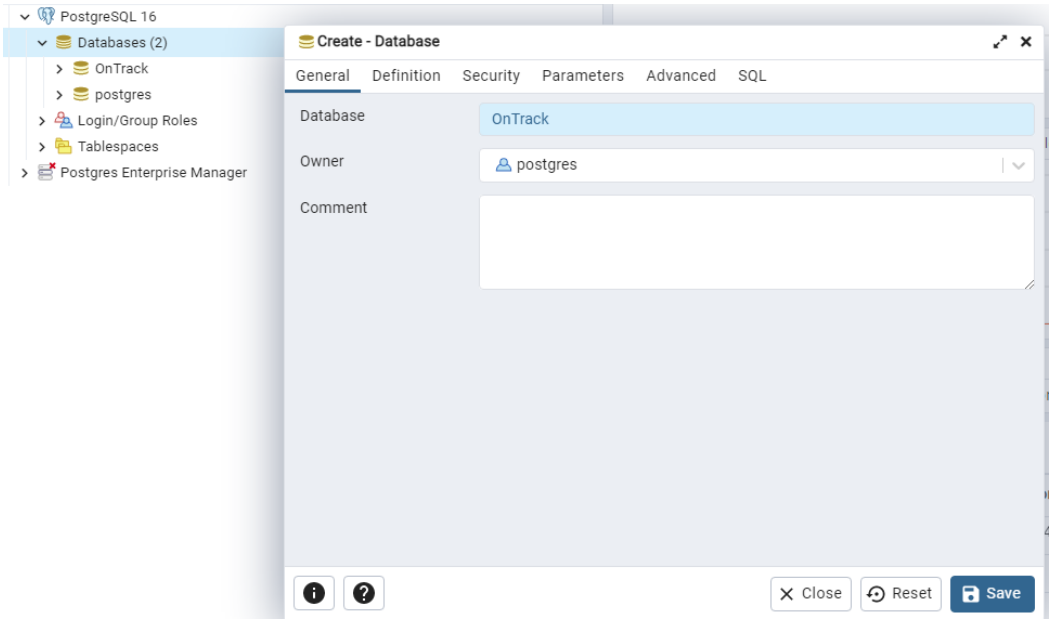
OnTrack használata

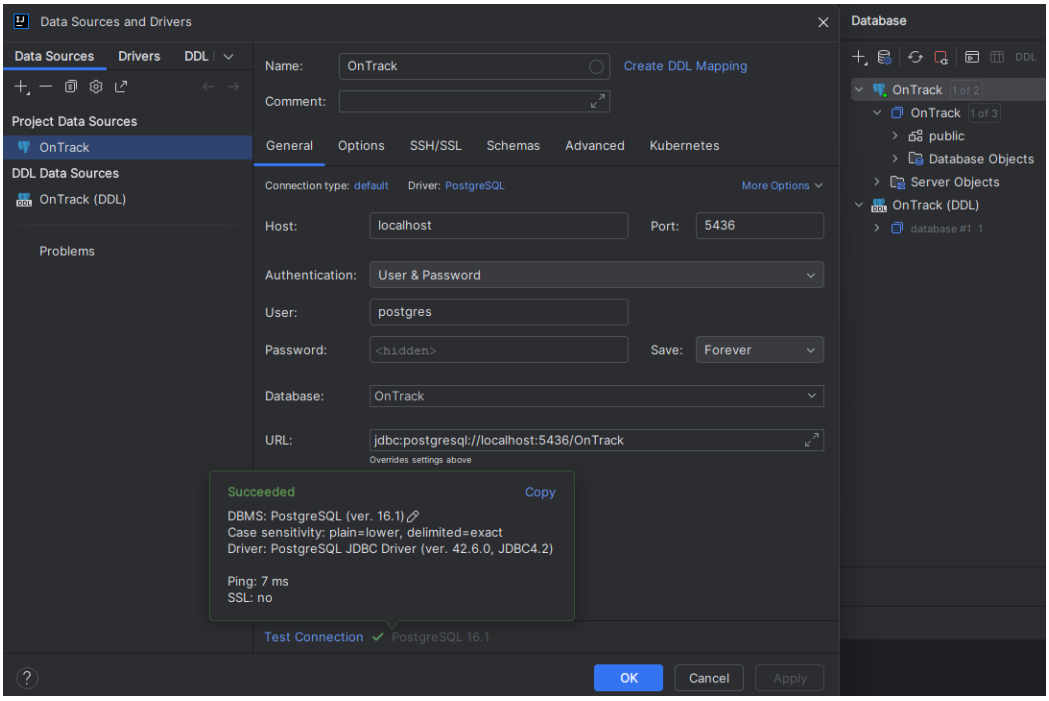
Adatbázis:

Az alkalmazás használatához szükség van a PostgreSql adtabázis kezelőhöz: https://www.postgresql.org/download/

Az új adatbázis készítéséhez meg kell nyitni a pgAdmin4 alkalmazást és ott a PostgreSQL-en belül a Databases-re jobbkilkk majd Create database. Elég a nevet beállítani.



