Önálló laboratórium 2 –

haladási napló

Október 5.

Létrehoztam egy View komponenst a profil adatok módosítására „SettingsProfileViewForm” néven. Ebben a komponensben helyet kapott egy „Form” elem, amin belül „Section”-ökben különítettem el a View egyes komponenseit. Az első Section-ben lehet felvenni különböző „TextField”-ekben a módosítani kívánt adatokat. A TextField-ek megkapják keződ értéknek az aktuálisan érvényes adatokat, amelyeket a felhasználó módosíthat.

A következő Sectionben lehetőség lesz módosítani a profilképet, erre létrehoztam egy gombot, amely megnyomásával majd tallózható lesz az új profilkép. Ezen felül beállítottam egy „toolbar”-t is a View-ban a billentyűzetre, erre azért volt szükség, mert a telefonszám mezőben numerikus billentyűzetet adok fel, és abból csak a „Done” gomb megnyomásával van lehetőség úgy kilépni, hogy a begépelt érték értékül legyen adva.

A létrehozott View-nak megfelelően módosítottam a „ProfileView” komponenst is. A „Button” elemet, amely megnyomásával az imént létrehozott View-ra lehetett volna navigálni, kicseréltem egy „NavigationLink”-re, amelynek a trailing closure-jében megadtam a lecserélt gomb szövegét, illetve a gomb modifier-jeit a NavigationLink-re állítottam be. Ezáltal az új „gomb” megnyomásával az újonnan létrehozott képernyőn kötünk ki.

Október 11.

A képernyőn, ahol a felhasználó módsíthatja az adait, bekötésre került az ImagePicker komponens. Ez lehetővé teszi, hogy a felhasználó az „Adatok módosítása” képernyőn a „Profilkép módosítása” gombra kattintva választhasson a galériából egy tetszőleges új profilképet. Ez a profilkép egyelőre csak az adatok módosítása képernyőn jelenik meg, a kép megváltoztatása nem lett átvezetve a fő, profil adatokat tartalmazó képernyőre (erre majd akkor kerül sor, ha az alkalmazásban meg lesz valósítva az adatbázissal történő kommunikáció).

Megvalósításra került a „Jelszó módosítás” képernyő. Ezen a képernyőn három darab „SecureField” mező található egy „Form” elemen belül. Ezek az elemek biztosítják, hogy a felhasználó biztonságosan tudjon szöveget bevinni, azaz az ilyen elemeket kell használni olyan esetben, amikor pl. a felhasználó a jelszavát akarja megadni. A Formban ezen kívül helyet kapott egy gomb is, amely megnyomásával, valid régi jelszó és sikeres új jelszó megerősítés mellett, a felhasználó módosíthatja a jelszavát.

Amennyiben a felhasználó hibásan adta meg a régi jelszavát, vagy az újonnan megadott jelszó és az új jelszó megerősítésére begépelt szöveg nem egyezik meg, az alkalmazás a felhasználót egy „alert” formájában értesíti. A sikeres jelszó megadások után a felhasználó a gombot megnyomva módosítja a jelszavát (ez a funkció akkor kerül megvalósításra, amikor az adatbázis/mockolt adatbázis be lesz kötve az alkalmazásba).

November 1.

