



# Adatbázis alapú web alkalmazás fejlesztése

## **Készítette**

Verebélyi Valentin

Programtervező Informatikus BSc

## **Témavezető**

Dr. Tajti Tibor Gábor

Egyetemi Docens

EGER, 2024

# Tartalomjegyzék

<b>Bevezetés</b>	<b>4</b>
<b>1. Tervezés</b>	<b>6</b>
1.1. A tervezéshez alkalmazott eszközök és technológiák . . . . .	7
1.1.1. DBDiagram . . . . .	7
1.1.2. PlantUML . . . . .	8
1.2. Logó . . . . .	10
1.3. Egyéb . . . . .	10
<b>2. Fejlesztés</b>	<b>11</b>
2.1. A fejlesztéshez alkalmazott eszközök és technológiák . . . . .	11
2.1.1. Laravel . . . . .	11
2.1.2. PHP . . . . .	12
2.1.3. MySQL . . . . .	12
2.1.4. PhpMyAdmin . . . . .	12
2.1.5. XAMPP . . . . .	13
2.1.6. Visual Studio Code . . . . .	13
2.1.7. GitHub . . . . .	13
2.1.8. GitHub Desktop . . . . .	13
2.1.9. HTML . . . . .	14
2.1.10. Tailwind CSS . . . . .	14
2.1.11. JavaScript . . . . .	14
<b>3. Tesztelés</b>	<b>15</b>
3.1. A teszteléshez alkalmazott eszközök és technológiák . . . . .	15
3.1.1. Manuális teszt . . . . .	15
3.1.2. Cypress . . . . .	16
3.1.3. Tesztek végrehajtása . . . . .	18
3.1.4. Tesztek eredménye . . . . .	18
<b>4. Felhasználói dokumentáció</b>	<b>20</b>
4.1. Admin . . . . .	20

4.1.1.	Felhasználókkal kapcsolatos művelet . . . . .	20
4.1.2.	Kategóriákkal kapcsolatos műveletek . . . . .	20
4.1.3.	Vármegyével és városokkal kapcsolatos műveletek . . . . .	21
4.1.4.	Hirdetéssel kapcsolatos műveletek . . . . .	21
4.2.	Felhasználó . . . . .	21
4.2.1.	Hirdetéssel kapcsolatos műveletek . . . . .	22
4.3.	Nézelődő . . . . .	22
4.4.	Egyéb funkciók . . . . .	22
<b>5.</b>	<b>Fejlesztői dokumentáció</b>	<b>24</b>
5.1.	Az alkalmazás felépítése . . . . .	24
5.1.1.	Adatbázis kapcsolat létrehozása . . . . .	24
5.1.2.	Migráció . . . . .	25
5.1.3.	Modell . . . . .	26
5.1.4.	Kontroller . . . . .	27
5.1.5.	Web.php . . . . .	28
5.1.6.	Blade . . . . .	29
5.2.	Jogosultságkezelés . . . . .	30
5.3.	Seeder . . . . .	30
<b>6.</b>	<b>Üzleti modell</b>	<b>31</b>
6.1.	Online hirdetések . . . . .	31
6.2.	Kiemelés . . . . .	31
6.3.	Együttműködések . . . . .	31
	<b>Összegzés</b>	<b>33</b>
	<b>Függelékek</b>	<b>34</b>
	<b>Irodalomjegyzék</b>	<b>38</b>

# Bevezetés

<https://github.com/rcfr84/Szakdolgozat>

Én abból az indíttatásból választottam az adatbázis alapú webalkalmazás projektet, mivel hozzám közel állnak a webes alkalmazások. Maga a projekt témája az nem más, mint egy apróhirdetés weboldal. Azért választottam ezt a projekt ötletet, mivel nagyon érdekesnek találtam, és a komplexitása is megfelelő, ami egy ilyen jellegű projektnél elvárható. Ezen kívül pedig az én véleményem szerint van aktualitása a projektemnek, mivel Magyarországon jelenleg nem lenne sok konkurenciája, illetve a tapasztalatom szerint sokan élnek az ilyenféle rendszerek használatával manapság. Fontosnak tartom megemlíteni, hogy ezt az alkalmazást a kezdetektől fogva a Magyarország területén élők számára terveztem megvalósítani. Az volt a célom az alkalmazással létrehozásával kapcsolatban, hogy belelássak egy ilyen komplex alkalmazás elkészítésébe, felépítésébe és ezeknek köszönhetően a szerzett tapasztalatok által bővítsem a tudásomat, tapasztalatomat a webes projektek területén és a jövőben ezek a tapasztalatok a hasznomra fognak válni. Az elkövetkezendő néhány fejezetben szeretném Önnek bemutatni és ismertetni az általam készített adatbázis alapú webalkalmazásomat. Először majd magáról az alkalmazásom tervezéséről lesz szó, röviden érinteni fogom a tervezéshez kötődő, általam hasznosnak vélt információkat. Ezt követően pedig a fejlesztésről teszek majd meg néhány említést, azonban ebben a fejezetben főként a fejlesztéshez használt eszközök és technológiák bemutatásáról fog szólni. Ezek után pedig a tesztelésről ejtek majd meg pár gondolatot, megosztok Önnel különböző tesztelési technikákat és röviden ismertetni is fogom őket. A célom, ezzel a fejezettel, hogy egy átfogó képet adjak magáról a tesztelésről, illetve a fontosságáról is. Majd a következő fejezetben részletesen bemutatom az adatbázis alapú webalkalmazásomban megvalósított különböző jogosultságokkal rendelkező felhasználók által elérhető funkciókat. Ezt követően pedig az utolsó előtti számozott fejezetben alaposan tárgyalom majd meg, hogy hogyan épül fel az alapoktól a megvalósításig egy a projektemben szereplő nézet és annak a hozzá tartozó funkciói. Végezetül pedig az apróhirdetés webalkalmazásomhoz fenntartásához, működéséhez és karbantartásához fogok opciókat javasolni. A könnyebb és átláthatóbb megértés érdekében, valamint a szakdolgozat illusztrálása céljából néhány ábrát és képet fogok használni az alkalmazásom egyes funkcióinak bemutatásához, illetve néhány kódrészletet is szeretnék bemutatni majd Önnek. Összességében tehát, az adatbázis

alapú webalkalmazás projekt választása nemcsak az érdeklődésemet tükrözi a webfejlesztés iránt, hanem lehetőséget kínál számomra arra is, hogy elmélyedjek egy izgalmas és változatos témában, és fejlesszem a tudásomat, illetve a tapasztalataimat is. Bízom benne, hogy sikerült felkeltenem az Ön érdeklődését a választott témával kapcsolatban.

# 1. fejezet

## Tervezés

A tervezés egy nagyon fontos, központi szerepet tölt be egy szoftvereknél. Az én megítélásom 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.

Azonban az imént felsoroltakon kívül még számtalan ok miatt szükséges a tervezni. A következőben az én általam készített adatbázis alapú webalkalmazásomnak a követelménylistája látható. A követelménylistában azokat a funkciókat szükséges dokumentálni, amikkel a programoknak rendelkeznie kell. Ez általában a különböző projektek esetében a követelmény specifikációban található meg. A követelmény specifikáció egy olyan dokumentáció, amelyben az igények, követelmények kerülnek meghatározásra, ennek az alapja a riportok. A riportok segítségével lehetséges felmérni a megrendelő követelményeit. A 34. oldalon megtekinthető a projektem követelmény listája.

