|  |  |  |
| --- | --- | --- |
|  | **Eötvös Loránd Tudományegyetem**  Informatikai Kar  Algoritmusok és Alkalmazásaik  Tanszék |  |

**Webes munkaközvetítő alkalmazás**

|  |  |
| --- | --- |
| *Témavezető:*  Nagy Sára  mestertanár | *Szerző:*  Járvás Tamás Szilveszter  programtervező informatikus BSc |

Budapest, 2018

Tartalomjegyzék

1. **Bevezetés4**
2. **Felhasználói dokumentáció5**

2.1 Látogatóknak elérhető felület5

2.1.1 Szakemberlista7

2.1.2 Szakember profilok8

2.1.3 Felhasználói profilok9

2.1.4 Felhasználói regisztráció10

2.1.5 Szakember regisztráció11

2.1.6 Bejelentkezés12

2.2 Szakembereknek elérhető felület12

2.2.1 Üzeneteim13

2.2.2 Profilom15

2.2.3 Feladataim16

2.2.4 Időpontjaim17

2.3 Felhasználóknak elérhető felület18

2.3.1 Szakemberek értékelése18

2.3.2 Kedvencek19

2.3.3 Időpontfoglalás19

2.4 Adminisztrátoroknak elérhető felület20

2.4.1 Admin felület20

**3. Fejlesztői dokumentáció23**

3.1 Tervezés23

3.2 Telepítés25

3.2.1 Szerver elindítása25

3.2.2 Frontend elindítása26

3.3 Használt eszközök/ technológiák27

3.4 Szerver31

3.4.1 A projekt felépítése31

3.4.2 Az adatbázis szerkezete32

3.4.3 Entitások33

3.4.4 Kontrollerek37

3.4.5 Service és Repository39

3.5 Frontend/ megjelenítés41

3.5.1 Komponensek42

3.6 Tesztelés47

3.6.1 Bejelentkezés48

3.6.2 Regisztráció48

3.6.3 Keresés49

3.6.4 Egyéb funkciók50

3.7 Továbbfejlesztési lehetőségek51

**4. Irodalomjegyzék52**

# Bevezetés

A szakdolgozat témája egy webes munkaközvetítő alkalmazás, a program célja, hogy a hétköznapokban gyakran előkerülő probléma, a ház körüli munkákra való szakemberkeresés folyamatát könnyítse. Manapság, ha az ember otthona bővítésére készül vagy esetleg meghibásodik valami nála és szerelőt kell, hogy hívjon, gyakori probléma, hogy nem tudja hová forduljon. A legjobb esetben van egy jó ismerős, aki minden ilyen jellegű munkát elvégez, de aki nincs ilyen helyzetben az csak hirdetési újságokban vagy mindenféle apróhirdetési portálon tud keresgélni. Ezekkel az a legnagyobb baj, hogy nincs semmiféle visszajelzés az adott szakember tudásáról, teljesítményéről, sokszor megbízhatatlanok vagy több kár van belőlük, mint haszon, így nagyon nehéz a jó szakembert megtalálni.

A program erre adna megoldást, a lehető legközvetlenebb módon összekapcsolja a különféle szakembereket kereső felhasználókat és a munkát kereső szakikat, így előnyhöz juttatva mindkét felet. Az imént említett problémára pedig az alkalmazásban lévő értékelési rendszer kínál megoldást, egyből látható a munkás átlagos értékelése, illetve az értékeléséhez tartozó kommentekből sokkal pontosabban meg lehet állapítani a rátermettségét.

A program igyekszik megelőzni a kapcsolatfelvétel során leggyakrabban elhangzó kérdéseket: „Hova tudna kijönni?”, „El tudná-e végezni ezt a feladatot?”, „Mikor tudna kijönni?”. A szakembereknek lehetősége van a szolgáltatási helyeiket megadni, ahol alapértelmezetten elérhetőek. Lehetőségük van emellett a jellemző feladataikat megadni, irányárral együtt, hogy a jövőbeli ügyfeleik képet kapjanak róluk, vagy ha esetleg valamilyen speciális feladatot is vállalnak a szakterületükön belül. A harmadik aspektus pedig a szabad időpontjaik megadása, melyből a felhasználók választhatnak és le is foglalhatnak, így akár minimális kommunikációval megoldható egy amúgy több hívásból álló folyamat.

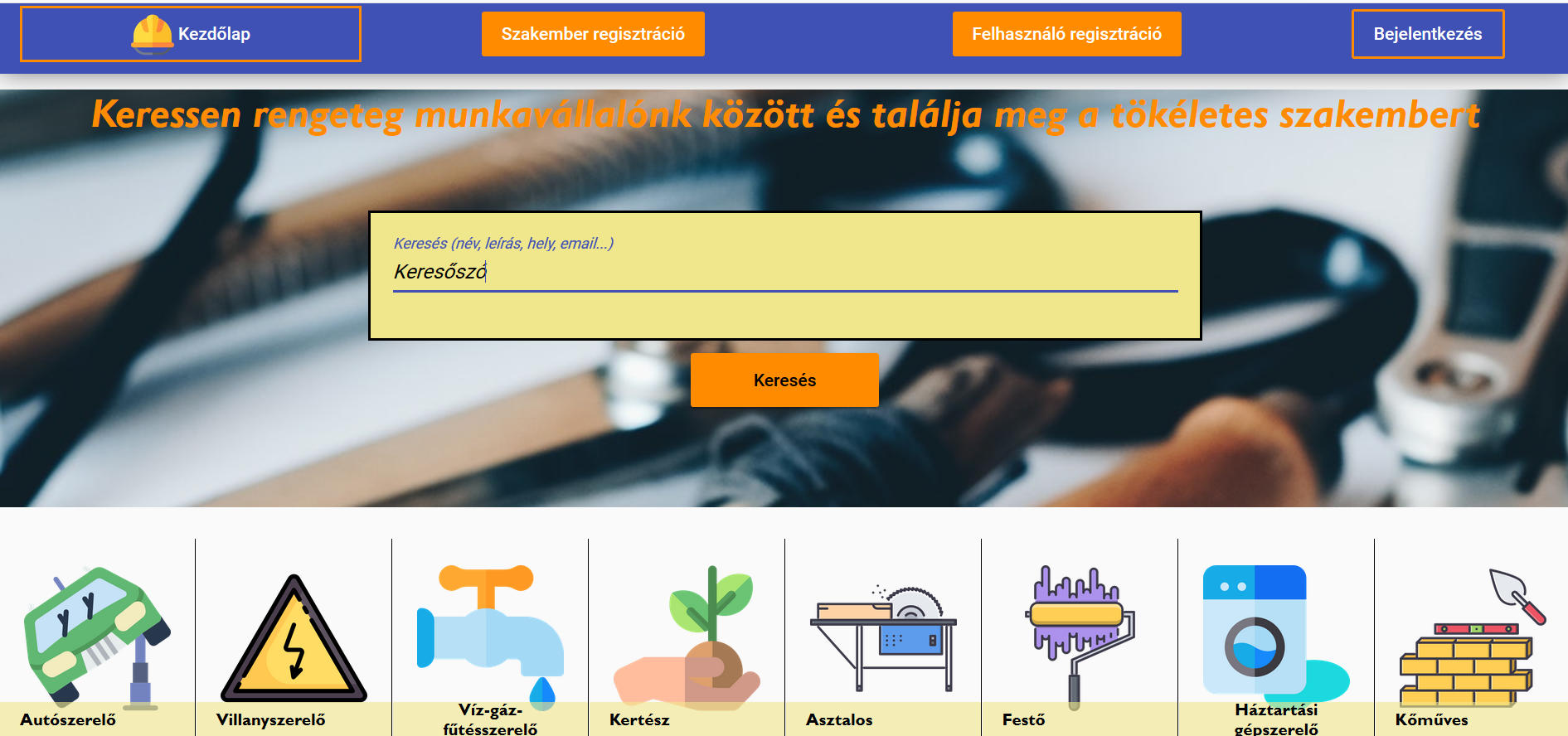
A szakdolgozatot webes környezetre készítettem, véleményem szerint ez a legplatformfüggetlenebb és széleskörűen elérhető bárhonnan, akár egyszerűbb mobileszközről, akár számítógépről legyen szó. A továbbiakban a fent bevezetett webalkalmazást fogom bemutatni felhasználói és fejlesztői szemszögből.

# Felhasználói dokumentáció

A futó alkalmazás használatához az egyetlen dolog, amire szükség van az egy böngészőprogram. A tökéletes megjelenítéshez ajánlott egy modern böngésző használata, ami jelenleg az Internet Explorer régebbi verzióin kívül bármely más, azaz a Google Chrome, Mozilla Firefox, Microsoft Edge stb. mind megfelelőek.

Az alkalmazás három szerepkör köré szerveződik, mindegyikhez egyedi jogosultságok, illetve funkciók tartoznak. Az adminisztrátorok a két másik típusú felhasználó közötti kapcsolat zavartalanságáért felelnek, felülvizsgálják a regisztrált felhasználókat, illetve karbantartják az adatbázist. A szakemberek feladata a profiljuk karbantartása és minél részletesebb kitöltése, a felhasználók feladata pedig a szakemberekkel való kapcsolatfelvétel, majd később értékelésük, ezen szerepkörök később a felületeikkel együtt részletezésre kerülnek.

## Látogatóknak elérhető felület

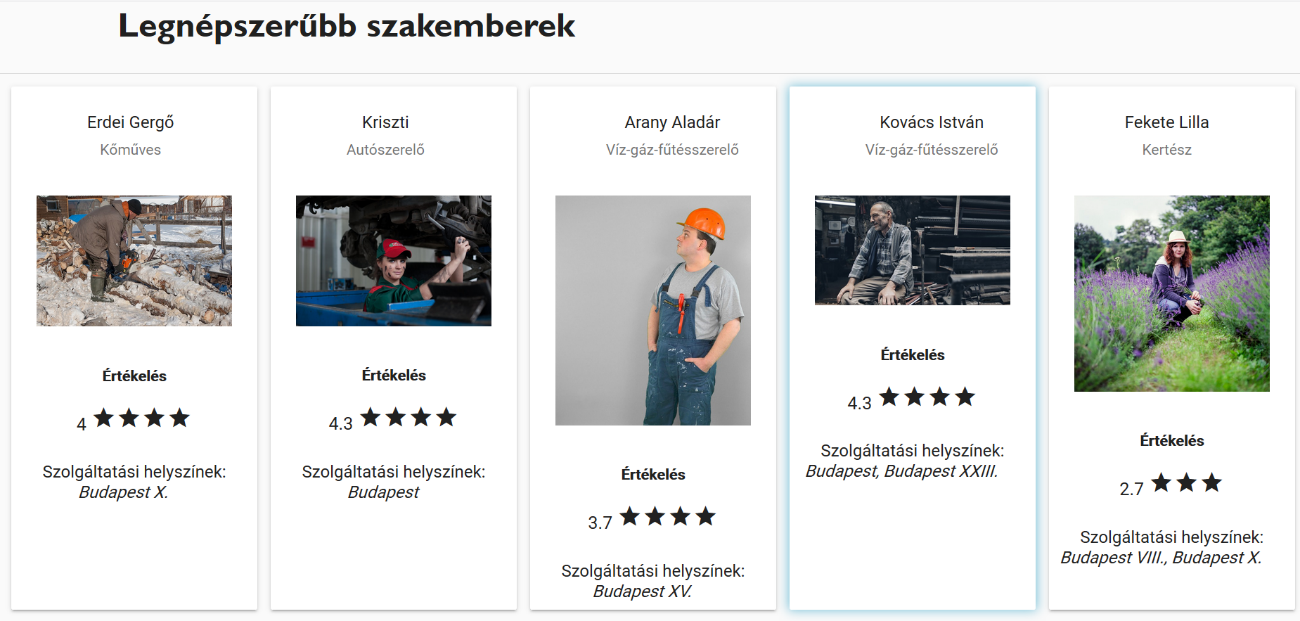
Az oldal megnyitásakor a kezdőlap tárul a bejelentkezetlen látogató elé, a felső állandó sávban a menüvel, keresőmezővel, kategóriákkal és a legnépszerűbb szakemberekkel.

**1. ábra** - A főoldal a menüvel, keresőmezővel és kategóriákkal

A menüsávban elérhető kezdőlap gombbal bármikor visszatérhetünk ide, a két regisztrációs gomb értelemszerűen a megfelelő regisztrációs űrlapra visz, míg a bejelentkezés gomb a bejelentkezési űrlapra.

A keresőmezőben a látogatónak lehetősége van keresni a szakemberek nevében, leírásában, email címében, illetve a szolgáltatási helyeikben. A keresés gomb aktiválódik, ha a látogató minimum három karaktert beír a mezőbe. A keresés eredménye a kategóriáknál a későbbiekben látható listában jelenik meg (3.ábra), illetve lehetőség van visszalépésre, ha a keresésünk eredménytelen volt.

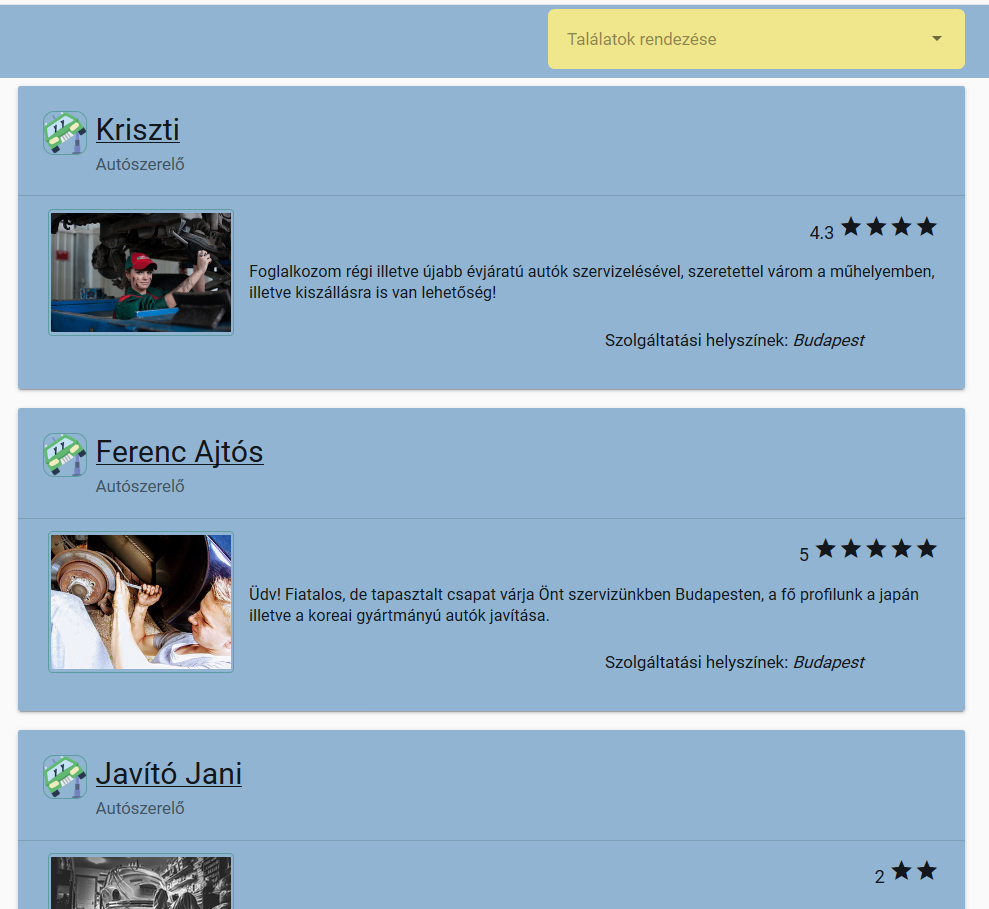
A kategóriák kiválasztása az adott kategóriában lévő összes szakembert felsorolja, a jelenleg elérhető kategóriák:

* autószerelő
* villanyszerelő
* víz-gáz-fűtésszerelő
* kertész
* asztalos
* festő
* háztartási gépszerelő
* kőműves

**2. ábra -** A legnépszerűbb szakemberek

A főoldal alsó részén a legnépszerűbb szakemberek kártyáit láthatjuk, ez a mennyiségileg a legtöbb értékelést kapott szakemberek öt főből álló listáját jelenti, kategóriájukkal, értékeléseik átlagával és szolgáltatási helyeikkel együtt.

