

Budapesti Műszaki Szakképzési Centrum
Petrik Lajos Két Tanítási Nyelvű Technikum

Szoftverfejlesztő és -tesztelő technikus szakma
5-0613-12-03

VIZSGAREMEK

Önkéntes Portál

ÖNKÉNTES PORTÁL



Önként Önért!

1. Tartalomjegyzék

1.	Tartalomjegyzék.....	1
2.	Nyilatkozat	4
3.	Fejlesztői csapat	4
4.	Bevezetés.....	5
5.	Témaválasztás	5
6.	Fejlesztői dokumentáció.....	6
6.1	Backend oldal - Szerver.....	6
6.1.1	Az alkalmazott fejlesztői eszközök	6
6.1.1.1	Keretrendszer	6
6.1.1.2	Programozási nyelv	6
6.1.1.3	Fejlesztői környezet	6
6.1.1.4	Adatbázis-kezelő rendszer	7
6.1.1.5	GitHub.....	7
6.1.2	Adatmodell leírása.....	7
6.1.3	Részletes feladat-specifikáció, algoritmusok.....	14
6.1.3.1	Modellek	14
6.1.3.2	API kontrollerek.....	17
6.1.4	Tesztelési dokumentáció	21
6.1.5	Továbbfejlesztési lehetőségek.....	29
6.2	Vékonykliens oldal – Webalkalmazás.....	31
6.2.1	Az alkalmazott fejlesztői eszközök	31
6.2.2	Részletes feladat-specifikáció, algoritmusok.....	31
6.2.2.1	Kezdolap	32
6.2.2.2	Profil	34
6.2.2.3	Hirdetes_kereses	38
6.2.2.4	Hirdetes_feladas.....	40
6.2.2.5	Statisztika.....	41
6.2.3	Tesztelési dokumentáció	42
6.2.3.1	Menürendszer és visszajelzések.....	42
6.2.3.2	Hirdetések keresése.....	43
6.2.3.3	Regisztráció.....	44
6.2.3.4	Bejelentkezés	47
6.2.3.5	Hirdetés feladása	47
6.2.3.6	Statisztika.....	48
6.2.3.7	Profil	48

Tartalomjegyzék

6.2.3.8	Profil adatok és jelszó módosítása	49
6.2.3.9	Profil és hirdetés törlése	49
6.2.3.10	Jelentkezés elfogadása és elutasítása	50
6.2.3.11	Webszerver, database.php adathibák	50
6.2.4	Továbbfejlesztési lehetőségek	51
6.3	Vékonykliens oldal - Mobilalkalmazás	53
6.3.1	Az alkalmazott fejlesztői eszközök	53
6.3.1.1	Fejlesztői környezet	53
6.3.1.2	Programozási nyelvek	53
6.3.1.3	Adatbázis-kezelő rendszer	53
6.3.2	Adatmodell leírása	53
6.3.3	Részletes feladat-specifikáció, algoritmusok	55
6.3.3.1	Kínézet-XML fájlok	55
6.3.3.2	Működés-Java fájlok	56
6.3.4	Tesztelési dokumentáció	60
6.3.5	Továbbfejlesztési lehetőségek	64
7.	Felhasználói dokumentáció	66
7.1	Webalkalmazás	66
7.1.1	A program általános specifikációja	66
7.1.2	Rendszerkövetelmények	66
7.1.3	A program telepítésének és konfigurálásának leírása	66
7.1.4	A program használatának részletes leírása	67
7.1.4.1	Regisztráció	67
7.1.4.2	Bejelentkezés	70
7.1.4.3	Hirdetések keresése	71
7.1.4.4	Profil	72
7.1.4.5	Hirdetés feladása	74
7.1.4.6	Ranglista	76
7.2	Mobilalkalmazás	77
7.2.1	A program általános specifikációja	77
7.2.2	Rendszerkövetelmények	77
7.2.3	A program telepítésének és konfigurálásának leírása	78
7.2.4	A program használatának részletes leírása	78
7.2.4.1	Fő oldalak és menüpontok	78
7.2.4.2	Regisztráció	79
7.2.4.3	Bejelentkezés	80
7.2.4.4	Hirdetés feladása	81

Tartalomjegyzék

7.2.4.5	Hirdetések keresése.....	82
7.2.4.6	Profil	84
7.2.4.7	Ranglista	87
8.	Összegzés	89
8.1	Megvalósítás	89
8.2	Nehézségek	89
8.3	Tapasztalatok	89
8.4	Szakmai fejlődés.....	89

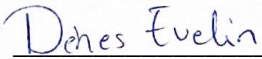
2. Nyilatkozat

Budapesti Műszaki Szakképzési Centrum
Petrik Lajos Két Tanítási Nyelvű Technikum

Szoftverfejlesztő és -tesztelő technikus szakma
5-0613-12-03


Alulírott Dénes Evelin és Gulácsi-Szabó Tímea kijelentem, hogy ez a vizsgaremek saját tudásom, önálló munkám terméke.

A vizsgaremek közös részeit Dénes Evelin és Gulácsi-Szabó Tímea készítette, ezeket pontosan jelöltük.



Aláírás

Dénes Evelin



Aláírás

Gulácsi-Szabó Tímea

Budapest, 2022.

3. Fejlesztői csapat

A fejlesztői csapat a 2020 szeptemberében induló, esti munkarendű 2/14. SL osztály végzős tanulóiból alakult ki és a képzés követelményeinek megfelelően 2 főből áll.

Csapattagok:

- Dénes Evelin – Mobilalkalmazás
- Gulácsi-Szabó Tímea – Webalkalmazás

4. Bevezetés

Az Önkéntes Portál egy internetes böngészőben használható web- és mobilalkalmazás, melynek célja, hogy a segítségre szoruló, valamint a társadalmi felelősségvállalás jegyében önként segíteni vágyó emberek egymásra találhassanak.

Az alkalmazás lehetővé teszi, hogy hitelesített regisztrációt követően a bejelentkezett felhasználó önkéntes munkákról szóló hirdetéseket adhasson fel, illetve a feladott hirdetések között kereshessen. A találati listában megtalálható adott hirdetésre egyszerre csak egy ember tud jelentkezni. Jelentkezést követően a hirdetést feladó személy jóváhagyásra váró hirdetési között kerül kilistázásra.

A felhasználó profiloldalán lehetőség nyílik bizonyos, a regisztrációkor megadott személyes adatok és jelszó módosítására, valamint a felhasználói fiók törlésére is. Ezenkívül minden felhasználó megtekintheti az általa feladott, a jóváhagyásra váró, valamint azon hirdetéseket is, amelyekre jelentkezett.

Bejelentkezést követően a felhasználónak módjában áll törölnie az általa feladott hirdetéseket, valamint a jóváhagyásra váró hirdetési esetén elfogadni vagy elutasítani az önkéntesek jelentkezését.

Az applikáció minden elvégzett önkéntes munka után 1 pontot jóváír az önkéntes számára, egyfajta sorrendet felállítva az összes beregisztrált felhasználó között. A pontszámok alapján kialakult rangsor, valamint a hirdetések kategória szerinti százalékos eloszlása is megtekinthető.

5. Témaválasztás

Napjainkban egyre nagyobb problémát jelent az elmagányosodás, a kiszolgáltatottság és a társadalmi elidegenedés. Fontosnak tartottuk, hogy a fejlesztett applikáció valós problémára nyújtson megoldást, életszerű legyen. A mai világban úgy gondoljuk, hogy fontos erősíteni a társadalmi felelősségvállalást, fejleszteni a szociális érzékenységet és népszerűsíteni az önzetlen segítségnyújtást. Jellemzően önkéntes munkát nagyobb karitatív szervezetek berkein belül lehetséges végezni, magánszemélyek közötti közvetlen kapcsolódásra sokkal kevesebb platform létezik. Témaválasztásunk véglegesítése előtt egyetlen hasonló alkalmazást találtunk az interneten, azonban a felületen elérhető tartalom mennyisége és feltöltésének időpontja alapján az is inaktívnak tűnt.

Olyan applikációt szerettünk volna fejleszteni, melynek van létjogosultsága, társadalmi haszna, és céljaival azonosulni tudunk, azok közel állnak hozzánk. A vizsgaremek elkészítéséhez szükséges fejlesztői csapat kialakítását megelőzően is a képzés minden csapatmunkát előíró feladatában egy csapatban voltunk, így nem volt kérdés számunkra, hogy a közel egy éven át tartó fejlesztés során is egy csapatot szeretnénk alkotni. Megtapsztaltuk, hogy jól tudunk együttműködni, nagy összhangban tudunk dolgozni, és nem utolsósorban közel azonos időbeosztás mellett, amely az esti képzés csapatmunkát igénylő feladataiban kulcsfontosságú szempont.

6. Fejlesztői dokumentáció

6.1 Backend oldal - Szerver

6.1.1 Az alkalmazott fejlesztői eszközök

6.1.1.1 Keretrendszer

A kliens és szerver közötti, az adatok továbbítását és az adatok lekérdezését is magában foglaló adatforgalom a 3.1.13 verziószámú CodeIgniter keretrendszer segítségével valósult meg. A CodeIgniter egy PHP programozási nyelvre épülő MVC (Model-View-Controller) keretrendszer. A vizsgaremek készítése során egy alkalommal - 2022. márciusában - verziófrissítést hajtottunk végre, és a korábban használt 3.1.11 verzióról áttértünk az újabb verzióra. A verziófrissítésre esetünkben azért volt szükség, mert a Session Library – Flashdata metódusa, melyet a felhasználókkal történő kommunikációra használtunk (hibaüzenetek, visszajelzések) PHP 8 verzió használata esetén nem működött megfelelően a régebbi verzióban [<https://www.codeigniter.com/userguide3>].

6.1.1.2 Programozási nyelv

Szerver-oldali programozási nyelvnek a PHP 8.0.10 verzióját választottuk, hiszen a képzés webfejlesztés tanórái során ezzel a nyelvvel ismerkedtünk meg mélyebben. A PHP futtatásához szükség van egy webszerverre, ennek jellemzőit a következő alfejezetben fejtjük ki részletesebben [<https://www.php.net/>].

6.1.1.3 Fejlesztői környezet

A szerver oldali programozás fejlesztői környezeteként a Microsoft által fejlesztett és ingyenesen hozzáférhető Visual Studio Code nevű programot használtuk. A fejlesztés hatékonyabb és gyorsabb kivitelezése érdekében a következő bővítményeket telepítettük:

- ci-snippets2
- PHP Debug

- PHP Extension Pack
- PHP Intelephense
- PHP IntelliSense
- GitHub Pull Requests and Issues

A PHP kódok futtatásához szükséges webszervert az ingyenesen hozzáférhető XAMPP platformfüggetlen webszerver-szoftvercsomag telepítésével biztosítottuk. A XAMPP tartalmazza mindazon alkotóelemeket, melyek szükségesek az alkalmazásunk backend oldali megvalósításához. Az általunk használt elemek a következők:

- Apache webkiszolgáló (webszerver)
- MariaDB (MySQL) adatbázis kiszolgáló
- PHP értelmező [<https://webiskola.hu>]

6.1.1.4 Adatbázis-kezelő rendszer

Az 5.1.1 verziószámú phpMyAdmin egy ingyenes szoftver, mely lehetővé teszi a XAMPP adatbázis kiszolgáló elemeinek - MySQL és MariaDB - kezelését. Segítségével adatbázisokat tudunk készíteni, valamint SQL parancsokat futtatni [<https://www.inf.u-szeged.hu/>].

Az applikációk megvalósítása során nem béreltünk ki webszervert, így saját számítógépünk látta el a webszerver feladatát is, a teljes fejlesztés lokálisan, localhoston és root felhasználónévvel történt. A két alkalmazás szerkezetileg teljesen azonos, de egyes táblák esetében tartalmilag eltérő adatbázist használ.

6.1.1.5 GitHub

A közösen végzett backend fejlesztésben nagy segítséget jelentett a GitHub verziókövető, hiszen azonnal láthattuk és egyúttal integrálhattuk saját lokális tárolónkba a fejlesztés elején létrehozott Git repositoryban található forráskódokban történő változtatásokat. Alapvetően Git alapparancsokat - add, commit, push és pull - alkalmaztunk a munkafolyamatok nyilvántartására.

6.1.2 Adatmodell leírása

Az alkalmazásunk elengedhetetlen és legfontosabb részét a harmadik normálformában lévő adatbázis képezi, melynek részletes jellemzésére a következő oldalakon kerül sor, többnyire táblázatos formában a szemléletesebb bemutatás érdekében.

1. táblázat: Adatbázis alapadatok

Forrás: Saját készítésű táblázat

Adatbázis neve	karitativ
Port	3306
Felhasználónév	root
Jelszó	
Illesztés	utf8mb4_general_ci

2. táblázat: User tábla

Forrás: Saját készítésű táblázat

Mezőnév	Típus	Alapértelmezett érték
user_id	int (11)	
nev	varchar (100)	
felhnev	varchar (100)	
szuldatum	date	
telszam	varchar (30)	
email	varchar (100)	
jelszo	varchar (100)	
profilkep	varchar (50)	
okmanykep	varchar (50)	
okmanyszam	varchar (100)	
hitelesites	enum ('true', 'false')	'false'
pontszam	int (11)	0
telepules_id	int (11)	
cim	varchar (100)	

A user tábla mezőinek jellemzése:

user_id: A regisztráló felhasználó egyedi azonosítója, AUTO_INCREMENT, PRIMARY KEY.

nev: A regisztrált felhasználó teljes neve.

felhnev: A regisztrált felhasználó felhasználóneve, UNIQUE.

szuldatum: A regisztrált felhasználó születési dátuma.

telszam: A regisztrált felhasználó telefonszáma.

email: A regisztrált felhasználó telefonszáma.

jelszo: A regisztrált felhasználó e-mail címe.

profilkep: A regisztrált felhasználó által feltöltött kép neve (profilkep.jpg).

okmanykep: A regisztrált felhasználó által feltöltött kép neve (okmanykep.jpg).

okmanyszam: A regisztrált felhasználó által feltöltött okmány száma.

hitelesites: A regisztrációt követően a feltöltött adatok hitelesítési folyamaton esnek át. Ennek eredménye 'true' vagy 'false' lehet.

pontszam: A regisztrált felhasználó által gyűjtött pontok száma, mely alapértelmezetten 0 és minden elvégzett önkéntes munkát követően eggyel növeljük az értékét. A pontnövelést egy trigger segítségével oldottuk meg, melyről részletesen az adatbázisunk jelentkezes nevű táblájának ismertetésekor fogunk írni.

telepules_id: A regisztrált felhasználó által választott település azonosítója, melyen keresztül a user tábla az adatbázis telepules táblájával kapcsolódik, FOREIGN KEY.

cim: A regisztrált felhasználó címe (irányítószám, utca, házszám, emelet, ajtó).

3. táblázat: Telepules tábla

Forrás: Saját készítésű táblázat

Mezőnév	Típus	Alapértelmezett érték
telepules_id	int (11)	
telepules	varchar (20)	

A telepules tábla mezőinek jellemzése:

telepules_id: A település azonosítója, AUTO_INCREMENT, PRIMARY KEY.

telepules: A település neve.

4. táblázat: Kategoria tábla

Forrás: Saját készítésű táblázat

Mezőnév	Típus
katategoria_id	int (11)
katategoria_nev	varchar (100)
katategoria_kep	varchar (50)

A katategoria tábla mezőinek jellemzése:

katategoria_id: A hirdetés kategóriájának egyedi azonosítója, AUTO_INCREMENT, PRIMARY KEY.

katategoria_nev: A hirdetés kategóriájának neve.

kategoria_kep: A hirdetés kategóriáját szemléltető kép neve (kategorianev.jpg).

5. táblázat: Hirdetes tábla

Forrás: Saját készítésű táblázat

Mezőnév	Típus	Alapértelmezett érték
hirdetes_id	int (11)	
kategoria_id	int (11)	
hirdeto_id	int (11)	
feladas_idopont	timestamp	current_timestamp()
kezdo_idopont	datetime	
zaro_idopont	datetime	
leiras	text	
hirdetes_telszam	varchar (30)	
hirdetes_cim	varchar (100)	
telepules_id	int (11)	

A hirdetes tábla mezőinek jellemzése:

hirdetes_id: A feladott hirdetés azonosítója, AUTO_INCREMENT, PRIMARY KEY.

kategoria_id: A kategória azonosítója, melyen keresztül a hirdetes tábla az adatbázis kategoria táblájával kapcsolódik, FOREIGN KEY.

hirdeto_id: A hirdetést feladó felhasználó azonosítója, melyen keresztül a hirdetes tábla az adatbázis user táblájával kapcsolódik, FOREIGN KEY.

feladas_idopont: A hirdetés feladásának időpontja.

kezdo_idopont: A feladott hirdetés kezdő időpontja.

zaro_idopont: A feladott hirdetés záró időpontja.

leiras: A feladott hirdetés ismertető leírása.

hirdetes_telszam: A hirdetés feladásakor megadott telefonszám.

hirdetes_cim: A hirdetés feladásakor megadott cím, mely a hirdetés helyszínét adja meg.

telepules_id: A regisztrált felhasználó által választott település azonosítója, melyen keresztül a hirdetes tábla az adatbázis telepules táblájával kapcsolódik, FOREIGN KEY.

6. táblázat: Jelentkezés tábla*Forrás: Saját készítésű táblázat*

Mezőnév	Típus	Alapértelmezett érték
jelentkezes_id	int (11)	
jelentkezo_id	int (11)	
idopont	timestamp	current_timestamp()
hirdetes_id	int (11)	
jovahagyas_hirdeto	enum ('true', 'false')	'false'
jovahagyas_onkentes	enum ('true', 'false')	'false'

A jelentkezes tábla mezőinek jellemzése:

jelentkezes_id: A hirdetésekre történő jelentkezés egyedi azonosítója, AUTO_INCREMENT, PRIMARY KEY.

jelentkezo_id: A hirdetésre jelentkezett felhasználó egyedi azonosítója (user_id), melyen keresztül a jelentkezes tábla az adatbázis user táblájával kapcsolódik, FOREIGN KEY.

idopont: A jelentkezés időpontja (időbélyeg).

hirdetes_id: Azon feladott hirdetés egyedi azonosítója, amelyre az adott jelentkezés vonatkozik. A jelentkezes tábla ezen keresztül kapcsolódik az adatbázis hirdetes táblájával, FOREIGN KEY, UNIQUE.

jovahagyas_hirdeto: A hirdető által feladott hirdetésre történő jelentkezés elfogadását vagy elutasítását jelzi. A jelentkezés elfogadása esetén értéke 'true', egyébként alapértelmezetten, a jelentkezés pillanatában 'false'.

jovahagyas_onkentes: Az adott hirdetésre történő jelentkezés esetén 'true' értéket vesz fel.

7. táblázat: Megszorítások*Forrás: Saját készítésű táblázat*

Táblanév	Mezőnév	Megszorítás
user	telepules_id	ON DELETE CASCADE
hirdetes	hirdeto_id	ON DELETE CASCADE
jelentkezes	hirdetes_id	ON DELETE CASCADE
jelentkezes	jelentkezo_id	ON DELETE CASCADE

1. ábra: Trigger

Forrás: Saját készítésű eljárás a XAMPP – phpMyAdmin programban

Módosítás

Részletek

Eseményindító neve

pontszamfrissites

Tábla

jelentkezes

Idő

AFTER

Esemény

UPDATE

Meghatározás

```

1 IF old.jovahagyas_hirdeto !=
  new.jovahagyas_hirdeto THEN
2 SET @pont = (SELECT pontszam FROM user WHERE
  user_id = old.jelentkezo_id) + 1;
3 UPDATE user SET pontszam = @pont
4 WHERE user_id = old.jelentkezo_id;
5 END IF

```

Meghatározó

root@localhost

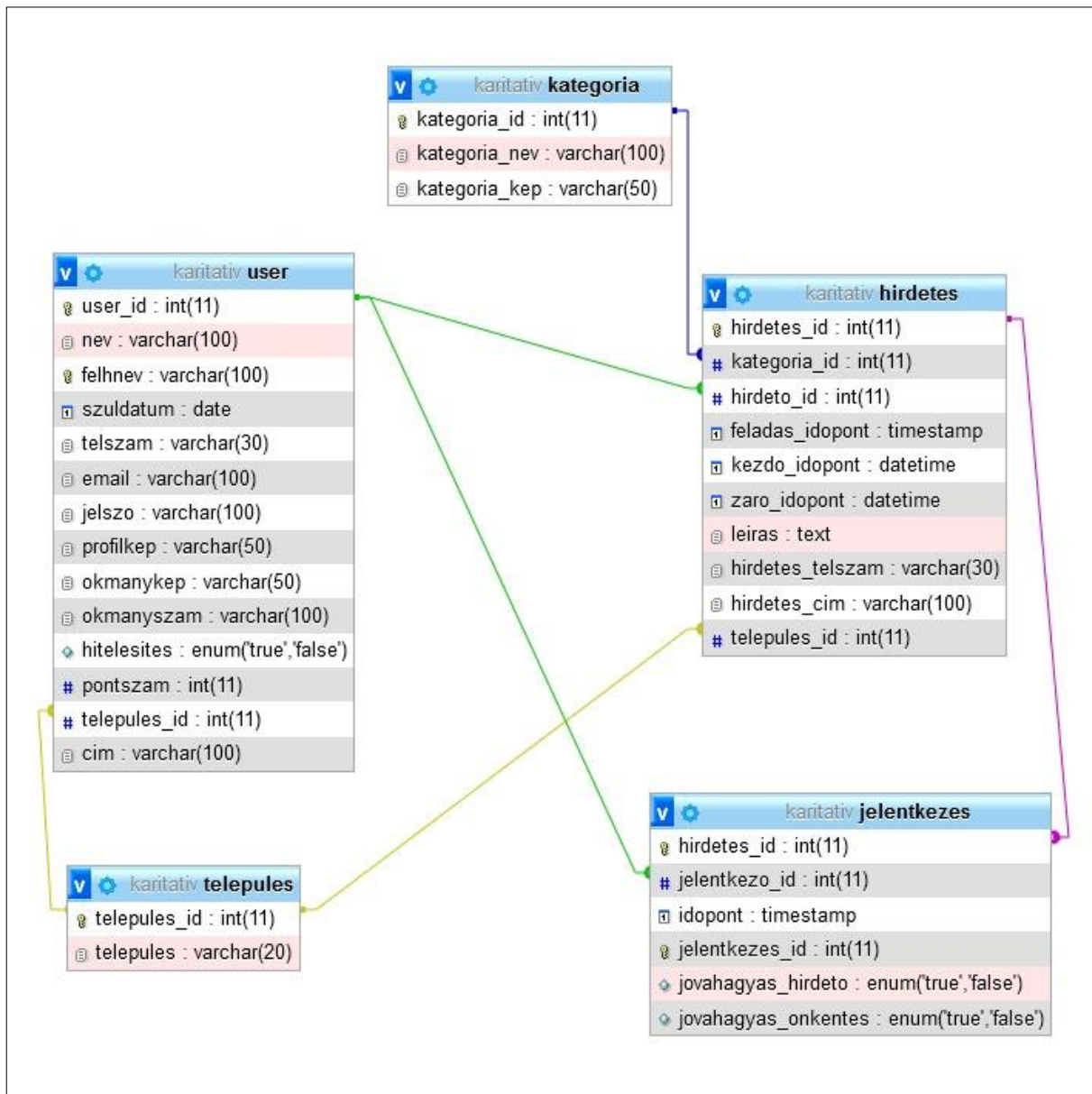
Indítás

Bezárás

Az ábrán jelölt pontszamfrissites elnevezésű trigger a jelentkezes tábla jovahagyas_hirdeto mezőjében tárolt érték változását ('false' → 'true') követően a user táblában az adott hirdetésre jelentkező személynél a pontszám mezőben tárolt értéket megnöveli eggyel.

