**Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Szakgimnáziuma**

**Az 54 213 05 számú Szoftverfejlesztő szakképesítés záródolgozata**

# **Car Docket**

Készítette:

Marton Zoltán István

Szeged

2020

Tartalomjegyzék kész

# **Bevezetés**

Személyiségemből adódóan nem szeretem a rendetlenséget az életemben és ezzel sokan másik is így vannak, valamint nagy rajongója vagyok az autóknak így adott volt, hogy egy autókkal kapcsolatos adatbáziskezelőt fogok elkészíteni.

A programom összetett megoldást kínál olyan szervezetek számára ahol nagyobb mennyiségű flotta van, de jelenleg Excel táblát vagy egyéb hasonló offline szoftver használnak a nyilvántartásra. Az ilyen offline programok legnagyobb akadálya pont abból adódik, hogy egyszerre csak egy gépen tudják módosítani, nem biztonságos, macerás a mozgatása, amennyiben nem kezeli a hibákat nagyon könnyű tönkre tenni egy-egy rossz adattal.

# **1. Fejlesztői dokumentáció**

## **1.1 Adatbázis tervezés**

A fejlesztés egyértelműen az adatbázis megtervezésével indult. Mivel már tisztában voltam azzal mit kell tudnia egy ilyen programnak ezért bátran kezdtem bele a tervezésbe. Egyedi azonosításhoz minden esetben a MySQL auto increment funkcióját használtam fel. Azzal kezdtem a tervezést, hogy feltérképeztem melyek azok az adatok, amelyeket érdemes nyilvántartani egy autóról és ezeket az adatokat milyen típusú változóban kellene tárolni.

1.táblázat

*Autók tábla*

|  |  |
| --- | --- |
| Adat | Változó |
| id (elsődleges kulcs) | int |
| marka | varchar |
| tipus | varchar |
| gyartasi\_ev | varchar |
| vetelar | int |
| rendszam | varchar |
| kilometeroraallas | int |
| alvazszam | varchar |
| gepkocsi\_tipusa | varchar |
| uzemanyag | varchar |
| sebessegvalto\_tipusa | varchar |
| tulid (másodlagos kulcs) | int |

Az autót tábla tervezésénél figyelnem kellett arra, hogy bár egy-egy autóról nagyon sokféle adatot lehet tárolni, a felhasználónak ne kelljen számára felesleges és a használat szempontjából lényegtelen adatokat felvinni és a későbbiekben kezelni. És persze az adatbázis mérete sem elhanyagolható szempont. Ezeket a szempontokat minden táblánál figyelembe vettem. A megfelelő változók használatára is oda kellett figyelni az autó gyártási éve szándékosan lett szöveges érték, mert a későbbiekben a program működésében erre volt szükségem a megfelelő működéshez. Az autók tábla feltöltését követően következhetett a tulajdonosok tábla megtervezése és feltöltése. A táblában a vételár és a kilóméteróraállás lehet nulla érték a többinél nem engedélyezett.

2.táblázat

*Tulajdonosok tábla*

|  |  |
| --- | --- |
| Adat | Változó |
| tulid (elsődleges kulcs) | int |
| tulajdonos\_nev | varchar |
| tulajdonos\_szemelyiigszam | varchar |
| jogositvany\_azon | int |
| email\_cim | varchar |
| telefonszam | int |
| cegid (másodlagos kulcs) | int |

A tulajdonosokat eltároló táblánál is figyelembe vettem azt, hogy ne tároljunk teljesen felesleges adatokat az autók tulajdonosairól, kizárólag azokat, amelyek szükségesek. Az autók és tulajdonosok tábla feltöltését követően összekapcsoltam őket. Az autók táblában csak a tulajdonos azonosítóját (tulid) tároltam el. Egy autónak csak egy tulajdonosa lehet, de egy tulajdonosnak több autója is. A tulajdonosok táblában egyáltalán nincs engedélyezve nulla érték egyik mezőben sem.

3.táblázat

*Cégek tábla*

|  |  |
| --- | --- |
| Adat | Változó |
| cegid (elsődleges kulcs) | int |
| cegnev | varchar |
| adoszam | int |
| varos | varchar |
| utca | szam |
| szam | int |
| ceg\_email\_cim | varchar |

Ennél a táblánál is kellő odafigyelést fordítottam arra, hogy csak a legszükségesebb adatokat tároljam adatbázisban. Ebben a táblában sincs engedélyezve nulla érték egyik mezőben sem. Ebben a táblában már nincsen másodlagos kulcs. Miután feltöltöttem a tulajdonosok és a cégek táblát ezután a cégek táblában azonosításra használt értékkel (cegid) összekötöttem a két táblát.

