

BGSzC Pestszentlőrinci Közgazdasági és Informatikai Szakgimnáziuma

1184 Budapest Hengersor 34.

Vizsgaremek dokumentáció

Lethal Devs Project

A csoport tagjai:

Lukács Dávid

Készítette:

Lukács Dávid

Tartalom

1	Bevezetés	3
1.1	Feladat leírás	3
1.2	A felhasznált ismeretek	3
1.3	A felhasznált szoftverek	3
2	Felhasználói dokumentáció.....	4
2.1	A program általános specifikációja.....	4
2.2	Rendszerkövetelmények	4
2.2.1	Hardver követelmények	4
2.2.2	Szoftver követelmények.....	4
2.3	A program telepítése.....	5
2.4	A program használatának a részletes leírása	8
3	Fejlesztői dokumentáció	15
3.1	Az alkalmazott fejlesztői eszközök	15
	Adatmodell leírása	15
3.2	Részletes feladatspecifikáció, algoritmusok.....	18
3.3	Tesztelési dokumentáció.....	24
	Összefoglalás.....	29
3.4	Önértékelés	29
3.5	Továbbfejlesztési lehetőségek	30
4	Felhasznált irodalom	31
5	Ábrajegyzék	32

1 Bevezetés

1.1 Feladat leírás

A projekt egy hirtelen ötlet alapján jött, először egy raktározó alkalmazáson gondolkodtunk azonban én végül egy Járműadatbázis rendszeren kezdtem el dolgozni, mivel az autók alapjáraton közel állnak hozzám, szeretem őket így egy olyan témájú alkalmazáson szerettem volna dolgozni.

1.2 A felhasznált ismeretek

Backend:

- C#
- PHP

Frontend:

- HTML5
- CSS
- JavaScript (jQuery)
- Bootstrap
- Windows Forms

1.3 A felhasznált szoftverek

Visual Studio 2022

Visual Studio Code

XAMPP

Bootstrap Studio

2 Felhasználói dokumentáció

2.1 A program általános specifikációja

Az alkalmazás járművek nyilvántartására alkalmas, az átlagos felhasználó tud regisztrálni, bejelentkezni. Ha esetleg elfelejtette a belépési adatait akkor tud jelszó visszaállítást kérni. Ezt meg tudja tenni egy webes felületről, amely reszponzív vagy a számítógépről egy webcímet elérve. Az alkalmazásban miután bejelentkezett a felhasználó pár gombnyomás segítségével vehet fel új járművet az adatbázisba vagy szerkesztheti a saját meglévőit. Ha adminisztrátor jogosultsággal rendelkezik, akkor az összes járművet tudja kezelni az asztali alkalmazásból, és hozzáférése van az összes felhasználóhoz, ahol látja az alapvető adataikat mint, felhasználónév, valódi név, születési dátum, lakcím, emailcím stb. Tud hozzáadni és törölni felhasználót a rendszerből, ugyan ezt tudja tenni a járművekkel, tud hozzáadni bárki nevéhez járművet vagy törölni azt. Módosíthatja őket vagy törölheti az adatbázisból. Ezen felül az admin felületen további statisztikai adatok találhatóak.

2.2 Rendszerkövetelmények

2.2.1 Hardver követelmények

Processzor: 2 magos legalább 1GHz sebességű vagy gyorsabb...

Memória: 1 Gigabyte (GB) vagy 2 GB 64-bit

Tárhely: 500 MB (megabyte) HDD vagy SSD

Grafikus gyorsító: DirectX 9 vagy újabb

Kijelző: 1280x720 vagy nagyobb felbontású

2.2.2 Szoftver követelmények

Windows 10 vagy újabb op, rendszer

.NET Runtime 4.7.2

MySQL Server (XAMPP)

Database fájl (Szerver)

Böngészőképes okostelefon (iPhone, Android) vagy Számítógép

2.3 A program telepítése

A program futtatásához szükséges egy Webszerver ehhez én az XAMPP nevű programot használtam, ezt is ajánlom, az adminisztrációs felület telepítését ez után találja.

- A program letöltéséhez keressük fel az alábbi linken KATT található weboldalt, és tölts le a programot.

Letöltés

A XAMPP egy könnyen telepíthető Apache disztribúció MariaDB, PHP és Perl komponensekkel. Csak tölts le és indítsa el a telepítőt. Ennyire egyszerű.

XAMPP Windows rendszerre 7.4.29, 8.0.18 & 8.1.5

Verzió	Ellenőrzőösszeg	Méret
7.4.29 / PHP 7.4.29 Mit tartalmaz?	md5 sha1	Letöltés (64 bit) 159 Mb
8.0.18 / PHP 8.0.18 Mit tartalmaz?	md5 sha1	Letöltés (64 bit) 161 Mb
8.1.5 / PHP 8.1.5 Mit tartalmaz?	md5 sha1	Letöltés (64 bit) 164 Mb

[Követelmények](#) [Kiegészítők](#) [További letöltések »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

1. XAMPP letöltése

- A programot ezután telepítse, különösebb odafigyelést nem igényel a telepítés, alapbeállítások megfelelőek.
- Ezután keressük fel a GitHub-on található projektet, és töltsük azt le a számítógépünkre, majd csomagoljuk ki.
- A VMS-Web mappában található fájlokat másoljuk ki, és illesszük be az XAMPP telepítési könyvtárának /htdocs nevű mappájába.
- Indítsuk el az XAMPP szoftvert és ezután indítsuk el az Apache és MySQL szervereket.
- Ha ezek sikeresen elindultak, a böngészőben írjuk be a következőt:
localhost/phpmyadmin
- Ha bejött a felület, hozzunk létre egy adatbázist, és a létrehozott adatbázisba importáljuk be a VMS-SQL mappában található MySQL fájlt.
- Ha ezzel megvagyunk bezárhatjuk a felületet.
- Ismét lépünk be a /htdocs mappába és a config.php fájlt nyissuk meg szövegszerkesztővel.

```
/* Database credentials. Assuming you
server with default setting (user 'root')
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'vms_lethaldevs');
```

2. PHP MySQL konfiguráció

- A következőket szerkesszük a saját adatainknak megfelelően, azaz a DB_SERVER legyen a szerverünk IP címe, esetünkben ez localhost lesz, a DB_USERNAME az az XAMPP-nél lehet root, a jelszó pedig üres, mert nem kér jelszót az alapbeállítás, a DB_NAME pedig a létrehozott adatbázisunk neve kell legyen.
- Ha ez sikerült mentjük el és zárjuk be a szövegszerkesztőt, majd lépünk be a /htdocs/php/lib/ mappába, és az itt lévő config.php fájlt is szerkesszük az előzőnek megfelelően.
- Ha ezzel megvagyunk akkor a weboldal telepítése ezzel kész is, innentől működni kell a weboldalnak.
- XAMPP-nél a localhost címet beírva érjük el az alkalmazást.



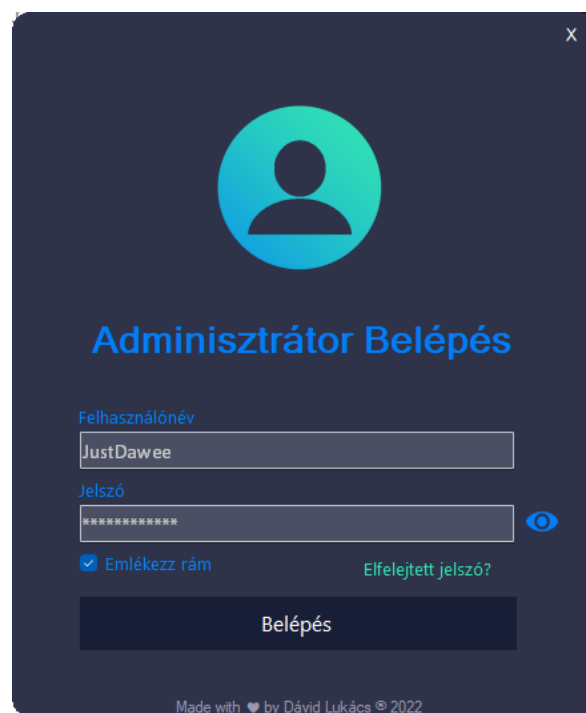
3. Weboldal bejelentkezés

- Az adminisztrációs felület telepítéséhez különösebb odafigyelés nem szükséges, az alkalmazást megtalálja a GitHub-on az alábbi linken.
- Töltse le az alkalmazást a jobb oldalt lévő Relases gombra kattintva, majd az oldal alján lévő VMS-AdminUI.zip fájlt töltse le és csomagolja ki.
- Hogy ez a felület működjön meg kell változtatni a MySQL adatokat melyet a Lethal Devs Project.exe.config-fájlban talál meg.

```
<Lethal_Devs_Project.Properties.Settings>
  <setting name="db_ip" serializeAs="String">
    <value>HOSTNAME</value>
  </setting>
  <setting name="db_port" serializeAs="String">
    <value>PORT</value>
  </setting>
  <setting name="db_uid" serializeAs="String">
    <value>USERNAME</value>
  </setting>
  <setting name="db_password" serializeAs="String">
    <value>PASSWORD</value>
  </setting>
  <setting name="db_database" serializeAs="String">
    <value>DATABASE</value>
  </setting>
</Lethal_Devs_Project.Properties.Settings>
```

4. MySQL csatlakozás konfiguráció

- Ha ez sikeresen megtörtént, indítsa el az alkalmazást a Lethal Devs Project.exe-vel és a bejelentkezési felület fogja fogadni.