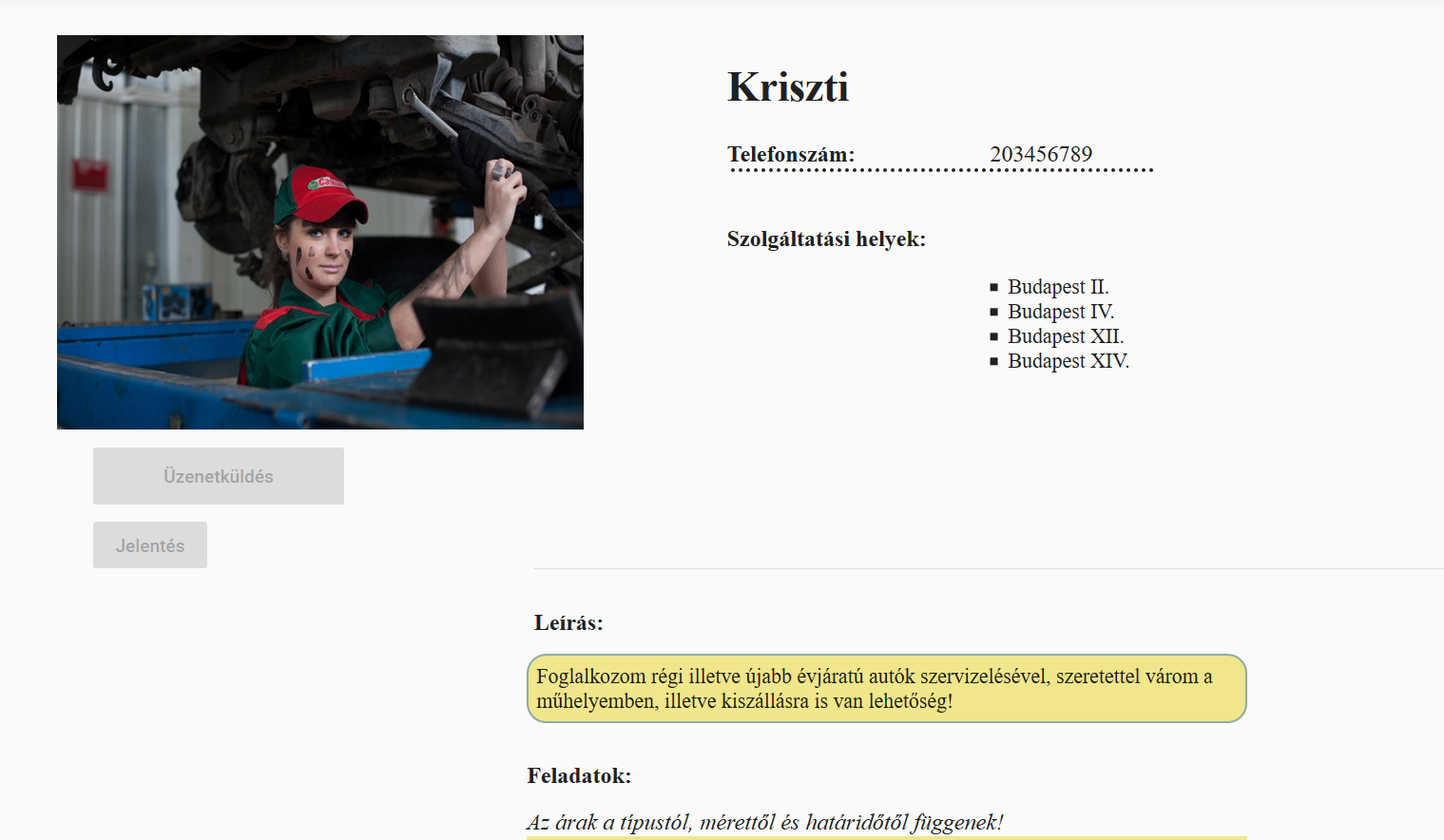
### Szakemberlista

Ha a látogató valamely kategóriára kattint vagy a keresést használja a találatok egy listában jelennek meg, a szakember néhány részletével együtt.

**3. ábra -** Szakemberlista

A listában a szakember neve, kategóriája, leírása, értékeléseinek átlaga, illetve szolgáltatási helyei jelennek meg. A találati lista rendezhető név szerint A-Z-ig vagy Z-A-ig, illetve értékelés szerint növekvő vagy csökkenő sorrendben.

### Szakember profilok

A szakemberlistán a névre vagy képre, illetve a legnépszerűbb szakemberekre kattintva a látogató az adott szakember részletes profiljára kerül.

**4. ábra** - Részletes szakember profil

Itt olvasható többek között a részletes leírás, az összes hely listázva, a telefonszám. Alatta a korábban említett jellemző feladatok táblázata látható, a feladatok nevével, illetve irányárával, mely később még pontosításra szorul a kapcsolatfelvétel folyamán. Az itt látható üzenetküldés, illetve jelentés gombok bejelentkezés nélkül nem elérhetőek, emiatt kiszürkítve jelennek meg és egy néhány másodpercig előjövő tipp is tájékoztatja a látogatót, hogy jelentkezzen be a gombok használatához.

**5. ábra** - Szakember feladatai

A profil utolsó harmadán a felhasználók által küldött értékelések listája látható, az öt fokozatú skálán a csillagokkal jelölve a fokozatokat, illetve a rövid szöveges értékeléssel. A profiloldalon alapértelmezetten a legutóbbi három értékelés jelenik meg, a lista alatt található „Még több értékelés…” gombbal viszont megjeleníthetjük az összes többit is. Az adott értékelésre kattintva a következőkben részletezett felhasználói profilra kerülünk.

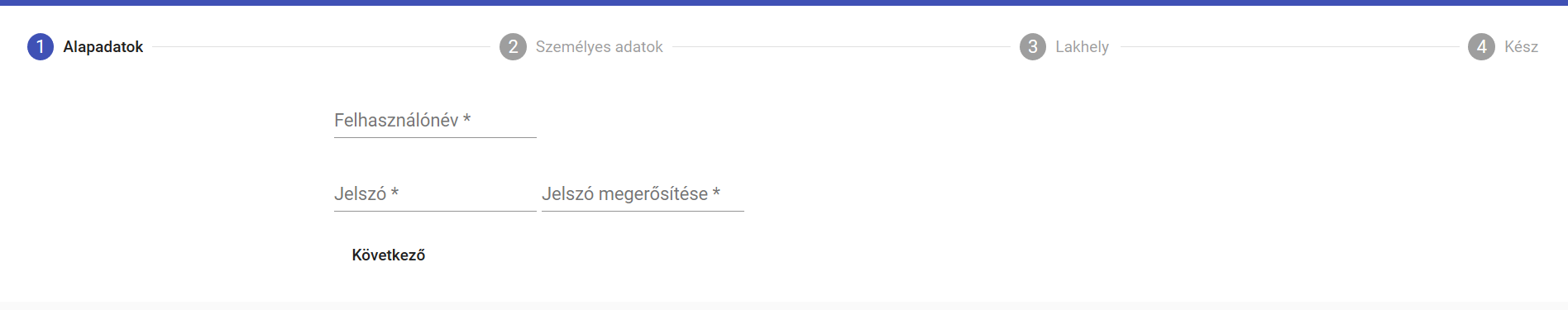
**6. ábra** - A szakember részletes értékelései

### Felhasználói profilok

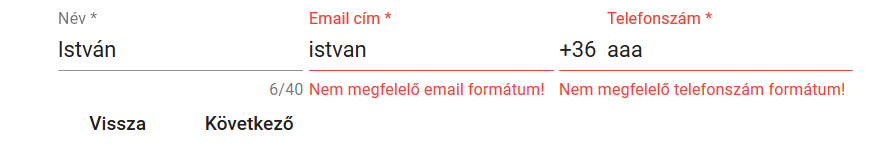
**7. ábra** - Felhasználói profil

A felhasználói profil hasonló a szakemberek profiljának első részéhez, a gombok ugyanúgy kikapcsolt állapotban vannak, amíg be nem jelentkezünk.

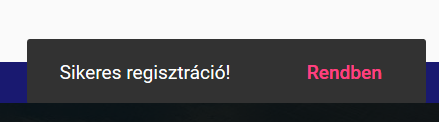
### Felhasználói regisztráció

A regisztrációs felület négy lépcsőből áll, melyeken lineárisan lehet csak haladni, azaz a következő lépésre csak akkor enged menni, ha az összes űrlapmező megfelelően ki lett töltve.

**8. ábra** - Négy lépéses regisztráció

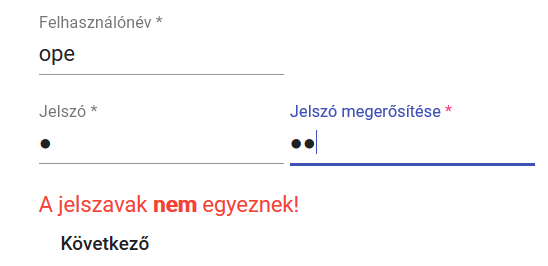
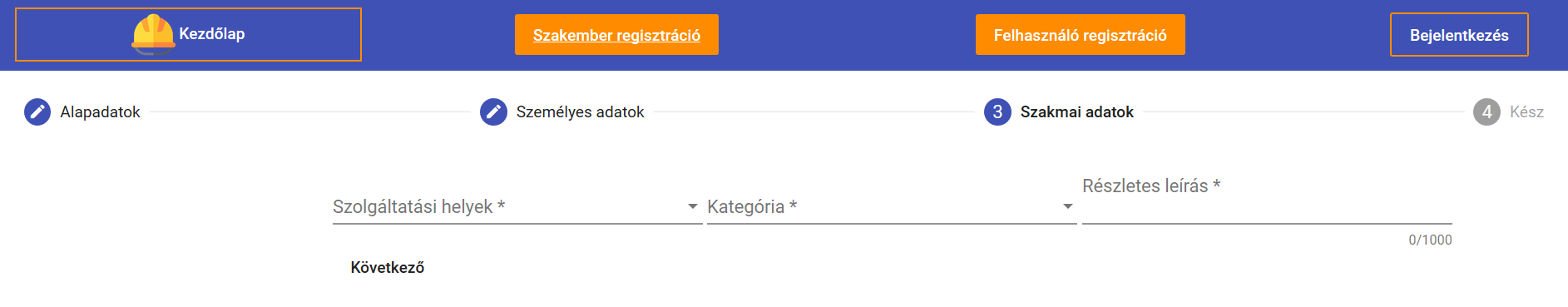
Az „Alapadatok” részen egy szokásos felhasználónév és jelszó párost kell megadni, illetve a jelszó megerősítését, melyet ellenőriz is a program. A személyes adatok elnevezésű lépésben a név, email cím, telefonszám hármast szükséges megadni kötelezően. Az email cím mezőbe csak email formátumot lehet megadni azaz „xx@yy” alakot fogad el az űrlap, illetve a telefonszámhoz is csak számokat lehet írni, maximum 9 számjegy hosszúságban. Ha ezek nem teljesülnek vagy egy mező üresen marad, akkor a program piros színnel és hibaüzenettel jelez.

**9. ábra** - Hibajelzések regisztrációnál

A „Lakhely” fülön a felhasználónak ki kell választania az előre megadott listából a lakhelyének megfelelő kerületet/ várost. Mivel a felhasználónak pontosan egy lakhelyet lehet választania, így a szakembereknek könnyen meghatározható, hogy elérhető közelségben vannak-e. A negyedik lépésben lenullázhatjuk az eddig beírtakat, vagy véglegesíthetjük a regisztrációnkat. Ezután kapunk egy alulról felbukkanó üzenetet (másnéven snackbar) a sikerességről. Ha már létezik ilyen felhasználónév, akkor pedig a regisztráció sikertelenségéről.

**10. ábra** - Sikeres regisztráció snackbar

### Szakember regisztráció

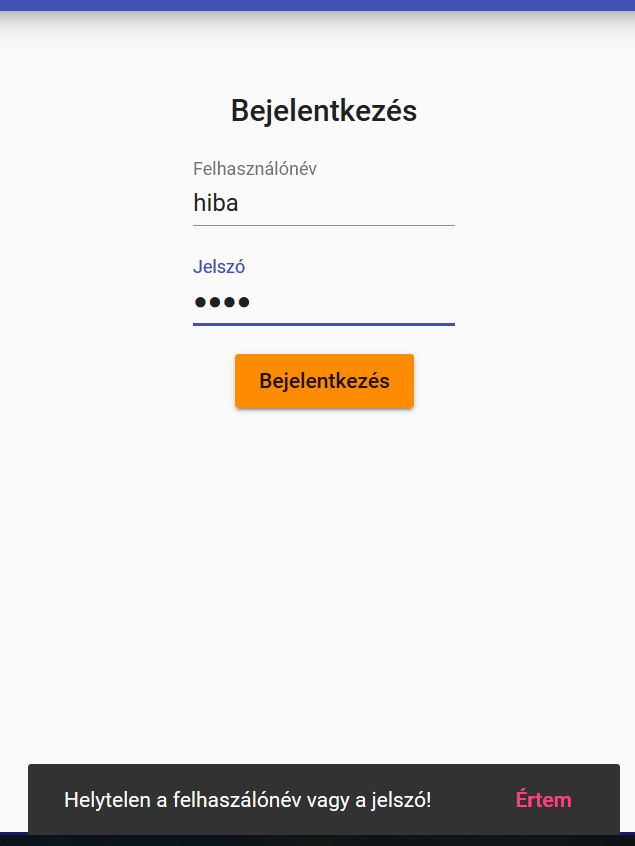
A szakemberek regisztrációja nagyban hasonlít a korábban ismertetett felhasználói regisztrációra, viszont a harmadik lépésben lényegesen különbözik. Az első két pont ugyanolyan hitelesítésekkel és ellenőrzésekkel rendelkezik, mint az általános felhasználóknál, pl. jelszó megerősítés. A „Szakmai adatok” fülön kell választania a jelenleg létező nyolc kategóriából, melyet később nem változtathat. A szolgáltatási helyeknél a felhasználóknál látott listából választhat és itt lehetőség van többszörös választásra is, hiszen előfordulhat, hogy a szakember több kerületben vagy városban is elérhető. Az utolsó mezőben egy maximum 1000 karakterből álló részletes leírást kell írnia a szakembernek. Fontos még megemlíteni, hogy mind a szakemberek, mind a felhasználók egy alapértelmezett profilképet kapnak sikeres regisztráció után, melyet később a profil szerkesztésével változtathatnak.