2. ábra: Tervező nézet

Forrás: Tervező nézet a XAMPP – phpMyAdmin programban



A fenti ábra az adatbázis táblái közötti kapcsolatokat szemlélteti oly módon, hogy az egyes táblákban található idegenkulcsok hogyan kapcsolódnak más táblák elsődleges kulcsához.

6.1.3 Részletes feladatspecifikáció, algoritmusok

Az alábbi fejezetekben a CodeIgniter MVC keretrendszerben megvalósított osztályokat és azok metódusait részletezzük. Az MVC (Model-View-Controller) rendszerben a modellek teremtik meg a kapcsolatot az adatbázissal, míg a controllerek vezérlőszerepet töltenek be. A view felelős az alkalmazás kinézetéért, így az nem képezi a backend fejlesztés részét.

6.1.3.1 Modellek

Minden model a CI_Model ősosztály leszármazottja, így a konstruktorban minden esetben meghívjuk a szülőosztály konstruktorát, valamint betöltjük a Database Class-t.

```
public function __construct(){
    parent::__construct();
    $this->load->database();
}
```

A modellekben található metódusokban a Query Builder Class segítségével hoztunk létre SQL lekérdezéseket.

1.) User_model

A felhasználó regisztrációjához, bejelentkezéséhez, személyes adatok megjelenítéséhez, a személyes adatok és jelszó módosításához, valamint a profil törléséhez szükséges metódusokat tartalmazza.

```
public function get_all(){
    $this->db->order_by('user_id', 'desc');
    return $this->db->get('user')->result_array();
}
```

A fenti paraméter nélküli get_all() metódus az adatbázis user táblájában szereplő összes felhasználó adatát adja vissza. A metódus a user_id szerinti csökkenő sorrendben adja vissza az összes felhasználó adatait tartalmazó többelemű tömböt.

```
public function get_by_id($id){
    $this->db->where('hirdetes_id', $id);
    return $this->db->get('hirdetes')->row_array();
}
```

A fenti paraméteres get_by_id(\$id) metódus az adatbázis user táblájában szereplő egy felhasználó adatát adja vissza a paraméterként megadott \$id (user_id) alapján egy egyelemű tömbben.

8. táblázat: User_model metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Funkció
user_rogzitese()	\$data	regisztráció
user_bejelentkezes()	\$felhnev, \$jelszo	bejelentkezés
user_modositasa()	\$id, \$data	módosítás
user_torlese()	\$id	törlés
kategoriaKepekListazasa()	\$user_id	díjak megjelenítése
userJelszoModositasa()	\$jelszo, \$user_id	jelszó módosítás

2.) Telepules_model

A felhasználó regisztrációjához és hirdetésfeladáshoz szükséges űrlapok település mezőjéhez, hirdetés kereséséhez, valamint a települések megjelenítéséhez (profil, hirdetés) kapcsolódó metódusokat tartalmazza.

9. táblázat: Telepules_model metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Funkció
get_all()		településlista megjelenítése
get_by_id()	\$id	egy település megjelenítése
telepulesekSzama()		települések darabszámának megjelenítése

3.) Kategoria_model

A felhasználó díjainak megjelenítéséhez, a hirdetésfeladáshoz szükséges űrlap kategória mezőjéhez, hirdetés kereséshez, valamint a hirdetések megjelenítéséhez kapcsolódó metódusokat tartalmazza.

10. táblázat: Kategoria_model metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Funkció
get_all()		kategórialista megjelenítése
get_by_id()	\$id	egy kategória megjelenítése
kategoriakSzama()		kategóriák darabszámának megjelenítése

4.) Hirdetes_model

A felhasználók hirdetéseinek megjelenítéséhez, törléséhez, hirdetések kereséséhez kapcsolódó metódusokat tartalmazza.

11. táblázat: Hirdetes_model metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Funkció
get_all()		hirdetéslista megjelenítése
get_by_id()	\$id	egy hirdetés megjelenítése
hirdetes_rogzitese()	\$data	hirdetés feladása
hirdetes_torlese()	\$id	hirdetés törlése
hirdetesKereses	\$telepules_id, \$kategoria_id, \$kezdo_idopont, \$user_id	hirdetések keresése
kategoriaKep()	\$hirdetes_id	hirdetés kategóriaképének megjelenítése
nevProfilkep()	\$hirdetes_id	feladó profilképének megjelenítése
telepulesnev()	\$hirdetes_id	hirdetés településnévének megjelenítése
kategorianev()	\$hirdetes_id	hirdetés kategórianevének megjelenítése
hirdetesekSzama()		hirdetések darabszámának megjelenítése
sajatHirdetesekAdatai()	\$user_id	saját hirdetések adatainak megjelenítése
jovahagyasravarohirdetesekAdatai()	\$user_id	jóváhagyásra váró hirdetések adatainak megjelenítése

5.) Jelentkezes_model

A hirdetésre történő jelentkezéshez, valamint a hirdetés jóváhagyásához és elutasításához kapcsolódó metódusokat tartalmazza.

12. táblázat: Jelentkezes_model metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Funkció
jelentkezes_rogzitese()	\$data	hirdetésre jelentkezés
jelentkezesJovahagysa()	\$hirdetes_id	hirdetés jóváhagyása és teljesítése
jelentkezesTorleseElutasitasMiatt()	\$hirdetes_id	jelentkezés elutasítása és törlése a jelentkezés táblából
jelentkezesTablaHirdetesIdk()		jelentkezés táblából kigyűjti a hirdetésazonosítókat

6.) Ranglista_model

A felhasználók pontszámai alapján kialakuló ranglista és a feladott hirdetésekre vonatkozó statisztika megjelenítéséhez kapcsolódó metódusokat tartalmazza.

13. táblázat: Ranglista_model metódusai

Forrás: Saját készítésű táblázat

Metódusnév	Paraméter	Funkció
kategoriankentiHirdetes()		kategóriánkénti hirdetések darabszáma
rangLista()		az első öt legnagyobb pontszámmal rendelkező felhasználó profilképének megjelenítése

6.1.3.2 API controllerek

Minden API controller a REST_Controller őssztály leszármazottja, így a konstruktorban minden esetben meghívjuk a szülőosztály konstruktorát, valamint betöltjük a felhasználni kívánt modelt.

```
public function __construct() {
    parent::__construct();
    $this->load->model('user_model');
}
```

A fejlesztés során a CRUD műveleteket valósítjuk meg az API metódusok segítségével. A mozaikszó adattárolási és -manipulációs műveletekre vonatkozik:

Create (POST): Új adat létrehozása

Read (GET): Adat lekérése

Update (PUT): Adat módosítása

Delete (DELETE): Adat törlése

A webszerver és a böngésző kommunikációja során minden kérésre érkezik válasz, ún. HTTP státuszkódok formájában. Ezek mindegyike háromjegyű szám, és az első számjegytől függően különböző jelentéssel bírnak.

1xx: tájékoztató jellegű információk

2xx: sikeres kérés

3xx: átirányítás

4xx: klienshiba

5xx: szerverhiba

Alkalmazásunk esetében a leggyakrabban előforduló státuszkódok a következők:

- 200 OK, 201 Created
- 401 Unauthorized, 404 Not Found
- 500 Internal Server Error [<https://en.wikipedia.org>]

1.) User

A felhasználó adatainak lekérdezéséhez és módosításához, regisztrációhoz, bejelentkezéshez és a felhasználó törléséhez kapcsolódó metódusokat tartalmazza.

```
public function index_get($id = 0){
    $adatok = [];
    $error = false;
    $message = "Felhasználó sikeresen lekérdezve.";
    $response_code = REST_Controller::HTTP_OK;
    if ($id == 0) {
        $adatok = $this->user_model->get_all();
    } else{
        $adatok = $this->user_model->get_by_id($id);
        if (is_null($adatok)) {
            $error = true;
            $message = "A megadott azonosítóval nem található felhasználó.";
            $adatok = [];
            $response_code = REST_Controller::HTTP_NOT_FOUND;
        } else{
            $adatok = [$adatok];
        }
    }
    $data = [
        'adatok' => $adatok,
        'error' => $error,
        'message' => $message,
    ];
    $this->response($data, $response_code);
}
```

A fenti `index_get()` metódus felelős egy vagy az összes felhasználó adatainak lekérdezéséért paramétertől függően (`$id`). Ha az `$id` paraméter nincs megadva, akkor a

user_model->get_all(), ellenkező esetben pedig a user_model->get_by_id() metódusa hívódik meg, mely lekérdezések eredményét az \$adatok tömbbe menti. Amennyiben az \$adatok változó értéke nem NULL, visszaadunk egy \$adatok tömböt a lekérdezett adatokkal. Ezután egy \$data nevű, 3 elemű asszociatív tömbnek adjuk át a következőket: a lekérdezett adatokat, error és a message értékét. Mindezt átadjuk a response metódusnak első paramétereként, illetve a \$response_code változót második paraméterként.

14. táblázat: User API metódusai

Forrás: Saját készítésű táblázat

Metódusnév	Paraméter	Kérés típusa	Funkció
index_post()		POST	regisztráció
bejelentkezés_post()		POST	bejelentkezés
index_put()	\$id	PUT	felhasználó adatainak módosítása
index_delete()	\$id	DELETE	felhasználó törlése

2.) Hirdetés

A hirdetés adatainak lekérdezéséhez, a hirdetés feladásához, valamint törléséhez kapcsolódó metódusokat tartalmazza.

```
public function index_post(){
    $adatok['kategoria_id'] = $this->post('kategoria_id');
    $adatok['hirdeto_id'] = $this->post('hirdeto_id');
    $adatok['kezdo_idopont'] = $this->post('kezdo_idopont');
    $adatok['zaro_idopont'] = $this->post('zaro_idopont');
    $adatok['leiras'] = $this->post('leiras');
    $adatok['hirdetes_telszam'] = $this->post('hirdetes_telszam');
    $adatok['telepules_id'] = $this->post('telepules_id');
    $adatok['hirdetes_cim'] = $this->post('hirdetes_cim');
    $id = $this->hirdetes_model->hirdetes_rogzitese($adatok);
    $this->response($adatok, REST_Controller::HTTP_CREATED);
}
```

A fenti index_post() metódus felelős egy új hirdetés feladásáért. Az \$adatok nevű asszociatív tömbnek átadjuk a feltölteni kívánt hirdetés adatait, mely megfeleltethető a hirdetes_model->hirdetes_rogzitese() metódus paraméterének. Ezt követően a response() metódus első paramétere az \$adatok tömb, második paramétere a REST_Controller absztrakt osztály HTTP_CREATED konstans változója (201).

15. táblázat: Hirdetes API metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Kérés típusa	Funkció
index_get()		GET	hirdetéslista megjelenítése
index_get()	\$id	GET	egy hirdetés megjelenítése
index_delete()	\$id	DELETE	hirdetés törlése

3.) Település

A települések adatainak, valamint a települések darabszámának lekérdezéséhez kapcsolódó metódusokat tartalmazza.

16. táblázat: Telepules API metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Kérés típusa	Funkció
index_get()		GET	településlista megjelenítése
index_get()	\$id	GET	egy település megjelenítése
telepulesekSzama_get()		GET	települések darabszámának megjelenítése

4.) Kategória

A kategóriák adatainak, valamint a kategóriák darabszámának lekérdezéséhez kapcsolódó metódusokat tartalmazza.

17. táblázat: Kategoria API metódusai*Forrás: Saját készítésű táblázat*

Metódusnév	Paraméter	Kérés típusa	Funkció
index_get()		GET	kategórialista megjelenítése
index_get()	\$id	GET	egy kategória megjelenítése
kategoriakSzama_get()		GET	kategóriák darabszámának megjelenítése

6.1.4 Tesztelési dokumentáció

A backend fejlesztéshez a már korábban említett CodeIgniter MVC keretrendszert használtuk. A modellekben elkészített, lekérdezésekért felelős metódusokat és az API controllerekben elkészített végpontokat az ingyenesen elérhető, Postman nevű API platform segítségével teszteltük.

1.) Összes település lekérdezése: **api/telepules**

Elvárt eredmény: Összes telepules_id és telepules megjelenik

Státusz kód: 200 OK

Kapott eredmény:

```
{
  "adatok": [
    {
      "telepules_id": "3180",
      "telepules": "Budapest 23."
    }
  ],
  "error": false,
  "message": "Település sikeresen lekérdezve."
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

2.) Egy település lekérdezése: **api/telepules/{telepules_id}**

Elvárt eredmény: Egy darab telepules_id és telepules megjelenik

Státusz kód: 200 OK

Kapott eredmény:

```
{
  "adatok": [
    {
      "telepules_id": "1",
      "telepules": "Szentendre"
    }
  ],
  "error": false,
  "message": "Település sikeresen lekérdezve."
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

3.) Települések számának lekérdezése: **api/telepules/telepulesekSzama**

Elvárt eredmény: Települések darabszáma

Státusz kód: 200 OK

Kapott eredmény:

```
{  
  "adatok": 3180  
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

4.) Összes kategória lekérdezése: **api/kategoria**

Elvárt eredmény: Összes `kategoria_id`, `kategoria_nev` és `kategoria_kep` megjelenik

Státusz kód: 200 OK

Kapott eredmény:

```
{  
  "adatok": [  
    {  
      "kategoria_id": "12",  
      "kategoria_nev": "Egyéb",  
      "kategoria_kep": "egyeb.jpg"  
    }  
  ],  
  "error": false,  
  "message": "Kategória sikeresen lekérdezve."  
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

5.) Egy kategória lekérdezése: **api/kategoria/{kategoria_id}**

Elvárt eredmény: Egy darab `kategoria_id`, `kategoria_nev` és `kategoria_kep` megjelenik

Státusz kód: 200 OK

Kapott eredmény:

```
{
  "adatok": [
    {
      "kategoria_id": "1",
      "kategoria_nev": "Kutyasétáltatás",
      "kategoria_kep": "kutyasetaltatas.jpg"
    }
  ],
  "error": false,
  "message": "Kategória sikeresen lekérdezve."
}
```

Státuszkód: 200 OK

Lehetséges hiba: 404 Not Found

6.) Kategóriák számának lekérdezése: **api/kategoria/kategoriakSzama**

Elvárt eredmény: Kategóriák darabszáma

Státuszkód: 200 OK

Kapott eredmény:

```
{
  "adatok": 12
}
```

Státuszkód: 200 OK

Lehetséges hiba: 404 Not Found

7.) Összes hirdetés lekérdezése: **api/hirdetes**

Elvárt eredmény: Összes hirdetes_id, kategoria_id, hirdeto_id, feladas_idopont, kezdo_idopont, zaro_idopont, leiras, hirdetes_telszam, hirdetes_cim, telepules_id megjelenik

Státuszkód: 200 OK

Kapott eredmény:


```
{
  "adatok": [
    {
      "hirdetes_id": "2",
      "kategoria_id": "2",
      "hirdeto_id": "1",
      "feladas_idopont": "2022-07-25 16:46:24",
      "kezdo_idopont": "2022-07-25 17:00:00",
      "zaro_idopont": "2022-07-29 19:00:00",
      "leiras": "Néhány tételre bevásárlásra lenne szükségem a következő 4 nap  
on belül valamikor a törökbálinti Auchan áruházból.",
      "hirdetes_telszam": "06 30 678 4567",
      "hirdetes_cim": "1345 Nagy utca 23.",
      "telepules_id": "19"
    }
  ],
  "error": false,
  "message": "Összes hirdetés sikeresen lekérdezve."
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

8.) Egy hirdetés lekérdezése: **api/hirdetes/{hirdetes_id}**

Elvárt eredmény: Egy darab hirdetes_id, kategoria_id, hirdeto_id, feladas_idopont, kezdo_idopont, zaro_idopont, leiras, hirdetes_telszam, hirdetes_cim, telepules_id megjelenik

Státusz kód: 200 OK

Kapott eredmény:

```
{
  "adatok": [
    {
      "hirdetes_id": "41",
      "kategoria_id": "7",
      "hirdeto_id": "1",
      "feladas_idopont": "2022-09-03 16:46:24",
      "kezdo_idopont": "2022-09-04 17:00:00",
      "zaro_idopont": "2022-09-04 19:00:00",
      "leiras": "Elromlott a mosógépünk és keresek valakit, aki meg tudná  
javítani. Sajnos nincs most lehetőségünk több tízezer Ft-ot kifizetni  
javításra.",
      "hirdetes_telszam": "06 30 123 4567",
      "hirdetes_cim": "1124 Teszt utca 5.",
      "telepules_id": "1"
    }
  ],
  "error": false,
  "message": "Összes hirdetés sikeresen lekérdezve."
}
```

```
{
  "error": false,
  "message": "Összes hirdetés sikeresen lekérdezve."
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

9.) Egy hirdetés létrehozása: **api/hirdetes**

Elvárt eredmény: A beírt adatok alapján létrejön egy új hirdetés az adatbázis hirdetes nevű táblájában.

Státusz kód: 201 Created

Kapott eredmény:

```
{
  "kategoria_id": "1",
  "hirdeto_id": "1",
  "kezdido_pont": "2022-09-19 15:00",
  "zaroido_pont": "2022-09-19 16:00",
  "leiras": "Kiskutyám sétáltatására szeretnék segítséget kérni. Nagyon barátságos.",
  "hirdetes_telszam": "+36301234567",
  "telepules_id": "1",
  "hirdetes_cim": "1111 Fenyő utca 8."
}
```

Státusz kód: 201 Created

Lehetséges hiba: 500 Internal Server Error

10.) Egy hirdetés törlése: **api/hirdetes/{hirdetes_id}**

Elvárt eredmény: A megadott hirdetes_id alapján az adott hirdetés törlése.

Státusz kód: 200 OK

Kapott eredmény:

```
{
  "message": "Hirdetés sikeresen törölve: 5",
  "success": true
}
```

Státusz kód: 200 OK

Lehetséges hiba: 404 Not Found

11.) Összes user lekérdezése: **api/user**

Elvárt eredmény: Összes user_id, nev, felhnev, szuldatum, telszam, email, jelszo, profilkep, okmanykep, okmanyszam, hitelesites, pontszam, telepules_id, cim megjelenik.

Státuszkód: 200 OK

Kapott eredmény:

```
{
  "adatok": [
    {
      "user_id": "1",
      "nev": "Teszt Elek",
      "felhnev": "tesztelek",
      "szuldatum": "2000-10-01",
      "telszam": "0620341234789",
      "email": "tesztelek@email.hu",
      "jelszo": "$2y$10$8q87W5Acsz8PwzvcLy6xLeHPHK06awV21xQ2DKyA9NU6ozW2G2FDS",
      "profilkep": "tesztelek_profil_2022_03_02_13_30_00.jpg",
      "okmanykep": "tesztelek_okmany_2022_03_02_13_30_00.jpg",
      "okmanyszam": "892378DA",
      "hitelesites": "true",
      "pontszam": "9",
      "telepules_id": "1",
      "cim": "1036 Teszt utca 5."
    }
  ],
  "error": false,
  "message": "Felhasználó sikeresen lekérdezve."
}
```

Státuszkód: 200 OK

Lehetséges hiba: 404 Not Found

12.) Egy user lekérdezése: **api/user/{user_id}**

Elvárt eredmény: Egy user_id, nev, felhnev, szuldatum, telszam, email, jelszo, profilkep, okmanykep, okmanyszam, hitelesites, pontszam, telepules_id, cim megjelenik.

Státuszkód: 200 OK

Kapott eredmény:

```
{
  "adatok": [
    {
      "user_id": "5",
      "nev": "Kis Béla",
      "felhnev": "kisbela",
      "szuldatum": "2000-10-06",
      "telszam": "06302467893",
      "email": "kisbela@kisbela.hu",
      "jelszo": "$2y$10$NufSdxs0ft8Kdmuk9zZOLO1QbcUMPi7S4dka.r7Y06N
v6CDhcNjje",
      "profilkep": "kisbela_profil_2022_08_04_18_05_11.jpg",
      "okmanykep": "kisbela_okmany_2022_08_04_18_05_10.jpg",
      "okmanyszam": "673459KE",
      "hitelesites": "true",
      "pontszam": "3",
      "telepules_id": "32",
      "cim": "1025 Béla utca 10."
    }
  ],
  "error": false,
  "message": "Felhasználó sikeresen lekérdezve."
}
```

Státuszkód: 200 OK

Lehetséges hiba: 404 Not Found

13.) Egy user létrehozása (regisztráció): **api/user**

Elvárt eredmény: A beírt adatok alapján létrejön egy új user az adatbázis user nevű táblájában.

