

Szakdolgozat

Csomor Benő

Debrecen

2023

Debreceni Egyetem

Informatika kar

Használt számítástechnikai eszközök értékesítését segítő weboldal

Témavezető:

Dr. Tiba Attila

Egyetemi Adjunktus

Készítette:

Csomor Benő

Programtervező Informatikus

Debrecen

2023

Tartalomjegyzék

1. Bevezetés	4
2. Irodalmi áttekintés	6
2.1 Html.....	6
2.2 CSS	6
2.3 Javascript	7
2.4 Bootstrap.....	8
2.5 MariaDB.....	9
2.6 PHP	10
2.7 JQuery.....	11
2.8 XAMPP	12
3. Az alkalmazás bemutatása	13
3.1 Specifikáció.....	13
3.1.1 Regisztráció - Bejelentkezés	13
3.1.2 Kijelentkezés.....	14
3.1.3 Hirdetés feltöltése	14
3.1.4 Hirdetés szerkesztése.....	14
3.1.5 Profil	15
3.1.6 Főoldal - Hirdetések közötti keresés	15
3.1.7 A hirdetés főoldala	16
3.1.8 Felhasználók közötti kommunikáció	17
3.1.9 Hirdetés jelentése	17
3.1.10 Admin oldal	18
3.1.11 Az oldalon való navigáció	19
3.2 Grafikus terv	20
3.2.1 Reszponzivitás	20
3.2.2 Az alkalmazás alapvető megjelenése	21
3.3. Adatbázis	25
3.3.1 Users tábla.....	25
3.3.2 Advertisement tábla.....	26
3.3.3 Ratings tábla.....	27
3.3.4 Messages tábla	28
3.3.5 Reports tábla	29

3.3.6 Táblák közötti kapcsolat	30
3.4 Implementáció	31
3.4.1 Lekérdezések	31
3.4.2 Regisztráció	32
3.4.3 Bejelentkezés	33
3.4.4 Kijelentkezés.....	33
3.4.5 Hirdetés feladása.....	34
3.4.6 Hirdetés szerkesztése.....	37
3.4.7 Profil	40
3.4.8 Felhasználók közötti kommunikáció	42
3.4.9 Főoldal – Hirdetések közötti keresés	45
3.4.10 Hirdetés főoldala	47
3.4.11 Hirdetések jelentése.....	48
3.4.12 Admin oldal	48
3.4.13 Az oldalon való navigáció	50
4. Tesztelés	51
5. Összefoglalás	52
6. Irodalomjegyzék	54

1. Bevezetés

A globalizáció és a technika egyre gyorsuló fejlődésével egyre szélesebb körben elérhetővé vált az internet, melyet napjainkban a bolygó lakosságának jelentős része használ a mindennapokban, akár személyes, akár üzleti célokra. Emiatt folyamatosan nő a piaci igény a webes alkalmazások felé, ma már nem csak a legnagyobb cégek rendelkeznek őket reprezentáló weboldallal, rengeteg magánvállalkozásnak, szolgáltatás elérhető pár kattintással az interneten keresztül, ami nagyban megkönnyíti a mindennapi életünket.

Szintén a modernizációnak, a technológia fejlődésének, és széles körben való elterjedésének köszönhető az, hogy napjainkban rengeteg háztartásban megtalálhatóak a technika újabbnál újabb vívmányai, a háztartási, konyhai gépektől kezdve egészen az asztali számítógépekig. Ezeknek az eszközöknek a fejlődésével egyre inkább nőni fog a keresletük is, hiszen nem csak megkönnyítik mindennapi életünket, de javítanak is az emberek életszínvonalán.

Ezen az úton elindulva egy olyan letisztult, könnyen kezelhető alkalmazás került kialakításra, amely elősegíti azoknak az embereknek az életét, akik használt számítástechnikai eszközeiket szeretnék értékesíteni, vagy éppen ilyen eszközöket szeretnének vásárolni. Az alkalmazás segítségével bárki könnyedén feltöltheti eladni kívánt eszközeit, megadhatja az azokkal kapcsolatos információkat vagy akár a megvásárolni kívánt termék kiválasztása után pár kattintással kommunikálhat a hirdetés feladójával. A dolgozat célja az előbb említett, és még megannyi funkció implementálása, hogy a weboldal minél könnyebben kezelhető legyen és minél nagyobb felhasználói élményt nyújtson. Ezen felül implementálásra kerül az alkalmazás működését lehetővé tevő adatbázis is, amelyben eltárolhatjuk, és amelyen keresztül elérhetjük a szükséges adatokat.

Az adatbázist, és annak a weboldallal való együttműködését egy XAMPP segítségével létrehozott MySQL szerverrel fogjuk tesztelni, az adatbázisból az adatok kinyeréséhez, illetve az alkalmazás mögött futó folyamatok implementálásához PHP-t és Javascript-et fogunk használni. A weboldal megjelenésének létrehozása, szerkesztése és a reszponzivitásának kialakítása pedig Html, CSS és Bootstrap segítségével fog történni.

2. Irodalmi áttekintés

2.1 Html

A Html az angol HyperText Markup Language kifejezés rövidítése, amely magyarul hiperszöveges jelölőnyelvet jelent [1]. Első verziója az 1990-es évek elején látott napvilágot, azóta folyik a fejlesztése, a legújabb verziója, a Html5 mely hivatalosan 2014-ben jelent meg. Weboldalak létrehozására fejlesztették ki, a W3C (World Wide Web Consortium) támogatásával mára internetes szabvánnyá vált, rengeteg webes alkalmazásnál használják [21].

A Html nem egy programozási nyelv, hanem egy kódnyelv, jelölőnyelv, amely segítségével létrehozhatjuk, definiálhatjuk egy webes dokumentum tartalmát, elrendezését. Segítségével egyszerűen hozzáadhatunk a weboldalunkhoz képeket, videókat és egyéb különböző elemeket is [2]. A Html fájlok általában szöveges állományok, melyek elérhetőek a böngésző számára egy szerveren, a benne lévő szimbólumokat a böngésző feldolgozza és ez alapján hozza létre a weboldalunkat.

2.2 CSS

A CSS (Cascading Style Sheet, magyarul: egymásba ágyazott stíluslapok) egy stílusleíró nyelv, amely a Html-hez hasonlóan webes szabvánnyá vált. Első verziója pár évvel a Html után jelent meg 1996-ban [3]. Létrehozásának fő oka az volt, hogy megnőtt az igény az egyre komplexebb, esztétikusabb kinézetű weboldalakra, amelyeket a Html egyre nehezebben tudott megvalósítani, minden elemnek egyenként kellett leírni a kinézetét, ezzel a Html elvesztette az egyszerűségét, átláthatóságát, ennek elkerülése érdekében hozták létre a CSS-t.

A CSS erőssége az, hogy elválasztja a weboldal megjelenését és a tartalmát, segítségével formázhatjuk weboldalunk vizuális megjelenését, akár egy külön állományban, anélkül, hogy a minden elem stílusát külön kellene definiálnunk [4].

A CSS segítségével a weboldalunk megjelenésének körülbelül minden tulajdonságát szerkeszthetjük a háttérszíntől, szövegmérettől, betűtípustól kezdve egészen a tartalom elhelyezkedéséig. Ráadásul egy CSS állományt használhat több Html fájl is, ezzel nagyban felgyorsítva egy nagyobb weboldal stílusának kialakítását.

2.3 Javascript

A Javascript egy olyan kliens oldali, objektumorientált programozási nyelv melyet kifejezetten webes használatra fejlesztettek ki. Első hivatalos kiadása 1997-ben jelent meg, akkor még „ECMAScript”-ként. Az ezt követő években rohamosan fejlődött, következő verziói, az ECMAScript 2 1998-ban, az ECMAScript 3 1999-ben, míg az ECMAScript 4 2000-ben jelent meg. Ezt követően egyre dominánsabb lett, sztenderdé vált a webes alkalmazások területén.

A Javascript más programozási nyelvekhez hasonlóan rengeteg dologra képes, létrehozhatunk benne változókat, függvényeket, osztályokat, tárolhatunk értékeket stb. Ezen felül Javascript segítségével rengeteg elemet, funkciót adhatunk a weboldalunkhoz, különböző események bekövetkezésekor változtathatjuk a weboldal megjelenését, tartalmát, ezzel is növelve a felhasználói élményt [5]. Lehetőségünk van arra, hogy olyan felhasználói műveleteket figyeljünk meg, mint például az elemekre való kattintások, vagy a weboldal átméretezése. Ezzel képesek vagyunk programozható elemeket elhelyezni a weboldalon, mint például egy gombot, melyre kattintás után lefut egy általunk definiált függvény.

Erre egy példa:

```
<button onclick=navigated(2)>Adatok mentése</button>

function navigated(arg){
    $.cookie('navigated', arg, { path: '/', expires: 10 });
}
```

A kódrészlet első fele egy Html gombot ábrázol, melyre való kattintás meg fogja hívni az onclick eseményt, vagyis le fog futni a Javascript segítségével definiált navigated függvény, amely létre fog hozni egy sütit navigated névvel és az argumentumban megadott értékkel.

Javascript kódot meg létre lehet hozni a Html fájlban belül, vagy a CSS-hez hasonlóan külön állományban, melyet importálás után akár több Html is használhat. Ahogy már említésre került, a Javascript kliens oldali, ami azt jelenti, hogy a felhasználó böngészőjében található Javascript motor fogja futtatni, így egyszerűen el tudja érni a felhasználó oldalán megjelenő Html elemeket, azok CSS tulajdonságait stb. [6]. Napjainkban már a legtöbb böngésző szoftver és modern okostelefon támogatja a Javascriptet.

2.4 Bootstrap

A Bootstrap egy ingyenes, nyílt forráskódú, kliens oldali Html, CSS és Javascript technológiákat használó keretrendszer. Először 2011-ben tették nyilvánosan is elérhetővé, legfrissebb verziója, a Bootstrap 5 2021-ben jelent meg. Napjainkban ez az egyik legelterjedtebben használt webes keretrendszer.

A Bootstrap előre definiált komponenseket tartalmaz, mint például gombok, táblázatok, képmegjelenítők, melyek különböző verzióit bárki használhatja, beépítheti a weboldalába. A komponensei pedig könnyen elérhetőek a keretrendszer letöltése nélkül is, a Html állományban hivatkozó link segítségével is használhatóak.

Azért használják olyan sokan, mert számos igényre, problémára kínál megoldásokat komponenseivel, melyek megfelelő Html, CSS és Javascript tudással átszerkeszthetőek, hogy azok megfeleljenek az adott weboldal elvárásainak. Emellett előnye, hogy a különböző verziói között nincs hatalmas különbség, egy másik verzióra váltáshoz nem kell újra tanulni a keretrendszer használatát [8]. Ezen felül a felkínált komponensek jól dokumentáltak, így könnyedén beépíthetőek bármilyen webes alkalmazásba, és reszponzívok, vagyis bármekkora oldalméret esetén esztétikus lesz a megjelenítése [7].

A projektben a Bootstrap 4-ből került felhasználásra és szerkesztésre egy képmegjelenítő, a Carousel.

2.5 MariaDB

A MariaDB egy nyílt forrású, SQL alapú relációs adatbázis kezelő rendszer vagy más néven RDBMS (Relational Database Management System). A MariaDB a MySQL egy továbbfejlesztett változata, melyet a MySQL eredeti fejlesztői készítettek 2009-ben, amikor az Oracle felvásárolta a MySQL-t és fizetőssé tette annak egyes változatait [12]. A MariaDB több funkcióval rendelkezik, mint a MySQL, emellett hatékonyabb, gyorsabb és megbízhatóbb is annál [9]. A két alkalmazás nagyban kompatibilis egymással, ezzel egyszerűsítve a MariaDB-re való átállást [10].

A MariaDB-t napjainkban elterjedten használják, olyan nagyvállalatok is alkalmazzák, mint például az Ubuntu, a Google, a Wikipedia, vagy a Tumblr. Rengeteg alkalmazás esetében szükség lehet arra, hogy különféle adatokat kezeljünk, a MariaDB ezt lehetővé teszi, segítségével különböző adatokat tárolhatunk, elérhetünk, módosíthatunk, törölhetünk. Emellett a tárolt adataink relációs adatmodell alapján vannak szervezve, ezzel átláthatóvá, könnyen kezelhetővé téve a rendszert. Népszerűségéhez hozzájárul az is, hogy a MariaDB több mint 20 platformon használható.

Az, hogy SQL alapú, az azt jelenti, hogy ennek a nyelvnek a segítségével kommunikálhatunk az adatbázissal, kezelhetjük a benne tárolt adatainkat. Az SQL (Structured Query Language) tehát egy olyan lekérdezési nyelv, amely hasonlít egy programozási nyelvhez, de annál könnyebben olvasható és megérthető [11]. Mára nemzetközi szabvánnyá vált, melyet számos adatbázis kezelő rendszer használ.

Egy példa az egyik legelterjedtebben használt SQL utasításra:

```
SELECT username, email, register_date FROM `users` WHERE id=1
```

username	email	register_date
ADMIN	hargera@gmail.com	2023-03-04 17:08:22

Ilyen módon elkérhetjük az 1-es azonosítójú felhasználó nevét, email-címét és a regisztrálásának dátumát.

