BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Házi feladat dokumentáció

Android alapú szoftverfejlesztés

2021 tavasz

Puppy Sitter

Seres Soma – LQH4X5

somaseres@gmail.com

Laborvezető: Kövesdán Gábor

MŰEGYETEM 1782

Bemutatás

Az alkalmazás egy Tinderhez hasonló, kutyák és azok bébiszittereinek készülne. Röviden a lényege az lenne, hogy a „kutyák” [itt nyilván a gazdikra kell gondolni] látják a bébiszittereket, és közülük válogathatnak képek és leírás alapján, míg a bébiszitterek a kutyákról látnak képeket és szintén rövid leírást, például a mozgásigényükről, etetési szokásokról, stb.

Az alkalmazás github repositoryja itt érhető el. A bemutató videó pedig itt.

Főbb funkciók

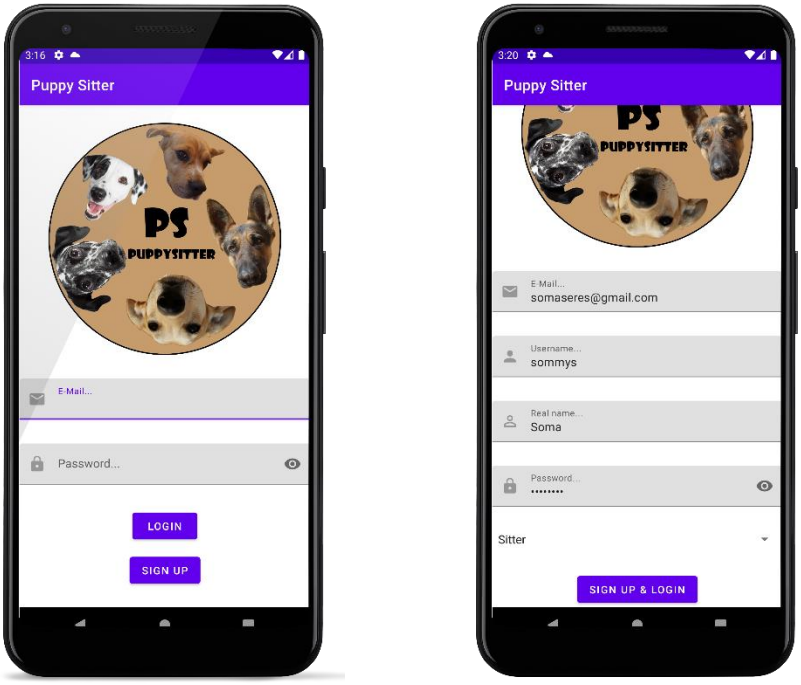
Mindkét fajta felhasználó tud regisztrálni [itt kell kiválasztani, milyen típusú fiókot szeretnénk] illetve regisztráció után belépni.

A felhasználók tudnak saját adatokat megadni, illetve szerkeszteni, képe[ke]t feltölteni, és preferenciákat állítani a másik fajta felhasználók tekintetében [pl. csak bizonyos korosztályban keresnek szittert a kutyák, vagy csak bizonyos mozgásigényű kutyákat keres a szitter, stb.]

A felhasználók a fő felületen a másik fajta felhasználók között tudnak válogatni két gomb segítségével. Ha rányomnak a megjelenő képre, akkor a részletes adatokat is megnézhetik, majd visszatérve a fő nézetbe választhatnak, ha döntöttek.