Státuszkód: 201 Created

Kapott eredmény:

```
{
  "nev": "Teszt Anna",
  "felhnev": "teszt.anna",
  "jelszo": "$2y$10$jx5dXlh8KY8SN1ZmJbv5vuX4KtGO84QC.AXFn8WTtPqwb83hufg
Fy",
  "cim": "Valamilyen utca 12/fszt. 4.",
  "szuldatum": "2000-11-01",
  "telszam": "06301234568",
  "email": "anna@gmail.com",
  "profilkep": "anna_profil.jpg",
  "okmanykep": "anna_okmany.jpg",
  "okmanyszam": "156345HJ",
  "telepules_id": "12"
}
```

Státuszkód: 201 Created

Lehetséges hiba: 500 Internal Server Error

14.) Egy user bejelentkezése: **api/user/bejelentkezés**

Elvárt eredmény: A beírt adatok alapján a korábban már regisztrált, user táblában szereplő felhasználó bejelentkezik.

Státuszkód: 200 OK

Kapott eredmény: 1 (user_id)

Státuszkód: 200 OK

Lehetséges hiba: 401 Unauthorized

15.) Egy user adatainak módosítása: **api/user/{user_id}**

Elvárt eredmény: A beírt adatok alapján a korábban már regisztrált, user táblában szereplő felhasználó megadott adatai megváltoznak.

Státuszkód: 200 OK

Kapott eredmény:

```
{
  "user_id": "18",
  "nev": "Teszt Anna",
  "felhnev": "teszt.anna",
  "szuldatum": "2000-11-01",
  "telszam": "0630111111",
  "email": "annauj@gmail.com",
  "jelszo": "$2y$10$jx5dXlh8KY8SNlZmJbv5vuX4KtGO84QC.AXFn8WTtPqwb83hufg
Fy",
  "profilkep": "anna_profil.jpg",
  "okmanykep": "anna_okmany.jpg",
  "okmanyszam": "156345HJ",
  "hitelesites": "false",
  "pontszam": "0",
  "telepules_id": "500",
  "cim": "Put utca 12/fszt. 4."
}
```

Státuszkód: 200 OK

Lehetséges hiba: 404 Not Found

16.) Egy user törlése: **api/user/{user_id}**

Elvárt eredmény: A megadott user_id alapján az adott user törlése.

Státuszkód: 200 OK

Kapott eredmény:

```
{
  "message": "Felhasználó sikeresen törölve: 18",
  "success": true
}
```

Státuszkód: 200 OK

Lehetséges hiba: 404 Not Found

6.1.5 Továbbfejlesztési lehetőségek

- A web- és mobilapplikáció valós idejű kommunikációja érdekében az adatbázis közös webserveren való elhelyezése. Ezzel a megoldással az adatok lekérdezése egyidőben és ugyanazon adatbázisból történne, mely biztosítaná, hogy például a mobilalkalmazásban feladott hirdetés a webes felületen is megjelenik.
- A felhasználók elől rejtett, az oldalt üzemeltető szervezet adminisztrációs és hitelesítő feladatait segítő oldal, ún. adminisztrációs felület létrehozása. Esetünkben a jelenleg manuálisan végzett hitelesítés hatékonyabb elvégzésére, személyes adatok ellenőrzésére és módosítására.
- A feladott hirdetésben szereplő adatok módosítása, az adott hirdetés ismételt feladása, illetve időzített feladási lehetőség.
- A regisztrált felhasználó további adatainak módosítása, így például név, okmányszám, profilkép és okmánykép.
- Az alkalmazást használó személyek közötti kapcsolatfelvételi és kommentelési lehetőség biztosítása, értékelő rendszer bevezetése.
- A jelenleg megvalósított, hirdetések keresését végző metódus átalakítása, a szűrési feltételek opcionális használata, azaz a felhasználó a feltételek bármely kombinációjával indíthat keresést. Jelenleg az összes feltétel megadása kötelező és köztük AND kapcsolat van. Szűrési feltételek bővítése, így például időtartam szerinti keresés megvalósítása.
- E-mailes és/vagy sms-s értesítés küldése a felhasználók részére. Jelen alkalmazás esetén ez hirdetésre történő jelentkezés, illetve jóváhagyás és/vagy elutasítás alkalmával történhetne.

- Az önkéntes munkát feladó felhasználó mellett az adott munkára jelentkező önkéntes is visszavonhassa jelentkezését.
- Regisztrációt követően a felhasználó a hitelesítés sikeréről kapjon értesítést, vagy ha probléma lép fel, újra tudjon regisztrálni ugyanazon felhasználónévvel. (Admin felület erre is alkalmas lehet.)
- Backend hibaüzenetek felhasználóbarát megjelenítése.
- Elfelejtett jelszó esetén e-mail cím megadási lehetőség, és ideiglenes jelszó megküldése a megadott címre.
- Karitatív szervezetekkel és cégekkel való bővülés.
- Település input mező keresőmezőként működjön, és adatbázis alapján kínálja fel a lehetséges opciókat. Jelenlegi megvalósítás szerint nem alfabetikus sorrendben találhatóak meg a települések, mely keresési hibákat eredményez begépeléskor, illetve Budapest csak kerületenként kereshető.
- Lejárt hirdetéseket ne listázza keresési találatként, akár input mezőbe írható dátum szabályozásával (max, min attribútumok).
- Egy adott hirdetés megtekintése esetén továbblépési lehetőség a feladó további hirdetéseire.
- A hirdetésben található cím térképes megjelenítése Google Maps API segítségével.
- Alkalmazás értesítés (emlékeztető üzenet) küldés például az önkéntes munka kezdete előtt 1 órával.
- A hirdetés ne váljon a jelentkezés elfogadásakor teljesítetté, csak a munka elvégzését követően. Az önkéntes munkára jelentkező személy csak a munka tényleges elvégzését követően kapjon 1 pontot.
- Szabadszöveges beviteli mezők ellenőrzése, így például az irrelevant szavak és nem helyénvaló tartalom kiszűrése.
- A regisztrációnál a Felhasználói feltételek és Adatvédelmi irányelvek kötelező elfogadása előtt megtekintési lehetőséget biztosítani.

6.2 Vékonykliens oldal – Webalkalmazás

A következő fejezetekben a webalkalmazás önállóan végzett backend és frontend fejlesztése kerül részletes bemutatásra.

6.2.1 Az alkalmazott fejlesztői eszközök

A webapplikáció egyéni backend és frontend fejlesztéséhez 1.71.0 verziószámú Visual Studio Code kódszerkesztőt, 3.1.13 verziószámú CodeIgniter keretrendszert, valamint 5.1.1 verziószámú phpMyAdmin szoftvert használtam. A backend fejlesztés 8.0.10 verziószámú PHP programozási nyelven történt, míg a frontend fejlesztéshez JavaScript programozási nyelvet, HTML leíró nyelvet, CSS stíluslapot és Bootstrap CSS keretrendszert alkalmaztam. A fejlesztői környezet kialakítása a következőkben ismertetett műveletek elvégzésével történt.

A XAMPP - phpMyAdmin letöltését követően, a számítógép telepítéskor megadott meghajtóján - esetemben ez a C meghajtó (Helyi lemez) - egy xampp nevű mappa található, amelynek egy almappja a htdocs nevű mappa. A fejlesztési folyamat nyomon követhetősége érdekében a GitHub nevű verziókezelőn létrehoztam a Vizsgaremek_Web elnevezésű remote repositoryt, melyet ezt követően klónozással lokálisan is inicializáltam a htdocs nevű mappában. A verziókövetést biztosító, .git elnevezésű mappát tartalmazó Vizsgaremek_Web mappába bemásoltam, majd kicsomagoltam a CodeIgniter keretrendszert tartalmazó .zip állományt. A mappában megtalálható egy .htaccess konfigurációs fájl, mely a weboldalakon a felhasználó által látható URL elnevezésének módosítását teszi lehetővé. A mappa tartalmaz még egy .env.development elnevezésű fájlt, mely a .env.example adott környezetre vonatkozó adatokkal elmentett verziója és biztosítja, hogy az alkalmazás környezeti változóként kezelje a CodeIgniter elérési útvonalát, valamint az adatbázisra vonatkozó adatokat.

A config.php elnevezésű fájlban a következő módosítást szükséges elvégezni:

```
$config['base_url'] = getenv("BASE_URL");
```

A database.php elnevezésű fájlban pedig az alábbi beállítások elvégzése szükséges:

```
'username' => getenv("DB_USERNAME"),  
'password' => getenv("DB_PASSWORD"),  
'database' => getenv("DB_DATABASE"),
```

6.2.2 Részletes feladat-specifikáció, algoritmusok

A CodeIgniter MVC keretrendszerben megvalósított, a vezérlésért felelős controller osztályokat, az azokban felhasznált model osztályokat, és ezek metódusait, valamint a frontendhez köthető, az adatok megjelenítéséért felelős nézeteket (view) jelen fejezetben ismertetem. A controllerek felelnek a modellekben található lekérdezések eredményeinek

feldolgozásáért, a kívánt nézet (view) betöltéséért és a nézeten történő megjelenítéshez szükséges adattovábbításért is.

Minden controller a CI_Controller őszosztály leszármazottja, így a konstruktorban minden esetben meghívom a szülőosztály konstruktorát, valamint betöltöm az URL Helper fájlt, mely az URL webcímek kezeléséhez szükséges metódusokat tartalmaz. Ezenkívül betöltöm a Session Library nevű könyvtárat, mely a munkamenetek alkalmazását teszi lehetővé, illetve minden olyan közösen elkészített modelt, melynek valamely metódusát az adott controller osztályban használom.

```
public function __construct(){
    parent::__construct();
    $this->load->helper('url');
    $this->load->library('session');
    $this->load->model('telepules_model'); (controllerenként eltérő)
}
```

6.2.2.1 Kezdolap

A Kezdolap controllerben a User_model és Telepules_model metódusait használom fel az osztályhoz megírt metódusokban, így ez a kettő model került betöltésre a konstruktorban.

Az **index()** metódus a header, a kezdolap és a footer nevű nézetek betöltését valósítja meg, mely a weboldal kezdőoldalának megjelenítéséhez szükséges. A header és footer nézetek betöltésével az alkalmazás minden egyes aloldala egységes fejléccel és lábléccel rendelkezik, így ezt minden controller aloldal betöltéséért felelős metódusánál elvégzem. Az alkalmazás menürendszere a header.php fájlban került elhelyezésre és attól függően változik, hogy történt-e bejelentkezés a weboldalon. Ennek vizsgálata többek között az alábbi kódrészlettel történik:

```
<?php if ($this->session->userdata('user') !== NULL) : ?>
```

Amennyiben a session munkamenetben eltárolt userdata('user') NULL értéket vesz fel, a menüpontok közül csak a Főoldal, a Bejelentkezés és a Regisztráció érhető el. Ellenkező esetben láthatóvá válik a Főoldal, a Hirdetések keresése, a Hirdetés feladása, a Profil, a Ranglista és a Kijelentkezés menüpont. A footer.php fájl az Önkéntes Portál elérhetőségeit tartalmazza.

A **regisztracio()** metódus a regisztracio view betöltéséért felel, és a Telepules_modelben található get_all() és telepulesekSzama() metódusokban található lekérdezések eredményeit egy \$data asszociatív tömbben eltárolva adja át a nézetnek. Ezzel a megoldással a regisztrációs űrlapon a település kiválasztásakor a felhasználók a települések nevét látják, de az adatbázisba a kiválasztott település azonosítója kerül be.

A **regisztracio_post()** metódus felel a regisztrációs űrlap felhasználó általi beküldések az input mezőkben tárolt adatok kezeléséért, backend validálásáért, valamint a megfelelő adatok `User_model user_rogzitese($data)` metódusának történő átadásáért. Az űrlapra vonatkozó form paraméterek az adatküldési metódus és a képfeltöltés miatt a következők:

```
method="post" enctype="multipart/form-data">
```

A paraméterként átadott `$data` tömbben található adatok az adatbázis `user` táblájába új rekordot létrehozva kerülnek be. Az adatok backend validálását a Form Validation Library segítségével, míg frontend validálását Javascript nyelven írt validálási függvényekkel, valamint HTML Input Attribútumok megadásával valósítottam meg. A jelszavak titkosítása a `password_hash()` metódussal történik. A felhasználók által feltöltött képek az erre a célra létrehozott `uploads` nevű mappába kerülnek, míg a weboldalhoz használt képek - kategóriaképek, helyezettképek és kezdőlapi képek - az `images` nevű mappában kerültek elhelyezésre. A fejlesztés során számos egyéni útvonalat hoztam létre a `routes.php` fájlban. A regisztrációra vonatkozó útvonalak például az alábbiak:

```
$route['regisztracio']['GET'] = 'kezdolap/regisztracio';
$route['regisztracio']['POST'] = 'kezdolap/regisztracio_post';
```

A **bejelentkezes()** metódus a bejelentkezes view betöltéséért, míg a **bejelentkezes_post()** metódus a bejelentkezést szolgáló űrlap felhasználó általi beküldések az input mezőkben tárolt adatok kezeléséért, backend validálásáért, valamint a megfelelő adatok `User_model kereses_felhnev_alapjan()` metódusának történő átadásáért felel. A model alapján, a regisztrált felhasználó adatait visszakapva eldönthető, hogy átesett-e már a hitelesítési folyamaton, illetve megfelelő-e az adott felhasználónévhez tartozó jelszó (`password_verify()` metódus).

```
$felhasznalo = $this->user_model->kereses_felhnev_alapjan($felhnev);
$array = array(
    'user' => $felhasznalo
);
$this->session->set_userdata($array);
```

Sikeres bejelentkezés esetén a `$_SESSION['user']` globális változóban eltárolt felhasználó adatai a munkamenet lejártáig (`$config['sess_expiration'] = 7200; [s]`) az alkalmazáson belül bárhol elérhetőek.

A regisztrációra vonatkozó útvonalak a következők:

```
$route['bejelentkezes']['GET'] = 'kezdolap/bejelentkezes';
$route['bejelentkezes']['POST'] = 'kezdolap/bejelentkezes_post';
```

A *kijelentkezés()* metódus a felhasználó kiléptetését végzi, a munkamenetben tárolt adatok törlése mellett.

A Kezdolap controllert a routes.php fájlban alapértelmezett controllerként definiáltam.

```
$route['default_controller'] = 'kezdolap';
```

6.2.2.2 Profil

A Profil controllerben a User_model, Telepules_model, Hirdetes_model és Jelentkezes_model metódusait használom fel az osztályhoz megírt metódusokban, így ez a négy model került betöltésre a konstruktorban.

A *profil_megtekintes()* metódus a profil view betöltését valósítja meg és a következőkben megjelölt modellek - nem teljeskörűen kifejtett, szemléltetésképpen bemutatott - metódusaiban található lekérdezések eredményeit egy \$data asszociatív tömbben eltárolva adja át a nézetnek:

18. táblázat: Profil controller – profil_megtekintes()

Forrás: Saját készítésű táblázat

User_model	
Metódusnév	user_lekerdezese_id_alapjan()
Paraméter	\$_SESSION['user']['user_id']
Adat	\$data['userAdatai'] = \$userAdatai
Funkció	Felhasználó adatainak megjelenítése az adatbázis user táblájából.
Metódusnév	kategoriaKepekListazasa()
Paraméter	\$_SESSION['user']['user_id']
Adat	\$data['kategoriaKepek'] = \$kategoriaKepek
Funkció	A felhasználó által teljesített hirdetésekhez tartozó kategóriaképek megjelenítése. (Díjaim)
Telepules_model	
Metódusnév	get_by_id()
Paraméter	\$userAdatai[0]['telepules_id']
Adat	\$data['telepules'] = \$telepules
Funkció	A user táblában tárolt telepules_id helyett a település nevének megjelenítése.
Hirdetes_model	
Metódusnév	kategoriaKepekListazasaJelentkezes()

Paraméter	<code>\$_SESSION['user']['user_id']</code>
Adat	<code>\$data['kategoriaKepekJelentkezesek'] = \$kategoriaKepekJelentkezesek</code>
Funkció	Azon hirdetésekhez tartozó kategóriaképek megjelenítése, amelyekre jelentkezett a felhasználó. (Hirdetések, amikre jelentkeztem)
Metódusnév	<code>hirdetesekAdataiAmikreJelentkezettAFelhasznalo()</code>
Paraméter	<code>\$_SESSION['user']['user_id']</code>
Adat	<code>\$data['hirdetesIdkJelentkezesek'] = \$hirdetesIdkJelentkezesek</code>
Funkció	Azon hirdetésekhez tartozó hirdetés azonosítók megjelenítése, amelyekre jelentkezett a felhasználó. (Hirdetések, amikre jelentkeztem)
Metódusnév	<code>kategoriaKepekSajatHirdetesek()</code>
Paraméter	<code>\$_SESSION['user']['user_id']</code>
Adat	<code>\$data['kategoriaKepekSajatHirdetesek'] = \$kategoriaKepekSajatHirdetesek</code>
Funkció	Azon hirdetésekhez tartozó kategóriaképek megjelenítése, amelyeket a felhasználó adott fel és még nem jelentkeztek rá vagy nem vált teljesítetté. (Saját hirdetéseim)
Metódusnév	<code>kategoriaKepekJovahagyasravarohirdetesek()</code>
Paraméter	<code>\$_SESSION['user']['user_id']</code>
Adat	<code>\$data['jovahagyasravarokategoriaKepek'] = \$jovahagyasravarokategoriaKepek</code>
Funkció	Azon hirdetésekhez tartozó kategóriaképek megjelenítése, amelyeket a felhasználó adott fel, jelentkeztek rá, és jóváhagyás szükséges. (Jóváhagyásra váró hirdetéseim)

A profiloldalon a különböző hirdetések megjelenítéséhez a nézetnek átadott asszociatív \$data adattömb megfelelő kulcsa alapján az adott tömb elemszámát használom fel ciklusfeltételként és for ciklus segítségével dinamikusán, Bootstrap cardok formájában jelenítem meg a hirdetéseket. Minden hirdetés alatt elhelyezkedik egy 'Tovább a hirdetésre' elnevezésű gomb, melynek segítségével a felhasználót átirányítom az adott hirdetés adatlapjára - hirdetes view. Az átirányítás megvalósítása az alábbi útvonal segítségével történik:

```
$route['hirdetes/(:num)'] = 'hirdetes_kereses/hirdetes_megtekintes/$1';
```

A (:num) és \$1 jelzi, hogy egy int típusú paramétert kell átadni ahhoz, hogy az átirányítás működjön. A hirdetes view ismertetésére a Hirdetes_kereses controller bemutatásánál kerül sor.

A *profil_modositas()* metódus a modositott_profil view betöltését valósítja meg és az adatbázis telepules táblájából származó összes adatot, a települések darabszámát és a felhasználó adatait \$data tömbben eltárolva adja át a nézetnek.

A *profil_modositas_post(\$user_id)* metódus a profil oldalon elhelyezett 'Profil módosítása' elnevezésű gombra történő kattintással elérhető, a személyes adatok módosítására szolgáló űrlap felhasználó általi beküldésekor az input mezőkben tárolt adatok kezeléséért, valamint a megfelelő adatok User_model user_modositasa(\$user_id, \$data) metódusának történő átadásáért felel. A gombra történő kattintással az alábbi átirányítás valósul meg:

```
$route['modositas/(:num)']['POST'] = 'profil/profil_modositas_post/$1';
```

A felhasználó adatait azért szükséges átadni a nézetnek, mert az űrlapba alapértelmezetten betöltésre kerülnek a bejelentkezett felhasználó adatai, amelyek módosítást követően már nem egyeznek meg a bejelentkezéskor a \$_SESSION globális változóban eltárolt személyes adatokkal.

A *profil_modositas_jelszo_post(\$user_id)* metódus a profil oldalon elhelyezett 'Profil módosítása' elnevezésű gombra történő kattintással elérhető, a jelszó módosítására szolgáló űrlap felhasználó általi beküldésekor az input mezőkben tárolt adatok kezeléséért, validálásáért, valamint a megfelelő adatok User_model userJelszoModositasa(\$jelszo,\$user_id) metódusának történő átadásáért felel. A gombra történő kattintással az alábbi átirányítás valósul meg:

```
$route['jelszomodositas/(:num)']['POST'] = 'profil/profil_modositas_jelszo_post/$1';
```

A jelszó módosítás folyamatában először a jelenlegi jelszó ellenőrzésére kerül sor a password_verify() metódussal, majd az újonnan megadott két jelszó egyezését szükséges megvizsgálni. Egyezés esetén az új jelszó adatbázisban történő rögzítése előtt a password_hash() metódus segítségével titkosítást kell végrehajtani.

A *profil_torles()* metódus a profil oldalon elhelyezett 'Profil törlése' elnevezésű gombra történő kattintást követően a felhasználó \$user_id azonosítóját átadva a User_model user_torlese() metódusának a felhasználót törli az adatbázis user táblájából és kijelentkezteti a weboldalról. Az oldalon elhelyeztem egy JavaScriptben íródott profilTorlese() nevű függvényt, mely a gombra kattintást követően a felhasználó megerősítését kéri a folyamat végrehajtása előtt.

