



Adatbázis alapú web alkalmazás fejlesztése

Készítette

Verebélyi Valentin

Programtervező Informatikus BSc

Témavezető

Dr. Tajti Tibor Gábor

Egyetemi Docens

EGER, 2024

Tartalomjegyzék

Bevezetés	4
1. Tervezés	5
1.1. A tervezéshez alkalmazott eszközök és technológiák	5
1.1.1. Dbdiagram	6
1.1.2. PlantUML	7
2. Fejlesztés	9
2.1. A fejlesztéshez alkalmazott eszközök és technológiák	9
2.1.1. Laravel	9
2.1.2. PHP	9
2.1.3. MySQL	10
2.1.4. PhpMyAdmin	10
2.1.5. XAMMP	10
2.1.6. Visual Studio Code	10
2.1.7. GitHub	10
2.1.8. GitHub Desktop	11
3. Tesztelés	12
3.1. A teszteléshez alkalmazott eszközök és technológiák	12
3.1.1. Cypress	12
4. Felhasználói dokumentáció	14
4.1. Admin	14
4.1.1. Felhasználókkal kapcsolatos művelet	14
4.1.2. Kategóriákkal kapcsolatos műveletek	15
4.1.3. Vármegyével és városokkal kapcsolatos műveletek	15
4.1.4. Hirdetéssel kapcsolatos műveletek	15
4.2. Felhasználó	16
4.2.1. Hirdetéssel kapcsolatos műveletek	16
4.3. Nézelődő	17
4.4. Egyéb funkciók	17

5. Fejlesztői dokumentáció	19
5.1. Az alkalmazás felépítése	19
5.1.1. Adatbázis kapcsolat létrehozása	19
5.1.2. Migráció	20
5.1.3. Modell	21
Összegzés	23
Irodalomjegyzék	24

Bevezetés

1. fejezet

Tervezés

A tervezés egy nagyon fontos szerepet tölt be egy szoftvereknél. Az én meglátásom és tudásom szerint a következő okok miatt szükséges tervezni:

1. Tervezési célok meghatározása: fontos az, hogy mind a fejlesztő, mind a megrendelő egyértelműen megértse, hogy pontosan mit kell elérni a szoftverrel.
2. Költség- és időmegtakarítás: implementálás előtt, ha kellő alaposággal tervezzük meg a szoftvert, akkor segíthet kiszűrni a későbbi fázisokban esetleges előforduló hibákat.
3. Bővíthetőség: ez alatt azt értem, hogy a szoftver úgy kell megtervezni, hogy fel legyen készítve arra, hogy lehessen hozzáadni könnyedén új funkciókat.
4. Funkciók és feladatok felosztása: ez azért lehet hasznos, mivel a fejlesztés során a fejlesztők pontosan csak az ő általuk elvállalt feladatokat valósítják meg.
5. Kommunikáció segítése: ha van egy jól kidolgozott terv, akkor ha a fejlesztők elakadnak valamiben vagy nem értik meg pontosan, hogy mit és hogyan kell implementálni, akkor elég, ha megnézik a tervben az adott dologhoz kapcsolódó részeket és ezáltal megtudják oldani a rájuk bízott feladatot.

1.1. A tervezéshez alkalmazott eszközök és technológiák

A következő alszakaszokban a tervezéshez használt eszközökről és technológiákról lesz majd szó.

1.1.1. Dbdiagram

A dbdiagram segítségével van lehetőség arra, hogy egy alkalmazás adatbázisának a sémáját és struktúráját megtervezzük és ehhez ad egy vizuális képet számunkra. A DBML-t, azaz Database Markup Language használja, amit magyarul talán adatbázis jelölőnyelvként tudnék lefordítani. Amiért a véleményem szerint nagyon hasznos még az nem más, mint, hogy rengetegféle olyan opciót ad számunkra, amire szükségünk lehet. Ilyen opciók például azok, hogy van lehetőség importálni be kódot MySQL, PostgreSQL, Rails valamint SQL szerverről. Ugyanezeket a típusú kódokat kilehet exportálni az imént megemlített típusokba, kivéve a Rails, mivel oda nem lehetséges. Ezenkívül van még olyan opció is, hogy ki lehet exportálni PNG, SVG vagy akár PDF formátumba is az elkészült tervet. A dbdiagram alap változata ingyenesen használható mindenki számára. [9]

