

# Login-API

Entwicklung einer RESTful API zur Registrierung und  
dem Login von Benutzern mit OAuth2 als Container-Image



ZANCOR

# Inhalt

- Vorstellung
- Unternehmen
- Projektbeginn
- Planung
- Entwurf
- Durchfuehrung
- Qualitaetsmanagement
- Uebergabe
- Fazit



# Vorstellung

- Daniel Schwarz
- 06.02.1985
- Berlin
- Buerokaufmann
- Angeln, Gaming

Github: @Axlken

The screenshot shows the GitHub profile for user **Axlken**. The profile features a large circular profile picture of a red and orange eye. Below the picture, the name **Axlken** is displayed twice. The profile includes sections for **Popular repositories**, **Highlights** (with a PRO badge), **Organizations** (listing a single organization icon), and **Contribution activity** (showing a heatmap of contributions from February 2024). The sidebar on the right allows users to **Customize your pins** and shows a timeline of years from 2017 to 2024.

Popular repositories

- ostree-pitti-workstation** Forked from [martinpitti/ostree-pitti-workstation](#) Public Python 1 star
- lookatme** Forked from [d0c-savage/lookatme](#) Public Python 1 star
- lookatme.contrib.image\_ueberzug** Forked from [d0c-savage/lookatme.contrib.image\\_ueberzug](#) Public Python 1 star
- projects\_IT33** My small Projects at Cimdata Public Shell
- pandoc-latex-template** Forked from [Wandmalfarbe/pandoc-latex-template](#) Public TeX
- neorg** Forked from [nvim-neorg/neorg](#) Public Lua

Axklen

Edit profile

1 follower • 0 following

Germany, Berlin

Highlights

PRO

Organizations

Contribution settings ▾

298 contributions in the last year

Contribution activity

January 2024

Axklen has no activity yet for this period.

Show more activity

Seeing something unexpected? Take a look at the [GitHub profile guide](#).

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Customize your pins

2024

2023

2022

2021

2020

2019

2018

2017

# Praktikumsunternehmen

ZANCOR Consulting & Development UG  
Rigaer Straße 49  
10247 Berlin

[www.zancor.de](http://www.zancor.de)

- IT-Softwareprojekte
- Consulting
- Schulungen



## ÜBER UNS

### Herausforderungen & Bewältigung dieser ist unser Leben



#### UNSERE VISION

Wir begleiten die Unternehmen in ihre digitale Zukunft. Mit unserem exzellenten und umfassenden technologischen Know-how sind wir ein starker und kompetenter Partner für die Unternehmen. Wir kennen die technologischen Trends, entwickeln digitale und innovative Lösungen und setzen diese gemeinsam mit den Unternehmen um - individuell und bedarfsoorientiert.



#### UNSERE MISSION

Digitalisierung, Innovation und Technologien sind unsere DNA. Wir erfassen die Probleme unserer Kunden, geben strategische Orientierung und setzen IT-Projekte lösungsorientiert um kompetent um. Unser Know-how geben wir gern an die Unternehmen weiter und leisten mit Leidenschaft und persönlichem Antrieb einen entscheidenden Beitrag für die Sicherung der Wettbewerbsfähigkeit für die digitale Zukunft. Kurzum: wir als unterstützende Kraft auf Ihrem Weg zum Erfolg!



#### SICHERHEIT

Unsere Kunden können sich darauf verlassen, dass ihre digitalen Systeme und damit Ihre Daten geschützt und sicher sind. Datensicherheit genießt höchste Priorität.



#### ZUVERLÄSSIGKEIT

Unsere Systeme sind auf Langfristigkeit und Stabilität ausgerichtet und überfüllen die Erwartungen unserer Kunden.