**12. ábra** - Szakember regisztráció

**11. ábra** - Jelszó megerősítés

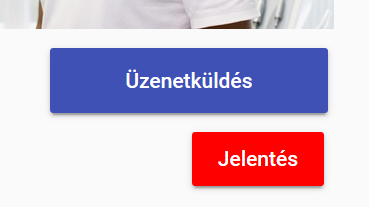
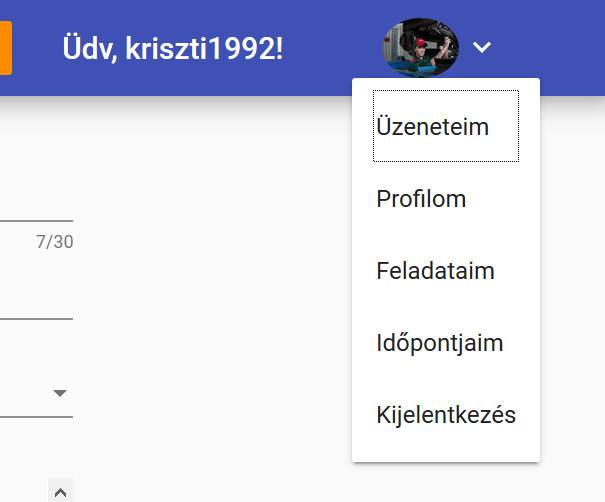
A program rendszerének fontos aspektusa, hogy a szakember a regisztrációja után nem kerül bele egyből az elérhető szakemberek listájába, hanem először egy adminisztrátornak engedélyeznie kell a profilját. Ha a profilját megfelelően töltötte ki, akkor az engedélyezést egy üzenet fogja jelezni, ha valami módosításra van szükség, akkor az adminisztrátor ezt levélben jelzi. Az engedélyezés után a profil megjelenik a szakemberlistában, illetve kereshető lesz.

### Bejelentkezés

A bejelentkezés a bármely más oldalakon is használt és megszokott bejelentkezési űrlapok formáját követi. Helytelen adatok megadása esetén egy újabb snackbar figyelmeztet a hibás felhasználónév vagy jelszó kiírással.

**13. ábra** - Bejelentkezés és hibaüzenet

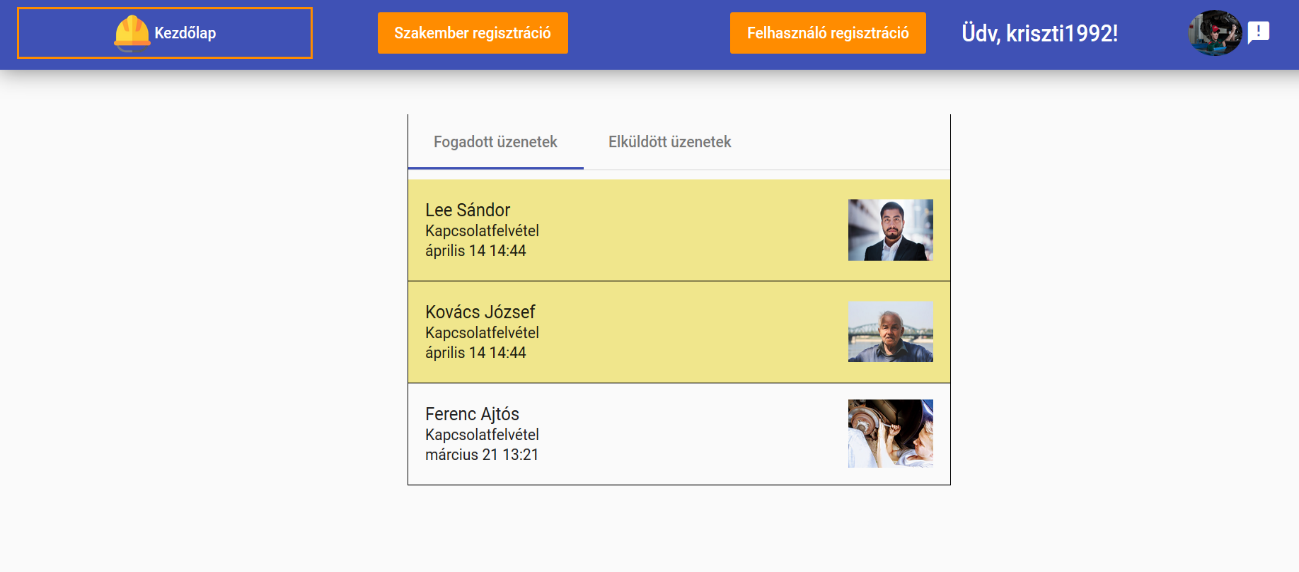
## Szakembereknek elérhető felület

A bejelentkezett szakemberek a jobb felső sarokban található privát menün kívül többé-kevésbé ugyanazt a felületet láthatják, mint a látogatók, ugyanakkor az eddig kikapcsolt gombokat természetesen használhatják.

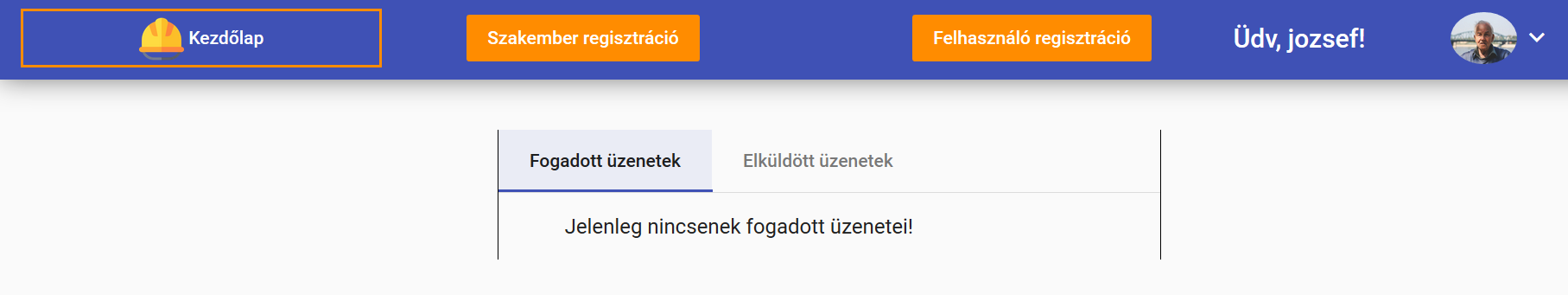
**14. ábra** - Szakember menü

**15. ábra -** Bejelentkezés után engedélyezett gombok

### Üzeneteim

Itt találjuk a fogadott, illetve az elküldött üzenetek listáját, (a fenntarthatósági szempontokat is figyelembe véve) a legutóbbi 1 hónapból.

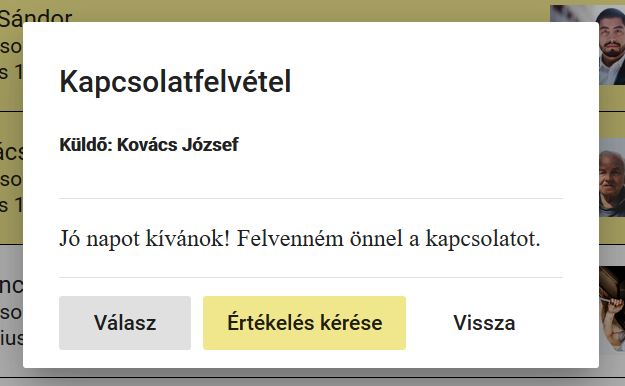
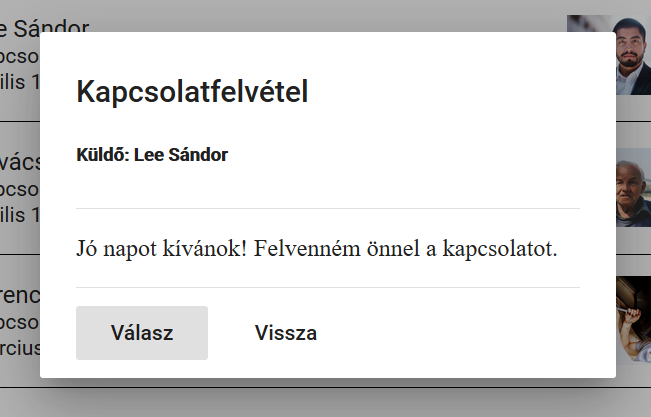
**16. ábra -** Fogadott üzenetek fül egy szakembernél

Az üzenetek listájában láthatjuk az üzenetek küldőjének nevét és profilképét, a levél tárgyát, illetve a küldés pontos időpontját. Ha van olvasatlan üzenetünk, akkor a privát menünk ikonja a 16.ábrán is látható módon a szokásos lefele nyíl helyett egy felkiáltójel ikonba megy át, így jelezve, hogy új üzenetünk van. Az eddig olvasatlan üzeneteket sárga színnel jelöli a program, megnyitásuk után az üzenet visszavált az alapértelmezett fehér színre.

**17. ábra** - Értesítés, ha nincs fogadott üzenet

Ha üres a fogadott vagy a küldött üzenetek listája, akkor a 17.ábrán látható értesítést láthatjuk.

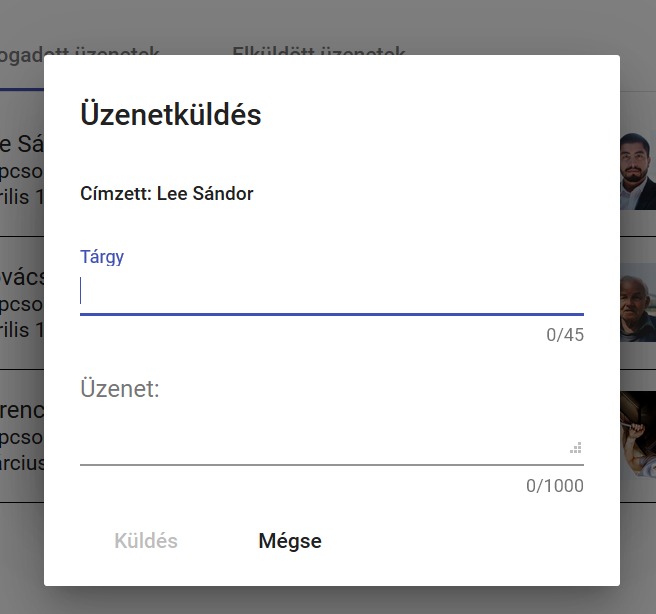
Az üzenetek megnyitásakor egy szokványos felugró ablak tárul elénk, melyben egy link vezet az üzenet küldőjére, illetve lehetőségünk van válaszolni.

Az „Értékelés kérése” gombbal a szakemberek értékelési kérelmet küldhetnek el az adott levelet küldő felhasználónak. Az értékelési rendszer a következőképpen alakul: a szakemberek értékelési kérelmet küldhetnek el azon normál felhasználóknak, melyektől még nem kaptak értékelést, ezt az előbb említett sárga színű gombbal tehetik meg. A felhasználó erről üzenetet kap, majd értékelheti a szakembert, ezt a későbbiekben részletezem. Ezzel a rendszerrel többé-kevésbé pontosabb értékeléseket lehet kapni, mintha bárki értékelhetne bármikor. A szakembernek célja lesz a minél jobb munkavégzés, illetve nem tud visszaélni azzal, hogy bárkinek elküldi az értékelési kérelmét, hiszen így kockáztatja, hogy rossz értékeléseket kap. Felhasználói szempontból nézve pedig nem tudunk bármikor értékelni, így megelőzve például a több felhasználótól érkező „lehúzó” értékeléseket, a rendszer arra törekszik, hogy ténylegesen olyanok között történjen az értékelés, akik munkakapcsolatban is voltak.

**20. ábra -** Felhasználó jelentése

**18. ábra** - Alap üzenet felugróablak

**19. ábra** - Üzenet ablak értékelési gombbal

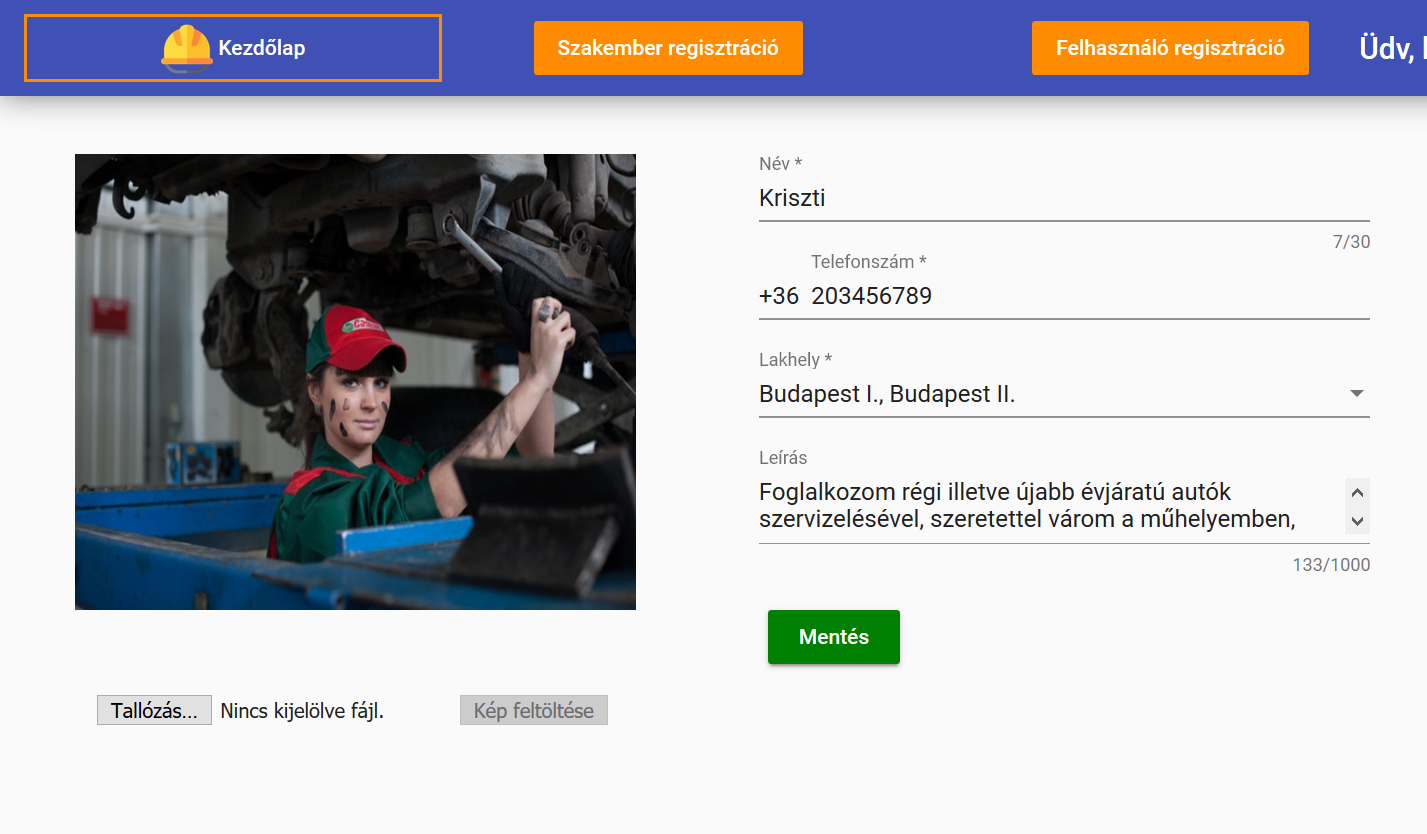
Ha a szakember ugyanakkor nem ajánlja fel az értékelést, például rossz munkavégzés esetén, akkor a felhasználóknak lehetősége van jelentést írni a profilján lévő piros jelentés gombbal. A jelentés természetesen az ellenkező irányba is működik, ha a szakembernek volt valamilyen problémája a felhasználóval. Ezeket a jelentéseket az adminisztrátorok látják és intézkednek azoknak megfelelően.