5. Admin bejelentkezés

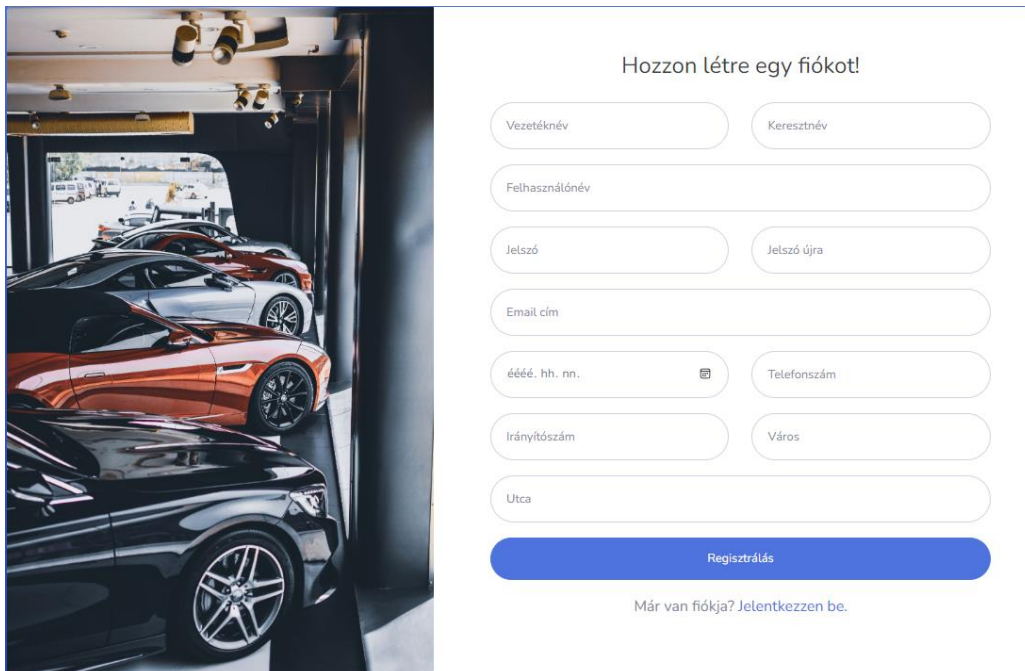
- Ezzel kész is lennénk, a program telepítése megtörtént, innentől tudja használni az alkalmazást, ha szeretnénk hogy ne csak a helyi hálózatról legyen elérhető, akkor a webszervert kell egy szolgáltatóhoz helyezni, a megfelelő adatokat módosítani.

2.4 A program használatának a részletes leírása

A weboldal használatához szükséges egy regisztráció, ahhoz, hogy ezt megtegyük írjuk be a webcímet, majd ha az oldal betöltött egy bejelentkezési felület fogad minket.

Weboldal használata:

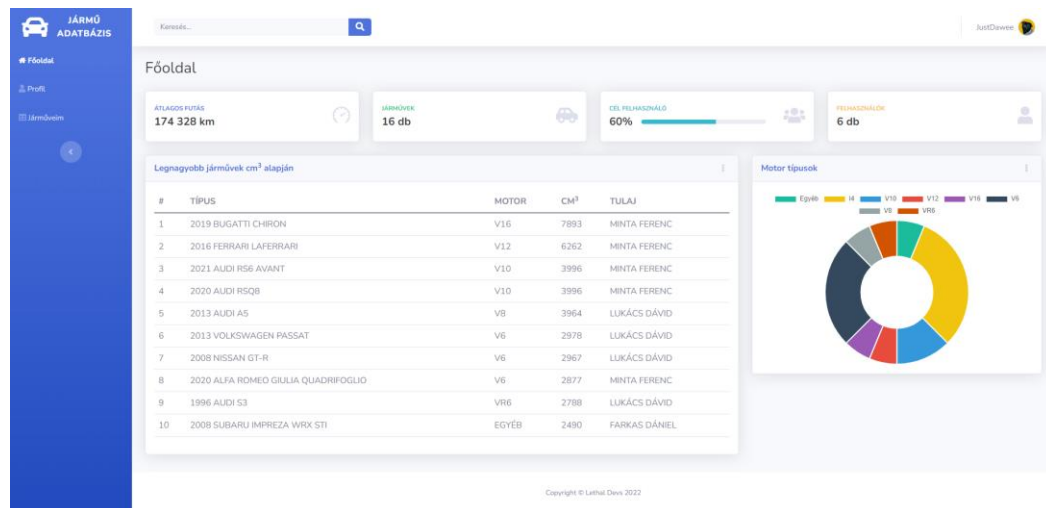
- Ha még nincs fiókunk, akkor hozzunk létre egyet a Hozz létre egy fiókot! -ra kattintva.
- A fiók létrehozásához meg kell adnunk egy felhasználónevet, jelszavat, email címet, születési dátumot, telefonszámot és egy címet.
- Ha megadtuk az adatokat akkor a Regisztrálás gombra kattintva hozhatjuk létre a fiókunkat.
- Ha sikeres volt a regisztráció, akkor a megadott felhasználónévvel és jelszóval tudunk belépni.



The image shows a screenshot of a web application for a car dealership. On the left, there is a photograph of several cars parked in a showroom. On the right, there is a registration form titled "Hozzon létre egy fiókot!". The form contains the following fields: "Vezetéknév" (Last name), "Keresztnév" (First name), "Felhasználónév" (Username), "Jelszó" (Password) with a "Jelszó újra" (Reset password) link, "Email cím" (Email address), "éééé. hh. nn." (Date of birth) with a calendar icon and "Telefonszám" (Phone number), "Irányítószám" (Postal code) and "Város" (City), and "Utca" (Street). A blue "Regisztrálás" (Register) button is at the bottom of the form. Below the button, there is a link that says "Már van fiókja? Jelentkezzen be." (Already have an account? Log in).

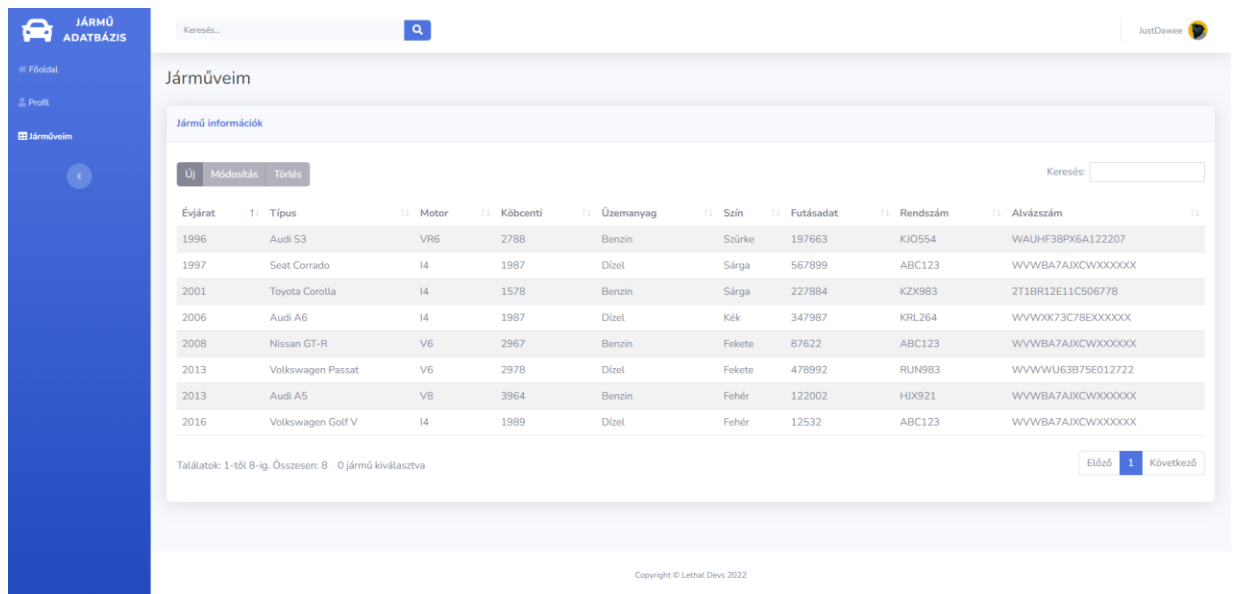
6. Weboldal regisztráció

- Sikeres bejelentkezés után a főoldal fogad minket, ahol néhány statisztikai adatot láthatunk, mint az adatbázisban lévő autók átlagos futása, hogy hány darab jármű van összesen az adatbázisban, egy célfelhasználó számlálót és hogy hány felhasználó van összesen.
- Ezen felül van egy lista a legnagyobb köbcentiméterrel rendelkező járművekről és egy diagram hogy milyen motorral szereltek a járművek.