#### LEISTUNGSFÄHIGKEIT



#### ZUSAMMENARBEIT

# Projektbeginn

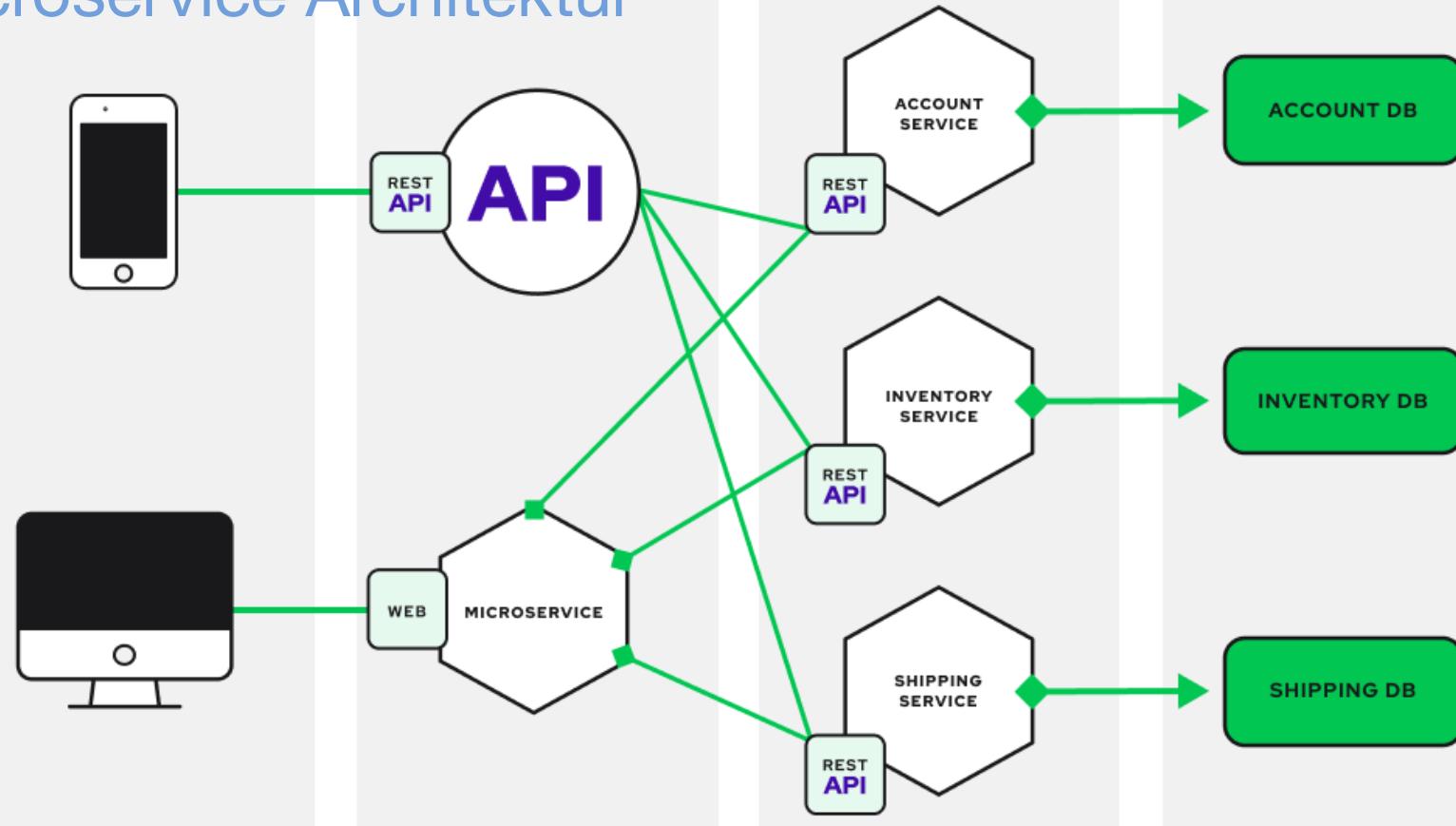
Eine immer wiederkehrende Aufgabe

# Das Problem

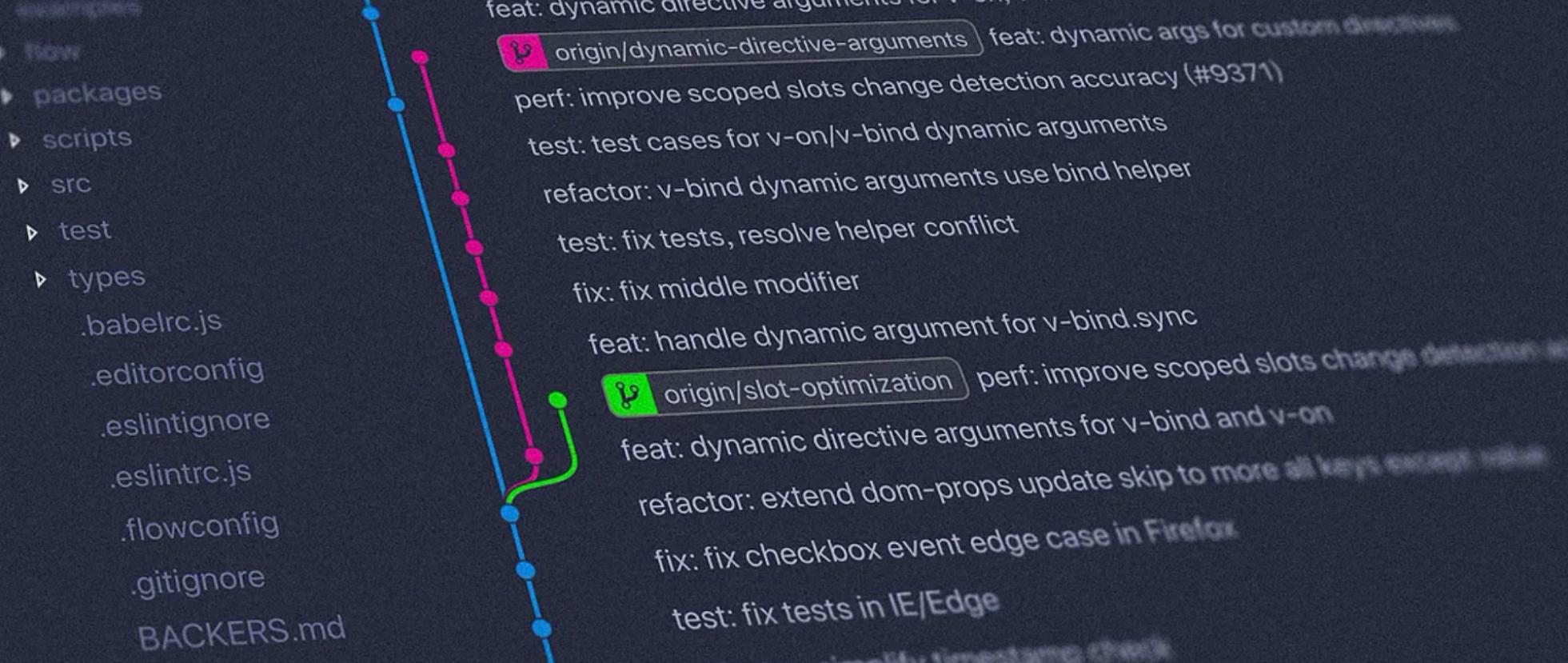
- Login Modul in fast jedem Projekt
- Monolithische Projekte
- keine Versionsverwaltung
- wechselnde Entwickler



# Microservice Architektur



# Versionsverwaltung



# Ziele

- RESTful API
- Geltende Sicherheitsstandards
- Email-Verifizierung
- Daten in Datenbank Speichern
- Containerimage
- Codequalität
- Versionskontrolle
- auf Cloudinstanz bereitstellen