A *hirdetes_torles(\$hirdetes_id)* metódus a felhasználó minden saját hirdetése alatt elhelyezett 'Törlés' elnevezésű gombra kattintást követően törli a felhasználó adott hirdetését

az adatbázis hirdetes táblájából oly módon, hogy a hirdetés azonosítóját paraméterként átadja a Hirdetes_model sajátHirdetesTorlese(\$hirdetes_id) metódusának. A törlés megvalósítása az alábbi útvonal segítségével történik:

```
$route['hirdetestorles/(:num)'] = 'profil/hirdetes_torles/$1';
```

Az oldalon elhelyeztem egy JavaScriptben íródott hirdetesTorlese(hirdetes_id) nevű függvényt, mely a gombra kattintást követően a felhasználó megerősítését kéri a törlési művelet végrehajtása előtt.

A **hirdetes_elfogadasa(\$hirdetes_id)** metódus a felhasználó minden jóváhagyásra váró hirdetése alatt elhelyezett 'Elfogadás' elnevezésű gombra kattintást követően az adott hirdetésre vonatkozóan az adatbázis jelentkezes táblájában található jovahagyas_hirdeto mezőben tárolt false értéket true értékre módosítja úgy, hogy a Jelentkezes_model jelentkezesJovahagyasa(\$hirdetes_id) metódusnak az adott hirdetés azonosítóját átadja. A jóváhagyás megvalósítása az alábbi útvonal segítségével történik:

```
$route['hirdeteselfogadas/(:num)'] = 'profil/hirdetes_elfogadasa/$1';
```

A profil oldalon az alábbi script kód felel a dinamikusan megjelenített hirdetések egyedi HTML id-ja alapján azért, hogy a HTML href attribútumában automatikusan a megfelelő útvonal legyen beállítva és megtörténjen a fenti átirányítás.

```
function hirdetesElfogadasa(hirdetes_id) {
    var utvonal = "<?php echo base_url(); ?>hirdeteselfogadas/" + hirdetes_id;
    var elfogadasid = "#elfogadasgomb" + hirdetes_id;
    document.querySelector(elfogadasid).setAttribute("href", utvonal);
}
```

A **hirdetes_elutasitasa(\$hirdetes_id)** metódus a felhasználó minden jóváhagyásra váró hirdetése alatt elhelyezett 'Elutasítás' elnevezésű gombra kattintást követően az adott hirdetésre vonatkozóan az adatbázis jelentkezes táblájából törli az adott hirdetésre vonatkozó jelentkezést úgy, hogy a Jelentkezes_model jelentkezesTorleseElutasitasMiatt(\$hirdetes_id) metódusnak az adott hirdetés azonosítóját átadja. A törlési műveletet eredményező elutasítás megvalósítása az alábbi útvonal segítségével történik:

```
$route['hirdeteselutasitas/(:num)'] = 'profil/hirdetes_elutasitasa/$1';
```

6.2.2.3 *Hirdetes_kereses*

A *Hirdetes_kereses* controllerben a *Kategoria_model*, *Telepules_model*, *Hirdetes_model* és *Jelentkezes_model* metódusait használom fel az osztályhoz megírt metódusokban, így ez a négy model került betöltésre a konstruktorban.

A *hirdetes_kereses()* metódus a *hirdetes_kereses* view betöltését valósítja meg és a következőkben megjelölt modellek - nem teljeskörűen kifejtett, szemléltetésképpen bemutatott - metódusaiban található lekérdezések eredményeit egy *\$data* asszociatív tömbben eltárolva adja át a nézetnek:

19. táblázat: Hirdetes_kereses controller – hirdetes_kereses()

Forrás: Saját készítésű táblázat

Kategoria_model	
Metódusnév	<i>kategoria_lista()</i>
Paraméter	
Adat	<i>\$data['kategoria_nev'] = \$kategoria_nev</i>
Funkció	Az adatbázis kategoria táblájában tárolt kategórianevek megjelenítése.
Telepules_model	
Metódusnév	<i>get_all()</i>
Paraméter	
Adat	<i>\$data['telepules'] = \$telepules</i>
Funkció	A telepules táblában tárolt összes adat megjelenítése.
Hirdetes_model	
Metódusnév	<i>hirdetesKereses()</i>
Paraméter	<i>\$telepules_id, \$kategoria_id, \$kezdo_idopont, \$user_id</i>
Adat	<i>\$data['lekerdezettHirdetesek'] = \$lekerdezettHirdetesek</i>
Funkció	Az adatbázis hirdetes táblájából azon hirdetések adatainak megjelenítése, melyeknél a megadott paraméterek megfelelnek a felhasználó által bejelölt feltételeknek, azaz a szűrés eredménye egy találati lista.

A hirdetések szűrését szolgáló űrlapon kategória, település és dátum szerint lehetséges keresést indítani. Minden szűrési feltétel megadása kötelező és csak mind a három feltétel együttes teljesülése esetén kapunk találatot. Amennyiben a Szűrés gomb megnyomásra került,

és a szűrési feltételeknek egyetlen hirdetés sem felel meg, az alábbi visszajelzést adom a felhasználónak:

```
<?php if(count($lekerdezettHirdetesek) == 0 && isset($_POST["szures"])){
    echo "Nincs találat a megadott paraméterek alapján.";
}?>
```

A hirdetések kilistázásakor a Bootstrap kártyákban való egységes megjelenítés érdekében a hirdetések eltérő hosszúságú leírását egységesen 50 karakterig íratom ki az alábbi kódrészletben található módon:

```
<?php      echo      strlen($lekerdezettHirdetesek[$i]['leiras'])      >      50      ?
substr($lekerdezettHirdetesek[$i]['leiras'], 0, 47) . "...": $lekerdezettHirdetesek[$i]['leiras']; ?>
```

A *hirdetes_megtekintes(\$id)* metódus a hirdetes view betöltését valósítja meg és a Telepules, Kategoria és Hirdetes modellek egyes metódusaiban található lekérdezések eredményeit egy \$data asszociatív tömbben eltárolva adja át a nézetnek. A profil nézetről és a hirdetesek_keresese nézetről ez a metódus biztosítja azt, hogy az adott azonosítóval rendelkező hirdetés adatlapjára történjen a felhasználó átirányítása. A hirdetes nézeten a paraméterként átadott azonosító alapján az adatbázis hirdetes táblájában található hirdetés adatait jelenítem meg, valamint a Hirdetes_model nevProfilkep(\$hirdetes_id) metódusa alapján az adott hirdetést feladó személy profilképét.

Minden egyes hirdetés adatlapján megtalálható egy 'Jelentkezem' gomb, mely az adott önkéntes munkára való jelentkezést teszi lehetővé. A *hirdetesre_jelentkezés(\$hirdetes_id)* metódus felel egy jelentkezés rögzítéséért úgy, hogy a Jelentkezés_model jelentkezes_rogzitese(\$data) metódusának az alábbi módon és tartalommal adja át a \$data tömböt:

```
$data['hirdetes_id'] = $hirdetes_id;
$data['jelentkezo_id'] = $_SESSION['user']['user_id'];
$data['jovahagyas_onkentes'] = "true";
$this->jelentkezés_model->jelentkezés_rogzitese($data);
```

A jelentkezés sikeres rögzítését megelőzően a jelentkezési folyamat megvalósíthatóságát a következő szempontok szerint ellenőrzöm:

20. táblázat: Hirdetes_kereses controller – hirdetesre_jelentkezés()*Forrás: Saját készítésű táblázat*

Jelentkezés_model	
Metódusnév	jelentkezesTablaHirdetesIdk()
Paraméter	
Adat	\$tomb['jelentkezesIdk'] = \$jelentkezesIdk
Funkció	Az adatbázis jelentkezes táblájában tárolt hirdetés azonosítók megjelenítése. Amennyiben az adott hirdetés már szerepel a táblában, újabb jelentkezés nem megvalósítható.
Hirdetes_model	
Metódusnév	hirdetesAzonositoAlapjan()
Paraméter	\$hirdetes_id
Adat	\$hirdetesekek
Funkció	Az adott hirdetésben található hirdeto_id értéket jeleníti meg. Ezzel az értékkel összehasonlítható a jelentkezést elindító felhasználó user azonosítója. Amennyiben a két id egyezik, a jelentkezés sikertelen.

A sikeres jelentkezéshez szükséges megfelelő átirányítás az alábbi módon történik:

\$route['hirdetesjelentkezes/(:num)'] = 'hirdetes_kereses/hirdetesre_jelentkezes/\$1';
--

6.2.2.4 Hirdetes_feladas

A Hirdetes_feladas controllerben a Katagoria_model, Telepules_model és Hirdetes_model metódusait használok fel az osztályhoz megírt metódusokban, így ez a három model került betöltésre a konstruktorban.

A *hirdetes_feladas()* metódus a hirdetes_feladas view betöltéséért felel és az adatbázis telepules és katagoria táblájából származó összes adatot valamint a települések és kategóriák darabszámát egy \$data tömbben eltárolva adja át a nézetnek. A lekérdezések eredményeinek felhasználásával megoldható, hogy a felhasználó a hirdetés feladásához szükséges űrlap kitöltésekor a települések és kategóriák nevét lássa, miközben az adatbázis hirdetes táblájában a telepules_id és katagoria_id kerül rögzítésre.

A *hirdetes_post()* metódus felel a hirdetés feladási űrlap felhasználó általi beküldésekor az input mezőkben tárolt adatok kezeléséért, backend validálásáért, valamint a megfelelő adatok Hirdetes_model hirdetes_rogzitese(\$data) metódusának történő átadásáért. A paraméterként átadott \$data tömbben található adatok az adatbázis hirdetes táblájába új rekordot létrehozva kerülnek be. Az adatok backend validálását a Form Validation Library

segítségével, míg frontend validálását Javascript nyelven írt validálási függvényekkel, valamint HTML Input Attribútumok megadásával valósítottam meg. A hirdetés feladására vonatkozó útvonalak - a kérés típusától függően - az alábbiak:

\$route['hirdetes_feladas']['GET'] = 'hirdetes_feladas/hirdetes_feladas';
\$route['hirdetes_feladas']['POST'] = 'hirdetes_feladas/hirdetes_post';

6.2.2.5 Statisztika

A Statisztika controllerben a Hirdetes_model, Ranglista_model és Kategoria_model metódusait használok fel az osztályhoz megírt metódusokban, így ez a három model került betöltésre a konstruktorban.

A *ranglista_megtekintes()* metódus a statisztika view betöltéséért felel és a következőkben megjelölt modellek - nem teljeskörűen kifejtett, szemléltetésképpen bemutatott - metódusaiban található lekérdezések eredményeit egy \$data asszociatív tömbben eltárolva adja át a nézetnek:

21. táblázat: Statisztika controller – ranglista_megtekintes()

Forrás: Saját készítésű táblázat

Ranglista_model	
Metódusnév	darabszamKategoriankent()
Adat	\$data['kategoriaDarabszamok'] = \$kategoriaDarabszamok;
Funkció	Kategóriánként megadja a hirdetések darabszámát.
Metódusnév	kategoriaAdatok()
Adat	\$data['kategoriaAdatok'] = \$kategoriaAdatok
Funkció	Megjeleníti a kategoria táblában található összes adatot.
Metódusnév	rangLista()
Adat	\$data['profilKepek'] = \$profilKepek
Funkció	Megjeleníti az első 5 legmagasabb pontszámmal rendelkező felhasználó profilképét.
Hirdetes_model	
Metódusnév	hirdetesekSzama()
Adat	\$data['hirdetes_szam'] = \$hirdetes_szam

Funkció	Az adatbázis hirdetes táblájában található összes hirdetés darabszámát adja vissza.
----------------	---

A statisztika view nézeten az adatbázisban szereplő felhasználók pontszámai alapján kialakult rangsor, valamint a hirdetések kategória szerinti százalékos eloszlása kerül megjelenítésre, mely az adatbázisból lekért adatok alapján egy egyszerű számítás elvégzését követően határozható meg:

```
<?php echo round($kategoriaDarabszamok[$i]['Darabszam']/$hirdetes_szam*100, 1) . "%"?>
```

6.2.3 Tesztelési dokumentáció

Jelen fejezetben a webalkalmazáshoz köthető teszteseteket mutatom be, menüpontonként ismertetve a program elvárt működésének megfelelő, valamint a fejlesztés során sikertelenül/hibásan megvalósított funkciókat. A felhasználó számára készített visszajelzések és hibaüzenetek többsége a session->set_flashdata() metódus segítségével került beállításra. Ezek lényege, hogy csak egyetlen átirányításig képezik a bejelentkezéskor indított munkamenet részét.

6.2.3.1 Menürendszer és visszajelzések

A program elvárt működése szerint az elérhető menüpontok aszerint változnak, hogy történt-e bejelentkezés a felületen, és a sikeres bejelentkezés és kijelentkezés tényéről tájékoztatást kap a felhasználó.

3. ábra: Bejelentkezés utáni menü és visszajelzés

Forrás: Saját készítésű ábra

Főoldal Hirdetések keresése Hirdetés feladása Profil Ranglista Kijelentkezés
Sikeres bejelentkezés!

4. ábra: Kijelentkezés utáni menü és visszajelzés

Forrás: Saját készítésű ábra

Főoldal Bejelentkezés Regisztráció
Sikeres kijelentkezés!

A weboldal számos helyen reszponzív működésű, azaz a betöltött nézetek megjelenítése a képernyő méretétől függően változik. Ennek tesztelését a menürendszer segítségével mutatom be a következő ábrán.

5. ábra: Reszponzivitás

Forrás: Saját készítésű ábra



A menüpontok színe alapértelmezetten fekete. Amennyiben a felhasználó egy adott menüponton áll, fehér, míg ha az egérmutatóval rááll a kiválasztott menüpontra, akkor bordó színűvé válik az adott menüpont szövege. A felsorolt beállításokat, valamint a header és footer egységes kinézetét a public\css mappában elhelyezett style.css egyéni stíluslapban található formázásokkal oldottam meg. A stíluslap a header view <head> részében került beimportálásra.

A helyes működést az alábbi kép szemlélteti:

6. ábra: Ranglista menüpont

Forrás: Saját készítésű ábra



6.2.3.2 Hirdetések keresése

Hirdetések keresésénél alapesetben a Szűrés gomb megnyomása előtt nem jelenik meg semmilyen szöveg a 'Találatok:' felirat alatt, míg a Szűrés gombot megnyomva 0 találat esetén visszajelzést kap a felhasználó.

7. ábra: Szűrés – visszajelzés

Forrás: Saját készítésű ábra



Amennyiben a keresési feltételeknek megfelelő, találatként kilistázott hirdetések alatti 'Tovább a hirdetésre' gombra kattint a felhasználó, az adott hirdetés adatlapja töltődik be, azonban visszalépve a **program hibára fut**, a keresési eredmények nem tekinthetők meg többé, és az alábbi képen látható hibaüzenet jelenik meg. A találatok kilistázásakor látható, hogy a szűrés indítása előtt megadott kritériumok is elvesznek, és a beviteli mezők visszaállnak alapértelmezett állapotba.

8. ábra: Hirdetések keresése – Hiba

Forrás: Saját készítésű ábra a kapott hibaüzenetről



6.2.3.3 Regisztráció

A regisztrációs űrlap validálása backend oldalon a CI Form Validation Library, frontend oldalon pedig JavaScript függvények és HTML attribútumok alkalmazásával történik. A Form Validation Library használata a következő kódrészletben látható:

```
$this->form_validation-
>set_rules('felhnev','Felhasználónév','trim|required|is_unique[user.felhnev]|min_length[6]|max_length[100]');
```

Amennyiben olyan hiba lép fel valamely beviteli mező esetében, melynek hibakezelését a Form Validation Library végzi, a hibaüzenetek angol nyelven jelennek meg. A CodeIgniter projekt application/language mappában elhelyeztem néhány beviteli mezőre vonatkozó magyar fordítást, azonban a config.php fájlban megtalálható \$config['language']= 'hungarian'; beállítást követően a program magyarul írta ki a lefordított hibaüzeneteket, de minden más szerver oldali hibajelzés esetén a program nem találta a lang fájlt. Ezt a fejlesztés során **nem sikerült megoldani**, így visszaállítottam a nyelvet angolra. Ennek következtében a felhasználó **angol nyelvű rendszerüzeneteket kap**, melyet az alábbi képen mutatok be:

9. ábra: Angol nyelvű hibaüzenet

Forrás: Saját készítésű ábra



A regisztrációs űrlap validálási feltételeit és a kapcsolódó teszt eseteket a következő táblázatban mutatom be.

Előforduló automatikus validálási hibaüzenetek:

- Töltse ki ezt a mezőt.
- A kért formátumban adja meg az adatot.
- + Form Validation Library angol nyelvű hibaüzenetek

JavaScript validálási függvényekkel megvalósított hibaüzenetek:

- A megadott felhasználónév nem lehet 6 karakternél rövidebb.
- A megadott jelszavak nem egyeznek meg.
- A megadott cím nem lehet 8 karakternél rövidebb.
- A megadott telefonszám nem lehet 7 karakternél rövidebb.

22. táblázat: Regisztráció - Teszt esetek

Forrás: Saját készítésű táblázat

Input mező: Felhasználónév			
Validálási feltételek: <i>egyedi, min. 6 karakter hosszú, kötelező</i>			
Megadott felhasználónév	Elvárt kimenet	Kimenet	Minősítés
Évi	hibaüzenet	hibaüzenet	✓
Évikee	nincs hibaüzenet	nincs hibaüzenet	✓
	hibaüzenet	hibaüzenet	✓
Évikee (de már van ilyen felhnev eltárolva az adatbázisban)	hibaüzenet	hibaüzenet	✓
Input mező: Teljes név			
Validálási feltételek: <i>nagy kezdőbetűk, csak betűk, kötelező</i>			
Megadott teljes név	Elvárt kimenet	Kimenet	Minősítés

Nagy éva	hibaüzenet	hibaüzenet	✓
Nagy Éva	nincs hibaüzenet	nincs hibaüzenet	✓
5	hibaüzenet	hibaüzenet	✓
	hibaüzenet	hibaüzenet	✓
Input mező: Telefonszám			
Validálási feltételek: <i>min. 7 karakter hosszú, kötelező</i>			
Megadott telefonszám	Elvárt kimenet	Kimenet	Minősítés
+361	hibaüzenet	hibaüzenet	✓
+361 111 1111	nincs hibaüzenet	nincs hibaüzenet	✓
	hibaüzenet	hibaüzenet	✓
Input mező: Cím			
Validálási feltételek: <i>min. 8 karakter hosszú, kötelező</i>			
1021,	hibaüzenet	hibaüzenet	✓
1021 Utca 5.	nincs hibaüzenet	nincs hibaüzenet	✓
	hibaüzenet	hibaüzenet	✓
Input mezők: Jelszó + Jelszó megerősítése			
Validálási feltételek: <i>min. 6 karakter hosszú, számot is tartalmaz, kötelező, megegyezik</i>			
évike, évike	hibaüzenet	hibaüzenet	✓
évike6, évike6	nincs hibaüzenet	nincs hibaüzenet	✓
évike, évike6	hibaüzenet	hibaüzenet	✓

Sikertelen regisztráció esetén több beviteli mezőbe is visszatöltésre kerülnek a felhasználó által megadott adatok, így nem szükséges mindent újból megadni. Ennek megvalósítására példa az alábbi kódrészlet:

```
<?php if ($this->session->flashdata('last_request') !== null) : ?> value="<?php echo ($this->session->flashdata('last_request')['okmanyszam']) ?>"
```

6.2.3.4 Bejelentkezés

A bejelentkezési űrlap validálását a már korábban ismertetett `bejelentkezes_post()` metódus végzi el. A sikertelen bejelentkezésnek két oka lehet, az egyik esetben helytelen vagy még nem regisztrált felhasználónév és/vagy jelszó kerül megadásra. Ilyenkor a következő ábrán szereplő hibaüzenetet írja ki a program:

10. ábra: Bejelentkezési hibaüzenet

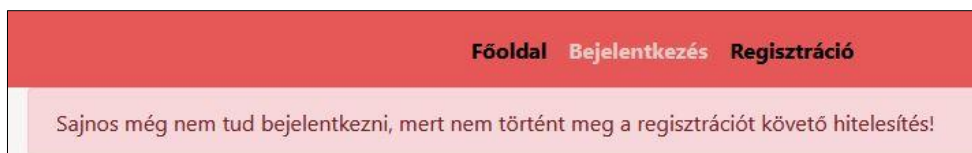
Forrás: Saját készítésű ábra



A második hibaüzenet a hitelesítési folyamat elvégzéséig íródik ki a bejelentkező felhasználók számára, mert ekkor a bejelentkezési folyamatot meggátolja a program:

11. ábra: Bejelentkezési hibaüzenet

Forrás: Saját készítésű ábra



6.2.3.5 Hirdetés feladása

A hirdetés feladását szolgáló űrlap validálása backend oldalon a CI Form Validation Library, frontend oldalon pedig JavaScript függvények és HTML attribútumok alkalmazásával történik.

Előforduló automatikus validálási hibaüzenetek:

- Töltse ki ezt a mezőt.
- + Form Validation Library angol nyelvű hibaüzenetek

JavaScript validálási függvényekkel megvalósított hibaüzenetek:

- A mező kitöltése kötelező.
- A megadott telefonszám nem lehet 7 karakternél rövidebb.
- A megadott cím nem lehet 8 karakternél rövidebb.

Az űrlapra vonatkozó tesztelési esetek a regisztrációs űrlaphoz tartozó tesztelési esetekkel állíthatóak párhuzamba.

6.2.3.6 Statisztika

A Ranglista menüpont hirdetések százalékos eloszlását bemutató szekciója manuálisan tesztelhető oly módon, hogy az egyes hirdetéskategóriák mellett megjelenő % -os értékeket összeadjuk. Amennyiben az összeg nem ~100 %, az értékeket kiszámító metódus helytelenül került megírásra, vagy pedig a számoláshoz szükséges adatokat biztosító lekérdezésekben található hiba.

6.2.3.7 Profil

A Profil menüpontban a lekérdezések alapján megjelenő, a bejelentkezett felhasználóra vonatkozó személyes adatok, valamint a díjaknál található képek és a különböző hirdetéslistákban (saját, jelentkezés, jóváhagyás) megjelenő hirdetések helyessége manuálisan ellenőrizhető egy adott felhasználóra vonatkozóan.