## 1.1. A tervezéshez alkalmazott eszközök és technológiák

A következő alszakaszokban a tervezéshez használt eszközökről és technológiákról lesz majd szó, melyeknek bemutatásaival az a célom, hogy alaposabb megértést nyújtsak az adatbázis alapú webalkalmazásomhoz kapcsoló alkalmazott eszközökről és technológiákról.

### 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 lehetőségünk importálni be kódot, ami lehet *MySQL* (többet megtudhat erről a 2. fejezetben), *PostgreSQL*<sup>1</sup>, *Rails*<sup>2</sup>. Ugyanezeket a típusú kódokat, nyelveket 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*<sup>3</sup> vagy akár *PDF* formátumba is az elkészült tervet. A *dbdiagram* alap változata ingyenesen használható mindenki számára. [9]



1.1. ábra. Adatbázisterv (Saját készítés)

A következőkben az 1.1. képen látható adatbázistervről szeretnék egy pár gondolatot megosztani. Pontosabban arról, hogy hogyan milyen fontosabb táblákból áll és

<sup>1</sup> Ez nem más, mint egy relációsadatbázis-kezelő rendszer

<sup>2</sup> Ez egy keretrendszer webalkalmazások készítéséhez.

<sup>3</sup> XML alapú leírónyelv.

milyen kapcsolatok találhatók meg táblák között. Látható, hogy ez a véleményem szerint egy közepes mennyiségű táblából álló adatbázis-terv. Kezdjük a felhasználó nevű táblával, ahogyan, mint minden sok más hasonló webalkalmazás esetén itt is szükséges eltárolni egy email címet, egy felhasználónevet, illetve egy jelszót. Ezeken kívül egy másik táblából, nevezetesen a szerep nevű táblából pedig kap egy a megfelelő jogszátságot ellenőrzéséhez egy azonosítót. Itt a két tábla között egy-a-többhöz kapcsolat áll fent, mivel egy szerepel rendelkező felhasználóból több is lehetséges. Ez a kapcsolat több helyen is előfordul még az adatbázis-tervben, például a város és vármegye tábla között is fennáll. Látható még egy üzenetek tábla is, amivel két felhasználó képes szöveges üzeneteket küldeni egymásnak. Itt a két tábla között már a több-a-többhöz kapcsolat áll fenn, mivel egy felhasználó több üzenetet küldhet vagy kaphat, tehát egy felhasználóhoz több üzenet is tartozhat. Utolsóként pedig a hirdetés tábláról szeretném megosztani a gondolataimat. A meglátásom szerint egy hirdetés eltárolásához egy hirdetésnév, egy ár, egy leírás, egy kategória név, egy város név és egy telefonszám mezők megadása nélkülözhetetlen. Ezek a mezők közül csak és kizárólag a telefonszám megadása nem kötelező.

### 1.1.2. PlantUML

Az UML a Unified Modeling Language rövidítése, azaz Egységes Modellezési Nyelv. Az UML arra jó, hogy szoftvert tervezzünk vele, a gondolatainkat letudjuk 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. [8]

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, hanem ő csak használja azt. Az *actor* gyakran lehet ember, AI vagy akár lehetséges egy másik informatikai rendszer is. 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.[3]

Az 1.2. képen látható a PlantUML-ben készített használati eset ábra az alkalmazásomhoz. Az 1.2. képen látható használati eset ábra magyarázata következik: Fontosnak tartom megemlíteni azt, hogy itt ezen az ábrán nem minden funkció kerül megjelení-





1.2. ábra. Használati eset (Saját készítés)

tésére, csak az általam vélt legfontosabbak. Többet megtudhat ezekről a 4. fejezetben. 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 regisztrációra is, 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 üzeneteket küldeni különböző felhasználóknak.
3. *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.

## 1.2. Logó

Az adatbázis alapú webalkalmazásomhoz készítettem el egy logót is, ami látható a projektem bejelentkezés és regisztrációs oldalon, illetve magán a navigációs sávon is. A logó elkészítéséhez segítségül vettem az interneten található képeket, majd ezeket összeszerkesztettem, végül pedig Microsoft PowerPoint-ban található funkció segítségével adtam a logónak egy művészi effektust. Az 1.3. ábrán megtekinthető a logó. Az alkalmazásomban szintén egy, az interneten található képet használtam háttérképként. [20, 21]



1.3. ábra. Logó (Saját készítés)

## 1.3. Egyéb

Amit még érdekesnek tartotok megemlíteni az olvasóban felmerülhet az a kérdés, hogy a Magyarország területén található vármegyékben elhelyezkedő városokat honnan gyűjtöttem össze. Ezen szükséges információkat a KSH, azaz Központi Statisztikai Hivatal oldalán található Excel táblázat alapján sikerült összegyűjtenem. Azonban sajnálatos módon ez 2015-ös információk alapján lett elkészítve, azóta nagy valószínűséggel néhány adat már elavult benne. [19]

## 2. fejezet

# Fejlesztés

### 2.1. A fejlesztéshez alkalmazott eszközök és technológiák

A következő alszakaszokban a fejlesztéshez használt eszközökről és technológiákról lesz majd szó. Részletezni fogom azokat az eszközöket és technológiákat, melyeket a fejlesztés során használtam és alkalmaztam.

#### 2.1.1. Laravel

A Laravel egy nyílt forráskódú, PHP webkeretrendszer, ami MVC tervezési mintán alapszik. Az MVC a Model-View-Controller rövidítése. Ez az egyik legősibb tervezési minta<sup>1</sup>.

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, kontroller): a felhasználói műveletek megfelelő kezeléséért ez a réteg a felelős.

Az elképzelése röviden az, hogy van egy modell és ha ez módosul, akkor erről értesíteni kell az összes nézetet. Minden nézet pontosan ugyanazon a vezérlőn tudja módosítani a modellt. 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 nézete is. Pozitívumok közzé sorolható az még, hogy számos

---

<sup>1</sup> A tervezési minták a gyakori problémákra nyújtanak kiforrott, általános megoldást.

alapvetőnek tekinthető funkcióhoz lehetséges használni néhány szükséges parancs megadásával. Például a webes alkalmazásomban használtam a *Laravel Breeze*-hez<sup>2</sup> tartozó funkciókat. Ezzel különféle autentikációhoz kapcsolódó funkciókat lehet használni, ebbe beletartozik például a bejelentkezés, a regisztráció és a felhasználóval kapcsolatos adatok módosítása is. Még használtam egy olyan parancs által megadható funkciót is, ami a lapozás megvalósítását teszi lehetővé. Ezeknek a pozitívumoknak köszönhetően szerintem a Laravel az nagyon megkönnyítheti a fejlesztők számára a munkavégzést. [2, 1]

### 2.1.2. PHP

A PHP egy szerveroldali szkriptnyelv, amelynek a segítségével dinamikus weboldalakat lehetséges 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 szervert 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.

[13]

### 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. Az eltérő táblákban szereplő mezők között lehetnek különféle kapcsolatok is. Ilyen lehet például az egy-az-egyhez kapcsolat vagy egy-a-többhöz reláció. A MySQL adatbázisok szerverére jellemző, hogy gyorsak, megbízhatóak és skálázhatóak. [5]

### 2.1.4. PhpMyAdmin

A phpMyAdmin egy ingyenes szoftver, amely lehetővé teszi a MySQL adatbázis adminisztrációját webes felületen keresztül. A phpMyAdmin segítségével elvégezhetők a legtöbb adminisztratív feladatok, ideértve az adatbázis létrehozását, lekérdezések futtatását és felhasználói fiókok hozzáadását. [6]

---

<sup>2</sup> Ez tulajdonképpen egy alap funkciókat tartalmazó csomag.

