# Hallgatói nyilatkozat

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra. Szakdolgozatunk a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium Szoftverfejlesztő és tesztelő technikus képzésén készítettük. Tudomásul vesszük, hogy szakdolgozatunkat a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium tárolja.

Név1 Név2

Név3

# Konzultációs lap

Vizsgázók neve:

Szakdolgozat címe:

Program által nyújtott szolgáltatások:

* …
* …

|  |  |  |
| --- | --- | --- |
| Sorszám | A konzultáció időpontja | A konzulens aláírása |
| 1. | 2024. október 11. |  |
| 2. | 2024. november 15. |  |
| 3. | 2024. december 13. |  |
| 4. | 2025. január 17. |  |
| 5. | 2025. február 14. |  |
| 6. | 2025. március 14. |  |

A szakdolgozat beadható: A szakdolgozatot átvettem:

Vác, 2025. Vác, 2025.

Konzulens A szakképzést folytató

intézmény felelőse

# Tartalomjegyzék

[Hallgatói nyilatkozat 3](#_Toc193361708)

[Konzultációs lap 4](#_Toc193361709)

[Tartalomjegyzék 5](#_Toc193361710)

[Témaválasztás 6](#_Toc193361711)

[Feladatspecifikáció 7](#_Toc193361712)

[Célok 7](#_Toc193361713)

[Nem funkcionális követelmények 7](#_Toc193361714)

[Funkcionális követelmények 8](#_Toc193361715)

[Munkamegosztás 10](#_Toc193361716)

[I. Adatbázis struktúra és backend 10](#_Toc193361717)

[II. Asztali alkalmazás fejlesztése 10](#_Toc193361718)

[III. Webes felület fejlesztése 11](#_Toc193361719)

[IV. Integráció és végleges tesztelés 12](#_Toc193361720)

[Fejlesztői dokumentáció 13](#_Toc193361721)

[Fejlesztői környezet 13](#_Toc193361722)

[Használt programozási nyelvek, technológiák, könyvtárak 13](#_Toc193361723)

[Adatbázis 16](#_Toc193361724)

[Jogosultságok, jogosultsági szintek 30](#_Toc193361725)

[Webalkalmazás 31](#_Toc193361726)

[Felhasználói dokumentáció 43](#_Toc193361727)

[Webalkalmazás 43](#_Toc193361728)

# Témaválasztás

Miért ezt a témát választottuk?

# Feladatspecifikáció

Egyes autószervizeknél sok információt és adatot papíron kezelnek, ezeknek a tárolása, rendszerezése nagy mennyiség esetén eléggé nehézkessé válhat. Összekeverhetik az ügyfeleket, autóikat, a raktár készletet sem tudják így nyilván tartani, ha nagy tételekről van szó.

Esetleg weboldallal sem rendelkeznek. Így nehezen tudnak új ügyfeleket fogadni, ha nincsenek jelen az interneten. Telefonálással esetleg bejelentkezhetnek hozzájuk, illetve szájhagyomány útján terjedve szerezhetnek a műhelyükről tudomást, vagy ha elmennek az üzletük előtt.

## Célok

Elsődleges célunk az autóműhelyek működésének elősegítése, hogy könnyen tudják nyilvántartani az éppen aktuális műhelyi feladatokat, készleteiket, és ügyfeleiket. A szerelők munkájában is el kell a segítség, ezt a hibakódok egyszerűen értelmezhető megtekintése, és minden autó típushoz illő megfelelő javítási útmutató, sablon megtalálása tenné lehetővé.

Az ügyfelek számára egy könnyedén használható weboldal lenne a cél, hogy autóikról minden fontos információt a tökéletesség igénye nélkül nyomon követhessenek, illetve minden autóra az éppen aktív szolgáltatásokat tekinthessék meg, foglalhassanak újat, előző javításokról is tájékozódhassanak.

## Nem funkcionális követelmények

A felhasználók adatai titkosítva vannak, a jelszót SHA 256-os algoritmussal titkosítjuk, így esetleges adatszivárgás esetén illetéktelen felhasználó nem férhet hozzá más fiókjához. Az Adminisztrátor, Szerelő és Ügyfél szereppel rendelkező felhasználóknak különböző jogosultságaik vannak, így nem mindenki fér hozzá mindenhez

A web- és asztali alkalmazás telepíthetősége könnyű, nem időigényes folyamat, így kezdő felhasználók számára is egyszerű ez a műveletsor.

