



**EÖTVÖS LORÁND TUDOMÁNYEGYETEM**  
**INFORMATIKAI KAR**  
**PROGRAMOZÁSI NYELVEK ÉS**  
**FORDÍTÓPROGRAMOK TANSZÉK**

---

## **Eladó autókat kezelő program**

Témavezető:

**Mészáros Mónika**

Mestertanár

ELTE IK

Szerző:

**Viczián Lilla**

programtervező informatikus BSc

nappali tagozat

Külső konzulens:

**Szita Balázs**

Szoftvermérnök

Nokia Solutions and Networks Kft

**Budapest, 2017.**

# Tartalomjegyzék

1	Bevezetés .....	3
2	Felhasználói dokumentáció.....	5
2.1	Hardver- és szoftverkövetelmények .....	5
2.2	Program használatához szükséges telepítések .....	6
2.2.1	Google Chrome .....	6
2.2.2	JDK.....	6
2.2.3	MySQL telepítése.....	7
2.2.4	Tomcat telepítése és program felmásolása.....	10
2.3	Program indítása .....	10
2.4	Vendég funkciói.....	11
2.4.1	Keresés és hirdetés megtekintése .....	11
2.4.2	Regisztráció .....	14
2.4.3	Bejelentkezés.....	15
2.5	Bejelentkezett felhasználó funkciói .....	15
2.5.1	Általános tudnivalók a felhasználókról .....	15
2.5.2	Hirdetés megtekintése, törlése, kedvencekhez adása, exportálása, kommentezése .....	16
2.5.3	Kedvencek megtekintése .....	17
2.5.4	Új hirdetés feladása .....	18
2.5.5	Hirdetésem megtekintése .....	19
2.5.6	Hirdetésem szerkesztése.....	20
2.5.7	Profil szerkesztése .....	20
2.5.8	Kijelentkezés .....	21
2.6	Admin funkciói.....	21
2.6.1	Általános információk az adminról .....	21
2.6.2	Régi hirdetések .....	21
2.6.3	Hirdetés törlése.....	22
2.6.4	Hozzászólások moderálása.....	23
2.7	Párhuzamos használat .....	23
3	Fejlesztői dokumentáció .....	24
3.1	A feladat.....	24

3.2	Megoldási terv .....	25
3.2.1	Use case diagram.....	25
3.2.2	Adatbázis .....	27
3.2.3	A kinézeti terv .....	28
3.2.4	Megvalósítandó menüpontok .....	28
3.3	Az MVC elve és megvalósítása.....	30
3.4	Megvalósítás .....	31
3.5	Fejlesztői környezet telepítése .....	32
3.5.1	Google Chrome .....	32
3.5.2	JDK.....	32
3.5.3	JetBrains IntelliJ IDEA .....	33
3.5.4	Tomcat telepítése.....	33
3.5.5	MySQL telepítése.....	36
3.6	A projekt mappaszerkezete.....	39
3.7	Fájlok leírása.....	41
3.7.1	MVC architektúra.....	43
3.7.1.1	Controller .....	43
3.7.1.2	Model .....	47
3.7.1.2.1	Entitások .....	47
3.7.1.2.2	Service-ek .....	50
3.7.1.3	View .....	52
3.8	Teszt.....	55
3.8.1	Use Case tesztek.....	55
3.8.1.1	Validációs szempont .....	55
3.8.1.2	Párhuzamos használat .....	60
3.8.2	Unit tesztek.....	62
3.9	Programkódból felhasználóbarát verziót - Maven telepítése.....	68
4	Továbbfejlesztési lehetőségek.....	69
5	Összegzés .....	70
6	Melléklet .....	71
7	Irodalomjegyzék.....	72

# 1 Bevezetés

Szakedolgozatom témája egy webes alkalmazás, amelyben eladó autókat tudunk kezelni.

Napjainkban az internet térhódításának köszönhetően, szinte bármit meg tudunk venni online is. Azonban sajnos nem mindig olyan egyszerű eligazodni, kiszűrni azt a terméket, amelyre szükségünk lenne. Főleg nem egy olyan weboldalon, amely tele van reklámmal, felesleges információkkal, rosszul elrendezett és túl régi vagy szinte üres hirdetéseket tartalmaz. Ez nem előnyös sem a kereső félnek, mert nehézkesen vagy egyáltalán nem is fogja megtalálni azt, amit keres, de az oldal üzemeltetőjének sem, hisz, így egyre kevesebben fogják látogatni az oldalát.

Ennek szellemében készítettem egy letisztult, könnyen átlátható weboldalt, amelyen megkövetelt a hirdetések pontos kitöltése.

Az oldalon regisztráció és belépés nélkül lehet eladó autókat keresni különböző szempontok szerint. A keresés eredménye dátum és ár szerint is sorba rendezhető. Az így kilistázott autók valamennyien megtekinthetők, a tulajdonságaikkal együtt. Az, hogy az autót melyik városban árulják, nem csak szövegesen, de grafikusan, térkép segítségével is megtekinthető. Ezenkívül a vendégnek van lehetősége bejelentkezni az oldalra, vagy ha még nem regisztrált, akkor először ezt megtenni, és azután bejelentkezni. Regisztráció esetén mindenki felhasználóvá válik. Az autentikációt követően, a keresés mellett, számos más funkcióra van lehetősége a felhasználónak. A regisztrációkor létrehozott profilját – a felhasználónév kivételével – szerkesztheti, új hirdetést tölthet fel, amelyet később szerkeszthet, vagy törölhet. A hirdetés feltöltése csak abban az esetben lesz sikeres, ha a rendelkezésre álló mezők közül, meghatározott számút megfelelően kitöltött. Ez a kitöltöttségi faktor hozzájárul ahhoz, hogy a keresés pontosabb legyen, és így gyorsabb az oldalt használók számára, valamint megakadályozza a szinte üres hirdetések feladását. A felhasználó az összes feladott – és még nem törölt – hirdetését megtekintheti egy helyen. Autók keresésekor, a találatok közül egy tetszésszerintit választva lehetőségünk van azt a kedvencekhez adni, vagy ha már ott van, akkor törölni a kedvencekből. A megtekintéskor látott szöveges adatokat exportálhatjuk is. Ezenkívül a felhasználónak joga van hozzászólást írnia bármely hirdetéshez. A webes alkalmazást admin ellenőrzi. Az admin a

felhasználói jogok mellett, egy hirdetést megtekintve képes a hirdetés hozzászólásait moderálni, illetve törölni az egész hirdetést. A hirdetések törlésére egy külön felülete is van, dátum szerint rendezve, így megkönnyítve a túl régi hirdetések egyszerre történő törlését.

A program Java nyelven készült, támaszkodva a Java Spring keretrendszer szolgáltatásaira.

## 2 Felhasználói dokumentáció

### 2.1 Hardver- és szoftverkövetelmények

A program platformfüggetlen, de a továbbiakban Microsoft Windows 10 operációs rendszert feltételezünk.

A program működéséhez szükség van

- az 1.8-as verziójú Java Development Kit-re,
- egy böngészőre, például Google Chrome-ra,
- a MySQL szerverre,
- a CD-n mellékelt fájlokra (a Tomcat szerver is itt található.)

Mindehhez szükség van megközelítőleg 850 MB tárhelyre a számítógépen. (Ennek jelentős részét az adatbázis teszi ki.)

Az alkalmazás egér és billentyűzet megszokott használata mellett működtethető.

Javasolt a Google Chrome nevű böngészőt teljes képernyős nézetben és 100%-os nagyítás mellett használni. Az említett böngésző a Google hivatalos oldaláról ingyenesen letölthető.<sup>1</sup>

Az exportált xls kiterjesztésű fájlok megtekintéséhez szükséges egy xls fájlokat olvasó programra. Ilyenek például a Microsoft Excel különböző verziói.

---

<sup>1</sup><https://support.google.com/chrome/answer/95346?co=GENIE.Platform%3DDesktop&hl=hu>

## 2.2 Program használatához szükséges telepítések

### 2.2.1 Google Chrome

A Google hivatalos oldaláról letölthető. Letöltés után könnyedén telepíthető.<sup>2</sup>

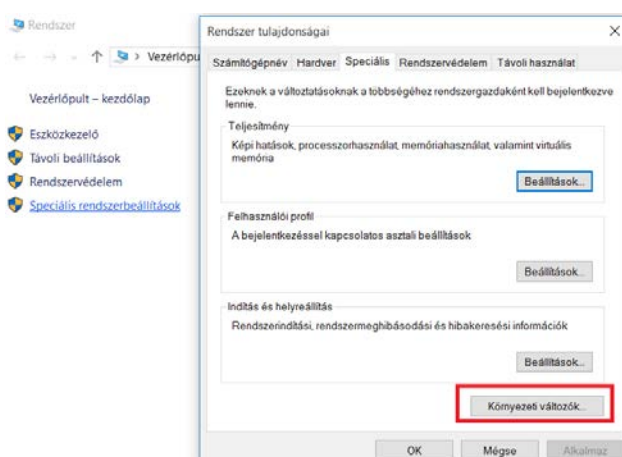
### 2.2.2 JDK

Az 1.8-as verziószámú JDK-t (Java Development Kit) először le kell töltenünk a hivatalos weboldáról.<sup>3</sup> Az exe kiterjesztésű alkalmazást tudjuk futtatni, ezzel feltelepül a program. Telepítés után állítsuk be a megfelelő környezeti változókat, például ha a C:\Program Files\Java helyre telepítettük a JDK-t, úgy a JAVA\_HOME környezeti változónak meg kell adni a C:\Program Files\Java\jdk1.8.0\_121 értéket, a JDK telepítési helyét.

Ugyanezt az elérési útvonalat állítsuk be a Path rendszerváltozóhoz is.

(Segítség a környezeti változók beállításához:

- az egér jobb gombjával kattintsunk a Startmenüre, válasszuk a Rendszer menüpontot,
- ezen belül a Speciális rendszerbeállításokat, majd a Környezeti változók... gombot. (1. ábra)
- Ezután a megfelelő környezeti változóra állva, a Szerkesztés gombbal tudjuk szerkeszteni az elérési útvonalakat.)



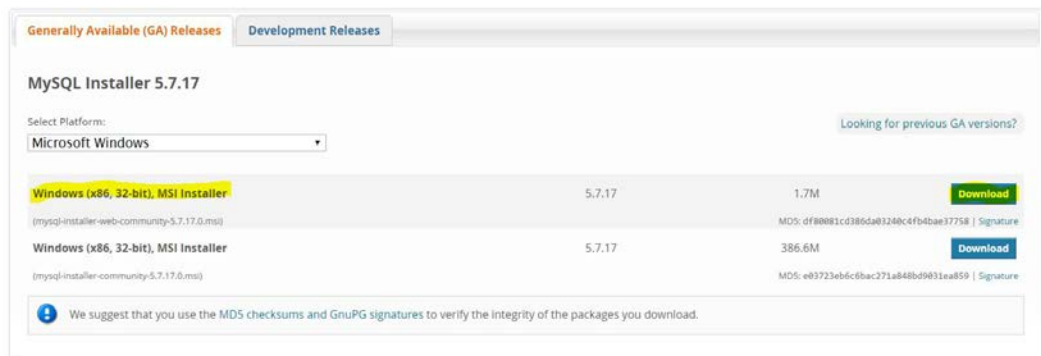
1. ábra Környezeti változók beállítása

<sup>2</sup><https://www.google.com/intl/hu/chrome/browser/desktop/>

<sup>3</sup><http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

## 2.2.3 MySQL telepítése

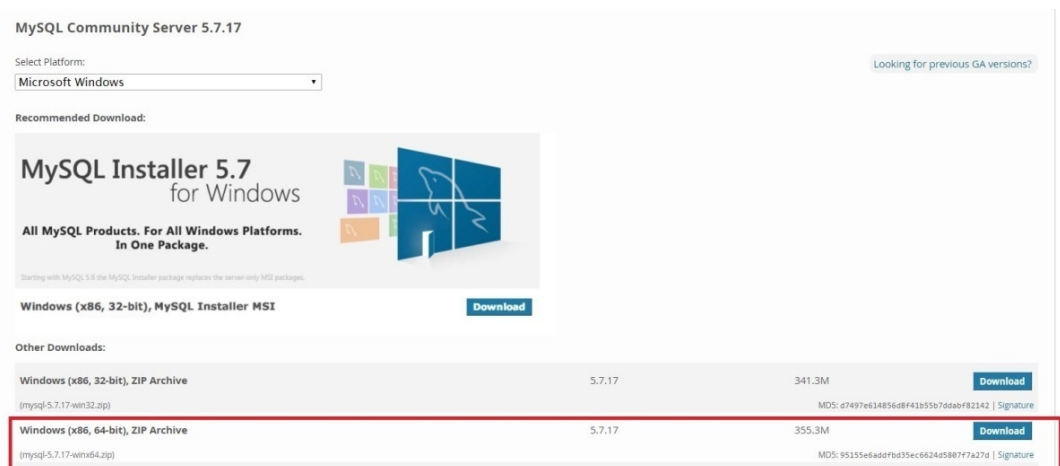
Az adatbázis-kezelő letöltése:<sup>4</sup>



2. ábra Töltsük le a kijelölt elemet!

A 32 bites MySQL Installer 5.7.17 (vagy ennél újabb verzió) megfelelő a 32 bites és 64 bites Windowsra is.

Amennyiben automatikusan nem töltődött le, töltsük le a MySQL szervert is.<sup>5</sup>



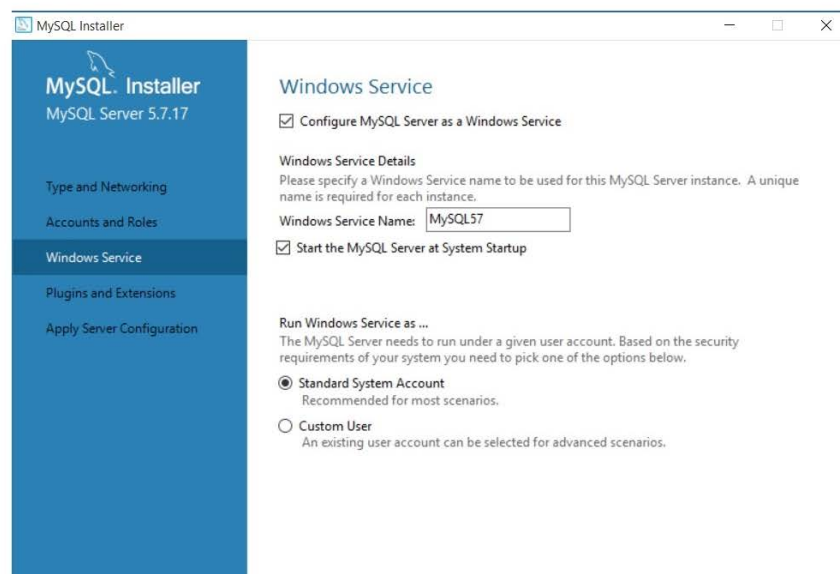
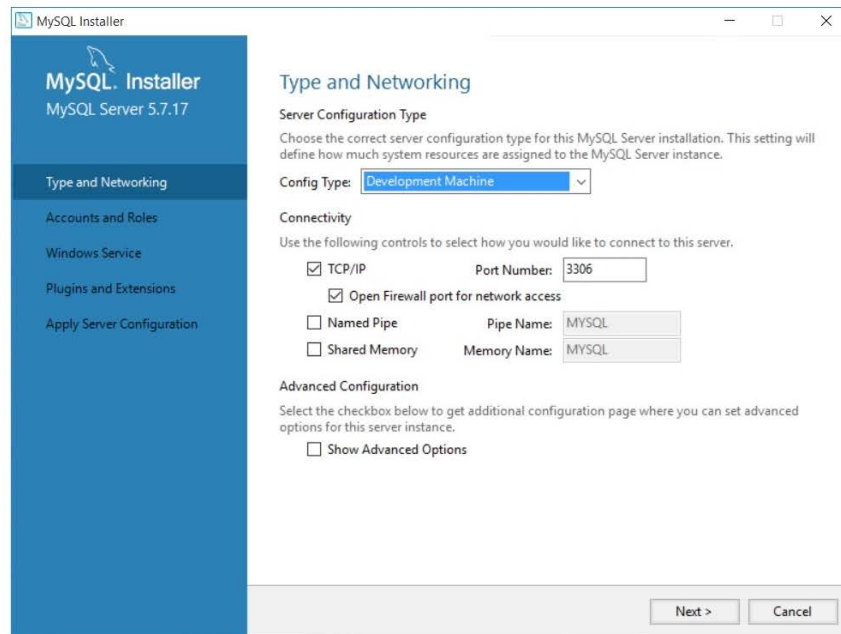
3. ábra Töltsük le a gépünknek megfelelő szervert

Telepítésnél mindent hagyhatunk az alapértelmezett beállításon, de figyeljünk, mert hozzá kell adnunk a szervert, és egy felhasználót (4. ábra, 5. ábra.) (Ez legyen: „admin” felhasználónevű és „asdf” jelszavú.)

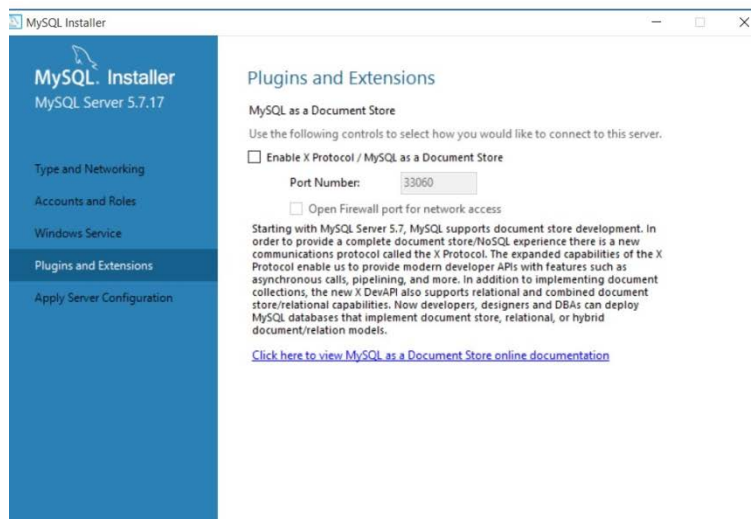
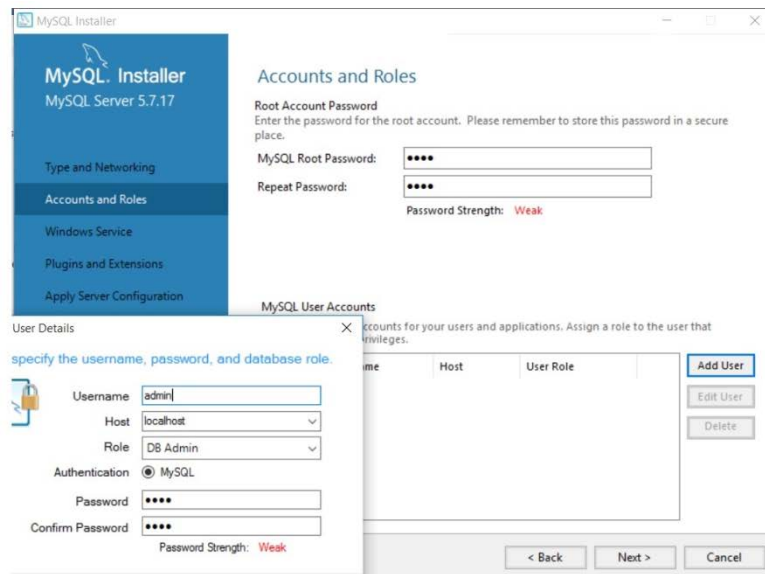
<sup>4</sup><https://dev.mysql.com/downloads/installer/>

<sup>5</sup><https://dev.mysql.com/downloads/mysql/>



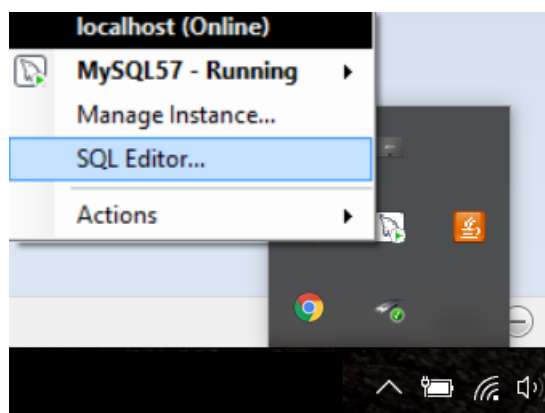


4. ábra A MySQL telepítésének kérdéses részei

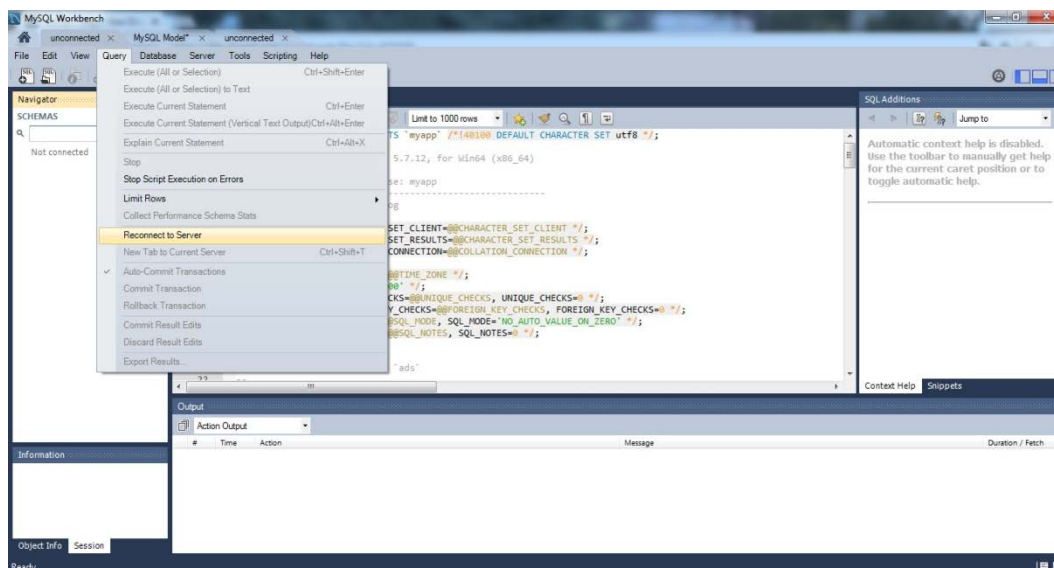


5. ábra Adatházis további kérdéses részei

Nyissuk meg a telepített SQL Editort:



6. ábra A tálcáról elérhető a felület



7. ábra Csatlakoztassuk a MySQL servert

A felületről nyissuk meg a CD-n található adatbazis.sql fájlt:



8. ábra Kattintsunk az ikonra

A fájl tartalmát futtassuk. Így létrejött az a programhoz szükséges adatbázis.