### 2.1.5. XAMPP

Az XAMPP nem más, mint egy PHP fejlesztői környezet. Az XAMPP egy angol mozaikszó, melyben az X platformfüggetlenséget jelenti, A az Apache-t jelöli, M a MySQL-t fejezi ki, az egyik P a PHP utal, míg a másik P pedig a Perl-t jelöli. Az Apache az nem más, mint egy webszerver. Az XAMPP rendszerben található Apache webszerver, ami csak helyi fejlesztést tesz lehetővé számunkra. A Perl pedig egy programozási nyelv. Az XAMPP-ról még azt érdemes tudni, hogy ingyenesen használható és rendkívül egyszerű telepítése, illetve a használata. [15]

### 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. Ezeken kívül pedig tartalmaz még beépített parancssort is, ami nagyon kényelmesé teszi ennek az IDE-nek a használatát.

### 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 és arra is, hogy úgynevezett *pull requestek*-nél<sup>3</sup> a nálunk magasabb pozícióban lévő munkatárs leellenőrizze az elkészült munkánkat és, ha helyesnek gondolja csak akkor kerül be a *main branch*-be. 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 munkavégzést, illetve azért is, mert ennek a használatához nem kell a terminálban beírni a

---

<sup>3</sup> A pull request azt jelenti, hogy valaki a saját ágának (branch) módosításait egy másik ágba (általában a fő ágba) szeretné beolvasztani és ezt egy magasabb pozícióban lévő fejlesztő beolvaszthatja vagy elutasíthatja azt.

Git-hez tartozó parancsokat, hanem egy-két kattintás segítségével könnyedén elvégezhetek egy konkrét parancsot. [7]

### 2.1.9. HTML

A HTML egy angol eredetű mozaikszó, aminek a jelentése HyperText Markup Language (magyarul: hiperszöveges jelölőnyelv). Ezt a nyelvet weboldalak készítéséhez szokás használni. Az én projektemben ezt a nyelvet a blade-ek készítésénél használtam. A bladek-ről többet megtudhat az 5.1.6. alfejezetben.

### 2.1.10. Tailwind CSS

Tailwind CSS használatával a felhasználói felületeket lehet kialakítani, ez csak külsőleges formázást jelent, tehát csak és kizárólag esztétikai szépítésre szolgál. Érdekes még róla tudni, hogy gyors, megbízható, rugalmas és nincsen futási ideje, azaz azonnal megtörténik a változás, nem kell rá várni semmit sem. A fejlesztés során néhány a webalkalmazásomban található gomb kinézetét, figyelmeztetések és információk megjelenítése esetén megjelenő stílust használtam fel, ezeknek a forrása megtekinthetők az irodalomjegyzékben [14, 18]

### 2.1.11. JavaScript

A JavaScript az egyik szkript- vagy programozási nyelv, aminek a segítségével van lehetőség bonyolultabb funkciók megjelenítésére a weboldalakon. Lehetővé teszi például egy dinamikusan frissülő tartalom létrehozását vagy akár képek animálását is. Ezen kívül még nagyon sok mindenre használható még. A projektemben használtam a JavaScript-et ahhoz, hogy amikor vármegyét választok ki egy legördülő listából, akkor egy másik listában csak az előzőleg kiválasztott vármegyébe tartozó város jelenjenek meg. A másik ahol szintén használtam az pedig a szűréshez tartozó részeket jelenítem meg vagy nem, annak a tükrében, hogy be van pipálva egy *checkbox* (magyarul: jelölőnégyzet) vagy nem.

## 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 írónak a tesztesetek.
2. *Szürkedobozos* (angolul: grey-box): amikor a forráskódnak csak egy rész ismert és csak ez alapján írónak a tesztesetek.

Azonban van egy harmadik fajta is, amit nem alkalmaztam. Az nem más, mint a *feketedobozos* (angolul: black-box), amikor a tesztesetek a specifikáció alapján írónak. [2]

### 3.1. A teszteléshez alkalmazott eszközök és technológiák

A következő alszakaszokban a teszteléshez használt eszközökről és technológiákról lesz majd szó. Kitérek majd arra is, hogy pontosan mit is jelentenek, miket lehet velük tesztelni és az előnyük és hátrányaikról is ejtek majd pár gondolatot.

#### 3.1.1. Manuális teszt

Manuális teszt az azt jelenti, hogy a tesztelő végig próbálgatja saját magától a funkciókat és leellenőrzi azok működését, illetve helyességét. Az ilyenféle tesztelés végzésekor nem alkalmazunk semmilyen automatizált tesztet. Én fejlesztés közben folyamatosan végeztem manuális tesztelést, abból kifolyólag, hogy megbizonyosodjak az elkészült funkció helyességéről, illetve, hogy feltárjam vele az esetleges hibákat, amiket nem vettem észre vagy még nem gondoltam rá. Az én meglátásom szerint a manuális tesztelésnek a legnagyobb hibája az, hogy minden lehetséges kombinációt ki kell próbálnunk

ahhoz, hogy bebizonyosodjunk arról, hogy a készített funkció a megfelelő módon működik. Még szintén hátrány lehet az, hogy rendkívül költséges lehet a manuális tesztelés. Végül pedig az, hogy az emberek nagyobb százalékban hibáznak, ami érthető is, hiszen tévedni emberi dolog. Azonban vannak előnyei is, ezek pedig véleményem szerint a következők lennének. Az első ilyen az lenne, hogy, ha van valamilyen hiba, akkor azt nagy valószínűséggel hamarabb észre lehet venni, amivel pénzt és időt is lehet spórolni. A másik ilyen előny az pedig, hogy a tesztelőnek van megfelelő intelligenciája ahhoz, hogy kiszűrje az esetleges hibákat, ellenben az automatizált tesztekkel szemben.

### 3.1.2. Cypress

A Cypress egy NodeJS-ben<sup>1</sup> írt frontend 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. Nézzük meg néhány előnyét és hátrányát is ennek használatára. Kezdjük a negatívumokkal először. A teszteknek JavaScript nyelven kell íródniuk. Erről többet megtudhat itt a 2.1.11. Hátrány még az is, hogy egyszerre csak egy ablakban vagy csak egy böngészőben lehetséges a tesztek végrehajtása. Most nézzük meg a pozitívumait is. Az egyik legnyilvánvalóbb ha ránézzünk egy ilyen kódra az, hogy nagyon olvasható bárki számára. Előny még az is, hogy igazi böngészőt használ a tesztek végrehajtására. Ez azért hasznos, mivel a valódi végfelhasználók is igazi böngészőkben fogják használni az alkalmazást, ezért jól lehet velük szimulálni a működését. Még előnynek tekinthető

---

<sup>1</sup> Ez egy ingyenes, nyílt forráskódú futtatókörnyezet, amellyel webes alkalmazásokat, szervereket is lehetséges létrehozni.



az is, hogy a hálózatról érkező kéréseket lehetséges ellenőrizni a Cypress segítségével. [4, 10, 11, 12, 16]

A szemléltetés kedvéért szeretné, Önnek bemutatni, hogy hogyan is kell elképzelni egy ilyenféle tesztet. A 3.1. kódon megtekinthető.