2.6 PHP

A PHP egy nyílt forráskódú, szerveroldali szkriptnyelv mely először 1995-ben látott napvilágot, azóta folyamatos fejlesztés alatt áll, legfrissebb verziója, a PHP 8.2 2022-ben jelent meg. A PHP szó jelentése eredetileg Personal Home Page Tools volt, ami arra utal, hogy személyes weboldalak karbantartására használtak [13]. Ez az azóta megváltozott, ma már Hypertext Preprocessor a rövidítés jelentése, ami rekurzivitásával arra utal, hogy a PHP folyamatos fejlődésével önállóan használható programozási nyelvvé vált, melyet nagyobb webes alkalmazások üzemeltetésére is használnak. Napjainkban rengeteg weboldal funkcionalitása PHP-ra épül [15].

Az, hogy szerveroldali szkriptnyelv, azt jelenti, hogy a műveletek végrehajtására képes, melyek nem a kliens (felhasználó) oldalán, hanem a weboldalt vagy programot futtató szerver oldalán fognak lefutni. Ilyen például egy bejelentkezés, melyhez a szükséges adatok nem a felhasználó eszközén érhetőek el, hanem egy szerveren, melynek küldünk egy kérést, és sikeres beazonosítás esetén beléptetjük a felhasználót a weboldalra [14].

Egy ilyen, vagy ehhez hasonló művelet előtt csatlakoznunk kell az adatokat tároló adatbázis szerverhez, ami PHP-ban egyetlen sorral megtehető:

```
$db = new mysqli('localhost', 'root', '', 'szakdoga');
```

Ahol az első argumentum a szerver címe, a második és harmadik argumentum rendre a csatlakozáshoz használt felhasználó neve és jelszava, a negyedik pedig a szerveren elérhető adatbázis neve, melyhez csatlakozni szeretnénk.

Ezt követően az SQL nyelv segítségével küldhetünk lekérdezéseket a szervernek. Ennek egy lehetséges módja a következő:

```
$stmt = $db->prepare("SELECT username, email, register_date FROM  
    users WHERE id=1");  
$stmt->execute();  
$result = $stmt->get_result();  
$user = $result->fetch_assoc();
```

Ebben az esetben az eredmény egy asszociatív tömbben fog tárolódni.

2.7 JQuery

A JQuery egy népszerű, nyílt forrású Javascript könyvtár, mely előnyei között szerepel, hogy teljesen böngésző kompatibilis [17], Ajax támogatással rendelkezik, segítségével átláthatóbb kódokat írhatunk és egyszerűen manipulálhatjuk a DOM-ot (Dokumentum Objektum Modell) elemeit. Első kiadása 2006-ban jelent meg, azóta rengeteg frissítést kapott, legújabb verziója 2021 óta elérhető.

Egy weboldal elemei csak akkor manipulálhatóak zökkenőmentesen, ha betöltött az oldal DOM-ja [16]. JQuery-vel használhatunk egy egyszerű függvényt, ami a DOM betöltése után fog lefutni:

```
$(document).ready(function() {  
    ...  
})
```

Emellett a szelektorok használata is kézenfekvőbb JQuery-ben a hagyományos Javascript-hez képest:

```
$('.email-line').css('flex-direction', 'column');
```

Például ez az utasítás az összes olyan elemet kiválasztja, melynek class azonosítója email-line, és ezek flex-direction CSS tulajdonságát column-ra változtatja.

Mint már említésre került, a JQuery Ajax támogatással is rendelkezik [18]. Az Ajax egy olyan webfejlesztési technika, mely lényege az, hogy kapcsolatot teremthetünk a szerverrel, illetve frissíthetjük az oldal egyes részeit a teljes weboldal újratöltése nélkül [19], ezt JQuery használatával megtehetjük az alábbi módon:

```
function deleteImg(param) {  
    $.ajax({  
        type: 'POST',  
        url: 'delete.php',  
        async: false,  
        dataType: "json",  
        data: {img:param},  
        success: function(data){  
            $('#'+data.message).remove();  
        }  
    });  
}
```

Ez a példa függvény meghíváskor elküldi egy POST kéréssel a delete.php-nak az img változót melynek értéke a függvény paramétere lesz. Ezután a PHP állomány sikeres végrehajtása után eltávolítja azt az elemet a weboldalról, melynek azonosítója megegyezik a delete.php által visszaküldött message értékével.

2.8 XAMPP

Az XAMPP egy nyílt forrású webszerver-szoftvercsomag, mely először 2002-ben jelent meg, legfrissebb kiadása pedig 2018-ban látott napvilágot. Neve a következő rövidítésekből tevődik össze: X (a platformfüggetlenségre utal) Apache, MariaDB, PHP, Perl.

Az XAMPP legnagyobb előnye, hogy tartalmaz minden olyan eszközt, mely elengedhetetlen a webes alkalmazások fejlesztéséhez [20]. Segítségével egyszerűen futtathatunk PHP kódokat, adatbázisokat hozhatunk létre, tesztelhetjük weboldalunkat és az implementált funkciók helyes működését.

3. Az alkalmazás bemutatása

3.1 Specifikáció

Ebben a fejezetben kerül sor annak az ismertetésére, hogy hogyan is kellene működnie az alkalmazásnak, felsorolásra kerülnek azok a funkciók, melyek az alkalmazás készítése során implementálásra fognak kerülni. Ezen kívül szó lesz arról is, hogy a felhasználó hogyan léphet interakcióba a rendszerre, milyen különböző használati esetek léphetnek fel és ezekre az esetekre hogyan kellene reagálnia az alkalmazásnak.

3.1.1 Regisztráció - Bejelentkezés

Mivel a hirdetések böngészésén kívül a legtöbb funkció eléréséhez, használatához felhasználói fiók szükséges, ezért a felhasználónak tudnia kell regisztrálni, létező fiók mellett pedig bejelentkezni a weboldalra. Az ezen funkciókhoz való navigálást lehetővé tevő gombokat jól látható helyen kell elhelyezni az alkalmazásban. Regisztrációhoz felhasználónév, email-cím és a jelszó kétszeri megadása szükséges. Ezután az adatok ellenőrzése következik az alábbi módon:

- Nem regisztrálhatunk már létező felhasználónévvel vagy email címmel
- A felhasználónév nem tartalmazhatja az admin szót
- A megadott két jelszónak meg kell egyeznie egymással
- A felhasználónév és a jelszó nem lehet hosszabb 15 karakternél
- Az email-cím nem lehet hosszabb 30 karakternél

Ha a megadott adatok eleget tesznek az említett feltételeknek, megtörténik a regisztráció és a felhasználó be lesz léptetve az oldalra. Hasonló módon működik a bejelentkezés is, érvényes felhasználónév-jelszó páros megadása esetén leszünk beléptetve az oldalra.

Ezen felül, ha egy felhasználó éppen be van jelentkezve az alkalmazásba, akkor a bejelentkezés és regisztráció gombokat le kell cserélni a felhasználó nevére és a kijelentkezést funkcióját megvalósító gombra.

3.1.2 Kijelentkezés

Természetesen, ha egy felhasználó képes bejelentkezni a rendszerbe, akkor a kijelentkezést is lehetővé kell tenni, ezáltal ugyanazon eszközön más, illetéktelen felhasználók nem fognak tudni hozzáférni a fiókjához. Kijelentkezés esetén az ezt lehetővé tevő gombot és a felhasználó nevét le kell cserélni a bejelentkezés és regisztráció gombokra.

3.1.3 Hirdetés feltöltése

Sikeres bejelentkezést követően a felhasználónak lehetőséget kell adni a hirdetések feltöltésére. Ezen funkció igénybevételekor az alábbiak megadása szükséges:

- A hirdetés címe
- A hirdetés leírása
- A termék ára (Forintban)
- A csomagküldés lehetősége
- A település
- A termék kategóriája (Előre megadott opciókból)
- Címlapkép

A termék könnyebb azonosítását, minőségének, épségének ellenőrzését elősegítheti, ha több kép is készül a termékről, ezért a címlapkép megadása mellett a felhasználónak lehetőséget kell adni 10 egyéb kép feltöltésére.

3.1.4 Hirdetés szerkesztése

Nagyon bosszantó lenne és csökkentené a felhasználói élményt, ha egy hirdetésnél megadott árat szeretnénk megváltoztatni, vagy egy leírásban történt elírást szeretnénk kijavítani, és hogy ezt megtegyük, ahhoz teljesen új hirdetést kellene feladnunk.

Ahhoz, hogy ezt elkerüljük szükséges egy opció, mely alatt már feladott hirdetéseinket szerkeszthetjük. A funkcióba való belépéskor meg kell jeleníteni a hirdetéshez megadott adatokat és feltöltött képeket, lehetővé kell tenni a szöveges adatok átírását a címlapkép cserélését (törlését nem, hiszen azzal kép nélkül maradhatna a hirdetés), a plusz képek törlését

és feltöltését. Emellett el kell helyezni egy funkciót a hirdetés szerkesztésének oldalán, mellyel a felhasználó törölheti a hirdetést

3.1.5 Profil

Minden felhasználónak rendelkeznie kell egy profillal, melynek legfontosabb funkciója az értékelések írása lesz. Minden felhasználónak lehetőséget kell adni arra, hogy értékelést írjon más felhasználókról a profiljuk alatt, melyhez írhat egy indoklást, illetve kiválaszthatja, hogy az értékelése pozitív vagy negatív kategóriába esik. Természetesen a felhasználóról írt értékeléseket bárki megtekintheti a profilja alatt.

Ezen felül lehetővé kell tenni azt is, hogy szükség esetén minden felhasználó megváltoztathassa a jelszavát (a jelenlegi jelszó ismeretében), illetve megad hasson olyan opcionális adatokat, mint a teljes neve vagy a telefonszáma. Ezeket a funkciók szintén a profil alatt legyenek elérhetők.

Emellett minden felhasználó profilja alatt fel kell tüntetni olyan adatokat, mint az email címe, a regisztráció dátuma, a teljes neve és a telefonszáma, ha ezek ismertek, illetve a felhasználóról írt értékelések mennyisége és a felhasználó pontszáma a weboldalon, mely a pozitív (+1) és negatív (-1) értékelések számából tevődik össze. Ezen kívül, ha nem a saját profilunkra lépünk, hanem egy másik felhasználóéra, akkor azon szerepelnie kell egy olyan opciónak is, mely segítségével beszélgetést kezdeményezhetünk vele.

3.1.6 Főoldal - Hirdetések közötti keresés

Az alkalmazás főoldalán, ahol az weboldalra lépéskor a hirdetések lesznek megjelenítve feladás dátuma alapján csökkenő sorrendben. Az itt látható hirdetések esetében nem szükséges az azokkal kapcsolatos minden információt feltüntetni elég, ha a legfontosabb adataik lesznek megjelenítve, mint például a hirdetés címe, a termék ára vagy a hirdetést feladó felhasználó neve és annak pontszáma. Emellett a felhasználó nevére való kattintás át kell navigáljon az adott felhasználó profiljára.

Implementálni kell a hirdetések közötti keresés lehetőségét, ezzel is könnyebbé téve egy adott termék megtalálását. Ha egy szűrésre több találat is érkezik, akkor azokat is feladás dátuma szerint csökkenő sorrendben kell megjeleníteni.

A következő szűrési lehetőségeket kell megvalósítani:

- Szöveg keresése a hirdetés címében
- Szűrés településre
- Minimum ár megadása
- Maximum ár megadása
- Kategória kiválasztása
- Csomagküldés lehetősége

3.1.7 A hirdetés főoldala

Az alkalmazás főoldalán egy hirdetés kiválasztása esetén át kell navigálni annak főoldalára, ahol fel kell tüntetni annak minden, a feladásnál megadott adatát. Emellett meg kell jeleníteni a hirdetéshez tartozó képeket, a feladás dátumát, a feladó felhasználó nevét és pontszámát is. A főoldalhoz hasonlóan a felhasználó nevére való kattintás itt is át kell navigáljon az illető profiljára.

Ezen felül, ha nem a saját hirdetésünket jelenítettük meg, hanem egy másik felhasználóét, akkor el kell helyezni ezen az oldalon két gombot is, melyek az alkalmazáson belüli navigációt teszik lehetővé. Az egyik segítségével üzenetet küldhetünk a hirdetés feladójának, a másikkal pedig jelenthetjük az adott hirdetést.

3.1.8 Felhasználók közötti kommunikáció

Egy ilyen, vagy ehhez hasonló alkalmazás esetében elengedhetetlen, hogy a felhasználók valamilyen módon tudjanak kommunikálni egymással, ezzel is megkönnyítve az esetleges megállapodások létrejöttét. Ezért a weboldalnak tartalmaznia kell egy funkciót, melynek segítségével a felhasználók üzeneteket tudnak küldeni egymásnak.

Lehetőséget kell adni a felhasználóknak, hogy üzenetek küldését kezdeményezhessék egy másik felhasználó profiljáról, vagy egy másik felhasználó által feladott hirdetés főoldaláról. Tehát ezekről az oldalakról át kell tudjanak navigálni egy olyan oldalra, ahol beszélgetést kezdeményezhetnek az adott felhasználóval. Ennek valós időben frissülnie kell, ezzel segítve a hosszabb várakozás, vagy az oldal újratöltése nélküli, folyamatos beszélgetéseket.