A következőkben az 1.2 képen látható adatbázisterv kódjáról szeretnék egy keveset még írni, pontosabban arról, hogy hogyan épül fel az előbb bemutatott terv. Csak a tervről mutatott be pár részletet az érdekesség kedvéért.

```
// tábla létrehozása
Table Hirdetes
{
  hirdetes_id integer [pk, increment, not null]
  felhasznalo_id integer [not null]
  varos_id integer [not null]
  kategoria_id integer [not null]
  telefonszam varchar [unique]
  nev varchar [not null]
  ar varchar [not null]
  leiras text
}

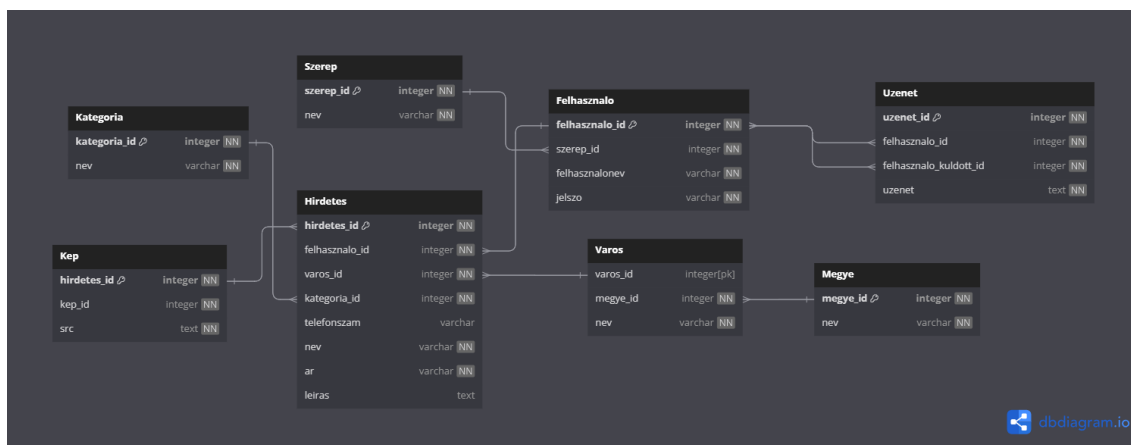
Table Varos
{
  varos_id integer [pk]
  megye_id integer [not null]
  nev varchar [unique, not null]
}

// táblák közötti kapcsolat megadása
Ref: Varos.varos_id < Hirdetes.varos_id

// < egy-a-többhöz kapcsolat
// > több-az-egyhez kapcsolat
// <> több-a-többhöz kapcsolat
```

1.1. ábra. DMBL kód részlet (Saját készítés)

Az 1.2 képen látható a dbdiagram-ban készített ábra az alkalmazásomhoz.



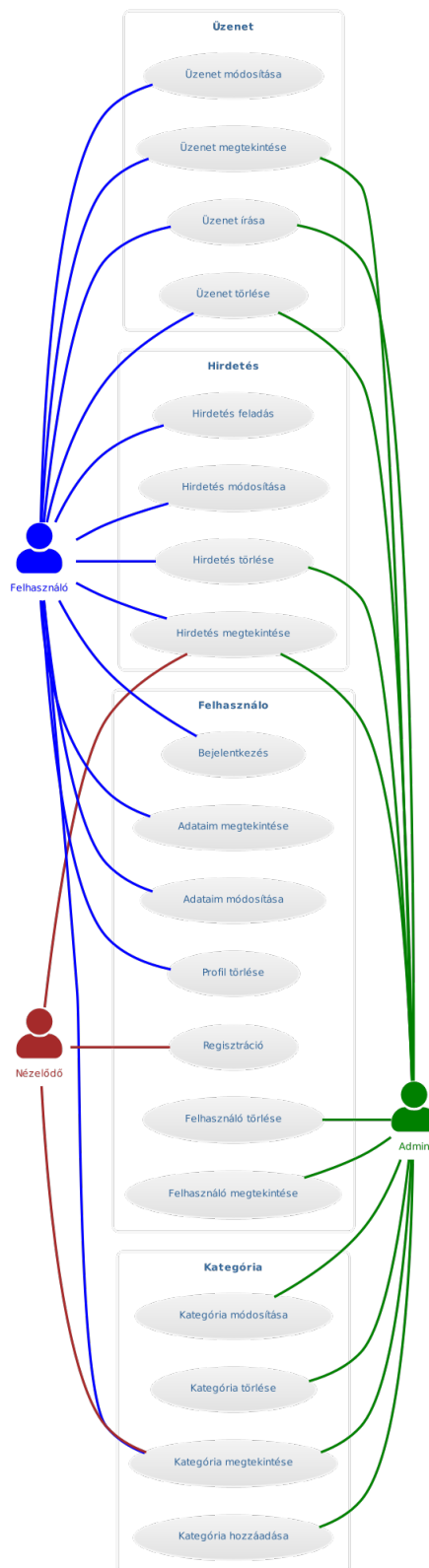
1.2. ábra. Adatbázisterv (Saját készítés)

