A képen szöveg látható

Automatikusan generált leírás

Java alkalmazások gyakorlat beadandó

Holyba Attila Patrik

FLX1D8

Git User: Beantech0

A feladatot egyedül készítettem el de két Git usert használtam fel.

*Github:* [*https://github.com/Beantech0/Java\_gyakorlat\_beadando\_spring.git*](https://github.com/Beantech0/Java_gyakorlat_beadando_spring.git)

**2024.11.30.**

Tartalom

[Fejlesztési és tesztelési környezet 3](#_Toc183966997)

[Feladat - Adatbázis 3](#_Toc183966998)

[Választott adatbázis 3](#_Toc183966999)

[1. Feladat látványos weboldal 4](#_Toc183967000)

[2. Regisztráció-Bejelentkezés-menüsor 4](#_Toc183967001)

[3. Felhasználók 5](#_Toc183967002)

[4. Adatmegjelenítés adatbázisból 6](#_Toc183967003)

[5. Kapcsolat űrlap, üzenetküldés 6](#_Toc183967004)

[6. Üzenetek megjelenítése 7](#_Toc183967005)

[7. RESTful API elkészítés és tesztelés 7](#_Toc183967006)

# Fejlesztési és tesztelési környezet

IntelliJ IDEA 2024.2.3 Ultimate Edition

Java verzió: OpenJDK ’17.0.12’ 2024-07-16

A jar fájl a PDF alapján lett elkészítve és a tesztkörnyezeten tesztelve.

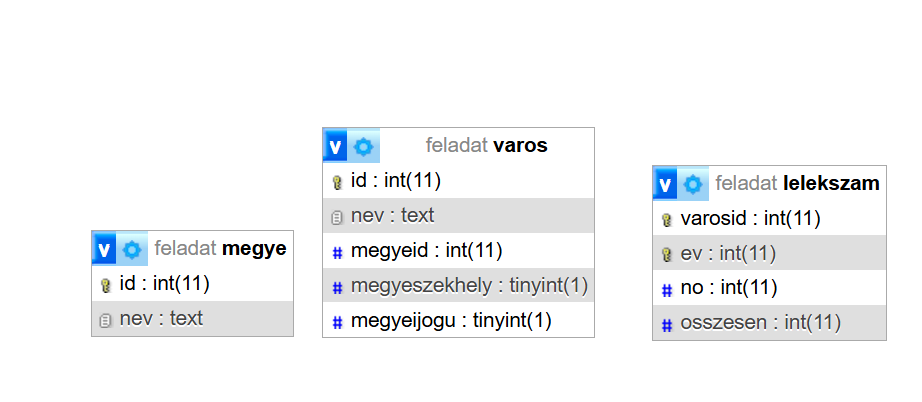
# Feladat - Adatbázis

## Választott adatbázis

A választott adatbázis: 1-20-Városok-3 táblás.

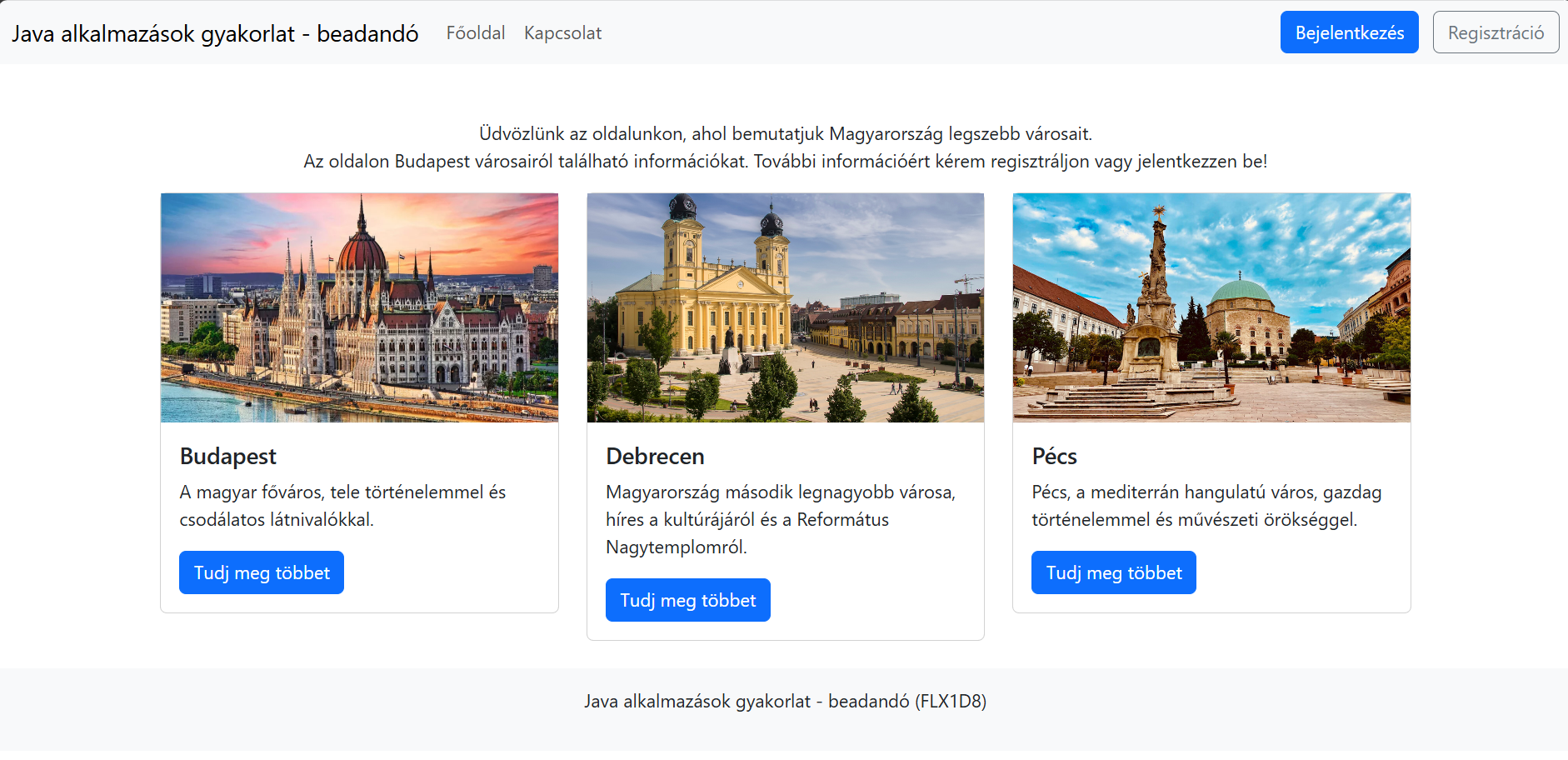
Az adatbázisban hazánk városairól találhatóak meg népességi és egyéb adatok.