4.táblázat

*Felhasználók tábla*

|  |  |
| --- | --- |
| Adat | Változó |
| id | int |
| felhasznalonev | varchar |
| jelszo | varchar |
| emailcimfelhasznalo | varchar |

Ez a tábla egy végtelenül egyszerű felépítéssel rendelkezik. A felhasználókat tárolja akik rendelkeznek hozzáféréssel a webes és asztali alkalmazáshoz. Későbbiekben írni fogok a felhasználók kezeléséről is. Ebben a táblában szintén nem engedélyezett egyik mezőben sem a nulla, mint érték. A táblában csak manuálisan adható hozzá két egyező felhasználónévvel rendelkező egyén. A weboldalon erre nincs lehetőség mert a ellenőrzés le fut a regisztrációnál arra is hogy létezik-e már ilyen nevű felhasználó az adatbázisban.

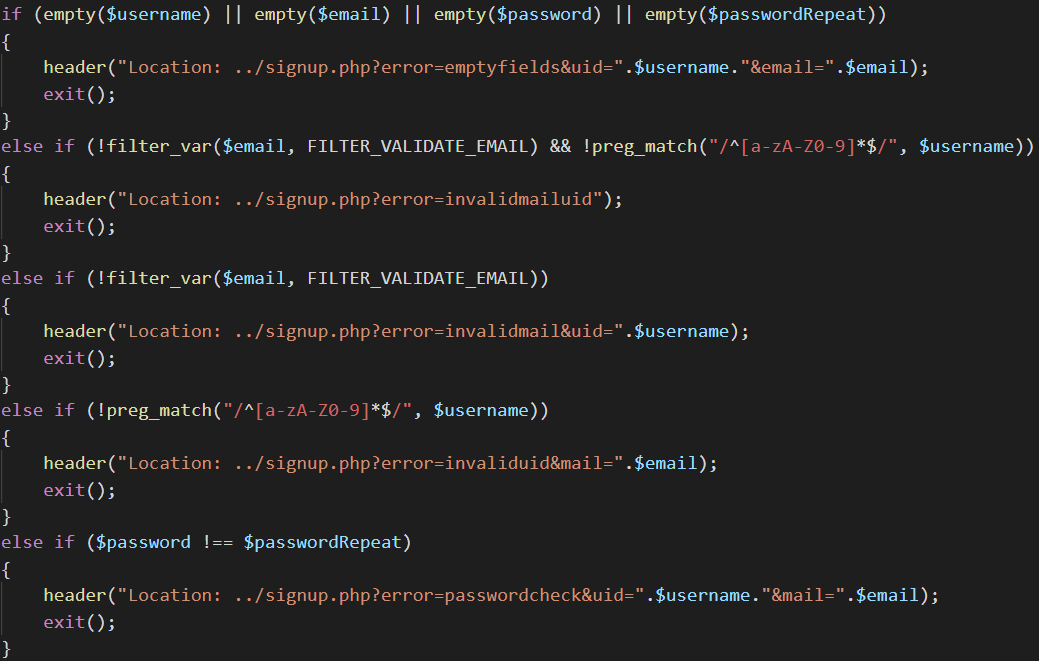
## **1.2 Weboldal fejlesztés**

A weboldal tervezése során szem előtt kellett tartanom a legfontosabb tulajdonságokat, amelyekkel rendelkezni kell egy ilyen oldalnak. A teljesség igénye nélkül felsorolnék párat. Először is nagyon fontos az átlátható és könnyen tanulható oldal, amelyen a felhasználó otthon érzi magát és nem ül percekig a monitor előtt azon gondolkozva, hogy tulajdonképpen ahhoz, hogy elérjen egy funkciót, mit is kellene megnyomnia, vagy mire kellene kattintania. Másodszor a mai világban nagyon fontos a reszponzivitás, az hogy a weboldal több méretben és telefonon is megfelelően és természetesen ugyanúgy használhatóan jelenjen meg alapfeltétel volt. Többek között emiatt és az egységes dizájn miatt a Bootstrap 4 segítségével terveztem meg az egész oldalt. Legelőször megterveztem a felépítését az oldalnak, hogy nagyjából hogyan is szeretném elhelyezni a menüpontokat, milyen legyen az egységes design. Ezek után kezdtem bele annak a meghatározásába, hogy milyen funkciókat szeretnék a weboldalba implementálni. A főoldal tetején megtalálható a menü valamint a bejelentkező felület.