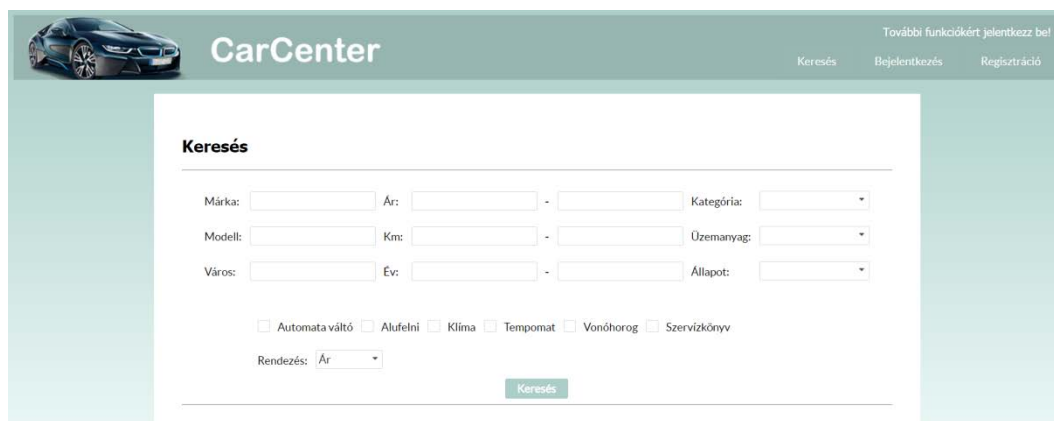
## 2.2.4 Tomcat telepítése és program felmásolása

A CD-n található Felhasználói mappa tartalmát – benne a Tomcat szerverrel, és a futtatáshoz szükséges egyéb fájlokkal – másoljuk fel a C:\ meghajtóra.

## 2.3 Program indítása

A C:\apache-tomcat-9.0.0.M17\apache-tomcat-9.0.0.M17\bin mappában a startup.bat fájljal manuálisan indíthatjuk a szervert, majd a Google Chrome böngészőnket. Írjuk be a címsorba a *localhost:8080/MyApp/index.xhtml* elérési útvonalat. Ekkor megkezdhetjük az alkalmazás tényleges használatát.

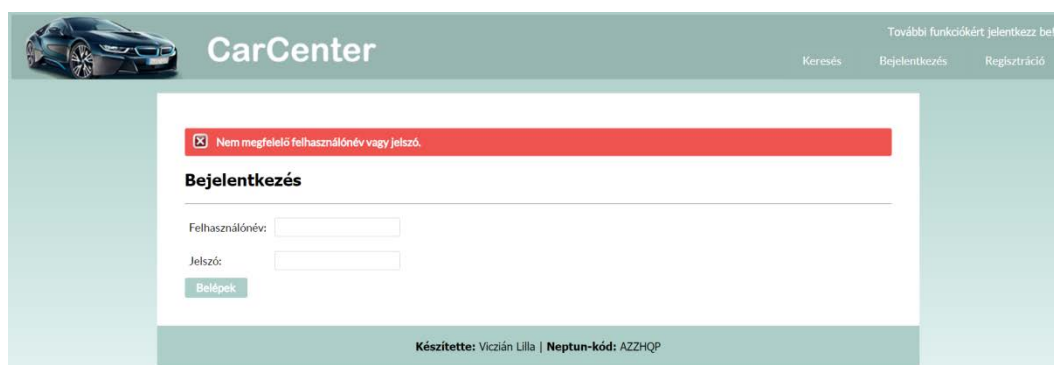
A weboldal megnyitása után az alábbi oldal fog betöltődni:



The screenshot shows the CarCenter website's main interface. At the top, there's a header with a car image, the 'CarCenter' logo, and navigation links: 'Keresés', 'Bejelentkezés', and 'Regisztráció'. A link 'További funkciókért jelentkez be!' is also present. The main content area features a 'Keresés' (Search) section with a form containing several input fields: 'Márka' (Brand), 'Ár' (Price), 'Kategória' (Category), 'Modell' (Model), 'Km', 'Üzemanyag' (Fuel), 'Város' (City), 'Év' (Year), and 'Állapot' (Status). Below these are checkboxes for 'Automata váltó', 'Alufelni', 'Klíma', 'Tempomat', 'Vonóhorog', and 'Szervízkönyv'. A 'Rendezés' (Sort) dropdown is set to 'Ár'. A green 'Keresés' button is at the bottom of the form.

**9. ábra Főoldal**

Az alkalmazásban a felső menüsor segítségével könnyen navigálhatunk az elérhető funkciók között. Ha a keresésnél, bejelentkezésnél, regisztrációnál, profil szerkesztésénél, hirdetés feltöltésénél, hirdetés szerkesztésénél hiba lépett fel a nem megfelelő kitöltés miatt, azt a menüsor alatt, de az aktuális aloldal törzse felett fogjuk látni hibüzenet formájában.



This screenshot shows the CarCenter website with a login error. A red banner at the top of the main content area displays the message: 'Nem megfelelő felhasználónév vagy jelszó.' (Incorrect username or password). Below this, the 'Bejelentkezés' (Login) section is visible, featuring input fields for 'Felhasználónév' (Username) and 'Jelszó' (Password), and a green 'Belépek' (Login) button. The footer contains the text 'Készítette: Viczián Lilla | Neptun-kód: AZZHQP'.

**10. ábra Hibás bejelentkezés**

A program szerverét, használat után ne felejtsük el leállítani, a shutdown.bat fájl futtatásával.

## 2.4 Vendég funkciói

### 2.4.1 Keresés és hirdetés megtekintése

Amennyiben nem regisztrálunk és/vagy nem jelentkezünk be, akkor is van lehetőségünk keresni a hirdetések között. A kezdőoldalon (amit a menüsáv *Keresés* gombjával is elérünk) böngészhetünk az összes hirdetés között. Amennyiben az összes találat soknak bizonyul, a keresés funkcióval szűkíthetünk rajta, amit szintén ezen az oldalon érhetünk el. Az autók számos tulajdonsága

alapján tudunk a találatokra szűrni, a megfelelő keresési mezők kitöltésével. Ha egy mezőt nem töltünk ki, arra nem vonatkozik a keresés. A szöveges mezők bevitelénél (márka, modell, város) maximum 20 karakter engedélyezett, ezenkívül nincs más megkötés, hogy milyen szóra kereshet a felhasználó. Azonban fontos tudni, hogy csak a pontosan megegyező találatok lesznek kilistázva. Ha az autó például márkája eltér akárcsak egy karakterrel is a keresésbeli márka mezőben megadott szótól, akkor az már nem számít érvényes találatnak. Ugyanígy működik a modell és a város keresése is.

Az ár, km, év mezőkbe csak számok írhatóak, az első mezőbe a megfelelő intervallum alsó, a másodikba a felső határát kell írni. Ha a felhasználó ezt a kettőt felcseréli, azt a program nem érzékeli hibának, csupán nem lesz találat a keresésre. Ennél a három input elemnél kisebb vagy egyenlő és nagyobb vagy egyenlő értékek adhatóak meg, azaz, ha például 2000-től 2000-ig indítjuk a keresést, akkor amennyiben van 2000-rel egyenlő, megfelelő adat, akkor az kilistázásra kerül. Az árnál és a km-nél maximum kilenc, az évnél négy számjegy engedélyezett. Ezeknél az adatoknál nem muszáj mind a két mezőt kitölteni. Amennyiben csak az elsőt, vagy csak a másodikat töltjük ki a keresés során, úgy annak az adatnak csak az alsó vagy csak a felső korlátja adódik hozzá a kereséshez.

A kategória, üzemanyag, állapot legördülő mezők kitöltése is opcionális.

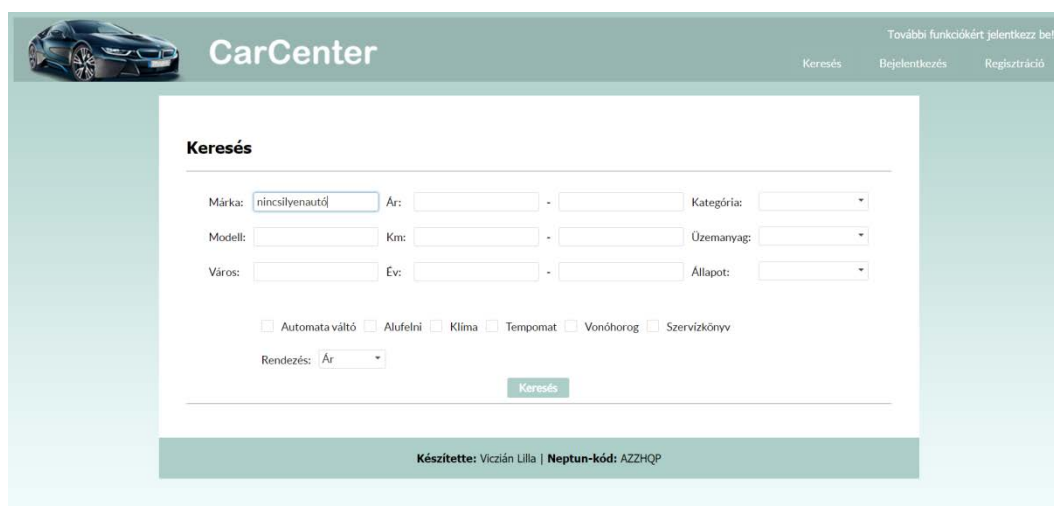
Bizonyos tulajdonságok és extrák meglétét a nekik megfelelő jelölőnégyzet (checkbox) bepipálásával adhatjuk hozzá a keresési feltételekhez. Amennyiben ezeket nem pipáljuk be, a keresés nem veszi figyelembe ezen tulajdonságokat.

Például a klíma checkbox bepipálása esetén csak a klímával rendelkező autókat fogja a program listázni, különben mind a klímás, mind a klímával nem rendelkezőket. Ez az elv igaz az összes checkboxra:

- automataváltó,
- alufelni,
- klíma,
- tempomat,
- vonóhorog
- és szervizkönyv.

Rendezhetjük is a keresés eredményét ár szerint növekvően, vagy dátum szerint növekvően. Alapértelmezetten ár szerint fognak növekedni a hirdetések.

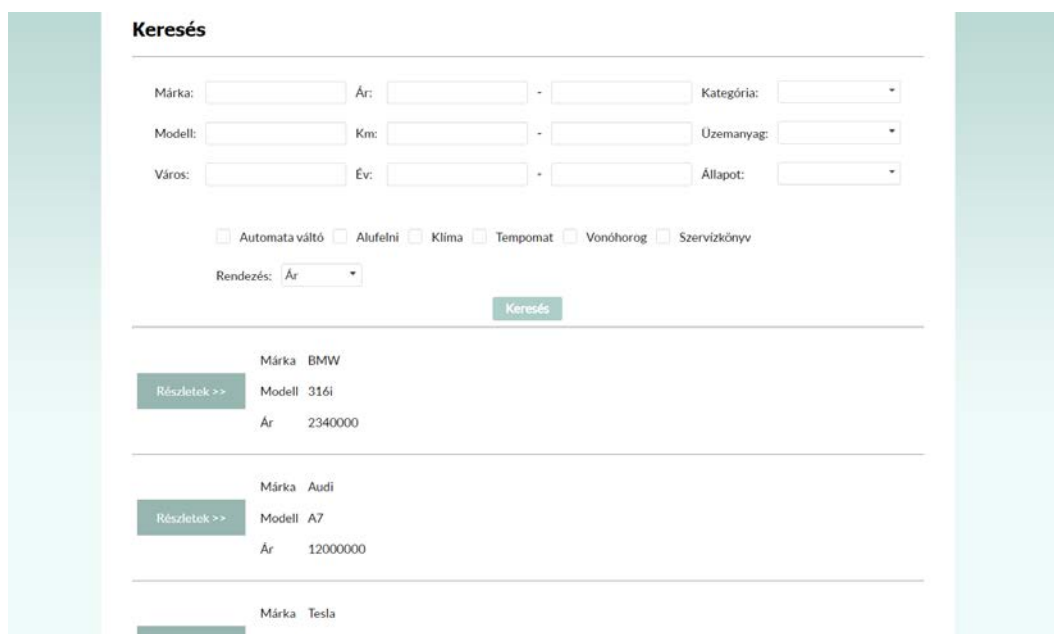
Ha minden keresési feltételt megadtunk, amit szerettünk volna, akkor kattintsunk a *Keresés* gombra. Sikeres keresés esetén a gomb alatt listázódnak a találatok. Amennyiben a keresési feltételeknek megfelelően nem talált autót a program, úgy egyetlen elem sem lesz kilistázva (11. ábra.)



The screenshot shows the CarCenter website's search interface. At the top, there's a header with the CarCenter logo and navigation links: Keresés, Bejelentkezés, and Regisztráció. The main search form is titled 'Keresés' and includes several input fields: Márka (set to 'nincsilyenautó'), Ár (range), Kategória (dropdown), Modell, Km (range), Üzemanyag (dropdown), Város, Év (range), and Állapot (dropdown). Below these are checkboxes for features: Automata váltó, Alufelni, Klíma, Tempomat, Vondóhorog, and Szervizkönyv. A 'Rendezés' dropdown is set to 'Ár'. A green 'Keresés' button is at the bottom of the form. At the very bottom, a footer note reads: 'Készítette: Viczián Lilla | Neptun-kód: AZZHQP'.

**11. ábra** Érvényes keresés eredmény nélkül

Validációs hiba esetén hibaüzenetet kapunk. (Érvénytelen adatok megadása során kerülhet elő ilyen hiba, például, ha a felhasználónak sikerül betűt írnia egy számot váró mezőbe.)



The screenshot shows the search results page on CarCenter. The search form is identical to the previous one. Below the form, a list of search results is displayed. Each result shows the 'Márka' (Brand), 'Modell' (Model), and 'Ár' (Price). The first result is for a BMW 316i with a price of 2340000. The second result is for an Audi A7 with a price of 12000000. The third result is for a Tesla. Each result has a 'Részletek >>' button next to it.

**12. ábra** Keresés alatt megjelenő hirdetések

A keresés alatt, egy tetszőleges hirdetés részleteit tudjuk megtekinteni a *Részletek* gombra kattintva. Táblázatos formában megtekinthető az autóról az összes tudnivaló, valamint térkép segítségével azonnal láthatjuk is, hogy pontosan hol helyezkedik el az a város, ahol az autót árulják (13. ábra.)

### Hirdetés megtekintése

Márka	BMW
Modell	316i
Város	Békéscsaba
Ár	2340000 Ft
Megtett km	270284 km
Év	2005
Kategória	gépkocsi
Üzemanyag	benzín
Állapot	új
Automataváltó	van
Alufelni	van
Klíma	van
Tempomat	van
Vonóhorog	van
Szervizkönyv	van
Leírás	Az autó kifogástalan állapotban van.
Hirdetést feltöltötte	admin

### Térkép

Város:

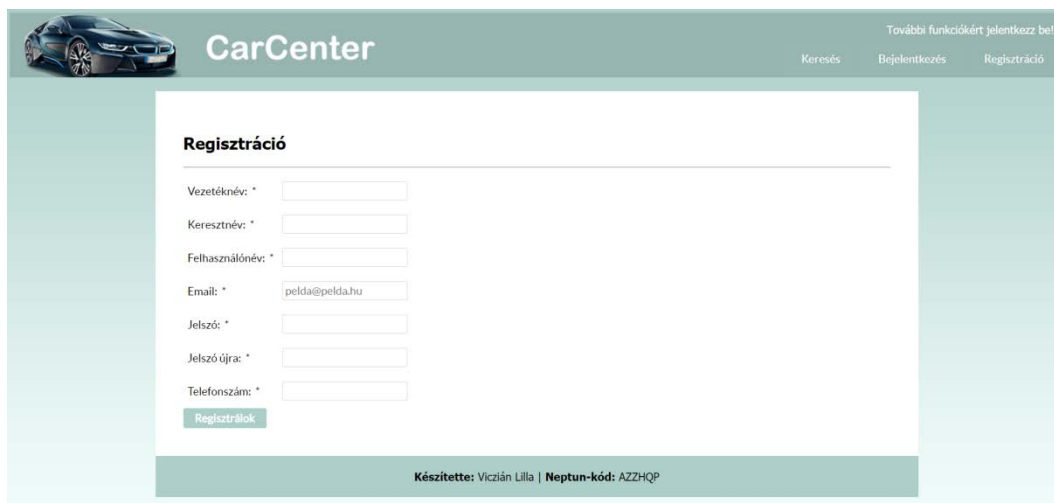
Készítette: Viczián Lilla | Neptun-kód: AZZHQP

13. ábra Hirdetés megtekintése vendégként

## 2.4.2 Regisztráció

Be nem jelentkezett állapotban elérhető a regisztrálási funkció. Minden mező kitöltése kötelező, a telefonszám maximum 13, az emailcím maximum 30, a többi mező pedig maximum 20 karakteres lehet. Az emailcím csak akkor fogadható el, ha emailcím formátuma van, (például [pelda@pelda.hu](mailto:pelda@pelda.hu)). Telefonszámnál csak számokat vár a program. Fontos, hogy már regisztrált felhasználónévvel nem lehet újra regisztrálni. Továbbá a menteni kívánt felhasználónév és jelszó csak az angol abc kis és nagy betűit, valamint számokat tartalmazhat. A jelszó és jelszó újra mezőknek egyezniük kell. Ha ezen feltételek valamelyike nem teljesül, hibaüzenetekbe fogunk ütközni.

Sikeres regisztráció esetén is kapunk visszajelzést, és így már be tudunk jelentkezni az oldalra.



14. ábra Regisztráció oldala

### 2.4.3 Bejelentkezés

Az oldal látogatója, amennyiben nincs még bejelentkezve, ezt megteheti a menüsor Bejelentkezés gombjára kattintva. A mezőket ki kell tölteni érvényes – már regisztrált felhasználó – adataival, ahhoz, hogy be tudjunk jelentkezni. Ha ennek hiányában próbálunk bejelentkezni, akkor hibaüzenetet fogunk kapni.



15. ábra Bejelentkezés oldala

## 2.5 Bejelentkezett felhasználó funkciói

### 2.5.1 Általános tudnivalók a felhasználókról

Sikeres bejelentkezés után, a menüsorban plusz gombok jelennek meg (Profil szerkesztése, Hirdetéseim, Kedvencek, Kijelentkezés.) Valamint a jobb felső



sarokban megjelenik egy üdvözlőüzenet a felhasználó nevével, és szerepével. (Felhasználó és admin lehetséges.)

A bejelentkezett felhasználó (továbbiakban felhasználó) ugyanúgy tud keresni, és hirdetést megtekinteni, mint a vendég (2.4.1). Azonban a megtekintett hirdetésnél több funkciót vehet igénybe, a vendéghez képest (Excelbe exportálás, Kedvencekhez adás, Törlés, Szerkesztés, Hozzászólás).

## 2.5.2 Hirdetés megtekintése, törlése, kedvencekhez adása, exportálása, kommentezése

A felhasználó itt tudja xls kiterjesztésű fájlba exportálni a megjelenített szöveges adatokat, azaz a hirdetés táblázatát. Az *Excelbe exportálás* gombot megnyomva le is töltődik a fájl (16. ábra.)