Ezen felül az alkalmazás navigációs sávján szerepelnie kell egy olyan menüpontnak, mely segítségével elérhetjük az összes olyan felhasználóval való beszélgetésünket, akiknek korábban üzenetet írtunk vagy akiktől üzenetet kaptunk. Emellett a navigációs sávon belül ezen a menüponton jól láthatóan jelezni kell azt is, ha az aktuálisan bejelentkezett felhasználó új üzenetet kapott, így akár az oldalra való lépéskor is látszódnia fog, ha valaki beszélgetést kezdeményezett velünk.

3.1.9 Hirdetés jelentése

A hirdetés jelentése funkcióra az adott hirdetés főoldaláról navigálhatunk át. Ennek a funkciónak a segítségével minden felhasználónak lehetőséget kell adni arra, hogy jelenthesse mások hirdetéseit, ha például nem valós adatokat adtak meg a leírásban, vagy egy másik termékről töltöttek fel képeket. Az ilyen, vagy ehhez hasonló problémákat ezen az oldalon jelezhetjük az alkalmazás üzemeltetőinek, akik az admin oldal segítségével megtekinthetik a beérkező jelentéseket és megfelelő módon kezelhetik az adott helyzetet.

3.1.10 Admin oldal

Az admin oldal fő feladata az, hogy megkönnyítse az alkalmazás üzemeltetését, és az azzal kapcsolatos problémák kezelését. Ezt az oldalt csakis az erre a célra létrehozott „ADMIN” felhasználó segítségével lehet elérni.

Felmerülhetne ilyen helyzetben az a probléma, hogy rosszindulatú felhasználók létrehoznak egy fiókot melynek neve tartalmazza az „admin” szót, ezzel az alkalmazás üzemeltetőjét megszemélyesítve próbálnának meg megtéveszteni másokat (például megpróbálhatnák elkérni egyes felhasználók jelszavát, vagy használati díjat követelhetnének az alkalmazás igénybevételéért). Ennek elkerülése végett senki sem regisztrálhat olyan felhasználónévvel melyben szerepel az „admin” vagy az „ADMIN” szó (az ADMIN felhasználót manuálisan, közvetlenül az adatbázisban kell létrehozni). Emellett az oldal üzemeltetője által használt ADMIN fiók üzenetei külön helyen kell megjelenjenek egy felhasználó beszélgetései között kiemelve, hogy az adott üzenetet az alkalmazás karbantartója küldte, ezzel is elkerülve az esetleges félreértéseket.

Az ADMIN felhasználó nem funkcionálhat úgy, mint a weboldal többi tagja, nem tölthet fel hirdetéseket, nem írhat értékeléseket, nem küldhet jelentéseket és nincs profilja sem, hiszen egyedüli feladata az alkalmazás karbantartása. Ebből a célból implementálni kell egy külön oldalt, melyhez csak az ADMIN felhasználó férhet hozzá. Ezen az oldalon keresztül érheti el az alkalmazás üzemeltetője a beérkező jelentéseket. Továbbá ezen az oldalon lehetőséget kell adni az ADMIN felhasználónak arra is, hogy a jelentések között a jelentett hirdetés tulajdonosára, vagy a jelentést beküldő felhasználó nevére keressen, ezzel gyorsítva azok kezelésének folyamatát.

A hirdetésekhez hasonlóan, ha kiválasztásra kerül egy jelentés, akkor át kell navigálni annak főoldalára, ahol elolvasható a jelentés szövege (mely az esetleges nagy terjedelme miatt nincs megjelenítve a több jelentést felsoroló nézetben), és ahonnan további navigálási opciók érhetőek el: a hirdetés nevére kattintva annak főoldalára, a jelentett hirdetés tulajdonosának, illetve a jelentés beküldő felhasználó nevére kattintva azok profiljára.

A jelentések kezelésének első lépése az lehet, hogy az ADMIN felhívja a jelentett hirdetés tulajdonosának figyelmét a problémára, ezért az ADMIN is, a többi felhasználóhoz hasonlóan küldhet üzeneteket, melyekre a fent említett módokon fel lesz hívva az adott

felhasználó figyelme. Viszont felmerülhet az is, hogy a hirdetéssel kapcsolatos problémát a tulajdonosa hosszú idő után sem oldja meg, ezért lehetőséget kell adni arra is, hogy az ADMIN ki is törölhesse más felhasználók hirdetéseit, ezzel megszüntetve a problémát.

3.1.11 Az oldalon való navigáció

Ahogy a 3.1-es fejezetben említésre került, az oldal legtöbb funkciójának rendeltetésszerű működéséhez elengedhetetlen az, hogy a felhasználó bejelentkezzen a fiókjába. Például hirdetés feladásánál szükségünk van annak tulajdonosára, értékelés írásánál tudnunk kell ki írta az adott értékelést.

Éppen ezért bejelentkezés nélkül csak a főoldal, hirdetések böngészése, azok közötti keresés és a hirdetések főoldalának megtekintése lesz elérhető. Nem lesznek elérhető olyan funkciók, mint a jelentés küldése, hirdetés feladása vagy felhasználók profiljának elérése és velük való beszélgetés kezdeményezése. Ha egy felhasználó mégis megpróbál elérni egy ilyen funkciót, akkor az alkalmazásnak át kell navigálnia a bejelentkező felületre, egy üzenettel felhívva a felhasználó figyelmét a belépésre.

3.2 Grafikus terv

3.2.1 Reszponzivitás

Mivel a cél egy modern webes alkalmazás készítése ezért szeretnénk a weboldalt úgy kialakítani, hogy az rezponzív legyen, kisebb kijelző méretek esetében is esztétikus legyen a megjelenése. Ezt a CSS által nyújtott eszközök közül az elterjedten használt Media Query segítségével fogjuk kialakítani. Ilyen módon könnyedén létrehozhatunk az oldalnak többféle, a kijelző mérete szerint változó megjelenést az alábbi módon:

```
.submit {  
    font-size: 22px;  
}
```

A submit class azonosítójú elemek betűmérete alapértelmezetten 22 pixel. Ezt Media Query-k segítségével megváltoztathatjuk:

```
@media (max-width: 620px) {  
    .submit {  
        font-size: 18px;  
    }  
}
```

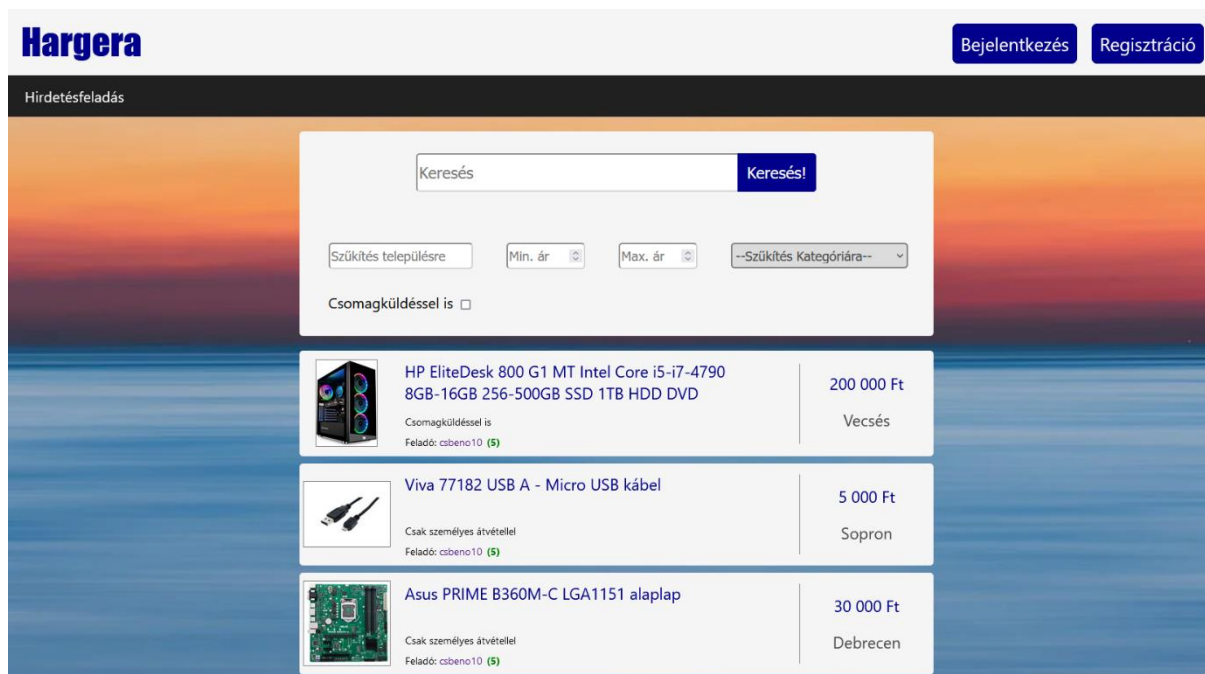
...

```
@media (max-width: 400px) {  
    .submit {  
        font-size: 16px;  
    }  
}
```

Ilyen módon ugyanezen elemek betűmérete 620 pixelnél kisebb, de 400 pixelnél nagyobb kijelző méretek esetében 18 pixel lesz, 400 pixelnél kisebb kijelzőkön pedig 16 pixelre lesz állítva.

3.2.2 Az alkalmazás alapvető megjelenése

Az weboldal megnyitásakor a felhasználó az alkalmazás főoldalára lesz navigálva, ahol az 1. ábrán látható képernyő fogja várni (1680 pixel széles kijelző esetében):



1. ábra: Az alkalmazás

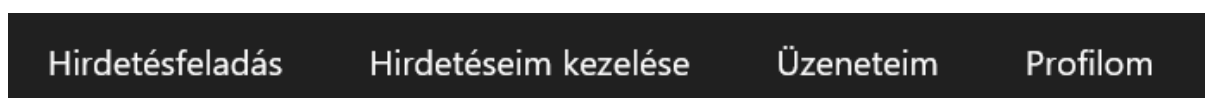
Az alkalmazás legtöbb oldalának megjelenése hasonló stílust fog követni, az oldal felső részén két sáv fog elhelyezkedni, melyek 800 pixelnél szélesebb kijelzők esetében a sticky CSS tulajdonsággal rendelkeznek. Ezt azt jelenti, hogy ez a két sáv az oldal legörgetésekor sem tűnik el, mindig fix helyen, az képernyő felső részén fognak elhelyezkedni. A sticky tulajdonság megszüntetésére kisebb kijelző méretek esetén azért van szükség, mert túl sok helyet foglalna el, kevésbé jól láthatóvá tenné az alkalmazás többi részét, vagy akár teljesen el is tüntetné azokat.

A felső, világosabb sáv bal oldalán az alkalmazás logója, „fantázianeve” fog elhelyezkedni. A sáv jobb oldalán pedig abban az esetben, ha a felhasználó még nem jelentkezett be a fiókjába, a bejelentkezésre és a regisztrációra navigáló gombok fognak megjelenni, melyek bejelentkezést követően le lesznek cserélve a kijelentkezés gombra, és a felhasználó nevére a 2. ábrán látható módon.



2. ábra: Navigáló gombok

Az előbb említett sáv alatt pedig egy navigációs sáv fog elhelyezkedni, mely az alkalmazás különböző oldalai közötti navigálást fogja segíteni. Bejelentkezést követően a 3. ábrán látható opciókat fogja tartalmazni



3. ábra: Navigációs sáv bejelentkezés esetén

Ezen opciók mindegyikének igénybevételéhez bejelentkezés szükséges ezért felmerülhet az a probléma, hogy bejelentkezés előtt milyen megjelenést kapjon ez a navigációs sáv. Mivel sem eltüntetni sem pedig üresen hagyni nem lenne célszerű, ezért bejelentkezés előtt egy rövidebb navigációs sávot fogunk feltüntetni (4. ábra), egyetlen opciót tartalmazva, ezzel is felhívva a felhasználó figyelmét erre a lehetőségre:



4. ábra: Navigációs sáv bejelentkezés nélkül

Természetesen, ahogy már a 3.11-es fejezetben említésre került, ennek az opciónak a kiválasztásakor a felhasználó el lesz navigálva a bejelentkezés oldalára, felhívva a belépés szükségességére a figyelmét.

Emellett, a 3.10-es fejezet szerint az alkalmazás karbantartására létrehozott ADMIN felhasználó nem fér hozzá olyan opciókhoz, mint például a hirdetésfeladás ezért ennek a felhasználónak más lehetőségek fognak szerepelni a navigációs sávján az 5. ábra alapján.



5. ábra: Az ADMIN felhasználó navigációs sávja

Az imént említett 2 felső sáv mellett az oldalak legalján, egy lábléc fog szerepelni, mely az oldal üzemeltetőjének elérhetőségeit fogja tartalmazni a 6. ábrán látható módon.