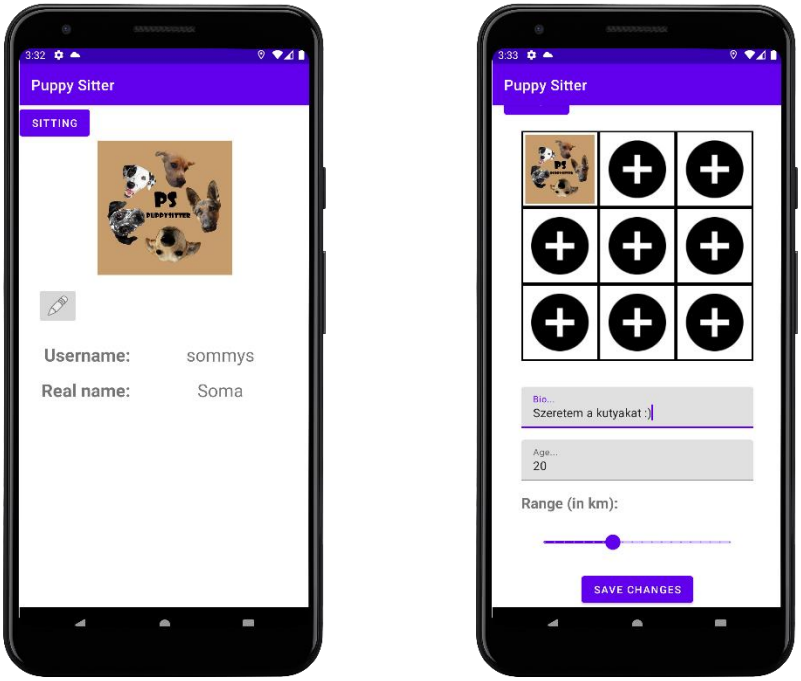
Felhasználói kézikönyv

Az alkalmazást megnyitva a bejelentkező képernyő fogad minket. Itt tudunk, a már regisztrált fiókunkba belépni. A „SIGN UP” gombra nyomva átvisz minket a regisztrációs felületre, ahol új fiókot hozhatunk létre.



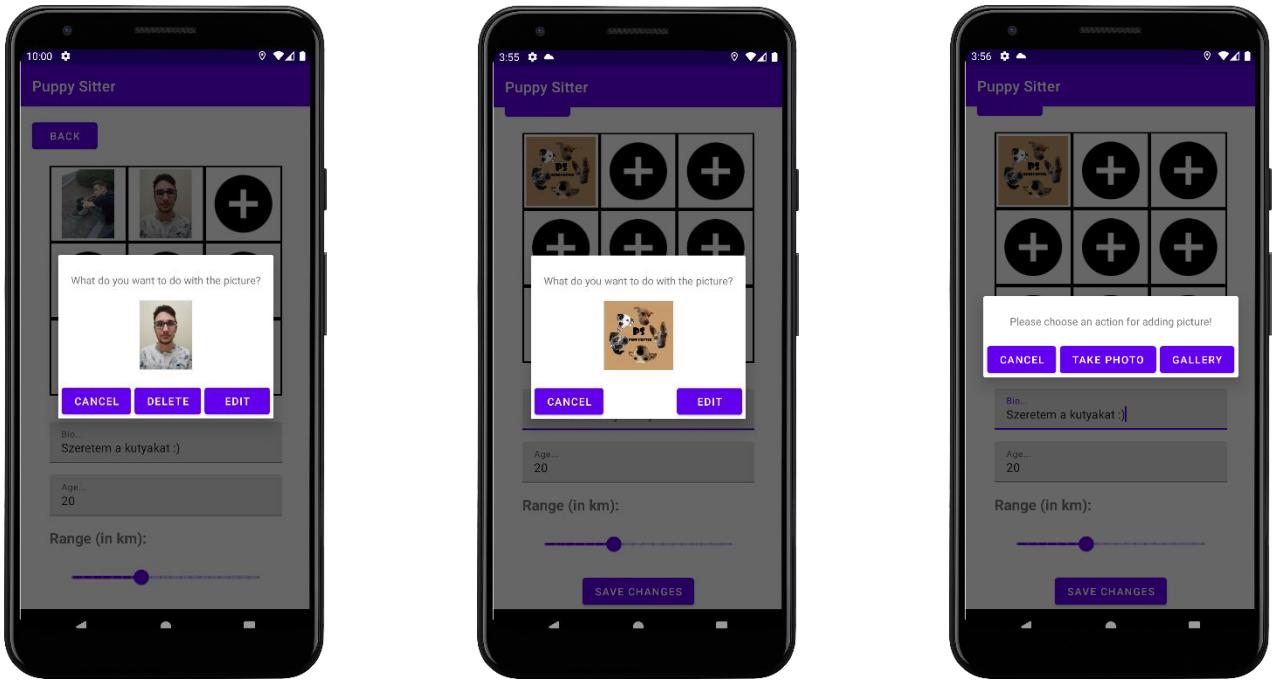
1. ábra: A bejelentkező és a regisztráló felület. Regisztrációkor automatikusan be is léptet minket a rendszer.

Belépés után a profilunk fogad minket, itt megjelenik a profilképünk, a megadott adataink, illetve két gomb is. A bal fenti gombra nyomva elkezdhetünk a közelben lévő [általunk megadott határon belül ld. profil adatoknál később] ellenkező típusú felhasználók közül válogatni, a képünk alatti kis ceruza ikonú gombra nyomva pedig módosíthatjuk adatainkat. Az adatok módosításán kívül még lehetőségünk van összesen 9 kép feltöltésére, amiket majd lát rólunk az ellenkező típusú felhasználó, mikor sorra kerülünk a válogatás közben.