1.1.2. PlantUML

Az UML a Unified Modeling Language rövidítése, azaz Egységes Modellezési Nyelv. Az UML arra jó, hogy szoftvert tervezzünk vele, a gondolataimat letudom vele rajzolni, skiccelni. Ezenkívül a kiforrott megoldások dokumentálására is használható. A PlantUML egy komponens, aminek a segítségével gyorsan tudunk létrehozni szekvencia-, használati eset- és osztálydiagramokat. Ezenkívül még rengetegféle diagramot tudunk készíteni. [7]

Manapság a tervezés az használati eset központú. A következő pár mondatban az ilyenféle alapú tervezésről lesz szó. Tulajdonképpen itt azt a kérdést tehetjük fel, hogy ki (angolul: actor) mire (funkció) tudja használni az adott informatika rendszert. Az actor-ról azt érdemes tudni, hogy ő nem része az informatikai rendszernek, hame ő csak használja azt. Az actor gyakran lehet ember, AI vagy akár egy másik informatikai rendszer. Fontos az is, hogy minden ábra, amit készítünk az visszavezethető legyen a már meghatározott követelmények szintjéig. Az használati alapú tervezés előnye az, hogy egyszerűen, érthetően írja le a rendszer funkcióit. Ebből kifolyólag, mind a megrendelő, mind a tervező számára könnyen érthető az ábra. Tulajdonképpen a használati eset alapú tervezés nem más, mint egy közös nyelv a megrendelő és a tervező között, amit mind a két fél ért.[2] Az 1.3 képen látható a plantUML-ben készített használati eset ábra az alkalmazásomhoz. Az 1.3 képen látható használati eset ábra magyarázata következik: Az ábrán három darab actor látható.

1. Nézelődő: ennek a szerepnek van a legkevesebb funkciója, ő az aki csak a már meglévő hirdetéseket és kategóriákat tudja megtekinteni, illetve képes még regisztráció, ami további meglévő funkciók használata miatt szükséges.
2. Felhasználó: ő képes a saját adatainak módosítására, a profil törlése is. Neki van már lehetősége saját hirdetéseket létrehozni és ezeket kezelni, valamint már van lehetősége üzenetet küldeni egy másik felhasználónak.



1.3. ábra. Használati eset (Saját készítés)

- Admin: neki már van lehetősége a kategóriákkal kapcsolatos műveletek elvégzésére is. Szintén képes üzenetet küldeni bárkinek, ő képes már felhasználókat és hirdetések törölni az adott esetben.

2. fejezet

Fejlesztés

2.1. A fejlesztéshez alkalmazott eszközök és technológiák

A következő alszakaszokban a fejlesztéshez használt eszközökről és technológiákról lesz majd szó.

2.1.1. Laravel

A Laravel egy nyílt forráskódú, PHP webkeretrendszer, ami MVC tervezési mintán alapszik. Az MVC a Model-View-Controller rövidítése.

1. Model (magyarul: modell): ez az adatokat kezelő réteg, az adatok tárolásáért és visszaolvasásáért felelős.
2. View (magyarul: nézet): ez a réteg felhasználói felületek megjelenítéséért illetékes.
3. Controller (magyarul: vezérlő): a felhasználói műveletek megfelelő kezeléséért ez a réteg a felelős.

Mivel három részre van bontva, ezért van ennek a tervezési mintának néhány előnye is. Ilyen például ha lecseréljük az egyik réteget, akkor a többihez nem kell már hozzányúlni, amivel pénzt és időt is lehet spórolni. Előnye még az is, hogy egy modellnek lehet több nézetes is. [1, 102-103. oldal]

2.1.2. PHP

A PHP egy szerveroldali szkriptnyelv, amelynek a segítségével dinamikus weboldalakat lehet vele létrehozni. Azt érdemes még róla tudni, hogy nyílt forráskódú. Ezen kívül még a következő feladatok megvalósítására lehet használni:

1. parancssoros szkripttelés: itt lehetőségünk van arra, hogy úgy futtassunk egy php szkriptet, hogy nem használunk se böngészőt, se semmilyen szerveret sem.
2. asztali alkalmazások készítése: grafikus felhasználó felülettel ellátott asztali alkalmazások esetén a php nem a legjobb választás, azonban van lehetőségünk arra, hogy platformfüggetlen alkalmazások megvalósítsunk.