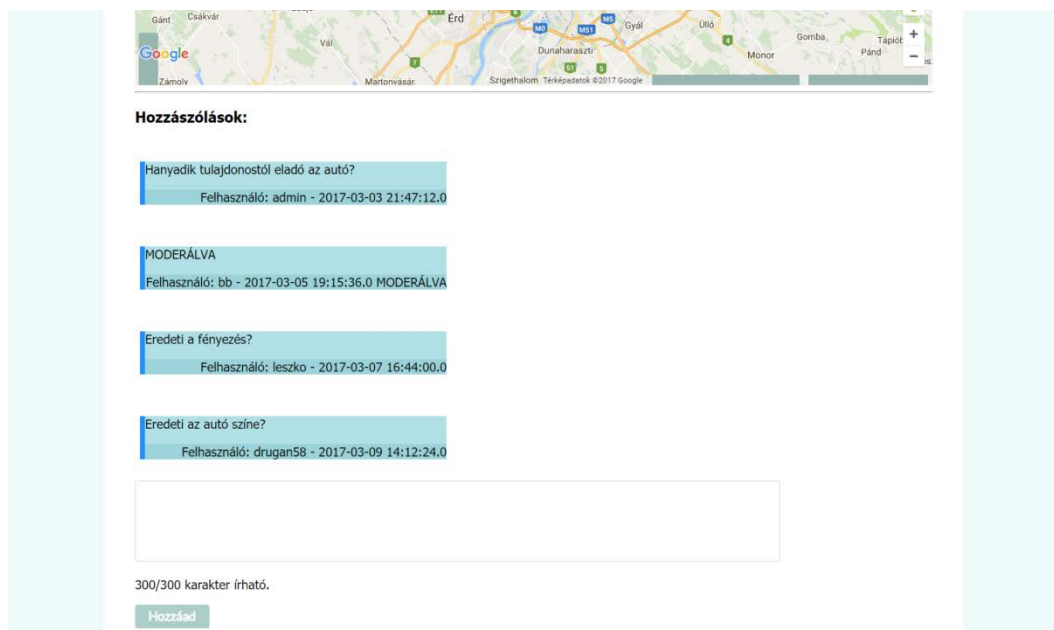
A felhasználónak lehetősége van a hirdetést a kedvencekhez adni, a *Kedvencek* gomb által. Amennyiben úgy dönt, később törölheti is a *Törlés a Kedvencekből* gomb segítségével (16. ábra.)

Amennyiben a megtekintett hirdetés saját feltöltés, úgy látunk egy *Szerkesztés* és egy *Törlés* gombot is, amelyekkel szerkeszthető vagy törölhető a hirdetés. Sikeres törlésről, szerkesztésről a felhasználó értesítést kap (16. ábra.)

A hirdetés alján a felhasználó olvashatja a korábbi hozzászólásokat – amennyiben voltak ilyenek – és írhat maga is hozzászólást (17. ábra.)

Hirdetés megtekintése	
Márka	Audi
Modell	R8
Város	Békéscsaba
Ár	30000000 Ft
Megtett km	100000 km
Év	2015
Kategória	gépkocsi
Üzemanyag	dízel
Állapot	új
Automataváltó	van
Alufelni	van
Klíma	van
Tempomat	van
Vonóhorog	nincs
Szervizkönyv	van
Leírás	Az autó nagyon jó ár-érték aránnyal rendelkezik!
Hirdetést feltöltötte	asdf
Excelbe exportálás Kedvencekhez Szerkesztés Törlés	
Térkép	

16. ábra Hirdetés megtekintése 1. rész



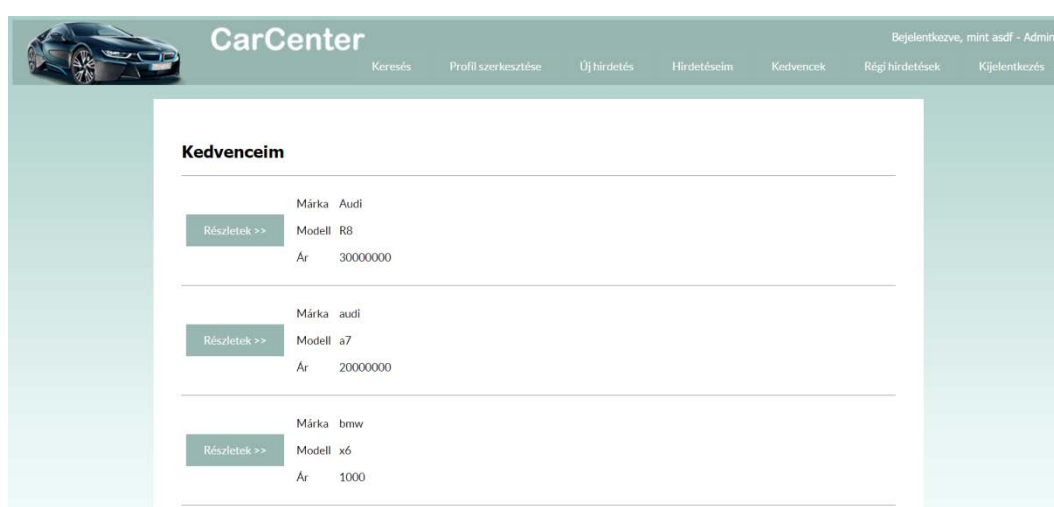
17. ábra Hirdetés megtekintése 2. rész

### 2.5.3 Kedvencek megtekintése

A menüsor *Kedvencek* gombjára kattintva megtekinthetőek a már kedvencekhez adott hirdetések (18. ábra.) Egy-egy hirdetés megtekintéséhez kattintsunk a *Részletek* gombra.

Amennyiben még nem adtunk autót a kedvencekhez, nem jelenítődik meg találat az oldalon.

A felhasználó a saját hirdetéseit is a kedvencekhez adhatja.



18. ábra Kedvenceim menüpont

### 2.5.4 Új hirdetés feladása

A felhasználó hirdetéseket adhat fel. Ezt a funkciót a menüsor *Új hirdetés* gombja segítségével érheti el (19. ábra.) Itt a márka, modell, város, ár mezők kötelezőek, a kategória, üzemanyag, és állapot értékek pedig rendelkeznek alapértelmezett értékekkel, így kitöltötnék minősülnek. Ezenkívül a megtett km, az év és a leírás mezők kitöltése opcionális ugyan, de a háromból minimum egyet ki kell tölteni, hogy a kitöltöttségi faktor elérje a 80%-os szintet, és ezáltal feltölthető legyen a hirdetés. A kitöltöttségi faktor a lap legalján lesz látható miután legalább egy mezőt elhagytunk. Az aktuális értéket mindig a mezőkből való kikattintás után írja ki.

Ha a checkboxok nem kerülnek kitöltésre, a program ezt úgy érzékeli, hogy a felhasználó szándékosan hagyta üresen. Azaz ha egy jármű tulajdonsága nincs kipipálva a megfelelő checkbox által, akkor azzal a tulajdonsággal nem rendelkezik a jármű.

A márka, modell, város szöveges mezőknek maximum 20 karakter adható meg. Az ár, és a km mezőkbe maximum 9 számjegy írható, minimális értékük 0. Az év minimális értéke 0, a maximális értéke az aktuális év. A leírás maximum 300 karakteres lehet. Ezek megsértésére a program hibaüzenetet fog jelezni.

Amennyiben a kitöltöttségi faktor elérte a 80%-os szintet, és a mezők helyesen lettek kitöltve, a *Hirdetés feladása* gomb segítségével a hirdetés mentésre kerül. (A gomb lenyomásakor a kitöltöttségi faktor újra meg lesz vizsgálva.)

A program jelzi, hogy sikeres a hirdetés feladása.

### Új hirdetés feladása

---

Márka: \*

Model: \*

Város: \*

Ár: \*

Megtett km:

Év:

Kategória: Gépkocsi ▾

Üzemanyag: Benzin ▾

Állapot: Új ▾

Leírás:

☐ Automata váltó  
☐ Alufélni  
☐ Klíma  
☐ Tempomat  
☐ Vonóhorog  
☐ Szervizkönyv


[Hirdetés feladása](#)

19. ábra Új hirdetés feladása

## 2.5.5 Hirdetéseim megtekintése

A felhasználó a saját hirdetéseit meg tudja tekinteni a menüsor Hirdetéseim gombja alatt (20. ábra.) Egy-egy hirdetés megtekintéséhez kattintsunk a *Részletek* gombra.

Amennyiben még nemadtunk fel hirdetést, nem jelenítődik meg találat az oldalon.



Bejelentkezve, mint asdf - Admin  
[Keresés](#)
[Profil szerkesztése](#)
[Új hirdetés](#)
[Hirdetéseim](#)
[Kedvencek](#)
[Régi hirdetések](#)
[Kijelentkezés](#)

---

### Hirdetéseim

---

<a href="#">Részletek &gt;&gt;</a>	Márka	1
	Modell	1
	Ár	1

---

<a href="#">Részletek &gt;&gt;</a>	Márka	1
	Modell	1
	Ár	1

---

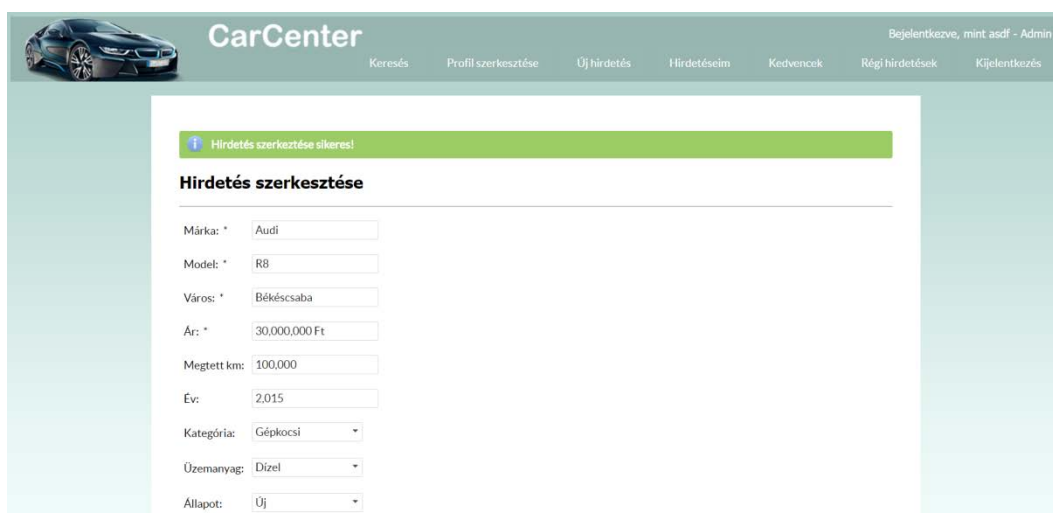
<a href="#">Részletek &gt;&gt;</a>	Márka	Tesla2
	Modell	Tesla
	Ár	43000000

---

20. ábra Hirdetéseim

## 2.5.6 Hirdetése szerkesztése

A felhasználó a saját hirdetését szerkesztheti. Ezt a funkciót a legegyszerűbben a menüsor *Hirdetéseim* menüpont alatt, a megfelelő hirdetés részleteire kattintva érheti el. (*Keresés* útján, és a *Kedvencek* menüpont alatt is el lehet érni.) Ha a felhasználó a saját hirdetését tekinti meg, akkor elérhető kell legyen a *Szerkesztés* gomb, amelyre kattintva elértük a kívánt funkciót. Itt ugyanazok a megszorítások érvényesek, mint az új hirdetés feltöltésénél. A kitöltöttségi faktor is ugyanúgy működik, kivédve ezzel azt, hogy egy helyesen kitöltött hirdetésből szinte üres hirdetést hozhasson létre a felhasználó. A kívánt módosításokat elvégezve, a *Módosítás* gombra kattintva, értesítést kapunk, hogy a szerkesztés sikeres vagy sikertelen volt (21. ábra.)



21. ábra Saját hirdetés szerkesztése

## 2.5.7 Profil szerkesztése

A felhasználó – a felhasználónevét kivéve – megváltoztathatja a regisztráláskor megadott profil adatait, a menüsor *Profil szerkesztése* gombjára kattintva (22. ábra.) Itt is ugyanúgy kötelező az összes mező kitöltése, mint regisztráláskor. A mezők maximum annyi karakteresek lehetnek, amennyi a regisztrációkor engedélyezett volt. Az email cím megfelelő formátumú kell legyen, valamint a telefonszám csak számokat tartalmazhat. A Jelszó és Jelszó újra mezőknek továbbra is meg kell egyezniük, valamint csak számokat és/vagy az angol abc kis/nagy betűit tartalmazhatják. A *Módosít* gombra kattintva menthetjük az adatokat. Értesítést kapunk a szerkesztés sikerességéről vagy sikertelenségéről.

22. ábra Saját profil szerkesztése

## 2.5.8 Kijelentkezés

A felhasználónak lehetősége van a menüsorból elérhető *Kijelentkezés* gombra kattintva kijelentkezni az oldalról. A program a bejelentkezési oldalra viszi át a felhasználót. Egy Session (munkamenet) a bejelentkezéstől a kijelentkezésig tart.

## 2.6 Admin funkciói

### 2.6.1 Általános információk az adminról

Az adminisztrátor (admin) képes minden olyan funkció elérésére, amit egy felhasználó is elérhet. Azonban ehhez a pozícióhoz társul néhány plusz funkció is. A program több admin kezelésére is alkalmas, azonban a minta adatok között egy admin („admin” néven) szerepel.

### 2.6.2 Régi hirdetések

A menüsorban az admin egy plusz gombot lát *Régi hirdetések* néven. Erre kattintva egy táblázat lesz látható az összes hirdetéssel együtt, dátum szerint növekvő sorrendben, mivel ez a funkció főképp arra szolgál, hogy a túl régi hirdetéseket minél könnyebben törölni lehessen. Emellett persze az admin dönthet úgy, hogy töröl bármely hirdetést, amelynek törlését szükségesnek ítéli meg. A táblázat utolsó oszlopa checkboxokból áll. Ha a táblázat fejlécének checkboxára kattint azzal az összes checkbox státuszát egyszerre tudja változtatni. Az egyenkénti és a Windowsban megszokott Shift-es, Ctrl-os kijelölések is

működnék (23. ábra.) A kijelöléshez elég a megfelelő sorra kattintatni, nem kell feltétlenül a checkboxra. Ha kijelöltük az összes törlendő elemet, kattintsunk a *Törlés* gombra. Értesítést nem kapunk a törlés sikerességéről, mivel a táblázat azonnal frissül és egyből láthatjuk, ha sikerült kitörölni a kívánt elemeket.

Márka	Modell	Ár	Dátum	
audi	a7	2000000	2017-03-07 00:00:00.0	<input type="checkbox"/>
bmw	x6	1000	2017-03-09 00:00:00.0	<input checked="" type="checkbox"/>
bmw	x6	1000	2017-03-09 00:00:00.0	<input checked="" type="checkbox"/>
bmw	x6	1000	2017-03-09 00:00:00.0	<input checked="" type="checkbox"/>
marka	model	100001	2017-03-27 13:42:57.0	<input checked="" type="checkbox"/>
1	1	1	2017-04-11 16:59:22.0	<input checked="" type="checkbox"/>
1	1	1	2017-04-11 17:13:07.0	<input checked="" type="checkbox"/>
Tesla2	Tesla	43000000	2017-04-24 10:43:18.0	<input checked="" type="checkbox"/>
01234567890123456789	01234567890123456789	123456789	2017-04-24 19:07:55.0	<input checked="" type="checkbox"/>
01234567890123456789	01234567890123456789	201324354	2017-04-24 19:09:15.0	<input checked="" type="checkbox"/>
01234567890123456789	01234567890123456789	1234567890	2017-04-24 19:13:21.0	<input checked="" type="checkbox"/>
99999999999999999999	99999999999999999999	9999	2017-04-24 19:26:51.0	<input checked="" type="checkbox"/>
99999999999999999999	99999999999999999999	999999999	2017-04-24 19:34:11.0	<input type="checkbox"/>
Audi	R8	30000000	2017-04-26 09:57:47.0	<input type="checkbox"/>
asdf	asdf	323	2017-05-02 10:34:37.0	<input type="checkbox"/>
éé	é	5	2017-05-02 16:42:19.0	<input type="checkbox"/>
0	0	0	2017-05-04 10:51:41.0	<input type="checkbox"/>

Exportálás excelbe Törlés

Készítette: Viczián Lilla | Neptun-kód: AZZHQP

23. ábra Régi hirdetések menüpont

Az *Exportálás excelbe* gombbal exportálhatjuk az oldalon látható teljes táblázatot (márka, modell, ár, dátum oszlopokkal) egy xls kiterjesztésű fájlba.

### 2.6.3 Hirdetés törlése

Az admin nem csak a *Régi hirdetések* gomb által elért oldalon tud törölni hirdetéseket, hanem a *Keresés*, *Hirdetéseim*, *Kedvencek* oldalakról elérhető bármely hirdetés részleteire kattintva. Itt elérhető számára egy *Törlés* gomb, pont ugyanúgy, mint ahogy a felhasználó a saját hirdetését tudja törölni. Hirdetés törlése esetén a hirdetés már a többi nézetből sem lesz elérhető.

Hirdetés megtekintése	
Márka	BMW
Modell	535xd
Város	Kiskunfélegyháza
Ár	19768810 Ft
Megtett km	129528 km
Év	2016
Kategória	gépkocsi
Üzemanyag	gáz
Állapot	új
Automataváltó	van
Alufelni	van
Klíma	van
Tempomat	van
Vonóhorog	van
Szervizkönyv	van
Leírás	Az autó szuper állapotban van.
Hirdetést feltöltötte	bb

[Excelbe exportálás](#)
[Kedvencekhez](#)
[Törlés](#)

Térkép

24. ábra Hirdetés megtekintésnél lévő Törlés gomb

## 2.6.4 Hozzászólások moderálása

Amennyiben a hirdetések alatt a hozzászólásokat olvasva, az admin úgy találja, hogy egy hozzászólás nem megfelelő, a hozzászólás mellett lévő *Moderálás* gombra kattinthat, így egyből látható, hogy a hozzászólás szövege helyére egy „MODERÁLVA” felirat kerül.

Hozzászólások:

MODERÁLVA  
Felhasználó: asdf - 2017-05-06 19:58:47.0 MODERÁLVA

25. ábra Hozzászólás

## 2.7 Párhuzamos használat

A programot egyszerre több felhasználó is képes használni. Ehhez nincs feltétlen szükség egy másik számítógépre, ugyanis ha már a böngészőnkben bejelentkeztünk egy felhasználóval (esetleg adminnal), egy másik böngészőt indítva, vagy a Google Chrome inkognitóját használva, bejelentkezhettünk másik felhasználóval (esetleg adminnal.) Így kipróbálható, hogy minden funkciót használhatnak egyszerre többen is. Amennyiben valamilyen hiba lép fel az oldalon, például egy törölt hirdetés funkcióit próbálja elérni egy másik felhasználó, úgy „A kért oldal nem található.” oldal fog betöltődni, mivel a program érzékeli, hogy a törölt hirdetés már nem elérhető.



## 3 Fejlesztői dokumentáció

### 3.1 A feladat

A szakdolgozatom célja egy olyan webes alkalmazás létrehozása, amellyel eladó autókat lehet kezelni. A programnak rendelkeznie kell egy webes, grafikus felülettel, amelyen keresztül az oldal különböző funkcióit lehet elérni. Az oldal látogatója lehet vendég, bejelentkezett felhasználó (későbbiekben felhasználó), adminisztrátor (későbbiekben admin.)

Vendégként legyen lehetőség keresni az autók között és tetszőlegeset megtekinteni. Az alapértelmezett keresési eredmény az összes hirdetés. Ha a keresésnél olyan feltételeket ad meg a vendég, hogy a keresés helyes, de nincs találat, akkor nem lesz találat megjelenítve. A megtekintésnél az autó adatai mellett, térképes felületen is látható legyen az autó elhelyezkedése.

Biztosítson felületet regisztrációhoz, bejelentkezéshez, profil adatok későbbi szerkesztéséhez. Jelezze a hibát helytelen adatok megadása esetén.

Felhasználóknak lehessen új hirdetést feladniuk, később szerkesztteni, törölni ezeket. Hiba esetén, ennek kiírása a képernyőre.

A felhasználó tudja egyszerre megtekinteni az összes általa feladott hirdetést.

A felhasználó tudja xls kiterjesztésű fájlba exportálni tetszőleges hirdetés adatait. Legyen lehetősége megtekinteni a hirdetéshez tartozó hozzászólásokat, új hozzászólást adni valamennyi hirdetéshez.

Amennyiben egy hirdetés megtetszik a felhasználónak, a kedvencekhez adhatja, illetve később törölheti onnan.

A felhasználók felügyelésére szolgál az adminként való bejelentkezés. A felhasználói funkciókon kívül biztosítson az admin számára egy felületet, ahol látható az összes hirdetés időrendben. Ez a táblázat exportálható legyen xls kiterjesztésben. Az admin tetszőleges hirdetést törölhet ezen a felületen is, valamint a hirdetés megtekintésénél is van lehetősége ezt megtenni.

Továbbá az admin képes legyen a hozzászólások moderálására is. A moderálás során a hozzászólás szövege eltűnik, és helyette fel kell tüntetni, hogy a hozzászólás moderálásra került.

Bármely bejelentkezett felhasználó ki is tudjon jelentkezni, ezzel újra vendég státuszba kerül.

Ugyanarról a hálózatról lehessen egyszerre többen is használni az alkalmazást, amennyiben ez nincs letiltva rendszergazda által, sem a hálózati beállításokban, sem a tűzfalon. Ha a bejelentkezett felhasználó a Google Chrome böngészőben új lapot nyit, ott is legyen beléptetve azzal a felhasználóval, amivel eredetileg bejelentkezett. Azonban ha a böngésző inkognitóablakban nyitja meg újra az oldalt, ott már lehetősége legyen bejelentkezni másik felhasználóval is. Ha másik böngészőt nyit, azon keresztül is újra bejelentkezhet. Ha ugyanarról a hálózatról, de másik gépről szeretné valaki elérni az oldalt, annak szintén újra lehetősége legyen bejelentkezni. Ha valamilyen hiba adódna, például az admin törli a felhasználó hirdetését, amelyet a felhasználó éppen szerkeszt, és mire a *Módosít* gombra kattint a felhasználó, addigra már nem is létezik a hirdetése, akkor a program értesítse a felhasználót, hogy a kért oldal nem található.