A táblaszerkezet így néz ki.



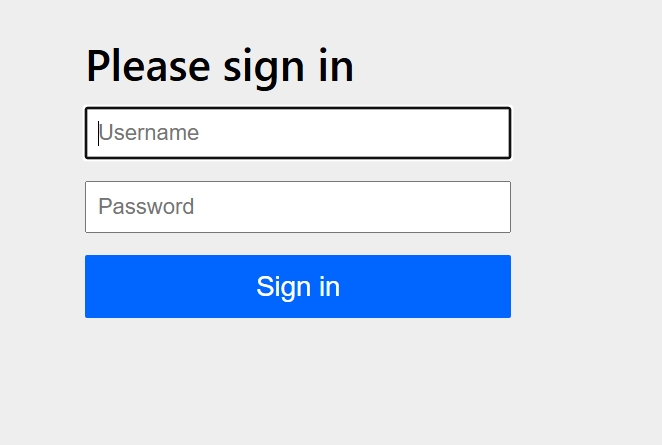
# Feladat látványos weboldal

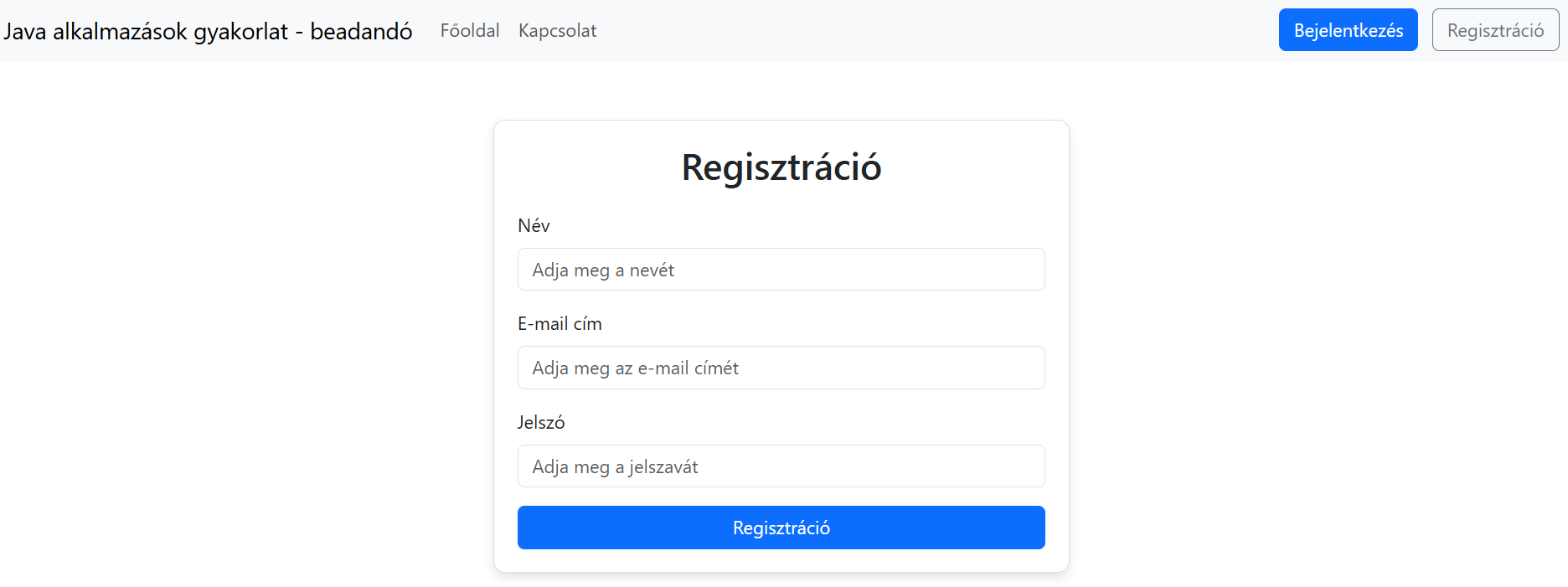
A weboldalhoz bootstraet használtam a reszponzív felület elérése érdekében.



# Regisztráció-Bejelentkezés-menüsor

A bejelentkezés-regisztráció és az egyéni menüsor a springbootsecurity3-al lett megoldva, az órán tanult megoldások alapján. A DB bövítve lett egy Users és egy Messages táblával, amikhez a repot is létrehoztam.