Kapcsolat:


✉ Email cím: hagera@gmail.com

☎ Telefonszám: +36 20-583-8346

6. ábra: Lábléc

Az alkalmazás egyes oldalain belül pedig horizontálisan középre rendezve fognak megjelenni az adott oldal főbb funkciói:


☐ Csomagküldéssel is



**HP EliteDesk 800 G1 MT Intel Core i5-i7-4790
8GB-16GB 256-500GB SSD 1TB HDD DVD**

Csomagküldéssel is
Feladó: csbeno10 (5)


200 000 Ft
Vecsés



Viva 77182 USB A - Micro USB kábel

Csak személyes átvétellel
Feladó: csbeno10 (5)

5 000 Ft
Sopron



Asus PRIME B360M-C LGA1151 alaplap

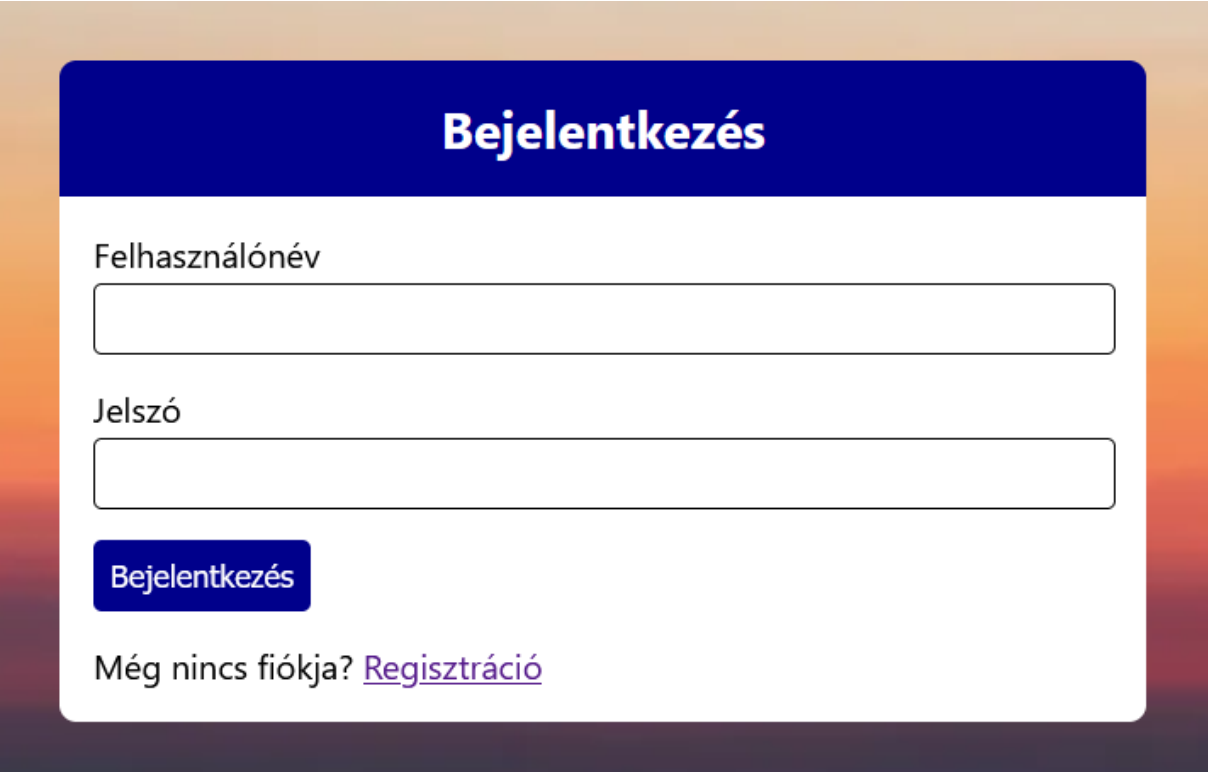
Csak személyes átvétellel
Feladó: csbeno10 (5)

30 000 Ft
Debrecen

7. ábra: Az alkalmazás főoldala

23

A bejelentkezést, regisztrációt és a hirdetések jelentését lehetővé tevő oldalak egyszerűbb megjelenést kapnak, ezeknek az oldalaknak az alsó részén nem lesz feltüntetve a kapcsolat lábléc, felső részén pedig nem fog szerepelni az előzőekben tárgyalt két sáv, tehát az oldal logója, a navigációs sáv, illetve a bejelentkezés és regisztráció gombok sem. Viszont ezeken az oldalakon is a többihez hasonlóan horizontálisan középre rendezve fognak megjelenni az adott oldal funkciói, melyek reszponzívak lesznek. Erre egy példa a 8. ábra.



The image shows a login form titled "Bejelentkezés" (Login) in a dark blue header. Below the header, there are two input fields: "Felhasználónév" (Username) and "Jelszó" (Password). Below the password field is a dark blue button labeled "Bejelentkezés". At the bottom, there is a link that says "Még nincs fiókja? [Regisztráció](#)". The form is set against a background with a vertical gradient from light orange to dark purple.

8. ábra: Bejelentkezés

Az alkalmaz oldalainak háttereként használt kép ingyenesen használható és letölthető az alábbi oldalról: <https://www.pexels.com/photo/calm-body-of-water-during-golden-hour-1631677/>

3.3. Adatbázis

Komplexebb webes alkalmazás esetében elengedhetetlen nagyobb adatmennyiségek tárolása, kinyerése adatbázisból. Ebben a fejezetben az alkalmazás által használt adatbázist, annak különböző tábláit fogjuk bemutatni.

3.3.1 Users tábla

Az adatbázis legfontosabb táblája, ebben lesznek tárolva az oldalra regisztrált felhasználók és azok különböző adatai:

users	
id	int(11)
username	varchar(15)
password	varchar(15)
email	varchar(255)
register_date	datetime
phone	varchar(20)
full_name	varchar(255)
score	int(11)

9. ábra: users tábla

A users (felhasználók) tábla mezői:

- id: A tábla elsődleges kulcsa
- username: A felhasználónév
- password: A felhasználó jelszava
- email: A felhasználó email címe
- register_date: A regisztráció dátuma
- phone: A felhasználó telefonszáma

- full_name: A felhasználó teljes neve
- score: A felhasználó pontszáma

3.3.2 Advertisement tábla

Ebben a táblában lesznek tárolva az egyes felhasználók által feladott hirdetések adatai:

advertisement	
id	int(11)
user_id	int(11)
title	varchar(100)
description	varchar(1500)
price	int(11)
shipping	varchar(255)
settlement	varchar(255)
category	varchar(255)
date	datetime

10. ábra: advertisement tábla

Az advertisement (hirdetés) tábla mezői:

- id: A tábla elsődleges kulcsa
- user_id: A hirdetést feladó felhasználó egyedi azonosítója
- title: A hirdetés címe
- description: A hirdetés leírása
- price: A termék ára
- shipping: A csomagküldés lehetősége
- settlement: A település neve, ahol a személyes átvétel lehetséges
- category: A termék kategóriája
- date: A hirdetés feladásának dátuma

3.3.3 Ratings tábla

Ennek a táblának a segítségével fogjuk tárolni a felhasználók egymásról írt értékelései, és azok különböző adatai:

ratings	
id	int(11)
writer_id	int(11)
target_id	int(11)
rate_date	datetime
text	varchar(500)
rating	varchar(255)

11. ábra: ratings tábla

A ratings (értékelések) tábla mezői:

- id: A tábla elsődleges kulcsa
- writer_id: Az értékelést író felhasználó egyedi azonosítója
- target_id: Annak a felhasználónak az egyedi azonosítója, akiről az értékelést írták
- rate_date: Az értékelés leadásának dátuma
- text: Az értékeléshez megadott indoklás, szöveg
- rating: A felhasználó minősítése (pozitív/negatív)

3.3.4 Messages tábla

Ebben a táblában lesznek tárolva a felhasználók egymásnak küldött üzenetei és az ezekkel kapcsolatos adatok:

messages	
id	int(11)
sender_id	int(11)
receiver_id	int(11)
m_date	datetime
text	varchar(300)
seen	tinyint(1)

12. ábra: messages tábla

A messages (üzenetek) tábla mezői:

- id: A tábla elsődleges kulcsa
- sender_id: Az üzenetet küldő felhasználó egyedi azonosítója
- receiver_id: Annak a felhasználónak az egyedi azonosítója, akinek az üzenetet küldték
- m_date: Az üzenet küldésének dátuma
- text: Az üzenet szövege
- seen: Egy olyan mező, mely azt tárolja, hogy a felhasználó, akinek az üzenetet küldték látta-e az adott üzenetet (1-es értéket vesz fel, ha a felhasználó látta az üzenetet, ellenkező esetben pedig 0-t)

3.3.5 Reports tábla

Ennek a táblának a segítségével fogjuk tárolni a különböző hirdetésekről beküldött jelentéseket és azok adatait:

reports	
id	int(11)
writer_id	int(11)
r_text	varchar(1000)
target_user_id	int(11)
target_ad_id	int(11)
r_date	datetime

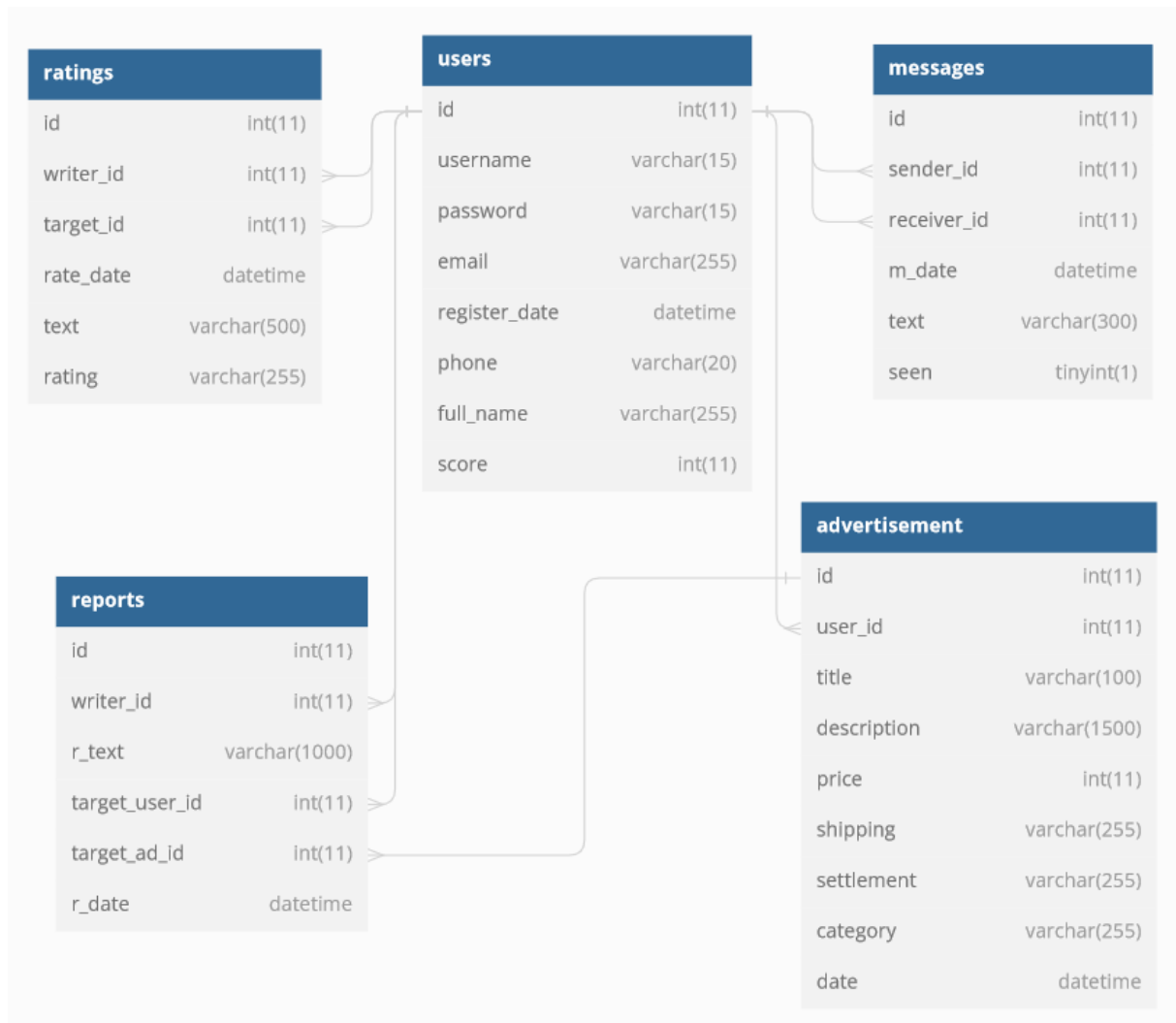
13. ábra: reports tábla

A reports (jelentések) tábla mezői:

- id: A tábla elsődleges kulcsa
- writer_id: A jelentést feladó felhasználó egyedi azonosítója
- r_text: A jelentéshez megadott indoklás, szöveg
- target_user_id: A jelentett hirdetés tulajdonosának egyedi azonosítója
- target_ad_id: A jelentett hirdetés egyedi azonosítója
- r_date: A jelentés elküldésének dátuma

3.3.6 Táblák közötti kapcsolat

Az alkalmazásban használt táblák közötti kapcsolat létrehozásához az egyes mezők közötti kapcsolat definiálása szükséges. Ezek a szülő táblában elsődleges kulcsok (Primary Key), a hozzá kapcsolt gyerek táblában pedig idegen kulcsok (Foreign Key) lesznek. Ezeket a kapcsolatokat ábrázolja a 14. ábra.



14. ábra: A táblák közötti kapcsolatok

Ezek az idegen kulcsok az adatbázisban az ON DELETE CASCADE megszorítással rendelkeznek. Ez azt jelenti, hogy ha a szülő rekord, amelyre az idegen kulcs mutat törlésre kerül, akkor a gyerek rekord is kitörlődik. Ez hasznos lehet például abban az esetben, ha egy felhasználó törli az egyik hirdetését, akkor azzal együtt törlődni fognak a hirdetésről feladott

jelentések is, így az alkalmazás karbantartójának már nem fog feleslegesen átnézni olyan jelentéseket, melyek nem létező hirdetésre mutatnak.