Majd IntelliJ-en belül ehhez csatlakozni a következő módon: Database/add/Data source/PostgreSql. Majd kitölteni a felugró ablakot



A táblák ez első futtatás után maguktól létrejönnek.

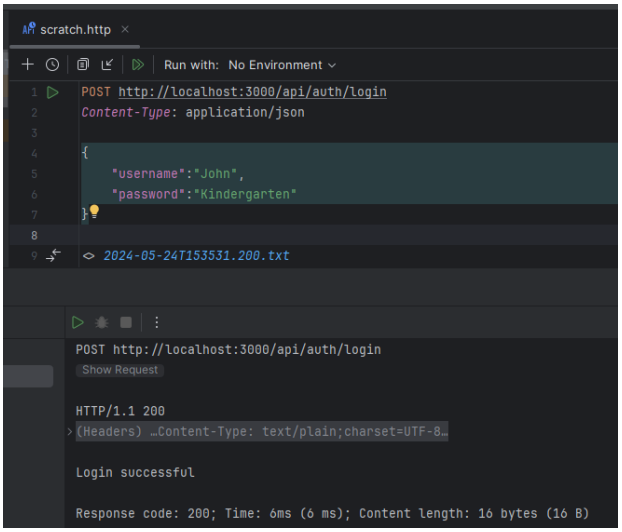
Tesztadatok:

A projektben egy db mappában mellékletem csv fájlokat, amelyek teszt adatokat tartalmaznak. Mindegyik fájl az azonos névvel ellátott táblához tartozik. Ezeket úgy lehet beimportálni, hogy tábla nevére jobbklikk majd Import/export azon belül pedig Import Data from file(s).

Lekérdezések:

A különböző endpointokat legkönnyebben az Intellij-ből lehet tesztelni. IntelliJ Idea-n belül a File/New/Scratch File/Http Request-et kell létrehozni és a kereséseket abba írni.

Pl:



Endpoint-ok:

Activities:

Get:

- /api/activities – Kiírja az összes activity-t rövidített formában
- /api/activities/<id> - Kiírja az adott id-hez tartozó activity-t minden adatát

Post:

- /api/activities/<userId> - Hozzáad az adott user-hez egy új activity-t (amit a request body-ban kell megadni), feltéve, hogy az a user van bejelentkezve. Alapból a userId-hez tartozó user-hez adja hozzá és id-t is generál magának.

Pl:

```
{
    "name":"Snack"
}
```

Patch:

- /api/activities - Módosítja az adott id-hez tartozó activity-t. Az új verziót a request body-ban kell megadni és csak az id, name és userId mezőt szabad megadni.

Pl:

```
{
    "id":552,
    "userId":152,
    "name":"Snack"
}
```

Delete:

- /api/activities/<id> - Kitörli az adott id-vel rendelkező activity-t

Auth

Post:

- /api/auth/register – Egy új user-t hoz létre a request body alapján. Meg kell adni a username-et és a password-ot. (Sajnos idő hiányában ezek tikosítatlanok:( )

Pl:

```
{
    "username":"Istvan",
    "password":"Halacska"
}
```

- /api/auth/login – Meglévő fiókba lehet itt bejelentkezni. A módszer ugyanaz mint a regisztrációnál
- /api/auth/logout – Kilépteti a felhasználót

Users

Get:

- /api/users – Visszaadja az összes az adatbázisban lévő user adatainak egy rövidített formájáz
- /api/users/<id> - Visszaadja az id-hez tartozó user minden adatát
- /api/scores/weekly/<id> - visszaadja a felhasználó heti pontszámát
- /api/score/daily/<id> - visszaadja a user napi pontját
- /api/scores/place/daily/<id> - visszaadja a user napi helyezését

Patch:

- /api/users – Frissíti a request body-ban kapott user alapján az user-t. Csak akkor, ha be van jelentkezve.

Pl:

```
{
    "id":"202",
    "username":"Istvan",
    "password":"Kutyuska"
}
```

Delete:

- /api/users/<id> - Törli az adott id-hez tartozó user-t

SubActivities

Get:

- api/subactivities – Kilistázza az összes subactivity adatait rövidített formában
- api/subactivities/<id> - Kiírja az adott id-hez tartozó subactivity összes adatatát

Post:

- api/subactivitities/<activityid> - Hozzáadja az adott activity-hez, a request body-ban kapott subactivity-t

Pl:

```
{
    "nane":"jogging",
    "calorie":"300",
    "type":"sport"
}
```

Patch:

- api/subactivity - Módosítja a subactivivity-t a requestbody-ban található subactivity alapján

Pl:

```
{
    "id": 152,
    "name":"pie",
    "calorie":"700",
    "type":"food"
}
```

Delete:

- api/subactivity/<id> - Törli az id-hez tartozó subactivity-t

Statisctics:

Get:

- api/statistics/top – Visszaadja a legtöbb ponttal rendelkező user-ek nevét és pontját
- api/statistics/popular<type> - Visszaadja a legnépszerűbb entitás nevét az adott kategóriában. Kategóriák: food, sport
- api/statistic/average – Visszaadja a user-ek pontszámának átlagát