7. Weblap főoldal

- A profil fülben az adatainkat tudjuk megváltoztatni, mint felhasználónév, email, cím, telefonszám.
- A járműveim fülben tudjuk igazán az alkalmazást kihasználni igazán, ha új a regisztrációnk akkor egy üres táblázat fogad minket.
- A járművet az új gombra kattintva tudjuk hozzáadni, ahol feldob egy űrlapot az oldal, amelyet ha kitöltünk és a hozzáadás gombbal felvisszük az adatbázisba, akkor onnantól látjuk, ezzel egy nyilvántartásba kerítve azt.
- Ha valamilyen adatot rosszul vittünk fel, vagy éppen megváltozott, akkor tudjuk módosítani azt, kattintson a módosítandó sorra, és utána a módosítás gombra kattintva ismét előjön az űrlap, amin a módosítandó adatot tudjuk felülírni.
- Ha a járművet törölni szeretné a nyilvántartásból, akkor jelölje ki azt a járművet, amelyet törölni szeretne (akár több darabot is kijelölhet a CTRL billentyű nyomva tartásával) és kattintson a törlés gombra. A rendszer megkérdezi hogy biztosan törölni szeretné-e a járműveket, hiszen félre is katinthatott.

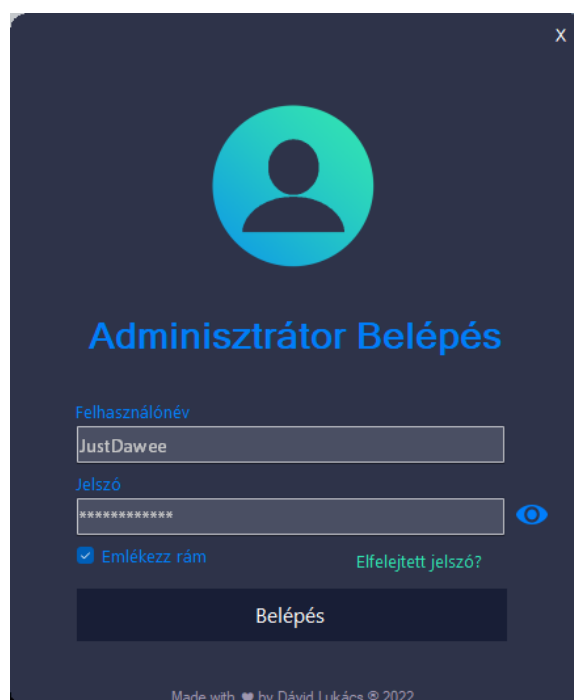


8. Járművek adattábla

Az admin felületen tudjuk kezelni (menedzselni) az egész rendszert. Ez egy asztali alkalmazás, amely csatlakozik az adatbázishoz, és megjelenít adatokat a rendszert kezelő számára.

Az admin felületen az alábbi funkciók találhatóak:

- Az első dolog ami fogad minket a program indításakor, az egy bejelentkező felület, ami hasonló ahhoz a felülethez amit a weboldalon is láttunk, szintén egy felhasználónévvel és egy jelszóval tudunk ide belépni.



9. Asztali alkalmazás belépés

- **FONTOS:** Hogy ide be tudjunk lépni, admin jogosultság szükséges. Ha ön a rendszer kezelője, akkor ezt a legkönnyebben úgy teheti meg, hogy belép a MYSQL adatbázisba a phpmyadmin vagy más MYSQL adatbázis kezelő alkalmazás segítségével pl. (Navicat) az adatbázisba, és a users táblában megkeresi a saját felhasználóját, és az admin résznél a 0-t átírja 1-re.
- Utóbbi csak először szükséges, ha még nincs adminisztrátor felvéve a rendszerbe, hiszen erről a felületről az admin jogosultságot is tudja majd kezelni.
- Ha sikeresen bejelentkezett akkor az alkalmazás főoldala fogadja majd, itt szintén statisztikai adatokkal találkozhatunk, ezek általános adatok, érdekességek a rendszerről és annak működéséről.

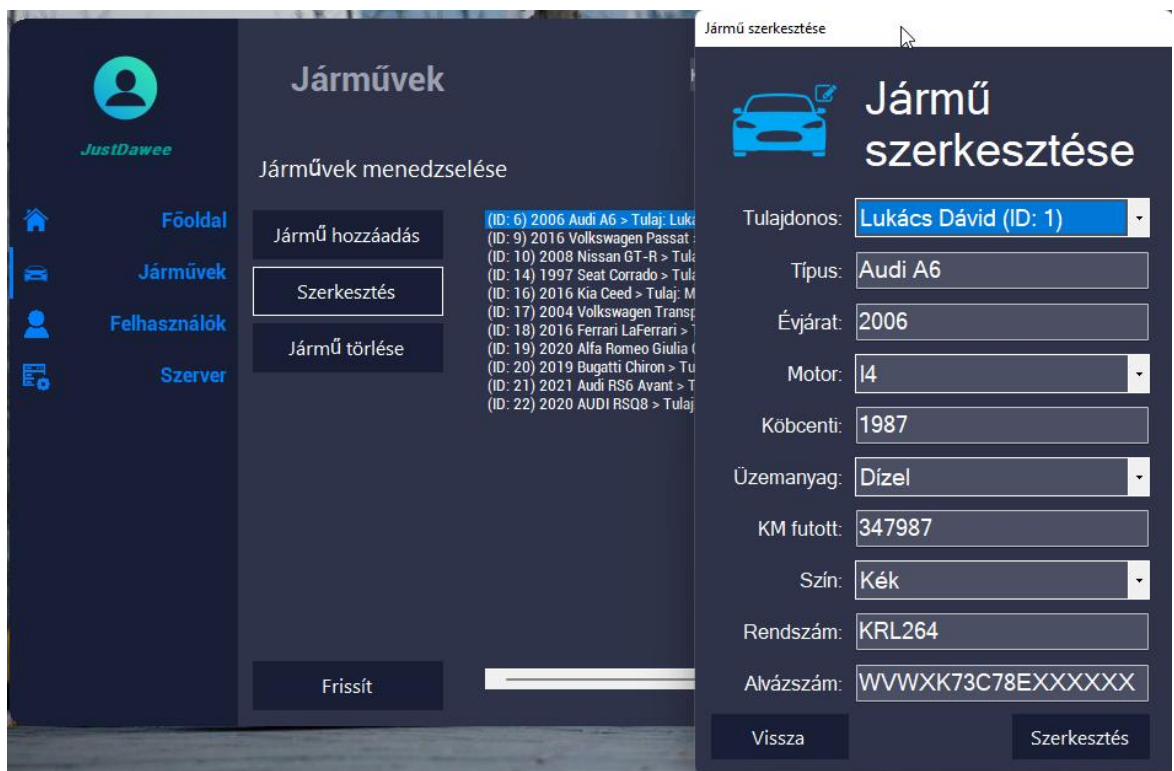


10. Asztali alkalmazás kezdőlap

- A járművek oldalon látható az összes jármű ami az adatbázisban szerepel, ezeket a járműveket tudja törölni, szerkeszteni vagy éppen újat hozzáadni.
- Hogy egy járművet szerkesszen ki kell választania a listából egy járművet, amelyre rákattint majd a módosítás gombbal előugrik egy űrlap ablak amelybe a jármű adatai vannak, és azokat tudjuk módosítani.
- Ha új járművet adunk hozzá, ki tudjuk választani a tulajdonosát, ezeket a regisztrált felhasználók közül tudjuk kiválasztani.