[11]

2.1.3. MySQL

A MySQL nem más, mint egy nyílt forráskódú adatbázis kezelő rendszer. A MySQL egy relációs adatbázis, ahol az adatokat különböző táblákban vannak eltárolva. A különböző táblákban szereplő mezők között lehetnek különféle kapcsolatok is. Ilyen lehet például az egy-az-egyhez kapcsolat vagy egy-a-többhöz kapcsolat. A MySQL adatbázisok szerverére jellemző, hogy gyorsak, megbízhatóak és skálázhatóak. [4]

2.1.4. PhpMyAdmin

A phpMyAdmin egy ingyenes szoftver, ami arra szolgál, hogy kezelje a MySQL-nek az adminisztrációját weben keresztül. A phpMyAdmin segítségével elvégezhetők a legtöbb adminisztratív feladatok, ideértve az adatbázis létrehozását, lekérdezések futtatását és felhasználói fiókok hozzáadását. [5]

2.1.5. XAMMP

Az XAMMP nem más, mint egy PHP fejlesztői környezet. Erről még azt érdemes tudni, hogy ingyenesen használható és rendkívül egyszerű telepítése, illetve a használata.

2.1.6. Visual Studio Code

Ez egy IDE (integrált fejlesztői környezet), ami ingyenes használható és az egyik legelterjedtebb és legnépszerűbb a fejlesztők körében. Fontosnak tartom megemlíteni, hogy a Visual Studio Code-ban van lehetőség különféle kiegészítők (angolul: extension) letöltésére is, például van lehetőség a python programozási nyelv vagy egy programozási nyelvhez tartozó szintaxis kiemelő letöltésére, illetve ezek használatára.

2.1.7. GitHub

A GitHub-ról azt érdemes tudni, hogy ez egy verziókövető rendszer. Ezek a rendszerek képesek állományok tartalmi változásait követni, azt is képesek megmondani, hogy ki és mikor módosította azokat, valamint van lehetőség arra is, hogy korábbi állapotokat is

képes előállítani. A main branch-be feleltethető meg a fő ágnak, amiből van lehetőség elágazások (angolul: branch) is létrehozni. Az elágazásokat arra valóak, hogy a fejlesztési funkciókat elkülönítsük. Még arra is használhatjuk, hogy kísérletezzünk vagy akár hibajavítások elvégzésére is lehet használni.

2.1.8. GitHub Desktop

A GitHub Desktop egy ingyenesen használható alkalmazás, aminek a segítségével tudunk dolgozni a GitHub-on vagy Git-tárhely-szolgáltatásokon tárolt fájlokkal. Én ezt az eszközt azért szeretem használni, mivel megkönnyíti és felgyorsítja számomra a munkavégzés, illetve azért is, mert ennek a használatához nem kell a terminálban beírni a Git-hez tartozó parancsokat, hanem egy-két kattintás segítségével elvégezhetek egy konkrét parancsot. [6]

3. fejezet

Tesztelés

A tesztelésre azért van szükség, hogy az alkalmazásomban megtaláljam az esetleges hibákat, amiket kijavítva növelhetem a szoftverem minőségét és megbízhatóságát. Abban sajnos nem lehetek biztos, hogy a tesztelés elvégzése után nem lesznek már hibák. A tesztelés során kettőféle tesztelési technikát alkalmaztam. Ezek pedig a következők:

1. Fehérdobozos (angolul: white-box): a forráskód alapján íródnak a tesztesetek.
2. Szürkedofozos (angolul: grey-box): amikor a forráskódnak csak egy rész ismert és csak ez alapján íródnak a tesztesetek.

Azonban van egy harmadik féle is, amit nem alkalmaztam. Az nem más, mint a fekete-dofozos (angolul: black-box), amikor a tesztesetek a specifikáció alapján íródnak. [1, 26-29. oldal]

3.1. A teszteléshez alkalmazott eszközök és technológiák

A következő alszakaszokban a teszteléshez használt eszközökről és technológiákról lesz majd szó.

