**Backend programozás (adatbázis lekérdezést is végző, néhány végpontot tartalmazó REST API kiszolgáló létrehozása): 15 pont**

Írjon egy backend oldali programot Java, PHP, vagy C# nyelven. A program felhasználók adatait tárolja, és ezek módosítására és törlésére ad lehetőséget. **A megoldás során a kész adatbázis exportját is mentse el a feladatmegoldás többi állománya mellé!**

1. Hozzon létre új adatbázist **vizsga** néven **utf-8**-as karakterkódolással! **(1 pont)**

2. Importálja be az adatbázisba a **vizsga.sql** fájlban található scriptet! **(1 pont)**

3. Készítsen egy új projektet egy backend oldali alkalmazáshoz. **(1 pont)**

4. Létesítsen kapcsolatot a backend alkalmazás és az adatbázis között! **(1 pont)**

5. Alakítsa ki a model réteget. A model osztályban szereplő propertyket az adatbázis tábla mezőinek megfelelően készítse el (Minden mező szerepeljen benne). Készítsen konstruktort a model réteghez. **(2 pont)**

6. A model rétegen készítse el az adatbázisban szereplő tárolt eljárások meghívására szolgáló metódusokat: **(4 pont)**

* **CreateUser**
  + Paraméterek: username, email, password
  + Visszatérési érték: boolean (Sikeres volt-e)
  + Működés: CreateUser eljárás meghívása, és paraméterezése. Sikeres létrehozás esetén true érték visszatérítése
* **DeleteUser**
  + Paraméterek: id
  + Visszatérési érték: boolean (Sikeres volt-e)
  + Működés: DeleteUser eljárás meghívása, és paraméterezése. Sikeres törlés esetén true érték visszatérítése
* **GetUser**
  + Paraméterek: id
  + Visszatérési érték: User entitás
  + Működés: GetUser eljárás meghívása, és paraméterezése. Sikeres kérés esetén a user adatainak visszatérítése
* **UpdateUser**
  + Paraméterek: id, username, email, password
  + Visszatérési érték: boolean (Sikeres volt-e)
  + Működés: UpdateUser eljárás meghívása, és paraméterezése. Sikeres módosítás esetén true érték visszatérítése

7. Készítsen egy kontrollert **User** néven. Készítse el rajta az alábbi végpontokat: **(4 pont)**

* **new**
  + Típus: POST
  + Request body adatai: username, email, password
  + Response:
    - 200 - OK státusz és sikeres üzenet sikeres létrehozás esetén
    - 415 - Unsupported Media Type státusz ha valamelyik adat hiányzik, vagy nem megfelelő formátumú. Megfelelő hibaüzenet visszaküldése
  + Működés: Adatok ellenőrzése után CreateUser metódus meghívása modell rétegről
* **delete**
  + Típus: DELETE
  + Request paraméterei: id
  + Response:
    - 200 - OK státusz és sikeres üzenet sikeres törlés esetén
    - 404 - Not Found státusz, ha törlés sikertelen, vagy nincs megadva id
  + Működés: DeleteUser metódus meghívása modell rétegről
* **GetUser**
  + Típus: GET
  + Request paraméterei: id
  + Response:
    - 200 - OK státusz és user adatainak visszatérítése sikeres lekérdezés esetén
    - 404 - Not Found státusz, ha a lekérdezés sikertelen, vagy nincs megadva id
  + Működés: GetUser metódus meghívása modell rétegről, sikeres kérés esetén a user adatainak visszatérítése
* **UpdateUser**
  + Típus: PATCH
  + Request body adatai: id, username, email, password
  + Response:
    - 200 - OK státusz és sikeres üzenet sikeres módosítás esetén
    - 404 - Not Found státusz, ha a lekérdezés sikertelen, vagy nincs megadva id
    - 415 - Unsupported Media Type státusz ha valamelyik adat hiányzik, vagy nem megfelelő formátumú. Megfelelő hibaüzenet visszaküldése
  + Működés: Adatok ellenőrzése után UpdateUser metódus meghívása modell rétegről

8. Állítsa be az alábbi respone header információkat az alkalmazásban: **(1 pont)**

* Content-Type: application/json; charset=utf-8
* Access-Control-Allow-Origin: \*
* Access-Control-Allow-Methods: GET, POST, PATCH, DELETE, OPTIONS

**A megoldás során a kész adatbázis exportját is mentse el a feladatmegoldás többi állománya mellé!**