**21. ábra** - Egyszerű üzenetküldés

Az utolsó lehetőségünk, de egyben a legegyszerűbb is az egyszerű üzenet írása, melyet a „Válasz” vagy az „Üzenetküldés” gombra kattintva érhetünk el. Kötelező megadni tárgyat és a levél tartalmát is természetesen, előbb 45, utóbbi 1000 karakterben van maximalizálva.

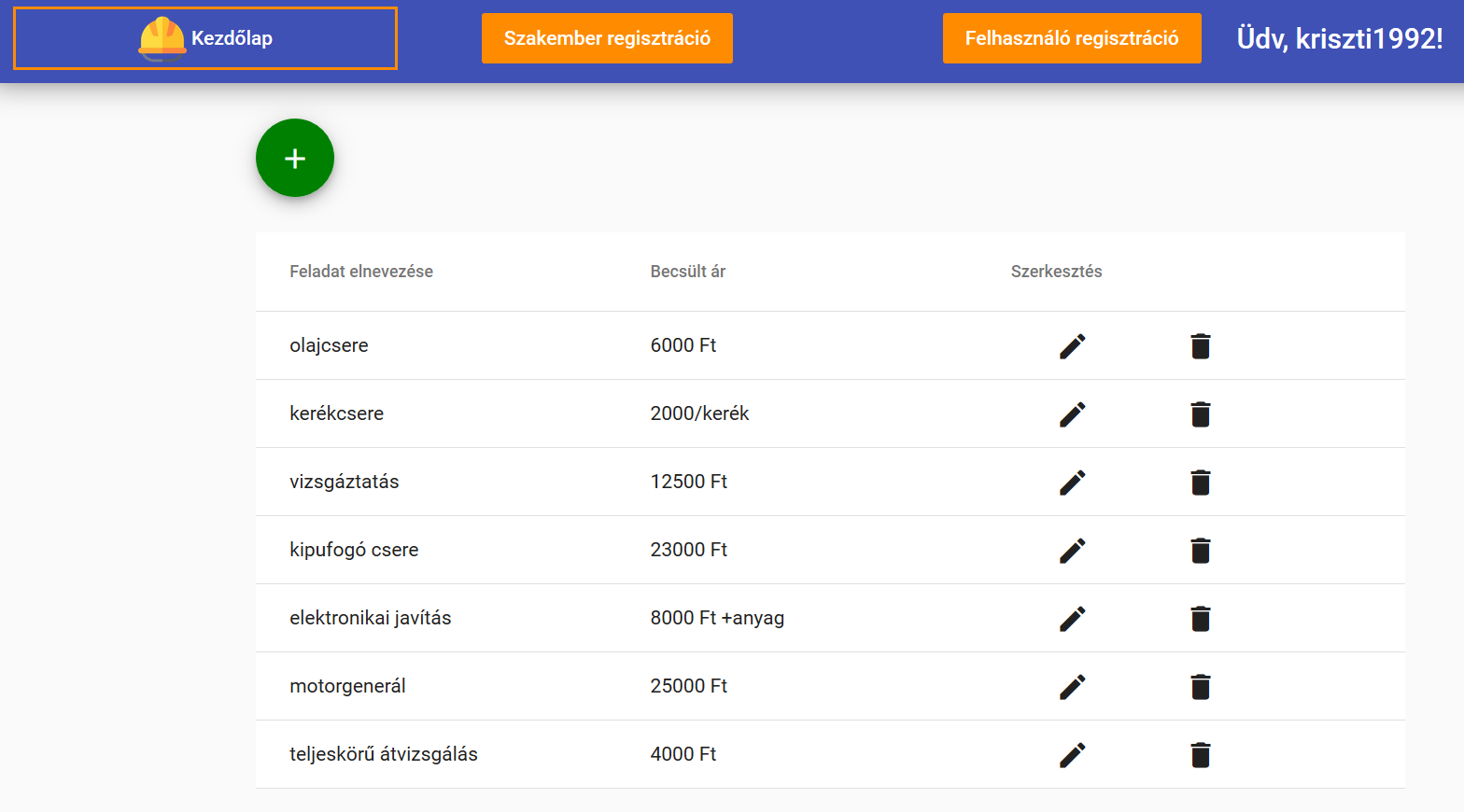
### Profilom

A menüpontra lépéskor a szakember megnézheti az éles profilját, amit a többi felhasználó is lát, de az Üzenetküldés/ Jelentés gombok helyett a „Profil szerkesztése” gomb jelenik meg. A gombra kattintva a szerkesztési űrlapra kerül a szakember, ahol űrlapmezőkként jelennek meg a profil szerkeszthető részletei. A regisztrációnál megismert formátumok és korlátozások persze itt is élnek. Itt lehet beállítani új profilképet is: a „Tallózás” gombra kattintva válasszuk ki a kívánt képet, majd töltsük fel a „Kép feltöltése” gombbal. A kép, illetve a profil egyéb részleteinek változtatását a zöld színű „Mentés” gombra kattintva véglegesíthetjük.



**22. ábra** Szakember profil szerkesztése

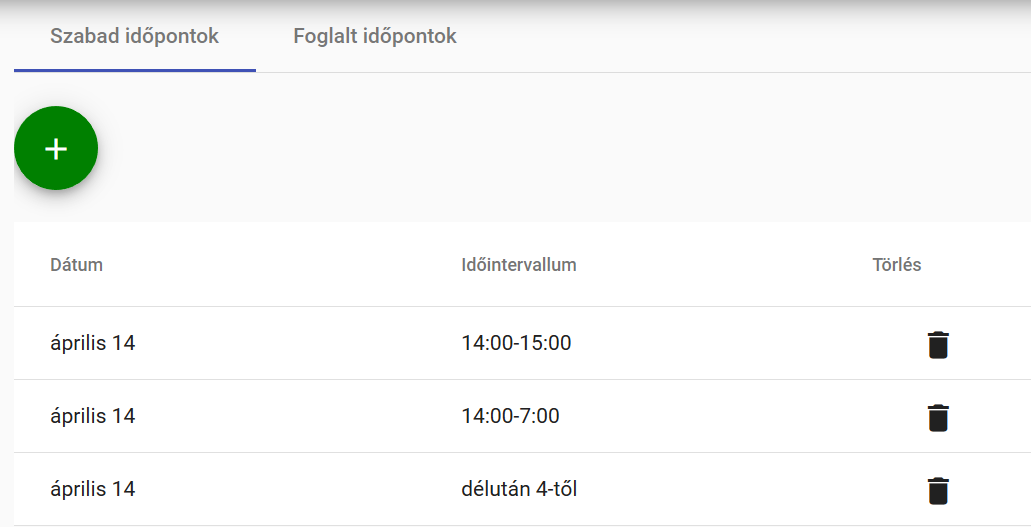
### Feladataim

Itt a szakember feladatainak teljes listája látható, minden tételnél lehetőség van szerkesztésre, illetve törlésre is.

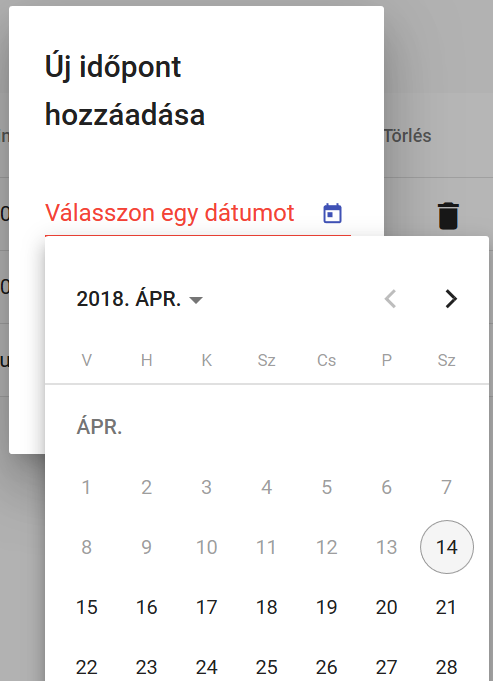
**23. ábra** - Szakember feladatainak táblázata

A plusz gombra kattintva pedig hozzáadhatunk új feladatot a feladat nevének, illetve becsült árának megjelölésével, mindkét attribútum max. 30 karakter lehet.

### Időpontjaim

A szakember időpontjai egy előbbihez hasonló táblázatban láthatóak, viszont itt különbséget teszünk szabad, illetve foglalt időpontok között. A táblázat tetején található fülekkel lehet ezek között váltogatni.

**24. ábra** Szabad időpontok táblázata

A korábban már látott módszerrel a zöld színű plusz gombbal lehet új időpontot hozzáadni. A párbeszédablak első mezőjében egy napot kell kiválasztani, melyben segítséget nyújt a jobb szélén látható naptár ikon, amivel egy dátumválasztó felületre érkezünk. Új dátum kiválasztásakor természetesen csak a maival megegyező vagy későbbi napot lehet választani. Az időintervallum mező pedig szöveges adattal kitölthető, így lehetőség van (pl.: délután 5 után) egyedi lehetőségek megadására.

**25. ábra** Új időpont hozzáadása párbeszédablak

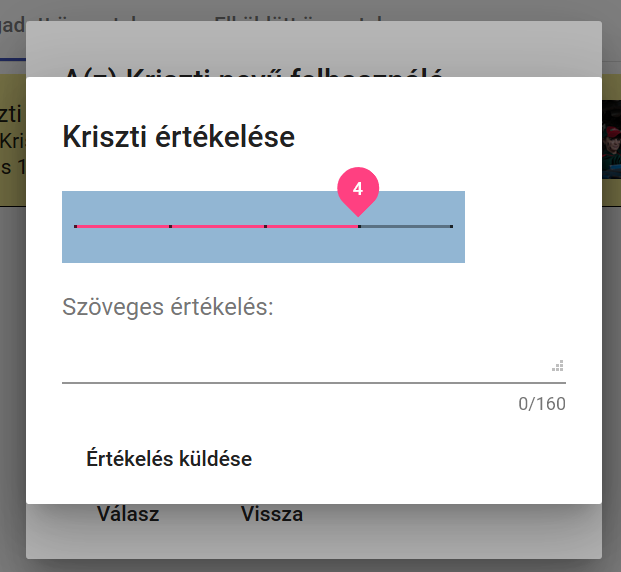
Az utolsó menüpont a triviális funkciójú „Kijelentkezés” gomb.

## Felhasználóknak elérhető felület

A felhasználók bejelentkezés után a szakembereknél ismert módon érhetik el a saját menüjüket, melynek pontjai kissé változtak, a következő módon:

* Üzeneteim
* Profilom
* Kedvenceim
* Kijelentkezés

### Szakemberek értékelése

Az üzenetek pontosan ugyanúgy működnek, mint a már korábban ismertetett szakembereknél, az egyetlen lényeges különbség az értékelés folyamata.

**27. ábra** - Értékelési párbeszédablak

**26. ábra** - Értékelési kérelem

Ha a szakembertől érkezik egy értékelési kérelem, azt a felhasználó a 26.ábrán látható üzenetként látja és az „Értékelem” gombra kattintva juthat el a 27.ábrán látható értékelési párbeszédablakba. A rózsaszínű csúszka segítségével értékelheti a szakembert az 1 és 5 közötti skálán, illetve opcionálisan megadhat egy maximum 160 karakterből álló rövid szöveges értékelést is, ami a szakember profilján meg is jelenik azonnal.

### Kedvencek

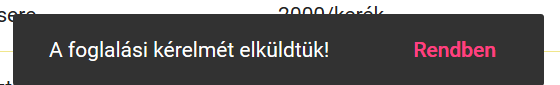
A „Profilom” menüpont megegyezik a szakembereknél már részletezett menüponttal, így azt nem is részletezném.

A „Kedvenceim” menüpontra kattintva az eddig a kedvencekbe tett szakemberek listája kerül elénk, a korábban már látott, rendezhető lista stílusban. A felhasználók bármely szakembert elhelyezhetik a kedvenceik között, ha a szakember profilján lévő piros szív ikonra kattintanak, illetve ugyanígy ki is vehetik a kedvenceik közül, ha a fekete szívre kattintanak.