3.4 Implementáció

3.4.1 Lekérdezések

Az alkalmazásban a lekérdezések prepared statement-ek (előkészített utasítások) segítségével kerültek megvalósításra, ezzel védekezve az oldalt érő esetleges SQL Injection támadások ellen, mely az egyik leggyakoribb támadási módszer a weboldalak ellen. Az Injection támadások célja az, hogy egy adatbevitelre szolgáló helyen, pl egy form mezőjében futtatható kódot helyeznek el, ezzel próbálnak olyan műveleteket végrehajtani, melyekhez alapvetően nem lenne hozzáférésük. Az SQL Injection támadások esetében SQL utasításokat helyeznek el egy beviteli mezőben, sikeres támadás esetén mezőket, táblákat vagy akár a teljes adatbázist is törölhetik.

A prepared statement-ek használata az egyik leghatásosabb védekezési módszer az ilyen támadások ellen. Egy ilyen lekérdezés első lépéseként meg kell adnunk a futtatni kívánt SQL utasítást, melyben a paramétereket kérdőjelekkel helyettesítjük:

```
$stmt = $db->prepare("SELECT * FROM users WHERE BINARY username=? OR BINARY email=? LIMIT 1");
```

Ezt követően meg kell adnunk, hogy az egyes kérdőjelek helyére milyen típusú adatokat várunk, illetve a változókat, melyek az utasítás paramétereként szolgálnak:

```
$stmt->bind_param("ss", $username, $email);
```

Ebben az esetben mindkét változó string típusú lesz, ezt definiálja a bind_param függvény első argumentumában megadott „ss” string. Ha például az első változó string, a második változó integer típusú lenne, akkor a függvény első argumentuma „si” lenne. A függvény többi argumentuma pedig a kérdőjelek helyére szánt változók, ebben az esetben az első kérdőjel helyére a \$username, a második kérdőjel helyére pedig az \$email változóban tárolt érték fog kerülni.

Ezután az alábbi módon hajthatjuk végre az utasítást, és tárolhatjuk annak eredményét egy asszociatív tömbben:

```
$stmt->execute();  
$result = $stmt->get_result();  
$user = $result->fetch_assoc();
```

3.4.2 Regisztráció

Az alkalmazásba való regisztráció egy Html form segítségével történik. A form feltöltése után megvizsgáljuk a feltöltött értékeket. Ha a vizsgálat során kiderül, hogy valamelyik érték nem felel meg a specifikációban ismertetett feltételeknek, akkor az arra vonatkozó hibaüzenetet eltárolunk az \$errors tömbben. Például:

```
if (preg_match("/admin/i", strtolower($username))) {  
    array_push($errors, "A felhasználónév nem tartalmazhatja az ad-  
    min szót");  
}
```

Ha az adatok ellenőrzése során nem kaptunk hibát, akkor létrehozunk az adatbázisban egy felhasználót a megadott adatok alapján:

```
$stmt = $db->prepare("INSERT INTO users(username, email, password,  
    register_date) VALUES (?, ?, ?, ?)");  
$stmt->bind_param("ssss", $username, $email, $password, $date);  
$stmt->execute();
```

Majd a felhasználót bejelentkezéshez hasonlóan beléptetjük az alkalmazásba, átnavigáljuk annak főoldalára, Session változók segítségével pedig eltároljuk a beléptetett felhasználót. A Session változók hasonlóan működnek, mint a süti, adott böngészőn belül bármely oldalon elérhetjük őket. Éppen ezért célszerű ilyen változókbán tárolnunk azokat az adatokat, melyeket az alkalmazáson belül több oldalon is fel szeretnénk használni.

Abban az esetben pedig, ha az ellenőrzések során hibákat kaptunk, akkor azokat megjelenítjük a felhasználó számára. A hibák megjelenítésekor az oldalt újratöltésre kerül, viszont a megadott adatokkal kapcsolatos problémák kijavítása érdekében célszerű lenne, ha a form mezőiből nem tűnnének el a beírt adatok. Ebből a célból a felhasználónevet és az email

címet tartalmazó mezők alapértelmezett értékeként megadjuk a felhasználó adatait az alábbi módon:

```
<input type="text" name="username" value="<?php  
if(isset($_POST['username'])) echo $_POST['username']; ?>">
```

3.4.3 Bejelentkezés

Az alkalmazásba való bejelentkezés hasonlóképpen működik, mint a regisztráció, a felhasználónév és a jelszó megadása után ellenőrizzük, hogy létezik-e rekord az adatbázisban a megadott felhasználónév-jelszó párossal:

```
$stmt = $db->prepare("SELECT * FROM users WHERE BINARY username=?  
AND BINARY password=?");  
$stmt->bind_param("ss", $username, $password);  
$stmt->execute();  
$stmt->store_result();  
$numrows = $stmt -> num_rows();
```

Ebben az esetben a \$numrows változó fogja tartalmazni a lekérdezés eredményeként kapott sorok számát, mely alapján könnyedén ellenőrizhetjük, hogy helyes adatokkal szeretne-e belépni a felhasználó. Ha egy valós fiók adatai kerültek megadásra, akkor beléptetjük a felhasználót az alkalmazás főoldalára, és eltároljuk a bejelentkezett felhasználót Session változók segítségével. A regisztrációhoz hasonlóan hibás adatok esetén megjelenítünk egy hibaüzenetet a felhasználó számára.

3.4.4 Kijelentkezés

Az alkalmazásban látható kijelentkezés gombra kattintva a felhasználó át lesz navigálva az alkalmazás főoldalára, ahol az alábbi függvény kerül meghívásra:

```
session_destroy();
```

Ez a függvény meghíváskor megsemmisíti az adott Session-t beleértve az összes Session változót is. Mivel az aktuálisan bejelentkezett felhasználót ilyen változóban tároltuk, ezért azok megsemmisítésekor az alkalmazás alaphelyzetbe lép, amikor nincs bejelentkezve

felhasználó az adott eszközön. A kijelentkezés megvalósításra kerül a böngésző bezárásakor is, hiszen a sütitkel ellentétben a Session változók megsemmisülnek a böngésző bezárásakor.

3.4.5 Hirdetés feladása

Az első elképzelés az volt, hogy egy olyan funkcionál, mint a hirdetés feltöltése már nem olyan egyszerű a felmerülő hibák megjelenítése, mint például a regisztráció esetében. Hiszen a regisztrációnál az oldal újratöltése után megjeleníthetjük a hibákat és egyúttal a form mezőinek értékeként megadhatjuk a felhasználó által megadott adatokat, hiszen azok valamilyen szöveget tartalmaznak. Hirdetés feltöltése esetén azonban már nem csak szöveges adatokkal dolgoztunk, a felhasználó által feltöltött képeket is kezelünk kell. Ezért született meg az a döntés, hogy a hirdetések feltöltésére JQuery-t fogunk használni, mely lehetővé teszi az adatok feltöltését és a hibák megjelenítését az oldal újratöltése nélkül, így sikertelen feltöltés esetén sem tűnnek el a form mezőiből a felhasználó által megadott adatok és képek.

```
$("#form#data").submit(function(e){
    e.preventDefault();
    var formData = new FormData(this);

    $.ajax({
        url: "uploadajax.php",
        type: 'POST',
        data: formData,
        async: false,
        success: function (data) {
            ...
        },
        contentType: false,
        processData: false,
        dataType: "json",
    });
})
```

Ilyen módon a data id azonosítójú form feltöltésekor az abban megadott adatokat formData-ként egy POST kéréssel elküldjük az uploadajax.php-nak. Ezután a szöveges adatok rövid ellenőrzése következik, mely során megvizsgáljuk, hogy minden mező kitöltésre került-e.

15. ábra: Néhány példa a kitöltendő mezőkre

Majd ellenőrizzük azt is, hogy a felhasználó adott-e fel ugyanilyen névvel hirdetést, a felmerülő hibákat pedig egy tömbben tároljuk. Ezt követően ellenőrizzük a feltöltött fájlok mennyiségét, hiányzó címlapkép, és összesen 11-nél több kép feltöltése esetén kapott hibák ugyanezen tömbbe kerülnek. Ha a képek mennyisége nem megfelelő, akkor a képek további ellenőrzése nem szükséges (hiszen lehet, hogy azok nem is léteznek).

Abban az esetben, ha megfelelő mennyiségű kép került feltöltésre akkor ezen képek ellenőrzése következik, miszerint egyik kép mérete sem haladhatja meg a 10 MB-ot és minden feltöltött fájl kiterjesztése jpg, png vagy jpeg kell legyen. Ezt például a címlapkép esetében az alábbi módon tehetjük meg:

```
if ($_FILES["thumbnail"]["size"] > 1024 * 1024 * 10) {
    array_push($errors, "A címlapkép túl nagy");
}

$imageFileType =
strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg") {
    array_push($errors, "A címlapkép nem megfelelő kiterjesztésű
(jpg, png, jpeg) fájl");
}
```

Ezen ellenőrzések során kapott hibákat is az erre létrehozott tömbben tároljuk. Majd az ellenőrzések befejezése után, ha azok során felmerült valamilyen hiba, akkor azokat json formátumban válaszként elküldjük az előzőekben tárgyalt POST kérést küldő \$.ajax() módszernek.

```
echo json_encode(array('result' => 0, 'errors' => $errors));
```

Abban az esetben, ha nem merült fel egyetlen hiba sem az ellenőrzések során, feltöltjük a hirdetéssel kapcsolatos adatokat az adatbázisba:

```
$stmt = $db->prepare("INSERT INTO advertisement(user_id, title,
description, price, shipping, settlement, category, date) VALUES
(?, ?, ?, ?, ?, ?, ?, ?)");
$stmt->bind_param("ississss", $user_id, $title, $description,
$price, $shipping, $settlement, $category, $date);
$stmt->execute();
```

Majd elmentjük a feltöltött képeket a megfelelő helyen. A hirdetések képei az uploads mappában lesznek tárolva. Azon belül minden felhasználónak létre lesz hozva egy saját mappa, amennyiben feltölt hirdetést, és ezen belül külön mappája lesz a felhasználó hirdetéseinek. Az említett mappák az adott felhasználó vagy a hirdetés egyedi azonosítójáról lesznek elnevezve:

```
$target_dir = "uploads/" . $user_id . "/" . $ad_id . "/";
if (!is_dir($target_dir)) {
    mkdir($target_dir);
}
```

A címlapkép „thumbnail” néven mentjük el, a többi kép pedig 0 és 9 közötti számokról lesz elnevezve, mindig a legkisebbre, amelyik még nem került kiosztásra. Ezt követően küldünk egy választ az \$.ajax() metódusnak a hirdetés sikeres feltöltéséről:

```
$target_file = $target_dir . "thumbnail.jpg";
move_uploaded_file($_FILES["thumbnail"]["tmp_name"], $target_file);
...
echo json_encode(array('result' => 1, 'ad_id' => $ad_id));
```

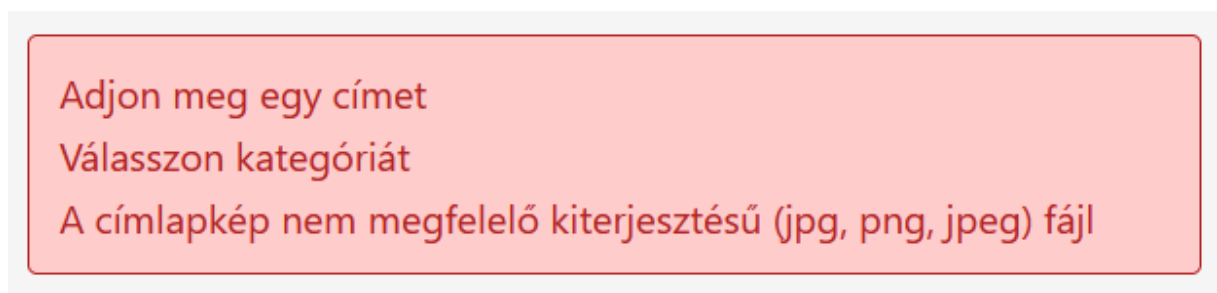
Végül lekezeljük a kapott választ. Sikeres hirdetésfeltöltés esetén átnavigálunk az imént létrehozott hirdetés főoldalára:

```
success: function (data) {
    if(data.result == 1){
        window.location.href = "ad_page.php?ad_id=" + data.ad_id;
    }
    ...
},
```

Ha pedig válaszként valamilyen hibát kaptunk, akkor azt megjelenítjük a felhasználó számára:

```
success: function (data) {  
    ...  
    else {  
        $('.ajaxerrors').css('display', 'block');  
        $('.ajaxerrors').empty();  
        for (let i = 0; i < data.errors.length; i++) {  
            $('#errors').append("<p class='error'>" + data.errors[i]  
                + "</p>");  
        }  
    }  
},
```

Melyeket a 2. ábrán látható módon tüntetünk fel a lap alján:



16. ábra: Hibák megjelenítése

3.4.6 Hirdetés szerkesztése

A hirdetések szerkesztése menüpontra kattintáskor a felhasználó először egy olyan oldalra lesz navigálva, ahol fel lesznek sorolva az adott felhasználó által feladott hirdetések, melyek melletti „szerkesztés” gombra kattintva léphet arra az oldalra, ahol módosíthatja a kiválasztott hirdetés adatait.