Bármelyik hirdetésre kattintva ellenőrizhető annak azonosítója, hiszen az az útvonalban megtalálható, így például: http://localhost/Vizsgaremek_Web/hirdetes/5.

Ide kapcsolódóan **hibaként** elmondható, hogy a fejlesztés során **nem tudtam megvalósítani** azt, hogy a felhasználó ne írhatta át az URL-t, és ezáltal ne érhesse el más aloldalakat. Ezenkívül például egy másik felhasználó által már teljesített hirdetés azonosítóját begépelve az adott azonosítóval rendelkező hirdetés adatlapját is megtekintheti. A problémát egyetlen esetben sikerült kezelnem oly módon, hogy amennyiben az adatbázis hirdetes táblájában nem található hirdetés az útvonalba beírt azonosítóval (http://localhost/Vizsgaremek_Web/hirdetes/10000), a következő hibaüzenetet kapja a felhasználó:

12. ábra: Nem létező hirdetés

Forrás: Saját készítésű ábra

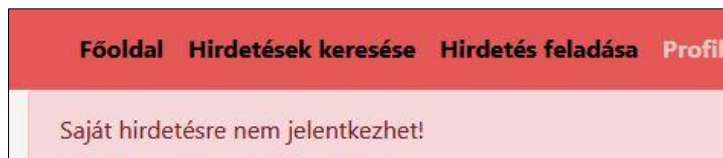


Amennyiben a felhasználó egy betűt ad meg azonosítóként (http://localhost/Vizsgaremek_Web/hirdetes/m) a program **hibára fut** a 404 Page Not Found hibaüzenet megjelenítésével.

Mivel minden hirdetés adatlapján megjelenik a 'Jelentkezem' gomb, ezért két speciális esetre vonatkozóan gátoltam a jelentkezési folyamat végrehajtását. A felhasználó nem jelentkezhet saját hirdetésére.

13. ábra: Jelentkezés gátolása

Forrás: Saját készítésű ábra



A felhasználó nem jelentkezhet olyan hirdetésre, amelyre már egy másik felhasználó jelentkezett.

14. ábra: Jelentkezés gátolása

Forrás: Saját készítésű ábra



6.2.3.8 Profil adatok és jelszó módosítása

A profil adatok és a jelszó módosítására szolgáló űrlapokban megadott adatokra nem készítettem JavaScript validációs függvényeket, valamint nem alkalmazom a Form Validation Library könyvtárat sem. HTML attribútumok segítségével végzem el a validációt a korábbi fejezetekben kifejtett feltételek alapján. A jelszó módosítására vonatkozóan három visszajelzést készítettem a felhasználók számára:

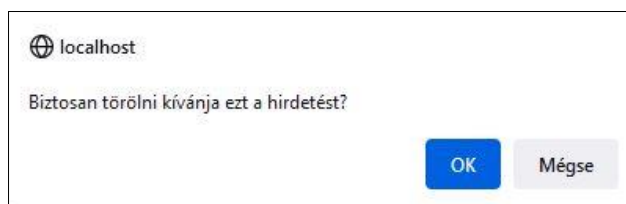
```
$this->session->set_flashdata('error', "Hibásan adta meg jelenlegi jelszavát!");
$this->session->set_flashdata('error', "Nem egyeznek meg az újonnan megadott jelszavak!");
$this->session->set_flashdata('success', "Sikeresen módosította jelszavát!");
```

6.2.3.9 Profil és hirdetés törlése

A felhasználó saját hirdetésének, valamint felhasználói fiókjának törlését megelőzően egy, a törlési szándék megerősítésére alkalmas felugró ablakot jelenítek meg.

15. ábra: Hirdetés törlés megerősítése

Forrás: Saját készítésű ábra



6.2.3.10 Jelentkezés elfogadása és elutasítása

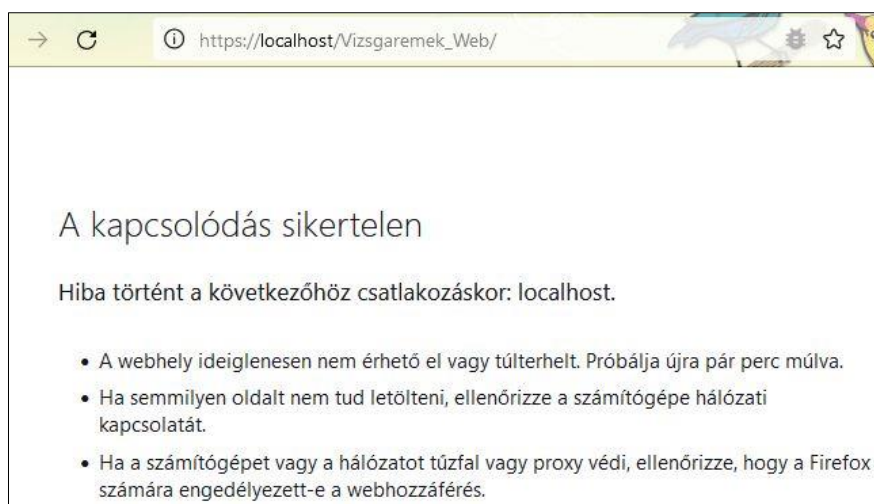
Amennyiben a felhasználó egy adott hirdetésére jelentkezett egy önkéntes, a hirdetés alatt található 'Elfogadás' gomb megnyomásával jóváhagyhatja, vagy az 'Elutasítás' gombot megnyomva elutasíthatja a jelentkezést. Mind a kettő esetben eltűnik a hirdetés a Jóváhagyásra váró hirdetések listájából. Amennyiben elutasítás történt, az adott hirdetés a profil oldalon található 'Saját hirdetésem' között kerül kilistázásra.

6.2.3.11 Webszerver, database.php adathibák

Amennyiben a XAMPP Control Panel v3.3.0 felületén nem kerül elindításra az Apache és MySQL modul, az alkalmazás nem indul el és az alábbi hibaüzenet jelenik meg:

16. ábra: Sikertelen kapcsolódás

Forrás: Saját készítésű ábra a kapott hibaüzenetről



Amennyiben például a database.php fájlban az adatbázisra vonatkozó adatokat helytelenül adom meg, a következő hibaüzenetek (részletek) megjelenítése mellett a program futása **leáll**:

An uncaught Exception was encountered Type: ParseError
A PHP Error was encountered Severity: Warning Message: mysqli::real_connect(): php_network_getaddresses: getaddrinfo failed: Nincs ilyen ismert llom s.

6.2.4 Továbbfejlesztési lehetőségek

- A CodeIgniter keretrendszerben található Form Validation Library angol nyelven elérhető validációs hibaüzeneteinek és visszajelzéseinek magyar nyelvű fordításának elkészítése. Ezzel növelhető az alkalmazás felhasználóbarát jellege.
- Weboldal akadálymentes változatának elkészítése, ezzel elősegítve, hogy azt minél több felhasználó használhassa kényelmesen.
- A CodeIgniter MVC keretrendszer mellett frontend - pl. React - és/vagy CSS keretrendszer alkalmazása a különböző nézetek megjelenítéséhez és formázásához.
- A weboldal frontend fejlesztése során kevesebb inline stílusformázás alkalmazása, a formázások .css stíluslapokba való kiszervezése.
- A weboldal fejlesztése során igyekeztem megvalósítani a reszponzivitást, azonban ez nem minden aloldalon teljesül, vagy csak részben valósul meg szép, egységes kivitelezésben. A teljes reszponzivitás megvalósítása elengedhetetlen ahhoz, hogy azt minden eszközről kényelmesen és átláthatóan lehessen használni.
- A ranglista menüpontot megvalósító view elrendezése nem kellő igényességgel került kialakításra.
- A hirdetések adatlapján megjelenő 'Jelentkezem' gomb eltávolítása vagy inaktívvá tétele azon hirdetések esetében, amelyeknél az adott felhasználó esetében nem engedélyezett a jelentkezés. Jelenleg egy darab hirdetes view készült az alkalmazáshoz, így esetleg több nézet elkészítésével és adott feltételektől függő betöltésével is orvosolható a jelzett probléma.
- A hirdetések keresése funkció tényleges, hiba nélküli megvalósítása a teljes oldal újratöltése nélkül. A korábban megadott keresési feltételek visszatöltése a beviteli mezőkbe.
- A közösen megírt backend API végpontok használata abban az esetben, ahol ez a webalkalmazások esetében előnyösebb vagy hatékonyabb működést eredményez.
- AJAX és jQuery JavaScript library használata azokban az esetekben, ahol ezek használata célravezetőbb.
- A felhasználóval való interakció növelése, minden validálási hibaüzenet kiírása, minden sikeres vagy sikertelen műveletről visszajelzés.

- A weboldal számos érzékeny adatot kezel, így sérülékenységét minimalizálni kell, megfelelő védelemmel kell ellátni azt (pl.: CAPTCHA alkalmazása).
- Minden esetben teljeskörű backend és frontend validálást szükséges megvalósítani a rosszindulatú felhasználók ellenében, valamint az adatbázis leterhelésének csökkentése érdekében.
- Biztosítani kell a fejlesztés során, hogy ne fordulhasson elő olyan eset, amikor a felhasználó lejárt hirdetésekre tud jelentkezni, illetve az önkéntes munka jóváhagyása esetén előbb váljon a hirdetés teljesítetté, mint a hirdetésben megadott dátum. Jelenlegi működés szerint egyik eset sincsen lekezelve, azaz a program működésében ez egy **logikai hiba**.
- Az adatbázisból származó adatokkal feltöltött település és kategória select input mezők visszatöltése nem valósul meg sem a hirdetések keresésekor, sem sikertelen regisztráció és hirdetés feladás, sem pedig a személyes adatok módosításánál. Ez egy **működési hiba** jelenleg, hiszen a felhasználó nem számít arra, hogy nem az általa megadott személyes adat töltődik vissza a beviteli mezőbe. Mindenképpen javítani szükséges, és amennyiben lehetséges, a profilkép és okmánykép visszatöltését vagy ideiglenes mentését is szükséges lenne megoldani.
- A ranglista menüpontban látható 5 felhasználó profilképe akkor is megjelenik, ha például az adatbázis user táblájában 10 regisztrált felhasználó található, és mindegyikük 0 ponttal rendelkezik. Pontegyezés esetén az adatbázis user táblájából nem általunk beállított módon kerül kiválasztásra az, hogy az azonos ponttal rendelkező felhasználók közül melyik 5 fő jelenjen meg. Ezt a megjelenítést, valamint egyenlő pontok esetén a kiválasztási módszert finomítani szükséges.
- 6.1.5 fejezetben felsorolt továbbfejlesztési lehetőségek.

6.3 Vékonykliens oldal - Mobilalkalmazás

6.3.1 Az alkalmazott fejlesztői eszközök

6.3.1.1 *Fejlesztői környezet*

A szerver oldali programozás fejlesztői környezete a Visual Studio Code. A mobilalkalmazás az MVC keretrendszerben létrehozott model és API controller osztályokat használja működése során.

A Visual Studio Code-ban megírt PHP kódok futtatásához a XAMPP szerveret választottuk. Ezek hosszabb kifejtése a Fejlesztői dokumentáció Backend részénél már teljesült.

Az Android alkalmazás fejlesztésére a Google Android Studio fejlesztői környezetét választottam. A projekt létrehozásakor API 23: Android 6.0 (Marshmallow) apiszintet választottam, hiszen ezt a verziót az eszközök 96,2 %-a támogatja. A futtatás során emulátorként Pixel 2 virtuális eszközt, és az ehhez letöltött Q nevű 29-es apiszintű operációs rendszert használtam.

6.3.1.2 *Programozási nyelvek*

A szerver oldali programozási nyelvnek a PHP 8.0.10-s verzióját használtuk.

Az Android Studio-ban használt programozási nyelv a Java, hiszen ezzel a nyelvvel ismerkedhettünk meg a képzés mobilalkalmazás tanórái során.

6.3.1.3 *Adatbázis-kezelő rendszer*

Az adatbázis létrehozására és kezelésére a MySQL adatbázis-kezelő szerverét, ennek grafikus menedzselésére a phpMyAdmin rendszerét használtuk.

6.3.2 Adatmodell leírása

Az adatbázis felépítése a közös backend részben már kifejtésre került. Ebben a részben az objektumorientált megvalósítás szempontjából fontos néhány osztályt ismertetem. Az adatbázis táblái alapján létrehozott és használt objektumok a User, a Hirdetés és a Kategória.

Mivel ennek a 3 objektumnak a felépítése nagyon hasonló, így a User osztályt részletezem az alábbiakban.

A User objektum adattagjai:

- int típusú user_id
- String típusú nev
- String típusú felhnev
- String típusú szuldatum

- int típusú telepules_id
- String típusú telszam
- String típusú email
- String típusú jelszo
- String típusú profilkep
- String típusú okmanykep
- String típusú okmanyszam
- String típusú cim
- int típusú pontszam

Az osztálynak 3 konstruktora van. Az egyik a regisztrációkor szükséges adatokat hozza létre. (Paraméterei: user_id, nev, felhnev, szuldatum, telepules_id, telszam, email, jelszo, profilkep, okmanykep, okmanyszam, cím). A másik a profilmódosítás oldalon, a bejelentkezett felhasználó adatmódosításánál használt. (Paraméterei: user_id, telepules_id, telszam, email, cím). A harmadik a profil oldalon a felhasználó pontszámának megjelenítéséért felel. (Paraméter: pontszam). Az adattagoknak létrehoztam get() metódusokat, illetve a profilmódosításnál használt adattagoknak set() metódusokat is.

A User API controller végpontja egy 3 elemű tömbben tér vissza, ami tartalmaz egy felhasználó adatait visszaadó listát, egy boolean típusú error változót és egy String típusú message változót. Ennek az átalakításához létrehoztam egy ApiValasz nevű absztrakt generikus osztályt, amiben meg lehet adni, hogy milyen típusú elemeket tartalmazzon.

Az ApiValasz objektum adattagjai:

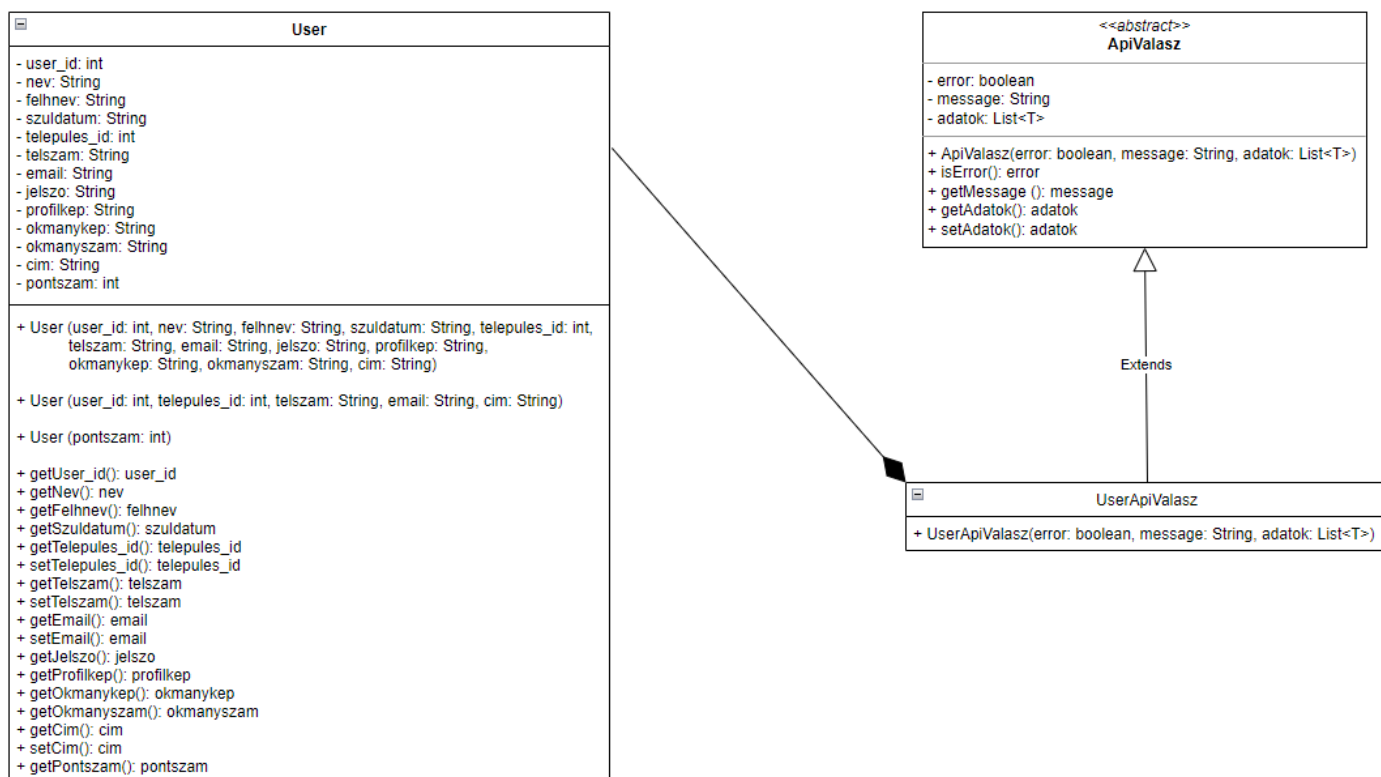
- boolean típusú error
- String típusú message
- generikus lista típusú adatok

Ezek kezelésére létrehoztam egy konstruktort, valamint get() és set() metódusokat. Az ApiValasz ősosztályt leszármaztattam egy UserApiValasz nevű osztályba. Ennek van egy konstruktora, ami meghívja az ősosztály konstruktorát.

Az alábbiakban látható a 3 osztály felépítését és kapcsolatát bemutató UML osztálydiagram.

17. ábra: UML diagram

Forrás: Saját készítésű UML osztálydiagram



6.3.3 Részletes feladat-specifikáció, algoritmusok

6.3.3.1 Kinézet-XML fájlok

Az alkalmazás kinézetéért a layout és menu mappákban szereplő xml fájlok felelnek. A menu mappán belül a menü kinézetére 2 fájlt hoztam létre. Az egyik a kijelentkezett állapotban megjelenített, a másik a bejelentkezett állapotban megjelenített menüpontokat tartalmazza. A layout mappában szereplő felhasználói felületek kialakításához többféle kinézetet alkalmaztam: `DrawerLayout`, `RelativeLayout`, `LinearLayout`, `ConstraintLayout`, `ScrollView`, `NavigationView`. Ezeken belül többféle vezérlőt használtam, attól függően, hogy mit szerettem volna megjeleníteni a felhasználó számára:

- `Toolbar`: a menürendszerhez
- `TextView`: valamilyen szöveg megjelenítéséhez
- `EditText`: a felhasználótól bekért adatokhoz
- `Button`: gomb a kattintáshoz
- `ImageView`: valamilyen kép megjelenítéséhez
- `Spinner`: értékválasztáshoz egy meghatározott érték-halmazból

- CalendarView: dátum kiválasztásához
- ListView: adatok listázásához
- CheckBox: felhasználó feltételek elfogadásához

Az alábbi kódrészletben egy Button vezérlő elemeit részletezem.

```
<Button
    android:id="@+id/buttonBejelentkezesBejelentkezes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editJelszoBejelentkezes"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="40dp"
    android:text="Bejelentkezés" />
```

A gomb kattintás hatására meghatározott műveletet képes végrehajtani. Az id a gomb egyedi azonosítója. A layout_width és layout_height elemekkel a méretet lehet szabályozni. Esetemben wrap_content, ami miatt a gomb mérete a beleírt szöveg méretével lesz egyenlő. A layout_below az igazítást adja meg egy másik vezérlőhöz képest. A layout_centerHorizontal="true" paranccsal a képernyő közepén helyezkedik el, vízszintes igazítását tekintve. A layout_marginTop a felső margót adja meg, ennek mértéke 40 dp (sűrűségfüggetlen pixel). A text a gombon megjelenített szövegért felel.

6.3.3.2 Működés-Java fájlok

1.) MainActivity.java és BejelentkezettActivity.java

Ez a két Activity az AppCompatActivity ősosztályból származik, és a drawer menürendszert implementálja. Itt töltöm be a toolbar-t és a menürendszert, valamint állítom be a Fragment-ek közötti mozgást.

A menüpontok közötti váltakozásért az onNavigationItemSelected() nevű metódus felel, ami paraméterként egy MenuItem típusú item-et vár. Az adott menüpontra kattintva a getSupportFragmentManager() metódus segítségével lehet betölteni a menüpontokhoz tartozó Fragment-eket.

A MainActivity jeleníti meg a kijelentkezett állapotban szükséges menüpontokat (Főoldal, Regisztráció, Bejelentkezés), a BejelentkezettActivity a bejelentkezett állapotban szükséges menüpontok megjelenítését adja meg (Főoldal, Hirdetések keresése, Hirdetés feladása, Profil, Ranglista, Kijelentkezés).

2.) Fooldal.java és FooldalBejelentkezett.java

Ez a két Fragment a főoldalon szereplő információkat jeleníti meg. Ha a felhasználó nincs bejelentkezve, akkor a Fooldal Fragment hívódik meg, amin szerepel egy bejelentkezés és regisztráció oldalakra navigáló gomb. Bejelentkezést követően a FooldalBejelentkezett Fragment hívódik meg, amin már nem szerepelnek ezek a gombok, ez csak a fooldal_bejelentkezett.xml fájl betöltéséért felel.

3.) RequestHandler.java és Response.java

A Response nevű objektum tartalmazza az int típusú responseCode és a String típusú content változókat. A responseCode tartalmazza, hogy milyen választ adott a kérésre a szerver.

Adatbáziskezeléshez a CRUD műveleteket használtam. A kérések kezeléséhez létrehoztam a RequestHandler nevű final osztályt, amiből nem lehet leszármaztatni. Ennek van egy privát konstruktor, amitől nem lehet példányosítani. Az itt létrehozott statikus függvények a különböző kérések végrehajtásáért felelnek.