Endpointok megtervezése, hívásokkal együtt

* Elérhető szállások listázása a megadott paraméterek alapján:
  + GET /accomodations/, body-ban elküldött adatok (JSON formátumban):
    - város (string), tartózkodás kezdete (dátum formátum), tartózkodás vége (dátum), fizetés módja (ha lehet enum típus akkor az, ellenkező esetben string és backend kezeli a stringet), preferált árkategória (numerikus érték - opcionális), árkategória pénzneme (enum vagy string), preferált értékelés (numerikus - opcionális), egyéb kategóriában felsorolt értékek (boolean)
    - Ha nincs a keresési paramétereknek megfelelő szállás, akkor 404-gyel tér vissza a backend, itt frontend oldalon egy hiba képernyőt kell megjeleníteni (lehet dialógusablak, amiben az alkalmazás kiírja a usernek, hogy nem volt sikeres a keresés, amit a user leokézhat)
    - Ha egy megadott paraméter nem megfelelő, akkor 400-as hibakódot küld vissza a backend, amit szintén dialógusablak formájában jelenít meg az alkalmazás a felhasználónak (elméletileg nem kéne lehetségesnek lennie ilyen kérésnek, de minden esetre kell ilyen válasz is)
    - Ha sikeres a keresés, akkor 200-zal tér vissza a backend, és egy tömbben visszaküldi a megfelelő szállások adatait:
      * id (numerikus – szükséges?), cím (string), földrajzi hosszúsági koordináta (numerikus – keleti irány a pozitív, nyugati a negatív), földrajzi szélességi koordináta (numerikus – északi irány a pozitív, déli a negatív), típus (enum vagy string (kanapé, szoba apartman, stb)), ár (numerikus), pénznem (enum vagy string), vendégek maximális száma (numerikus), egyéb tulajdonságok (boolean érték minden tulajdonságnak)
* Szoba kiadása
  + POST /accomodations/, body-ban elküldött adatok:
    - user id (numerikus), cím (string), földrajzi koordináták (numerikusak), szállás típusa (enum/string), elfogadott fizetési módok (boolean minden fizetési módnak), ár (numerikus), pénznem (enum/string), egyéb tulajdonságok (boolean)
    - Ha sikertelen a kérés, akkor 404-gyel tér vissza a backend és dialógusablakot dob fel a felhasználónak, amit leokézhat
    - Ha egy megadott paraméter nem megfelelő, akkor 400-as hibakódot küld vissza a backend, amit szintén dialógusablak formájában jelenít meg az alkalmazás a felhasználónak (elméletileg nem kéne lehetségesnek lennie ilyen kérésnek, de minden esetre kell ilyen válasz is)
    - Ha sikeres volt a backend részén a kérés feldolgozása (adatbázisba írás), akkor 201-es státuszkóddal tér vissza a backend és a sikeres feldolgozásról dialógusablakban tájékoztatja a felhasználót az alkalmazás
* Ranglista (foglalók)
  + GET /scoreboard/tenants/
    - Nem kerül elküldésre semmilyen adat a body-ban, sem más formában
    - Sikertelen kérés esetén 404-gyel tér vissza a backend (itt csak ez a sikertelen státuszkód kerül elküldésre a mobilalkalmazás felé, mert nincs olyan kitölteni szükséges paraméter, amit elronthatna a felhasználó)
    - Sikeres kérés esetén egy tömbben elküldi a backend a foglalók adatait a pontszámuk alapján sorba rendezve: id (numerikus), név (string), felhasználónév (string), pontszám (numerikus), kép (ennek a tárolására ki kell majd találni valamilyen hatékony módszert), státuszkód 200
    - Az id-ra azért van szükség, mert ha a felhasználó meg akarja tekinteni az adott felhasználó profilját, akkor a következő hívásban szükség lesz az id-ra
* Ranglista (foglalók)
  + GET /scoreboard/owners/
    - Nem kerül elküldésre semmilyen adat a body-ban, sem más formában
    - Sikertelen kérés esetén 404-gyel tér vissza a backend (itt csak ez a sikertelen státuszkód kerül elküldésre a mobilalkalmazás felé, mert nincs olyan kitölteni szükséges paraméter, amit elronthatna a felhasználó)
    - Sikeres kérés esetén egy tömbben elküldi a backend a kiadók adatait a pontszámuk alapján sorba rendezve: id (numerikus), név (string), felhasználónév (string), pontszám (numerikus), kép (ennek a tárolására ki kell majd találni valamilyen hatékony módszert), státuszkód 200
* Adott profil megtekintése
  + GET /users/id={user\_id}/
    - A kérés body-ban nem kerül elküldésre semmilyen adat
    - Sikertelen kérés esetén 404-gyel tér vissza a backend
    - Ha nem megfelelő user id kerül megadása path paraméterként, akkor 401-es státuszkóddal visszatérve dialógusablakban tájékoztatja a felhasználót az alkalmazás, hogy nincs a keresésnek megfelelő felhasználó
    - Sikeres kérés esetén 200-as státuszkóddal visszaküldi a backend az adott felhasználó nevét (string), felhasználónevét (string), születési dátumát (dátum), email címét (string), telefonszámát (string/numerikus – ez a karaktereken múlik majd), és profilképét
    - A felhasználó a saját profilját is ilyen kérés után tudja megtekinteni – az alkalmazás lokálisan eltárolja a bejelentkezés után a felhasználó saját azonosítóját, amit felhasznál majd a saját profil adatok megtekintésekor
* Saját adatok módosítása
  + POST /users/id={saját\_user\_id}/
    - A kérés body-ban a felhasználó elküldi az összes adatot – beleértve a nem módosítottakat is, amelyek értéke a korábbiakhoz képest változatlan – név (string), felhasználónév (string), email (string), telefonszám (numerikus/string)
    - A felhasználó akkor hajthatja végre ezt a POST kérést, ha a saját user id-ja áll a path paraméterben – ellenkező esetben 404-es hibakóddal tér vissza a backend (403-assal is lehetne, de annak érdekében, hogy a jogosulatlan felhasználó a hibakódokból ne tudjon következtetéseket levonni, 404 kerül visszaküldésre)
    - Sikeres módosítás esetén 201-es státuszkód kerül elküldésre, majd a felhasználói felületen dialógusablak formájában a felhasználó tájékoztatása is megvalósul a sikeres módosításról
* Jelszó módosítás
  + PUT /users/password/id={saját\_user\_id}/
    - Body-ban elküldésre kerül a régi jelszó (string), amelynek az egyezésének a vizsgálatát a backend végzi majd, és az új jelszó (string)
    - A jelszavak küldése történhet valamilyen titkosítás bevonásával
    - Ha valamilyen hiba történik, akkor 404-gyel tér vissza a backend, a hibáról dialógusablak formájában kap tájékoztatást a felhasználó
    - A fenti POST kéréshez hasonlóan csak a user csak a saját jelszavát módosíthajta (hitelesítés)
    - 400-as hibakóddal tér vissza a backend, ha rossz régi jelszót adott meg a user – dialógusablakban tájékoztatás
    - Ha sikeres a módosítás, akkor 201-es státuszkód kerül elküldésre