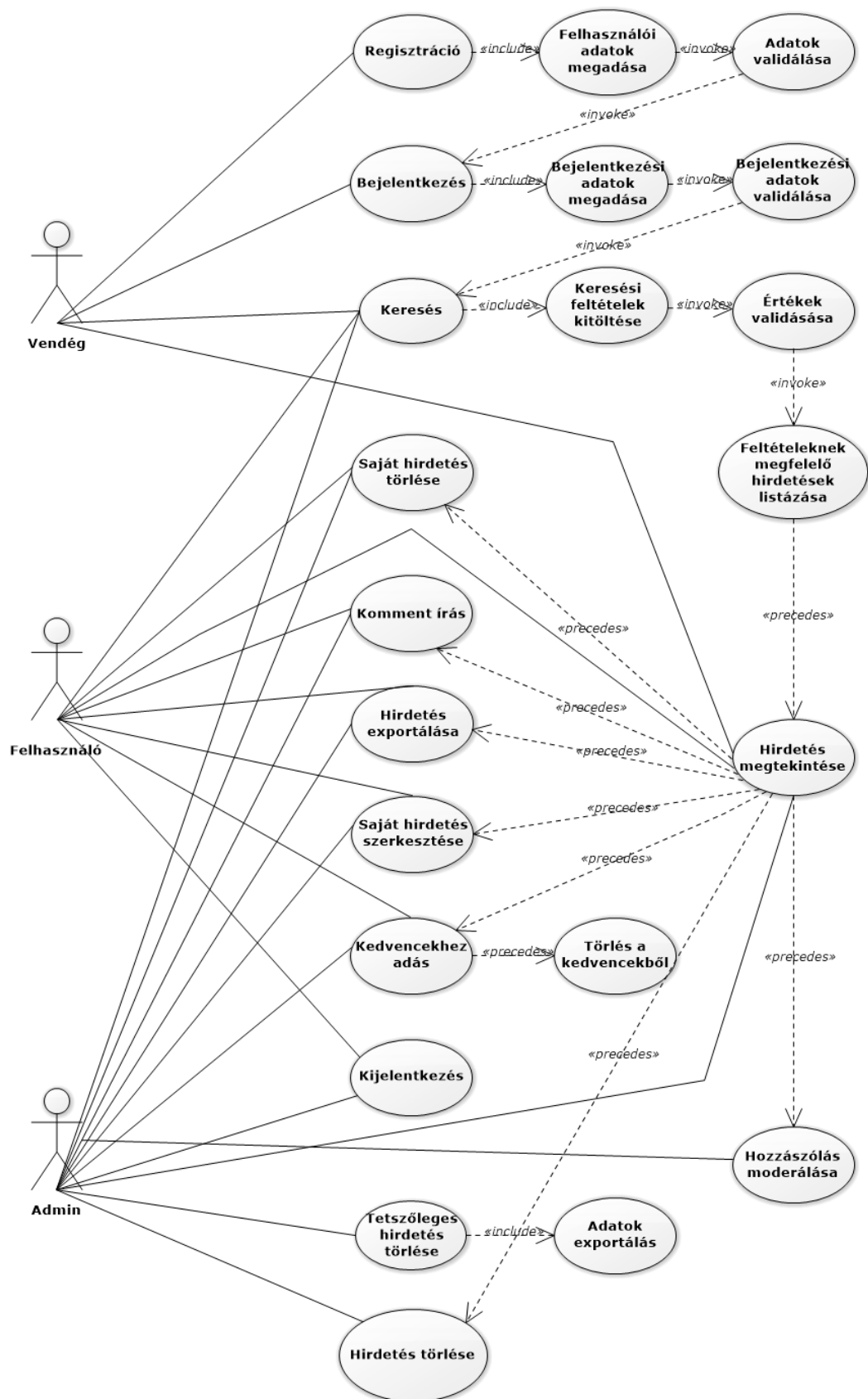
26. ábra „A kért oldal nem található.” hiba oldal

## 3.2 Megoldási terv

Az alkalmazásban, grafikus felületen, a weboldalak tetején egy menüsorból, közvetlenül vagy közvetetten érhető el az összes funkció az oldalt használók számára.

### 3.2.1 Use case diagram

A lehetséges használati eseteket a következő ábrán találjuk:

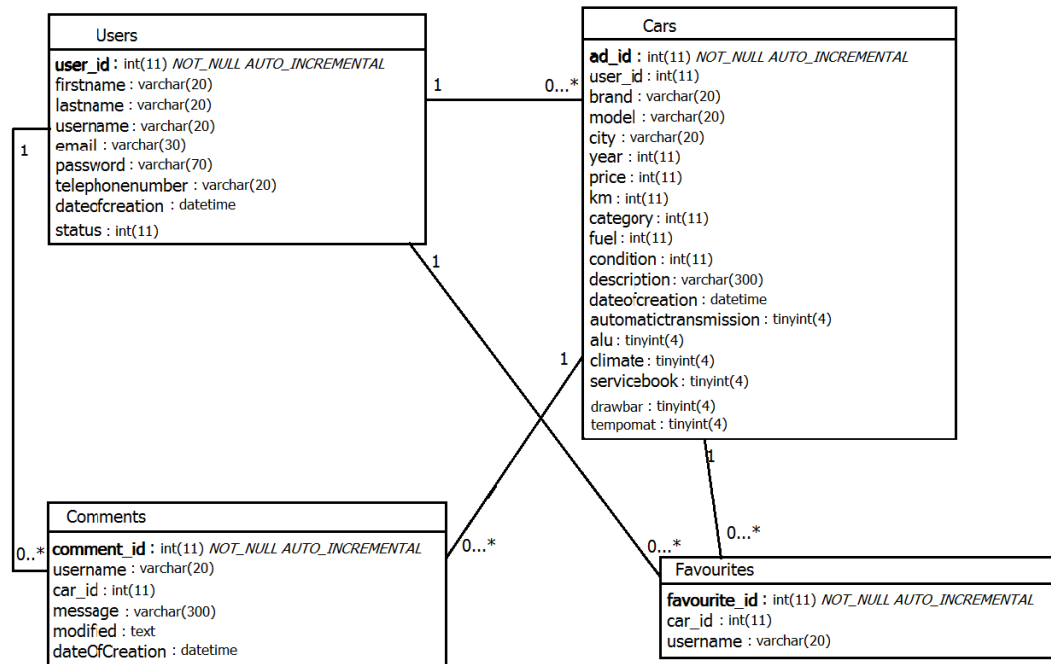


27. ábra Use case diagram

### 3.2.2 Adatbázis

A feladathoz szükséges felhasználók adatai, hirdetések adatai, kedvencek, hozzászólások MySQL adatbázisban lesznek tárolva.

#### Adatbázis kapcsolat:



28. ábra Adatbázis felépítése

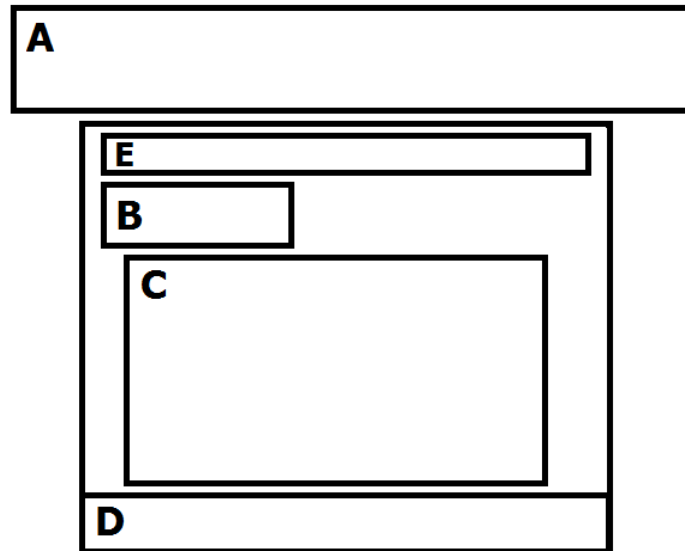
A **Users** táblában a felhasználók kerülnek tárolásra, a következő adattagokkal: felhasználói azonosító (`user_id`), keresztnév (`firstname`), vezetéknév (`lastname`), felhasználónév (`username`), emailcím (`email`), jelszó (`password`), telefonszám (`telephonenumber`), létrehozás dátuma (`dateofcreation`), státusz (`status`.)

A **Cars** táblában a hirdetések kerülnek tárolásra, a következő adattagokkal: hirdetés azonosítója (`ad_id`), felhasználói azonosító (`user_id`), márka (`brand`), modell (`model`), város (`city`), év (`year`), ár (`price`), km (`km`), kategória (`category`), üzemanyag (`fuel`), állapot (`condition`), leírás (`description`), létrehozás dátuma (`dateofcreation`), automataváltó (`automatictransmission`), alufelni (`alu`), klíma (`climate`), szervizkönyv (`servicebook`), vonóhorog (`drawbar`), tempomat (`tempomat`.)

A **Comments** táblában a hozzászólások kerülnek tárolásra, a következő adattagokkal: hozzászólás azonosítója (`comment_id`), hozzászólás készítőjének felhasználóneve (`username`), a hirdetés, amelyhez a hozzászólás tartozik (`car_id`), hozzászólás szövege (`message`), moderált-e a hozzászólás (`modified`), létrehozás dátuma (`dateOfCreation`.)

A **Favourites** táblában a kedvencek kerülnek tárolásra, a következő adattagokkal: a kedvenc azonosítója (favourite\_id), az hirdetés azonosítója (car\_id), és a felhasználónév (username.)

### 3.2.3 A kinézeti terv



29. ábra Az oldal elrendezésének terve

- A: menüsor
- B: aloldal neve/fő funkciója
- C: aloldal tartalma
- D: lábrészben a fejlesztő neve és Neptun-kódja lesz feltüntetve
- E: esetleges hibaüzenetek, információértékű értesítések helye

### 3.2.4 Megvalósítandó menüpontok

A vendég számára elérhetőek:

- Keresés
  - o Itt lesz lehetőség keresés előtt megtekinteni az összes hirdetést,
  - o Az autó adatai alapján keresni, ár vagy létrehozás dátuma szerint rendezni a keresés eredményét.
  - o A keresés eredményéből kiválasztani egy hirdetést, amelyet megtekintünk.
    - A megtekintett hirdetésnél láthatóak az autó adatai, és egy térkép, amely az autó helyét jelöli.

- Bejelentkezés
  - o Ez a menüpont szolgál arra, hogy bejelentkezzünk a már regisztrált felhasználónk felhasználónevével és jelszavával.
- Regisztráció
  - o Új felhasználói fiókot tudunk létrehozni a megadott mezők helyes kitöltésével.

#### **A felhasználó számára elérhetőek:**

- Keresés
  - o A keresés menete megegyezik a vendég számára elérhető keresés menetével. Az innen elérhető hirdetés megtekintés azonban kiegészül néhány funkcióval:
    - A felhasználó a saját hirdetését itt szerkesztheti,
    - törölheti.
    - Bármely hirdetést hozzáadhat a Kedvencekhez,
    - vagy törölhet a Kedvencekből.
    - Bármely hirdetést exportálhat,
    - olvashatja a megírt hozzászólásokat,
    - hozzászólhat.
- Profil szerkesztése
  - o A felhasználónak itt lehetősége van szerkeszteni a felhasználói adatait, kivételt képez a felhasználónév.
- Új hirdetés feladása
  - o A felhasználó ennél a menüpontnál új hirdetést tud feladni. Ehhez fontos, hogy helyesen kitöltsön a kitöltöttségi faktornak is megfelelő számú mezőt.
- Hirdetésem
  - o A felhasználó ezalatt a menüpont alatt tekintheti meg a már feltöltött hirdetéseit, amennyiben van feltöltött hirdetése. Ha nincs, akkor nem listázódik ki egy hirdetés sem.
  - o Abban az esetben, ha van feltöltött hirdetése, meg is tekintheti azt.
- Kedvencek

- A felhasználó itt tekintheti meg a már a kedvencekhez adott hirdetéseit, amennyiben van ilyen. Ha nincs, akkor nem listázódik ki egy hirdetés sem.
- Abban az esetben, ha van kedvence, meg is tekintheti őket, mint hirdetéseket.
- Kijelentkezés
  - A bejelentkezett felhasználó kijelentkezhet az oldalról ezzel a menüponttal.

### **Az admin számára elérhetőek:**

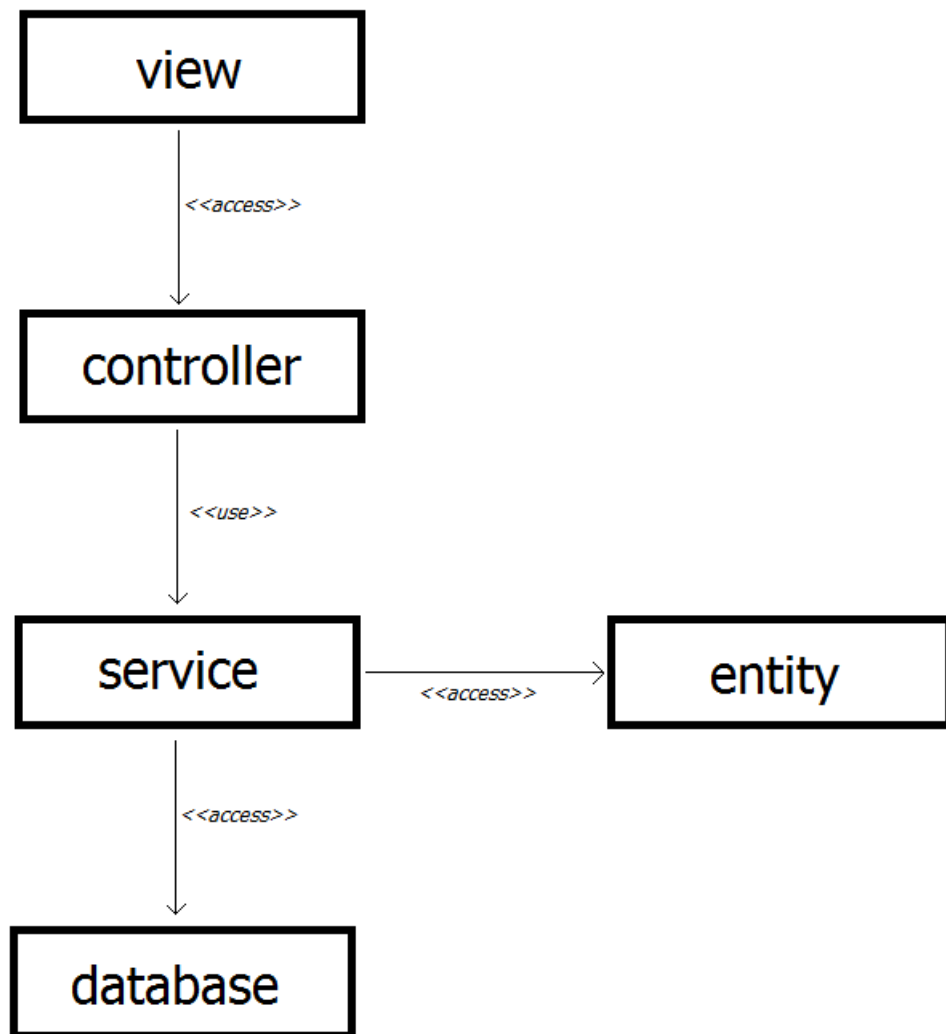
A felhasználó számára elérhető funkciók az admin számára is elérhetőek, de kiegészülnek néhány menüpontban:

- Keresés
  - A keresés menüpont alatt elérhető hirdetést megtekintve, moderálhatja a már megírt hozzászólásokat,
  - illetve törölheti a teljes hirdetést.

(A hirdetés megtekintése adott esetben elérhető lehet a Hirdetésem, és a Kedvencek menüpont alatt is. Ezekből a menüpontokból is ugyanúgy működnek a funkciók.)
- Régi hirdetések
  - Ebben a menüpontban táblázatos formában láthatja az admin, az összes feladott hirdetést, dátum szerint rendezve. Ezek közül bármennyit törölhet, akár többet is egyszerre, kijelölés segítségével.
  - Ezenkívül xls fájlba exportálhatja az összes hirdetést.

## **3.3 Az MVC elve és megvalósítása**

A program megvalósítása során törekszem az MVC architektúra kialakítására. [6] Azaz legyen szerkezetileg model, view, és controller részre bontható a program kódja. A modelnek felelnek majd meg az entity és a service osztályok. A service-ek kommunikálnak majd az adatbázissal. A view a weboldal megjelenítéséért felelős fájlokat fogja tartalmazni. A controller, nem más, mint a vezérlő réteg, a view-t és a modelt kapcsolja össze.



30. ábra MVC architektúra terve

### 3.4 Megvalósítás

A weboldalak megnyitásához Google Chrome-ot használtam.

A Java kód fordításához, futtatásához mindenképp szükség van az 1.8-as verziószámú JDK-ra (Java Development Kit.)

A program elkészítéséhez JetBrains IntelliJ IDEA (röviden IDEA) fejlesztőkörnyezetet használtam.

Egy Maven projektet hoztam létre, és felhasználtam a Java Spring keretrendszer szolgáltatásait is.

A program futtatásához Tomcat webszervert választottam.



Az adatok tárolása MySQL adatbázisban történik, a MySQL Editorral, a MySQL saját adatbázis-kezelőjével könnyen áttekinthettem a fejlesztés során az adatokat.

A Hibernate programkönyvtárat használtam az adatbázis és a Java-kód közötti kommunikáció megkönnyítésére, mivel a Hibernate automatikus konverziót biztosít a java osztályok (entitások) és a relációs adatbázis között.

A program kinézetének elkészítéséhez szükségem volt a PrimeUI keretrendszerre. Ez egy könyvtárcsomag a Java Server Faces-hez (JSF-hez.) Ez a keretrendszer teszi lehetővé, hogy xhtml oldalakat kössünk össze java-s osztályokkal.

## 3.5 Fejlesztői környezet telepítése

### 3.5.1 Google Chrome

A böngésző letöltést és telepítést igényel.<sup>6</sup>

### 3.5.2 JDK

Az 1.8-as verziószámú JDK-t (Java Development Kit) töltsük le a hivatalos weboldaltól.<sup>7</sup> Az exe kiterjesztésű alkalmazást futtassuk, így telepítve a programot. Telepítés után állítsuk be a megfelelő környezeti változókat, például ha a C:\Program Files\Java helyre telepítettük a JDK-t, úgy a JAVA\_HOME környezeti változó értékének meg kell adni a C:\Program Files\Java\jdk1.8.0\_121 értéket, a JDK telepítési helyét.

Ugyanezt az elérési útvonalat állítsuk be a Path rendszerváltozóhoz is.

(Segítség a környezeti változók beállításához:

- az egér jobb gombjával kattintsunk a Startmenüre, válasszuk a Rendszer menüpontot,
- ezen belül a Speciális rendszerbeállításokat, majd a Környezeti változók... gombot.
- Ezután a megfelelő környezeti változóra állva, a Szerkesztés gombbal tudjuk szerkeszteni az elérési útvonalakat.)

---

<sup>6</sup><https://www.google.com/intl/hu/chrome/browser/desktop/>

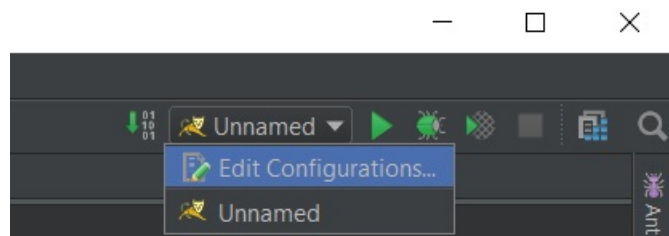
<sup>7</sup><http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

### 3.5.3 JetBrains IntelliJ IDEA

A JetBrains hivatalos oldaláról az Ultimate típusú IDEA-t válasszuk. Ez is ingyenesen letölthető<sup>8</sup>, és az ELTE hallgatói számára ingyenesen használható. A letöltött állomány könnyedén telepíthető.