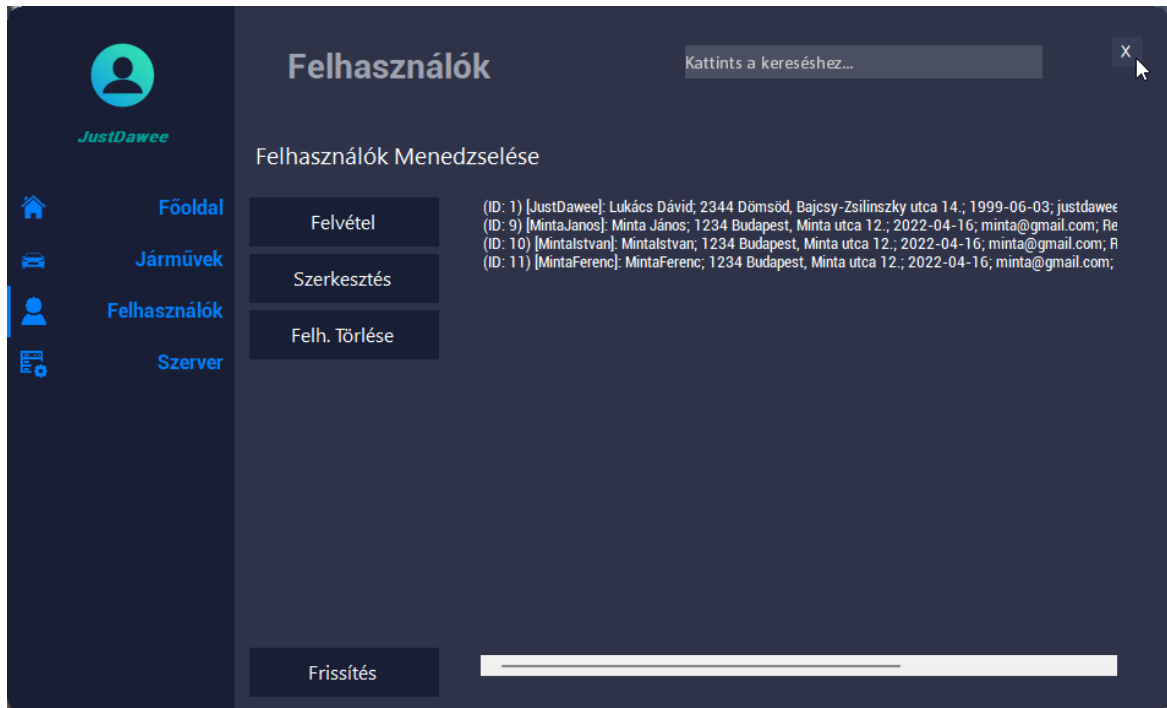
11. Járművek listázása



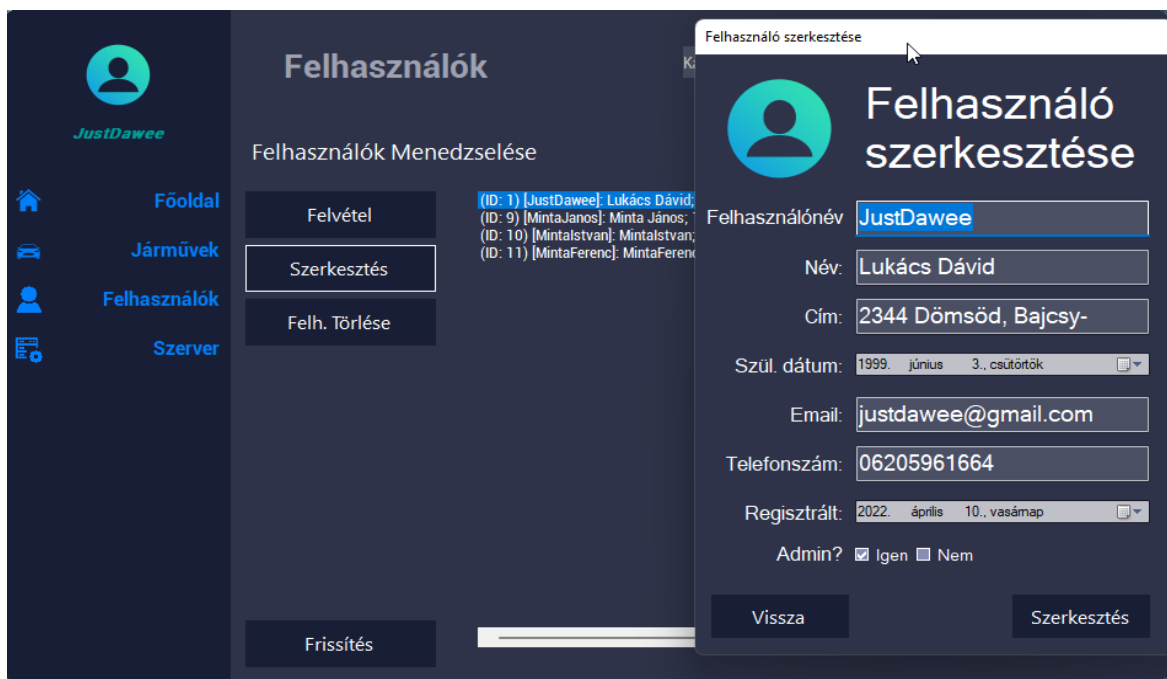
12. Jármű szerkesztése

- A következő oldalon a felhasználókat találjuk, ezeket hasonlóan, mint az előzőnél a járműveket, kilistázva látjuk.
- Tudjuk a felhasználókat módosítani, törölni, vagy újat hozzáadni.

- A felhasználó adatai csak az adminisztrátor számára láthatóak, de még ő sem lát mindent, például a jelszavát nem tudja módosítani, ezáltal nem is látja azt.



13. Felhasználók listázása



14. Felhasználó szerkesztése

- A szerver felületen a szerverrel kapcsolatos dolgokat tudjuk átállítani, itt a MySQL szerver adatait tudjuk átírni, hogy a program milyen adatokkal csatlakozzon a MySQL szerverhez.

- Tudunk biztonsági mentést készíteni az adatbázisról, a Biztonsági Mentés gombra kattintva a program létrehoz egy SQL fájlt a program mappájában, a jelenlegi adatokkal.
- Megtalálhatóak olyan adatok is ezen az oldalon, mint például hogy mióta fut a program, hogy hány SQL műveletet hajtottunk végre az indítás óta, és hogy jelenleg mekkora az adatbázis mérete. (Utóbbi kiírása annak függvényében változik, hogy mekkora fájl nagysággal dolgozunk, ha 1024KB felé megy akkor 1MB ha 1024MB felé megy akkor 1GB stb...)



15. Szerver beállítások oldal

3 Fejlesztői dokumentáció

Az asztali alkalmazás C# programnyelven, Visual Studio 2022 fejlesztői környezetben (IDE) készült, Windows Forms alkalmazás alapon. A keretrendszer .NET Framework 4.7.2, ez volt a hosszútávon támogatott alapbeállítás. Ebben az alkalmazásban az alap koncepció az objektumorientáltság volt, ezért törekedtem arra hogy minden részét elkülönítsem egymástól, ez többé kevésbé sikerült is. Az alkalmazás MySQL-en keresztül kommunikál egy adatbázissal ahová adatokat visz fel vagy kezeli azokat.

A másik része a programnak egy weboldal, amely reszponzívnak kellett hogy legyen, tehát mobilon, táblagépen, számítógépen is egyaránt arányainak megfelelően jelenik meg, ehhez Bootstrap-et használtam, amit a rácsos rendszernek megfelelően osztottam fel kis szekciókra. Ezáltal tud a weblap reszponzív lenni. A weboldal backend része PHP alapú, ez is MySQL szerverrel kommunikál és ezek alapján jeleníti/kezeli az adatokat.

3.1 Az alkalmazott fejlesztői eszközök

Visual Studio 2022 (C# - Windows Forms Application)

Visual Studio Code (HTML – JS – CSS – PHP)

phpmyadmin (MySQL adatbázis kezelése)

XAMPP (Apache, MySQL webszerver)

Adobe Photoshop (Logók, kisképek szerkesztése)

- DataTables ([forrás](#))
- Chart.JS ([forrás](#))
- FontAwesome ([forrás](#))
- Bootstrap ([forrás](#))
- SB Admin2 Bootstrap Panel ([forrás](#))
- AES 256 encryption for PHP and C# ([forrás](#))

Adatmodell leírása

A projektemben a MySQL-t használtam mint adatbázis, ezen belül is a MariaDB adatbázis motort.

- Az adatbázis 2 táblából áll, egy users és egy vehicles táblából.