# Planung

Das Wann und Womit

# Planung

- 06.11. - 24.11.2024
- max. 80h
- Homeoffice
- keine Bereitstellungen

# Phasen

Projektphasen	Zeit (in h)
Analyse/Planung	8
Entwurf	9
Implementierung	39
Abnahme	4
Dokumentation	13
<b>Gesamt</b>	<b>73</b>

# Kosten

Stelle	Kosten
Entwickler	1499,42€
Geschaeftsleiter	354,84€
Cloud-Server	2,51€
<b>Gesamt</b>	<b>1856,77€</b>

# Armotisation

- 10 Projekte im Jahr 2022
- Kosten: 1643,20€
- Zukünftige Kosten: 410,80€
- Ersparnis: 1232,40€/Jahr

# Amortisationsdauer

18 Monate

# Nutzwert API

Kriterium	Gewicht	FastAPI		Django		Flask	
		Punkte	Gewichtet	Punkte	Gewichtet	Punkte	Gewichtet
Performance	40%	10	4	6	2,4	8	3,2
Wartbarkeit	30%	9	2,7	5	1,5	6	1,8
Nützlichkeit	30%	10	3	10	3	9	2,7
Summe			9,7		6,9		7,8

# Entwurf

Die Eckpfeiler

# Datenmodell

*Users(id, email<sub>NN</sub>, hashed\_pwd<sub>NN</sub>, username<sub>NN</sub>, is\_activated<sub>NN</sub>, created\_at<sub>NN</sub>)*

# Endpunkte

- /register -> POST - Benutzerregistrierung - offen
- /login -> POST - Benutzerlogin - offen
- /user/change\_email -> POST - Benutzer Email ändern - abgesichert
- /users -> GET - Erhalte eine Liste aller Nutzer - abgesichert
- /user/by\_mail -> GET - Erhalte einen Benutzer mit EMail Angabe - abgesichert
- /user/change\_pwd -> PUT - Benutzerpasswort erneuern - abgesichert
- /user -> DELETE - Lösche einen Benutzer - abgesichert

# Qualitätssichernde Maßnahmen

- Unitests
- Linter
- Formatter
- Typechecking
- Dependency-Management
- automatisiert
- CI/CD
- sichere Verschlüsselung



# Implementierung

Hier gehts zur Sache

# Ablauf

- Github-Repositories
- CI / automatisiertes Testen
- Cloud-Instanz
- Container-Images Datenbank & API Service
- REST-API
  - Module Installieren
  - FastAPI
  - Endpunkte
  - OAuth2 Authentifizierung
  - ORM
  - Password-Hashing
  - Unitests



# Continuous Integration

- pytest - Unitest
- deptry - Coverageauswertung
- mypy - Statischer-Typen-Checker
- tox - kompatibilitaetstest
- ruff - Linter & Formatter
- git-hooks - automatisierung

The screenshot shows a GitHub Actions workflow named 'fixed ENV error #23'. The workflow consists of several jobs:

- quality**:
  - tox (3.8)
  - tox (3.9)
  - tox (3.10)
  - tox (3.11)
  - check-docs
  - build-and-push-image

Each job has a detailed log view. The 'quality' job log shows the following steps and outcomes:

```
1  ↗ Run make check
14 ↗ Checking Poetry lock file consistency with 'pyproject.toml': Running poetry lock --check
15 All set!
16 ↗ Linting code: Running pre-commit
17 [INFO] Installing environment for https://github.com/pre-commit/pre-commit-hooks.
18 [INFO] Once installed this environment will be reused.
19 [INFO] This may take a few minutes...
20 [INFO] Installing environment for https://github.com/charliermarsh/ruff-pre-commit.
21 [INFO] Once installed this environment will be reused.
22 [INFO] This may take a few minutes...
23 [INFO] Installing environment for https://github.com/nsf/black.
24 [INFO] Once installed this environment will be reused.
25 [INFO] This may take a few minutes...
26 check for case conflicts.....Passed
27 check for merge conflicts.....Passed
28 check toml.....Passed
29 check yaml.....Passed
30 fix end of files.....Passed
31 trim trailing whitespace.....Passed
32 ruff.....Passed
33 black.....Passed
34 prettier.....Passed
35 ↗ Static type checking: Running mypy
36 Success: no issues found in 10 source files
37 ↗ Checking for obsolete dependencies: Running deptry
38 Scanning 10 files...
39
40 Success! No dependency issues found.
```