### 3.5.4 Tomcat telepítése

A Tomcat hivatalos oldaláról elérhető a Tomcat szerver.<sup>9</sup> A C:\apache-tomcat-9.0.0.M17\apache-tomcat-9.0.0.M17\bin mappában a startup.bat fájlal manuálisan is indíthatjuk a szervert, de ekkor használat után ne felejtsük el leállítani azt a shutdown.bat fájlal. Azonban ha az IDEA-t megfelelően beállítjuk, akkor kényelmesebben is indíthatjuk az alkalmazást:

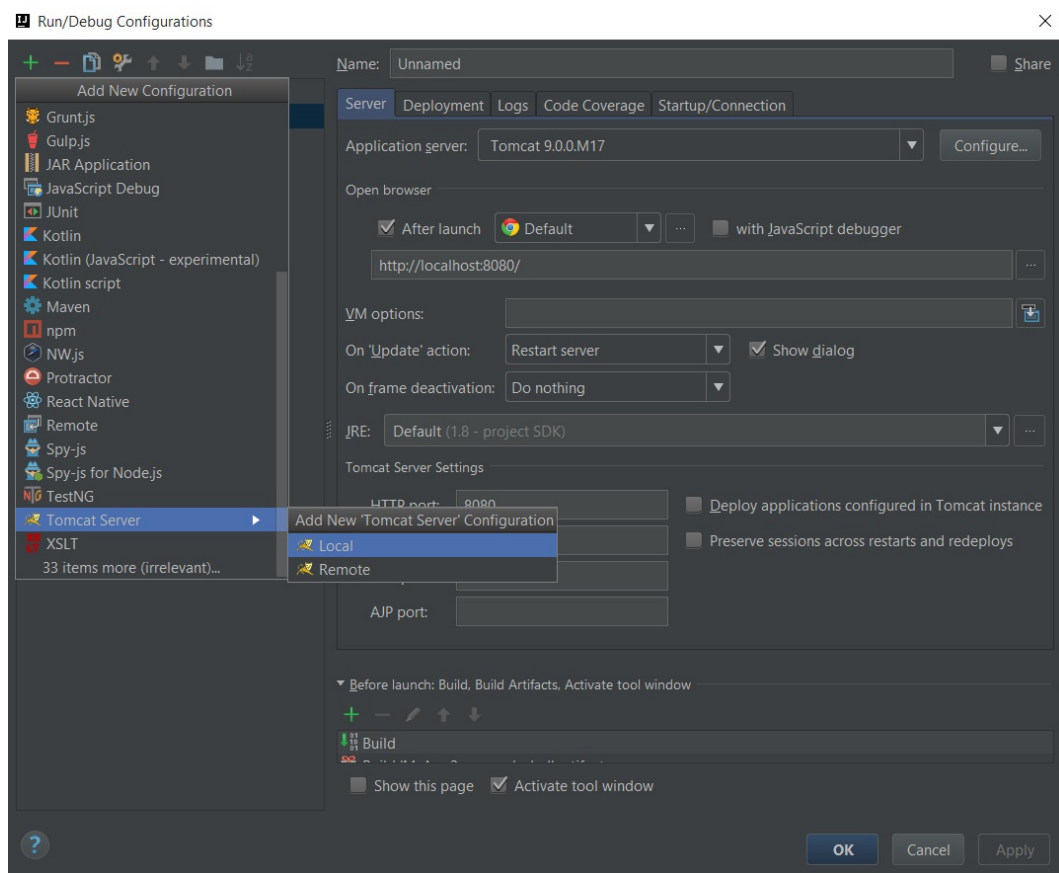


**31. ábra** Kattintsunk az „Edit Configurations...” feliratra

A megjelenő ablak bal felső sarkában található, zöld pluszjelre kattintva, válasszuk ki a következőket:

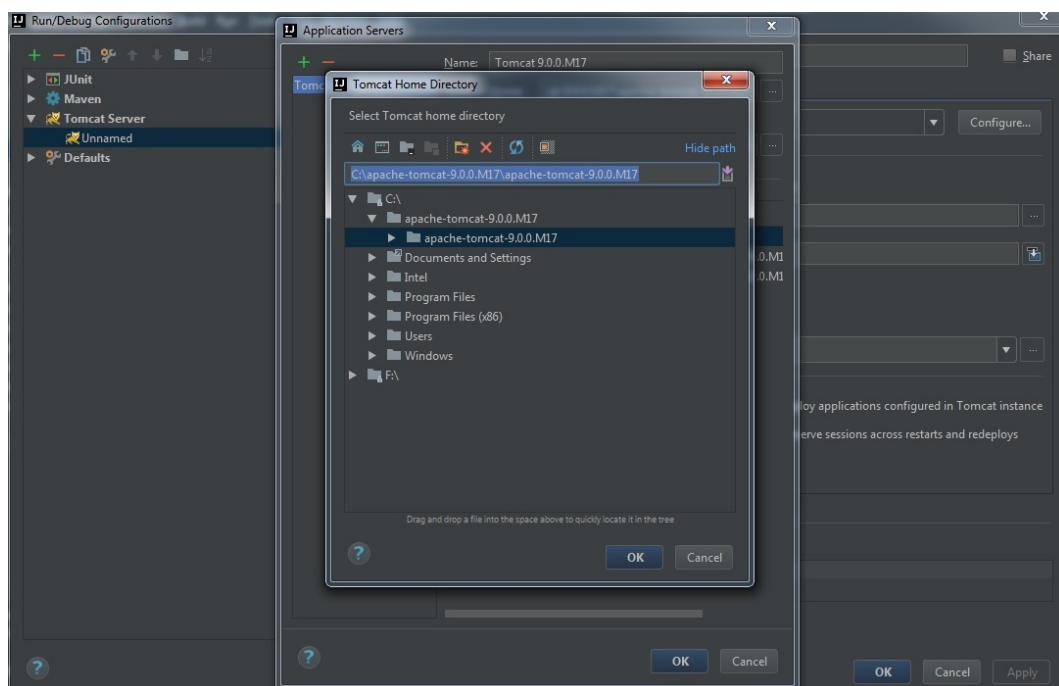
<sup>8</sup><https://www.jetbrains.com/idea/download/#section=windows>

<sup>9</sup><http://tomcat.apache.org/download-90.cgi>



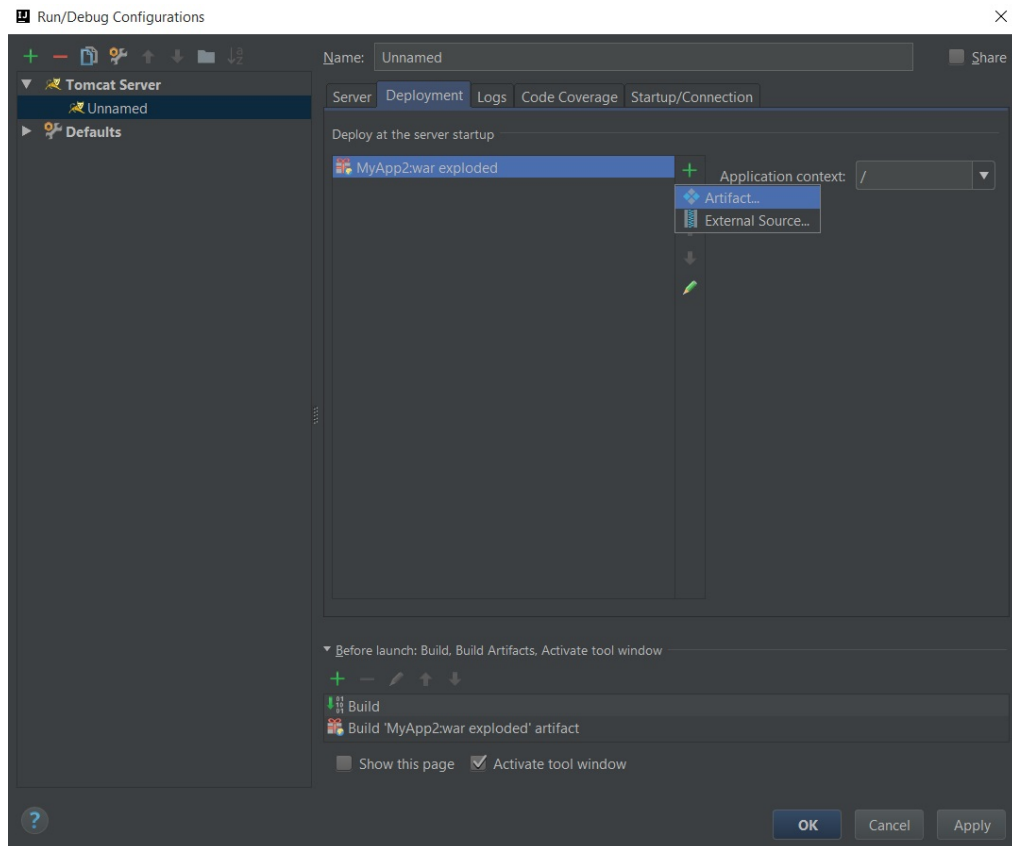
### 32. ábraServer beállításai

Figyeljünk a JDK és a Tomcat helyes elérési útvonalára.



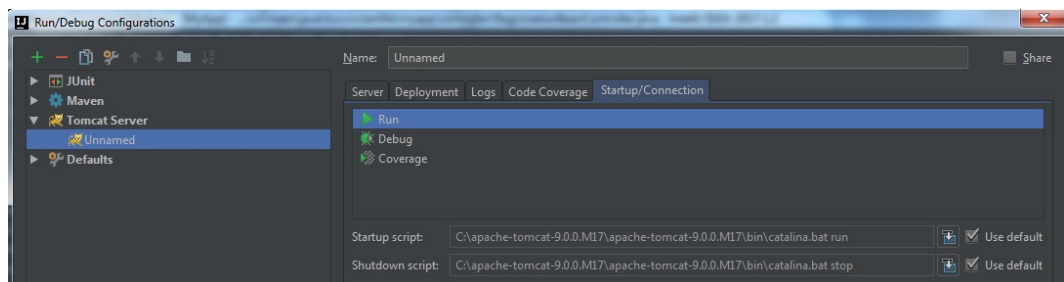
### 33. ábra Tomcat elérési címe

A Tomcat további beállításai:



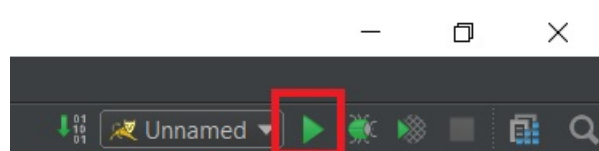
34. ábra Deployment beállításai

A Startup/Connection opciónál állíthatjuk be, hogy az IDEA hogyan indítsa a Tomcat szerveret:



35. ábra Startup/Connection beállításai

A megfelelő beállítások után a következő ikonra kattintva futtathatjuk a szerveret, és indíthatjuk a programot:

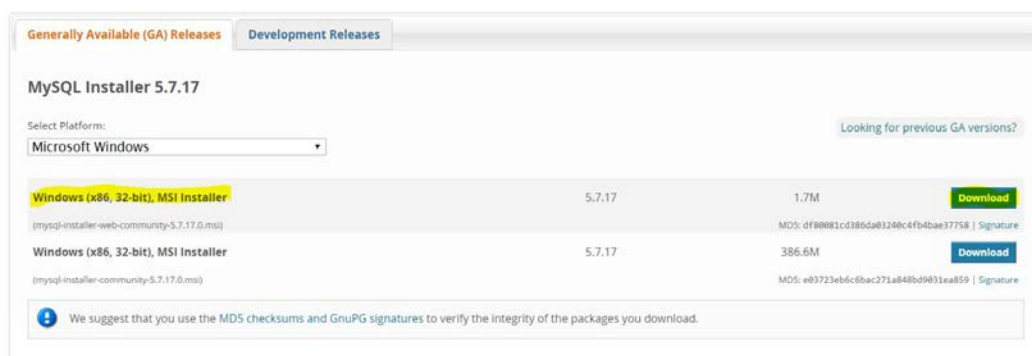


36. ábra Server indítása

(Az indításhoz szükség van a JAVA\_HOME környezeti változó megfelelő beállítására, amelyet a (3.4.2) sorszámú fejezetben találunk.)

### 3.5.5 MySQL telepítése

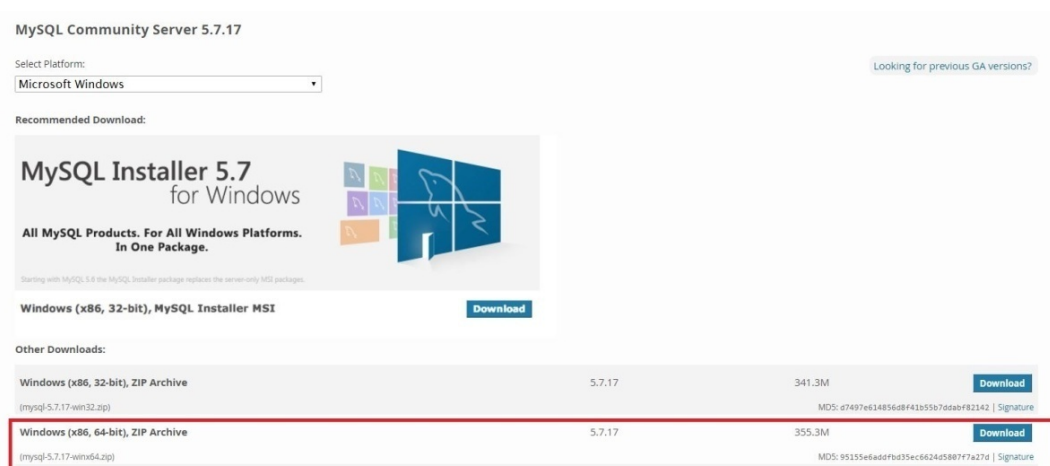
Az adatbázis-kezelő letöltése:<sup>10</sup>



37. ábra Töltsük le a kijelölt elemet!

A 32 bites MySQL Installer 5.7.17 (vagy ennél újabb verzió) megfelelő a 32 bites és 64 bites Windowsra is.

Amennyiben automatikusan nem töltődött le, töltsük le a MySQL szerveret is.<sup>11</sup>

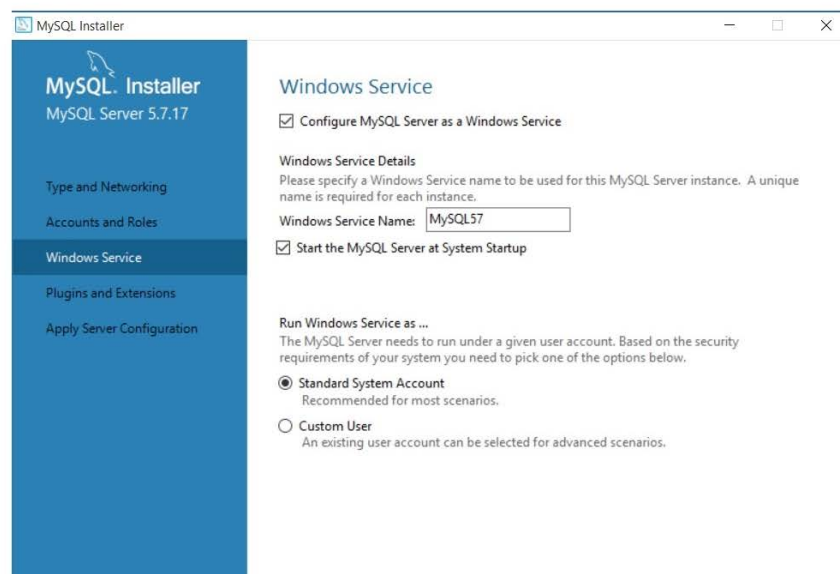
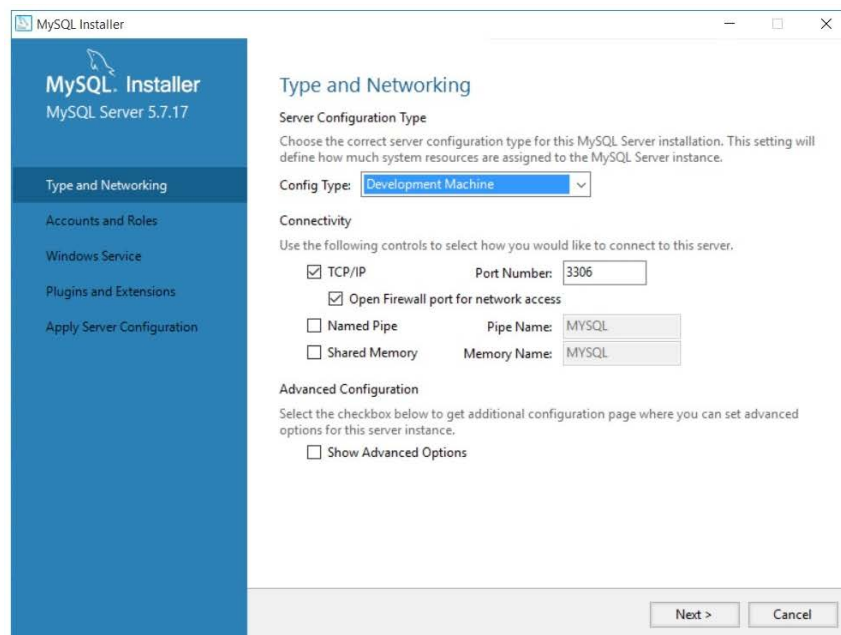


38. ábra Töltsük le a gépünknek megfelelő szerveret

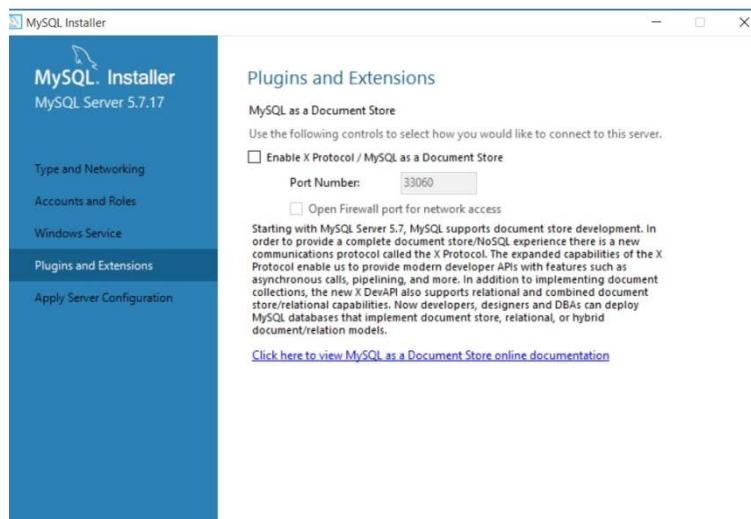
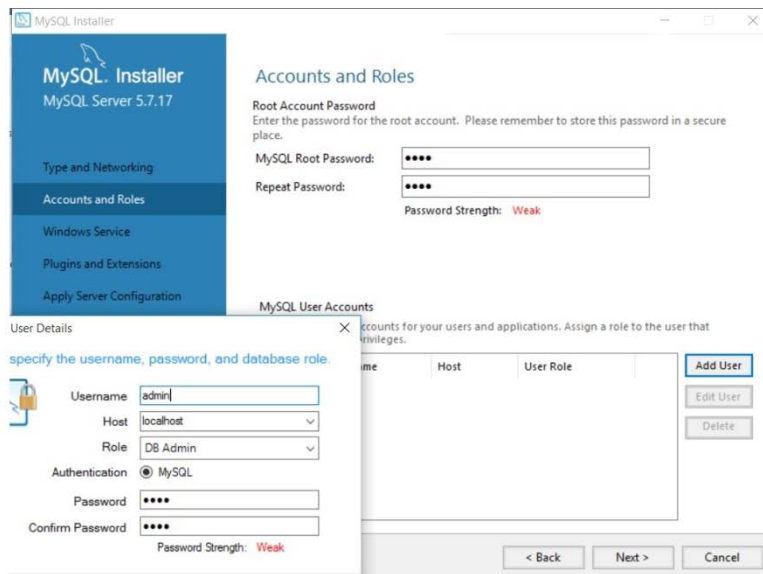
Telepítésnél mindent hagyhatunk az alapértelmezett beállításon, de figyeljünk, mert hozzá kell adnunk a szerveret, és egy felhasználót (39. ábra, 40. ábra.) (Ez legyen: „admin” felhasználónevű és „asdf” jelszavú.)

<sup>10</sup><https://dev.mysql.com/downloads/installer/>

<sup>11</sup><https://dev.mysql.com/downloads/mysql/>

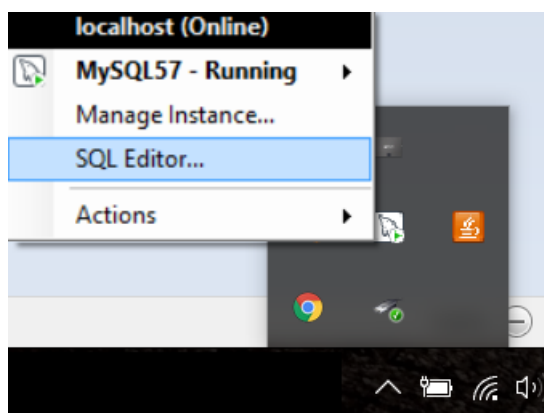


**39. ábra A MySQL telepítésének kérdéses részei**



40. ábra Adatbázis további kérdéses részei

Nyissuk meg a telepített SQL Editort:



41. ábra A tálcáról elérhető a felület





- SessionScopeBean.java
- ShowAdBeanController.java
- ShowMyAdsBeanController.java
- ShowMyFavouritesBeanController.java
- ShowOldAdsBeanController.java
- **entity**
  - Ad.java
  - Comment.java
  - Favourite.java
  - User.java
- **service**
  - AdService.java
  - CommentService.java
  - CustomUserDetailsService.java
  - UserService.java
- **resources\spring**
  - application.properties
- **webapp**
  - admin
    - showOldAds.xhtml
  - **resources**
    - **css**
      - layout.css
    - **img**
      - back3.png
      - cari8.png
    - **js**
      - filledRate.js
  - **user**
    - editAd.xhtml
    - editProfile.xhtml
    - newAd.xhtml
    - showMyAds.xhtml

- showMyFavourites.xhtml
- **WEB-INF**
  - **lib**
    - poi-3.8-20120326.jar
  - faces-config.xml
  - web.xml
- error.xhtml
- index.xhtml
- layout.xhtml
- login.xhtml
- registration.xhtml
- showAd.xhtml
- **test/java/hu/viczianlilla/myapp**
  - **configuration**
    - JpaTestConfig.java
    - MyCustomHsqlDialect.java
  - **service**
    - AdServiceTest.java
    - CommentServiceTest.java
    - UserServiceTest.java
- pom.xml

## 3.7 Fájlok leírása

A projekt létrehozásakor automatikusan generált **pom.xml** fájl információkat tartalmaz a projektről és konfigurációs adatokat a Maven számára (Project Object Model.) Ebben a fájlban kell megadni, hogy milyen keretrendszereket, programkönyvtárakat, függőségeket használjon a program.