A hirdetések szerkesztésének oldala hasonlóképpen épül fel, mint a hirdetések feladásának oldala, a form szöveges mezőiben megjelenítésre kerülnek a hirdetéshez korábban megadott adatok, emellett fel lesznek tüntetve a hirdetéshez feltöltött képek is. A címlapképet új kép feltöltésével lehet cserélni, erre azért van szükség, hogy ne maradjon címlapkép nélkül a hirdetés. A többi képet viszont egy a kép alatt elhelyezett „törlés” szövegre kattintva eltávolíthatjuk, melyet a 17. ábra szemléltet.



17. ábra: Képek törlése

Ilyenkor a kattintás meghív egy Javascript függvényt mely JQuery segítségével elküldi a törölni kívánt kép nevét kiterjesztéssel együtt egy php állománynak, ami a kép sikeres törlése esetén visszaküld egy json formátumú válaszban a törölt kép nevét

```
$filename = "uploads/" . $user_id . "/" . $ad_id . "/" . $img;
if(unlink($filename)) {
    $num = $img[0];
    $res = array('result' => 1, 'message' => $num);
    echo json_encode($res);
}
```

Majd a válaszban kapott képet eltávolítjuk a felhasználói felületről, melyet tartalmazó konténer id azonosítója megegyezik a kép nevével:

```
success:function(data){
    if(data.result == 1){
        $('#'+data.message).remove();
    }
}
```

A szerkesztett adatok és feltöltött képek mentése hasonlóképpen működik az előző, 6.5-ös fejezetben tárgyaltakhoz, JQuery segítségével egy POST kéréssel elküldjük az adatokat egy php állománynak, mely hasonló ellenőrzések után frissíti az adatbázist az új adatokkal, és elmenti a képeket a megfelelő mappába. Az ellenőrzésekben arra kellett figyelni, hogy a címlapképen kívül 10 képet tölthetünk fel összesen. Ennek az vizsgálata másképpen működik, mint a hirdetés feladásánál, hiszen szerkesztésnél az előzőleg, és az újonnan feltöltött képek összegét kell vizsgálnunk. Emellett, mivel a képek neve 0 és 9 közötti számok, ezért célszerű lenne, ha tudnánk mely számok nem kerültek még kiosztásra. Ebből a célból a még fel nem használt neveket egy tömbben tároljuk:

```

for ($i = 0; $i < 10; $i++) {
    $filecheck = $target_dir . $i . ".jpg";
    if(!file_exists($filecheck)) {array_push($availableNames, $i .
        ".jpg");}
}

```

Ennek segítségével vizsgálhatjuk a feltöltött képek maximális számát:

```

if ((count($_FILES['files']['name']) - count($availableNames)) > 0){
    array_push($errors, "Maximum 10 képe lehet a címlapképen
        kívül");
}

```

Majd a hirdetések feltöltésénél használt ellenőrzések után ugyanezen tömböt használva elmentjük a képeket, oly módon, hogy az új képek a legkisebb, még fel nem használt számokról lesznek elnevezve:

```

foreach($_FILES['files']['name'] as $key=>$val) {
    $target_file = $target_dir . array_shift($availableNames);
    move_uploaded_file($_FILES["files"]["tmp_name"][$key],
        $target_file);
}

```

A hirdetés szerkesztésének oldalán egy gomb is el lesz helyezve, mellyel törölhetjük az adott hirdetést. Ez a funkció hasonlóan működik, mint az egyes képek törlése, JQuery segítségével jelezzük a szerver oldalnak, hogy törölni szeretnénk az éppen szerkesztett hirdetést ezzel eltávolítva az adatbázis megfelelő rekordját. Emellett törlésre kerül a hirdetéshez tartozó mappa és az abban tárolt képek is:

```

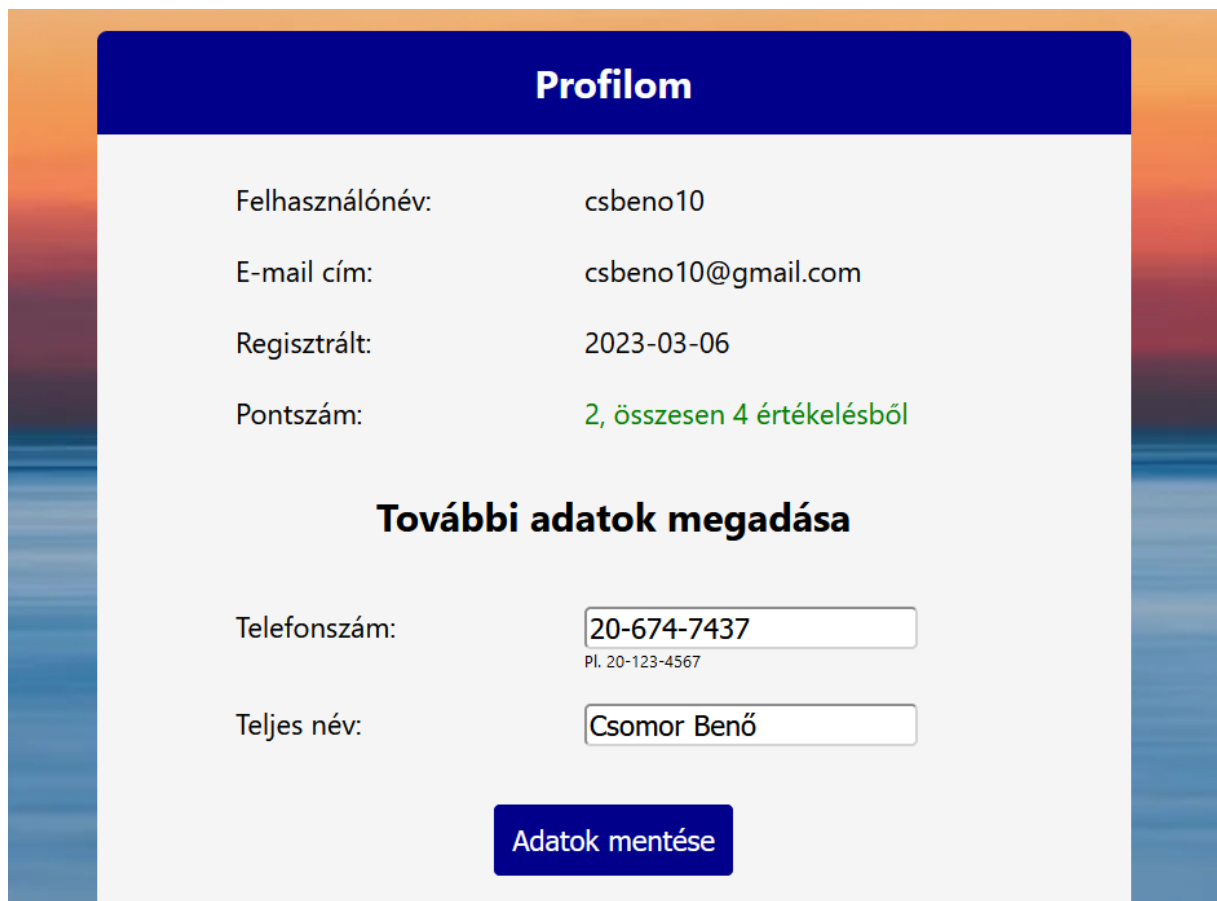
$foldername = "uploads/" . $user_id . "/" . $ad_id;
$files = glob($foldername . '/*');
foreach ($files as $file) {
    unlink($file);
}
rmdir($foldername);

```


3.4.7 Profil

Egy profil megnyitásakor először megvizsgálásra kerül, hogy a bejelentkezett vagy egy másik felhasználó profilját szükséges megjelenítenünk. Ezt a Session változóban tárolt aktuálisan bejelentkezett felhasználó segítségével tesszük meg.

Abban az esetben, ha a saját profilunkra navigáltunk az adataink és a rólunk írt értékelések mellett feltüntetünk 2 kisebb formot is, az egyik segítségével megadhatjuk a teljes nevünket és a telefonszámunkat, a másikkal pedig megváltoztathatjuk a jelszavunkat. A teljes név és telefonszám szerkesztése esetén frissítjük az adatbázist a megadott értékekkel, később ezek az adatok a form beviteli mezőiben fognak megjelenni. A jelszó megváltoztatásához az aktuális jelszó kétszeri megadása is szükséges, melyek helyességének ellenőrzése után frissítjük az adatbázist az új jelszóval.



The screenshot shows a web interface for a user profile. At the top, there is a dark blue header with the word "Profilom" in white. Below this, the profile information is displayed in a light gray box. The information includes: "Felhasználónév: csbeno10", "E-mail cím: csbeno10@gmail.com", "Regisztrált: 2023-03-06", and "Pontszám: 2, összesen 4 értékelésből". Below this information, there is a section titled "További adatok megadása" in bold. This section contains two input fields: "Telefonszám:" with the value "20-674-7437" and a small text "Pl. 20-123-4567" below it, and "Teljes név:" with the value "Csomor Benő". At the bottom of this section, there is a dark blue button with the text "Adatok mentése".

Felhasználónév:	csbeno10
E-mail cím:	csbeno10@gmail.com
Regisztrált:	2023-03-06
Pontszám:	2, összesen 4 értékelésből

További adatok megadása

Telefonszám:	<input type="text" value="20-674-7437"/>
	<small>Pl. 20-123-4567</small>
Teljes név:	<input type="text" value="Csomor Benő"/>

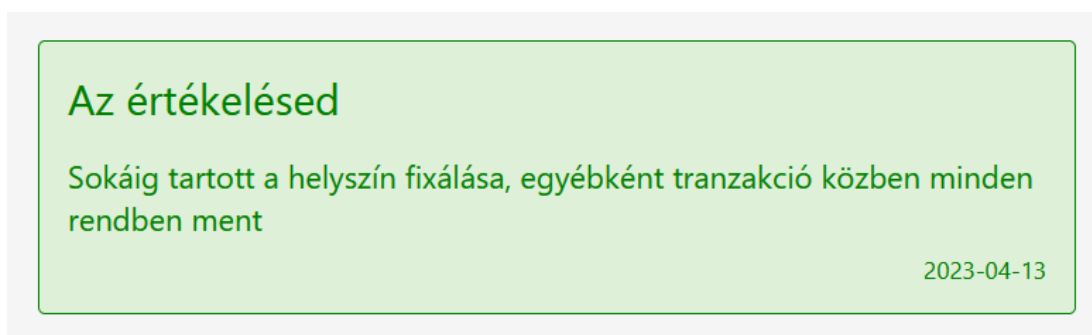
18. ábra: A felhasználó saját profilja

Ha egy másik felhasználó profiljára navigáltunk, akkor annak adatai és a többi felhasználó által írt értékelések mellett meg fog jelenni egy form is, melynek segítségével értékelést adhatunk le a felhasználóról az 19. ábra alapján.



19. ábra: Értékelés írása

Az értékelés kitöltése után azt feltöltjük az adatbázisba, és frissítjük az adott felhasználó pontszámát is. Ezt követően az adott felhasználó profiljára lépve az értékelés írását lehetővé tevő form helyén az előzőleg írt értékelésünk fog megjelenni, melyet a 20. ábra szemléltet.



20. ábra: Értékelés megjelenítése

Emellett elhelyezésre kerül egy gomb is, melynek segítségével arra az oldalra navigálhatunk, ahol beszélgethetünk az adott felhasználóval, melyet a 6.8-as fejezetben részletezünk.

3.4.8 Felhasználók közötti kommunikáció

Az alkalmazás navigációs sávján található „üzeneteim” opcióra kattintva a felhasználó el lesz navigálva egy olyan oldalra, ahol az adott felhasználó beszélgetései fognak megjelenni a 21. ábra alapján.