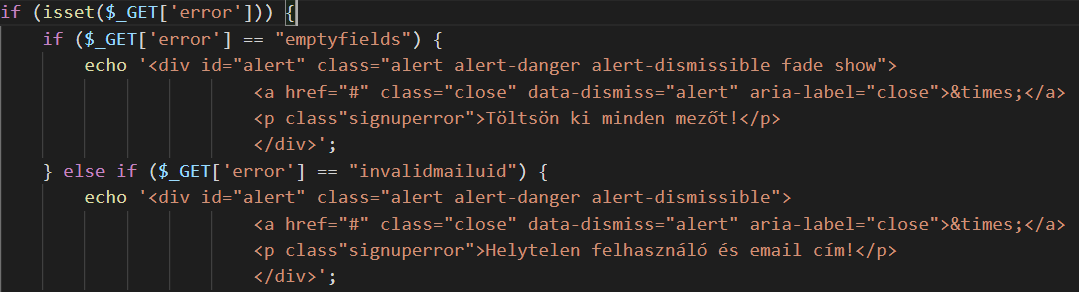
### **1.2.1 Regisztráció**

A regisztráció során, ha a felhasználó helyesen adja meg a kért adatokat, abban az esetben hozzá adódik a rendszerhez az új felhasználó és az ő általa megadott adatok feltöltődnek a regisztrációkor az adatbázis előre megadott és erre a célra létrehozott táblájába a signup.inc nevű PHP fájl segítségével, amely lefut a „Regisztrálok” gomb lenyomására. A regisztrációs oldalt, amelyen a kitöltendő beviteli mezők találhatóak szintén PHP nyelvben írtam meg. A regisztráció csak abban az esetben lehet sikeres, ha a felhasználó a megadott feltételeket betartja, amikor kitölti az űrlapot. Jelen esetben két helyen is vizsgálom a beírt adatokat. Először kliens oldalon vizsgálom a bemeneti adatok helyességét valamint azt, hogy a felhasználó mindent oda írjon, ahova rendeltetésszerűen kell. Ebben az esetben, ha a felhasználó nem helyes adatot ad akkor egy figyelmeztető üzenet jelenik, meg amely tájékoztatja a felhasználót arról, hogy a bevitt adat hibás vagy adott esetben üres a mező. Másodjára szerver oldalon is vizsgálat alá vonom a bevitt adatokat még azelőtt, hogy feltöltődtek volna az adatbázisba. Legelőször az az ellenőrzés fut le, amely azt vizsgálja, hogy a beviteli mezők üresek-e. A felhasználónév esetében RegEx-et használtam annak ellenőrzésére, hogy a felhasználónév megfelel-e a követelményeknek. A bevitt email címet a PHP beépített email ellenőrzésével ellenőriztem le. A jelszó esetén az ellenőrzés azt vizsgálja, hogy a megadott jelszavak megegyeznek-e vagy sem. Amennyiben bármelyik vizsgálat szerver oldalon hibát ad vissza, abban az esetben header php parancs segítségével visszairányítom a felhasználót a signup nevű php fájlra és az url-be kiiratom hogy hol is futott hibára az lefutása alatt valamint kiléptetem a signup.inc nevű php fájl lefutásából egy exit() használatával. Ennek a PHP kódja látható az alábbi képen.