- PrimeFaces-hez,
- szerverhez,
- Hibernate-hez,
- MySQL-hez,
- Java Spring keretrendszerhez,

- Java Spring Security-hez,
- Unit tesztekhez,
- Java szintű validációhoz szükséges dependenciák kerültek bele a fájlba az alkalmazás működése érdekében.

A **MyApp\src\main\java\hu\viczianlilla\myapp\configuration** mappában a különböző konfigurációs fájlok találhatóak meg. Az **AppContextConfiguration.java** fájlban került létrehozásra a Java Spring által használt Session Factory, amely az adatbázisbeli tranzakciók nyitásáért/zárásáért felel. A Session Factory segítségével működik a Hibernate. Valamint a Hibernate is beállításra kerül a `resources/spring/application.properties` fájl alapján.

A **LoginErrorPhaseListener.java** weboldalra való hibás belépés jelzéséért felelős osztály. Az oldalon történő minden egyes kérés előtt lefut ennek az osztálynak a *beforePhase* függvénye, és ellenőrzi, hogy volt-e hibás belépés. Abban az esetben, ha helytelenül próbált meg a vendég belépni az oldalra, kiváltódik egy példánya a `BadCredentialsException` kivételnek, és a felhasználóval tudatjuk, hogy rossz adatokat adott meg:



Nem megfelelő felhasználónév vagy jelszó.

Bejelentkezés

#### 44. ábra Helytelen bejelentkezés

A[3]-ban található oldal utasításai alapján készítettem el ezt a funkciót.

A **SecurityConfiguration.java** fájlban került beállításra, hogy pontosan milyen szerepkörű (felhasználó, admin, egyik sem) látogató mely funkciókat érheti el az oldalon. Közvetlenül a webapp mappán belüli `/*.*` fájlok mindenki számára látogathatóak. A webapp mappán belüli `/user/*.*` fájlok csak a felhasználó és az admin számára, míg az `/admin/*.*` fájlok csak az admin számára érhetőek el. Itt került meghatározásra, hogy honnan történik a bejelentkezés, és hogy a program hova navigáljon a kijelentkezés után. Amennyiben a látogató olyan oldalt próbál elérni, amihez nincs hozzáférése, esetleg nem létezik, úgy az oldal visszairányítja a bejelentkezési oldalra.[4]

A **MyApp\src\main\resources\spring\application.properties** fájl tartalmazza az adatbázis eléréséhez szükséges információkat, valamint a Hibernate beállításához tartozó értékeket.

### 3.7.1 MVC architektúra

A mappaszerkezet felépítésénél törekedtem az MVC architektúra betartására. [6] Azaz szerkezetileg model, view, és controller részre bontható a program kódja. A modelnek felelnek meg az entity és a service mappában lévő fájlok. A view-nak a webapp mappában, és almappjaiban lévő fájlok. A controllernek pedig megfelelnek a controller csomagbanlévő java osztályok.

A program felületén, a felhasználotól adatot kérünk némely funkció esetén. A kitöltött mezők helyessége érdekében a view, model, és controller rétegekben is célom volt lekezelni az esetlegesen fellépő hibákat.

A view-t és a modelt összekapcsoló réteg a controller (vezérlő).

#### 3.7.1.1 Controller

Az MVC-beli controllernek megfelelnek a **MyApp\src\main\java\hu\viczianilla\myapp\controller** mappában lévő java kiterjesztésű fájlok. A controllerek feladata a felhasználói felületről kapott adatok feldolgozása, és helyes adatok esetén továbbítása az MVC-beli modelnek.

Az **EditProfileBeanController.java** osztály, a felhasználó adatainak módosításáért felelős. A felhasználói felületről bekért adatokat megkapja ez az osztály és amennyiben nem történt hiba, továbbküldi – a jelszót hash-elve – a service/UserService.java osztálynak, azaz az MVC-beli modelnek, és visszaküldi a felület számára a sikerességről szóló üzenetet. Amennyiben a felhasználó helytelen adatokat próbált menteni, úgy kiíródik a sikertelenségről szóló üzenet, és a modelnek nem kerülnek továbbításra a helytelen adatok.

A **GeocodeView.java** a PrimeFaces hivatalos oldaláról<sup>12</sup> letöltött fájl, amely szükséges a Térkép működéséhez a hirdetések megtekintésénél.

Az **IndexBeanController.java** betölti a service/AdService.java osztálytól, azaz a modeltől kapott hirdetéseket. Amennyiben kivétel váltódott ki, a hibaüzenet

---

<sup>12</sup><https://www.primefaces.org/showcase/ui/data/gmap/geocode.xhtml>

továbbítva lesz a felhasználó számára. Az oldal látogatója, miután keresést indít, a *search()* függvényen keresztül a keresési szűrők, és a rendezés átadásra kerül a model számára. Természetesen csak akkor, ha helyes értékek kerültek a mezőkbe. Ellenkező esetben értesítést kap a látogató a sikertelen keresésről.

A **NewAdBeanController.java** az új hirdetés feladásáért felelős controller. Az *addNewAd()* függvényében a view-ból kapott – checkboxokat nem tartalmazó – paramétereket megvizsgálja, hogy legalább 8 darab van-e belőlük. Ha kevesebb, akkor a felhasználó értesül róla, hogy kevés mezőt töltött ki. Ezután a checkboxokat, dátumot, és felhasználót beállítja és tárolja a függvény, majd megpróbálja átadni mindezt a modelnek. Sikeres (46. ábra) és sikertelen (45. ábra) hirdetés feladásról is értesítést kap a felhasználó.



45. ábra Nem megfelelő kitöltöttségi faktorral rendelkező hirdetés



46. ábra Sikeres hirdetés feladás

A **RegistrationBeanController.java** a regisztrációért felelős controller. Az *addNewUser()* a view-tól kapott adatokkal feltöltött felhasználót, megfelelő dátummal ellátva, megpróbálja továbbadni a model számára. Ha olyan felhasználónévvel próbál regisztrálni, amely már szerepel a rendszerben, akkor egy *PersistenceException* váltódik ki, a felhasználó értesül róla, hogy foglalt ez a felhasználónév (48. ábra.) A sikeres vagy sikertelen (47. ábra) regisztrációról értesítést kap a látogató. Sikeres regisztráció esetén a látogatót a bejelentkezési felületre navigálja az oldal (49. ábra.)



47. ábra Sikertelen regisztráció

 Felhasználónév foglalt!

## Regisztráció

---

### 48. ábra Sikertelen regisztráció

 Sikeres regisztráció!

## Bejelentkezés

---

### 49. ábra Sikertelen regisztráció

A **SessionScopeBean.java** elsősorban azért lett létrehozva, mert a Java Spring Security-nek van egy saját User entitása, ami nem azonos az alkalmazás saját entity/User entitásával, és emiatt biztosítanunk kell egy mappinget (konverziót) a két User fogalom között. Az autentikáció során a Springes User kerül mentésre, ennek csak a felhasználónevét, és a jelszavát ismerjük. A SessionScopeBean osztályban létrehoztam egy *getLoggedInUser()* nevű függvényt, amely, ha a jelenlegi felhasználó nem null, azaz ténylegesen be van jelentkezve egy felhasználó, akkor a függvény visszatérésiértékeként visszaadja az aktuális felhasználót az alkalmazás saját entitásosztálya formájában. Így nem csak a felhasználónevét és a jelszavát tudjuk elérni a bejelentkezett felhasználónknak, hanem bármely más adattagját is. A *getLoggedInUsername()* a Java Spring Userének, azaz az aktuálisan bejelentkezett felhasználónak a felhasználónevével tér vissza. Amennyiben senki nincs bejelentkezve, úgy egy „anonymousUser” szöveggel tér vissza a függvény. A *currentUserIsAdmin()* függvény visszatérési értéke igaz, ha a bejelentkezett felhasználó admin, egyébként hamis. Az *addErrorMessage(String message)* segítségével lesz tájékoztatva a felhasználó az oldalakon történt hibákról. Az *addInfoMessage(String message)* függvény segítségével pedig informálódhat az oldalon történt műveletek sikerességéről. A *getStatus()* szövegesen visszaadja aktuális felhasználónk státuszát, vagy üres Stringet, ha nincs felhasználónk. A *getCurrentYear()* pedig visszaadja az aktuális évszámot.

A **ShowAdBeanController.java** fájl a hirdetés megtekintését lehetővé tevő controller. Az osztály konstruktora után automatikusan lefutó *init()* nevű inicializáló függvény nagyobb terjedelmű, mint az eddigi fájlokban lévők, így érdemes kitérni rá. Ebben a függvényben kerül beállításra az oldal elérési

útvonala, azaz a linkben szereplő id-k itt lesznek megadva az oldalnak. Amennyiben a felhasználó olyan id-val rendelkező hirdetést próbál megjeleníteni, ami valójában nem létezik, úgy át lesz irányítva az `error.xhtml` oldalra. Egy `map`-et megtölt a kategória nevekkal, és az aktuális hirdetés adataival, ez szükséges a `view` számára, az adatok megjelenítéséhez. A `checkboxesChecked()` nevű függvény visszatér egy `String` tömbbel, amelyekben szerepelnek a felhasználó által bepipált checkboxok nevei. A `modifyAd()` a hirdetés módosításáért felelő függvény. Megfelelő kitöltöttségi faktor hiányában, nem végzi el a módosítást, csak közli a felhasználóval, hogy kicsi a kitöltöttségi faktor. Azonban ha elég mező ki van töltve, akkor beállítja a megfelelő adatokat, és továbbadja őket a `model`-nek. Akár sikeres volt a mentés, akár nem, értesülni fog róla a felhasználó. A `delete()` függvény a hirdetés törlését végzi. Amennyiben sikeres a törlés, úgy a saját hirdetéseinkre viszi a látogatót a függvény, ellenkező esetben kiírja a hibaüzenetet. Az `addToMyFavourites()` függvény az aktuális hirdetés kedvencekhez adására szolgál. A `deleteFromMyFavourites()` függvény az aktuális hirdetést törli a kedvencekből. Ennél a két függvéynél is van hibakezelés. Az `isShownFavouriteButton()` függvény dönti el, hogy az aktuális hirdetést, az aktuális felhasználó már a kedvencekhez adta-e. Az `addNewComment()` függvény a hozzászólás írását teszi lehetővé. A megfelelő adatokat kitöltve továbbadja a `model`nek az adatokat, és meghívja az `init()` függvényt újra, hogy egyből látszódjon a felhasználó számára a megírt hozzászólás. Ha az adatok átadása közben hiba lépett fel, akkor értesítést kap a felhasználó. A `setModification()` képes moderálni a hozzászólást, ekkor a hozzászólás korábbi tartalma felülíródik a „MODERÁLVA” konstanssal. Hiba esetén a felhasználó értesül annak bekövetkezéséről.

A **ShowMyAdsBeanController.java** osztály a `model`ből betölti az összes hirdetést a bejelentkezett felhasználónak, amit így a `view` is el tud érni. Hiba esetén kiíródik a hibaüzenet.

A **ShowMyFavouritesBeanController.java** osztály a `model`ből betölti az aktuális felhasználó összes kedvencekhez adott hirdetését, amit így a `view` is el tud érni. Az esetleges betöltési hibákról a felhasználó értesülni fog.

A **ShowOldAdsBeanController.java** osztály kezdetben betölti a `model`ből az összes hirdetést. Az esetlegesen fellépett hibáról értesítést kap az admin. A

*deleteSelectedItems()* függvény a kijelölt elemekre egyenként meghívja a model függvényét, hogy azzal törölje az elemeket. Végül újrainicializálja az osztályt, hogy a felületen a törölt elemek azonnal eltűnjenek. Az esetleges hibákról itt is értesül az admin.

### 3.7.1.2 Model

A **MyApp\src\main\java\hu\viczianlilla\myapp\entity** és **MyApp\src\main\java\hu\viczianlilla\myapp\service** mappákban találhatóak az MVC architektúra modelnek megfelelő részei. Az entity mappában az entitásokat hoztam létre. Ezek közös jellemzője, hogy az összes adattaguk `private`, és az ezekhez tartozó gettereik és settereik publikus tagfüggvények. Az entitások (és adattagjaik) egy-egy adattáblát írnak le Java nyelven. Mind a négy entitásnak (`Ad`, `User`, `Comment`, `Favourite`) van egy automatikusan generálódó, és növekvő `id`-je, amely az adatbázisba való beszúráskor jön létre. A projektben a `@Table` és `@Column` annotációk segítségével könnyedén közölhetjük a Hibernate-tel, hogy melyik adattáblából, melyik entitást hozza létre, a táblázat oszlopainak az entitás adattagjai fognak megfelelni. Fontos, hogy a megfelelő annotáció a megfelelő adattag elé kerüljön. Az egyes adattagoknál jelölve van a maximális méret a `@Size` annotációval, vagy a minimális, maximális érték a `@Min`, `@Max` annotációkkal. Ezeket az annotációkat az alkalmazás automatikusan figyelembe veszi a validáció során. Az `@Id` és `@GeneratedValue` segítségével értem el azt, hogy a megfelelő adattagok (jellemzően az entitások azonosítói) ne Java oldalon kapjanak értéket.

#### 3.7.1.2.1 Entitások

##### Ad osztály

Az alkalmazásban a hirdetések az **Ad.java** osztályban lettek definiálva. Eltérően a többi entitástól az `Ad` osztálynak két konstruktora van, az egyik egy üres konstruktor, a másik pedig az `id` kivételével az összes adattagot paraméterként kapja meg. A felhasználó azonosítója ugyan `User` típusként van megadva az osztályban, de az annotációkkal megadott adatkapcsolat miatt a Hibernate tudja értelmezni, hogy az adatbázisban csak a felhasználó azonosítója, és nem az egész felhasználókerült tárolásra. Majd ez alapján az adatbázis összekapcsolja a két



entitást. A szöveges mezők Stringként kerültek tárolásra, a többi adattag számként, azaz int, Integer vagy Long típusként, a dátum Date típusként.

Ad
<pre> int ad_id; User user_id; String brand; String model; Integer year; int fuel; Long price; Integer km; int condition; int category; int automatictransmission; int alu; int climate; int tempomat; int servicebook; int drawbar; String city; Date dateOfCreation; String description; </pre>
<pre> Ad() Ad(User user_id, String brand, String model,     Integer year, int fuel, Long price, Integer km,     int condition, int category, int automatictransmission, int alu,     int climate, int tempomat, int servicebook, int drawbar, String city,     Date dateOfCreation, String description) int getAd_id() void setAd_id(int ad_id) User getUser_id() void setUser_id(User user_id) String getBrand() void setBrand(String brand) String getModel() void setModel(String model) Integer getYear() void setYear(Integer year) int getFuel() void setFuel(int fuel) Long getPrice() void setPrice(Long price) Integer getKm() void setKm(Integer km) int getCondition() void setCondition(int condition) int getCategory() void setCategory(int category) int getAutomatictransmission() void setAutomatictransmission(int automatictransmission) int getAlu() void setAlu(int alu) int getClimate() void setClimate(int climate) int getTempomat() void setTempomat(int tempomat) int getServicebook() void setServicebook(int servicebook) int getDrawbar() void setDrawbar(int drawbar) String getCity() void setCity(String city) Date getDateOfCreation() void setDateOfCreation(Date dateOfCreation) String getDescription() void setDescription(String description) </pre>

50. ábra Hirdetést megvalósító osztály

## Comment osztály

Az **Comment.java** osztály a hozzászólásokat valósítja meg.

Comment
<pre> int comment_id; int car_id; String username; String message; Date dateOfCreation; String modified; </pre>
<pre> int getComment_id() void setComment_id(int comment_id) int getCar_id() void setCar_id(int car_id) String getUsername() void setUsername(String username) String getMessage() void setMessage(String message) Date getDateOfCreation() void setDateOfCreation(Date dateOfCreation) String getModified() void setModified(String modified) </pre>

51. ábra Hozzászólást megvalósító osztály

## Favourite osztály

A Favourite.java osztály a kedvenceket valósítja meg.

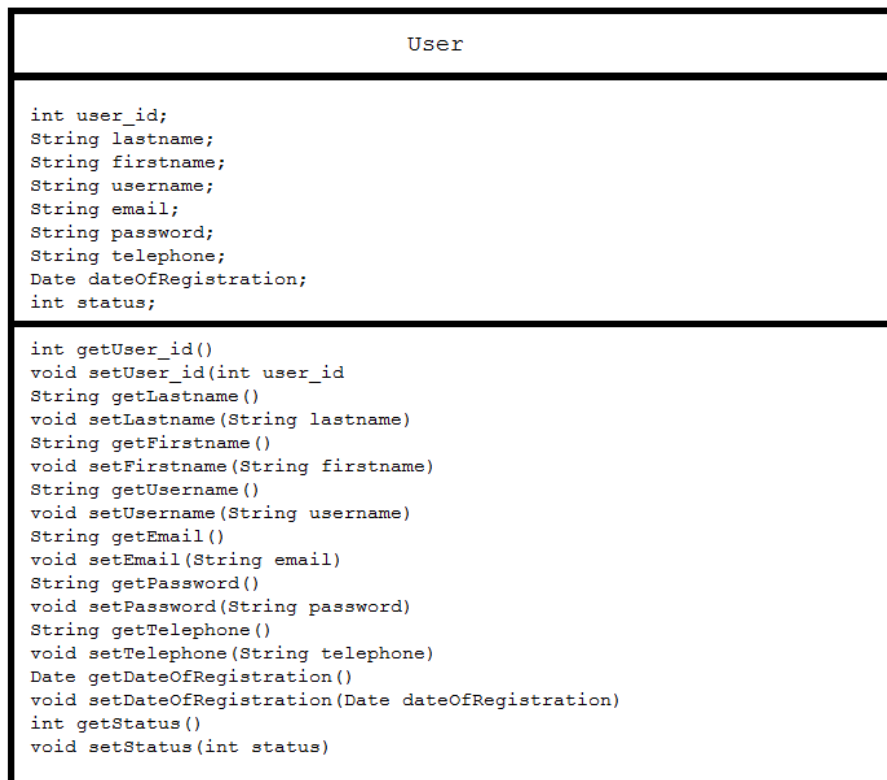
Favourite
<pre> int favourite_id; int car_id; String username; </pre>
<pre> int getFavourite_id() void setFavourite_id(int favourite_id) int getCar_id() void setCar_id(int car_id) String getUsername() void setUsername(String username) </pre>

52. ábra Kedvenceket megvalósító osztály

## User osztály

A **User.java** osztály a felhasználót valósítja meg. A felhasználó emailcíme csak egy megadott reguláris kifejezésnek<sup>13</sup> megfelelő emailcím lehet. A status adattag 0 esetén felhasználót jelent, 1 esetén admint.

<sup>13</sup> $^{\wedge}[_{A-Za-z0-9-}\wedge]+(\wedge[_{A-Za-z0-9-}\wedge])^{*}@[A-Za-z0-9-]\wedge[_{A-Za-z0-9-}\wedge])^{*}(\wedge[_{A-Za-z}\{2,\}\wedge])^{*}$



53. ábra Felhasználót megvalósító osztály

### 3.7.1.2.2 Service-ek

A `MyApp\src\main\java\hu\viczianlilla\myapp\service` mappában található az összes service osztály. Ezek kommunikálnak az adatbázissal a Hibernate-beli Session Factory segítségével. Pontosabban a Session Factory példányán megfelelő függvényhívásokat végezve lehetséges az adatbázisból való lekérdezés a service osztályokon belül. A Springnek köszönhető annotációkat a service osztályokban is alkalmaztam. Az `@Autowired` annotáció segítségével a Spring tudja, hogy honnan és hogyan húzza be az adattagot. A `@Transactional` annotáció segítségével a Spring automatikusan nyitja és zárja a tranzakciókat a függvény hívásakor.

Az **AdService.java** osztályban az *insert(Ad ad)* függvény egy hirdetést szűr be az adatbázis ads nevű táblájába. A *getAdById(int id)* függvény visszatér a paraméterül kapott azonosítójú hirdetéssel az ads nevű táblát vizsgálva. A *getAdById(int carId,int userId)* függvény visszatér a paraméterül kapott autó és felhasználó azonosító alapján kiszűrt hirdetéssel az ads nevű táblát figyelembe véve. A *getAllAds()* függvény visszatér az összes hirdetéssel dátum szerint rendezetten az ads táblából. A *getAllMyAds(int user\_id)* függvény visszatér a paraméterül kapott felhasználó azonosítóhoz tartozó ads táblabeli hirdetésekkel. A

*getAllMyFavouriteAds(String username)* függvény, a kapott felhasználónév paraméterhez tartozó felhasználó kedvenc hirdetéseit listaként adja vissza, az ads és a favourites tábla segítségével. A *deleteAdById(int id)* nevű függvény a hirdetés azonosítója alapján törli az adatbázis ads táblájából a hirdetést. Az *update(Ad ad)* nevű függvény a paraméterben kapott hirdetéssel felülírja a már eltárolt megegyező azonosítós hirdetését az ads táblában. Az osztály *addToMyFavourites(int car\_id, String username)* függvénye az adatbázis favourites táblájában létrehoz egy új kedvencet a paraméterben kapott adatokkal. A *deleteFromMyFavourites(int car\_id, String username)* nevű függvény törli azt a hirdetést a kedvencek közül, amelynek a paraméterben elsőként kapott szám a hirdetés azonosítója, és a paraméterben másodikként kapott szöveg a felhasználóneve. Az *isShownFavouriteButton(int car\_id, String username)* függvény a kapott paraméterek alapján logikai értékkel tér vissza annak megfelelően, hogy van-e már eltárolva a favourites táblában ehhez az autó azonosítóhoz, és felhasználónévhez kedvenc. Az osztály *filterAds(Map<String, Object> filters, String orderBy)* függvénye végzi a keresést az adatbázis ads tábláján. A függvény az adatbázisból lekérdezett, és a feltételeknek megfelelt hirdetések listájával tér vissza.