- Az users táblában a felhasználói adatok kerülnek mentésre, mint például felhasználónév, jelszó (titkosítva), email cím stb...
- A vehicles táblában a járművek adatai kerülnek, például szín, motor típusa, futásadatok, évjárat stb...

vms_lethaldevs users	vms_lethaldevs vehicles
id : int(11)	vehid : int(11)
username : varchar(50)	owner : varchar(255)
password : varchar(255)	type : varchar(255)
name : varchar(255)	# prodyear : int(11)
# admin : int(1)	vin : varchar(17)
birthdate : date	license : varchar(7)
address : varchar(120)	color : varchar(15)
regdate : datetime	mileage : varchar(7)
lastvisit : datetime	engine : varchar(255)
phonenummer : varchar(255)	# ccm : int(5)
email : varchar(255)	fueltype : varchar(255)
	added : datetime
	# active : tinyint(1)

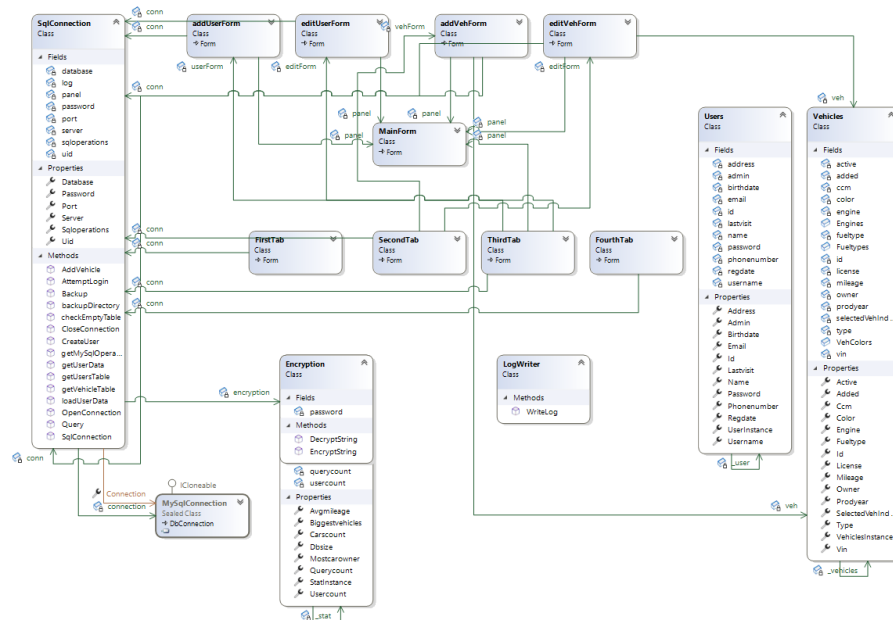
16. SQL struktúra

- A users táblában az id az elsődleges kulcs, az admin alapbeállítása 0 azaz nem admin, a regdate a jelenlegi dátum.
- A vehicles táblában a vehid az elsődleges kulcs, az added mező szintén a jelenlegi dátummal töltődik fel alapértelmezetten, és az active pedig 1.

A C# asztali alkalmazás OOP (Objektumorientált program) tehát készült hozzá egy UML diagram, az osztályokon keresztül változókat tud lekérni a program, majd azokkal a változókkal feltölt egy listát. Így tud a program adatokat lekérdezni a MySQL szerverről, persze előtte a csatlakozást végrehajtja a MySqlConnection osztály. Ezt az osztályt nem szükségeltetik példányosítani, mivel statikus osztály, tehát bárholnan elérjük az eredeti példányát annak. A program kezelőfelületének felépítéséhez is létre kellett hozni egy osztályt, erre azért volt szükség, hogyha megjelenítünk egy felületet, akkor arra egyként tudunk hivatkozni, és ne hozza a program létre minden alkalommal azt az ablakot, ha egy gombbal megjelenítjük azt, hanem azt a konkrét ablakot nyissa meg amit az osztályban definiáltunk. Az alapvető MySQL lekérdezéssel kapcsolatos metódusok azok a MySqlConnection osztályban vannak, ezzel egyszerűbbé és átláthatóbbá téve a kódot, azaz ne legyen szükség oda-vissza járni a sok fájl között keresve, hogy ez a metódus mit és hogyan csinál.

A kezelőfelület (form) -ok egy bázis felületre kerülnek rá, amire helyezve van egy Panel. Erre a panelre jeleníti meg a program a Controlokat, azaz a mi esetünkben az adott menüket. Minden menü egy külön Form amelyet a gombra kattintva hoz elő a program, az alap Form amit a program induláskor betölt az a FirstTab.

A programhoz készült egy UML Diagramm is ami alább látható:



17. UML diagram

3.2 Részletes feladatspecifikáció, algoritmusok

A weboldalon a Backend PHP motoron fut, ahol törekedtem az objektumorientáltság kialakítására, ezért például a MySQL szerverhez való csatlakozás is ilyen módon lett kialakítva.

```
<?php      You, 4 days ago • Uploading Web Files ...
/* Database credentials. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'vms_lethaldevs');

/* Attempt to connect to MySQL database */
$mysqli = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if($mysqli === false){
    die("ERROR: Could not connect. " . $mysqli->connect_error);
}
```

18. PHP MySQL csatlakozás

- Az oldal Bootstrap 5 kialakítású, ehhez használtam az SB Admin2 névre hallgató témát, melyet az igényeknek megfelelően szerkesztettem majd kódoltam hozzá a Backend részét.
- A weboldal egy bejelentkező felülettel fogad, és a böngésző sütijeiben (cookie) tárolja a belépett felhasználót.
- Ha a felhasználó be van jelentkezve, akkor addig belépve is marad amíg ki nem jelentkezik.
- A regisztráció (ha új felhasználók vagyunk) egy űrlap kitöltésével történik, majd, ha az adatokat helyesen töltöttük ki, akkor a PHP a POST methoddal lekéri az adatokat és megvizsgálja azok helyességét.
- Ha minden adat megfelelő, akkor illeszti azokat bele a MySQL adatbázisba.

```
// Check input errors before inserting in database
if(empty($username_err) && empty($password_err) && empty($confirm_password_err)){

    // Prepare an insert statement
    $sql = "INSERT INTO users (username, password, email, phonenumber, name, birthdate, address) VALUES (?, ?, ?, ?, ?, ?, ?)";

    if($stmt = $mysqli->prepare($sql)){
        // Bind variables to the prepared statement as parameters
        $stmt->bind_param("sssssss", $param_username, $param_password, $param_email, $param_phone, $param_fullname, $param_borndate, $param_address);

        // Set parameters
        $param_username = $username;
        $param_password = encrypt($password);
        $param_email = $email;
        $param_phone = $phone;
        $param_fullname = $firstname . ' ' . $lastname;
        $param_borndate = $borndate;

        $streetlastchar = substr(trim($street), -1);

        if ($streetlastchar == '.') {
            $param_address = $postalcode . ' ' . $city . ', ' . $street;
        }
        else {
            $param_address = $postalcode . ' ' . $city . ', ' . $street . '.';
        }

        // Attempt to execute the prepared statement
        if($stmt->execute()){
            // Redirect to login page
            header("location: login.php");
        } else{
            echo "Váratlan hiba történt!";
        }

        // Close statement
        $stmt->close();
    }
}
```

19. PHP Regisztráció kódrészlet

```
// Check if username exists, if yes then verify password
if($stmt->num_rows == 1){
    // Bind result variables
    $stmt->bind_result($id, $username, $hashed_password, $name);
    if($stmt->fetch()){
        if($hashed_password == encrypt($password)){
            // Store data in session variables
            $_SESSION["loggedin"] = true;
            $_SESSION["id"] = $id;
            $_SESSION["username"] = $username;

            $currdate = date('Y-m-d H:i:s');
            $mysqli->query("UPDATE users SET lastvisit = '$currdate' WHERE users.username = '$username'");
            $mysqli->close();

            $_SESSION["realname"] = $name;

            header("location: index.php");
        } else{
            $login_err = "Hibás felhasználónév/jelszó.";
        }
    }
} else{
    $login_err = "Hibás felhasználónév/jelszó.";
}
} else{
    echo "Váratlan hiba történt! Próbálkozz később...";
}

// Close statement
$stmt->close();

// Close connection
$mysqli->close();
}
```

20. PHP bejelentkezés kódrészlet

- A weboldalon a táblázatok egy úgynevezett DataTables keretrendszer alapján jönnek létre, ez egy ingyenesen is felhasználható keretrendszer, van fizetős

verziója, amelyben több komplexebb funkció és kiegészítő található meg, a mi esetünkben ezekre a funkciókra nem volt szükség.