## Funkcionális követelmények

### Webes felület

A felhasználók teljes nevük, felhasználónevük, email címük, jelszavuk kétszeri, és telefonszámuk opcionális megadása után tudnak regisztrálni az oldalunkra. A regisztrációt követően automatikusan bejelentkezteti őket a rendszer. Meglévő felhasználók felhasználónevük és jelszavuk megadása után jelentkezhetnek be. Az adataik módosítása során a felhasználónéven kívül mindent megváltoztathatnak.

Egy felhasználó új autót hozhat létre még nem az adatbázisban lévő rendszámmal, ezt inaktív szolgáltatások ellenében törölheti is, illetve módosíthatja annak rendszámát. Az autójának fontos adatait részletesen megtekintheti a felületen, ilyenek annak márkája és típusa, rendszáma, aktív szolgáltatásai, illetve az előző javítások.

Új szolgáltatásokat is lehet foglalni, miután kiválasztottuk a megfelelő járművet. Ezeket akár később le is mondhatjuk abban az esetben, ha az állapot még foglalt, tehát még nincs szervízben az autó.

### Asztali felület

* **Ügyfélnyilvántartás:**
  + Ügyféladatok tárolása (név, elérhetőség, autó adatai, előző javítások története).
  + Járművek adatai: rendszám, típus, évjárat, motor és futómű specifikációk, korábbi javítások és karbantartások nyilvántartása.
  + Szervizcsomagok és előre fizetett szolgáltatások kezelése.
* **Alkatrészkészlet Kezelés:**
  + Raktárkészlet nyomon követése: alkatrészek hozzáadása, kivonása, készletszint riasztások.
  + Alkatrész rendelések kezelése és követése.
* **Szerelési Segédrendszer:**
  + Szerelési útmutatók és leírások integrálása a rendszerbe, amelyeket a szerelők közvetlenül elérhetnek. (repair manuals)
  + Hibakód-adatbázis (pl. OBD II kódok jelentése) a gyors diagnosztikához.
  + Munkafolyamat-sablonok az általános javítási munkákhoz (pl. olajcsere, bizonyos alkatrészek csereéje, azoknak helye az autóban).

# Munkamegosztás

## I. Adatbázis struktúra és backend

* Adatbázis tervezése és építése
* Fejlesztés: Fenyvesi Ákos
* Tesztelés: Chrén András
* Dokumentáció: Fenyvesi Ákos, Chrén András
* Határidő: 2024. 09. 29

## II. Asztali alkalmazás fejlesztése

### Ügyfélnyilvántartás

* Fejlesztés: Feynvesi Ákos
* Tesztelés: Szenográdi Márk Kornél
* Dokumentáció: Feynvesi Ákos
* Határidő: 2024. 10. 27.

### Járművek adatai és előző javítások

* Fejlesztés: Chrén András
* Tesztelés: Szenográdi Márk Kornél
* Dokumentáció: Chrén András
* Határidő: 2024. 11. 24.

### Szervizcsomagok kezelése

* Fejlesztés: Chrén András
* Tesztelés: Fenyvesi Ákos
* Dokumentáció: Chrén András
* Határidő: 2024. 12. 15

### Alkatrészkészlet kezelés

* Fejlesztés: Szenográdi Márk Kornél
* Tesztelés: Chrén András
* Dokumentáció: Szenográdi Márk Kornél
* Határidő: 2025. 01. 12

### Szerelési segédrendszer (útmutatók és hibakódok)

* Fejlesztés: Fenyvesi Ákos
* Tesztelés: Szenográdi Márk Kornél
* Dokumentáció: Fenyvesi Ákos
* Határidő: 2025. 02. 16

## III. Webes felület fejlesztése

### Időpontfoglalás modul (ügyfelek számára)

* Fejlesztés: Szenográdi Márk Kornél
* Tesztelés: Szenográdi Márk Kornél
* Dokumentáció: Szenográdi Márk Kornél
* Határidő: 2024. 10. 27.

### Szolgáltatások és árak megjelenítése

* Fejlesztés: Szenográdi Márk Kornél
* Tesztelés: Fenyvesi Ákos
* Dokumentáció: Szenográdi Márk Kornél
* Határidő: 2024. 11. 10.

### Jármű javítási státusz követése (ügyfelek számára)

* Fejlesztés: Szenográdi Márk Kornél
* Tesztelés: Chrén András
* Dokumentáció: Szenográdi Márk Kornél
* Határidő: 2024. 12. 01.

## IV. Integráció és végleges tesztelés

### Webes és asztali rendszer integrációja az adatbázissal

* Felelős: Fenyvesi Ákos
* Határidő: 2025.02.28
* Tesztelés: Fenyvesi Ákos, Chrén András, Szenográdi Márk Kornél

### Rendszertesztelés és hibajavítás

* Felelős: Szenográdi Márk Kornél
* Határidő: 2025. 03. 07.