### 3.1. kódrészlet. Egy Cypress teszt egy hirdetés hozzáadására

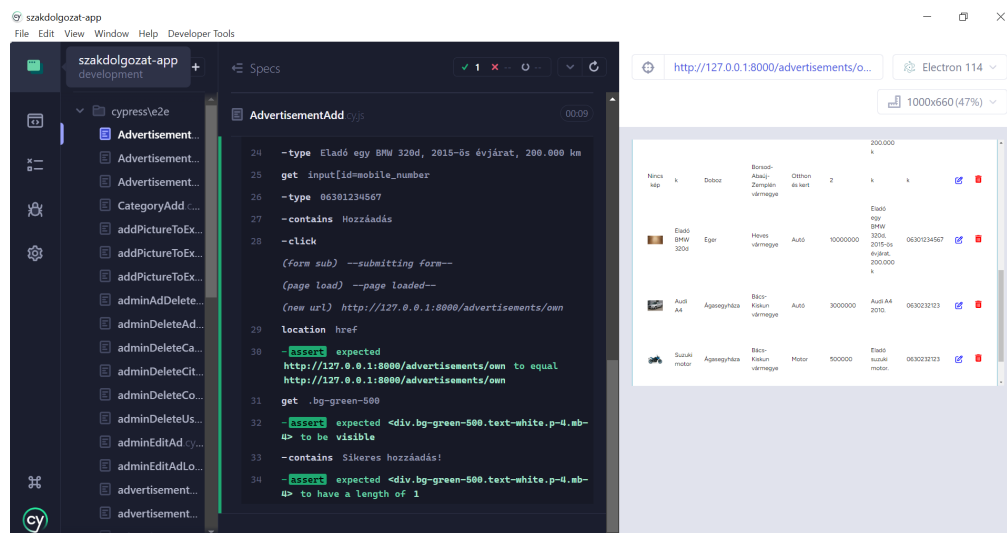
```
1 describe('template_spec', () => {
2   it('passes', () => {
3
4     cy.visit('http://127.0.0.1:8000/advertisements/create')
5
6     cy.get('input[id=email]').type('toth.jozsef@gmail.com')
7     cy.get('input[id=password]').type('password')
8     cy.contains('Bejelentkezés').click()
9
10    cy.get('select[id=countySelect]').select('Heves_vármegye')
11    cy.get('select[id=citySelect]').select('Eger')
12    cy.get('select[id=category]').select('Autó')
13
14    const picture1 = 'background_image.png'
15    const picture2 = 'background.jpg'
16    cy.get('input[id=pictures1]').attachFile(picture1)
17    cy.get('input[id=pictures2]').attachFile(picture2)
18
19    cy.get('input[id=title]').type('Eladó_BMW_320d')
20    cy.get('input[id=price]').type('10000000')
21    cy.get('label[for=description]').next('textarea').type('Eladó_egy_
    ↳ BMW_320d,_2015-ös_évjárat,_200.000_km');
22    cy.get('input[id=mobile_number]').type('06301234567')
23
24    cy.contains('Hozzáadás').click()
25
26    cy.location('href').should('eq', 'http://127.0.0.1:8000/
    ↳ advertisements/own')
27    cy.get('.bg-green-500').should('be.visible').contains('Sikeres_
    ↳ hozzáadás!').should('have.length', 1);
28
29  })
30 })
```

Itt a *visit* (magyarul: meglátogat) metódus segítségével lehetséges egy konkrét útvonalra eljutni. Ezután bejelentkezni szükség, mivel csak akkor van lehetőség hirdetés hozzáadására. Id, azaz valamilyen azonosító alapján szükséges megkeresni a *form*-on az adatok megadásának a helyét. Ahol mi adunk meg valamilyen információt, ott a *type* (magyarul: begépel) metódus segítségével tehetjük meg. Ahol legördülő menüből kell kiválasztani a szükséges adatokat, ott pedig a *select* (magyarul: kiválaszt) metódussal

lehetséges. Ezen kívül még két darab kép hozzáadása is látható még, ezt az *attachFile* (magyarul: fájl csatolása) metódussal történik, azonban ehhez szükséges még egy csomagot letölteni, de ezt csak megemlítettem, mivel többet erről nem lesz szó. Gombokat lehetséges szöveg alapján megkeresni és utána rákattintani, ez pedig a *contains* (magyarul: tartalmaz) és *click* (magyarul: rákattint) metódusokkal történhet meg. Végezetül pedig egy átirányítása megvalósítása látszik, ha minden sikerrel zárult és leellenőrzöm azt, hogy látszik-e majd az átirányított oldalon a megjelenítendő üzenet. Tehát véleményem szerint kijelenthető, hogy Cypress tesztek írása egyszerű és nagyon érthető még akár egy hétköznapi ember számára is.

### 3.1.3. Tesztek végrehajtása

A 3.1. ábrán az látható, hogy egy hirdetés hozzáadása teszt végrehajtása megtörténik. Ez a tesztet a 3.1. kód alapján hajtódik végre. A bal oldali sávon az elkészített tesztek láthatóak, középen maga a tesztet lépésről lépésre való lefutása tekinthető meg, jobb oldalon meg a konkrét folyamat szintén lépésről lépésre tekinthető meg, ahogyan a webböngészőben végrehajtódik. Van arra lehetőségünk, hogy a tesztben leírt konkrét lépéseket megtekintsük egyesével, ehhez csak annyit szükséges tennünk, hogy az egeret a középen lévő valamelyik kívánt információra rávisszük. Ha például magára a hirdetés hozzáadására vagyunk kíváncsiak, akkor a 28. sorban látható *click* szóra szükséges rávinni az egeret.



3.1. ábra. Hirdetés hozzáadása teszt végrehajtása (Saját készítés)

### 3.1.4. Tesztek eredménye

Az elvégzett Cypress tesztek eredményessége alapján megállapítható, hogy jelentős mértékben sikeresek voltak. Néhányszor előfordult, hogy bizonyos esetek eleinte nem

feleltek meg a teszteseteknek, de végül sikeresnek bizonyultak. Összességében a véleményem szerint a tesztek nagy sikerrel javították az apróhirdetés webalkalmazásom megbízhatóságát.

## 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. Az alkalmazásomban található ikonokat egy interneten lévő oldalon találtam meg, az irodalomjegyzékben megtalálható a keresett oldal.[17]

### 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. A törlendő felhasználók megtalálása miatt egy keresés funkciót is implementáltam bele, aminek a segítségével könnyebben megtalálható a keresett felhasználó.

#### 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 4.1. 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		

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		

4.1. ábra. Kategóriák és Vármegyék (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 4.1. 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, új képet nem tud hozzáadni az admin, csak és kizárólag törölni képes. Az adminisztrátor jogosultsággal rendelkező felhasználók nem képesek a vármegye, város megváltoztatására, hiszen ezt a hirdetés tulajdonosának a feladata, amennyiben szüksége lenne rá.