- A DataTables Editorral létre tudunk hozni olyan táblázatokat, amelyeket a keretrendszer kezel, ezáltal tudjuk módosítani őket a kritériumoknak megfelelően.

```
// Build our Editor instance and process the data coming from _POST
Editor::inst( $db, 'vehicles', 'vehid' )
->fields(
    Field::inst( 'owner' ),      You, yesterday * Fixed vehicle creation issues + localization ...
    Field::inst( 'prodyear' )
        ->validator( Validate::notEmpty() )
        ->validator( Validate::minMaxNum( 1000, 2999 ) ),
    Field::inst( 'type' )
        ->validator( Validate::notEmpty() )
        ->validator( Validate::maxLen( 255 ) ),
    Field::inst( 'engine' )
        ->validator( Validate::notEmpty() ),
    Field::inst( 'ccm' )
        ->validator( Validate::notEmpty() )
        ->validator( Validate::minMaxNum( 1, 99999 ) ),
    Field::inst( 'fueltype' )
        ->validator( Validate::notEmpty() ),
    Field::inst( 'color' )
        ->validator( Validate::notEmpty() ),
    Field::inst( 'mileage' )
        ->validator( Validate::notEmpty() )
        ->validator( Validate::maxLen( 7 ) ),
    Field::inst( 'license' )
        ->validator( Validate::notEmpty() )
        ->validator( Validate::minMaxLen( 6, 6 ) ),
    Field::inst( 'vin' )
        ->validator( Validate::notEmpty() )
        ->validator( Validate::minMaxLen( 17, 17 ) )
)
->on( 'preCreate', function ( $editor, &$values ) {
    $editor
        ->field( 'owner' )
        ->setValue( $_SESSION['realname'] );
} )
```

21. PHP DataTables inicializáló

- Amely táblázat nem a keretrendszerrel készült, mint például a főoldalon lévő statisztikai táblázat, azokat egy PHP kód kezeli.

```

<div class="card-body text-break text-uppercase">
  <div class="table-responsive">
    <?php
    $rowcount = 1;
    $conn = mysqli;
    $result = mysqli_query($conn,"SELECT prodyear, type, engine, ccm, owner FROM vehicles ORDER BY ccm DESC LIMIT 10");

    echo "<table class='table'>
    <thead>
    <tr>
    <th>#</th>
    <th>Típus</th>
    <th>Motor</th>
    <th>cm<sup>3</sup></th>
    <th>Tulaj</th>
    </tr>
    </thead>";

    while($row = mysqli_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>" . $rowcount . "</td>";
      echo "<td>" . $row['prodyear'] . " " . $row['type'] . "</td>";
      echo "<td>" . $row['engine'] . "</td>";
      echo "<td>" . $row['ccm'] . "</td>";
      echo "<td>" . $row['owner'] . "</td>";
      echo "</tr>";
      $rowcount++;
    }
    echo "</table>";
  >
</div>
</div>

```

22. PHP tábla kezelés

Az alkalmazásunk C# applikáció, amely a Windows Forms .NET alapon fut. Az alkalmazás a .NET Framework 4.7.2 (LTS) keretrendszerrel dolgozik.

- Mivel itt is a MySQL adatbázissal szeretnénk dolgozni, ezért először annak kell egy csatlakozási metódust definiálni

```

9 references
public SqlConnection()
{
    Server = Properties.Settings.Default.db_ip;
    Database = Properties.Settings.Default.db_database;
    Uid = Properties.Settings.Default.db_uid;
    Password = Properties.Settings.Default.db_password;
    Port = Properties.Settings.Default.db_port;

    string connectionString;
    connectionString = "server=" + Server + ";" + "user=" + Uid + ";" + "database=" +
    Database + ";" + "port=" + Port + ";" + "password=" + Password + ";";

    Connection = new MySqlConnection(connectionString);
}

```

23. C# MySQL csatlakozás

- A csatlakozás adatait a beépített tulajdonságkezelő kezeli, itt létre lehet hozni egy alap beállítást a csatlakozásnak, amelyet a felhasználó majd tud módosítani később.
- Maga a csatlakozás OOP alapokon fekszik, tehát mivel az objektumra több kódsorban is szükség lesz majd, ezért ennek is létre kellett hozni egy különálló osztályt.

- Minden MySQL-el kapcsolatos metódust megpróbáltunk ebbe az osztályba helyezni, hogy együttes legyen a kód, ne legyen szanaszét itt-ott a kódban egy-egy metódus elszórva ami a MySQL-hez tartozik.
- Mivel van pár statisztikai adat amihez az adatbázisból szükséges adatokat kell lekérdezni, ezért ezeket az eljárásokat ebben az osztályban készítettem el.

```
1 reference
private void AverageMiles()
{
    if (conn.checkEmptyTable("vehicles") == false)
    {
        double allmiles = 0;
        double numberofcars = 0;
        double avg = 0;
        var vehicles = conn.getVehicleTable();
        foreach (var vehicle in vehicles)
        {
            numberofcars++;
            allmiles += Convert.ToInt32(vehicle.Mileage);
        }
        avg = allmiles / numberofcars;
        var nfi = (NumberFormatInfo)CultureInfo.InvariantCulture.NumberFormat.Clone();
        nfi.NumberGroupSeparator = " ";
        var formatted = avg.ToString("#,0", nfi); // "1 234 897"

        lblAvgMileage.Text = $"{formatted} km";
    }
}
```

24. C# statisztikai metódus

- A járműveket, az adatbázisban szereplő adatokat egy listába töltjük fel, majd azzal a listával dolgozunk a továbbiakban.

```
7 references
public List<Vehicles> getVehicleTable()
{
    var query = "SELECT * FROM vehicles";
    List<Vehicles> vehicles = new List<Vehicles>();

    if (OpenConnection())
    {
        Sqloperations++;
        MySqlCommand cmd = new MySqlCommand(query, Connection);
        MySqlDataReader dataReader = cmd.ExecuteReader();
        while (dataReader.Read())
        {
            Vehicles vehicle = new Vehicles();
            vehicle.Id = Convert.ToInt32(dataReader["vehid"]);
            vehicle.Owner = (string)dataReader["owner"];
            vehicle.Type = (string)dataReader["type"];
            vehicle.Prodyear = Convert.ToInt32(dataReader["prodyear"]);
            vehicle.Vin = (string)dataReader["vin"];
            vehicle.License = (string)dataReader["license"];
            vehicle.Color = (string)dataReader["color"];
            vehicle.Mileage = (string)dataReader["mileage"];
            vehicle.Engine = (string)dataReader["engine"];
            vehicle.Ccm = Convert.ToInt32(dataReader["ccm"]);
            vehicle.Fueltype = (string)dataReader["fueltype"];
            vehicle.Added = (DateTime)dataReader["added"];
            vehicle.Active = Convert.ToInt32(dataReader["active"]);
            vehicles.Add(vehicle);
        }
        dataReader.Close();
    }
    CloseConnection();
    return vehicles;
}
```

25. C# járművek lekérdezése

- A járműveket egy osztály kezeli, ahová a Getter-Setter használatával a privatizált változókba tudunk adatot felvinni.
- A vehicles objektummal feltöltött lista ezáltal tartalmazni fogja az összes jármű objektumunkat amivel aztán később tudunk majd dolgozni.

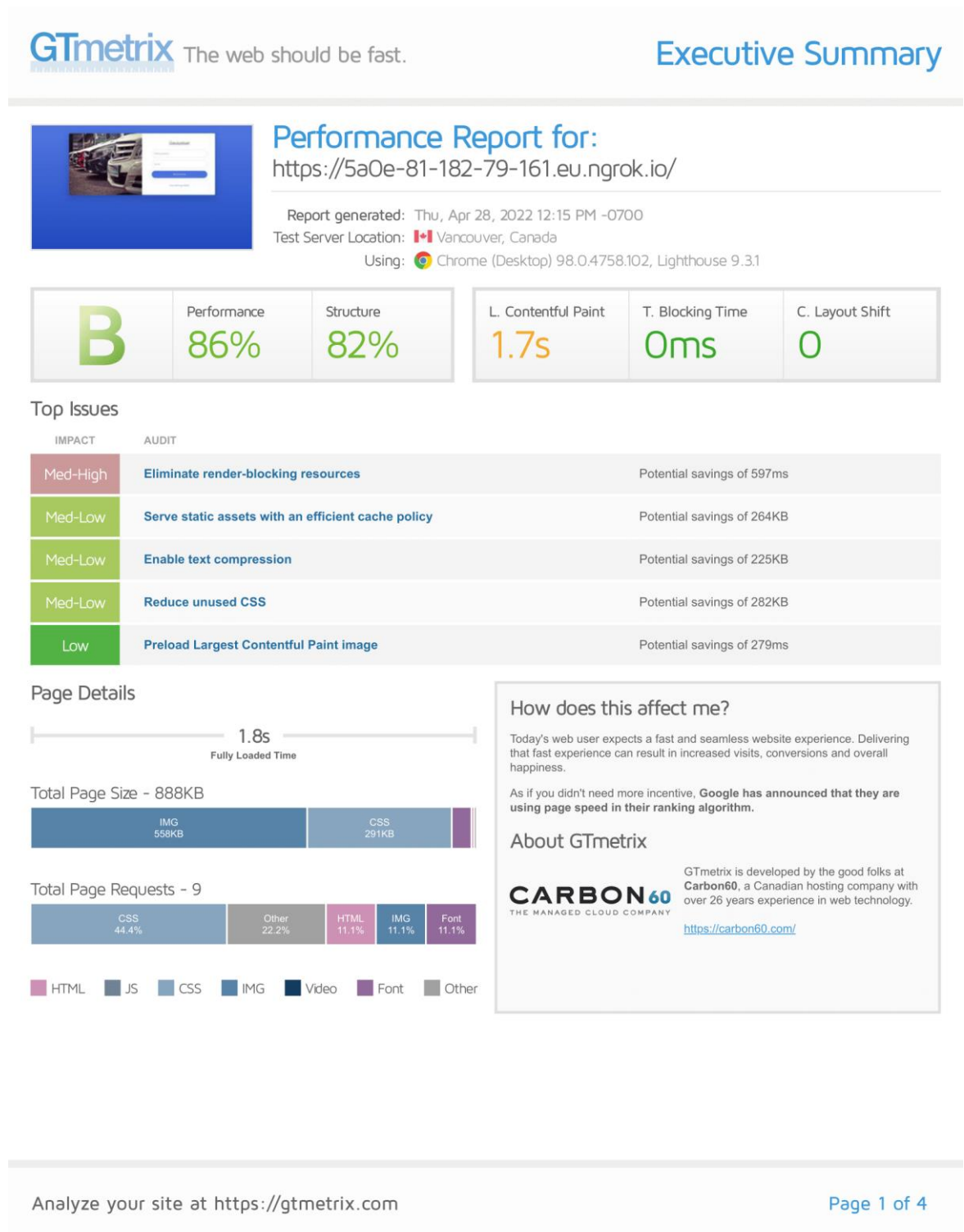
```
8 references
public int Id { get => id; set => id = value; }
7 references
public string Username { get => username; set => username = value; }
2 references
public string Password { get => password; set => password = value; }
9 references
public string Name { get => name; set => name = value; }
3 references
public int Admin { get => admin; set => admin = value; }
6 references
public DateTime Birthdate { get => birthdate; set => birthdate = value; }
6 references
public string Address { get => address; set => address = value; }
6 references
public DateTime Regdate { get => regdate; set => regdate = value; }
1 reference
public DateTime Lastvisit { get => lastvisit; set => lastvisit = value; }
3 references
public string Phonenummer { get => phonenummer; set => phonenummer = value; }
6 references
public string Email { get => email; set => email = value; }
}
```