A **CommentService.java** osztály *insert(Comment comment)* függvénye az adatbázis comments táblájába beszúrja a paraméterül kapott hozzászólást. A *getCommentsByCarId(int car\_id)* függvény visszaadja egy listában azokat a hozzászólásokat, amelyek hirdetés azonosítója a paraméterül kapott car\_id-vel megegyezik a comments táblában. A *findCommentByCommentId(int comment\_id)* függvény a comments táblából kiszűri a paraméterrel megegyező azonosítójú hozzászólást, és visszaadja Comment entitásként. Az osztály *setModification(Comment comment)* függvénye menti a comments táblában a paraméterül kapott moderált hozzászólást.

A **CustomUserDetailsService.java** osztály *findByUserName(String username)* függvénye a users táblában megkeresi a paraméterül kapott felhasználónévű felhasználót és visszatér vele User entitásként. Az osztály *loadUserByUsername(String username)* függvénye a paraméter alapján megkeresi a users táblából a felhasználót, és áttaszformálja Springes Userre, hogy azzal be lehessen jelentkezni. Ha nem létezik a paraméterben kapott felhasználónévű user

a users táblában, akkor egy `UsernameNotFoundException` váltódik ki. A `getGrantedAuthorities(User user)` függvény hozzárendeli az admin vagy a felhasználói szerepkört a paraméterül kapott user számára, és visszatér a felhasználó szerepköreivel.

A **UserService.java** osztály az `insert(User user)` függvénnyel a users táblába beszúrja a paraméterül kapott usert. Beszúrás előtt a felhasználó jelszavát hasheli. A `getUserByUsername(String username)` függvény visszaadja a users tábla paraméterben megkapott felhasználó nevű felhasználóját User entitásként. Az `update(User user)` nevű függvény a paraméterben kapott felhasználóval felülírja a már eltárolt megegyező azonosítós felhasználót a users táblában.

### 3.7.1.3 View

A view-nak (nézet) a webapp mappában, és almappáiban lévő fájlok felelnek meg.

A **MyApp\src\main\webapp\resources\css\layout.css** fájl a program egyetlen css kiterjesztésű fájlja. A PrimeFaces alapbeállításainak köszönhető az alkalmazásom alapkinézete. Például a szöveges mezők, legördülő mezők, checkboxok kinézetén nem változtattam. Azonban az egységesebb kinézet érdekében megváltoztattam például a színeket, betűméreteket, betűtípusokat az oldal nagy részén. Képet szúrtam be a menüsorba,<sup>14</sup> és a háttérhez.

A **MyApp\src\main\webapp\resources\css\img** mappa két képet tartalmaz. A **cari8.png** fájl a menüsorba került beszúrásra. A **back3.png** pedig háttérképként lett beállítva.

A **MyApp\src\main\webapp\resources\js\filledRate.js** javascript nyelvű fájlja a hirdetés feladásánál és szerkesztésénél használt kitöltöttségi faktor kódját tartalmazza. A hirdetés feladása és szerkesztése közben meghívódik minden alkalommal, mikor kikattint egy mezőből a felhasználó, valamint a *Hirdetés feladása* vagy *Módosít* gomb lenyomásakor szintén, hogy biztos helyes értéke legyen a kitöltöttségi faktornak.

---

<sup>14</sup><https://www.bmwgroup.com>

A **MyApp\src\main\webapp\WEB-INF\lib\poi-3.8-20120326.jar** fájl az exportáláshoz szükséges. Az Apache hivatalos oldaláról ingyenesen letölthető.<sup>15</sup>

A **MyApp\src\main\webapp\WEB-INF\faces-config.xml** fájlban lett beállítva, hogy az xhtml kiterjesztésű fájlokba nem csak statikus értékeket írhatunk, de megfelelő szintaxis mellett dinamikus kifejezéseket is. (A dinamikus kifejezésre példa: `{PeldaJavaOsztaly.adatTag}`. Adattag helyett ugyanígy függvényt is hívhatunk: `{PeldaJavaOsztaly.fuggvenyNeve()}`). Továbbá itt állítottam be, hogy az esetleges hibás bejelentkezéseket mindig figyelje, és kezelje az alkalmazás. Ezenkívül az admin oldali régi hirdetések felület navigálása itt található, azonban a navigáció nagy része nem ezen a fájlon keresztül lett megoldva.

A **MyApp\src\main\webapp\WEB-INF\web.xml** fájlban beállításra került, hogy a HTTP requesteket a Java Spring fogja kezelni. Itt került meghatározásra, hogy ha egy máshol le nem kezelt Java exception idáig eljut, akkor az oldal navigáljon az error.xhtml-re. 404-es (a kért oldal nem létezik) vagy 500-as hiba (belső kiszolgáló hiba) esetén vigyen a bejelentkezési oldalra. Az xhtml fájlokban található hozzászólásokat hagyja figyelmen kívül. A PrimeFaces témái közül itt adtam meg, hogy az Omega nevű témát társítsa a weboldalakhoz. Valamint beállításra került, hogy a JSF az xhtml kiterjesztésű oldalakat vizsgálja.

A **MyApp\src\main\webapp\admin\showOldAds.xhtml** oldal csak admin számára érhető el. Az összes hirdetést láthatja egy táblázatban összesítve, dátum szerint rendezve. Joga van törölni bármely hirdetéseket. A táblázat exportálására is lehetősége van.

A **MyApp\src\main\webapp\user** mappában lévő xhtml típusú fájlok, felhasználóként (és adminként) érhetőek el.

Az **editAd.xhtml** oldal segítségével a felhasználó szerkeszteni tudja a saját hirdetést.

Az **editProfile.xhtml** oldal felületet biztosít a felhasználónak, a saját, regisztráláskor megadott adatainak szerkesztésére. A felhasználónév nem szerkeszthető. A jelszó pedig nem kerül megjelenítésre biztonsági okokból.

A **newAd.xhtml** oldalon a felhasználó új hirdetést adhat fel.

---

<sup>15</sup><http://www.apache.org/dyn/closer.cgi/poi/release/bin/poi-bin-3.8-20120326.zip>

A **showMyAds.xhtml** fájl a felhasználó saját hirdetéseinek megtekintéséhez nyújt grafikus felületet.

A **showMyFavourites.xhtml** fájl segítségével a felhasználó a saját kedvencekhez adott hirdetéseit tudja megtekinteni.

A **MyApp\src\main\webapp\** mappában lévő xhtml típusú fájlok, vendégként is elérhetőek.

Az **error.xhtml** oldalra lesz navigálva a vendég, egy nem várt hiba esetén. (Pl.: már nem létező hirdetés szerkesztése.)

Az **index.xhtml** oldal biztosítja a felületet a kereséshez.

A **layout.xhtml** fájlban készült el a weboldal állandó elemeit tartalmazó része. A menüsor, az oldal törzse, lábrésze, ebben az xhtml típusú fájlban van, és a többi oldalon ebből a fájlból kerülnek betöltésre az ennek megfelelő részek.

A **login.xhtml** a bejelentkezéshez szolgáltatja a felületet.

A **registration.xhtml** a regisztrációhoz szolgáltatja a felületet.

A **showAd.xhtml** a hirdetés megtekintésére szolgál.

## 3.8 Teszt

A Unit tesztekben a Service osztályok függvényei kerültek tesztelésre, hogy valóban helyes adatok kerülnek-e tárolásra az adatbázisban.

A Use Case tesztekre azért van szükség, mert sok funkcionál már a felületen, és/vagy Java oldalon meghatároztuk, hogy milyen adatokat fogadhatunk el az oldal látogatójától.

### 3.8.1 Use Case tesztek

A honlap megnyitása után, az oldal látogatója a menüsorból – közvetlenül vagy közvetetten – elérhet minden menüpontot, amelyhez joga van. Valamennyi funkciót egyesével kipróbáltam. A validációs szempontoknak is, és a feladat meghatározásakor megadott elvárásoknak is megfelelt az oldal.

#### 3.8.1.1 Validációs szempont

**A vendég számára elérhetőek:**

- Keresés
  - Keresés előtt, a vendég a keresési mezők alatt láthatja az összes hirdetést.
  - A vendég az autó adatai alapján tud keresni, ár vagy létrehozás dátuma szerint rendezni a keresés eredményét.
  - A keresés eredményéből lehet kiválasztani egy hirdetést, amelyet megtekintenénk.
    - A megtekintett hirdetésnél láthatóak az autó adatai, és egy térkép, amely az autó helyét jelöli.
  - Amennyiben olyan keresést adott meg a felhasználó, amelynek nincs találat, úgy nem lesz elem listázva a *Keresés* gomb alatt.
  - A keresőmezőkre különböző validációk kerültek:
    - A márka, modell, város mezők maximum 20 karaktert engednek begépelni.
    - Az ár, km mezőkben 0-999 999 999 között adhatunk meg értékeket.
    - Az évnél 0-9999 közötti érték adható meg.



- Az ár, km, év mezők esetén az első mezőbe szükséges írni a kisebb számot, a másodikba a nagyobbat. Ha a vendég a két mezőt felcseréli, a program nem tekinti hibának, csak nem lesz találat a keresésnek.
  - A keresés többi mezőjénél a vendég nem tud hibás adatot megadni.
  - A keresés során a kis és nagy betűk között nincs különbség.
- Bejelentkezés
  - Ez a menüpont szolgál arra, hogy bejelentkezzünk a már regisztrált felhasználónk felhasználónevével és jelszavával.
    - A bejelentkezés lehet sikeres, ha megfelelő adatokkal töltöttük ki a mezőket.
    - A bejelentkezés lehet sikertelen, ha rossz jelszót és/vagy felhasználónevet adtunk meg.
    - A bejelentkezéskor maximum 20 karakteres felhasználónevet, és jelszót lehet megadni. A felület több karakter begépelését nem engedi.
- Regisztráció
  - Új felhasználói fiókot tudunk létrehozni a megadott mezők helyes kitöltésével. Ekkor átkerülünk a Bejelentkezés oldalra, és üzenetet kapunk a sikeres regisztrációról.
  - Helytelen regisztrációs adatok során a hibás adatokról értesítést kap a vendég, és lehetősége van módosítani azokat. Az adatok validálásakor a program figyelembe veszi, hogy:
    - Az összes mező kitöltése kötelező.
    - A Felhasználónév, Jelszó, Jelszó újra mezők csak számokat, az angol abc kis/nagy betűit tartalmazhatják. (Ékezetes karaktereket, szóközöket, tabulátorokat, egyéb speciális karaktereket nem.)
    - Az emailcím emailcím formátumú és maximálisan 30 hosszúságú lehet.
    - A telefonszám értéke nem negatív és maximum 13 hosszúságú szám lehet.
    - Az összes többi mező maximum 20 hosszúságú lehet.

- A Jelszó és Jelszó újra mezők tartalmának meg kell egyezniük.
- A megadott felhasználónév helytelennek számít, ha olyan felhasználónéven már létezik felhasználó.

A vendég a mezők maximális hosszúságára vonatkozó kritériumokat nem tudja megsérteni, mivel a program nem engedi, hogy több karaktert írjon be a felhasználó a megengedettnél. A többi kritérium megsértése esetén, a vendég az aktuális regisztrációs oldalon értesül a hibákról. Javíthatja az elrontott mezőket.

### **A felhasználó számára elérhetőek:**

#### **- Keresés**

- A keresés menete megegyezik a vendég számára elérhető keresés menetével. Az innen elérhető hirdetés megtekintése azonban kiegészül néhány funkcióval:

- A felhasználó a saját hirdetését itt szerkesztheti.
  - Szerkesztéskor ugyanazok a validációk vannak érvényben, mint hirdetés feladásánál. (Lásd Új hirdetés feladása.)
- Törölheti.
  - Sikeres törlés esetén a felhasználó üzenetet kap a hirdetés törléséről, és átkerül a Hirdetéseim menüpont alá. Ezután már sem a Keresésnél, sem a Kedvenceknél, sem a Hirdetéseimnél nem lesz megtalálható a törölt hirdetés.
- Bármely hirdetést hozzáadhat a Kedvencekhez,
- vagy törölhet a Kedvencekből.
- Bármely hirdetést exportálhat,
- olvashatja a megírt hozzászólásokat,
- hozzászólhat.
  - Hozzászólás írása során 300 karaktert írhat a mezőbe maximálisan.

- Ha szóközt, tabulátort, enter nem tartalmaz a hozzászólása, akkor automatikusan tördelve fogja megjeleníteni a felület. Ehhez adatbázisban nem kerül tárolásra sem szóköz, sem tabulátor, sem enter.
- Profil szerkesztése
  - A felhasználónak itt lehetősége van szerkeszteni a felhasználói adatait, kivételt képez a felhasználónév.
    - A profil szerkesztésénél ugyanolyan validációs feltételeknek kell megfelelni, mint a regisztráció során. (Lásd Regisztráció.) Az egyetlen különbség az, hogy a felhasználónév nem szerkeszthető.

A sikeres és sikertelen módosításról is értesítést kap a felhasználó a Módosítás gomb lenyomása után, az aktuális oldalon.

- Új hirdetés feladása
  - A felhasználó ennél a menüpontnál új hirdetést tud feladni. Ehhez fontos, hogy helyesen kitöltsön a kitöltöttségi faktornak is megfelelő számú mezőt.
    - A márka, modell, város, ár mezők kötelezőek.
    - A márka, modell, város egy maximum 20 karakteres szöveget vár.
    - Az ár, km mezők 0 és 999 999 999 közötti értékű számot várnak.
    - Az év 0 és az aktuális év közötti számot vár.
    - A kategória, üzemanyag, állapot értékének van alapértelmezett értéke.
    - A leírás maximum 300 szóból állhat. Ha nem tartalmaz sem szóközt, sem tabulátort, sem enter, a hirdetés megtekintésénél tördelve lesz megjelenítve.
    - Az automataváltó, vonóhorog, alufelni, klíma, tempomat, szervízkönyv checkboxok kitöltése nem kötelező, hisz van alapértelmezett értékük.

- A hirdetés feltöltéséhez a 80%-os kitöltöttségi faktornak is meg kell feleljen a hirdetés. A faktor számításához a checkboxokon kívül minden mező hozzátartozik. A 10 mezőből az első négy kötelező, és háromnak van alapértelmezett értéke. A maradék három mezőből minimum egyet ki kell tölteni, hogy a kitöltöttségi faktor elérje a 80%-ot. A faktort számoló függvény minden mezőből való kikattintáskor lefut, valamint a lap alján lévő gomb lenyomásakor is újra lefut. Így a feltöltéskor biztosan helyes lesz a kitöltöttségi faktor.

Amennyiben nem töltöttük ki a kötelező mezőket és/vagy a kitöltöttségi faktor túl kicsi, úgy az aktuális oldalon értesülünk a pontos hibáról és lehetőségünk van korrigálni.

Helyes adatok megadásakor az aktuális oldalon értesülünk a funkció sikerességéről.

- Hirdetéseim
  - A felhasználó ezalatt a menüpont alatt tekintheti meg a már feltöltött hirdetéseit, amennyiben van feltöltött hirdetése. Ha nincs, akkor nem listázódik ki egy hirdetés sem.
  - Abban az esetben, ha van feltöltött hirdetése, meg is tekintheti azt.
- Kedvencek
  - A felhasználó itt tekintheti meg a már a kedvencekhez adott hirdetéseit, amennyiben van ilyen. Ha nincs, akkor nem listázódik ki egy hirdetés sem.
  - Abban az esetben, ha van kedvence, meg is tekintheti őket, mint hirdetéseket.
- Kijelentkezés
  - A bejelentkezett felhasználó kijelentkezhet az oldalról ezzel a menüponttal.

#### **Az admin számára elérhetőek:**

A felhasználó számára elérhető funkciók az admin számára is elérhetőek, de kiegészülnek néhány menüpontban:

- Keresés, hirdetés megtekintés
    - o A keresés menüpont alatt elérhető hirdetést megtekintve, moderálhatja a már megírt hozzászólásokat.
      - Ekkor a hozzászólások azonnal frissülnek, láthatóvá válik a „MODERÁLVA” felirat a hozzászólás eredeti szövege helyett.
    - o Törölheti a teljes hirdetést.
      - A menüpont ugyanúgy működik, mint felhasználók számára.
- (A hirdetés megtekintése adott esetben elérhető lehet a Hirdetéseim, és a Kedvencek menüpont alatt is. Ezekből a menüpontokból is ugyanúgy működnek a funkciók.)
- Régi hirdetések
    - o Ebben a menüpontban táblázatos formában láthatja az admin, az összes feladott hirdetést, dátum szerint rendezve. Ezek közül bármennyit törölhet, akár többet is egyszerre, kijelölés segítségével.
      - A Törlés gomb lenyomásakor a látható táblázat azonnal frissül, így látjuk, ha a törölt elem már nem szerepel az adatbázisban.
    - o Ezenkívül xls fájlba exportálhatja az összes hirdetést.

Az említett eseteket kipróbáltam, teszteltem, és minden az elvárásoknak megfelelően, helyesen működött.

### **3.8.1.2 Párhuzamos használat**

Az oldal használata egyszerre több látogató számára lehetséges. Amennyiben rendszergazda által nincs letiltva, úgy azonos hálózatról használhatja több, különböző számítógépről, több különböző látogató. A legtöbb funkció probléma mentesen használható egyszerre többek számára is. Azonban vannak olyan esetek, amelyek működését érdemesebb részletezni.

### **Törölt hirdetésből adódó lehetséges problémák lekezelése:**

Amennyiben a felhasználó olyan hirdetés *Részletek* gombjára kattint rá, amelyet a keresés óta töröltek, „A kért oldal nem található.” oldalra lesz navigálva.

Ha egy felhasználó (user1) megtekint egy hirdetést, majd egy másik felhasználó (user2) ezt a hirdetést törli, akkor a user1 a megtekintés oldalon lévő bármely gomb (*Exportálás excelbe*, *Kedvencekhez*, *Törlés a kedvencekből*, *Szerkesztés*, *Törlés*, hozzászólásnál *Hozzád* gomb) lenyomásakor, „A kért oldal nem található.” oldalra lesz irányítva.

Ha a *Régi hirdetések* menüpont alatt, egy admin kijelöl törlésre egy elemet, de mire a *Töröl* gombra nyom, már törölve lesz a hirdetés, akkor az admin számára csak annyi lesz látható, hogy a táblázatban már nem szerepel a törölni kívánt elem. (Nem azért, mert sikeresen törölte az elemet, amely már ott sem volt, hanem azért, mert bár nem volt olyan elem, amit törölhetett, de a gomb lenyomása során az oldal táblázata frissül, így a friss táblázat már nem tartalmazza a korábban törölt elemet.)

Ha a hirdetést feladó felhasználó szerkeszti a hirdetést, az admin törli azt, majd a felhasználó megpróbálja menteni a módosításait, akkor, mivel az aktuális hirdetése már nem létezik, a felhasználó átkerül „A kért oldal nem található.” oldalra.

Ha egy olyan hirdetés kerül törlésre, amelyet valahány felhasználó a kedvenceihez adott és/vagy a hirdetésnek hozzászólásai is voltak, akkor az adatbázis kapcsolat miatt, a hirdetés törlésekor törlődik a hirdetéssel kapcsolatos összes hozzászólás, és kedvenc is a táblázatokból. Így a felhasználó nem tudja már megtekinteni a törölt hirdetést, még a *Kedvencek* menüpontból sem.

### **Hirdetés egyszerre történő szerkesztése**

A hirdetés szerkesztésére csak a hirdetés feladójának van lehetősége. Amennyiben két helyről jelentkezik be és szerkesztésre megnyitja ugyanazon hirdetést, majd mindkettőnél a *Módosít* gombot megnyomja, úgy a hirdetés kétszer lesz mentve egymás után. A később lenyomott *Módosítás* gombbal felülírjuk, az előbb lenyomott *Módosítás* gomb mentésének eredményét. Így végeredményben az oldalon a későbbi mentés lesz látható.

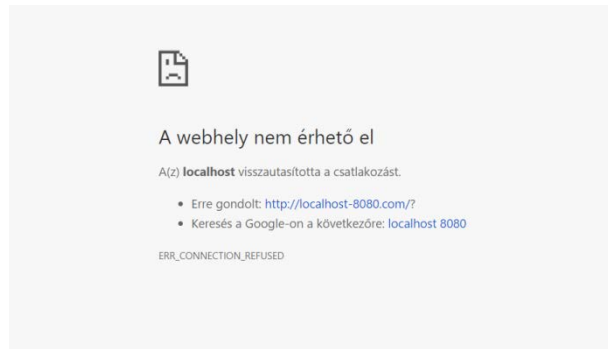
## Nem létező cím

Amennyiben nem létező, vagy a bejelentkezett felhasználó jogosultságaival nem elérhető oldalt próbálunk betölteni (a direkt URL betöltésével) a rendszer a bejelentkező képernyőre navigál.