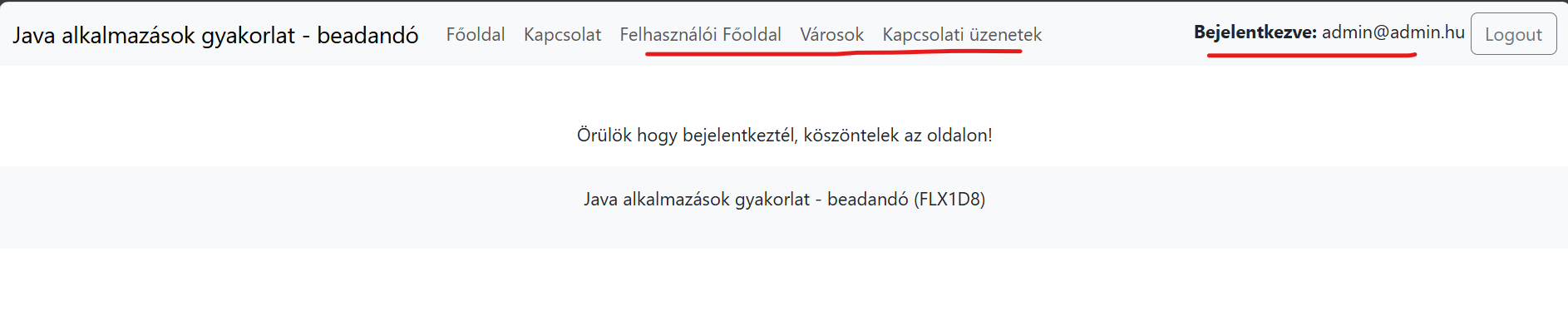




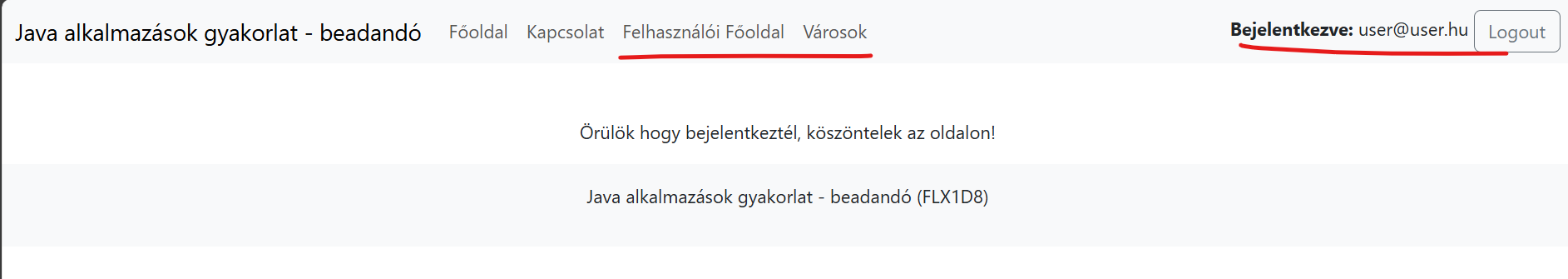
A regisztráció validálás a **“/registration\_validation”** url alatt történik meg amihez a UserDetailsService sringboot funkciók implementáltam.

# Felhasználók

Bejelentkezés után Admin esetében elérhető a “Kapcsolati üzenetek” menüpont is.



Bejelentkezett felhasználó és admin esetében a Felhasználói Főoldal és a Városok menüpont is elérhető.