21. ábra: Üzeneteim opció

Először lekérdezzük az adott felhasználó, és az oldal karbantartója közötti legutolsó üzenetet:

```
$stmt = $db->prepare("SELECT sender_id, seen FROM messages WHERE  
    (sender_id=1 and receiver_id=?) or (sender_id=? AND  
    receiver_id=1) ORDER BY m_date DESC limit 1");  
$stmt->bind_param("ii", $currentUserId, $currentUserId);  
$stmt->execute();  
$result = $stmt->get_result();  
$row = $result->fetch_assoc();
```

A lekérdezés eredménye a \$row változóban lesz tárolva egy asszociatív tömbként, ezt az eredményt vizsgálva eldöntjük, hogy volt-e korábbi üzenet az adott felhasználó és az oldal karbantartója között, ha igen, akkor megjelenítünk egy a beszélgetésre navigáló mezőt:

```

if ($row['sender_id'] != null) {
    echo "<a class='contact' style='background-color:#ffcccb;margin-
    bottom:40px;border:2px solid
    firebrick'href='chat.php?partner=1'>
    <div class='contact-user' >Az oldal karbantartójának
    üzenete</div>";
    ...
}

```

Ezután abban az esetben, ha a legutóbbi üzenetet a másik fél küldte, és a bejelentkezett felhasználó ezt az üzenetet még nem látta, akkor megjelenítünk egy új üzenetre figyelmeztető szöveget is:

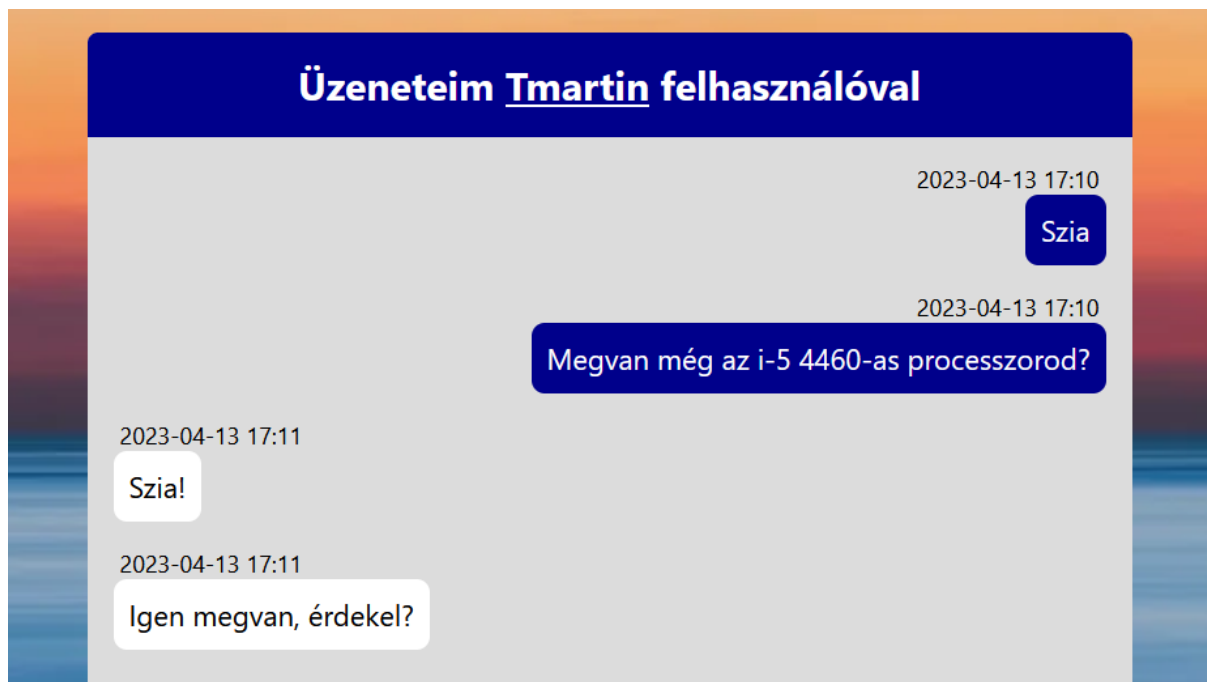
```

if ($row['sender_id'] != null) {
    ...
    if ($row['sender_id'] == 1 && $row['seen'] == 0){
        echo "<div class='contact-new-message'>Új!</div>";
    }
    echo "</a>";
}

```

Hasonló módon lekérdezések segítségével megjelenítjük a többi felhasználót is, akikkel a bejelentkezett felhasználónak volt korábban beszélgetése. Először azokat a felhasználókat jelenítjük meg, akiktől kaptunk új, még nem látott üzenetet.

Az imént említett mezőkre kattintva az alkalmazás átnavigál az adott felhasználóval való beszélgetésbe, ahol megjelenítésre kerülnek a két felhasználó egymásnak küldött üzenetei, a 22. ábra alapján. Egyúttal az oldalra navigáláskor frissítve lesz az adatbázis, minden, a partnerünk által küldött új üzenet „seen” értéke 1 lesz, hiszen minden üzenetét láttuk.



22. ábra: A beszélgetések megjelenítése

Új üzenetet egy szöveges beviteli mező kitöltése utáni „küldés” gombra kattintással küldhetünk, ezzel az üzenetünk JQuery segítségével, az előző fejezetekben tárgyalt módon fel lesz töltve az adatbázisba az oldal frissítése nélkül, majd az új üzenetünk a mi oldalunkon meg lesz jelenítve. A beszélgetés valós időben frissül, ami azt jelenti, hogy a partnerünk által küldött üzenete az oldal újratöltése nélkül meg lesznek jelenítve számunkra. Ebből a célból egy függvény segítségével adott időközönként (jelen példában 5 másodperc) meghívásra kerül a „Refresh” függvény:

```
setInterval(Refresh, 5000);
```

Ez a függvény pedig JQuery segítségével meghív egy php állományt, melytől válaszként a partnerünk új, általunk még nem látott üzeneteit várja, a válasz megérkezése után pedig megjeleníti azokat az üzeneteket a felhasználói felületen. Az említett php állományban lekérdezzük az összes, a partnerünk által küldött új üzenetet, majd ezeket eltároljuk egy tömbben:

```

$stmt = $db->prepare("SELECT * FROM messages WHERE sender_id=? AND
    receiver_id=? AND seen=0");
$stmt->bind_param("ii", $sender, $receiver);
$stmt->execute();
$result = $stmt->get_result();

$messages = array();
while ($row = $result->fetch_assoc()) {
    array_push($messages, array('date' => substr($row['m_date'],0,
        -3), 'text' => $row['text']));
}

```

Ezt követően átállítjuk ugyanezen üzenetek „seen” értékét az adatbázisban és visszaküldjük válaszként az eltárolt üzeneteket, hogy azok megjelenítésre kerülhessenek.

Az alkalmazás egyes oldalainak navigációs sávján az „üzeneteim” menüpont mellett meg lesz jelenítve az is, hogy hány felhasználótól kaptunk új üzenetet, ezt a számot az alábbi lekérdezés segítségével hozzuk létre:

```

$stmt = $db->prepare("SELECT sender_id FROM messages WHERE
    receiver_id=? AND seen = 0 GROUP BY sender_id");
$stmt->bind_param("i", $_SESSION['hargeraCurrentUserIdhargera']);
$stmt->execute();
$stmt->store_result();
$newMessages = $stmt -> num_rows();

```

Melyet ezután a \$newMessages változó segítségével jelenítünk meg:

```

<li><a href="messages.php">Üzeneteim<?php if($newMessages != 0){
    echo "<div class='new-message'> {$newMessages} </div>"
}></a></li>

```

3.4.9 Főoldal – Hirdetések közötti keresés

A hirdetésekre alkalmazott szűrés funkcióját dinamikusan, az alkalmazott szűrők alapján bővülő lekérdezésekkel lett megoldva. 2 lekérdezésre volt szükségünk, az elsővel meghatározásra került, hogy hány találat van összesen az adott keresésre. Erre azért van szükség, mert ennek segítségével el tudjuk dönteni, hogy hány oldal szükséges összesen a találatok megjelenítéséhez. A másodikban pedig lekérdeztük az adatbázisból az adott oldalon

megjelenítendő hirdetések adatait. Ezek alapján a lekérdezéseket előkészítettük a következő módon:

```
$query1 = "SELECT Count(*) FROM advertisement as a WHERE 1=1";
$query2 = "SELECT u.username as user, a.title, a.price, a.shipping,
            a.settlement, u.score, a.date, a.id, a.user_id FROM
            advertisement as a left JOIN users as u ON a.user_id = u.id
            WHERE 1=1";
$params = array();
$types = "";
```

A lekérdezések imént említett dinamikus bővülésére egy példa a kategóriára való szűrés esetében:

```
if($category != "default"){
    $query1 = $query1 . " AND a.category LIKE ?";
    $query2 = $query2 . " AND a.category LIKE ?";
    $params[] = "%$category%";
    $types = $types . "s";
}
```

A második lekérdezést futtatás előtt az alábbi módon ki kell egészíteni, hogy a jelenlegi oldalon megjelenítendő rekordokat adja vissza:

```
$from = ($page - 1) * 10;
$query2 = $query2 . " ORDER BY a.date DESC LIMIT {$from},10";
```

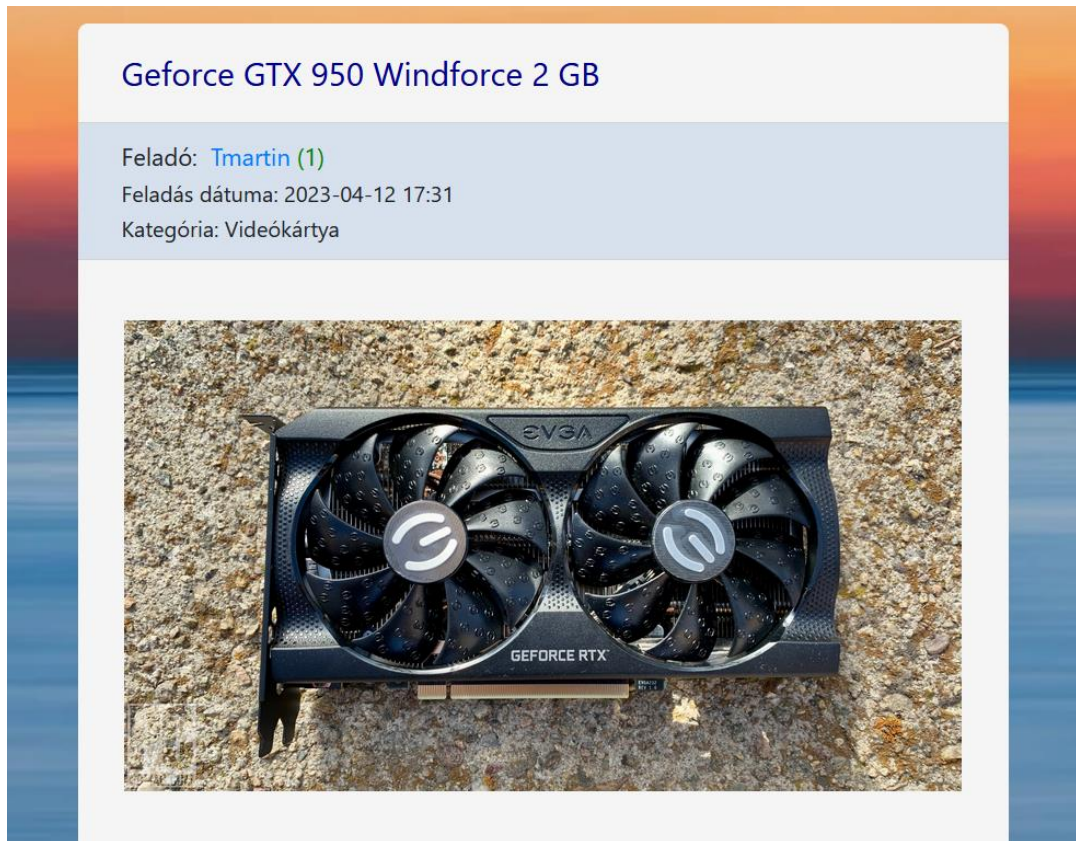
Ettől eltekintve mindkét lekérdezést ugyanazon módszerrel paraméterezhetjük, a szűrők használata esetén:

```
$stmt = $db->prepare($query2);
if (count($params) > 0){
    $stmt->bind_param($types, ...$params);
}
$stmt->execute();
```

Ezek után a maximális és a jelenlegi oldalszám összevetésével létrehozuk az előző és a következő oldalra navigáló mezőket, a szűrők és az aktuális oldalszám alapján pedig megjelenítjük az egyes hirdetéseket.

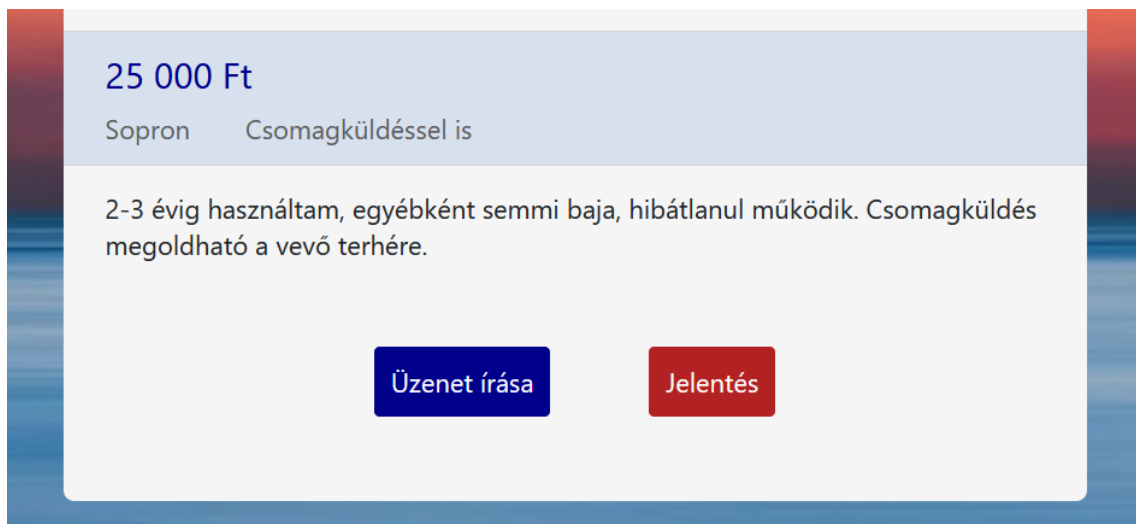
3.4.10 Hirdetés főoldala

A hirdetések főoldalán a hirdetés és a feladójával kapcsolatos adatok megjelenítése azok adatbázisból való lekérdezése után történik. A hirdetéshez feltöltött képek megjelenítéséhez a Bootstrap 4-es Carousel-t, képnézegetőt használtunk néhány kisebb változtatással.



23. ábra: A hirdetés főoldala

Emellett abban az esetben, ha egy másik felhasználó hirdetésének oldalán vagyunk, akkor elhelyezésre kerül 2 gomb, az egyikkel a hirdetés tulajdonosával való beszélgetés oldalára navigálhatunk, a másikkal pedig arra az oldalra, ahol jelentést adhatunk le az adott hirdetésről.