3.1.1. Cypress

A Cypress egy NodeJS-ben írt front end tesztelési keretrendszer. Ahhoz, hogy tudjuk futtatni a Cypress-t, ahhoz előtte telepíteni kell a NodeJS-t. A Cypress segítségével tudunk készíteni végponttól végpontig tartó-, komponens-, integrációs- valamint a unitteszteket is. Az imént felsorolt teszt típusok jelentése a következő:

1. Végponttól végpontig tartó teszt: ennél a típusnál a szoftver funkcionalitását és teljesítményét teszteljük a kezdetektől a végéig, ami a vég felhasználók által megvalósított interakciókat szimulálja valós adatokkal.

2. **Komponensteszt:** ez nem más, mint, amikor a rendszernek csak egy önálló komponensét teszteljük.
3. **Integrációs teszt:** ez egy olyanféle teszt, aminél legalább kettő különböző komponens együttműködését teszteljük.
4. **Unitteszt (magyarul: egységteszt):** ez egy olyan teszt, ami a metódusokat teszteli le. Itt nézzük meg, hogy a tényleges visszatérési érték azonos-e az elvárttal. Ez a komponensteszt egyik fajtája.

Ezzel a tesztelési keretrendszer segítségével bármit lehet tesztelni ami a böngészőben fut. [8, 3, 10]

4. fejezet

Felhasználói dokumentáció

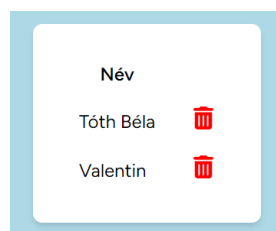
Ebben a fejezetben szó lesz majd, arról, hogy különböző jogosultságú felhasználóknak milyen lehetőségei vannak a webes alkalmazásom használata során. Minden ebben a fejezetben lévő alfejezetben szereplő jogosultsághoz szinte csak azokat a funkciókat említem majd meg, amiket csak ők tudnák használni. A különböző szerepű felhasználókról az 1.1.2 alfejezetben már megemlítettem róluk pár szót.

4.1. Admin

Az admin (magyarul: adminisztrátor) van néhány szükséges egyedi funkció az alkalmazásomnak. Az adminisztrátoroknak nagyon fontos szerepük bármilyen alkalmazás életében. Az webes projektomban a következő admin funkciókat készítettem el.

4.1.1. Felhasználókkal kapcsolatos művelet

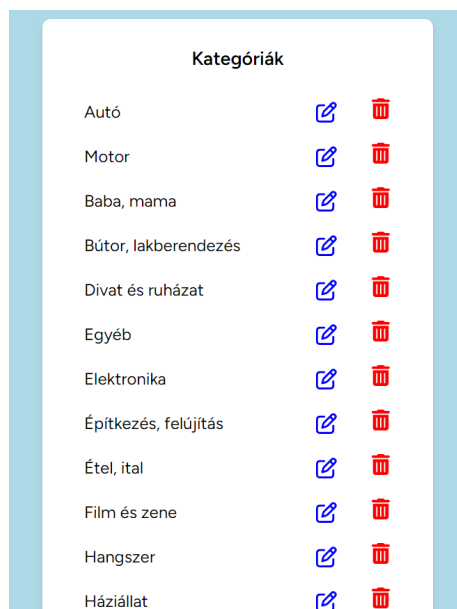
Az én meglévő tudásom birtokában úgy gondoltam, hogy a csak és kizárólag törölni tudjon már regisztrált felhasználókat. Admin jogosultságú felhasználókat nem lehetséges kitörölni, azaz amennyiben van több ilyen, akkor nem tudják kitörölni egymást. Ennek a jogosultságnak a birtokában képes új kategóriát hozzáadni, módosítani már meglévőt és még képes törölni is, amennyiben szükséges. A 4.1 ábrán a piros színű ikon segítségével törölni lehet a felhasználót.


























4.1. ábra. Felhasználók (Saját készítés)

4.1.2. Kategóriákkal kapcsolatos műveletek

Ennek a jogosultságnak a birtokában képes új kategóriát hozzáadni, módosítani már meglévőt és még képes törölni is, amennyiben szükséges. A 4.2 ábrán a kék színű ikon segítségével módosítani lehet a kategóriát, a piros színűvel pedig törölni. A képen nem látszik viszont van ott egy gomb, aminek a segítségével tudunk új kategóriát hozzáadni.