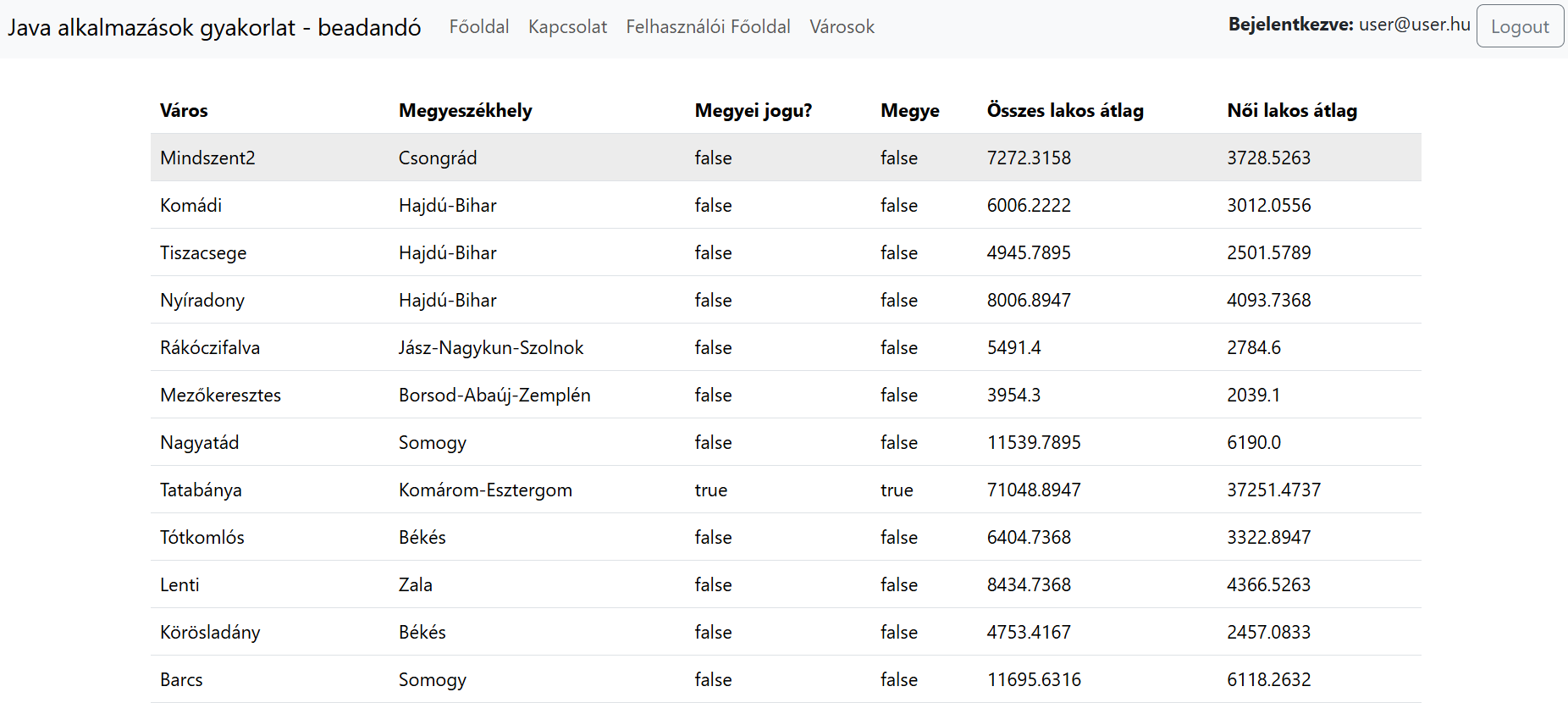


Ezek mind a layout.html-ben a security modullal lettek megoldva és a WebSecurityConfig osztályban.

Létrehozott felhasználók:

|  |  |  |
| --- | --- | --- |
| Jogkör | Felhasználónév | Jelszó |
| User | user@user.hu | asd |
| Admin | admin@admin.hu | admin |

# Adatmegjelenítés adatbázisból

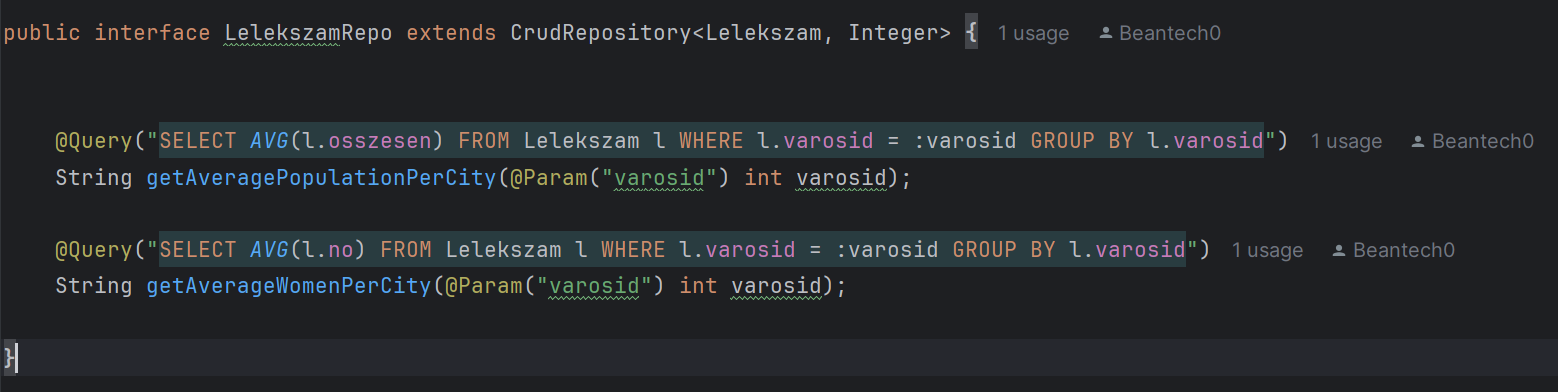


Az adatbázisból való adatmegjelenítést a “Városok” menüpont alatt oldottam meg.

Ami meghívja a “VarosokData” függvényt ami egy formázott stringet ad vissza.

Itt a varosRepo alapján lekérem az összes várost és kííratom a város és megye adatokat.

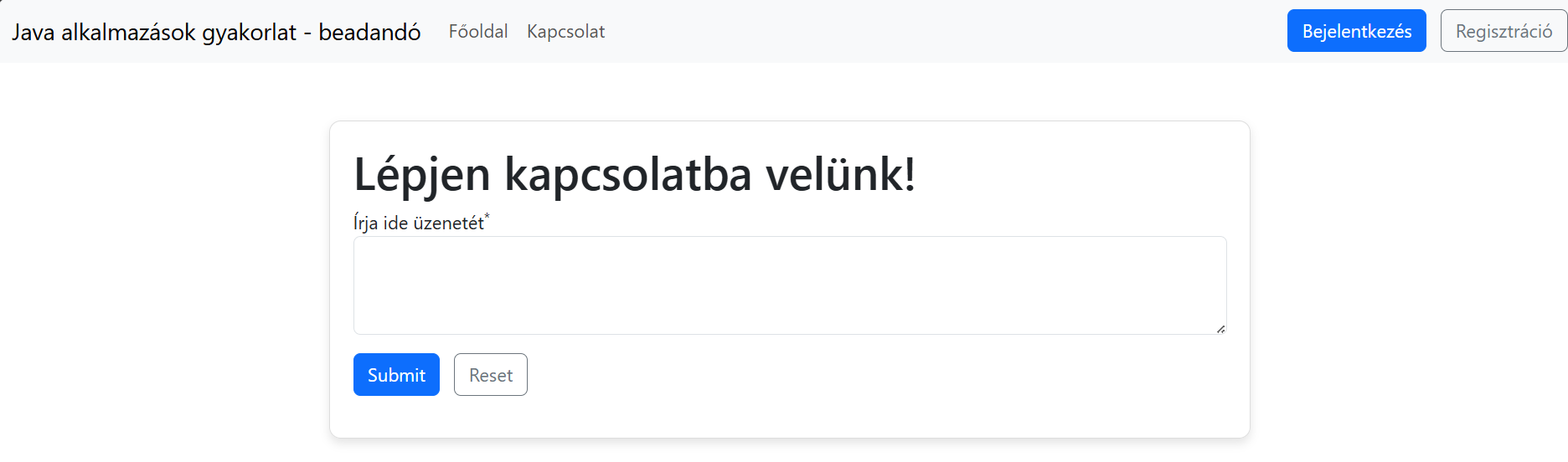
Továbbá a lélekszám táblára készítettem egy egyedi lekérdezést a LelekszamRepo interface-ben ami visszaadja az adott varosid alapján hogy mennyi volt az átlagos összes lakosság és a női lakosok átlaga az összes évet tekintve.