November 8.

Létrehoztam a backendet megvalósító Spring alkalmazást. A Spring Initializr-ben beállítottam, hogy Kotlin nyelven szeretném létrehozni az alkalmazást, Gradle típussal, a JDK verziót pedig 11-re állítottam be. Kezdetben a függőségek között felsoroltam a Spring Webet, a Rest Repositoriest, a Spring Security-t, a Lombokot, a Liquibase Migrationt és a MySQL Drivert.

A Spring alkalmazás létrehozása után a MySQL workbanchben létrehoztam az adatbázist, amely az alkalmazáshoz fog tartozni. A MySQL workbanch működésre bírása nagyobb erőfeszítést igényelt, mint azt korábban gondoltam volna, és így sem tudom mi volt a probléma, de az újratelepítés segített megoldani a problémát. A MySQL workbench alkalmazásban létrehoztam az alkalmazáshoz tartozó adatbázist „couchsurfing\_db” néven, amelyben létrehoztam egyelőre csak a „user” táblát – ennek oka, hogy először ezen az egy táblán fogom kipróbálni a Spring működését, azaz az adatbázis bekötését a Spring alkalmazásba, majd létrehozni a megfelelő osztályokat (DTO, controller, repository, stb.), amely a tábla manipulálását végzik.

November 15.

Az adatmodell alapján létrehoztam MySQL-ben az adatbázis sémát, amelyben feltöltöttem dummy adatokkal a táblákat.

Kérdés: a spring security által létrehozott users táblát el lehet érni a saját adatbázisomban, vagy ez külön kell kezelni? Illetve, amennyiben el lehet érni, akkor ezzel ki lehet bővíteni valahogy a saját user táblámat, vagy ez nem legjobb út tervezési szempontból?

November 22.

Spring security tutorial: <https://spring.io/guides/gs/securing-web/>

Spring autentikáció UserDataService-szel, ami mögött adatbázis van: <https://www.baeldung.com/spring-security-authentication-with-a-database>

A model osztály létrehozásánál nem ismerte fel az IntelliJ a „@Entity” annotációt. Kis keresgélés után megtaláltam a hiba okát: a JPA használatához az entity osztályoknak rendelkeznie kell egy paraméterek nélküli konstruktorral, amellyel a Kotlin osztályok default esetben nem rendelkeznek. Ennek a generálásához egy JPA plugin-re volt szükség, amelyet fel kellett venni a build fájlba.

Ehhez a build fájl plugins mezőjébe felvettem egy „kotlin(„jpa”) verison „1.8”” sort

Ezután felismerte a projekt a JPA annotációkat.

Újabb hibába ütköztem: valami trial preiódus lejárt a JPA -> új jpa plugint kellett keresni, ezért a fent említett sort kicseréltem a „id(„org.jetbrains.kotlin.plugin.jpa”) version „1.6.21””-re, ahol figyelni kellett hogy ennek a pluginnek a verziója megegyezzen a JVM és a Spring pluginek verziójával.

A model osztály létrehozásánál probléma volt, hogy az egyes mezőknek nem adtam kezdetben default értéket (mint ahogyan a tutorialokban sem írták) -> probléma, hogy adtam nekik?