Kategóriák		
Autó		
Motor		
Baba, mama		
Bútor, lakberendezés		
Divat és ruházat		
Egyéb		
Elektronika		
Építkezés, felújítás		
Étel, ital		
Film és zene		
Hangszer		
Háziállat		






















4.2. ábra. Kategóriák (Saját készítés)

4.1.3. Vármegyével és városokkal kapcsolatos műveletek

Itt az admin jogosultságú felhasználó képes új vármegyét hozzáadni, meglévőt módosítani, illetve törölni. Szintén képes még különböző vármegyékhez hozzáadni új városokat, meglévőt módosítani és törölni. A 4.3 ábrán a zöld színű ikon segítségével lehet megtekinteni az adott vármegyéhez tartozó városokat megjeleníteni, ez ugyan úgy épül fel, mint az imént említett kategóriák a 4.1.2 alfejezetben. A kék színű ikon segítségével módosítani lehet a kategóriát, a piros színűvel pedig törölni. A képen nem látszik viszont van ott egy gomb, aminek a segítségével tudunk új vármegyét hozzáadni.

4.1.4. Hirdetéssel kapcsolatos műveletek

Ez lenne az egyetlen kivétel, itt a sima felhasználó is képes módosítani vagy törölni a saját hirdetéseit. Képes a már az összes meglévő hirdetéseket módosítani, illetve törölni azokat, ha valamilyen oknál fogva nem megfelelő nyelvezetet vagy képet használ.

Vármegye		
Bács-Kiskun vármegye		
Baranya vármegye		
Békés vármegye		
Borsod-Abaúj-Zemplén vármegye		
Csongrád vármegye		
Fejér vármegye		
Győr-Moson-Sopron vármegye		
Hajdú-Bihar vármegye		
Heves vármegye		
Jász-Nagykun-Szolnok vármegye		
Komárom-Esztergom vármegye		

4.3. ábra. Vármegyék (Saját készítés)

4.2. Felhasználó

A következő alfejezetekben csak a sima felhasználó egyedi funkcióijáról lesz szó.

4.2.1. Hirdetéssel kapcsolatos műveletek

Regisztrált felhasználóként van arra lehetőség, hogy a hirdetéseivel kapcsolatos műveleteket végezzen el, legyen szó akár csak azok megtekintéséről, módosításairól vagy törléseiről. Ő már képes új hirdetések létrehozni, ami a 4.4 ábrán lesz látható. A 4.4

<p>Vármegye</p> <p>Válassz vármegyét</p> <p>Város</p> <p>Válassz várost</p> <p>Kategória</p> <p>Válassz kategóriát</p> <p>Kép 1</p> <p>Fájl kiválasztása Nincs fájl kiválasztva</p> <p>Kép 2</p> <p>Fájl kiválasztása Nincs fájl kiválasztva</p> <p>Kép 3</p> <p>Fájl kiválasztása Nincs fájl kiválasztva</p>	<p>Cím</p> <p></p> <p>Ár</p> <p>Ft</p> <p>Leírás</p> <p></p> <p>Telefonszám</p> <p></p> <p>Hozzáadás</p>
---	---

4.4. ábra. Új hirdetés hozzáadása (Saját készítés)

ábrán magyarázata következik. A vármegyét egy legördülő listából lehet kiválasztani, ha ezt megtettük csak utána tudjuk majd a várost kiválasztani szintén legördülő listából. A hirdetés kategóriáját is ilyenféle módon választhatjuk ki. A hirdetés címét, árát és leírását pedig a felhasználó adja majd meg. Ezek voltak a kötelezően megadandó információk. Telefonszámot, illetve képeket nem szükséges megadni. A megadható képek számát limitáltam 5 darabra.

4.3. Nézelődő

Ő nem rendelkezik semmilyen jogosultsággal sem, ezért is nincsen egyedi funkciója ennek a szerepkörnek. Ő csak a kategóriákat képes megtekinteni, illetve a meglévő hirdetéseket tudja megnézni. Keresésre és szűrésre is van itt lehetőség. Ezekről a 4.4 alfejezetben olvashat többet.

4.4. Egyéb funkciók

1. Keresés: a hirdetéseknel van arra lehetőség, hogy keresni lehessen, ez a hirdetés címe alapján lehetséges.
2. Szűrés: van arra lehetőség, hogy szűrjünk ára, vármegyére, városra vagy kategóriára is.
3. Rendezés: a hirdetéseket rendezhetjük ár szerint növekvő, csökkenőbe vagy lehet alapértelmezett is.