Utána ezt a stringet adja vissza a varosok.html ahol a <p th:text=”${str}/> résznél kiiratom a str értékét.

# Kapcsolat űrlap, üzenetküldés

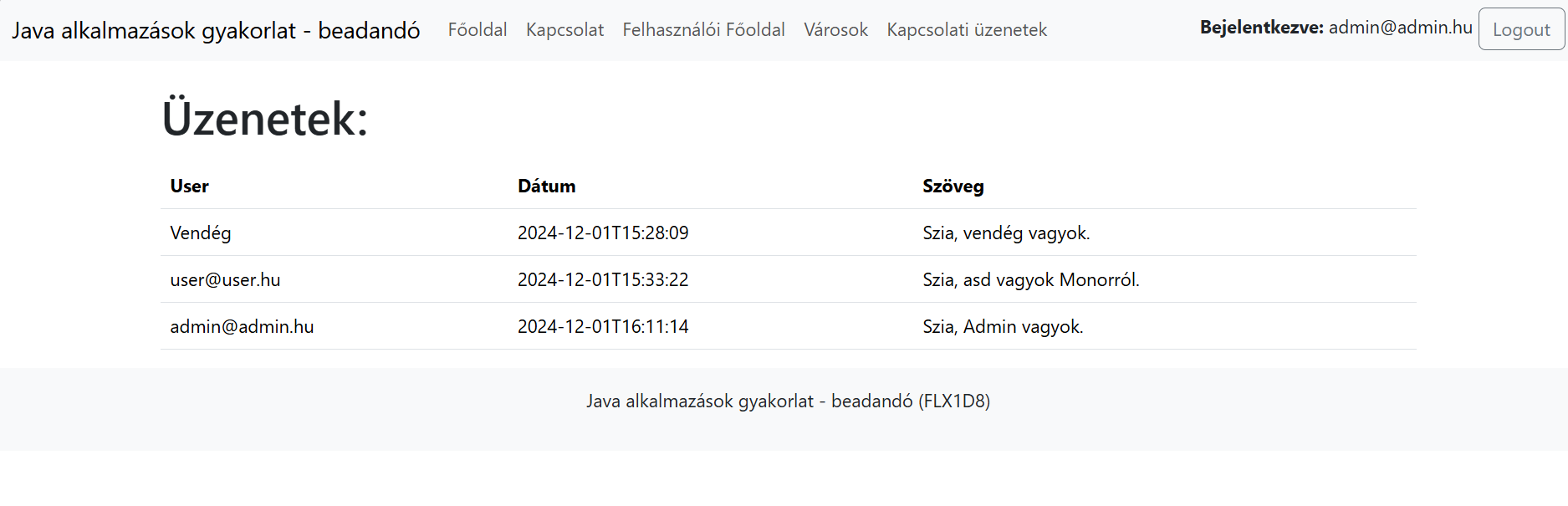
A kapcsolat menűpont alatt van lehetőség üzenetet küldeni az adminnak az alábbi formal.



Ha nincs bejelentkezve senki akkor Vendég néven fogja elmenteni az üzenetet egyébként pedig a user felhasználóneve alapján. A /kapcsolat url-ről a Submit gomb átküld a /kapcsolat\_success URL-re ahol az urlapSubmit elmenti a Messages táblába az üznetet. Az felhasználói adatok kiolvasása után a messagesRepo.save metódusával kerül mentésre az üzenet.

# Üzenetek megjelenítése

Az üzenetek megtekintésére az adminnak van lehetősége a “Kapcsolati üzenetek” menüpont alatt.



Itt hasonlóan a városok listázásához az AdminUzenetek függvény egy stringet ad vissza ahol lekérem az üzeneteket és formázott stringként megjelenítem a táblázat bodyban.