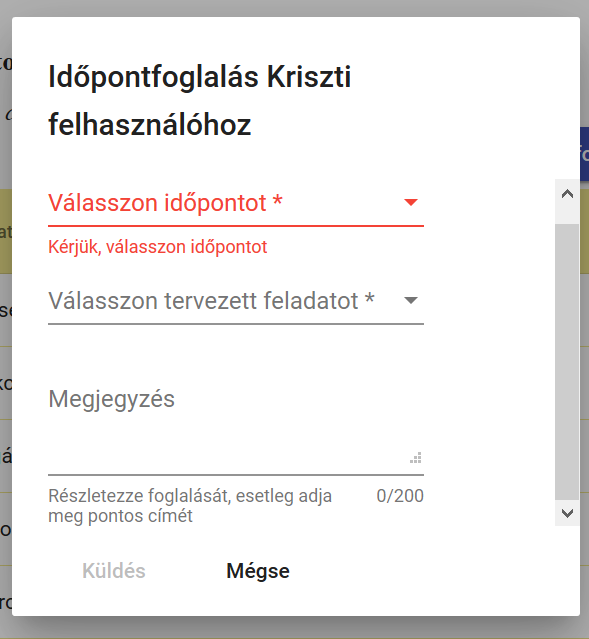
**29. ábra** - Szakember kikedvelése

**28. ábra** - Szakember bekedvelése

### Időpontfoglalás

A felhasználóknak lehetőségük van a szakemberek szabad időpontjaiból foglalni, ezt a szakember profilján látható „Időpontfoglalás” gombra kattintva tehetik meg. Ekkor előjön a 31.ábrán látható időpontfoglalási párbeszédablak, ahol, ha nincsen szabad időpont egy piros felirat tájékoztat erről. Ha van, akkor a felhasználónak kötelező választania egy időpontot és egy feladatot a legördülő listákból, illetve opcionálisan megadhat egy max. 200 karakterből álló megjegyzést, amiben részletezheti kérését vagy megírhatja a pontos címét akár.

**30. ábra** - Kérelem elküldve snackbar

A kérelem elküldése után a szakember üzenetet kap az időpontfoglalási kérelemről, amit elfogadhat vagy elutasíthat, erről a felhasználó rendszerüzenetet kap, illetve elfogadás esetén az időpont kikerül a választható időpontok közül és a szakember foglalt időpontjaihoz adódik.

**32. ábra** - Szakember időpontfoglalási üzenete

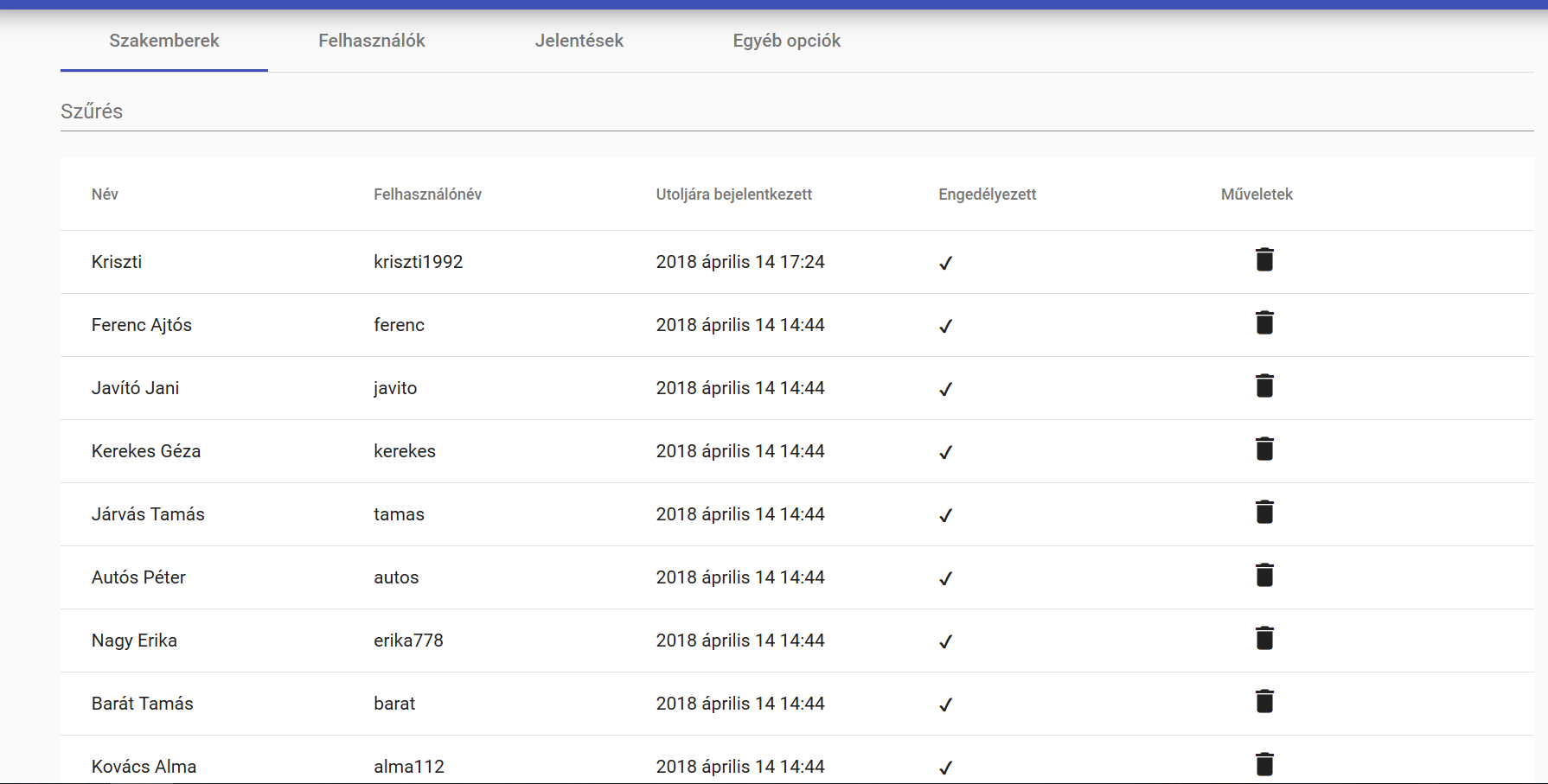
**31. ábra** - Időpontfoglalási párbeszédablak

## Adminisztrátoroknak elérhető felület

Az adminisztrátorok is ugyanazt látják, ugyanúgy böngészhetnek az oldalon, mint a másik két szerepű felhasználó, illetve „Üzeneteim” menüponttal is rendelkeznek, a lényeges különbség a másik menüpontjukban rejlik, ami az „Admin felület”.

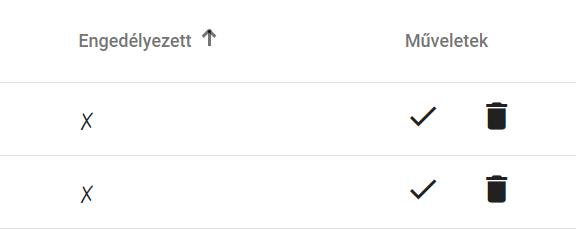
### Admin felület

Az admin felület egy négy fülből álló kezelőpanel, melyről az adminisztrátorok minden fontos információt elérhetnek ahhoz, hogy az alkalmazás zavartalan működését és fenntarthatóságát biztosítsák.



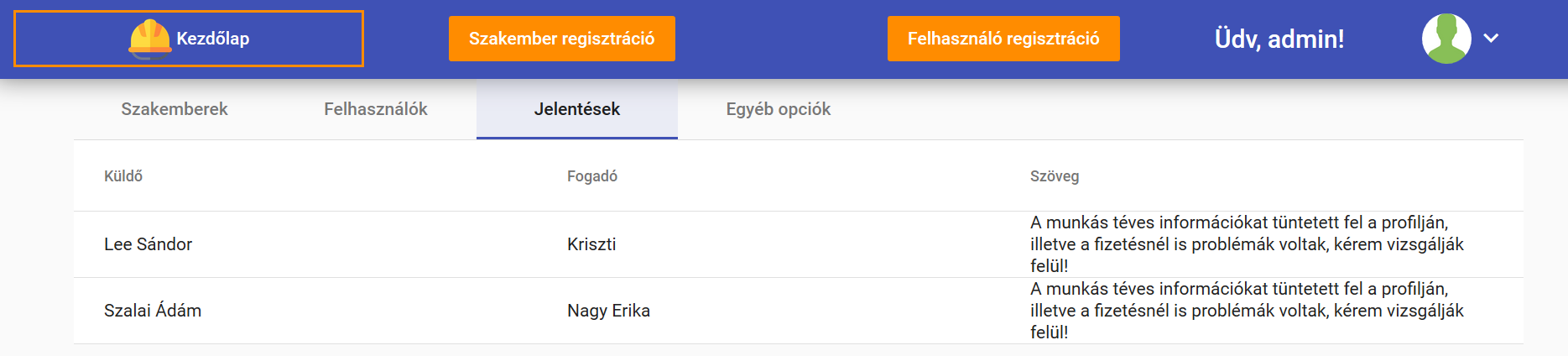
**33. ábra -** Admin felület

Az első fülön az összes szakember látható táblázatos elrendezésben, az oszlopok sorrendben: teljes név, felhasználónév, utoljára bejelentkezés időpontja, engedélyezett-e a profil, illetve a műveletek oszlop. A műveleteknél a pipa ikonra kattintva az adott profilt engedélyezheti az adminisztrátor miután megbizonyosodott róla, hogy minden rendben van vele, illetve a kuka gombra kattintva egyszerűen törölheti is azt.

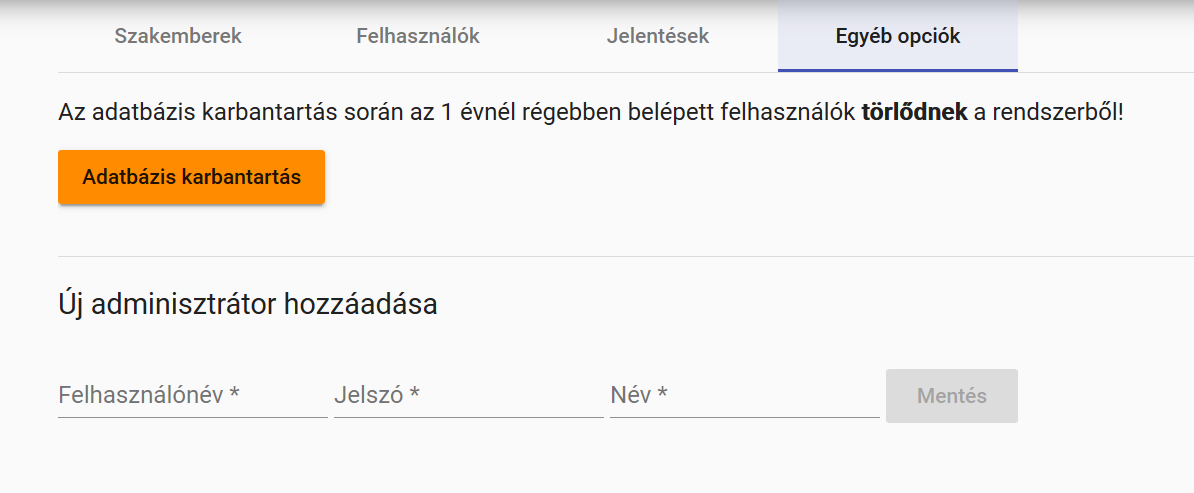
A második fülön a felhasználók táblázata található, kinézetében hasonló módon, mint a most látott szakemberek tábla, kivéve az engedélyezést, hiszen a sima felhasználóknál erre nincs szükség. Mindkét táblázatban lehet keresni a felső keresősáv használatával, rendezhető az oszlopok mentén és lapokra vannak bontva az átláthatóság miatt.

**34. ábra** - Szakember engedélyezése

A Jelentések fülön a felhasználók vagy akár a szakemberek által írt jelentések jelennek meg. Mivel a jelentés küldőjének és fogadójának profiljára is azonnali link vezet, ezért könnyen nyomon követhető a jelentés. A táblázatban a jelentés szövege is olvasható, ezért az adminisztrátor rögtön látja, hogy mi a probléma, amit orvosolni kell, és üzenetírással vagy akár profiltörléssel oldhatja meg.

Az utolsó fülön az egyéb opciókat találjuk, ami két funkciót takar: adatbázis karbantartás, illetve új adminisztrátor hozzáadása. Az adatbázis karbantartása gomb segítségével törölhetjük az összes olyan regisztrált felhasználót, aki 1 évnél régebben jelentkezett be, így megőrizhetjük a rendszer frissességét, ezzel elejét véve a túlzottan elszaporodó passzív profiloknak.

**35. ábra** - Jelentések listája

Az adminisztrátor hozzáadása funkcióval pedig új admint adhatunk hozzá a felhasználónév, jelszó, teljes név hármas megadásával.

**36. ábra** - Egyéb opciók fül

# Fejlesztői dokumentáció

## Tervezés

A szakdolgozati program megvalósításának tervezésekor a célom az volt, hogy a képzésen elsajátított tananyagot és a megszerzett ismereteket alkalmazva készítsem el a kitűzött feladatot. Az előbbieket egy erős alapnak használva, ugyanakkor a szakdolgozati követelményeket szem előtt tartva, a tanult módszereket fejlesztve és kibővítve próbáltam megoldani a feladatot.

A program alapvető szerkezete két részből áll: a Java Spring alapú szerver részből és az Angular alapú megjelenítési részből. Ezek a részek, főként a frontend réteg, négy további csoportra osztható, méghozzá a felhasználók szerepkörei szerint. A bárki számára elérhető felületekre, az egyszerű felhasználók által elérhető felületekre, a szakemberek által elérhető felületekre és végül az adminisztrátorok által elérhető felületekre.

A szerver elkészítésére az általam preferált Java programnyelv miatt a Javára épülő Spring Boot keretrendszert választottam. Ezen keretrendszer mellett álltak a következő érvek is: könnyű kezelhetőség, nagymértékű konfigurálhatóság, illetve részletes dokumentáltság (ezek mind gördülékenyebbé teszik a munkát). A szerverhez tartozó adatbázis szerkezetének felépítésekor pedig figyelembe vettem az alapvető adatbázisokra vonatkozó konzisztencia szabályokat. A megjelenítésért felelő frontend réteg előkészítése során a Google által fejlesztett Angular keretrendszerre esett a választásom. A szervernél ismertetett szempontok itt is fontosak voltak, illetve az Angularral szinte szimbiózisban működő Material frameworkkel pedig a végső kinézetet lehet szabványosra és esztétikusra alakítani.

A mai nagyobb programok esetében erősen ajánlott úgynevezett programtervezési minták alkalmazása és ez itt sincs másként. A programom esetében kívánatos az adathoz (model) és a felhasználói felülethez (view) tartozó dolgok szétválasztása, hogy a felhasználói felület ne befolyásolja az adatkezelést, és az adatok átszervezhetőek legyenek a felhasználói felület változtatása nélkül. Ezt a két réteget különíti el a vezérlő (controller) réteg és ez a tagozódás a Model-View-Controller szerkezeti mintának felel meg a legjobban.

A program általános működése a 37.ábrán látható: a felhasználói inputot és tevékenységeket az Angular komponensek értelmezik és alakítják olyan formátumra, hogy a szerverre való átvitelre alkalmas legyen. Az adatok ezután a megjelenítési réteg service osztályaihoz kerülnek, aminek feladata, hogy megcélozza a szerver egy végpontját és a kapott adatokat elküldje a szervernek (POST), lekérdezze a szervert (GET) vagy töröljön valamit a szerveren (DELETE). Az imént említett végpontokat a szerver controller osztályai határozzák meg és a munkájuk, hogy a kérés típusának megfelelően lekezeljék őket és továbbítsák a service osztályoknak. A service osztályok végzik el a műveletek érdemi részét, ők adják vissza a lekérdezések eredményét például. A legutolsó lépcsőfok pedig az adatbázissal közvetlen kapcsolatban álló repository osztályok, melyeket a service osztályok hívnak meg, ha szükség van az adatbázis lekérdezésére vagy módosítására.