## 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 a felhasználó, legyen szó akár csak azok megtekintéséről, módosításairól vagy törléseiről. Ő már képes új hirdetések létrehozni, ami a 4.2. ábrán lesz látható. A 4.2. ábrán magyarázata következik. A vármegyét egy legördülő listából lehet

The form is titled 'Új hirdetés hozzáadása (Saját készítés)'. It contains the following fields:

- Vármegye:** A dropdown menu with the placeholder text 'Válassz vármegyét'.
- Város:** A dropdown menu with the placeholder text 'Válassz várost'.
- Kategória:** A dropdown menu with the placeholder text 'Válassz kategóriát'.
- Kép 1:** A file upload field with a 'Fájl kiválasztása' button and the text 'Nincs fájl kiválasztva'.
- Kép 2:** A file upload field with a 'Fájl kiválasztása' button and the text 'Nincs fájl kiválasztva'.
- Kép 3:** A file upload field with a 'Fájl kiválasztása' button and the text 'Nincs fájl kiválasztva'.
- Cím:** A text input field.
- Ár:** A text input field with the placeholder text 'Ft'.
- Leírás:** A large text area for the description.
- Telefonszám:** A text input field.
- Hozzáadás:** A green button to submit the form.

4.2. ábra. Új hirdetés hozzáadása (Saját készítés)

kiválasztani, ha ezt megtettük csak utána tudjuk majd a várost kiválasztani szintén legördülő listából. A hirdetés kategóriáját is ilyenféle módon választhatjuk ki. A 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 lehetősége, 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.

Keresés cím alapján Keresés

☒ Szűrés megjelenítése

Város: Válassz várost

Kategória: Válassz kategóriát

Vármegye: Válassz vármegyét

Minimum ár: Ft

Maximum ár: Ft

Rendezés: Alapértelmezett

Alkalmaz

4.3. 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 maguk által korábban elküldött üzeneteket is. Itt a megkülönböztetés és a felismerés céljából az a felhasználó, aki adminisztrátor jogosultsággal rendelkezi, akkor annak zöld színnel jelenik meg a neve. Ezáltal a sima felhasználók tudhatják, hogy valóban egy ilyen jogosultságú felhasználótól kaptak szöveges üzenetet.
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 és kizárólag törölni tudja őket.
6. *Saját hirdetések listázása*: a felhasználó által létrehozott hirdetéseket megtekintésére, módosítására és törlésére ezen a helyen van lehetőség rá. Itt szinten van ara mód, hogy keresni tudjunk a saját hirdetéseink között, ez a hirdetés neve alapján történhet meg.
7. *Hirdetések listázása kategóriák alapján*: ez lenne azaz oldal, amit a felhasználók elsőnek látnának meg, tulajdonképpen ez a főoldal. Ez hasonló, mint a szűrés funkció, azonban itt a kategória nevére rákattintva már rögtön megjelenik az adott kategóriába tartozó hirdetéseket.

## 5. fejezet

# Fejlesztői dokumentáció

A fejezetben szó lesz az alkalmazásom felépítéséről, ezen kívül még szót ejtek majd a jogosultságkezelésről is, hogy hogyan oldottam meg és milyen lehetőségeim voltak a fejlesztés során.

### 5.1. Az alkalmazás felépítése

Ebben az alfejezetben szeretném mutatni Önnek az alkalmazásom fő komponenseit kódrészletek segítségével a könnyebb megértés reményében. Egy nagyon keveset már megemlítettem korábban a 2.1.1. alfejezetben, de nem mindent, most lépésről lépésre elmesélem, hogy hogyan lehetséges megjeleníteni egy konkrét információkat 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 állítottam be. 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 hirdetéshez tartozó migrációs fájl.

5.1. kódrészlet. A hirdetéshez tartozó migráció

```
1 Schema::create('advertisements', function (Blueprint $table) {
2     $table->id('advertisement_id');
3     $table->unsignedBigInteger('user_id');
4     $table->unsignedBigInteger('city_id');
5     $table->unsignedBigInteger('category_id');
6     $table->string('title');
7     $table->integer('price');
8     $table->text('description');
9     $table->string('mobile_number')->nullable();
10    $table->timestamps();
11
12    $table->foreign('user_id')->references('user_id')->on('users')
13        ↳ ->onDelete('cascade');
14    $table->foreign('city_id')->references('city_id')->on('cities')
15        ↳ ->onDelete('cascade');
16    $table->foreign('category_id')->references('category_id')->on('categories')
17        ↳ ->onDelete('cascade');
18 });
```

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 meg lehet 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 *Eloquent*-et, ami egy ORM (angolul: 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.[1]

5.2. kódrészlet. A hirdetéshez tartozó modell

```
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', '
                ↪ advertisement_id');
23     }
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, amiknek az értékük nem lehet üres. 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. Egy modell létrehozása a következő parancs segítségével történhet meg:

*php artisan make:model név .*

#### 5.1.4. Kontroller

A kontrollerek tulajdonképpen arra használatosak, hogy a különböző kéréséhez szükséges metódusokat itt definiálhatjuk. A kontrollerekben általában sokféle metódusokat írhatunk, azonban alap esetekben a következő lehetnek. Lehet index nevű, ami az összes adat vagy más néven erőforrások megjelenítésére szolgál, lehet még create (magyarul: létrehozás), ami az adatok bekéréséért felelős *form*, ehhez szorosan kapcsolódik a *store* (magyarul: tárol, mentés), ami pedig elmenti a kívánt erőforrást. Beszélhetünk még a *show* (magyarul: megnéz) és a *destroy* (magyarul: törlés) metódusokról, az előbbi egy konkrét erőforrást jelenít meg, míg utóbbi segítségével az erőforrást lehetséges kitörölni. Végezetül még két metódus van, az első az *edit* (magyarul: módosítás), ami a *create* metódushoz hasonlóan egy *form*-ot jelenít meg, aminek a segítségével lehetséges az adott erőforrás módosítása. A másik pedig az *update* (magyarul: módosítás), ez pedig konkrétan módosítja az imént említett *edit form*-on az eszközölt változtatásokat. Példa képen a show metódust mutatom be Önnek. [1]

5.3. kódrészlet. A hirdetéshez tartozó kontrollerben lévő metódus

```
1 public function show($id)
2 {
3     $advertisement = Advertisement::find($id);
4
5     if (!$advertisement)
6     {
7         return redirect()->route('advertisements.index')->with('error', '
           ↳ Nincsen ilyen hirdetés!');
8     }
9
10    return view('advertisements.show', compact('advertisement'));
11 }
```

A következőben röviden megmagyarázom az 5.3. kódot. Látható, hogy a hirdetéshez tartozó kontrollerben található a metódus, ami paraméterként kér egy id-t, az id

segítségével tudom, hogy pontosan melyik hirdetést szeretném megjeleníteni. Ezután megkeresem az adott bekért id alapján a hirdetést, majd leellenőrzöm, hogy egyáltalán létezik-e, ha nem akkor hibát adok vissza, amit majd a blade tudunk majd felhasználni, ahhoz, hogy értesítsem a felhasználót a hibáról. Végül pedig, ha nincsen hiba, akkor megjelenítem egy másik blade-n a felhasználó számára. A *compact* nevű metódusra, azért van szükségem, hogy a blade-n majd megjeleníteni tudjam a kívánt hirdetést. A bladek-ről többet az 5.1.6. alfejezetben tudhat meg. Egy kontroller létrehozása a következő parancs segítségével történhet meg:

```
php artisan make:controller névController .
```

### 5.1.5. Web.php

A web.php nevű fájl, arra szolgál, hogy a kontrollerekben elkészített különböző metódusokhoz *route*-t (magyarul: útvonal) rendeljük hozzá. Ez azért szükséges, hogy a böngészőkben eltudjuk érni magát a konkrét blade-t, ami a kontrollerben definiált metódust fogja megvalósítani. Itt még annyit érdemes megemlítenem, hogy a *middleware* (magyarul: köztes szoftver) segítségével van lehetőség szabályozni, azt hogy ki férhessen hozzá az adott útvonalhoz. A következőben pedig egy útvonalat fogok mutatni, hogy eltudják képzelni, hogy hogyan is nézz ki.

5.4. kódrészlet. A web.php szereplő show metódus útvonala