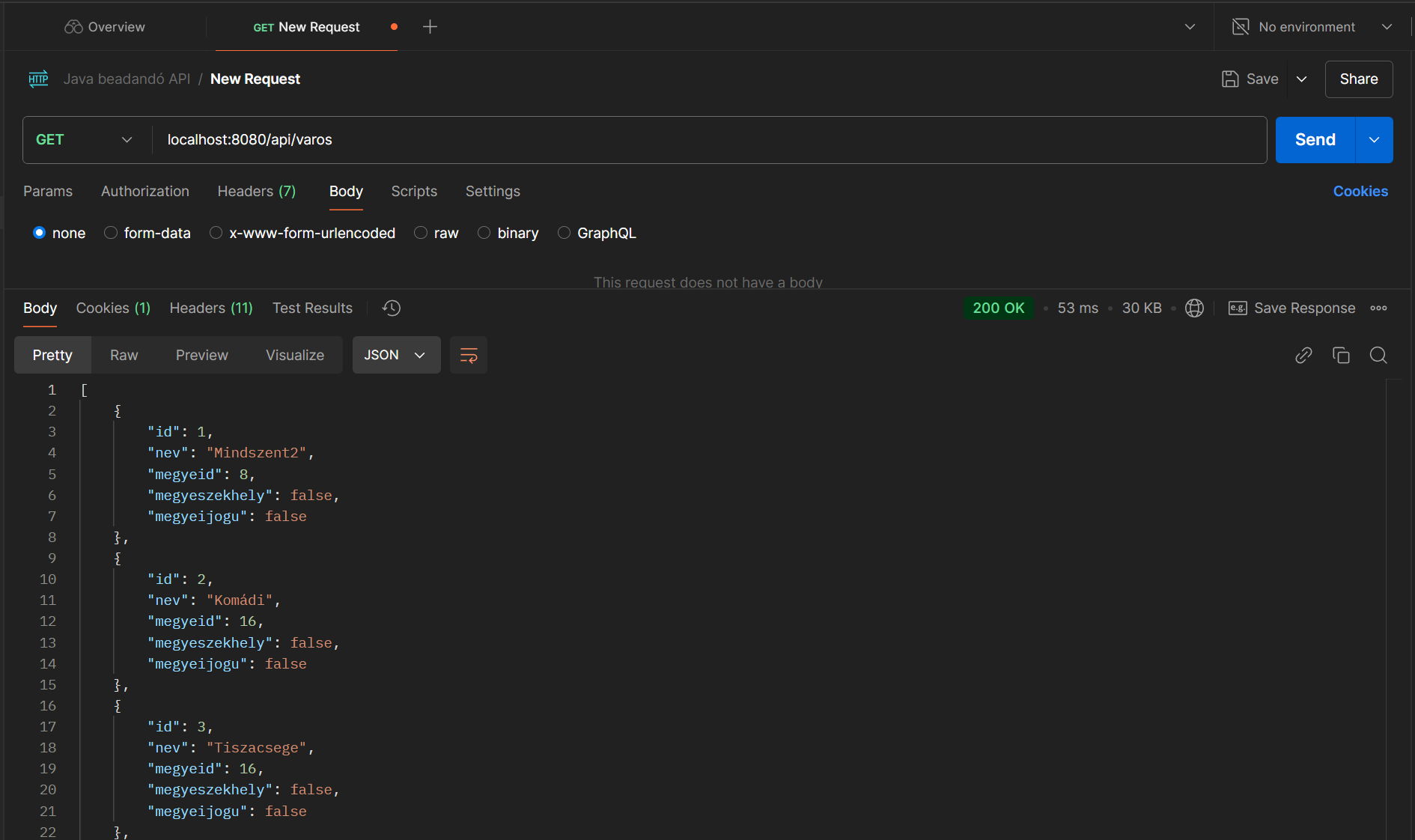
# RESTful API elkészítés és tesztelés

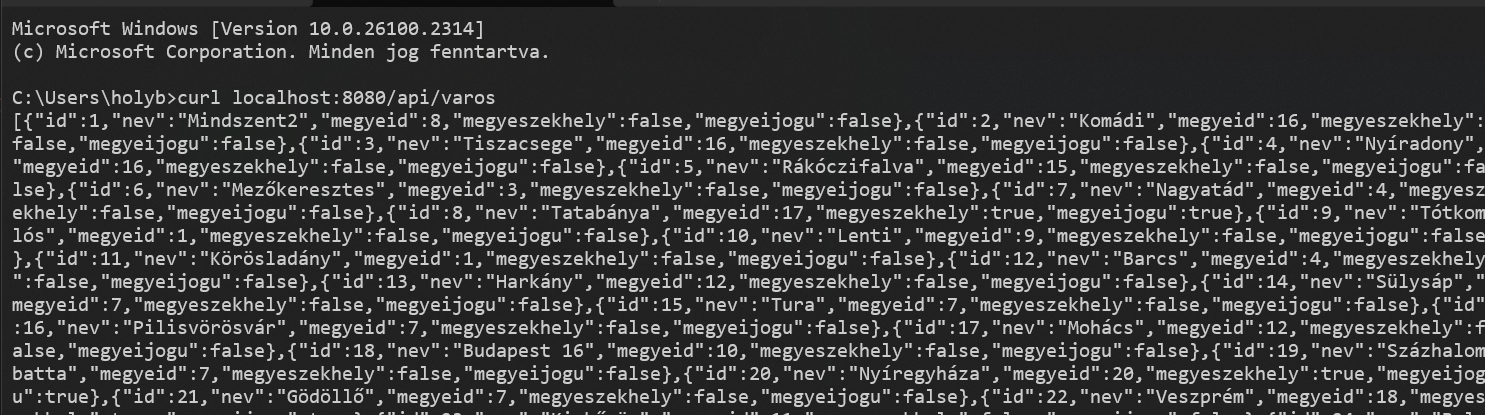
A REST API hívásokat az órán tanult módszerrel készítettem el. Csináltam egy VarosokRestController osztályt aminél megadtam a @RestController kapcsolót.

Az API az /API/ url alatt érhető el.