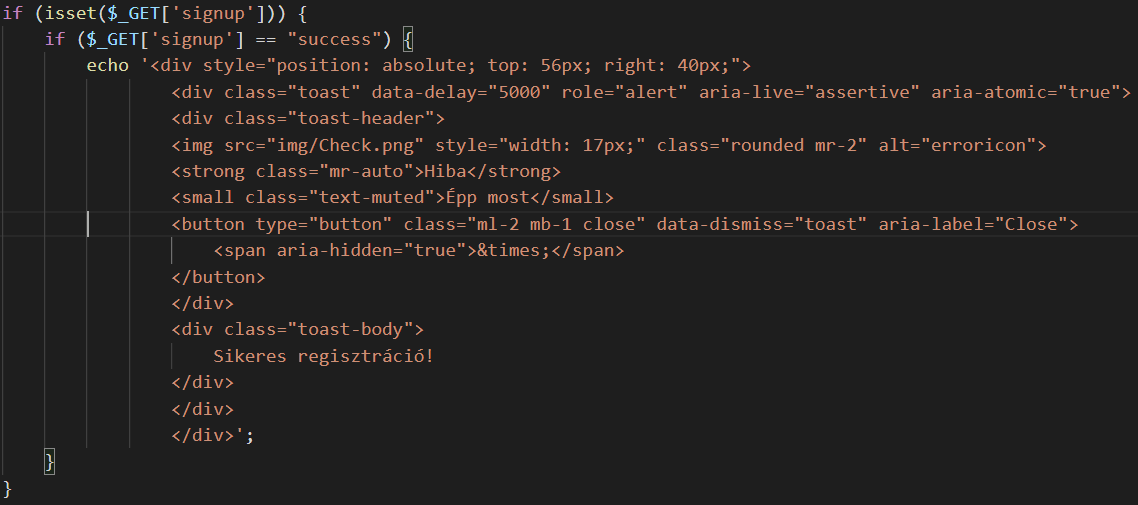
1.sz kép



Hibás adatok bevitelekor a felhasználó több féle hiba üzenetet is kaphat, attól függően, hogy milyen hiba adódott. Többek között emiatt is szükség van a headerrel való átirányításokra, mert az átirányított url tartalmát vizsgálom a signup.php-ban és ezért mindig a helyes hibaüzenet jelenik meg. A $\_GET-el vizsgálom meg az átirányított url kérdőjel utáni tartalmát és ezalapján lefut a helyes echo és megjelenik a helyzetnek megfelelő hibaüzenet. Az alábbi kód részletben kettő darab ilyen kiírás látható.



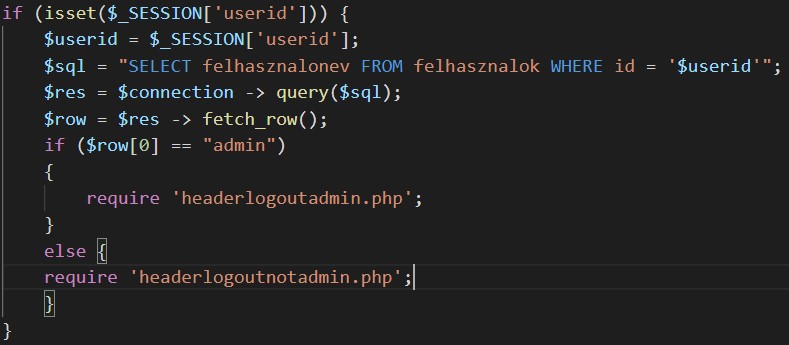
Amennyiben a regisztráció sikeres egy boostrapes toast ablak jelenik meg közvetlen a bejelentkezés után. Ezért a megjelenítésért felelős kódot itt láthatják.



A felhasználónevet RegEx-el ellenőriztem le, csak betűt és számot tartalmazhat különben hibát fog dobni a felhasználónak. Email címnél a fentebb említett beépített ellenőrző metódust használom.

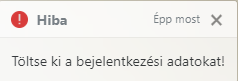
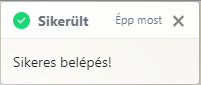
### **1.2.2 Bejelentkezés**

Sikeres bejelentkezés utána felhasználónak felóldódnak azok a funkciók, amelyek megtekintéséhez rendelkezik a kellő jogosultsággal. Kettő darab felhasználói szintet hoztam létre a weboldalon jelenleg ennyi érhető el. Ennek a két szintnek az elkülönítését úgy oldottam meg, hogy a weboldal kettő fajta header tud megjeleníteni belépés után. Amennyiben a felhasználó userid egyenlő adminnal, azaz a felhasználó neve admin (korlátozva van, mivel nem jöhet létre kettő ugyanolyan nevű profil ezért maximum manuálisan az adatbázis szerkesztésével lehet több admin nevű felhasználó, a weboldal nem enged többet regisztrálni). Ezt az alábbi kóddal oldottam meg.



A fenti kódrészlet először is megnézi, hogy van e userid a jelenlegi session-ben, amennyiben van, az azt jelenti, hogy biztos, hogy a felhasználó bejelentkezett. Következő lényeges lépés a lekérés maga, amely lekéri a belépett felhasználó nevét.

Sikeres belépés esetén a felhasználónak megjelenik egy kis üzenet, ami arról tájékoztatja, hogy a belépés sikeres volt vagy sikertelen.



Amennyiben a név egyenlő adminnal, abban az esetben az a headerlogoutadmin.php tölt be. Ezen megjelenítésre kerül a felhasználók kezelése fül, amelyen keresztül elérhetjük a már regisztrált felhasználók adatait.



Amennyiben a felhasználó neve nem egyenlő azzal, hogy admin abban az esetben a másodlagos headerlogoutnotadmin.php fog betölteni.