```
1 Route::get('/advertisements/{advertisementId}/show', [  
    ↪ AdvertisementController::class, 'show'])->name('advertisements.  
    ↪ show');
```

Röviden az látható az 5.4. kód részleten, hogy a *route* és két darab kettőspont után van lehetőségünk a kérést típusát megadni, attól függően, hogy melyik metódust hívjuk meg. Jelen esetben most *get* (magyarul: kap) található ott, ami tulajdonképpen a listázásért felelős. Gyakran használatosak még a *post* (magyarul: hozzáadás, beszúrás), a *put* (magyarul: módosítás) és a *delete* (magyarul: törlés) elnevezések. Ezt követően következik az első paraméter, ami az elérési útvonal, a másik pedig azon az útvonalon található metódust lesz. Még megtalálható a végen ott egy *name* (magyarul: elnevezés) metódus, aminek a segítségével könnyebben tudunk majd hivatkozni a kívánt blade-n. Ennek a használata azért előnyös, mivel így nem kell a teljes útvonalat megadni, ha hivatkozni szeretnénk a metódusra és ha esetlegesen változtatni kellene az útvonalon, akkor azt csak egy helyen elég megtenni, itt a web.php-ben. Ha nem neveznénk el és több blade is használná ezt az útvonalat, akkor mindenhol meg kellene változtatni, ami nem lenne túl szerencsés. A bladek-ről többet az 5.1.6. alfejezetben tudhat meg.

Megközelítőleg az alkalmazásomban ötven darab útvonal található meg, és ezek közül számosban fennlelhető valamilyen id, azaz azonosító. Erre azért van szükség,

hogy a különböző erőforrásokat azonosítani lehessen és azért is, hogy ezeknek a segítségével valamilyen műveletet is lehessen rajtuk végezni. Ide tartozhat a lekérdezés, a hozzáadás, a módosítás és a törlés műveletek. Fontosnak tartom még azt kiemelni, hogy a webböngészőben az URL-ben<sup>1</sup> lévő azonosítókat lehet módosítani és ennek a dolognak a kivédése érdekében megfelelő lépéseket megtettem. Amennyiben, hogyha a felhasználó például a saját hirdetését kívánja módosítani, akkor az URL-ben láthatja a hirdetés azonosítóját és, ha ezt megváltoztassa, akkor már egy másik hirdetéshez tartozó módosítása nézetet tekintené meg, ami nagy valószínűséggel egy másik felhasználóhoz tartozik, de ez nem történhet meg mivel raktam a programomba az ilyenféle hibák miatt hibakezelést. Ezt átirányítás és megfelelő hiba üzenet segítségével tudatom a felhasználóval. Ha a felhasználó egy nem létező URL-t adna meg, akkor pedig szintén ugyanazokat lépéseket teszem meg, mint az előző, az azonosítóval kapcsolatos problémánál. Amennyiben az adott felhasználó olyan útvonalat ad meg amihez nincsen meg a kellő jogosultsága, akkor vagy szintén átirányítás és hiba üzenet segítségével tudatom vele, vagy pedig, akkor átirányítom a bejelentkezés felületre, mivel ekkor ő még csak egy sima nézelődőnek tekinthető. A bejelentkezés elvégzése után, kellő jogosultság birtokában már képes a kívánt művelet elvégzésére, amennyiben nem adminisztrátor tartozó műveleteket szeretne elérni, kivéve, ha az adott felhasználó már rendelkezik az imént említett jogosultsági szinttel.

#### 5.1.6. Blade

A blade-k segítségével azokat nézeteket tudjuk definiálni, amiket látni lehet a böngészőben. Tulajdonképpen ez lenne a UI (user interface), azaz a felhasználói felület. Egy hirdetés megjelenítéséért felelős felület szeretném bemutatni, amiről már szó esett az előző két alfejezetben. A 36. oldalon megtekinthető a kód.

Erről csak röviden szeretnénk beszélni, rögtön az elején van ként részlet, ami említésre méltó. Az *extends* (magyarul: kibővít) kulcsszó segítségével kitudjuk bővíteni egy másik felületet, a másik kulcsszó pedig a *section* (magyarul: szekció), ami pedig a kibővített felületen belüli *content* (magyarul: tartalom) helyére kerül be a taglalt felület. Táblázatos formában jelenítettem meg a hirdetést, és ennek a Tailwind CSS (2.1.10) segítségével való formázás látható még ott. Az 5.1.4. alfejezetben esik szó a *compact* nevű metódusról, aminek a használata azért volt szükséges, hogy elérjük a *\$advertisement* (magyarul: hirdetés) változót. Még egy dolgot szeretnénk kiemelni, ami pedig a 36. sorban olvasható. Erről már volt szó ebben az 5.2. alfejezetben. Itt a modellek miatt tudom ezt megtenni, hogy így megtudjam jeleníteni a hozzá tartozó kategória nevét, ennek a hiányában a hirdetéshez tartozó kategória id-t tudnám csak megjeleníteni.

---

<sup>1</sup> Az angol uniform resource locator kifejezés rövidítése. Ez egy egyedi azonosító, amely egy erőforrást (például egy weboldalt) segít megtalálni az interneten.

## 5.2. Jogosultságkezelés

A jogosultságkezelés minden projektnél egy nagyon kulcsfontosságú, lényeges szerepet tölt be, mivel általában szükség van arra, hogy lehessen szabályozni a különböző funkciók használatát. A Laravel keretrendszer számos lehetőséget ad számunkra ezen a téren. Ilyen opció például a *middleware* (magyarul: köztes réteg) vagy a *policy* (magyarul: szabályzat). Én az utóbbi mellett tettem le a voksomat. Ennek a használata nagyon egyszerű és érthető volt számomra. A policy-ben egy adott modellhez vagy erőforráshoz tartozó autentikációs logikát szükséges implementálni. Egy ilyen fájl egy egyszerű parancs segítségével lehet generáltatni. Egy policy-ben azokat a metódusokat kell megírunk, amelyben szeretnénk valamiféle jogosultságkezelést használni. Minden metódushoz külön kell őket definiálni. Miután elkészült egy policy, akkor tudatni kell a Laravellel, hogy az adott modellhez valamilyen autentikációs megszorítást hajtottunk végre, ezért szükséges regisztrálni ezeket a fájlokat az úgynevezett *AuthServiceProvider* fájlba. Ezek után pedig már csak annyit kell megtennünk, hogy a vezérlőkbe ezeket implementálni kell, ez nagyon egyszerűen végrehajtható. Például a hirdetés módosítása metódus esetében így lehetséges ezt végrehajtani:

```
$this->authorize('edit', $advertisement).
```

Itt a *this* szó az aktuális objektumot jelöli, az *authorize* a metódus neve, és ez jelen esetben két paramétert vár. Az első az *edit*, ami a hirdetéshez tartozó *policy*-ben található metódus neve, amit szeretnénk most alkalmazni. A második paraméter meg maga a konkrét módosítani kívánt hirdetés. [1]

## 5.3. Seeder

A seeder-ek segítségével van lehetőség az adatbázisban lévő táblák adatait feltölteni gyorsan és egyszerűen. Én ezt a lehetőséget azért használtam, mivel például a manuális tesztelés elvégzése közben a hirdetések törlésének a tesztelését végeztem el és, ha ezek elfogytak, akkor egy parancs lefuttatásának a segítségével újra megjelentek az adatbázisomban, amivel sok időt tudtam megspórolni, mivel nem kellett egyenként hozzáadnom a hirdetéseket. Ezenfelül már rendelkezik éles, valós tesztadatokkal, ami az általam készített alkalmazásom bemutatása során hasznos lehet.

## 6. fejezet

# Üzleti modell

Az én meglátásom szerint az apróhirdetés weboldalak fenntartásához, működéséhez és karbantartásához szükség van némi anyagi forrásra. A következőben szeretném Önnnek bemutatni, hogy milyen megoldások segítségével tudnám én ezt elképzelni. Azt kiemelném, hogy ha a következő javaslatok közül némelyik opció implementálására az adatbázis alapú webalkalmazásom jelenlegi formájában még nem lenne alkalmas rá.

### 6.1. Online hirdetések

Tulajdonképpen ez az ötlet jutna a legtöbb ember számára eszébe, mint pénz szerzési lehetőség. Én is erre gondoltam legelőször, amikor ezt a kérdést tettem fel magamnak. A különböző hirdetések megjelenítését én előreláthatólag csak a főoldalon, egy hirdetés megtekintése oldalon és a saját hirdetések megnézése oldalon tenném lehetővé ezt az opciót.

### 6.2. Kiemelés

A kiemelés opció alatt azt értem, hogy lenne a felhasználóknak olyan lehetőségük, hogy némi pénz reményében a saját hirdetéseiket előresorolná, kiemelné az alkalmazás, legyen szó akár a hirdetések böngészésekor vagy például akkor amikor szűrünk vagy keressünk egy adott hirdetésre.

### 6.3. Együttműködések

Az együttműködés alatt én az érteném, hogy lenne olyan lehetőség az alkalmazásomban, hogy a különböző cégek termékeit lehessen reklámozni, és ezt úgy megjeleníteni, mint, ahogyan egy hétköznapi felhasználó saját hirdetését. Azonban pár változtatásra ilyenkor a véleményem szerint szükség lenne, ilyen változtatás lenne például az, hogy

az ilyenféle hirdetésekhez valamiféle információ közlést tennék, hogy a felhasználók annak a tudatában nézzék meg a hirdetést vagy vegyék meg azt, hogy nem egy átlagos embertől származik, hanem egy nagyobb cégtől.



# Összegzés

A véleményem szerint sikerült elérnem a kitűzött célt, miszerint egy apróhirdetés webalkalmazást hozzak létre, amely megfelel az általam felállított követelményeknek. Azonban, mint minden más alkalmazás esetében itt is adva van lehetőség a tovább fejlesztésre, valamint új funkciók hozzáadására is. A meglátásom szerint a következő funkciókkal lehetne fejleszteni tovább az adatbázis alapú webalkalmazásomat. Az első ilyen például az, hogy lenne egy kedvencek nevű rész is, ahová a felhasználók felvehetnék a saját meglátásaik szerint a hozzá közelálló hirdetéseket, ezáltal gyorsabban megtudnák találni az adott hirdetést, ha későbbiekben szükségük lenne rá. A másik ilyen fejlesztési opció az lenne, hogy a felhasználóknak lenne egy megbízhatóság nevű tulajdonságuk, aminek a segítségével ellenőrizni lehetne az adott felhasználó megbízhatóságát. Ennek a funkciónak a logikai implementálást alaposan át kellene gondolni a későbbiekben, mivel a meglátásom szerint összetett, bonyolult is lehet. Végül pedig amin még lehetne további módosítások elvégezni az nem más, mint a design. Az alkalmazás fejlesztése során csak néhány apró hibába botlottam, amiket szerencsére sikerült megoldanom, kivéve az elfelejtett jelszó funkciónál fellépő hibát. Többszöri próbálkozások ellenére sajnos nem sikerült megoldanom a fennálló problémát. Végeredményben úgy vélem, hogy az apróhirdetés webalkalmazásom tervezése és fejlesztése során elért eredmények és tapasztalatok alapján, hogy képes hatékonyan kielégíteni a felhasználók igényeit.

# Függelékek

## Követelménylista

<i>Id</i>	<i>Modul</i>	<i>Név</i>	<i>Leírás, megjegyzés</i>
K1	Jogosultság	Regisztráció	A webalkalmazásom teljes körű használatához regisztrációra van szükség.
K2	Jogosultság	Bejelentkezés	Email és jelszóval van lehetőség belépni. Hibák esetén visszajelzést kapnak a felhasználók.
K3	Jogosultság	Kijelentkezés	
K4	Jogosultság	Profillal kapcsolatos műveletek	A felhasználónak van lehetősége a nevét, email címét és jelszavát is módosítani, illetve törölni a saját profilját. Hibák esetén visszajelzést kapnak a felhasználók.
K5	Jogosultság	Jogosultsági szintek	<i>Admin, Felhasználó, Nézelődő.</i>
K6	Felület	Regisztráció	Egy név, email cím és a jelszó kétszeri megadása szükséges. Hibák esetén visszajelzést kapnak a felhasználók.
K7	Felület	Bejelentkezés	Email és jelszóval van lehetőség belépni. Hibák esetén visszajelzést kapnak a felhasználók.
K8	Felület	Hirdetések	Az összes hirdetés kilistázva való megtekintése.
K9	Felület	Hirdetések szűrése, rendezése, keresése	Az összes hirdetés kilistázva való megtekintése a szűrésre, keresésre vagy rendezésre. A keresés az a hirdetés neve alapján történik.
K10	Felület	Saját hirdetések	A saját hirdetések kilistázva való megtekintése, törlése.

Folytatódik a következő oldalon

táblázat folytatása

<i>Id</i>	<i>Modul</i>	<i>Név</i>	<i>Leírás, megjegyzés</i>
K11	Felület	Saját hirdetések módosítása	A hirdetés összes adatát lehetséges megváltoztatni, kivéve a hirdető nevét.
K12	Felület	Saját hirdetések keresése	A hirdetés neve alapján történik.
K13	Felület	Új hirdetés hozzáadása	
K14	Felület	Saját beszélgetések megtekintése	Az összes beszélgetés megtekintése, csak az utolsó üzenet látjuk.
K15	Felület	Saját üzenetek megtekintése, törlése	
K16	Felület	Saját üzenetek módosítása	
K17	Felület	Saját beszélgetések megtekintése	
K18	Felület	Kategóriák megtekintése, törlése	A törlés csak admin jogosultsággal lehetséges.
K19	Felület	Új kategória hozzáadása	Csak admin jogosultsággal lehetséges.
K20	Felület	Kategória módosítása	Csak admin jogosultsággal lehetséges.
K21	Felület	Vármegyék megtekintése, törlése	Csak admin jogosultsággal lehetséges.
K22	Felület	Vármegye hozzáadása	Csak admin jogosultsággal lehetséges.
K23	Felület	Vármegye nevének módosítása	Csak admin jogosultsággal lehetséges.
K24	Felület	Városok megtekintése, törlése	Csak admin jogosultsággal lehetséges.
K25	Felület	Városok hozzáadása	Csak admin jogosultsággal lehetséges.
K26	Felület	Városok nevének módosítása	Csak admin jogosultsággal lehetséges.

Folytatódik a következő oldalon

táblázat folytatása

<i>Id</i>	<i>Modul</i>	<i>Név</i>	<i>Leírás, megjegyzés</i>
K27	Felület	Felhasználók megtekintése, törlése	Csak admin jogosultsággal lehetséges. Az adminok itt nem kerülnek kilistázásra.
K28	Felület	Felhasználók keresése	Csak admin jogosultsággal lehetséges. Az adminok itt nem kerülnek kilistázásra. A felhasználók neve alapján történik a keresés.

## Egy hirdetés megjelenítésére szolgáló blade