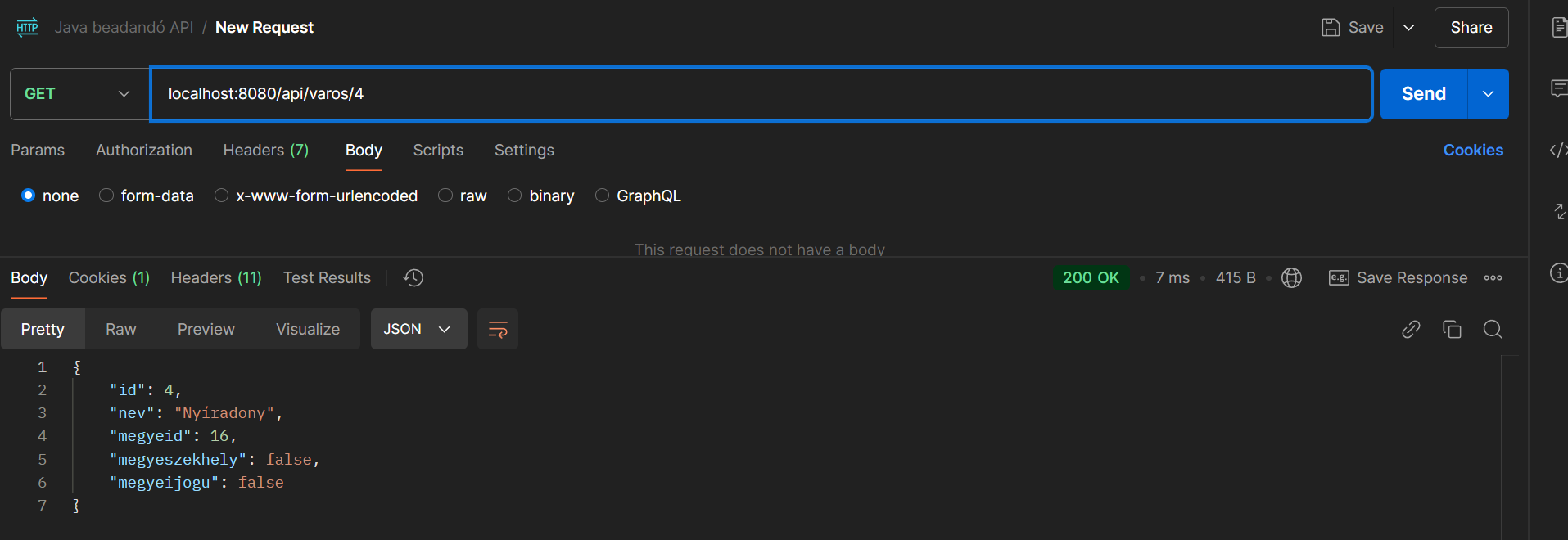
Elsőként az GET /api/varosok URI alatt listázom az összes várost a VarosRepo segítségével.

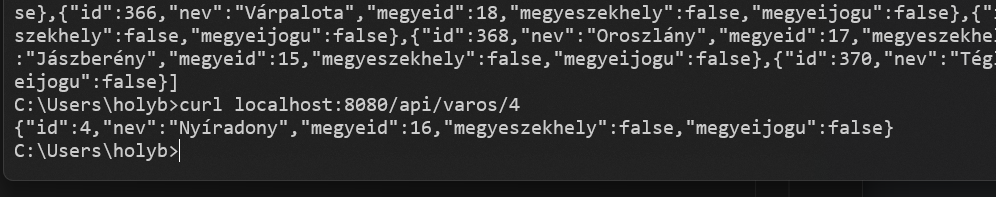
A tesztelésről képernyőképeket Postmanről és CURL-ről beletettem.