A kapcsolatBeallitas() nevű függvény egy String típusú url-t vár paraméterként. Ebben létrehoztam egy új URL objektumot. A HttpURLConnection-t használva nyitottam meg a kapcsolatot ehhez az URL objektumhoz. A válaszra várakozás időtartamának 15 másodpercet állítottam be. A függvény végén a kérésekben meghívott conn változót adom vissza. Az egyes kérésekben ezt a függvényt hívom meg. A valaszFeldolgozas() nevű függvény a visszaadott választ kezeli. A visszaadott HTTP státuszkódtól függően kezelem a választ.

A getRequest() nevű függvény az adatok lekérdezéséért, a postRequest() az adatok létrehozásáért, a putRequest() az adatok módosításáért, a deleteRequest() pedig a törlésért felel. Az alábbiakban a postRequest()-t részletezem.

```

public static Response postRequest(String url, String params) throws IOException {
    HttpURLConnection conn = kapcsolatBeallitas(url);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Content-Type", "application/json");
    OutputStream os = conn.getOutputStream();
    BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-
8"));
    writer.write(params);
    writer.flush();
    writer.close();
    os.close();

    return valaszFeldolgozas(conn);
}

```

A kérés végrehajtásához szükséges paraméterek az url és egy plusz paraméter. Itt megadtam a kapcsolatBeállítás() függvényben kezelt url-t. A setRequestMethod()-dal megadtam, hogy ez egy POST típusú kérés. A json típusú alakításhoz a setRequestProperty()-t használtam. A szervernek való elküldéshez az OutputStream-et, a paraméterek beírásához a BufferedWriter-t használtam. Visszatérési érték a valaszFeldolgozas() metódus válasza.

4.) Regisztracio.java

Ez a Fragment végzi a regisztrációt. Az aszinkron megvalósítást az AsyncTask generikus osztályból leszármaztatott RequestTask() végzi, aminek harmadik paramétere Response típusú. Ennek a doInBackground() metódusa adja meg, hogy mit csináljon a folyamat egy POST kérés esetén.

Az adatok beírása után a regisztráció gombjára kattintva egy új User objektum jön létre a validalEsUserLetrehoz() metódus segítségével. A validálást követően a függvény visszatér egy új User objektummal. A Google Gson könyvtárát használom az objektumról json-re alakításhoz. Létrehozok egy új RequestTask elemet, amiben az url-t és a POST kéréstípust megadva végrehajtódik a RequestHandler postRequest() metódusa. A hibaüzenet kiírását a HibaRunnalbe osztállyal kezelem. A sikeres regisztrációt követően a felhasználónak Toast üzenetben írom ki a szöveget.

5.) Bejelentkezes.java, Adatletolto.java és Adattarolo.java

A Bejelentkezes Fargment segítségével tud bejelentkezni a felhasználó. A gombra kattintva meghívódik az Adatletolto osztály bejelentkezes() metódusa. Az elvárt paraméterek a kontextus, a felhasználónév, a jelszó és a UserListener interfész, ami a kérés sikerességét adja

meg. A felhasználó beírt paramétereit egy POST kéréssel továbbítom a szerver felé. Amennyiben az API User controller bejelentkezés_post(), és User_model user_bejelentkezés() metódusaiban megírt feltételek teljesülnek, (létezik ilyen felhasználónév + a beírt jelszó az ehhez tartozó jelszó + a hitelesítés mező értéke true), úgy az API és az Adattartó osztály segítségével lemenésre kerül a bejelentkező felhasználó user_id-ja. A bejelentkezés sikerességéről vagy sikertelenségéről Toast üzenetben tájékoztatom a felhasználót, a bejelentkezést követően a BejelentkezettActivity jelenik meg.

6.) HirdetesFeladasa.java, HirdetesekKeresese.java, HirdetesekListazasa.java, Hirdetes.java, HirdetesApiValasz.java

HirdetesFeladasa Fragment segítségével tud feladni hirdetést a bejelentkezett felhasználó. Az összes adat kitöltése után a gombra kattintva megtörténik a validálás és új Hirdetes objektum létrehozása. A RequestTask osztály meghívásával a RequestHandler postRequest() metódusának meghívása. A feladás sikerességéről Toast üzenetben tájékoztatom a felhasználót.

A HirdetesekKeresese Fragment tartalmazza a szűrési feltételeket. Ezeknek megvalósítása nem jött létre, így itt a szűrés gombra kattintva töltődik be a HirdetesekListazasa Fragment.

A HirdetesekListazasa Fragment megjeleníti az összes adatbázisban szereplő hirdetést. A RequestTask doInBackground() metódusa meghívja a RequestHandler getRequest() metódusát. Az onPostExecute() metódusban a refreshHirdetesList() meghívásával betöltődnek a Hirdetes objektum adatai a listába. Az ArrayAdapter-ből leszármaztatott HirdetesAdapter visszaadott értéke a listItem, aminek segítségével láthatóak lesznek a lehívott hirdetések.

7.) Profil.java, ProfilModositas.java, User.java, UserApiValasz.java

A Profil Fragment jeleníti meg a bejelentkezett felhasználóhoz tartozó adatokat. A RequestTask az url, a lementett id és a GET típus paraméterekkel meghívja a RequestHandler getRequest() metódusát. A lekért adatokat listába rakja, amit a UserAdapter segítségével jelenít meg. A törlés gombra kattintva egy megerősítő Toast üzenetet követően a megadott paraméterek segítségével a RequestHandler deleteRequest() metódusa végzi a törlést. Törlést követően az alkalmazás kijelentkezteti a felhasználót, aki többé már nem tud bejelentkezni. A módosítás gombra kattintva a ProfilModositas oldal töltődik be.

A ProfilModositas Fragment userBetoltese() metódusa betölti a jelenlegi adatokat. A modosit() metódus a RequestTask RequestHandler putRequest() függvényét hívja meg. A módosítást követően a refreshUserList()-ben a listába a módosított user adatai kerülnek be. A

módosítást követően Toast üzenetben tájékoztatom a felhasználót, majd a mobilalkalmazás visszatér a profil oldalra a friss adatokat betöltve. A jelszómódosítás megvalósítása nem jött létre.

8.) Ranglista.java, Kategoria.java, KategoriaApiValasz.java

A Ranglista Fragment kilistázza az 5 legtöbb ponttal rendelkező felhasználót. Ennek megvalósítása nem történt meg. A statisztika részben az összes kategória neve és képe töltődik be. A kategoriakListazasa() metódus létrehoz egy új RequestTask-ot a megadott url és GET paraméterekkel, ami meghívja a RequestHandler getRequest() metódusát és visszaadja a választ. Az onPostExecute() a válasz alapján meghívja a refreshKategoriaList() metódust ami feltölti a listát, és létrehoz egy KategoriaAdapter-t, ami a megjelenítést végzi.

6.3.4 Tesztelési dokumentáció

Ebben a fejezetben szimultán mutatom be a fekete doboz tesztelési módszert a felhasználó szemszögéből és a fehér doboz tesztelési módszert a program kódjainak ismertetésével.

A regisztráció felületen az egyes mezők nem megfelelő kitöltése esetén a felhasználó segítő üzenetet kap a gomb alatti részben.

18. ábra: Regisztráció oldalon visszajelzés a felhasználónak

Forrás: Saját készítésű kép



A felhasználó a nem megfelelő kitöltés esetén az alábbi táblázatban ismertetett hibaüzeneteket kapja.

23. táblázat: Hibaüzenetek a regisztráció során

Forrás: Saját készítésű táblázat

Elem, amire vonatkozik	Üzenet
Bármelyik üresen hagyott elem esetén	Minden mező kitöltése kötelező.
Teljes név	Nem jól adta meg a nevét. (Elvart formátum: Teszt Elek)
Felhasználónév	A felhasználónév 6 és 100 karakter közötti lehet.
Telefonszám	A telefonszám 7 és 30 karakter közötti lehet.
E-mail cím	Nem megfelelő az e-mail cím formátuma. (Elvart formátum: example@example.com)

Elem, amire vonatkozik	Üzenet
Jelszó + Jelszó ismét	Nem ugyanaz a két jelszó.
Jelszó	A jelszó 6 és 100 karakter közötti lehet, tartalmaznia kell minimum egy számot is.
Cím	A cím 8 és 100 karakter közötti lehet
Felhasználói feltételek	Nem fogadta el a feltételeket.

A mezők validálására az alábbi metódusokat alkalmaztam:

- isEmpty(): Üres-e a vizsgált elem.
- matches(): A vizsgált elem megegyezik-e a megadott feltétellel.
- length(): A vizsgált elem hossza.
- equals(): A vizsgált elem megegyezik-e egy másik vizsgált elemmel.
- isChecked(): Ki van-e pipálva a checkbox

A név, jelszó és az e-mail cím esetében a vizsgált elemnek egy-egy regex feltétellel kell megegyeznie.

24. táblázat: Regex felételek

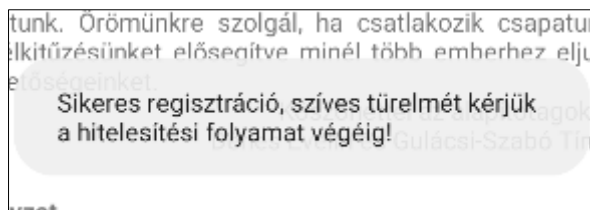
Forrás: Saját készítésű táblázat

Regex feltétel	Mit ellenőriz
[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}\$	Az e-mail cím lehet karakter vagy szám, de tartalmaznia kell egy '@' és egy '.' jelet.
(?=\d)(?=.*[A-Za-z]).{6,100}	A jelszónak tartalmaznia kell betűt és számot, 6 és 100 karakter közötti lehet.
^((Mr\. Dr\. dr\. Ms\. Mrs\.)\s)?([A-ZÁÉÍÓÖŐÚÜÚ][a-záéíóöőúüű]+([-][A-ZÁÉÍÓÖŐÚÜÚ][a-záéíóöőúüű]+){0,1})(\s[A-ZÉÍÓÖŐÚÜÚ][a-záéíóöőúüű]+(\s[A-ZÉÍÓÖŐÚÜÚ][a-záéíóöőúüű]+){0,1})	A névnek minimum 2 szóból kell állnia, nem tartalmazhat számot, minden kezdőbetűnek nagynak kell lennie, tartalmazhat név előtti címeteket.

Sikeres regisztrációt követően, amennyiben minden adatot megfelelően adott meg a felhasználó, az alábbi Toast üzenet jelenik meg.

19. ábra: Üzenet sikeres regisztráció esetén

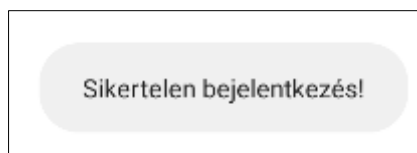
Forrás: Saját készítésű kép



A bejelentkezési felületen a felhasználó által nem jól megadott adatok, illetve a hitelesítés végrehajtása előtt Toast üzenet jelenik meg.

20. ábra: Üzenet sikertelen bejelentkezés esetén

Forrás: Saját készítésű kép

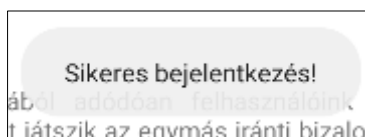


A `User_model user_bejelentkezes()` metódusát használó User API kontroller `bejelentkezes_post()` metódusa végzi el a validálást, az `Adatletolto` és `Bejelentkezes` osztályok végzik a megjelenítést.

Amennyiben az `Adatletolto bejelentkezes()` metódusának értéke igaz, úgy az alábbi visszaigazoló Toast üzenetet kapja a felhasználó.

21. ábra: Üzenet sikeres bejelentkezést követően

Forrás: Saját készítésű kép



Hirdetés feladása esetén a már korábban kifejtett hibaüzeneteket kapja a felhasználó a validálás miatt kitöltetlen mező, rosszul megadott telefonszám és cím esetén.

22. ábra: Hirdetés oldalon visszajelzés a felhasználónak

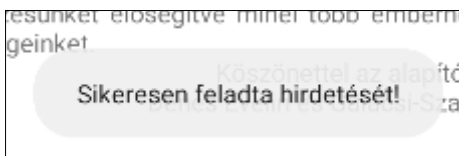
Forrás: Saját készítésű kép



A megfelelően megadott adatok esetén visszaigazoló Toast üzenetet kap a felhasználó.

23. ábra: Üzenet sikeres hirdetésfeladást követően

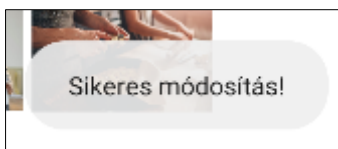
Forrás: Saját készítésű kép



Profilmódosítás esetén is, a már korábban részletezett validációt hajtjuk végre kitöltetlen mező, telefonszám, e-mail cím és cím esetén. A felhasználó a korábban bemutatott visszajelzéseket kapja. Sikeres módosítás esetén is kap visszajelzést a felhasználó.

24. ábra: Üzenet profilmódosítást követően

Forrás: Saját készítésű kép



Törlés esetén egy AlertDialog várja a megerősítést a felhasználótól.

25. ábra: Megerősítés törlés esetén

Forrás: Saját készítésű kép



Amennyiben bármilyen egyéb probléma lép fel (pl. csatlakozási probléma vagy már regisztrált felhasználónév) a felhasználó nem kap róla értesítést.

6.3.5 Továbbfejlesztési lehetőségek

A közös backend részben már kifejtettük a program továbbfejlesztési lehetőségeit, így ebben a részben azokat a dolgokat sorolom fel, amit szerettem volna, de nem sikerült megvalósítani.

- Regisztráció, Hirdetésfeladás és Profil módosítás esetén a település jelenleg egy szám beírásával adható meg. Célszerű lenne Spinner vezérlőbe betölteni a településneveket.
- A profilkép és az okmánykép jelenleg egy szöveget váró mező. Olyan funkcióval kell ezt megvalósítani, ahol a felhasználó képeket választhat ki a saját eszközéről.
- A születési dátum és az okmányszám validálása.
- Bejelentkezéskor más hibaüzenet küldése abban az esetben, ha a hitelesítés miatt nem tud még bejelentkezni a felhasználó.
- Hirdetések keresésénél a szűrő lehetőségek beállítása, a kiválasztott paraméterek alapján listázza ki a hirdetéseket.
- Konkrét hirdetés esetén a hirdetés_id megjelenítése helyett a hirdetést feladó személy neve szerepeljen.
- Konkrét hirdetés, Profil és Profilmódosítás esetén a telepules_id helyett a településnév szerepeljen.
- Konkrét hirdetésnél a categoria_id helyett a kategória neve szerepeljen.
- Jelentkezés gomb aktívva tétele a hirdetésre jelentkezése esetén.
- Minden hibaüzenetről visszajelzés a felhasználónak.
- A profil adatainak betöltése ne listview-n keresztül történjen.
- Felesleges kódismétlések megszüntetése.
- A Profil és a Konkrét hirdetés oldalain a felhasználó képének betöltése.
- A felhasználó díjainak megjelenítése
- A profil oldalon a hirdetések valós betöltése, törlés, elfogadás és elutasítás gomb aktiválása. Az egyes hirdetésképekre kattintva jelenjen meg az adott hirdetés konkrét adatlapja.
- Jelszó módosítás lehetővé tétele.

Fejlesztői dokumentáció

- Ranglista oldalon a top 5 felhasználó betöltése, a kategóriákhoz tartozó %-os adatok megjelenítése.
- Az alkalmazás kinézetének esztétikusabbá tétele, Lottie animáció használata.
- Az API kontrollerek minden metódusának visszatérési értéke tartalmazzon error és message változókat.
- Minden esetben backend és frontend validálás megvalósítása.
- Biztonságosabb jelszótárolás alkalmazása.

7. Felhasználói dokumentáció

7.1 Webalkalmazás

Ebben a fejezetben az alkalmazás felhasználók számára készített leírása található. A program működését és a megvalósított funkciókat a szemléletesebb ismertetés érdekében számos helyen ábrákkal illusztrálom.

7.1.1 A program általános specifikációja

A webalkalmazás egy önkéntes portál, amelyen regisztrációt követően önkéntes munkák feladása és a feladott hirdetésekre való jelentkezés valósítható meg. A programot használó személyek előre meghatározott szűrési feltételek megadása mellett kereshetnek a saját preferenciáiknak megfelelő önkéntes munkát, megtekinthetik a kiválasztott hirdetés adatlapját, és jelentkezhetnek rá. A jelentkezést követően a hirdetést feladó személy elfogadhatja vagy elutasíthatja az önkéntes jelentkezését. A felhasználók minden egyes elvégzett önkéntes munkát követően kapnak egy pontot, valamint személyes profiljukon is megjelenik az elvégzett hirdetés kategóriaképe. A pontszám alapján a felhasználók között rangsor alakul ki és az első öt legmagasabb pontszámmal rendelkező felhasználó, valamint a hirdetések kategória szerinti százalékos eloszlása megtekinthető a weboldal Ranglista menüpontjában. A Profil menüpontban a bejelentkezett felhasználó megtekintheti személyes adatait, az általa feladott és a jóváhagyásra váró hirdetéseket, valamint azokat is, amelyekre ő jelentkezett. Minden felhasználónak lehetősége nyílik módosítani személyes adatainak egy részét, valamint jelszavát. A feladott hirdetések törölhetők, illetve amennyiben a felhasználó nem kívánja regisztrált fiókját megtartani, módjában áll törölnie teljes személyes profilját.

7.1.2 Rendszerkövetelmények

A webalkalmazás, amennyiben nem localhoston történt volna a fejlesztés, minden eszközről - számítógép, laptop, táblagép, mobiltelefon - és az összes internetes böngészőben használható lenne (pl.: www.onkentesportal.hu domain néven), operációs rendszertől függetlenül, megfelelő internetkapcsolat mellett. Mivel jelen program csak a XAMPP szoftver telepítését és elindítását követően működik, hiszen az adatbázis és a teljes program forráskódját tartalmazó mappa lokálisan helyezkedik el, a program indításához szükséges beállításokat a következő fejezetben ismertetem.

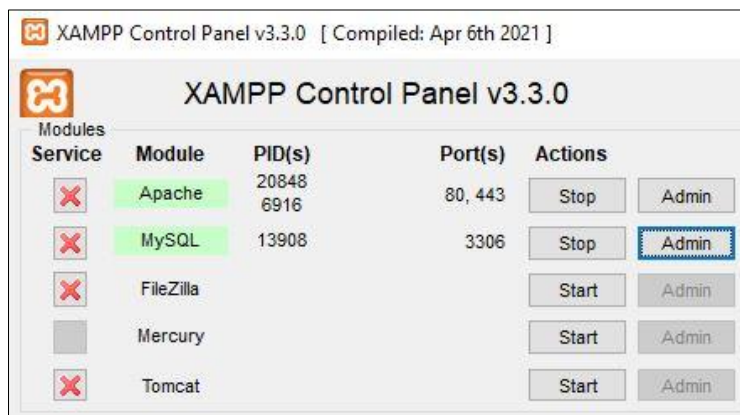
7.1.3 A program telepítésének és konfigurálásának leírása

A letöltött program forráskódját tartalmazó mappát a XAMPP szoftver telepítését követően a Helyi lemez (C:) xampp/htdocs mappájába kell bemásolni. Ezt követően el kell

indítani a XAMPP Apache és MySQL modulját a modulok mellett található Start gomb megnyomásával. A helyes működést az alábbi ábra szemlélteti:

26. ábra: Lokális webszerver elindítása

Forrás: XAMPP Control Panel v3.3.0



Az 'Admin' gombra kattintva a phpMyAdmin felületén az Importálás gombra kattintva a Tallózás opciót választva a xampp/htdocs mappába lementett program Vizsgaremek_Web mappájában található, karitativ.sql elnevezésű, az adatbázis exportját tartalmazó fájlt szükséges kiválasztani, majd az Indítás gombra kattintani. Ezt követően a számítógépen elérhető internetes böngészőben a http://localhost/Vizsgaremek_Web URL megadása szükséges. Fentiek sikeres elvégzését követően a webalkalmazás készen áll a használatra. Az alkalmazás nyújtotta funkciókról és a hozzájuk tartozó felületekről a következő fejezetben található útmutatás.

7.1.4 A program használatának részletes leírása

7.1.4.1 Regisztráció

A program funkciói regisztrációt követően érhetőek el, így először a Regisztráció menüpontot kiválasztva a megjelenő regisztrációs űrlapon található minden beviteli mező helyes kitöltését szükséges végrehajtani, majd a végén a jelölőnégyzetet kipipálni és rákattintani a 'Regisztráció' gombra. Amennyiben a jelölőnégyzetbe nem kerül pipa, az alábbi hibaüzenet jelenik meg.

27. ábra: Regisztráció hibaüzenet

Forrás: Saját készítésű ábra

The Elfogadom a Felhasználói feltételeket és az Adatvédelmi irányelveket. field is required.

28. ábra: Regisztráció menüpont

Forrás: Saját készítésű ábra

The screenshot shows a registration form with a red header bar containing the navigation links: Főoldal, Bejelentkezés, and Regisztráció. The form fields are as follows:

- Teljes név: [Text input field]
- Felhasználónév: [Text input field]
- Születési dátum: [Date picker showing 'éé.éé. hh. nn.']
- Telefonszám: [Text input field]
- E-mail cím: [Text input field]
- Település: [Dropdown menu showing 'Szentendre']
- Cím: [Text input field with placeholder 'Írányítószám, utca, házszám, emelet, ajtó']
- Feltöltött okmány száma: [Text input field]
- Okmánykép feltöltése: [File upload button 'Tallózás...' and text 'Nincs kijelölve fájl']
- Profilkép feltöltése: [File upload button 'Tallózás...' and text 'Nincs kijelölve fájl']
- Jelszó: [Text input field]
- Jelszó megerősítése: [Text input field]
- ☐ Elfogadom a Felhasználói feltételeket és az Adatvédelmi irányelveket.
- [Yellow button labeled 'Regisztráció']

Beviteli mezőkre vonatkozó szabályok:

Teljes név: Vezetéknév és keresztnév megadása nagy kezdőbetűkkel és szóközzel elválasztva. Nem tartalmazhat számot.