26. get-set metódusok

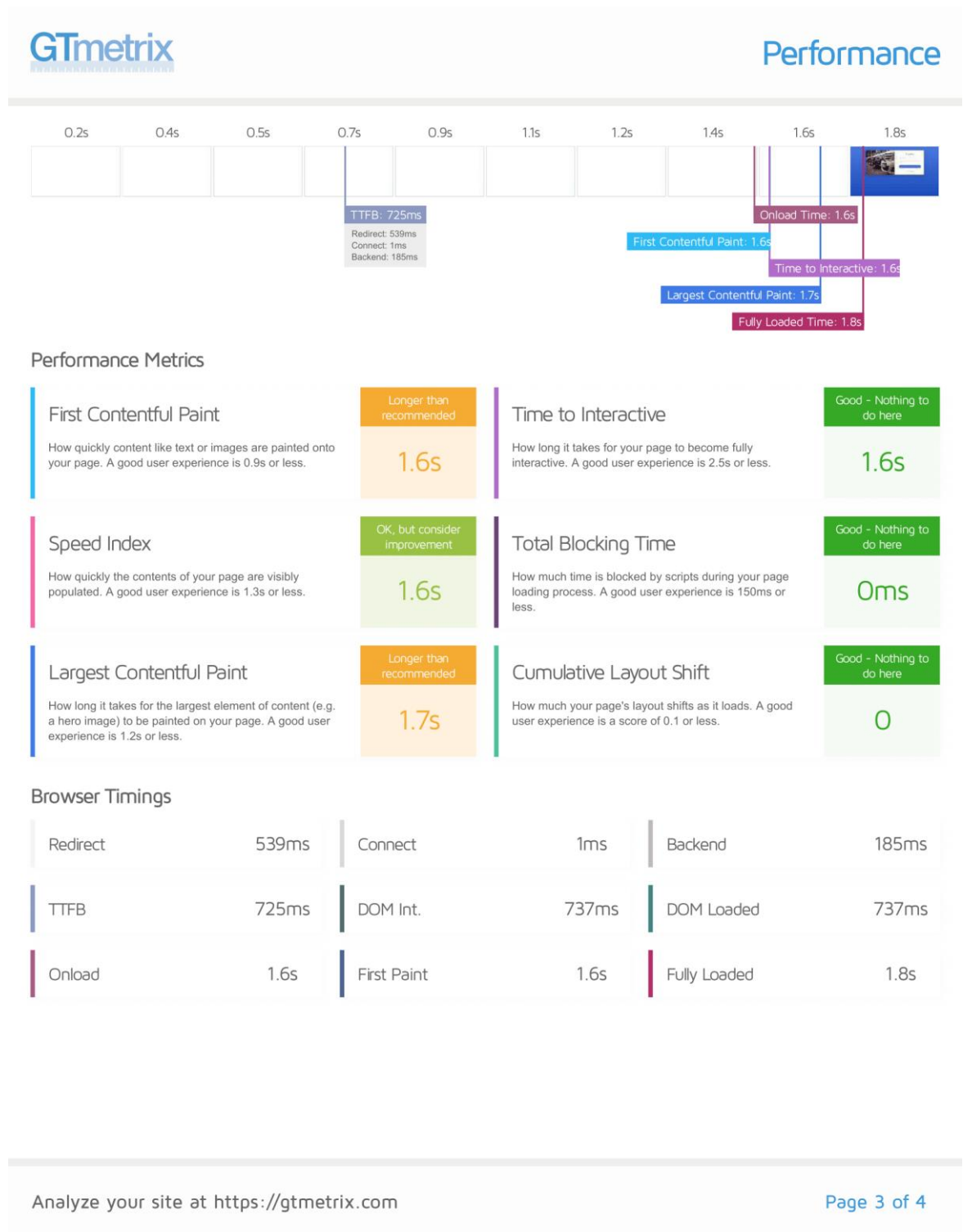
3.3 Tesztelési dokumentáció

1. A weboldal tesztjei:

- A weboldalon futtatva lett egy összesítő teszt, ami minden aspektusát megnézi a weboldalnak, mint hogy mennyi idő szükséges az elemek betöltéséhez, és hogy az adott eszközökről milyen a weboldal megjeleníthetősége.



27. GTMetrix teljesítmény teszt



28. GTMetrix teljesítmény mérések

- Ezen felül egy PHP lekérés teszt is futtatva lett amelynek eredményei a következők:

Filter by

All Requests
Clear Requests

GET /index.php	200 OK	27.05ms
GET /profile.php	200 OK	2.5ms
GET /php/table.vehicles.php	200 OK	12.34ms
GET /table.php	200 OK	1.5ms
GET /profile.php	200 OK	4.31ms
GET /assets/img/favicon-32x32.png	200 OK	0.71ms
GET /assets/img/favicon-16x16.png	200 OK	2.97ms
GET /assets/fonts/Simple-Line-Icons.woff2	200 OK	3.73ms
GET /assets/fonts/fontawesome-webfont.woff2	200 OK	3.56ms
GET /assets/fonts/fa-solid-900.woff2	200 OK	3.89ms
GET /assets/img/avatars/avatar5.jpeg	200 OK	1.67ms
GET /assets/js/theme.js	200 OK	6.25ms
GET /assets/js/bs-init.js	200 OK	5.89ms
GET /js/table.vehicles.js	200 OK	0.5ms
GET /js/dataTables.editor.min.js	200 OK	81.5ms
GET /assets/bootstrap/js/bootstrap.min.js	200 OK	12.71ms
GET /assets/js/jquery.min.js	200 OK	34.09ms
GET /assets/fonts/fontawesome5-overrides.min.css	200 OK	5ms

about 3 hours ago
Duration 27.05ms

GET /index.php

Summary
Headers
Raw
Binary
Replay

200 OK

Summary
Headers
Raw
Binary

15871 bytes text/html; charset=UTF-8

```

<!DOCTYPE html>
<html lang="hu">


<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
  <title>VMS - LethalDevs</title>
  <link rel="apple-touch-icon" type="image/png" sizes="180x180" href="assets/img/favicon-180x180.png">
  <link rel="icon" type="image/png" sizes="16x16" href="assets/img/favicon-16x16.png">
  <link rel="icon" type="image/png" sizes="32x32" href="assets/img/favicon-32x32.png">
  <link rel="icon" type="image/png" sizes="180x180" href="assets/img/favicon-180x180.png">
  <link rel="icon" type="image/png" sizes="192x192" href="assets/img/favicon-192x192.png">
  <link rel="icon" type="image/png" sizes="512x512" href="assets/img/favicon-512x512.png">
  <link rel="stylesheet" href="assets/bootstrap/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i">

```

29. PHP Request teszt

2. Normál tesztet, extrém tesztet (bolondbiztosság tesztelése)

- A weboldal nem enged rossz, illetve üres adatokat felvinni, erre figyelmezteti a felhasználót ezzel „bolondbiztossá” téve azt.