24. ábra: A hirdetés főoldala – navigáló gombok

3.4.11 Hirdetések jelentése

A hirdetésekről egy a bejelentkezéshez és a regisztrációhoz hasonló oldalon adhatunk le jelentést egy form segítségével, mely feltöltés után bekerül az adatbázisba. Ezeket a jelentéseket az „ADMIN” felhasználó kezelheti az általa elért oldalon melyről a 6.12-es fejezetben lesz szó.

3.4.12 Admin oldal

Az „ADMIN” felhasználót manuálisan létrehoztuk az adatbázisban, 1-es egyedi azonosítóval, hiszen „ADMIN” felhasználónévvel nem regisztrálhatunk az oldalra. Ezután egyedivé tettük ezt a felhasználót, ha Session változóban tárolt aktuálisan bejelentkezett felhasználó az admin, akkor más menüpontok lesznek megjelenítve az alkalmazás navigációs sávján. Ha ez a felhasználó linken keresztül megpróbál belépni egy olyan oldalra melyet nem szabadna elérnie (például hirdetés feltöltése), akkor egy hibaüzenet mellett át lesz navigálva a bejelentkezés oldalára, melyről bővebben a 6.13-as fejezetben lesz szó. Ha pedig egy másik felhasználó profiljára lép, akkor azon nem lesz feltüntetve az értékelés írásának lehetősége. Emellett, ha egy link segítségével valaki el szeretné érni az admin felhasználó profilját, akkor egy hibaüzenetet fog kapni miszerint nem létezik az adott felhasználó.

Jelentések	
Küldő	<input type="text"/>
Hirdetés tulajdonosa	<input type="text"/>
<input type="button" value="Keresés"/>	
Küldő:	Tmartin
Hirdetés tulajdonosa:	csbeno10
Jelentett hirdetés:	NVIDIA Geforce Rtx 3050 8GB
Jelentés dátuma:	2023-04-13 17:47

25. ábra: A jelentések kezelésének oldala

A jelentések kezelésének oldala hasonlóképpen működik, mint az alkalmazás főoldala. Egy más alatti kerülnek megjelenítésre a feladott jelentések, melyek között két szűrő segítségével kereshetünk. Az egyikkel a jelentést küldőjére a másikkal a jelentett hirdetés tulajdonosára. Ennek megvalósítása a hirdetések közötti kereséshez hasonlóan, a 6.9-es fejezetben tárgyalt dinamikusan bővülő lekérdezésekkel történik. Kettő, a szűrők alkalmazása esetén bővülő lekérdezést használunk, az első eredménye a találatok száma, mellyel meghatározzuk a maximális oldalszámot, a második lekérdezés pedig az adott oldalon megjelenítendő jelentéseket tartalmazza.

Ha kiválasztásra kerül egy jelentés akkor, a hirdetésekhez hasonlóan az alkalmazás átnavigál annak főoldalára, ahol megjelenítésre kerülnek a jelentéssel kapcsolatos adatok és a jelentéshez leadott indoklás is. Ezen az oldalon a jelentést leadó felhasználó vagy a jelentett hirdetés tulajdonosának kiválasztása esetén az alkalmazás át fog navigálni azok profiljára. Hasonló módon a jelentett hirdetésre kattintás esetén annak főoldalára kerülünk.

Az admin felhasználó a többi felhasználóhoz hasonlóan navigálhat az üzenetek küldésének oldalára: adott felhasználó profiljából, vagy az általa feladott hirdetésből. Viszont egy hirdetés főoldalára lépve a „Jelentés” gomb helyén „Hirdetés Törlése” fog megjelenni, mely hasonlóan a 6.6-os fejezetben tárgyaltakhoz eltávolítja a hirdetést az adatbázisból, illetve kitörli a hirdetés mappáját és az abban tárolt képeket a mappaszerkezetből.

3.4.13 Az oldalon való navigáció

Ha bejelentkezés nélkül próbálunk meg elérni egy olyan funkciót, mint például a hirdetések feltöltése, akkor át leszünk navigálva a bejelentkezés oldalára:

```
if (!isset($_SESSION['hargeraCurrentUserIdhargera'])) {  
    header("location: login.php?mustlogin='1'");  
}
```

Ebben az esetben használt „hargeraCurrentUserIdhargera” Session változót vizsgáljuk, melyben belépés esetén az aktuálisan bejelentkezett felhasználót egyedi azonosítóját tároljuk. Mivel a Session változókat adott böngészőn belül bármely oldal elérhető ezért célszerű olyan nevet választanunk neki, amit más weboldal nem használ. Navigálás után egy hibaüzenetet is hozzáadunk az ezeket tároló tömbhöz, hogy azt később megjeleníthessük:

```
if (isset($_GET['mustlogin'])) {  
    unset($_SESSION['mustlogin']);  
    array_push($errors, "A művelethez végrehajtásához bejelentkezés  
szükséges!");  
}
```

Hasonló átnavigálást eredményez, ha egy általános felhasználóval szeretnénk elérni a hirdetések kezelésének oldalát, vagy ha az admin felhasználóval próbálnánk elérni a profilunkat.

4. Tesztelés

Az alkalmazás teszteléséhez egy XAMPP segítségével létrehozott lokális webszerver lett használva. Első lépésként az oldal kinézete és responzivitása került megvizsgálásra a napjainkban legelterjedtebben használt böngészőkben, mint a Chrome, az Edge, a Firefox vagy az Opera. Megbizonyosodtam arról, hogy az alkalmazás oldalai esztétikusan néznek ki különböző böngészők és kijelzőméretek használata esetén.

Ezután több felhasználó is regisztrálásra került és minden felhasználó esetén ellenőrizve lettek az alkalmazás funkciói, az adatbázisban létrehozott rekordok helyessége, képek feltöltése esetén azok mappaszerkezetben megfelelő helyen való lementése, és a hirdetések és képek eltávolítása is. Minden felhasználói inputot igénylő funkciónál meg lettek vizsgálva a különböző hibalehetőségek: például üresen hagyott mezők vagy rossz kiterjesztésű fájlok feltöltése. Ezzel megbizonyosodtam arról, hogy az alkalmazás megfelelően kezeli ezeket a problémákat. Ezt követően megkértem 2 barátomat, hogy próbálják ki az oldal funkcióit, valamilyen hiányosság felmerülése esetén azt jelezzék felém.

Végül létrehozásra került az admin felhasználó és belépés után le lettek tesztelve az admin oldal funkciói is. Ily módon megbizonyosodtam arról, hogy az alkalmazás eleget tesz a specifikációban említett követelményeknek.

5. Összefoglalás

A szakdolgozat elkezdésekor az a cél került kitűzésre, hogy olyan elterjedten használt eszközök segítségével kerüljön létrehozásra az alkalmazás, melyekre későbbiekben is támaszkodhatom, ha nagyobb, komplexebb projektek fejlesztésében kell részt vennem. A dolgozat készítése közben betekintést kaptam és tapasztalatot szereztem olyan programozási nyelvek, eszközök használatába, mint például a PHP, a Javascript vagy az Ajax.

A Szakdolgozat témájának azért választottam egy használt számítástechnikai eszközök értékesítését segítő weboldal létrehozását, mert az egyre gyorsabban fejlődő világunkban már most is hatalmas a kereslet az ilyen eszközökre, mely egyre csak nőni fog. Viszont a mai embereknek nincs túl sok szabadideje, ezért az ilyen termékek vásárlására, értékesítésére is kevés időt tudnak szánni. Szerencsére az interneten keresztül rengeteg dolgot gyorsan elintézhetünk, akár pár kattintással, ezért tűztem ki célul egy könnyen kezelhető weboldal létrehozását, mely megkönnyítheti rengeteg ember mindennapjait.

A fejlesztés során sikerült létrehozni egy olyan alkalmazást, mely megfelelt az elvárásoknak. Ezzel párhuzamosan sokat tanultam, és átfogóbb képet kaptam a webes alkalmazások fejlesztéséről. A jövőben szeretném ezt a tudásomat a weboldallal együtt fejleszteni.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Tiba Attilának
amiért a félév során nyújtott észrevételeivel és tanácsaival segítette a
szakdolgozat megírását.

6. Irodalomjegyzék

- [1] HTML jelentése. URL: <https://lexiq.hu/html> (elérés dátuma 2023.03.16.)
- [2] Mit is jelent pontosan az a HTML kifejezés? URL: <https://matebalazs.hu/html.html> (2023.03.16.)
- [3] Mi az a CSS? – Digitális kiadványok. URL: <https://digikiad.gitbook.io/digitalis-kiadvanyok/css/mi-a-css> (elérés dátuma 2023.03.16.)
- [4] Mi az a HTML, CSS, Javascript? És hogyan működnek a weboldalon? URL: <https://webshark.hu/hirek/html/> (elérés dátuma 2023.03.16.)
- [5] Mi az a Javascript? A Javascript (JS) bemutatása – WEBiskola. URL: <https://webiskola.hu/javascript-ismeretek/mi-az-a-javascript-a-js-bemutatasa/> (elérés dátuma 2023.13.18.)
- [6] Mit is jelent pontosan a Javascript kifejezés? URL: <https://matebalazs.hu/javascript.html> (elérés dátuma 2023.03.18.)
- [7] Mi az a Bootstrap. URL: <https://pallas70.hu/frontend-fejleszto/mi-a-bootstrap> (elérés dátuma 2023.03.18.)
- [8] Mi az a Bootstrap 4? Hogyan érdemes használnunk? URL: <https://gremmedia.hu/mi-az-a-bootstrap-4-hogyan-hasznaljuk> (elérés dátuma 2023.03.18.)
- [9] Mi az a MariaDB? URL: <https://www.awh.hu/kb/webtarhely/mi-az-a-mariadb> (elérés dátuma 2023.03.19.)
- [10] Mi a MariaDB? URL: <https://fiberhost.hu/faq/help-center/articles/33/mi-a-mariadb> (elérés dátuma 2023.03.19.)
- [11] SQL, MySQL hogyan használjuk őket? URL: <https://gremmedia.hu/mi-az-sql-hogyan-kell-hasznalni> (elérés dátuma 2023.03.19.)
- [12] Mi az a MySQL? URL: <https://www.awh.hu/kb/webtarhely/mi-az-a-mysql> (elérés dátuma 2023.03.19.)

- [13] PHP – Wikipédia. URL: <https://hu.wikipedia.org/wiki/PHP> (elérés dátuma 2023.03.19.)
- [14] Mi az a PHP? PHP fogalma és bemutatása. URL: <https://webiskola.hu/php-ismeretek/mi-az-a-php-fogalma-bemutatasa/> (elérés dátuma 2023.03.19.)
- [15] A PHP programozási nyelv részletes bemutatása példákkal együtt. URL: <https://gremmedia.hu/php-programozasi-nyelv-reszletes-bemutatasa-peldakkal-egyutt> (elérés dátuma 2023.03.19.)
- [16] Bevezetés a JQuery-be. URL: <https://webdesigntanfolyam.com/bevezetes-a-jquery-be/> (elérés dátuma 2023.03.20.)
- [17] Mi az a JQuery, miért használjuk? URL: <https://gremmedia.hu/mi-az-a-jquery-a-legjobb-jquery-widgetek> (elérés dátuma 2023.03.20.)
- [18] Ajax (programozás) – Wikipédia URL: [https://hu.wikipedia.org/wiki/Ajax_\(programoz%C3%A1s\)](https://hu.wikipedia.org/wiki/Ajax_(programoz%C3%A1s)) (elérés dátuma 2023.03.20.)
- [19] Mi az az Ajax? URL: <https://hwellkft.hu/marketing-szotar/ajax> (elérés dátuma 2023.03.20.)
- [20] PHP, Apache és XAMPP letöltése, telepítése. URL: <https://webiskola.hu/php-ismeretek/php-apache-xampp-letoltese-telepitese/> (elérés dátuma 2023.03.20.)
- [21] A History of Html5 and How It Works. URL: <https://www.fasthosts.co.uk/blog/look-back-html5/> (elérés dátuma 2023.03.16.)
- [22] HTML Forms. URL: https://www.w3schools.com/html/html_forms.asp (elérés dátuma 2023.03.06.)
- [23] PHP Documentation. URL: <https://www.php.net/docs.php> (elérés dátuma 2023.03.25)
- [24] JavaScript and HTML DOM Reference. URL: <https://www.w3schools.com/jsref/> (elérés dátuma 2023.03.08.)
- [25] JQuery Tutorial. URL: <https://www.w3schools.com/jquery/default.asp> (elérés dátuma 2023.03.22.)
- [26] SQL Tutorial. URL: <https://www.w3schools.com/sql/default.asp> (elérés dátuma 2023.03.12.)

- [27] Szabó Bálint: Webprogramozás I. 2013. URL:
<https://mek.oszk.hu/14000/14068/pdf/14068.pdf> (elérés dátuma 2023.02.16.)
- [28] Bruce Lawson, Remy Sharp: Bemutakozik a Html 5. Perfact-Pro Kft. 2013, ISBN:
9789639929289
- [29] Richard Blum: PHP, MySQL & JavaScript 7 könyv 1-ben. Taramix Kft. 2020, ISBN:
9786155186721
- [30] Laura Thomson, Luke Welling: PHP and MySQL Web Development, Fifth Edition.
Addison-Wesley Professional, 2016, ISBN: 9780321833891