Einschränkungen

Es wird keine vollwertige Datenbank verwendet, sondern sämtliche Informationen werden in csv und json gespeichert, was für große Datenmengen sehr unpraktisch wäre. Außerdem ist das Projekt nicht dafür geeignet mehrere Nutzer gleichzeitig zu bedienen.

## 6. Lessons Learned

- Objekt-orientiertes Programmieren durch Router und Auslagerung der Klassen in eine „Models.py“-Datei, um Deklarierungen von Implementierungen zu trennen und Modularisierung zu ermöglichen.
- Umsetzung von Cookies und Speicherung im Server statt im Client.
- Authentifizierung und Zugriffsverwaltung.
- Erstellung und Zugriff auf APIs.
- Versionskontrolle durch Git und Docker.