**54. ábra** „A kért oldal nem található.” hiba oldal

Szerverrel való kapcsolat megszakadása esetén a böngészőnk, a Google Chrome fog minket tájékoztatni arról, hogy a webhely nem elérhető.

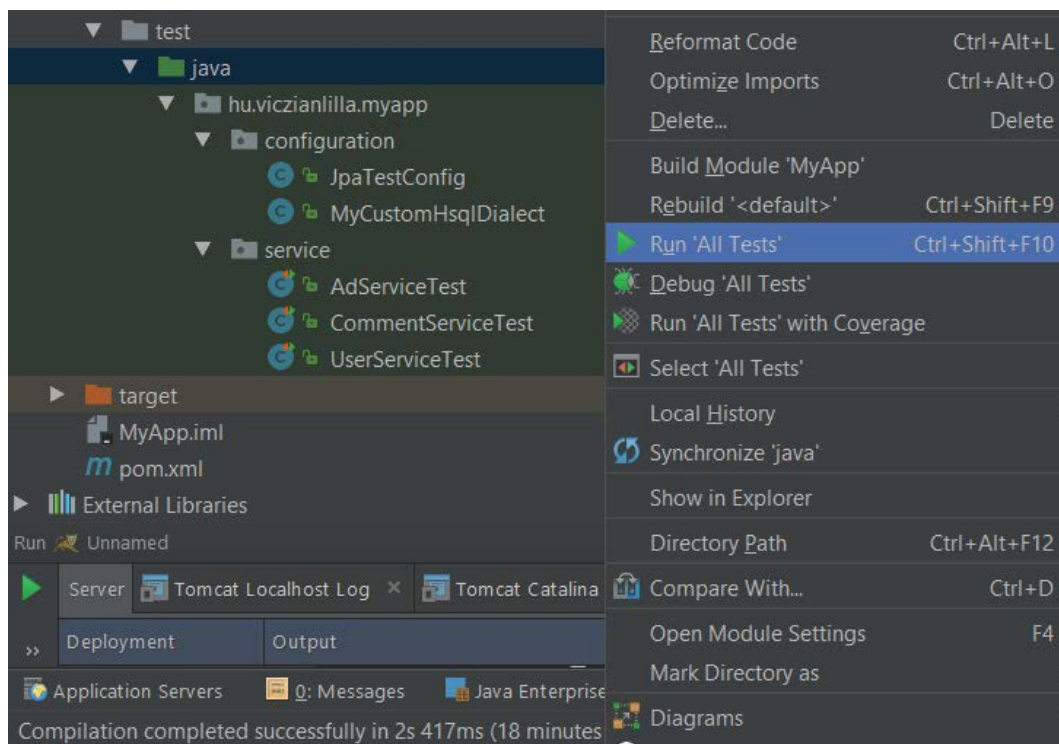


**55. ábra** Google Chrome által nem elérhető webhely

A fent említett eseteket kipróbáltam, teszteltem, és minden az elvárásoknak megfelelően, helyesen működött.

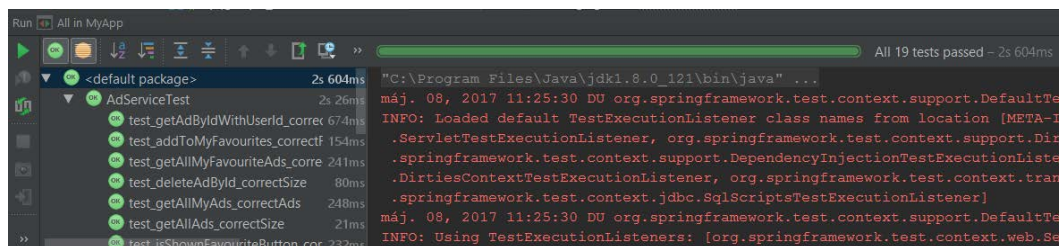
### 3.8.2 Unit tesztek

A `MyApp\src\main\java\hu\viczianlilla\myapp\service` mappában található service osztályok függvényeihez készültek Unit tesztek. A teszteket az IDEA fejlesztőkörnyezetből futtatom, a Java package-re jobb gombbal kattintva, Run All Tests-et indítva (56. ábra.)



**56. ábra** Tesztek futtatása

Amennyiben a tesztek helyesen lefutottak, egy „All 19 tests passed” visszajelzést kapunk:



**57. ábra** Tesztek futása utáni helyes végeredmény

Megjegyzés: amennyiben a Maven telepítve van, és a Path környezeti változó be van állítva (3.9), úgy elég kiadni a pom.xml-t tartalmazó mappában, parancsablakból egy parancsot: „mvn test”.



```
C:\WINDOWS\system32\cmd.exe
m8j. 16, 2017 5:21:21 DU org.springframework.test.context.transaction.TransactionContext startTransaction
INFO: Began transaction (1) for test context [DefaultTestContext@80b6dc6 testClass = UserServiceTest, testInstance = hu.viczianlilla.myapplication.service.UserServiceTest@59dc36d4, testMethod = testUserService@UserServiceTest, testException = [null]], mergedContextConfiguration = [MergedContextConfiguration@cdb2d95 testClass = UserServiceTest, locations = '{}', classes = '{}', propertySourceLocations = '{}', propertySourceProperties = '{}', contextInitializerClasses = '[]', activeProfiles = '{}', contextCustomizers = set[[empty]], contextLoader = 'org.springframework.test.context.support.DelegatingSmartContextLoader', parent = [null]]; transaction manager [org.springframework.orm.hibernate5.HibernateTransactionManager@309028af]; rollback [true]
m8j. 16, 2017 5:21:21 DU org.springframework.test.context.transaction.TransactionContext endTransaction
INFO: Rolled back transaction for test context [DefaultTestContext@80b6dc6 testClass = UserServiceTest, testInstance = hu.viczianlilla.myapplication.service.UserServiceTest@59dc36d4, testMethod = testUserService@UserServiceTest, testException = [null]], mergedContextConfiguration = [MergedContextConfiguration@cdb2d95 testClass = UserServiceTest, locations = '{}', classes = '{}', propertySourceLocations = '{}', propertySourceProperties = '{}', contextInitializerClasses = '[]', activeProfiles = '{}', contextCustomizers = set[[empty]], contextLoader = 'org.springframework.test.context.support.DelegatingSmartContextLoader', parent = [null]].
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.511 sec

Results :

Tests run: 19, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.802 s
[INFO] Finished at: 2017-05-16T17:21:21+02:00
[INFO] Final Memory: 9M/245M
[INFO] -----
C:\Users\Lilla\Desktop\MyApp>
```

58. ábra Sikeres tesztek lefutása parancsablakból

A **MyApp\src\test\java\hu\viczianlilla\myapp\configuration** mappa a tesztekhez szükséges konfigurációs fájlokat tartalmazza.

A **JpaTestConfig.java** osztályban adtam meg, hogy a tesztek futtatásakor ne a tényleges adatbázishoz kapcsolódjon a Hibernate, hanem egy önálló, csak a tesztek futása alatt létező, memóriában tárolt adatbázis jöjjön létre. Beállítottam, hogy milyen entitásokat használok, hogy a Hibernate memóriában tárolt adatbázist használjon, és létrehoztam az adatbázis szerkezetét.

A **MyCustomHsqlDialect.java** osztályban van meghatározva, hogy a memóriában tárolt adatbázis, hogy legyen kezelve.

A **MyApp\src\test\java\hu\viczianlilla\myapp\service** mappa a konkrét tesztek tartalmazza. A tesztek **@Before** annotációval ellátott **before()** függvénye a **@Test** annotációval megjelölt metódusok előtt futnak le. A tesztfüggvények nevei konvenciót követnek. Tartalmazzák, hogy ők teszt függvények, melyik függvényt tesztelik, és mit vizsgálnak, vagy milyen eredményt várnak. A négy entitás a teszteknel is a megfelelő táblában kerül mentésre:

- hirdetés (ad) az ads táblában,
- felhasználó (user) a users táblában,
- hozzászólás (comment) a comments táblában,
- kedvenc (favourite) a favourites táblában.

Az **AdServiceTest.java** tesztsztály az AdService osztály összes tagfüggvényét teszteli, a hirdetések az ads adattáblába, a kedvenceket a favourites adattáblába menti. A *test\_insert\_hasAd()* függvénye az AdService osztály insert függvényét vizsgálja, hogy ameddig nem kerül beszúrára elem, addig a táblázat mérete egyenlő-e 0-val, és amint beszúrára kerül egy elem, a tábla mérete már nem nulla. A *test\_getAdById\_correctAd()* tesztfüggvény vizsgálja, hogy az entitásként létrehozott hirdetés, setterei segítségével feltöltve, ugyanolyan adatokat tartalmaz-e, mint az adattáblába való beszúrást követően, az odakerült hirdetés. A *test\_getAdByIdWithUserId\_correctAd()* tesztfüggvény vizsgálja, hogy egy hirdetés beszúrást követően, keresve user\_id alapján a keresés eredménye nem lesz-e null, eggyel nagyobb user\_id esetén null-e és a talált eredmény tényleg azokkal az adatokkal rendelkezik-e, mint amit eredetileg be akartunk szűrni a táblába. A *test\_getAllAds\_correctSize()* tesztfüggvény vizsgálja, hogy az adService getAllAds() függvénye által kapott hirdetéseknek a mérete megegyezik-e nullával beszúrást megelőzően. 3 elem beszúrást követően viszont 3 lesz a mérete. A *test\_getAllMyAds\_correctAds()* függvény ellenőrzi, hogy egy újonnan létrehozott user-nél még nincs hirdetés, majd egy hirdetés beszúrást követően a felhasználóhoz tartozó hirdetések száma megegyezik-e eggyel. Valamint, az utoljára beszúrt user azonosítójánál eggyel nagyobb azonosítók száma nulla. A *test\_getAllMyFavouriteAds\_correctFavourites()* tesztfüggvény vizsgálja, hogy egy user és hirdetés beszúrást követően a kedvencek táblában valóban nincs még kedvenc. Egy kedvenc beszúrást követően, az adService getAllMyFavouriteAds függvénye vissza ad egy listát, amely mérete egyenlő eggyel, valamint a beszúrt elem adatai megegyeznek az általunk beszúrni kívánt elem adataival. A *test\_deleteAdById\_correctSize()* tesztfüggvény vizsgálja az adService deleteAdById függvényét. Hirdetés beszúrást megelőzően a getAllAds() függvény visszaadja az összes hirdetést egy listában, amely méretét vizsgáljuk, hogy vajon megegyezik-e nullával, beszúrást követően egyezik-e eggyel, még egy elem beszúrást követően megegyezik-e kettővel, és törlés után újra egyezik-e eggyel, majd nullával. A *test\_update\_correctAd()* tesztfüggvény vizsgálja, hogy hirdetés beszúrást követően a hirdetések száma egyenlő-e eggyel, és tényleg a beszúrni kívánt elem került-e beszúrást követően. Hirdetés módosítása után a hirdetések száma továbbra is egy-e, de a beszúrt elem tartalma már a módosított hirdetéssel egyezik-e meg. A *test\_addToMyFavourites\_correctFav()* tesztfüggvény vizsgálja, hogy adott user-

hez kezdetben még nincs-e beszúrva kedvenc. Majd, az adott user egy hirdetést a kedvencekhez ad. Az user-hez tartozó kedvencek száma egyenlő-e eggyel. Majd vizsgálja azt is, hogy a táblázatban tényleg a megfelelő hirdetés azonosítója került-e mentésre. A *test\_deleteFromMyFavourites\_correctSize()* tesztfüggvény vizsgálja, hogy az *adService deleteFromMyFavourites* függvény tényleg törli-e a megfelelő kedvencet a kedvencek táblából. A *test\_isShownFavouriteButton\_correctTrueFalse()* tesztfüggvény vizsgálja, hogy egy újonnan létrehozott hirdetés után az *adService isShownFavouriteButton* függvény valóban igazgal tér-e vissza, és miután hozzáadtuk a kedvencekhez a hirdetést, úgy valóban hamissal tér-e vissza, valamint, ha kitöröljük a hirdetést a kedvencekből, akkor szintén újra igazgal tér-e vissza a függvény. A *test\_filterAds\_correctSearch()* tesztfüggvény teszteli az *adService filterAds* függvényét, azaz a keresés helyes működését. Egy adattaghoz értéket rendel, ellenőrzi, hogy a függvénnyel megkereshető-e a hirdetés, majd törli az adattagot, és vizsgálja a következőt. Tehát így az összes kereshető adattag egyesével vizsgálatra kerül. A tesztfüggvény végén vizsgálatra kerül az összes mező helyes kitöltésének esete. A megadott adatok szerint pontosan egy érvényes találat lesz a keresésnek.

A **CommentServiceTest.java** tesztszámítógépes tesztosztály a *commentService* tagfüggvényeit teszteli. A *test\_insert\_hasComment()* tesztfüggvény vizsgálja, hogy új hozzászólás beszúrása esetén a táblázatba tényleg bekerült-e a hozzászólás. A *test\_getCommentsByCarId\_correctComment()* tesztfüggvény vizsgálja, hogy kezdetben, nem tartozik-e az adott hirdetéshez, hozzászólás a *comments* táblázatban. Majd az adott hirdetéshez történő hozzászólások beszúrása során vizsgálja, hogy megegyezik-e a hozzászólások mérete eggyel, majd kettővel és más hirdetés azonosítójával keresve továbbra is nulla hirdetés lesz-e az eredmény. A *test\_findCommentByCommentId\_hasComment()* tesztfüggvény vizsgálja, hogy hozzászólás beszúrása után a hozzászólás azonosítójával való keresés nem lesz-e null, és olyan üzenettel rendelkezik-e a beszúrt elem, mint az a hozzászólás, amelyet be kívántunk szűrni. A *test\_setModification\_correctUpdate()* tesztfüggvény vizsgálja, hogy a *commentService setModification* függvénye ténylegesen beállítja-e a moderálást adott hozzászólásra.

A **UserServiceTest.java** tesztosztály a userService tagfüggvényeit teszteli. A *test\_insert\_hasUser()* tesztfüggvény vizsgálja, hogy új „u” felhasználónevű user beszúrása után, tényleg lesz-e ilyen nevű user a users táblázatban. A *test\_getUserByUsername\_correctUser()* tesztfüggvény vizsgálja, hogy az újonnan beszúrt felhasználó, felhasználónevével és jelszavával ténylegesen létrejött ilyen user. A *test\_update\_correctUpdatedUser()* függvény vizsgálja, hogy helyesen módosította-e a user-t a UserService update() függvénye. A *test\_insert\_duplicateUserThrowsException()* tesztfüggvény vizsgálja, hogyha egy olyan felhasználónevű user-t szeretnénk beszúrni a táblába, amely már létezik, akkor PersistenceException kivétel váltódik-e ki.

## 3.9 Programkódból felhasználóbarát verziót - Maven telepítése

A Maven egy projektkezelő szoftver. Külön telepítést – a program fejlesztőkörnyezetbeli futtatásához nem igényel, mivel az IDEA biztosítja az ehhez szükséges fájlokat – azonban érdemes telepíteni a hivatalos oldaláról letölthető zip-et <sup>16</sup>, ahhoz, hogy könnyen indítható fájlt hozzassunk létre felhasználók számára:

- Töltsük le a Mavent hivatalos oldalról <sup>17</sup>,
- csomagoljuk ki a zip fájlt a C:\ mappába,
- PATH környezeti változóhoz adjuk hozzá a C:\apache-maven-3.5.0-bin\apache-maven-3.5.0\bin elérési útvonalat.

A Maven telepítése itt véget ért, a következő lépések a war fájl létrehozásához, működtetéséhez szükségesek:

- A projekt mappáján belül indítsunk parancsablakot és futtassuk a mvn -Dproject.build.sourceEncoding=UTF-8 clean install parancsot. Ennek eredményeképp létrejött a war kiterjesztésű fájlunk (egy tömörített fájl, amely webes környezetbe való deployolást tesz lehetővé),
- Ezt átmásolva a Tomcat szerver telepítési helyére, azaz a C:\apache-tomcat-9.0.0.M17\apache-tomcat-9.0.0.M17\webapps mappába könnyedén futtathatjuk az alkalmazást.
- A futtatáshoz először is indítsuk el a Tomcat szervert manuálisan, a C:\apache-tomcat-9.0.0.M17\apache-tomcat-9.0.0.M17\bin mappában lévő startup.bat nevű fájjal.
- Ekkor a Goolge Chrome megnyitása után a localhost:8080/myApp/index.xhtml linken elérhetjük a kezdőoldalt. Használhatjuk az alkalmazást.
- Végül ne felejtsük el leállítani a szervert a C:\apache-tomcat-9.0.0.M17\apache-tomcat-9.0.0.M17\bin mappában lévő shutdown.bat nevű fájjal.

---

<sup>16</sup><http://www-eu.apache.org/dist/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.zip>

<sup>17</sup><http://www-eu.apache.org/dist/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.zip>

## 4 Továbbfejlesztési lehetőségek

A létrehozott alkalmazás továbbfejlesztési lehetőségekkel rendelkezik:

- A keresés hatékonyságának növelése, azaz a szöveges mezők pontatlan kitöltése (egy-két karakter eltérése) esetén is legyen eredménye a keresésnek.
- Többféle sorba rendezési lehetőség a keresésnél.
- A felhasználó által megírt hozzászólások, szerkeszthetők és törölhetők legyenek.
- Az admin számára legyen egy felület, amelyen a felhasználókat admin jogosultsággal láthatja el.
- A hirdetéseknel képek feltöltésének lehetősége, és ezek megjelenítése.
- Az eladókat a vásárlók értékelhessék.
- Legyen lehetőség több hirdetés egyszerre történő feladására egy xls kiterjesztésű fájlból importálva.
- Az oldalon látható legyen, hogy a felhasználó magánszemély vagy autókereskedés.

## 5 Összegzés

A szakdolgozat célja egy olyan alkalmazás elkészítése volt, amely segítségével eladó autókat lehet kezelni. Az oldal látogatója kereshet a hirdetések között, regisztrálhat, bejelentkezhet. Bejelentkezés után a felhasználó a keresésen kívül, hirdetést adhat fel, szerkesztheti azt, és profilját is. Bármely hirdetést megtekintheti, kommentezheti, exportálhatja, kedvencekhez adhatja, vagy törölheti onnan. Saját hirdetését törölheti is. Megtekintheti egy külön oldalon a saját hirdetéseit, egy másikon a kedvenceit. Amennyiben a bejelentkezett felhasználó admin jogosultsággal rendelkezik, úgy bárki hirdetését törölheti, az összes hirdetést exportálhatja, moderálhatja a hozzászólásokat.

A programozás folyamán a legnagyobb kihívást a különböző keretrendszerek, programcsomagok, komponensek, módszerek megismerése jelentette.

Ezúton köszönöm meg témavezetőmnek, Mészáros Mónika tanárnőnek és külső konzulensemnek, Szita Baláznak, hogy a segítségemre voltak a szakdolgozat elkészítésében.

## 6 Melléklet

A mellékelt CD-n megtalálható a MyApp webes alkalmazás

- forráskódja (MyApp mappa),
- dokumentációja (Dokumentáció.pdf),
- war kiterjesztésű fájl a felhasználbarát megtekintéshez
- a Tomcat szerver (utóbbi kettő a Fejlesztői mappa tartalma),
- és az adatbázis létrehozásához elengedhetetlen fájl (adatbázis.sql).

CD tartalma mappaszerkezet szetint:

- Dokumentáció/Dokumentáció.pdf
- Fejlesztői mappa
  - o Forráskód/MyApp
  - o adatbázis.sql
- Felhasználói mappa



## 7 Irodalomjegyzék

- [1] Nyékyné Dr. Gaizler Judit: JAVA – Útikalauz programozóknak, ELTE TTK Hallgatói Alapítvány, 2009, [1408], ISBN-978-9630640923
- [2] Dorothy Graham, Erik Van Veenendaal, Isabel Evans, Rex Black: Foundations of Software Testing, Cengage Learning EMEA, 2008, [258], ISBN-978-1844809899
- [3] A belépés hibajelzéséhez:  
<http://www.ocpsoft.org/java/acegi-spring-security-jsf-login-page/>  
utolsó elérés: 2017. 05. 15.
- [4] A Spring Security helyes beállításaihoz:  
<http://www.mkyong.com/spring-security/spring-security-hibernate-annotation-example/>  
utolsó elérés: 2017. 05. 15.
- [5] Tervezés  
<https://www.softwareideas.net/>  
utolsó elérés: 2017. 05. 15.
- [6] MVC architektúra  
<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>  
utolsó elérés: 2017. 05. 15.
- [7] Java dokumentáció  
<https://docs.oracle.com/javase/7/docs/api/>  
utolsó elérés: 2017. 05. 15.
- [8] Java Spring hivatalos oldala  
<https://spring.io/>  
utolsó elérés: 2017. 05. 15.

[9] Az alkalmazás során előkerült hibák kijavításához:

<https://stackoverflow.com/>

utolsó elérés: 2017. 05. 15.

[10] A weboldal kinézetéhez és néhány eleméhez:

<https://www.primefaces.org/>

utolsó elérés: 2017. 05. 15.