*Controller*

*Repository*

*DBry*

*Service*

*Componentsry*

*Servicery*

*Felhasználóry*

**37. ábra** - A program általános működésének ábrája

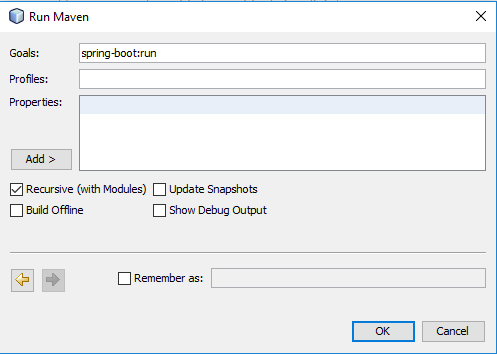
## Telepítés

### 3.2.1 Szerver elindítása

A program kicsomagolásakor két könyvtárt láthatunk:

* *jobs*: a szervert tartalmazza
* *jobfinder-app*: a frontend részt tartalmazza

A szerver elindítása többféleképpen is történhet, attól függően, hogy rendelkezünk-e bizonyos eszközökkel. Ha rendelkezünk valamilyen IDE-vel (Integrated Development Environment), akkor a szerver futtatásához szükséges dependenciákat, függőségeket leíró *pom.xml* fájl segítségével könnyedén elindíthatjuk magát a szervert, miután importáltuk a projektet.



**38. ábra** - Szerver indítása NetBeansben

Az IDE-ben a fájlra jobb gombbal kattintva és a futtatás lehetőséget választva megadhatjuk a futtatás paramétereit, ahol a **spring-boot:run** parancs beírásával el is indul a szerver.

A szerver indítása fejlesztőkörnyezet nélkül is lehetséges, viszont ekkor szükség van, hogy a pom.xml fájl futtatásához szükséges Maven telepítve legyen. A telepítés létezéséről megbizonyosodhatunk, ha megnyitjuk a parancssort és beírjuk a következő parancsot: **mvn -v**, ha hibaüzenetet kapunk, akkor telepítenünk kell a programot a <https://maven.apache.org/download.cgi> oldalról.

Ha megvan a Maven, akkor a szerver gyökérkönyvtárában a parancssor megnyitásával és az **mvn spring-boot:run** paranccsal indul el a szerver.

A harmadik lehetőség az összeállított jar állomány futtatása a parancssorból, ezt a *jobs* könyvtárban kiadott **java -jar target/jobs-0.0.1-SNAPSHOT.jar** paranccsal tehetjük meg (hiba esetén kiegészítve az **--add-modules java.xml.bind** opcióval).

Ha a szerver megfelelően elstartolt, akkor a *„Started JobsApplication in … seconds”* üzenet fog megjelenni a kimeneten.

### 3.2.2 Frontend elindítása

A szerver elindítása után váltsunk át a frontendet tartalmazó *jobfinder-app* könyvtárra és itt is indítsunk el egy parancssori környezetet. A futtatáshoz itt is szükség van egy külső program telepítésére, ez a Node.js illetve az npm, de a Node.js újabb verzióiba integrálva van az npm, ezért elég csak erről az oldalról (<https://nodejs.org/en/download>) a megfelelő telepítőt letölteni.

Ezután a következő három egyszerű parancsot kiadva a gyökérkönyvtárból telepíthető a kliens a parancssorból:



**39. ábra - Angular** telepítése



**40. ábra** - Frontend dependenciák telepítése a **package.json** fájl alapján



**41. ábra** - Az alkalmazás elindítása localhoston

A telepítési folyamat során a *node\_modules* mappába települnek a **package.json** fájlból olvasott modulok, futás közben innen érheti el őket a program, ha ez a mappa már létezik, elég a 3. paranccsal indítani. A sikeres indítás után a böngészőbe a localhost:4200-at beírva jelenik meg az alkalmazás és azonnal használatba is vehető.

## Használt eszközök/ technológiák

Netbeans

A szerver fejlesztésekor az Integrált Fejlesztő Környezetek (angolul IDE-k) közül a NetBeans-re esett a választásom, ez ma az egyik legnépszerűbb IDE, illetve a Java8 hivatalos fejlesztőkörnyezete. Az automata kiegészítésen és az automatikus kódgeneráláson kívül még rengeteg hasznos funkcióval rendelkezik, melyek megkönnyítik a kódolást. Emellett alkalmas HTML, JavaScript és CSS fájlokkal való munkára is, de jelen esetben kizárólag a Java nyelven írt szerver készítésére használtam.

Java Spring Boot

Az alkalmazás szerver része teljes mértékben Java nyelven íródott, a Spring Boot keretrendszer használatával. A Java Spring egy nyílt forráskódú, inversion of controlt megvalósító Java alkalmazás keretrendszer és manapság az egyik legnépszerűbb eszköz a Java alapú webalkalmazások felépítéséhez és futtatásához. Az előbb említett inversion of control (magyarul: a kontroll megfordítása) egy technika a program komponenseinek összeillesztésére, konfigurálásra és kezelésére. A lényege, hogy a komponenskezelést (pl. létrehozás, példányosítás, megszüntetés, metódus hívás) kiemeljük a programkódból és egy külső keretrendszerre bízzuk, ami jelen esetben a Spring, így növelve a modularitást és egyszerűsítve a bővíthetőséget. A Spring Boot a Spring framework még fejlesztőbarátabb változata, célja, hogy minél előbb fusson alkalmazásunk, ezért minimális konfigurációval és könnyen kezelhető függőségekkel segít, hogy pár pillanat alatt már működjön is programunk és a sok felesleges beállítás helyett magára a fejlesztésre koncentrálhassunk.

Maven

Az Apache Maven teljes szoftverprojektek menedzselésére és a build folyamat automatizálására való program. Szerepe a már korábban látott **pom.xml** fájl kapcsán van, ez a **P**roject **O**bject **M**odel fájl írja le a felépítendő projektet, illetve a projekt függőségeit. A Maven végzi az xml fájlban megjelölt komponensek/ beépülő modulok internetről való letöltését is.

Hibernate

A Hibernate egy objektum-relációs leképezést (ORM) megvalósító programkönyvtár Java platformra, mellyel osztályokat tudunk adattáblákká leképezni, illetve az adatbázisban lévő rekordokat objektumokként kezelhetjük. A leképezést a Java kódban elhelyezett annotációk segítségével vezérelhetjük, így a klasszikus POJO-k (Plain Old Java Object) perzisztenciáját könnyen elvégezhetjük.

JPA

A Java Persistence API egy keretrendszer, amivel a relációs adatbázisok kezelhetők Java alkalmazásokban. Erősen összefügg az előbb ismertetett Hibernate-el, ugyanis a JPA a specifikáció, a Hibernate pedig maga az implementáció. Ez a két eszköz kiküszöböli az SQL kód írását, mindössze a JPA szintaxisát kell ismerni és bármilyen lekérdezést vagy tábla létrehozási műveletet automatikusan legenerál nekünk. Hasznos tudni azt is, hogy ha az adatbázis elérési útját megváltoztatjuk például egy PostgreSQL adatbázisról egy MySQL-re, akkor a Hibernate automatikusan érzékeli és úgy állítja össze a lekérdezéseket, hogy kompatibilis legyen.

H2

Az adatbázis motor szerepére a H2-t választottam, mivel nagyon gyors, könnyűsúlyú, nyílt forráskódú és a beépített, böngészőből elérhető konzoljával könnyű ellenőrizni az adatokat. Ez az adatbázis motor fejlesztési, tesztelési és bemutatási célra egyaránt megfelelő, de ha később egy komolyabb megoldásra lenne szükség, akkor az adatbázis könnyen áthelyezhető egy MySQL vagy PostgreSQL rendszerre.

MVC

A teljes alkalmazás megvalósításakor a Model-View-Controller elveket tartottam szem előtt, ez később a projekt szerkezetének ismertetésekor is látszani fog. A view rétegnek megfelelő frontend a model rétegnek megfelelő entitásokkal kizárólag a controller osztályokon át kommunikál, a program ilyen szempontból jól tagolt.

Spring Security

A Spring alkalmazások számára a standard biztonsági keretrendszer az erősen személyre szabható Spring Security. Jelen esetben funkcionalitásának csak egy nagyon kis részét használtam, a felhasználók authentikációjánál volt rá szükség.

Visual Studio Code

Az alkalmazás megjelenítési rétegének fejlesztésekor a VS Code kódszerkesztőt használtam, mely kifejezetten webalkalmazások kódolására lett optimalizálva. Ez a gyors és könnyen használható fejlesztőkörnyezet 100%-ban támogatja a frontenden használt TypeScript, HTML és CSS nyelveket, többek között automatikus kódkiegészítéssel és formázással.

Angular 5

Az Angular egy TypeScript alapú, nyílt forráskódú webalkalmazás keretrendszer, melyet a Google fejleszt. Lehetővé teszi, hogy egyszerű módon, de ugyanakkor bármennyire komplex webalkalmazásokat fejlesszünk bármilyen platformra, legyen az mobil vagy akár asztali környezet.

Az Angular segítségével a programot logikai felépítés mentén teljesen szét lehet tagolni, a többször használatos kódokat ki lehet emelni és mindenhova, ahova kell beinjektálni. Számos ingyenes beépülő modul és komponens érhető el ehhez is, de beépítetten is tartalmaz rengeteg olyan funkciót, ami egy modern weboldalhoz elengedhetetlen.

Angular Material

Az Angular csapat által fejlesztett, közvetlenül Angular alkalmazásokra optimalizált frontend keretrendszer az oldalak közvetlen megjelenéséért felel. Rengeteg előre megírt komponenst tartalmaz kezdve az űrlapokon használatos csúszkáktól, listáktól, gomboktól egészen a táblázatokig. A komponensek kinézetükben abszolút szabványosak, a Google weboldalaihoz hasonló külső megjelenéssel és emellett könnyen paraméterezhetőek és személyre szabhatóak.

## Szerver

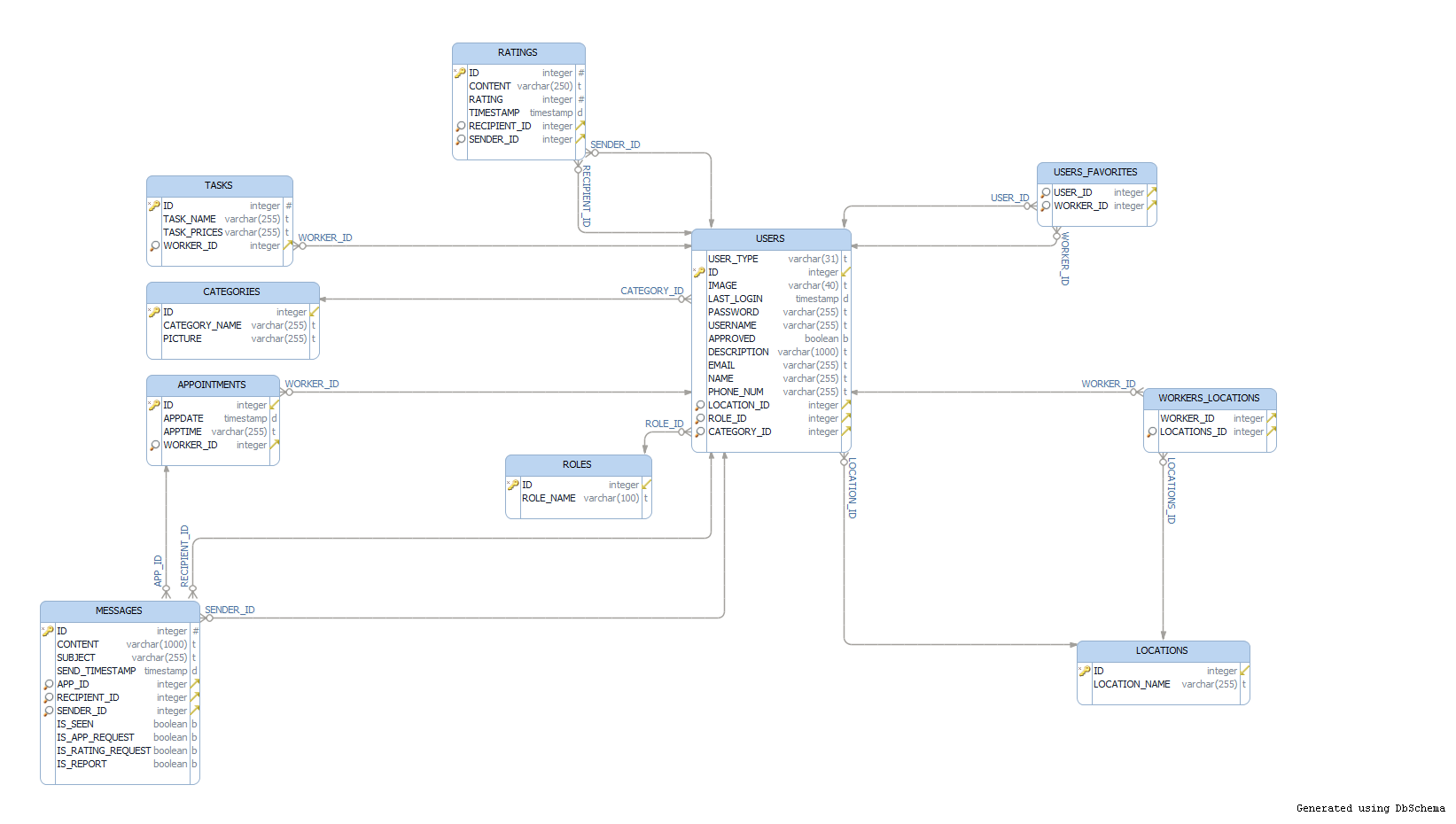
### A projekt felépítése

A főbb csomagok:

* config: az authentikációhoz szükséges security osztályok találhatók itt.
* controller: az entitásokhoz tartozó controller osztályok, melyek a frontenddel tartják a kapcsolatot a kommunikációs végpontokon át.
* entity: az összes entitás, melyek a JPA segítségével az adattábláknak felelnek meg.
* repository: minden entitáshoz tartozik egy repository interface, ami a már létező CRUDrepository interfacet terjeszti ki, melyek a különféle előre megírt Create, Read, Update, Delete műveletekért felelősek az adatbázisban.
* service: a szerviz réteg; a repositoryk és a controllerek közötti kapcsolatot teremti meg, illetve az „üzleti logikát” tartalmazza.