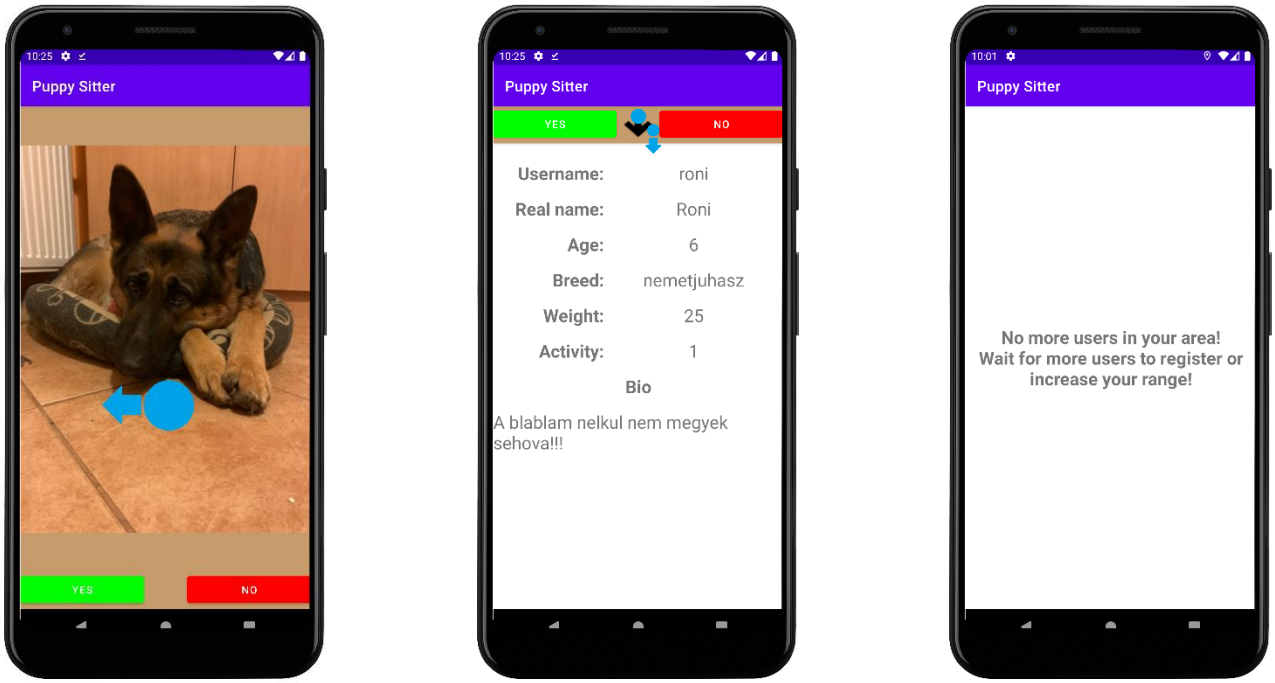
2. ábra: A bal oldalon látható a profil nézet, a jobb oldalon pedig az adatszerkesztő nézet.

Adatszerkesztéskor a képeket tudjuk módosítani, törölni [amennyiben nem az utolsó képről van szó], illetve tudunk újat is feltölteni, ha olyan helyre nyomunk, ahol még nincs kép [tehát ahol a kis plusz jelet látjuk].



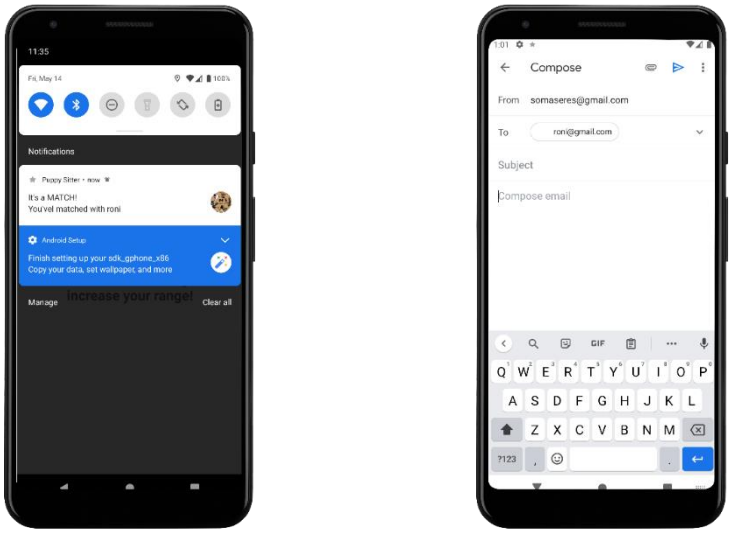
3. ábra: Már meglévő képre kattintva az első képernyő fogad minket, ahol választhatunk, hogy törölni, vagy módosítani szeretnénk azt. Ha ez a kép az utolsó kép, ami még maradt, akkor törlésre nincs lehetőség, ezt a második képernyő mutatja. Új kép feltöltésekor, illetve meglévő módosításakor a harmadik képernyőn látható dialógus fogad minket, ahol dönthetünk a kép megadásának módjáról, kamerával készítjük el, vagy kiválasztjuk a fájlok közül.