The image shows a web interface for searching and filtering advertisements. At the top, there is a search bar with a dropdown menu labeled 'Keresés cím alapján' and a green 'Keresés' button. Below the search bar, there are several filter options. On the left, there is a section for 'Vármegye:' with a dropdown menu labeled 'Válassz vármegyét'. In the center, there is a checkbox labeled 'Szűrés megjelenítése' which is checked. Below the checkbox, there is a section for 'Város:' with a dropdown menu labeled 'Válassz várost'. On the right, there is a section for 'Kategória:' with a dropdown menu labeled 'Válassz kategóriát'. Below these, there are two input fields for price: 'Minimum ár:' and 'Maximum ár:', both with 'Ft' (Hungarian Forint) units. To the right of these, there is a 'Rendezés:' dropdown menu set to 'Alapértelmezett'. At the bottom center, there is a green 'Szűrés' button.

4.5. ábra. Keresés, szűrés és rendezés (Saját készítés)

4. Üzenetek: a regisztrált felhasználók vagy adminisztrátorok képesek üzenetek küldeni egymásnak, ezeket megtekinteni, módosítani vagy törölni a saját általuk korábban elküldött üzeneteket is.

5. Profil módosítása: a webes alkalmazásban adva van a lehetőség arra, hogy a regisztrált felhasználó tudják módosítani a saját adataikat, illetve képesek törölni a profiljukat is. Az adminisztrátor nem képes arra, hogy módosítsa a felhasználók adatait, csak törölni tudja őket.

5. fejezet

Fejlesztői dokumentáció

5.1. Az alkalmazás felépítése

Ebben az alfejezetben szeretném mutatni Önöknek az alkalmazásom fő komponenseit kódrészletek segítségével a könnyebb megértés reményében. Egy nagyon keveset már említettem megemlítettem korábban a 2.1.1 alfejezetben, de nem mindent, most lépésről lépésre elmesélem, hogy hogyan lehetséges megjeleníteni egy konkrét adatokat az adatbázisomból.

5.1.1. Adatbázis kapcsolat létrehozása

Magát az adatbázist többféle módon van lehetőségünk létrehozni, de ezekbe most nem mennék bele. Most itt még azt kell megemlítenem, hogy a projektben ezt hogyan szükséges beállítani. Ehhez meg kell keresni a projektben a .env nevű fájlt és meg kell adni néhány információt. A következőket szükséges megadni.

1. DB_CONNECTION (magyarul: adatbázis kapcsolat): itt magát az adatbázis fajtáját kell megadnunk. Alapértelmezetten mysql van megadva.
2. DB_HOST (magyarul: adatbázis kiszolgáló): ez az adatbázis IP-címe lenne.
3. DB_PORT (magyarul: adatbázis port): ennek a segítségével tudunk kapcsolódni az adatbázis szerverhez.
4. DB_DATABASE (magyarul: adatbázis neve): itt az magát az elnevezését kell megadnunk.
5. DB_USERNAME (magyarul: felhasználónév az adatbázishoz): meg kell adnunk egy felhasználónevet, amelyet az alkalmazás használ a belépéshez.
6. DB_PASSWORD (magyarul: jelszó az adatbázishoz): a belépéshez meg kell adnunk még egy jelszót is.

5.1.2. Migráció

A migrációk segítségével tulajdonképpen az adatbázis struktúráját tudjuk definiálni. Van arra lehetőség, hogy egy migrációs fájlt egy parancs segítségével le lehessen generálni struktúra szintjén, de már a benne szükséges mezőket és az esetleges különböző megszorításokat már nekünk kell definiálni. A következő parancs segítségével lehetséges egy migrációs fájlt generálni:

```
php artisan make:migration create_név_table .
```

Következőnek nézzük azt meg, hogy hogyan is épül fel egy migrációs fájlt, konkrétan az a hirdetéshez tartozó.

5.1. kód. Egy migrációra példa kód

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('advertisements', function (Blueprint $table) {
15             $table->id('advertisement_id');
16             $table->unsignedBigInteger('user_id');
17             $table->unsignedBigInteger('city_id');
18             $table->unsignedBigInteger('category_id');
19             $table->string('title');
20             $table->integer('price');
21             $table->text('description');
22             $table->string('mobile_number')->nullable();
23             $table->timestamps();
24
25             $table->foreign('user_id')->references('user_id')->on('users')
26                 ↳ ->onDelete('cascade');
27             $table->foreign('city_id')->references('city_id')->on('cities')
28                 ↳ ->onDelete('cascade');
29             $table->foreign('category_id')->references('category_id')->on('categories')
30                 ↳ ->onDelete('cascade');
31         });
32     }
33 }
34 /**
```