Hozzon létre egy fiókot!

Vezetéknév

Keresztnév

Pethasznáknév

Jelszó

Jelszó újra

Email cím

Év, hó, nap

Telefonszám

Irányítószám

Város

Utca

Regisztrálás

Már van fiókja? Jelentkezzen be.

30. Bolondbiztosság garantálása

3. A tesztelés során számos hiba történt, de ez természetes folyamat, olyan figyelmetlenségek vagy elgépелések melyek a kód lefutását befolyásolják, vagy amelyek nem befolyásolják, csak a program működését tehát a fordító nem fog hibát érzékelni, a program viszont attól még hibásan próbál működni. Ebből számtalan volt, amelyet egy egyszerű Unit teszt-el vizsgáltunk meg, a fejlesztői környezet erre automatizáltan ad lehetőséget és ezáltal tudjuk tesztelni a programunkat.

```
16 references
public bool checkEmptyTable(string table)
{
    var query = "SELECT COUNT(*) from " + table;
    Sqloperations += 1;
    try
    {
        OpenConnection();
        MySqlCommand cmd = new MySqlCommand(query, Connection);
        var result = int.Parse(cmd.ExecuteScalar().ToString());
        return result == 0;
    }
    catch (Exception ex)
    {
        Console.WriteLine("Váratlan hiba történt. " + ex.Message);
        return false;
    }
    finally
    {
        CloseConnection();
    }
}
```

31. Program hibakezelés

Erre ezt a példát hoztam, ez egy hibakezeléses eljárás, amely megnézi hogy a kért táblában van-e adat, azaz üres-e, amely egy true vagy false értékkel tér vissza. Ha a lekérdezés során bármi hiba történne pl. hibás táblát kérünk, akkor ez esetben a program egy hibaüzenetet fog nekünk írni, ha nem kezelnénk ezt az esetet akkor egy automatikus hiba ablak fogadna, tele érthetetlen kifejezésekkel és kódokkal a felhasználó számára.

Összefoglalás

3.4 Önértékelés

A program amelyet készítettem, számomra az első ilyen nagyobb projekt volt amit valaha készítettem, sajnos a saját hibámból kifolyólag későn kezdtem a projekthez, sokáig az ötleten gondolkodva hogy milyen programot is kéne készítenek, majd az ötlet után a projekt megtervezése is elég sok időt emésztett fel, mert az ember nem tudja hogy kezdjen neki, de úgy gondolom hogy ahhoz képest hogy sosem készítettem hasonlót, például Windows Forms-os applikációkkal is csak érintésszerűen találkoztam eddig, de PHP-val meg soha életemben. De ahogy a mondás tartja, manapság minden anyagot megtalál az ember az interneten és ezek segítségével az időhöz és hogy egymagam készítettem el az alkalmazást, viszonylag jól sikerült kivitelezni azt.

Amin mindenképp fejlesztenék, azok a kódok maguk, jók így magukban, de a tiszta kód elvének nem nagyon felelnek meg, eléggé rossz a kommentár rajtuk és az elnevezések sem mindenhol egységesek (valahol angol, valahol magyar). Ezen mindenképp kell még fejlesztenem magam, és ezen felül az objektumorientáltságot is gyakorolnom kell, interfészekkel és származtatott osztályokkal, kevesebb getter-setterrel illetve mysql query-vel is meglehetne oldani egy ilyen feladatot, de mint említettem ez az első alkalom, hogy ilyen mélyen foglalkoztam objektumorientáltsággal, és egy egységes „komplex” programmal.

Ezen felül sajnos az idő hiányában, nem lett tökéletes a program bolondbiztossá tétele sem, azaz ha rossz adatot viszünk fel az admin felületen, akkor azt nem csekkolja le a program, tehát hogy mit írtunk be, számok helyett betűket, vagy hogy például egy telefonszámot helyesen írtunk-e be, ha egy pl. minimum 15 karakteres stringet vár a program, de a felhasználó úgy töltötte ki hogy az kevesebb, akkor is hibás lesz. Erre lehetett volna még körülbelül csak 1 hetet fordítani, hogy tényleg 100%-ig ne lehessen rossz helyre rossz adatokat felvinni, de sajnos erre nem jutott idő, mert amikor elkészültem az asztali alkalmazással, egy hetem volt megcsinálni hozzá a webes responzív részét.

Nagyjából ezeken változtatnék, ha most újból nekikezdenék, de hát ebből tanul az ember és így tud fejlődni.

3.5 Továbbfejlesztési lehetőségek

- Szerettem volna, hogy a felhasználó tudjon egy Excel táblából adatokat felvinni, azaz ne egyesével kelljen felvinni, ha pl. egy cég több száz járművet szeretne nyilvántartásba felvinni, akkor azt meg tudja tenni, hogy egy Excel táblából beimportálja és azok rögzítésre kerülnek.
- A profilkép választást sem tudtam befejezni, ezt a funkciót is úgy terveztem, hogy még benne lesz majd.
- Ha lehetne a járműről képeket vagy képet feltölteni, és azt is megjeleníteni, vizuálisan tudná fejleszteni a programot.
- Mobilra egy másfajta elrendezés lenne az igazi, mert így ha nagyon kicsi az eszközünk akkor nem feltétlen jelenik meg szépen a jármű tábla.
- Az elfelejtett jelszó részére sem jutott idő, eredetileg az volt a terv, hogy meg kell adjuk az email címünket, és arra a címre küldött volna az oldal egy helyreállító linket, amin új jelszót tudtunk volna beállítani.
- Szerettem volna statisztikai adatokat megjeleníteni csak az ügyfél járműveire levetítve, hogy azoknak mennyi az átlagos futásideje, milyen évjáratúak, hány darabot vittünk fel az elmúlt 1 hónapban, határ a képzelet.
- Lehetne bővíteni még a programot egy exportálással is, amely a járműveinket egy Excel táblába exportálja, ezzel aztán tovább dolgozva ha valamely ügyfélnek erre lenne szüksége.

4 Felhasznált irodalom

W3Schools: [C# OOP \(Object-Oriented Programming\) \(w3schools.com\)](https://www.w3schools.com/Csharp/Csharp_OOP.asp)

Stackoverflow: [Access to an object field or property from another class c# - Stack Overflow](https://stackoverflow.com/questions/10408222/access-to-an-object-field-or-property-from-another-class-csharp)

[chart.js - create Chart using Chartjs and PHP - Stack Overflow](https://stackoverflow.com/questions/4481421/chartjs-create-chart-using-chartjs-and-php)

[php - mysqli::prepare\(\): Couldn't fetch MySQL - Stack Overflow](https://stackoverflow.com/questions/4481421/chartjs-create-chart-using-chartjs-and-php)

[mysql - "SELECT email FROM table WHERE name=\\$name"; in PHP - Stack Overflow](https://stackoverflow.com/questions/4481421/chartjs-create-chart-using-chartjs-and-php)

Microsoft C# dokumentáció: [C# docs - get started, tutorials, reference. | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/csharp/)

PHP dokumentáció: [PHP: explode - Manual](https://www.php.net/manual/en/)

AES-256 encryption: [AES-256 encryption and decryption in PHP and C# \(github.com\)](https://github.com/0x00sec/aes256-encryption-decryption-in-php-and-csharp)

TutorialRepublic: [Creating a User Login System with PHP and MySQL - Tutorial Republic](https://www.tutorialrepublic.com/php/tutorial/creating-a-user-login-system-with-php-and-mysql/)

5 Ábrajegyzék

1. XAMPP letöltése	5
2. PHP MySQL konfiguráció.....	6
3. Weboldal bejelentkezés	6
4. MySQL csatlakozás konfiguráció.....	7
5. Admin bejelentkezés	7
6. Weboldal regisztráció	8
7. Weblap főoldal.....	9
8. Járművek adattábla.....	10
9. Asztali alkalmazás belépés	10
10. Asztali alkalmazás kezdőlap	11
11. Járművek listázása	12
12. Jármű szerkesztése	12
13. Felhasználók listázása	13
14. Felhasználó szerkesztése.....	13
15. Szerver beállítások oldal	14
16. SQL struktúra.....	16
17. UML diagram.....	17
18. PHP MySQL csatlakozás	18
19. PHP Regisztráció kódrészlet.....	19
20. PHP bejelentkezés kódrészlet	19
21. PHP DataTables inicializáló	20
22. PHP tábla kezelés.....	21
23. C# MySQL csatlakozás.....	21
24. C# statisztikai metódus	22
25. C# járművek lekérdezése	23
26. get-set metódusok	24
27. GTMetrix teljesítmény teszt	25
28. GTMetrix teljesítmény mérések	26
29. PHP Request teszt	27
30. Bolondbiztonság garantálása	27
31. Program hibakezelés	28