Other sections of the workflow include 'Run details', 'Usage', and 'Workflow file'.

# Container-Image

```
1  # syntax=docker/dockerfile:1
2  FROM python:3.11-slim-buster
3  ENV POETRY_VERSION=1.7.1 \
4  POETRY_VIRTUALENVS_CREATE=false \
5  POSTGRE_USER=auchGeheim \
6  POSTGRE_PASS=geheim \
7  POSTGRE_DB=postgres \
8  SECRET_KEY=sehrGeheimerKeyDerMitPythonSecretsErstelltWurde
9  ENV URL_DATABASE=postgresql://$POSTGRE_USER:$POSTGRE_PASS@localhost
10 :5432/$POSTGRE_DB
11 # Install poetry
12 RUN pip install "poetry==$POETRY_VERSION"
13 # Copy only requirements to cache them in docker layer
14 WORKDIR /code
15 COPY poetry.lock pyproject.toml /code/
16 # Project initialization:
17 RUN poetry install --no-interaction --no-ansi --no-root --no-dev
18 # Copy Python code to the Docker image
19 COPY login_api /code/login_api/
20 CMD [ "uvicorn", "login_api.main:app", "--reload", "--port", "8000", "
21 --host", "0.0.0.0"]
22
```

# Endpunkte

```
1   ...
2   @app.get("/users", tags=["user"])
3   async def get_all_users() → None:
4       pass
5   @app.get("/user/by_mail", tags=["user"])
6   async def get_user_by_mail() → None:
7       pass
8   @app.post("/register", tags=["user"])
9   async def register_user() → None:
10      pass
11  @app.post("/login", tags=["user"])
12  async def login_user() → None:
13      pass
14  @app.put("/user/change_email", tags=["user"])
15  async def update_email() → None:
16      pass
17  ...
18
```

# Oauth2 - `auth\_handler.py`

```
1  ...
2  oauth2_scheme = OAuth2PasswordBearer(tokenUrl="login")
3
4  SECRET_KEY = str.getenv("SECRET_KEY")
5  ALGORITHM = "HS256"
6  ACCESS_TOKEN_EXPIRE_MINUTES = 30
7
8  # data = {"email": user_email}
9  def create_access_token(data: dict, expires_delta: Union[timedelta, None] = None) → str:
10     to_encode_data = data.copy()
11     if expires_delta:
12         expires = datetime.utcnow() + expires_delta
13     else:
14         expires = datetime.now() + timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
15     to_encode_data.update({"exp": expires})
16     # Accesstoken erstellen
17     encoded_jwt = jwt.encode(to_encode_data, SECRET_KEY, algorithm=ALGORITHM)
18     return encoded_jwt
19 ...
20 ...
```

# ORM - `models.py`

```
1  from datetime import datetime
2
3  from sqlalchemy import Boolean, DateTime, Integer, String
4  from sqlalchemy.orm import DeclarativeBase, mapped_column
5
6  class Base(DeclarativeBase):
7      pass
8
9  class User(Base):
10      __tablename__ = "users"
11
12      id = mapped_column(Integer, primary_key=True, index=True)
13      email = mapped_column(String, unique=True)
14      username = mapped_column(String, unique=True)
15      hashed_pwd = mapped_column(String)
16      is_activated = mapped_column(Boolean, default=False)
17      created_at = mapped_column(DateTime, default=datetime.now())
18
```