```

31      * Reverse the migrations.
32      */
33      public function down(): void
34      {
35          Schema::dropIfExists('advertisements');
36      }
37 };

```

Látható tehát, hogy egy mezőt úgy szükséges megadni, hogy \$table->mező típusa, majd zárójelbe a mező nevét. A 15. sorban az elsődleges kulcs megadása látható. Az alatta lévő sorban pedig egy olyan mezőt hoztam létre, ami majd egy másik migrációs fájlból kapja majd meg az értékét, tehát ez egy idegen kulcs. A 25. sorban pedig az imént említett mezőre így lehetséges hivatkozni. Itt ennek a sornak végén lévő kód az azt jelenti, hogy ha kitörlődik az adott felhasználó, akkor a hirdetés is törlődni fog vele együtt. Alapértelmezetten egyik mező értéke sem lehet nulla, ha azonban szükségünk van rá, akkor meglehet tenni úgy, ahogyan én a 22. sorban csináltam. Ha elkészültünk a migrációs fájlokkal, akkor a következő parancs segítségével lehet majd végrehajtani azt:

php artisan migrate .

5.1.3. Modell

A laravel már tartalmazza az eloquentet, ami egy ORM (angoul: Object-Relational Mapping, magyarul: objektum-relációs leképzés). Amikor ezt használjuk, akkor minden egyes adatbázis táblához tartozik egy modell is, aminek a segítségével interaktálni tudunk a táblával. Az ilyenféle modellek használatával lehetséges beszúrni, módosítani vagy törölni az adott adatbázis táblából. Itt a modellekben még megadhatunk különféle metódusokat is és még itt van lehetőség az eloquent kapcsolatok megadására is. Ezekre még majd kitérek a példa bemutatása során.[12]

5.2. kód. Egy modellre példa kód részlet

```

1  <?php
2  class Advertisement extends Model
3  {
4      use HasFactory;
5
6      protected $table = 'advertisements';
7      protected $primaryKey = 'advertisement_id';
8
9      protected $fillable = [
10         'city_id',
11         'category_id',
12         'title',

```

```

13     'price',
14     'description',
15 ];
16 public function user()
17 {
18     return $this->belongsTo(User::class, 'user_id', 'user_id');
19 }
20 public function pictures()
21 {
22     return $this->hasMany(Picture::class, 'advertisement_id', '
23         ↪ advertisement_id');
24 }

```

Az 5.2 kódrészleten a 6. és 7. sorban az adatbázis tábla neve és elsődleges kulcs megadása látható. Rögtön alatta pedig az látszik, hogy egy tömbbe megadom a kötelezően megadandó mezők nevét, amik nem lehetnek üresek. Itt található még egy `belongsTo` és egy `hasMany` kapcsolatra egy példa. Az első azt röviden az azt jelenti, egy modell egy másik modellhez tartozik, azaz, hogy egy hirdetés egy felhasználóhoz tartozik. Míg az utóbbi azt jelenti, hogy egy modell több modellhez tartozik, azaz egy hirdetéshez több kép is tartozhat.

5.1.4. Kontroller

5.1.5. Blade

5.1.6. Webp.pph

Összegzés

Irodalomjegyzék

- [1] KUSPER GÁBOR: *Programozási technológiák*, Eger, 2015.
- [2] KUSPER GÁBOR: *Informatikai rendszerek tervezése*, Eger, 2023, 0.8.7.2-es verzió
- [3] KUSPER GÁBOR: *Szoftvertesztelés*, Eger, 2023, 2023.10.14-es verzió
- [4] ORACLE: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>, 2024-es verzió.
- [5] PHPMYADMIN: <https://www.phpmyadmin.net/>, 2024-es verzió.
- [6] GITHUB DESKTOP: <https://docs.github.com/en/desktop/overview/about-github-desktop>. 2024-es verzió.
- [7] PLANTUML: *PlantUML Language Reference Guide*, <https://plantuml.com/guide>, 2023. novemberi verzió.
- [8] CYPRESS: <https://docs.cypress.io/guides/overview/why-cypress>, 2024-es verzió.
- [9] DBDIAGRAM: <https://dbml.dbdiagram.io/docs/>, 2024-es verzió.
- [10] KATALON: <https://katalon.com/resources-center/blog/end-to-end-e2e-testing>, 2024-es verzió.
- [11] PHP: <https://www.php.net/docs.php>, 2024-es verzió.
- [12] LARAVEL: <https://laravel.com/docs/10.x>, 2024-es verzió.