Felhasználónév: Egyedi, így amennyiben korábban már másik felhasználó regisztrált a megadott felhasználónévvel, akkor hibaüzenet érkezik. Minimum 6 karakterből áll.

29. ábra: Felhasználónév hiba

Forrás: Saját készítésű ábra



Felhasználónév:

aaaaa

A megadott felhasználónév nem lehet 6 karakternél rövidebb.

Telefonszám: Minimum 7 karakterből áll. (következtesen számok, esetleg + karakter megadása célszerű)

E-mail cím: Valós e-mail cím megadása szükséges, mely megfelel az e-mail cím formátumnak, azaz tartalmazzon @ és . karaktert is.

Település: Az előre betöltött településnevek közül kell kiválasztani egyet. Amennyiben a mezőben a település nevét elkezdi a felhasználó begépelni, az adott településnév megjelenik és kiválasztható.

Cím: Minimum 8 karakterből áll.

Feltöltött okmány száma: A feltöltésre kerülő okmányképen szereplő karaktersorozat (számok és betűk) pontos megadása.

Okmánykép feltöltése: A Tallózás gombra kattintva a személyigazolványt szükséges feltölteni .jpg, .png, .jpeg vagy .bmp formátumban.

Profilkép feltöltése: A Tallózás gombra kattintva az alkalmazásban megjeleníteni kívánt profilképet szükséges feltölteni .jpg, .png, .jpeg vagy .bmp formátumban.

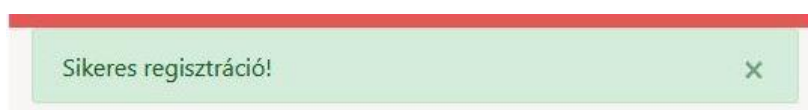
Jelszó: Minimum 6 karakterből áll, és legalább egy számot tartalmaznia kell. Speciális karakterek használata nem engedélyezett, csak betűk és számok kombinációja adható meg.

Jelszó megerősítése: A jelszó mezőben megadott jelszó ismételt megadása szükséges.

Sikeres regisztrációt követően az alábbi visszajelzés látható:

30. ábra: Sikeres regisztráció

Forrás: Saját készítésű ábra




Sikertelen regisztráció esetén fontos megjegyezni, hogy minden esetben **újból** szükséges kiválasztani a megfelelő települést, valamint feltölteni az okmány és profilképeket.

7.1.4.2 Bejelentkezés

Sikeres regisztrációt követően a regisztráció során megadott adatok hitelesítési folyamaton esnek át, így a felhasználó nem tud azonnal bejelentkezni a felületen.

31. ábra: Bejelentkezés menüpont

Forrás: Saját készítésű ábra



The screenshot shows a web interface with a red header bar containing the navigation links: **Főoldal**, **Bejelentkezés**, and **Regisztráció**. Below the header, there is a light gray box containing the login form. The form has two input fields: 'Felhasználónév:' and 'Jelszó:'. Below these fields is a yellow button labeled 'Bejelentkezés'.

Arról, hogy az adott felhasználóra vonatkozó hitelesítési eljárás még nem zárult le, az alábbi visszajelzés ad tájékoztatást:

32. ábra: Bejelentkezési hibaüzenet

Forrás: Saját készítésű ábra




The screenshot shows the same web interface as before, but with a red error message displayed below the navigation bar: 'Sajnos még nem tud bejelentkezni, mert nem történt meg a regisztrációt követő hitelesítés!'.

Amennyiben nem megfelelő felhasználónév és jelszó párost ad meg a felhasználó, az alábbi hibaüzenetet kapja:

33. ábra: Bejelentkezési hiba

Forrás: Saját készítésű ábra



The screenshot shows the same web interface as before, but with a red error message displayed below the navigation bar: 'Hibás felhasználónév vagy jelszó!'.

7.1.4.3 Hirdetések keresése

Sikeres bejelentkezést követően az oldal a felhasználót a Hirdetések keresése menüpontra irányítja. A szűrési feltételek - kategória, település és dátum - mindegyikét ki kell tölteni a felhasználó preferenciáinak megfelelően. A 'Szűrés' gombra kattintással elindítható a hirdetések közötti keresés. Az oldal csak akkor ad keresési találatot, amennyiben az adott hirdetés az összes kritériumnak egyszerre felel meg. Előfordulhat, hogy egyetlen hirdetés sem felel meg a megadott kritériumoknak, ebben az esetben az alábbi szöveg jelenik meg: Nincs találat a megadott paraméterek alapján. Az alábbi ábrán egy olyan eset látható amikor egyetlen hirdetés jelent meg a megadott feltételek alapján.

34. ábra: Keresési találatok

Forrás: Saját készítésű ábra

A Szűrés elindítása előtt megadott keresési feltételeket minden egyes esetben **újból** meg kell adni, mert a beviteli mezők alapértelmezett értékeket vesznek fel. A 'Tovább a hirdetésre' gombra kattintva megtekinthető az adott hirdetés adatlapja, mely a hirdetésre vonatkozó információkat tartalmazza. A hirdetés adatlapján a 'Jelentkezem' gombra kattintva lehetséges jelentkezni egy adott önkéntes munkára. Ekkor a hirdetést feladó személynél a hirdetés bekerül a jóváhagyásra váró hirdetések közé, ezt követően elfogadhatja vagy elutasíthatja a jelentkezést. Ennek menetéről a következő fejezetben lesz szó részletesebben. A hirdetés adatlapjáról a

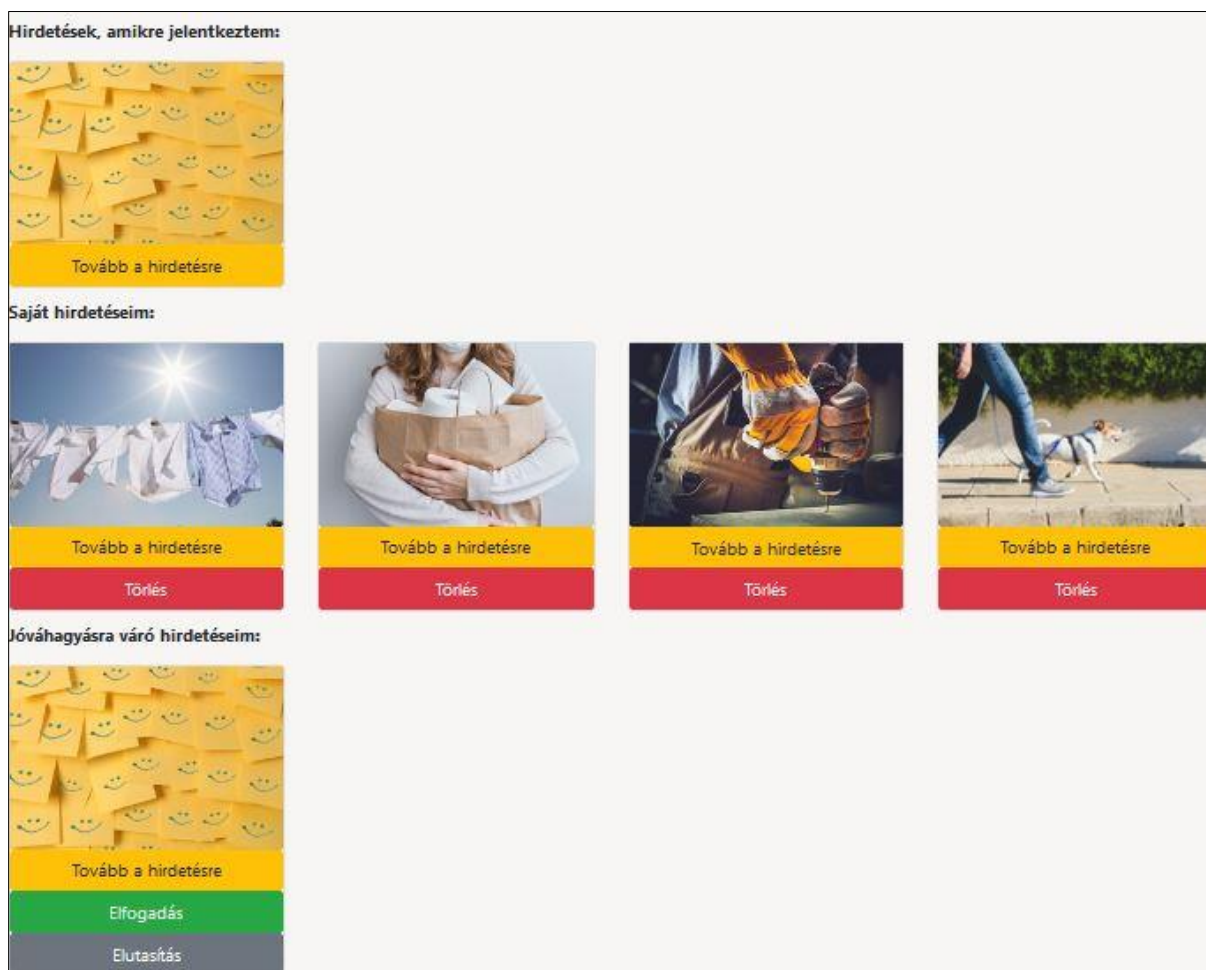
keresési felületre történő visszajutáshoz az Alt + Balra (Ugrás az előző oldalra) gombok kétszeri megnyomása szükséges.

7.1.4.4 Profil

A Profil menüpontban a felhasználó megtekintheti a személyes adatait, valamint a 'Díjaim' felirat alatt láthatja azon kategóriaképeket felsorolva, amely kategóriákból már legalább egy önkéntes munkát teljesített. Ezenkívül kilistázásra kerülnek azon hirdetések, amikre a felhasználó jelentkezett, valamint az általa feladott és a jóváhagyására váró hirdetések is.

35. ábra: Profil menüpont - Hirdetések

Forrás: Saját készítésű ábra



A felhasználó a 'Tovább a hirdetésre' gombra kattintva az adott hirdetés adatlapjára jut, ahol minden esetben megtalálható a Jelentkezem gomb. Saját hirdetésre, valamint olyan hirdetésre, amire már korábban egy másik felhasználó jelentkezett (jóváhagyásra vár/teljesített) a felhasználónak nincsen lehetősége jelentkezni. Mind a kettő esetben hibaüzenet kerül kiírásra, az alábbi szövegekkel:

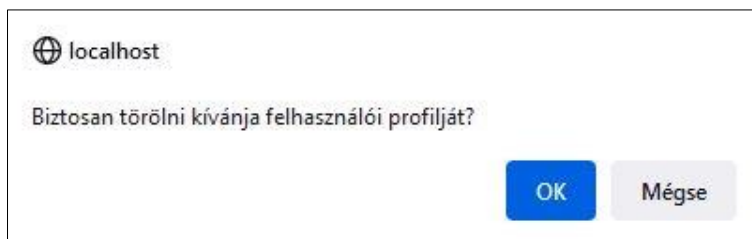
- Saját hirdetésre nem jelentkezhet!
- Erre a hirdetésre már jelentkeztek!

Jóváhagyásra váró hirdetések esetében az Elfogadás gomb megnyomásával elfogadásra kerül az önkéntes jelentkezése, és az adott önkéntes pontszáma eggyel nő. A hirdetés többé nem jelenik meg a profil oldalon. Ezzel szemben az elutasítás gomb megnyomásával a hirdetés visszakerül a saját hirdetések közé, és újra jelentkezhet rá másik felhasználó. A hirdetésre jelentkező felhasználó esetében pedig a profil oldalon az elutasítást követően a hirdetés többé nem kerül kilistázásra a 'Hirdetések, amikre jelentkeztem' felirat alatt.

A profil oldalon lehetőség nyílik saját hirdetések törlésére a 'Törlés' gomb megnyomásával. Ekkor egy felugró ablakban a törlési szándék az 'OK' megnyomásával erősíthető meg, illetve a 'Mégse' gombbal vonható vissza. Amennyiben a felhasználó teljes fiókját törölni kívánja, a 'Profil törlése' gombra kattintva, valamint a felugró ablakban az 'OK' opciót választva megteheti azt.

36. ábra: Profil törlése

Forrás: Saját készítésű ábra



A regisztrációkor megadott személyes adatok és jelszó megváltoztatására is lehetőség van a profil oldalon elhelyezett 'Profil módosítása' gombra kattintva. A személyes adatok módosításához szükséges űrlapon a felhasználó regisztrációkor megadott adatai találhatók, **kivéve** a Település mező esetében, ahol **ismételten ki kell választani** a megfelelő települést, amennyiben ténylegesen adاتمódosításra kerül sor (gomb megnyomásra kerül). Minden mező kitöltése kötelező, és a beviteli mezőkbe a regisztrációnál már kifejtett szabályok szerint adhatóak meg az adatok. A jelszó módosításához mindenképpen meg kell adni a jelenlegi jelszót is, valamint az új jelszavak beírásakor a két beviteli mezőbe ugyanaz a tartalom kerüljön. Sikeres módosítások esetén az alábbi visszajelzések láthatóak:

- Sikeresen módosította profilját!
- Sikeresen módosította jelszavát!

Az alábbi ábrán a módosításokhoz szükséges űrlapok kinézete látható.

37. ábra: Profil és jelszó módosítása

Forrás: Saját készítésű ábra

The image shows a web form with two main sections. The top section, titled 'Profil módosítása:', contains four input fields: 'Település:' with a dropdown menu showing 'Szentendre', 'Cím:' with the text '2000, Tészt utca 10.', 'E-mail cím:' with 'tesztelek@gmail.com', and 'Telefonszám:' with '06203412347'. Below these fields is a yellow button labeled 'Profil módosítása'. The bottom section, titled 'Jelszó módosítása:', contains three input fields: 'Jelenlegi jelszó:', 'Új jelszó:', and 'Új jelszó újra:'. Below these fields is a yellow button labeled 'Jelszó módosítása'.

7.1.4.5 Hirdetés feladása

A 'Hirdetés feladása' menüpontra kattintva válik elérhetővé az új hirdetés feladására szolgáló űrlap. A regisztrációs űrlaphoz hasonlóan, itt is minden beviteli mező kitöltése kötelező.

Kezdő időpont: Az önkéntes munka kezdeti időpontját kell megadni a beviteli mezőben jelzett formátumban (pl.: 2022.09.21. 09:00).

Záró időpont: Az önkéntes munka záró időpontját kell megadni a beviteli mezőben jelzett formátumban (pl.: 2022.09.21. 11:00).

Kategória: A hirdetéshez leginkább illő kategória kiválasztása.

Leírás: Az önkéntes munka szöveges ismertetése. Tartalmaz minden olyan információt, mely fontos lehet a munkára jelentkező önkéntes számára.

Település: Önkéntes munka helyszínének megadása (település). Az előre betöltött településnevek közül kell kiválasztani egyet. Amennyiben a mezőben a település nevét elkezdí a felhasználó begépelni, az adott településnév megjelenik és kiválasztható.

Cím: Önkéntes munka helyszínének megadása (cím). Minimum 8 karakterből áll.

Telefonszám: Hirdetéssel kapcsolatban hívható telefonszám megadása. Minimum 7 karakterből áll. (következetesen számok, esetleg + karakter megadása célszerű)

38. ábra: Hirdetés feladási űrlap

Forrás: Saját készítésű ábra

Új hirdetés adatai:

Kezdő időpont:
éééé. hh. nn. --:--

Záró időpont:
éééé. hh. nn. --:--

Kategória:
Kutyasétáltatás

Leírás:

Település:
Szentendre

Cím:

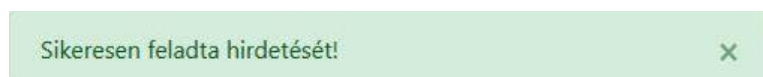
Telefonszám:

Hirdetés feladása

A 'Hirdetés feladása' gombra kattintva feladható a hirdetés. A feladott hirdetés feladást követően megtekinthető a profil oldalon a saját hirdetések listájában. A sikeres rögzítést az alábbi üzenet jelzi:

39. ábra: Sikeres hirdetés feladás

Forrás: Saját készítésű ábra

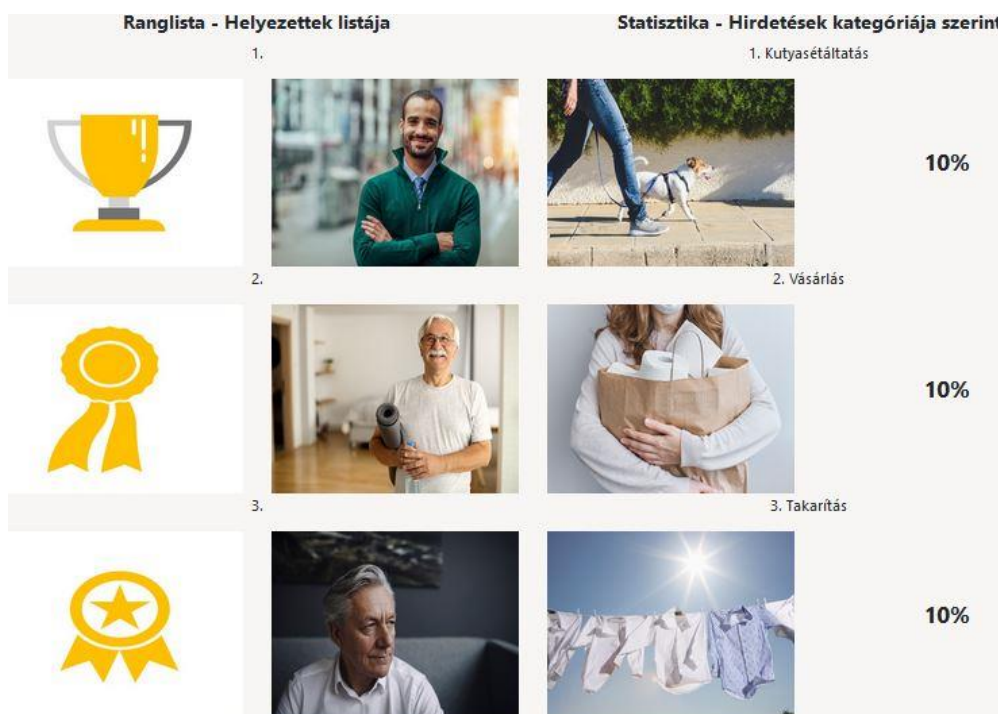


7.1.4.6 Ranglista

Minden teljesített önkéntes munka elvégzésével az adott felhasználó egy pontot szerez. A pontszám alapján kialakul egy rangsor a felhasználók között, amely megtekinthető a Ranglista menüpontra kattintva. A Helyezettek listájában az első öt legmagasabb pontszámmal rendelkező felhasználó regisztrációkor feltöltött profilképe tekinthető meg. A 'Statisztika - Hirdetések kategóriája szerint' oszlopban a 12 darab hirdetéskategória képe, valamint az egyes kategóriák alkalmazásban feladott összes hirdetéshez képest meghatározott %-os aránya látható. A feltüntetett adatokból a hirdetések böngészése nélkül is látszódik, hogy melyik kategóriákból adták fel a legtöbb hirdetést, illetve melyik az a kategória, amelyikből esetleg egyetlen hirdetés sem található meg a weboldalon.

40. ábra: Ranglista menüpont

Forrás: Saját készítésű ábra



7.2 Mobilalkalmazás

7.2.1 A program általános specifikációja

Az önkéntes portál mobilapplikáció azoknak a felhasználóknak készült, akik valamilyen nehézség miatt segítségre szorulnak, illetve azoknak, akik segítséget szeretnének nyújtani embertársaiknak.

A regisztrációt követően a biztonságos használat érdekében egy hitelesítési folyamaton mennek keresztül a megadott személyes adatok. A sikeres hitelesítést követően a felhasználónak lehetősége van feladni saját hirdetését, melyben pontosan megadhatja, hogy mikor, hol, milyen segítségre van szüksége. Amennyiben valaki önkéntes munkát szeretne végezni, úgy a feladott hirdetések között keresgélve választhatja ki a számára megfelelő az időpontokat, várost és a leírást megtekintve.

A fő funkciója az alkalmazásnak a hirdetések menedzselése, de ezen kívül létrejött még egy profil oldal a személyes adatok megtekintésére, módosítására, a fiók a törlésére. A Ranglista oldal célja, hogy motiválja a regisztrálókat minél több és különféle kategóriájú munka elvégzésére.

7.2.2 Rendszerkövetelmények

Amennyiben a fejlesztés nem localhost-on valósult volna meg, úgy a mobilapplikáció a Google Play Áruházból letöltve azonnal használható lenne a mobil eszközök 96,2 %-án. Jelenleg a program futtatásához szükséges a XAMPP szerver és az Android Studio program is. A szerver telepítése a fentiekben már kifejtésre került, így én az Android Studio rendszerkövetelményét és telepítését fogom részletezni.

A program letölthető Microsoft Windows, macOS és Linux operációs rendszerekre is. A minimum rendszerkövetelmények a futtatáshoz Windows-on:

- Operációs rendszer: Windows 8/8.1/10/11 (64-bit)
- Processzor: Második generációs Intel, AMD processzor Windows Hypervisor támogatással
- RAM: 8 GB
- Lemezterület: 8 GB
- Képernyő felbontás: 1280 x 800

[<https://developer.android.com/studio>]

7.2.3 A program telepítésének és konfigurálásának leírása

Az Android Studio telepítése előtt le kell tölteni a Java JDK legutolsó verzióját az www.oracle.com oldaláról. A saját rendszerünknek megfelelően kiválaszthatjuk a megfelelő linket. Erre kattintva a letöltés után egyszerűen a Next gombokra kattintva telepíthetjük is. A telepítés után bezárhatjuk az ablakot.