**42. ábra** – A szerver szerkezete

### Az adatbázis szerkezete



**43. ábra** - Az adatbázis szerkezeti diagramja

### Entitások

Az entitások az adatbázistábláknak felelnek meg gyakorlatilag, hiszen a JPA az annotációkkal megtűzdelt osztályokat átalakítja SQL utasításokká és így adatbázis sémák lesznek belőlük. Az entitásokban nincsenek különleges metódusok, viszont a getterek és setterek kötelezőek, hiszen a program ezek alapján térképezi majd fel (mappeli) a frontenddel való kommunikáció során kapott JSON (JavaScript Object Notation) szöveges fájlokat.

Az entitások közötti kapcsolatok között a következők fordulnak elő:

* *@OneToMany*: egy-több kapcsolat, azaz pl. egy szakemberhez több feladat is tartozhat, ennek fordítottja a *@ManyToOne* annotáció
* *@ManyToMany*: több-több kapcsolat, erre a példa, hogy egy szakemberhez több szolgáltatási hely is tartozhat és egy szolgáltatási helyhez több szakember is tartozhat. Az ilyen típusú kapcsolatok egy külön kapcsolati táblával vannak megvalósítva, ami az első táblában lévő azonosítókat rendeli párba a második tábla azonosítóival.

#### BaseEntity

Egy darab long típusú **Id** mezőt tartalmaz és minden másik entitást ebből származtatunk, hogy minden entitás kapjon egy azonosítást könnyítő ID tulajdonságot.

#### BaseUser

Absztrakt osztály, melynek a három szerepkörnek megfelelő „User”, „Worker” és „Admin” osztályok a leszármazottjai, tehát olyan adattagokat tartalmaz, melyek mind a háromra egyaránt érvényesek.

Az egyedi **username** mező a felhasználónevet, a **password** a hozzá tartozó jelszót tartalmazza. A jelszó küldése és tárolása biztonságos módon történik, sose kerül szövegként kiírásra vagy eltárolásra, az adatbázisba MD5 hasheléssel kerül. Az **image** mező a felhasználóhoz tartozó profilkép pontos nevét tárolja el. A **lastLogin** a felhasználók legutolsó bejelentkezésének időpontját tárolja, ez a későbbi karbantartásnál kerül előtérbe. A **senderMessages** és **receiverMessages** adattagok a Messages táblával vannak *@OneToMany* kapcsolatban, azaz minden BaseUser-hez tartozhat több küldött és fogadott üzenet is. Végül a **role** a felhasználó szerep objektumát határozza meg.

#### Admin

Mivel az adminisztrátoroknak nincsenek különösebb tulajdonságai, ezért az eddigi örökölt attribútumokon kívül egy **name**, azaz teljes név mezővel rendelkeznek.

#### User

Az **email, name, phoneNum** szöveges mezők rendre az email, név és telefonszám hármas tárolásáért felelnek, míg a **location** pedig a felhasználók lakhelyének azonosítóját jelöli a Locations táblából. A **ratings** @OneToMany kapcsolatot létesít a Ratings táblával és a felhasználó által írt értékeléseket tárolja. Az első *@ManyToMany* kapcsolat a **favorites** adattagnál figyelhető meg. A kapcsolat a már korábban említett asszociatív táblával van megvalósítva, ami egy user\_id-t rendel egy worker\_id-hez, így egy felhasználó több szakembert kedvelhet és egy szakembert egyszerre többen is kedvelhetnek.

#### Worker

Az **email, name, phoneNum, a ratings** és a kedvencekhez kapcsolódó **userList** adattagok az előbbi User entitással megegyeznek, vagy a kapcsolat másik oldalát alkotják. A **tasks**, **appointments és locations** mezők a szakemberek fontos jellemzőit, a feladatokat, időpontokat és a szolgáltatási helyeket tárolja. A @Transient annotációval ellátott **rating** mező nem kerül eltárolásra az adatbázisban ugyanis azt futás közben a lekérdezés pillanatában számolja ki a program. A **description** a szakember hosszú leírását, a **category** a kategóriáját, az **approved** flag pedig a szakember profiljának engedélyezettségi állapotát mutatja.

#### Message

Az üzenet entitás az alapvető üzenetek tárolásához szükséges adattagokat modellezi: a **content** az üzenet tartalmát, a **subject** az üzenet tárgyát, a **sendTimestamp** pedig a küldés dátumát. A **sender** egy BaseUser objektumnak felel meg, mivel bármelyik implementációja az absztrakt osztálynak képes üzenetet küldeni, ugyanez a helyzet a **receiver** mező esetén is. Az időpontfoglalási rendszerhez szükség volt az üzenetben lévő időpont tárolására is, ezt az **appointment** adattag jelöli. Az **isSeen, isRatingRequest, isAppRequest és isReport** boolean flagek rendre jelölik, hogy látták-e az üzenetet, értékelési kérés-e az üzenet, időpontkérés-e az üzenet vagy jelentés-e az üzenet.

#### Role

Egyszerű entitás, ami a **roleName** mezővel a szerepkör nevét (Admin,User,Worker) és a szerepkörbe tartozó felhasználók listáját (**users**) tárolja.

#### Location

A User entitásokhoz *@OneToMany* kapcsolattal csatlakozik a **users**, a Worker entitásokhoz *@ManyToMany* kapcsolattal a **worker** mezőn keresztül. A **locationName** magától értetődően a hely nevét jelöli.

#### Category

A **categoryName** és a **picture** szöveges mezők a kategória nevét és a kategóriához tartozó kép elérési útját jelölik. A **workers** listával a kategóriához tartozó Workereket perzisztáljuk.

#### Rating

A Message entitásnál megismert módon vannak implementálva a **sender** és **receiver** mezők annyi megkötéssel, hogy itt a küldő csak User, a fogadó pedig csak Worker lehet értelemszerűen. Az értékelés másik két aspektusát jelöli a **content** és a **rating** adattag, a szöveges értékelést, illetve az 1-5 terjedelmű skálán adott egész számú értéket tárolják. A **timestamp** az értékelés dátumát adja meg.

#### Task

A **taskName** és **taskPrices** szöveges attribútumok rendre a feladat elnevezését és a feladat árait/ becsült árát perzisztálják. A **worker** *@ManyToOne* kapcsolatot hoz létre a feladat tulajdonosával.

#### 3.4.3.12 Appointment

A **worker** mező a Task entitásban lévővel teljesen megegyezik, míg az **isFree** flag az időpont foglaltságát mutatja. Az **appDate** az időpont pontos dátumát, az **appTime** pedig a szakember által szövegesen megadott időintervallumot tárolja.

### Kontrollerek

A legfontosabb részei a projektnek a kontroller osztályok, melyek tartják a kapcsolatot a frontenddel a http végpontokon keresztül, deszerializálják a JSON formában kapott adatokat, illetve továbbküldik az adatokat a service réteg felé a további feldolgozásra. Minden entitáshoz tartozik saját kontroller osztály, ezeket fogom ismertetni, néhány jellemző vagy fontosabb végponttal egyetemben.

#### UserController

A legnagyobb terjedelemmel bíró kontroller, mind a három szerepkörű felhasználóhoz kapcsolódó műveleteket végzi, legalapvetőbb funkciói a **login**, **logout**, **registerWorker**, **registerUser**, **registerAdmin**, melyek a be-, kijelentkezés és a regisztráció folyamatait végzik. Egy klasszikus GET végpont bemutatható a keresés funkció végpontjával: kérés érkezik a *„/workers/search/{searchword}”* végpontra, innen kiolvasva a keresőszó paramétert a kontroller a UserServicenek elküldi az adatokat paraméterként, ahol majd megtörténik maga a keresés folyamata. Ezután a megkeresett szakemberek listáját megkapva a kontroller visszaalakítja JSON formátumra és az eredményt visszaküldi a frontendre. A kontroller még sok végpontot tartalmaz, melyek közül sok az alapvető lekérdezés, pl. az összes szakember, user, kategóriánkénti listázás. Az egyetlen felhasználóra vonatkozó műveletek közül néhány példa: módosítás, kedvencekbe elhelyezés, törlés.

#### TaskController

A szakember tulajdonában álló feladatok kezelése a célja, a **create** a létrehozásért felel, miután a paraméterként megkapott Task entitást megkapja, azt lementi az adatbázisba. Az **update** és **delete** metódusok magától értetődően frissítik és letörlik az egyes feladatokat, míg a **getTasksByUser** a felhasználó ID alapján visszaadja az adott user feladatait. Ez a szerveződés a korábban már ismertetett CRUD (create,read,update,delete) funkciókra egy tökéletes példa.

#### AppointmentController

Az előzőhöz hasonló módon ez a kontroller is megvalósítja az alapvető CRUD műveleteket annyi különbséggel, hogy itt a lekérdezés két részre van szedve a foglalt és a szabad időpontok mentén, illetve a **reserve** metódus a szabad időpontot foglaltra állítja.

#### LocationController

A helyek kontrollerét a Worker típusú felhasználók használják főként, ugyanis a User-ek csak egyetlen lakhellyel rendelkeznek és ezért a lekérdezésükkor az egyéb tulajdonságaik mellett a hely is megjelenik, így nincs szükség további lekérdezésekre. A kontroller lehetőséget ad a helyek szakemberenkénti lekérdezésére, illetve a módosítására. A **getLocations** metódus visszaadja az adatbázisban lévő összes helyet, ezt például a regisztrációnál használatos legördülő listákban használja a program.

#### RatingController

A **getAllRatingsByWorker** funkció a profilon való megjelenítéshez lekérdezi az adott szakemberhez tartozó összes értékelést. A **newRating** pedig a frontendről kapott Rating objektum segítségével új értékelést ad hozzá az adatbázishoz.

#### UploadController

A felhasználók által feltöltött profilképek lementését és elhelyezését koordinálja, illetve a későbbiekben való elérésüket teszi lehetővé a **getFile** és **handleFileUpload** metódusokkal.

#### MessageController

Egy darab POST és egy darab DELETE végponttal rendelkezik, a **sendMessage** a frontendtől kapott Message objektumot menti le az adatbázisba, míg a **deleteMessage** pedig a kapott azonosítójú üzenetet törli. A többi GET metódus a következő lekérdezésekre alkalmas: küldött és fogadott üzenetek lekérése, üzenet láttamozása, jelentések lekérése, olvasatlan üzenetek listázása.

### Service és Repository

Repository interfacek és service osztályok a kontrollerekhez hasonlóan majdnem minden entitáshoz léteznek. A repositoryk tartják a kapcsolatot az adatbázissal, ők végzik a lekérdezéseket és az egyéb létrehozó vagy törlő utasításokat.

Mivel a JPA segít nekünk, ezért nem szükséges SQL utasításokkal bajlódni, például a **findPeopleDistinctBylocationNameContainingAllIgnoreCase(String str)** a paraméterként megadott szöveg alapján keres a helyek között és visszaadja azokat, melynek a nevében bárhol előfordul az adott String, az IgnoreCase a kis és nagybetűk közötti különbséget, a Distinct pedig a duplikátumokat küszöböli ki.

A Service osztályok a kontrollerek és a repositoryk közötti kapcsolatot hozzák létre, ez a valóságban a következőképp történik: a kontroller meghívja az entitásnak megfelelő service osztályt, ami elvégzi a megfelelő műveleteket és számításokat, illetve, ha szükség van az adatbázisra, akkor a repositoryn keresztül lekéri a megfelelő adatokat. Ezután az eredmény visszakerül a kontrollerhez, ami azt visszaküldi a frontend felé, megjelenítés céljára. A service osztályok számításai közül néhány példa:

* a rating kiszámolása a lekérdezés pillanatában: ezt az eddig beérkezett értékelések átlagolásából kapjuk meg
* privát adattagok lekérdezése
* a keresés lebonyolítása minden szükséges helyen (név,hely,leírás,email)
* karbantartás során az 1 évnél régebben bejelentkezett felhasználók törlése
* rendszerüzenetek elküldése
* képfeltöltések részletei
* visszaadandó listák szűrése, pl. ne adja vissza a nem engedélyezetteket

## Frontend / megjelenítés

A megjelenítésért a korábban ismertetett Angular keretrendszer felel, az alkalmazás felépítése három nagyobb részre bontható, ez a mappaszerkezetben is ugyanígy van reprezentálva:

* model: a szerver entitásainak modellezése, hogy a szervertől érkező JSON formátumú adatot objektummá tudjuk alakítani itt is, illetve a szerverre való küldésnél vissza is tudjuk alakítani. A TypeScript entitás modellek az adattagokat és egy felparaméterezett konstruktort tartalmaznak.
* service: itt a service osztályok a szerveren lévő kontrollerek tökéletes párjai. Az ezen osztályokban lévő metódusok küldik az információkat a szerver végpontjaira, megcímzik őket és a JSON formában lévő üzeneteket elküldi, majd vár a visszaérkező válaszra.

**44. ábra** - Service metódus példa

A 44.ábrán látható **getWorker (id:number)** metódus egyetlen szakembert kérdez le a szervertől a paraméterként kapott id alapján. Egy http GET kérést küld a szerver *„/user/worker/{id}”* URL-jén elérhető végpontjára és vár a visszaküldött Worker objektumra.

* pages: az Angular alkalmazások komponensekbe rendeződnek, így lehet, hogy az újrahasználható komponensek többször is beágyazhatóak, így átláthatóbbá és kompaktabbá téve a kódot. A pages könyvtárban van az összes ilyen komponens elhelyezve, ezek három részből állnak: egy TypeScript fájl, ami a komponens működését írja le és a service osztályokat hívhatja meg; egy HTML fájl, ami a megjelenítést végzi a lekérdezett adatok alapján; egy CSS fájl, ami a stílusbeállításokat hordozza. Minden komponensnek van egy szelektor kifejezése (pl*. „worker-list”*), amit a HTML kódba írva gyakorlatilag beszúrhatjuk az adott komponenst az oldalra. A következő pontban ezeket részletesen bemutatom.

### Komponensek

#### admin-dashboard

Az admin felület megjelenítéséért felel, a szerverről lekéri a szakembereket, felhasználókat és jelentéseket, ezeket elhelyezi egy, a táblázat számára megjeleníthető adatszerkezetbe, majd ezek kerülnek megjelenítésre. Ezen kívül a táblázat rendezéséhez és kereshetőségéhez kapcsolódó logikát valósítja meg, illetve az új admin hozzáadását végzi.

#### edit-appointments

Az **Időpontok** menüpont megjelenítéséért felelős komponens, a szabad és foglalt időpontok lekérdezése után megjeleníti őket a táblázatban. Az időpontok szerkesztése vagy a hozzáadás gombra kattintással egy felugró ablakot dob fel, melyből összegyűjti a dátum és időintervallum felhasználói inputot és egy időpont objektumot konstruál, melyet tovább küld a szerverre.

#### edit-tasks

Az előző ponttal szinte teljesen megegyezik ez a komponens, ez a **Feladatok** menüpont tartalmát jeleníti meg, miután lekérdezte az adott szakemberhez tartozó feladatok listáját. Itt is van lehetőség szerkesztésre, törlésre és új hozzáadására, ekkor a felbukkanó űrlapok kitöltése után a feladat nevét és árát tárolja el.