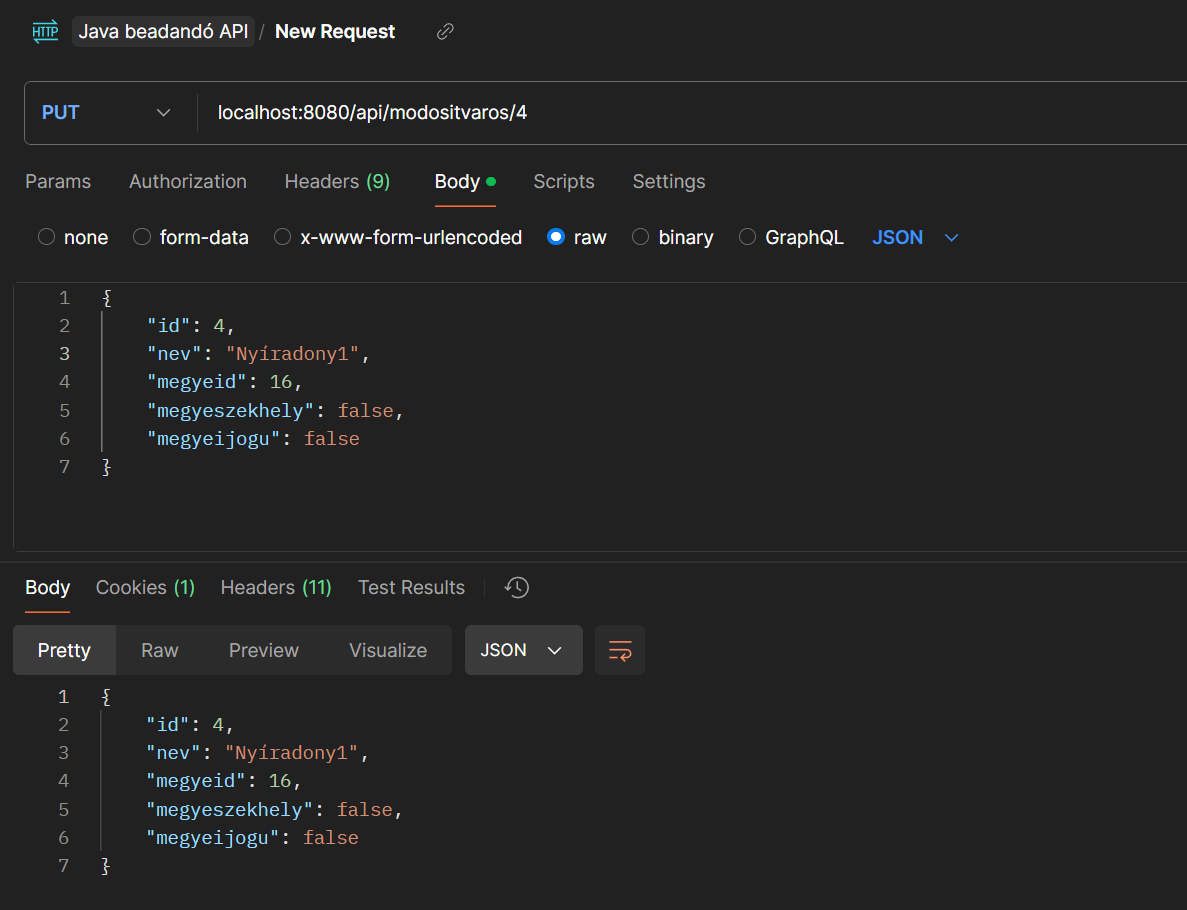


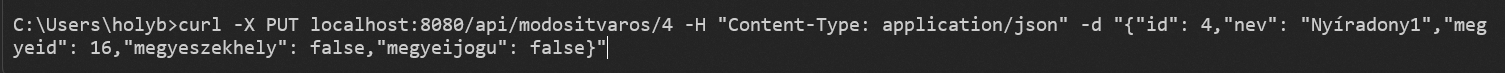
A másik GET lekérés az /api/varos/{id} mező alapján egy várost ad vissza



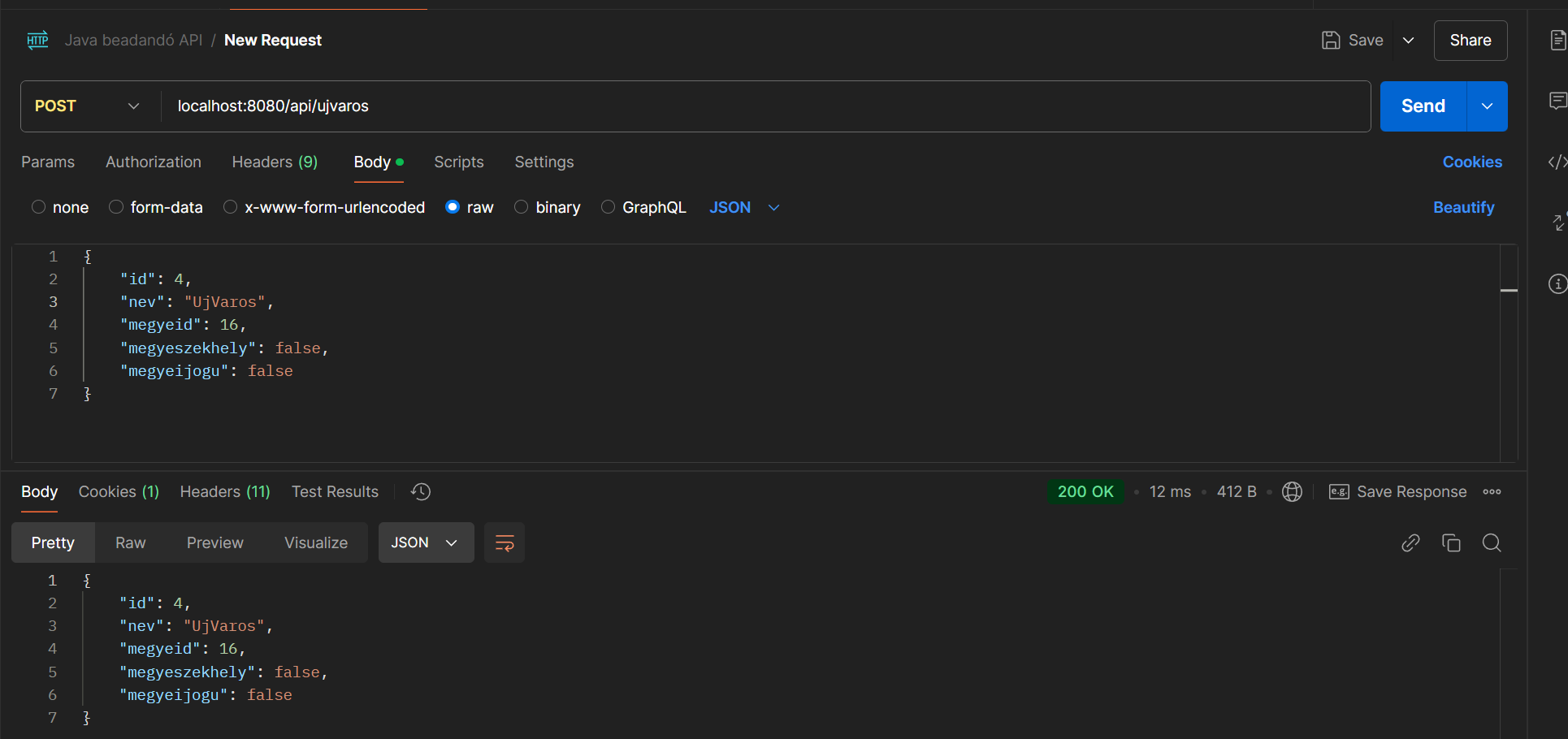


A PUT /api/modositvaros/{id} módosítja a megadott városid értékeit, itt json-ben kell megadni az új értékeket. Itt kiolvassuk az adatokat a kapott json-ből és ezeket elmentjük az id-hoz. Ha nincs ilyen ID akkor létrehoz egy új rekordot.





A PUT-hoz hasonlóan a POST /api/ujvaros létrehoz egy új várost, itt nem kell megadni ID-t mert azt létrehozza majd a DB autoincrement miatt. Egy JSON fájlt kell megadni.



/api/varostorol/{id} DELETE method-al készült egy törlési API endpoint ahol ID alapján lehet várost törölni. Látható hogy 200-at ad vissza az API és törlödött a város.