# Unit-Tests - `test\_main.py`

```
1  def test_change_email_no_auth():
2      response = client.put("/user/change_email", json={"new_email": "roland@peter.de"})
3      assert response.json() == {"detail": "Not authenticated"}
4      assert response.status_code == 401
5
6
7  # test Userendpoints
8  def test_get_all_users_with_auth():
9      response = client.get("/users", headers=access_token_header)
10     assert response.status_code == 200
11     response_data = response.json()
12     assert response_data[-1]["email"] == TEST_EMAIL
13
14
15 def test_get_user_by_mail_with_auth():
16     url = f"/user/by_mail?email={TEST_EMAIL}"
17     response = client.get(url, headers=access_token_header)
18     assert response.status_code == 200
19     response_data = response.json()
20     assert response_data["email"] == TEST_EMAIL
21     assert response_data["username"] == TEST_USERNAME
22
```

# Projektsruktur

```
Documents/IHK-Projekt/login-api main • lt --level=4 --icons
.
├── docs
│   ├── index.md
│   └── modules.md
├── login_api
│   ├── auth
│   │   ├── __init__.py
│   │   ├── auth_handler.py
│   │   └── pwd_handler.py
│   ├── db
│   │   ├── __init__.py
│   │   ├── crud.py
│   │   ├── database.py
│   │   ├── models.py
│   │   └── schemas.py
│   ├── __init__.py
│   └── main.py
└── tests
    ├── config.py
    └── test_main.py
! codecov.yaml
CONTRIBUTING.md
Dockerfile
LICENSE
Makefile
! mkdocs.yml
poetry.lock
poetry.toml
pyproject.toml
README.md
tox.ini
```

# Codecovage

Axklen / 🔒 login-api / 🏷 main

Coverage Flags Commits Pulls Settings

Branch Context

main ▾

Coverage on branch

80.88%

165 of 204 lines covered

7 Days ▾ trend

+80.88%

Source: latest commit da323d8

Show Chart

Code tree File list 7 total files login-api

Search for files

Files	Tracked lines	Covered	Partial	↓ Missed	Coverage %
login_api/auth/auth_handler.py	40	24	2	14	<div style="width: 60.0%;"><div style="width: 60.0%;">60.00%</div></div>
login_api/main.py	73	59	7	7	<div style="width: 80.82%;"><div style="width: 80.82%;">80.82%</div></div>
login_api/db/database.py	13	9	0	4	<div style="width: 69.23%;"><div style="width: 69.23%;">69.23%</div></div>
login_api/db/crud.py	47	42	2	3	<div style="width: 89.36%;"><div style="width: 89.36%;">89.36%</div></div>
login_api/auth/pwd_handler.py	6	6	0	0	<div style="width: 100.00%;"><div style="width: 100.00%;">100.00%</div></div>
login_api/db/models.py	13	13	0	0	<div style="width: 100.00%;"><div style="width: 100.00%;">100.00%</div></div>
login_api/db/schemas.py	12	12	0	0	<div style="width: 100.00%;"><div style="width: 100.00%;">100.00%</div></div>

# Testen & Bereitstellen

Die Zitter-Phase

# Manualles Testen - `swagger-UI`

**user**

GET /users Get All Users

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/users' \
-H 'accept: application/json'
```

Request URL

```
http://localhost:8080/users
```

Server response

Code Details

*Undocumented* TypeError: NetworkError when attempting to fetch resource.

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

GET /user/by\_mail Get User By Mail

POST /register Register User

POST /login Login User

# Bereitstellung

- Containerimage via Github-Package-Registry
- PostgreSQL Container
- Login\_API Container
- System-D Service
- Portweiterleitung Cloudfirewall
- Erreichbarkeit sicherstellen

# Uebergabe

# Fazit

# Danksagung