Ha beállítottunk mindent, amit szerettünk volna, akkor kezdődhet a válogatás! A válogató nézetben mindig egy, a másik típusú felhasználó adatait látjuk, először nagy nézetben a képeit, amik a képen látható „swipe” gesztúrával lapozgathatók [amennyiben van neki több is], de a képet felfele húzva megjeleníthetjük alatta a további adatokat is. Ekkor a kép alatt, az elfogadó, illetve elutasító gomb között megjelenik egy nyíl, amire nyomva visszahozhatjuk teljes képernyőre a képnézegető nézetet, de ugyanúgy mozgatható „swipe” gesztúrával is. A „YES” feliratú gombra nyomva értelemszerűen szeretnénk a másik típusú felhasználóval párba kerülni, a „NO” gombra nyomva pedig nem. Amennyiben már nincs több felhasználó a körzetünkben, erre utaló üzenetet kapunk a képernyőre.



4. ábra: A bal oldali képernyőn a képnézegető látható, az adott gesztúrával lehet lapozni a képek között. A középső képernyőn a további adatok látszanak a példaprofilról, a nyílacskára nyomva [első gesztúra] újra visszajön teljes nézetre a képnézegető, lefele húzva [második gesztúra] fokozatosan engedhetjük le a képet. A jobb oldali képernyőn láthatjuk mi történik, ha nincs a közelünkben másik típusú felhasználó.

Amennyiben a másik felhasználó már igen-t mondott ránk, és mi is igent mondunk, egy értesítést kapunk, amire kattintva automatikusan megnyitja az email alkalmazást, címzettnek megadva az új párunk email címét, hogy kapcsolatba léphessünk vele, és megbeszélhessük a további részleteket. Amennyiben nem mi matchelünk a másikkal, úgy a profil nézetbe való belépéskor a legutóbbi ottlétünk alatt keletkezett párokat begyűjti az alkalmazás, és megkapjuk róluk az értesítéseket.



5. ábra: A bal oldalon látható az értesítés a párba állásról, a jobb oldalon pedig a rányomva megnyitott email app.

Felhasznált technológiák

- A képernyőkön **Fragment**-eket jelenítek meg
- **Firebase** használata a következőkre:
    - **Firebase Authentication:** regisztráció és bejelentkezés kezelése
    - **Firebase Firestore:** alapvető adatok tárolása [név, email, leírás, stb.]
    - **Firebase Storage:** fájl tárolás [képek tárolása]
- Firebase hívások aszinkron kezelésére a **Kotlinx-Coroutines** könyvtárat használtam
- Intent használata a **képek feltöltéséhez** – **Kamerával** kép készíthető, illetve **a fájlok közül lehet választani** képet
- Képek betöltése a legtöbb helyen a **Glide** segítségével van megoldva
- Engedélykezelésre a **PermissionDispatchers** könyvtárat használtam
- A navigációt az appon belül **Navigation Componenttel** oldottam meg
- Saját **LocationService** használata a hely lekövetéséhez
- **Fused Location API** használata helymeghatározásra
- **Notificationök** használata a matchek jelzésére

Fontosabb technológiai megoldások

A legnagyobb problémám a dolgok betöltésével és azoknak a megvárásával volt. Mikor már bejelentkezett a felhasználó, még megkellett várni, míg a Firestore-ból lejönnek az adatok, illetve letöltődik a képének az elérhetősége is. Rengeteg null-pointer, cannot cast null és hasonló hibát kaptam eleinte, majd elkezdtem az onsuccesslistener-be berakni a dolgokat, de aztán ez eléggé csúnyának bizonyult. Később sok helyen átírtam, illetve az új dolgokat már a Kotlinx-Coroutines segítségével oldottam meg, ahol couroutine környezetben lehet a Task-ok illetve Deferred-eket megvárni az await() függvénnyel, így már sokkal átláthatóbbá és egyszerűbben kezelhetővé vált a kód.

A másik nagy problémám a képekkel volt, random elforgatta őket néha a BitmapFactory, illetve baromi nagy méretű bitmapeket állított elő, ami a hálózatot is túlterhelte, és lassította az appot is, ezért átálltam Glide-ra ahol szükséges volt.