Az Android Studiot az android hivatalos oldaláról tudjuk letölteni www.developer.android.com. Itt is a saját rendszerünknek megfelelő linkre szükséges kattintanunk és a felhasználói feltételek elfogadása után megtörténik a letöltés. A telepítést itt is a Next és az Install gombokra kattintva végezhetjük el. Telepítés után, az elindítást követően a Next gombok segítségével, a standard opciót kiválasztva állathatjuk be a programot. Választhatunk világos vagy sötét témát. A kiválasztott beállításokat a Finish gomb megnyomásával tölti le a program.

A megnyitást követően célszerű beállítanunk néhány dolgot. A Tools-SDK Manager menüpont SDK Platforms fülén érdemes letöltenünk az Android 10.0 (Q) vagy 11.0 (R) verziókat, ugyanis ezek a legstabilabbak. A futtatáshoz szükségünk van egy emulátorra, amit a AVD Manager felületén állíthatunk be. Itt a Create Virtual Device gomba kattintva beállíthatjuk a virtuális eszközt. Javasolt a Pixel 2 Phone és Q operációs rendszer, hiszen én is ezeket használtam a létrehozáskor. Ezeket a Next, Download és Finish gombokkal könnyedén megadhatjuk.

7.2.4 A program használatának részletes leírása

Az alábbiakban az android alkalmazás egyes oldalait és funkcióit mutatom be képekkel illusztrálva.

7.2.4.1 Fő oldalak és menüpontok

Az applikáció megnyitását követően a fő oldalt láthatjuk. Ezen szerepel egy bejelentkezés és regisztráció gomb a gyorsabb elérés érdekében, de ugyanezeket a menüből is elérhetjük. A Bejelentkezés gombra vagy a menüben a Bejelentkezés menüpontra kattintva a bejelentkezésért, a Regisztráció gombra vagy a menüben a Regisztráció menüpontra kattintva a regisztrációért felelős oldal jelenik meg. A fő oldalon szerepel egy rövid bemutatkozás, a szabályzat ismertetése és az elérhetőségek.

41. ábra: Főoldal

Forrás: Saját készítésű kép



7.2.4.2 Regisztráció

Ezen az oldalon szükséges kitölteni a regisztrációhoz szükséges személyes adatokat. Az adatok megadását segítő feliratok mutatják. Minden mező kitöltése kötelező a megfelelő formátumokkal. Erre az esetleges hibaüzenetek is figyelmeztetnek. A feltételek elfogadását követően, amennyiben mindent megfelelően töltöttünk ki, a regisztráció gombra kattintva megtörténik a regisztráció. Ekkor egy üzenetet is kapunk, ami türelmet kér a hitelesítés végrehajtásáig.

42. ábra: Regisztráció

Forrás: Saját készítésű kép

Önkéntes Portál	
Felhasználónév	
Születési dátum	
Telefonszám	
E-mail cím	
Település ID	
Cím	
Okmányszám	
Okmánykép neve	
Profilkép neve	
Jelszó	
Jelszó ismét	
<input type="checkbox"/>	Elfogadom a szerződési feltételeket és az Adatvédelmi irányelveket.
REGISZTRÁCIÓ	

7.2.4.3 Bejelentkezés

A helyes felhasználónév és jelszó beírása után, amennyiben a regisztráció végbement a hitelesítés folyamatán, a bejelentkezés gombra kattintva tudunk bejelentkezni az alkalmazásba. A bejelentkezést követően ismételtlen a főoldal nyílik meg, de itt már nem szerepelnek a regisztrációhoz és bejelentkezéshez szükséges gombok. A menürendszer is megváltozik, bejelentkezett állapotban más menüpontokat érhetünk el: Főoldal, Hirdetések keresése, Hirdetés feladása, Profil, Ranglista, Kijelentkezés.

43. ábra: Bejelentkezés

Forrás: Saját készítésű kép



Önkéntes Portál

Bejelentkezés

Felhasználónév

Jelszó

BEJELENTKEZÉS

7.2.4.4 Hirdetés feladása

Hirdetés feladásakor célszerű minél pontosabban megadnunk az adatokat és megfogalmazni a kérésünket. Minden mező kitöltése kötelező a megfelelő formátumban, erre itt is esetleges segítőüzenetek figyelmeztetnek. A kezdő és záró időpontban az elvégzendő munka napját és körülbelüli óráit adhatjuk meg. A kategóriában kiválaszthatjuk, hogy milyen típusú segítségre van szükségünk. A kategóriát egy legörülő listából választhatjuk ki. A helyes kitöltés után a mentés gombra kattintva egy visszaigazoló üzenet jelzi a feladás sikerességét.

44. ábra: Hirdetés feladása

Forrás: Saját készítésű kép

The screenshot shows a web form titled 'Önkéntes Portál' (Volunteer Portal) with a red header bar. The form contains several input fields and a submit button. At the top, there are two date fields: 'Kezdő időpont' (Start date) and 'Záró időpont' (End date). Below these is a 'Kategória' (Category) dropdown menu currently set to 'Kutyasétáltatás' (Dog walking). Underneath is a 'Település' (City) field. The form continues with a 'Cím' (Title) field, a 'Telefonszám' (Phone number) field, and a large 'Leírás' (Description) text area. At the bottom right of the form is a red button labeled 'MENTÉS' (Save).

7.2.4.5 Hirdetések keresése

A hirdetések keresése oldalon kategória, település és dátum szerinti szűrési feltételek szerepelnek. Ennek a megvalósítása még fejlesztés alatt áll, így a szűrés gombra kattintva az összes feladott hirdetés megjelenik.

45. ábra: Hirdetések listázása

Forrás: Saját készítésű kép



A hirdetés kategóriájának képe, a hirdetés leírása és egy tovább gomb látható. A tovább gombra kattintva megjelennek az adott hirdetéshez tartozó adatok. A vissza a hirdetésekre gombbal visszatérhetünk a korábbi listához. A jelentkezés gomb egyelőre még nem aktív. Ezzel a gombbal lehet majd jelentkezni az adott hirdetésre.

46. ábra: Konkrét hirdetés

Forrás: Saját készítésű kép

 Önkéntes Portál



8
2108
8000, István tér 9.
06708916734

11

Időpont kezdete: 2022-09-10 14:00:00
Időpont vége: 2022-09-10 16:00:00

Egyedül élő idős emberként néhány óra segítséget keresek kerti munkák, így például fűnyírás, levélgereblyezés elvégzésére.

JELENTKEZEM

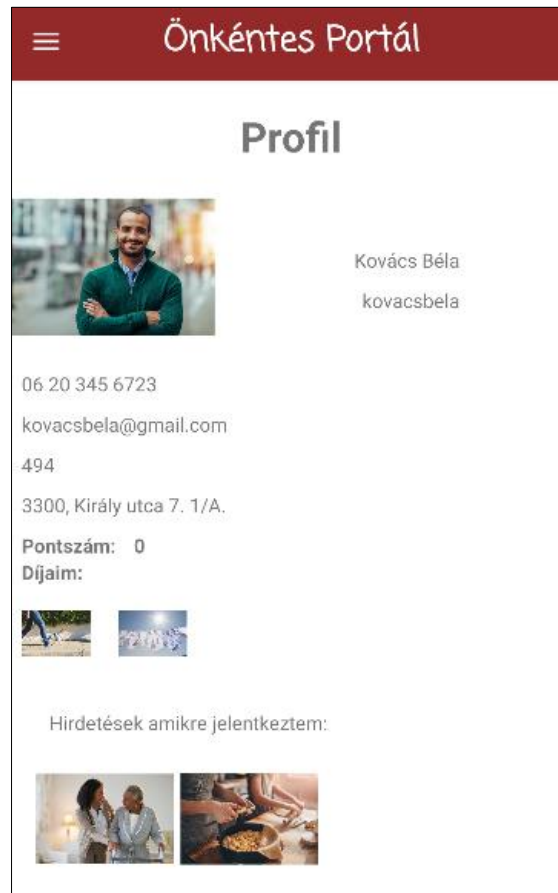
VISSZA A HIRDETÉSEKRE

7.2.4.6 Profil

A profil oldalon láthatjuk a bejelentkezett felhasználó adatait és hirdetéseit. A hirdetések betöltése, törlése, elfogadása és elutasítása még nem valósult meg.

47. ábra: Profil oldal I.

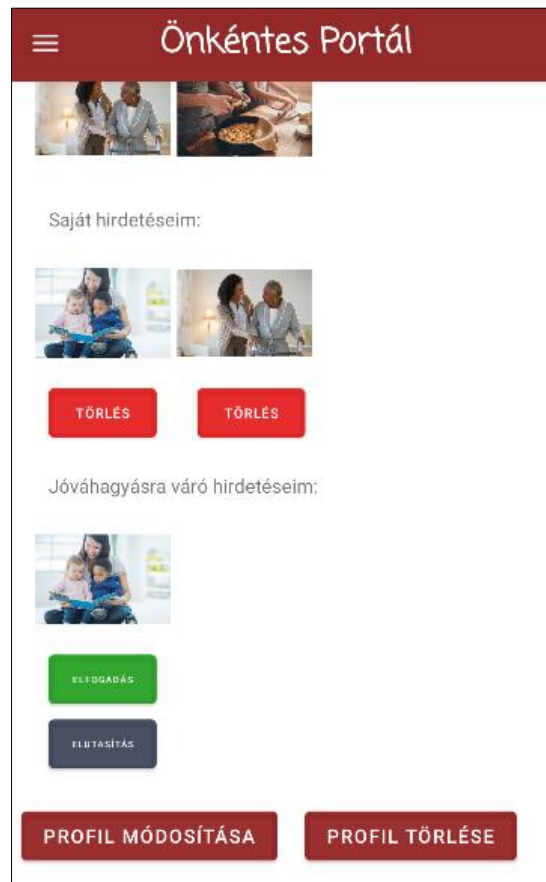
Forrás: Saját készítésű kép



Az oldal alján szereplő profil törlése gombra kattintva tudjuk törölni a felhasználói profilunkat. A felugró ablak NEM gombjára kattintva nem történik semmi, az IGEN gombra kattintva töröljük az oldalunkat és kiléptet a program. A profil módosítása gombra kattintva a módosítási felület nyílik meg.

48. ábra: Profil oldal II.

Forrás: Saját készítésű kép



A betöltött adatokat a formátumnak megfelelően átírva, és a profil módosítás gombra kattintva az applikáció visszaléptet a profil oldalra és megjeleníti a módosított adatokat. A sikeres módosításról visszajelzést is kapunk. A jelszó módosításra egyelőre még nincs lehetőség. A vissza a profilra gombbal is visszatérünk a profilra, ebben az esetben nem történik módosítás.

49. ábra: Profil módosítás

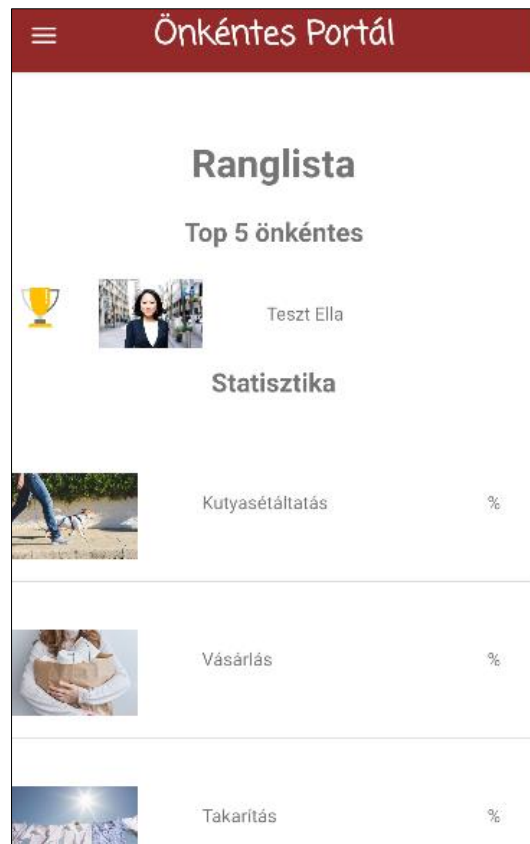
Forrás: Saját készítésű kép

7.2.4.7 Ranglista

A ranglista oldalon tekintheti meg a felhasználó az 5 legtöbb pontszámmal rendelkező önkéntes profilképét és nevét. A statisztika részben a kategóriaképek, a kategóriák neve és egy %-os érték látható, ami az adott kategóriát tartalmazó elvégzett hirdetések mértékét jelöli. Az itt megjelenített funkciók részben működőképesek.

50. ábra: Ranglista

Forrás: Saját készítésű kép



Az utolsó menüpont a kijelentkezés, amire kattintva kijelentkezhetünk az alkalmazásból.

8. Összegzés

8.1 Megvalósítás

A megvalósított funkciókat a Fejlesztési és Felhasználói dokumentáció fejezetekben részletesen kifejtettük, a hibás és hiányos működést pedig a Tesztelési dokumentációban és a Továbbfejlesztési lehetőségeknél taglaltuk. Az előzetesen kitűzött célokhoz, valamint a 2021 őszen leadott Felülettervekhez képest kevesebb funkciót tudtunk megvalósítani, és a megvalósított részekben is előfordulnak hiányosságok. Ettől függetlenül több menüpontban is sikerült elérni a kívánt működést, de ez rengeteg utánajárást és internetes böngészést igényelt. A fejlesztés során nagy segítséget jelentett a nagyrészt online formában zajló képzés miatt elérhető videók ismételt tanulmányozása.

8.2 Nehézségek

A vizsgaremek követelményrendszere úgy éreztük, hogy sok szempontból nincsen összhangban a leadott tananyaggal. Az első, új követelményrendszerben indított osztály tagjaiként azt tapasztaltuk, hogy a tematika és a tananyagok ütemezése a képzéssel egyidőben formálódott, melynek következtében az elvárások is folyamatában alakultak ki. A témaválasztás és a hozzá kapcsolódó felületterv leadása a tananyag ismerete nélkül történt. A vékonykliens technológiák - mobil, web, desktop - megválasztását nehezítette, hogy mire egyik programozási nyelvben elmélyedtünk volna, addigra előről kezdtük a programozást egy másik nyelvben. A kiválasztás pillanatában nem volt egyértelműen kiemelve, hogy a mobilprogramozás tanórák önmagukban nem adnak elegendő segítséget egy, a vizsgaremek követelményeinek megfelelő alkalmazás elkészítéséhez.

8.3 Tapasztalatok

Általánosságban elmondható, hogy a vizsgaremek csapatban történő elkészítése a csapattagok időbeosztása, a feladathoz való hozzáállása és élethelyzete miatt sok esetben nehézkes. Számunkra azonban ez könnyebbséget jelentett a Témaválasztás fejezetben már kifejtett okokból kifolyólag. A közös backend fejlesztés során támogatni és segíteni tudtuk egymást, és jó érzés volt közösen küzdeni a kitűzött céljainkért.

8.4 Szakmai fejlődés

A program készítése során nagyon sok új ismerettel gazdagodtunk, és alaposabban megismerkedtünk az alkalmazott programozási nyelvekkel és fejlesztői környezetekkel. Egy-egy probléma megoldása során gyakran fordultunk a hivatalos dokumentációkhoz. Az alkalmazás megvalósítása közben úgy érezzük, hogy szakmailag is fejlődtünk, és szakmai

Összegzés

szókincsünk is bővült. A csapatmunka nyomon követése érdekében alkalmazott verziókövetést is magabiztosabban használjuk és előnyeit is megtapasztaltuk.

Ábrajegyzék

1.	ábra: Trigger.....	12
2.	ábra: Tervező nézet	13
3.	ábra: Bejelentkezés utáni menü és visszajelzés.....	42
4.	ábra: Kijelentkezés utáni menü és visszajelzés	42
5.	ábra: Reszponzivitás.....	43
6.	ábra: Ranglista menüpont.....	43
7.	ábra: Szűrés – visszajelzés	43
8.	ábra: Hirdetések keresése – Hiba	44
9.	ábra: Angol nyelvű hibaüzenet.....	45
10.	ábra: Bejelentkezési hibaüzenet	47
11.	ábra: Bejelentkezési hibaüzenet	47
12.	ábra: Nem létező hirdetés	48
13.	ábra: Jelentkezés gátolása.....	49
14.	ábra: Jelentkezés gátolása.....	49
15.	ábra: Hirdetés törlés megerősítése.....	49
16.	ábra: Sikertelen kapcsolódás	50
17.	ábra: UML diagram.....	55
18.	ábra: Regisztráció oldalon visszajelzés a felhasználónak	60
19.	ábra: Üzenet sikeres regisztráció esetén	62
20.	ábra: Üzenet sikertelen bejelentkezés esetén.....	62
21.	ábra: Üzenet sikeres bejelentkezést követően	62
22.	ábra: Hirdetés oldalon visszajelzés a felhasználónak.....	62
23.	ábra: Üzenet sikeres hirdetésfeladást követően.....	63
24.	ábra: Üzenet profilmódosítást követően.....	63
25.	ábra: Megerősítés törlés esetén	63
26.	ábra: Lokális webszerver elindítása	67
27.	ábra: Regisztráció hibaüzenet.....	67
28.	ábra: Regisztráció menüpont.....	68
29.	ábra: Felhasználónév hiba	69
30.	ábra: Sikeres regisztráció	69
31.	ábra: Bejelentkezés menüpont.....	70
32.	ábra: Bejelentkezési hibaüzenet	70
33.	ábra: Bejelentkezési hiba.....	70
34.	ábra: Keresési találatok	71
35.	ábra: Profil menüpont - Hirdetések	72
36.	ábra: Profil törlése	73
37.	ábra: Profil és jelszó módosítása	74
38.	ábra: Hirdetés feladási űrlap.....	75
39.	ábra: Sikeres hirdetés feladás	75
40.	ábra: Ranglista menüpont.....	76
41.	ábra: Főoldal.....	79
42.	ábra: Regisztráció	80
43.	ábra: Bejelentkezés.....	81
44.	ábra: Hirdetés feladása	82
45.	ábra: Hirdetések listázása	83
46.	ábra: Konkrét hirdetés	84
47.	ábra: Profil oldal I.	85
48.	ábra: Profil oldal II.	86

Összegzés

49.	ábra: Profil módosítás.....	87
50.	ábra: Ranglista.....	88

Táblázatjegyzék

1.	táblázat: Adatbázis alapadatok	8
2.	táblázat: User tábla	8
3.	táblázat: Telepules tábla	9
4.	táblázat: Kategoria tábla	9
5.	táblázat: Hirdetes tábla	10
6.	táblázat: Jelentkezes tábla	11
7.	táblázat: Megszorítások	11
8.	táblázat: User_model metódusai	15
9.	táblázat: Telepules_model metódusai	15
10.	táblázat: Kategoria_model metódusai	15
11.	táblázat: Hirdetes_model metódusai	16
12.	táblázat: Jelentkezes_model metódusai	16
13.	táblázat: Ranglista_model metódusai	17
14.	táblázat: User API metódusai	19
15.	táblázat: Hirdetes API metódusai	20
16.	táblázat: Telepules API metódusai	20
17.	táblázat: Kategoria API metódusai	20
18.	táblázat: Profil controller – profil_megtekintes()	34
19.	táblázat: Hirdetes_kereses controller – hirdetes_kereses()	38
20.	táblázat: Hirdetes_kereses controller – hirdetesre_jelentkezes()	40
21.	táblázat: Statisztika controller – ranglista_megtekintes()	41
22.	táblázat: Regisztráció - Tesztesetek	45
23.	táblázat: Hibaüzenetek a regisztráció során	60
24.	táblázat: Regex felételek	61

Irodalomjegyzék

1. <https://www.codeigniter.com/userguide3/>, letöltve: 2022.09.03.
2. <https://www.php.net/>, letöltve: 2022.09.03.
3. <https://developer.android.com/reference/>, letöltve: 2022.09.03.
4. <https://www.w3schools.com/>, letöltve: 2022.09.03.
5. <https://stackoverflow.com/>, letöltve: 2022.09.03.
6. <https://www.geeksforgeeks.org/>, letöltve: 2022.09.03.
7. <https://regex101.com/>, letöltve: 2022.09.03.
8. <https://www.postman.com/>, letöltve: 2022.09.03.
9. https://www.tutorialspoint.com/android/android_json_parser.htm, letöltve: 2022.09.03.
10. https://github.com/darkbeast0106/Android_DolgozoApiRestClient, letöltve: 2022.09.03.
11. <https://github.com/darkbeast0106/Autokolcsonzo>, letöltve: 2022.09.03.
12. <https://github.com/darkbeast0106/DolgozoCodeigniterApi>, letöltve: 2022.09.03.
13. <https://webiskola.hu/php-ismeretek/php-apache-xampp-letoltese-telepitese/>, letöltve: 2022.09.03.
14. https://www.inf.u-szeged.hu/~gnemeth/adatbgyak/exe/MySQL_XAMPP_JDBC/phpmyadmin_a_mysql_kezelshez.html, letöltve: 2022.09.03.
15. <https://www.posta.hu/szolgaltatasok/iranyitoszam-kereso>, település táblához a táblázat letöltve: 2022.02.20.
16. https://en.wikipedia.org/wiki/List_of_HTTP_status_codes, letöltve: 2022.09.05.
17. <https://app.diagrams.net>, UML diagram készítéséhez, letöltve 2022.09.09.
18. <https://developer.android.com/studio>, rendszerkövetelményekhez, letöltve: 2022.09.09.
19. Vizsgaremekhez felhasznált profil, kategória, kezdőlapi és okmányképek forrása: Microsoft Word/Beszúrás/Képek/Stockképek/Képek/, letöltve: 2022.08.20.
20. Vizsgaremekhez felhasznált díjképek forrása: Microsoft Word/Beszúrás/Képek/Stockképek/Ábrák/, letöltve: 2022.08.20.
21. Vizsgaremekhez készített logó ikonjának forrása: Microsoft Word/Beszúrás/Képek/Stockképek/Ikonok/, letöltve: 2022.08.20.
22. A képzés során leadott tananyagok, elkészített forráskódok és videófelvevételek.