#### locations

Kisméretű komponens, az adott szakemberhez tartozó szolgáltatási helyeket szerzi meg és jeleníti meg őket vesszővel elválasztva, ezért több helyen is látható.

#### edit-user-profile

A Profil szerkesztése gombra kattintással a komponens az adott felhasználó profilját egyszerű megjelenítésből egy űrlappá alakítja át, ahol lehetőségünk van szerkeszteni a mezőket. A komponens a jelenlegi értékekkel kitölti az űrlapmezőket, hogy ne kelljen újra beírnunk az információkat. A „Mentés” gombra kattintáskor összeszedi a beírt adatokat, egy User objektumot hoz létre, melyet elküld a szerverre, ahol az mentésre kerül, így módosítva a már létező Usert.

#### edit-worker-profile

Az előzővel közel megegyező komponens, itt egy szakember objektumot építünk fel a módosított adatok alapján, a különbség az, hogy itt a hosszú leírást is lehet módosítani, illetve a helyek közül a többszörös választás lehetősége áll fenn. Az űrlapok az egész alkalmazásban validátorokkal vannak ellátva, például, hogy nem maradhat mező üresen, a telefonszám csak szám lehet vagy a különféle maximum karakter korlátozások.

#### worker-list

A leggyakrabban megjelenő komponens, ha keresünk a főoldalon vagy akár a kategóriánkénti megjelenítést választjuk, akkor a szakemberekből álló lista kerül elő. Itt jól megmutatkozik a komponensek kombinálhatósága, mert a szolgáltatási helyeket a már említett locations komponens beszúrásával jelenítjük meg, míg az átlag értékelést a rating komponenssel. A lista rendezhető is, ezt egyszerű rendező metódusokkal valósítom meg a szakemberek tömbjére.

#### rating

Az átlagos értékelést jeleníti meg a bemeneti paraméter alapján, a komponens a kapott double értéket felfelé kerekíti és kirajzolja a megfelelő számú csillagot.

#### rating-detailed

Az értékeléseket nem csak a rövid formájában kell megjeleníteni, hanem a szakemberek profilján a hosszabb változatában jön elő, ennek a komponensnek ez a feladata. A komponens egyszerre csak a három legújabb értékelést mutatja meg, de ha a beépített *„Még több értékelés”* gombra megyünk akkor a teljes tömböt megjeleníti.

#### worker-profile és user-profile

A részletes szakemberi profilt jeleníti meg a tetején a profilképpel és a személyes adatokkal, alatta a task-table komponens a feladatok táblázatát jeleníti meg, majd az előbb említett rating-detailed komponens zárja le az oldalt. A komponensen látható gombok aktiválását a TypeScript kód ellenőrzi az alapján, hogy be van-e jelentkezve a felhasználó. Mindhárom gomb kattintása esetén felugró párbeszédablakokkal történik a kommunikáció, az üzenetküldés és jelentés gomb esetén lehetőség van egy szöveges mezőbe írni a levél tartalmát. Az időpontfoglalási ablakban két legördülő listából választhatjuk ki a kívánt időpontot és feladatot, majd a szöveges input mezőbe írhatjuk be a megjegyzésünket. Minden esetben a frontend egy üzenetet generál a beírt adatokból a különbség csak annyiban rejlik, hogy különböző flagekkel látja el őket, hogy a szerver észlelje, hogy mi a célja az üzenetnek. A user-profile komponens egy lecsökkentett verziója a szakemberekének, hiszen itt nincs szükség annyi információ megjelenítésére, viszont működésükben és logikájukban megegyeznek.

#### messages

A message.service osztály használatával lekérdezi a fogadott és a küldött üzeneteit az adott usernek, majd megjeleníti listás elrendezésben. Az üzenetre való kattintáskor felugró ablakban megjelenik a teljes üzenet, a küldőre pedig link mutat. Ha nincs üzenet, amit megjelenítsen, akkor erről tájékoztató üzenetet láthatunk.

#### register-worker és register-user

Mindkét típusú felhasználó regisztrációja egy lineáris lépcsős rendszerrel van megvalósítva (stepper), tehát amíg nincsen kitöltve elfogadhatóan egy lépcső, addig nem enged továbblépni a következőre. Az első három lépés mindegyike különálló űrlapmezőkből áll és tartalmazza a megfelelő validátorokat. A negyedik lépésben az eddig megadott adatokat lenullázhatjuk, vagy a regisztráció gombra kattintva a program egy szakember (vagy felhasználó) objektumot küld a szerverre, mentés céljából.

#### login

A bejelentkezésnél előjövő egyszerű, két mezőből álló (felhasználónév és jelszó) űrlap megjelenítéséért felel. Ha a beírt adatokkal nem sikerül a belépés, akkor egy alulról feljövő rövid üzenetfület mutat fel, ez az úgynevezett snackbar és az ilyen jellegű, rövid üzenetek kiírására több helyen is sor kerül a program használata közben.

#### menu

A konstans jelenlévő felső menüt jeleníti meg ez a komponens: az első gomb a kezdőlapra visz akárhonnan, a következő két gomb a regisztrációs komponensekhez, az utolsó pedig a bejelentkezéshez. Ha egy látogató bejelentkezik, akkor lekérdezi a profilképét és a privát menüjének gombján megjeleníti azt, illetve egy felkiáltójel ikont annak függvényében, hogy van-e olvasatlan levele vagy nincs.

#### main

A főoldal komponense, fentről lefelé megjeleníti a következőket: keresési mező a keresőgombbal, kategóriák az ikonjaikkal, a top 5 legnépszerűbb szakember. A keresőt használva a worker-list komponens használatával jelennek meg a találatok, akárcsak a kategóriaválasztót használva. A mennyiségileg legtöbb értékelést kapott öt szakember kerül a legnépszerűbb szakemberek közé, ezeket kártyás elrendezéssel jeleníti meg az alkalmazás és kiválasztásuk esetén a worker-profile komponens aktiválódik.

## Tesztelés

A tesztelést már fejlesztés közben is folyamatosan végeztem, igyekeztem olyan inputokkal is próbálkozni az általánosak mellett, amik rendeltetetésszerű használat mellett nem is érkezhetnének be a szerverhez, így elkerülhetőek az ebből fakadó hibák. A felhasználók által beírt input adatok helyességellenőrzése és típusellenőrzése a frontend részen történik meg, így a szerverhez elküldendő objektumokat már ezen ellenőrzések után, manuálisan állítja össze a program, így biztosan megfelelőek lesznek az objektumok struktúrái és adattagjai. Ha esetleg ez mégsem történne meg, az adatbázis akkor se engedi lementeni a nem megfelelő típusú adatokat, így megőrzi a konzisztenciáját.

A webes felületen route guardot használtam, ami meggátolja, hogy olyan felületre tévedjünk véletlenül vagy szándékosan, ahova nincsen megfelelő jogosultságunk vagy nem hozzánk tartozik. Ha ilyen eset történik akkor vagy egy üres oldal jelenik meg és onnan léphetünk vissza, vagy átirányít a router service egy másik komponensre, ahol például bejelentkezhetünk.

A legnagyobb hangsúlyt a felület funkcióinak tesztelésére helyeztem, a felhasználói élményt, tapasztalatot előtérbe helyezve, ez gyakorlatilag a fekete dobozos tesztelésnek felel meg, hiszen a kód/algoritmus működésének ismerete nélkül, a felhasználó szemszögéből teszteltem. Az eredményeket a következőkben kérdés-válasz formában foglalom össze, a lista kérdés része a felhasználó által végzett tevékenységet, a válasz oszlopa az erre észlelhető válaszreakciót jelenti.

### 3.6.1 Bejelentkezés

* Mi fog történni, ha a felhasználó nem ad meg felhasználónevet/jelszót?
  + A bejelentkezés gomb ki van kapcsolva, amíg a felhasználó nem ad meg felhasználónevet/jelszót, illetve hibaüzenet is jelzi a beviteli mező alatt, hogy kötelező megadni.
* Mi fog történni, ha a felhasználó rossz felhasználónevet/ jelszót ad meg?
  + A szerver visszajelzése után egy alulról felbukkanó rövid üzenet (snackbar) tájékoztatja, hogy helytelen a felhasználónév/ jelszó.
* Mi fog történni, ha a felhasználó helyes adatokkal akar bejelentkezni?
  + A szerver visszajelzése után a rendszer bejelentkezteti és átirányítja a főoldalra.

### 3.6.2 Regisztráció

* Mi fog történni, ha a felhasználó nem tölti ki a szükséges mezőket?
  + A program egészében a csillaggal jelölt mezők kitöltése kötelező, addig a felhasználó nem mehet tovább az adott felületről és ez a regisztrációnál is igaz, ezt piros színnel is jelöli a hiányzó mező.
* Mi fog történni, ha a felhasználó túl hosszú szöveges bemenetet ír be?
  + A program egészében a szöveges mezőknek van egy maximálisan beírható hosszúságuk, ha ezt eléri, akkor egyszerűen nem lehet többet írni. Ha ez a hosszúság valamilyen specifikus szám, akkor ezt rendszerint egy számláló is jelzi a jelenlegi és a maximálisan beírható karakterek számával.
* Mi fog történni, ha a felhasználó nem megfelelő emailt ír be?
  + Az email cím beviteli mezőkön minden esetben email formátum validátor van, tehát hibaüzenettel jelez, ha a beadott szöveg nem email (xx@yy) formátumú és addig nem is enged tovább.
* Mi fog történni, ha a felhasználó nem megfelelő telefonszámot ír be?
  + A telefonszám beviteli mezőkön minden esetben számformátum validátor van, azaz hibaüzenettel jelez, ha a megadott bemenet nem számokból áll és addig nem is enged tovább.
* Mi fog történni, ha a felhasználó nem megfelelő jelszópárost ad meg?
  + Ha a megerősítés során a jelszavak nem egyeznek, akkor ezt piros hibaüzenet jelzi és nem enged továbblépni.
* Mi fog történni, ha már létező felhasználónévvel próbál regisztrálni?
  + A regisztráció véglegesítésekor egy alulról feljövő snackbar tájékoztatja erről a felhasználót.
* Mi fog történni, ha a felhasználó helyes adatokkal regisztrál?
  + Az új felhasználó létrehozásra kerül, snackbar üzenetet kap a felhasználó és átirányítódik a login oldalra.

### 3.6.3 Keresés

* Mi fog történni, ha a felhasználó üres keresést akar végrehajtani?
  + Semmi, hiszen a keresőgomb addig nem aktiválódik, amíg nem írt be minimum három karaktert.
* Mi fog történni, ha a felhasználó hibás adatokkal keres?
  + Üres találati listát kap vissza, ahonnan visszaléphet a kezdőlapra.
* Mi fog történni, ha a felhasználó helyes adatokkal keres?
  + Az adatoknak megfelelő találati listát kapja meg.

### 3.6.4 Egyéb funkciók

* Mi fog történni, ha a felhasználó üres üzenetet vagy jelentést szeretne elküldeni?
  + Ez nem lehetséges, mivel üres tárgy vagy üzenet szöveg esetén a Küldés gomb kikapcsolt állapotban van.
* Mi fog történni, ha a felhasználó ismételten a kedvencek közé helyezés gombra kattint?
  + A szakember kikerül a kedvencei közül és a gomb is visszavált az eredeti piros színére és ezt ismétli akárhányszor rákattintunk.
* Mi fog történni, ha a felhasználó időpontot akar foglalni, de nincsen szabad?
  + A felugró párbeszédablakban kikapcsolt Küldés gomb és egy piros hibaüzenet várja, miszerint *„Nincs szabad időpont jelenleg!”.*
* Mi fog történni, ha a felhasználó nem képet akar feltölteni profilképnek?
  + A tallózás után, ha a kiválasztott fájl **nem** kép formátumú, akkor egy hibaüzenetet kap és maga a feltöltés nem engedélyezett.
* Mi fog történni, ha a felhasználó már létező nevű feladatot vagy létező dátumú időpontot akar hozzáadni?
  + Semmi, a program engedélyezi az ugyanolyan feladatokat vagy időpontokat, a szakember feladata, hogy a profilját karbantartsa és ne használjon duplikátumokat.

## Továbbfejlesztési lehetőségek

Véleményem szerint bármely program esetén a folyamatos fejlesztés a siker kulcsa, hiszen egy alkalmazás szinte soha sincs teljesen készen, mindig van lehetőség új funkciók, ötletek megvalósítására, hibák javítására és megelőzésére, illetve refaktorizációra. Ezért jelen programomnál is elég nagy a továbbfejlesztési lehetőségek halmaza, ezért felsorolok néhány példát a teljesség igénye nélkül, mellyel nagymértékben lehetne növelni a felhasználói élményt vagy az alkalmazás valamely más aspektusát:

* regisztrációkor email megerősítés és esetleges rendszerüzenetek küldése
* képek egységes formázása, mondjuk egy egységes méretű körbevágással, hogy ugyanúgy nézzen ki mindenütt
* adatbázis karbantartásának finomhangolása
* az alkalmazás hosszútávú életére való tekintettel egy nagyobb teherbírású adatbázismotorra való átültetés
* értékelési statisztikák megjelenítése, például: összesen hány értékelés, ebből mennyi öt csillagos, stb.
* az adminisztrátorok új helyeket adhatnak meg a grafikus felületen, illetve ideiglenesen is letilthatnak felhasználókat
* a szakemberek és felhasználók helyeinek valós idejű összehasonlítása, így látszana, hogy elérhetőek-e a felhasználó lakhelyén
* plusz szűrési és keresési feltételek
* az üzenetekben legyen lehetőség képek küldésére is, így illusztrálni is tudnák a felhasználók a problémáikat

# Irodalomjegyzék

* <https://hu.wikipedia.org/wiki/Spring_keretrendszer> [2018.04.17.]
* <https://hu.wikipedia.org/wiki/Apache_Maven> [2018.04.17.]
* <https://hu.wikipedia.org/wiki/Hibernate> [2018.04.17.]
* <https://angular.io/> [2018.04.17.]
* <https://material.angular.io/> [2018.04.17.]
* <https://nodejs.org/en/download/> [2018.04.17.]
* <https://maven.apache.org/download.cgi> [2018.04.17.]
* Az alkalmazásban látható stock képek és ikonok a Creative Commons Zero licensz alá esnek. Ez azt jelenti, hogy szabadon és ingyen használhatóak személyes használatra és akár reklám vagy pénzszerzési célokra is. A képek forrásai a következő oldalak:
* <https://www.pexels.com/> [2018.03.23.]
* <https://pixabay.com/> [2018.03.23.]
* <https://www.flaticon.com/> [2018.03.23.]
* Az ikonok készítőit ajánlott feltüntetni, ezek a következők: Freepik from Flaticon, Nikita Golubev from Flaticon, prettycons from Flaticon.