```

1 @extends('layouts.app')
2 @section('content')
3     <div class="min-h-screen container mx-auto mt-8">
4         <div class="flex justify-center">
5             <div class="w-2/3">
6                 <div class="bg-white p-6 rounded-lg shadow-md">
7                     <table class="table-auto w-full">
8                         <thead>
9                             <tr>
10                                <th class="px-4 py-2">Kép</th>
11                                <th></th>
12                                <td class="px-4 py-2">
13                                    <div class="grid grid-cols-5 gap-2">
14                                        @foreach ($advertisement->pictures as
15                                            ↪ $i => $picture)
16                                            <a href="{{asset('storage/' .
17                                                ↪ $picture->src)}}" data-
18                                                ↪ title="Image_{{ $i+1 }}"
19                                                ↪ >
20                                            
22                                            </a>
23                                        @endforeach
24                                    </div>
25                                </td>
26                            </tr>
27                            <tr>
28                                <th class="px-4 py-2">Cím</th>
29                                <td class="px-4 py-2">{{ $advertisement->
30                                    ↪ title }}</td>
31                            </tr>
32                            <tr>
33                                <th class="px-4 py-2">Város</th>

```

```

28         <td class="px-4_lpy-2">{{ $advertisement->
           ↳ city->name }}</td>
29     </tr>
30     <tr>
31         <th class="px-4_lpy-2">Vármegye</th>
32         <td class="px-4_lpy-2">{{ $advertisement->
           ↳ city->county->name }}</td>
33     </tr>
34     <tr>
35         <th class="px-4_lpy-2">Kategória</th>
36         <td class="px-4_lpy-2">{{ $advertisement->
           ↳ category->name }}</td>
37     </tr>
38     <tr>
39         <th class="px-4_lpy-2">Ár</th>
40         <td class="px-4_lpy-2">{{ $advertisement->
           ↳ price }}</td>
41     </tr>
42     <tr>
43         <th class="px-4_lpy-2">Leírás</th>
44         <td class="px-4_lpy-2_break-all">{{
           ↳ $advertisement->description }}</td>
45     </tr>
46     <tr>
47         <th class="px-4_lpy-2">Eladó</th>
48         <td class="px-4_lpy-2">{{ $advertisement->
           ↳ user->name }}</td>
49     </tr>
50     <tr>
51         <th class="px-4_lpy-2">Telefonszám</th>
52         <td class="px-4_lpy-2">{{ $advertisement->
           ↳ mobile_number }}</td>
53     </tr>
54 </thead>
55 </table>
56 </div>
57 </div>
58 </div>
59 </div>
60 @endsection

```

# Irodalomjegyzék

- [1] LARAVEL: <https://laravel.com/docs/10.x>, megtekintve: 2024.04.01.
- [2] KUSPER GÁBOR: *Programozási technológiák*, Eger, 2015.
- [3] KUSPER GÁBOR: *Informatikai rendszerek tervezése*, Eger, megtekintve: 2023, 0.8.7.2-es verzió.
- [4] KUSPER GÁBOR: *Szoftvertesztelés*, Eger, 2023, megtekintve: 2024, 2023.10.14-es verzió.
- [5] ORACLE: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>, megtekintve: 2024.02.01.
- [6] PHPMYADMIN: <https://www.phpmyadmin.net/>, megtekintve: 2024.02.01.
- [7] GITHUB DESKTOP: <https://docs.github.com/en/desktop/overview/about-github-desktop>, megtekintve: 2024.02.01.
- [8] PLANTUML: *PlantUML Language Reference Guide*, <https://plantuml.com/guide>, megtekintve: 2024.02.03.
- [9] DBDIAGRAM: <https://dbml.dbdiagram.io/docs/>, megtekintve: 2024.02.03.
- [10] CYPRESS: <https://docs.cypress.io>, megtekintve: 2024.02.15.
- [11] KATALON: <https://katalon.com/resources-center/blog/end-to-end-e2e-testing>, megtekintve: 2024.02.15.
- [12] MEDIUM: <https://skakarh.medium.com/advantages-and-disadvantages-of-cypress-end-to-end-testing-tool-before-choosing-it-as-your-347b6436dec8>, megtekintve: 2024.02.15.
- [13] PHP: <https://www.php.net/docs.php>, megtekintve: 2024.02.17.
- [14] TAILWIND CSS: <https://tailwindcss.com/>, megtekintve: 2024.02.18.
- [15] SITE GYAAN: <https://www.linkedin.com/pulse/what-xampp-how-does-work-everything-you-need-know-server-site-gyaa>, megtekintve: 2024.03.23.

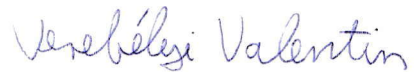
- [16] NODEJS: <https://nodejs.org/en>, megtekintve: 2024.03.28.
- [17] FONTAWESOME: <https://fontawesome.com/>
- [18] FLOWBITE: <https://flowbite.com/>
- [19] KSH: [https://www.ksh.hu/docs/hun/hnk/hnk\\_2015.xls](https://www.ksh.hu/docs/hun/hnk/hnk_2015.xls)
- [20] PIXTELLER: <https://pixteller.com/templates/custom-visuals/simple-background-backgrounds-passion-id1585825>
- [21] LOGÓ:
- <https://www.vecteezy.com/vector-art/6361809-car-icon-car-icon-vector-car-icon-simple-sign>,
  - <https://pngtree.com/free-png-vectors/clothes-icon>,
  - <https://hu.pinterest.com/pin/dog-logos--502292164691709590/>,
  - <https://www.vecteezy.com/vector-art/4416880-simple-house-icon-on-white-background>,
  - <https://images.vexels.com/media/users/3/157570/isolated/lists/4b39b362c76ea5a00de62f8ff839b5ed-simple-smartphone-icon.png>,
  - <https://hu.pinterest.com/pin/51650726955966410/>,
  - <https://hu.pinterest.com/pin/578923727087745719/>,
  - [https://static.vecteezy.com/system/resources/thumbnails/017/764/046/small\\_2x/eps10-black-line-art-sofa-abstract-icon-or-logo-isolated-on-white-background-living-room-furniture-outline-symbol-in-a-simple-flat-trendy-modern-style-for-your-website-design-and-mobile-app-vector.jpg](https://static.vecteezy.com/system/resources/thumbnails/017/764/046/small_2x/eps10-black-line-art-sofa-abstract-icon-or-logo-isolated-on-white-background-living-room-furniture-outline-symbol-in-a-simple-flat-trendy-modern-style-for-your-website-design-and-mobile-app-vector.jpg),
  - <https://depositphotos.com/hu/vector/hand-writing-soccer-ball-football-cartoon-12785922.html>

# Nyilatkozat

Alulírott *Verebélyi Valentin*, büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, *Adatbázis alapú web alkalmazás fejlesztése* című szakdolgozat önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Aláírással igazolom, hogy az elektronikusan feltöltött és a papíralapú szakdolgozatom formai és tartalmi szempontból mindenben megegyezik.

Eger, 2024. április 14.



aláírás