# Fejlesztői dokumentáció

## Fejlesztői környezet

### Visual Studio Code

A Visual Studio Code (vagy VSCode, illetve VS Code) egy Microsoft által fejlesztett nyílt forráskódú kódszerkesztő. Kiegészítő könyvtárral rendelkezik, ahonnan már megírt bővítményeket tölthetünk le és használhatunk, amelyek megkönnyítik a fejlesztők munkáját például intelligens kódkiegészítéssel, vagy akár szebbé teszik a felületet.

### Firefox Developer Edition

A Firefox Developer Edition a Mozilla által fejlesztett ismert Firefox böngésző weboldalak készítésére szánt változata, amely fejlesztői eszközökkel rendelkezik, így megkönnyítve a honlapok készítésének folyamatát. Ezen eszközök segítségével például egyszerűen láthatjuk magunk előtt vizuálisan is az oldalon található rácsok elrendezését, ha az ennek megfelelő eszközt használjuk.

### XAMPP

A XAMPP egy nyílt forráskódú webszerver-szoftvercsomag, amely platformfüggetlen, így Windows, Linux és macOS környezetben is könnyedén használható. Fontos alkotóelemei közé tartozik az Apache webszerver, a MariaDB adatbázis-kezelő, valamint a PHP programozási nyelv értelmezője, végrehajtó rendszere. Általában weboldalak és webes alkalmazások készítéséhez előszeretettel használják ezt a fejlesztők.

## Használt programozási nyelvek, technológiák, könyvtárak

### HTML5

A HTML5 a Hypertext Markup Language legújabb, 5.2-es változata, amely a web fő hiperszöveges jelölőnyelve. Ez egy leíró nyelv, weboldalak készítéséhez fejlesztették ki, ez adja az oldalak “csontvázát”, alapját. Ez írja le a megjelenítő programnak, adott esetben egy böngészőnek, hogy hogyan is kellene feldolgozni és megjeleníteni az állomány tartalmát, hogy egy eredményes weboldalt kapjunk.

### CSS

A CSS vagy Cascading Style Sheets egy stílusleíró nyelv, ez írja le a HTML típusú strukturált dokumentumok megjelenítését. Ennek segítségével lehet állítani, módosítani a lapok színét, betűtípusát, elrendezését, és kinézetének minden egyes elemét. Létrehozásának egyik fő oka és célja volt, hogy az oldal struktúrájának leírását elkülönítsék a megjelenítésétől.

### JavaScript

A JavaScript egy objektumorientált programozási nyelv, melyet a weboldalak interaktív és dinamikus viselkedésének fejlesztéséhez használnak. Ennek segítségével reagálnak az oldal elemei a felhasználói interakciókra. Ezen kívül készíthetőek a nyelven alapuló webes játékok és szoftverek is.

### JQuery

A jQuery egy igen népszerű JavaScript könyvtár, amely kényelmes kommunikációt biztosít a weblap elemeivel eseményvezérlők és CSS szelektorok használatával. A dinamikus AJAX kérések küldésére ez a legegyszerűbb megoldás, illetve egyszerű CSS animációkat is létrehozhatunk ennek segítségével.

### Ajax

Az Ajax-ot (vagy Asynchronous JavaScript and XML) interaktív webalkalmazások létrehozásánál előszeretettel használják. Ennek segítségével weblapok újratöltése nélkül kis adacsere mennyiséggel kommunikálhatunk a szerverrel, így növelhetjük a honlap interaktivitását, sebességét és használhatóságát.

### PHP

A PHP (vagy PHP: Hypertext Preprocessor) egy általános szerveroldali szkriptnyelv, melyet dinamikus weboldalak készítésénél használnak. A kód HTML oldalba ágyazható, és a webszerver PHP feldolgozómodulja értelmezi, ezáltal dinamikus weboldalt hozva létre, azonban a kódot nem továbbítja a szerver az ügyfélnek. Ezen nyelv és a MySQL segítségével könnyedén kommunikálhatunk a szerveren található adatbázissal.

### MySQL

A MySQL egy SQL-alapú relációs adatbázis-kezelő szerver. Ennek segítségével tárolja el a szerveren található kód az információkat, legyenek azok a felhasználók bejelentkezési, regisztrációs adatai, és egyéb, a weblap további működéséhez szükséges adatok. XAMPP és a phpMyAdmin segítségével egy admin kezelőfelületet kapott, így itt könnyen tudunk adatbázisokat és táblákat létrehozni illetve kezelni.

### JSON Web Token / JWT

A JWT (vagy JSON Web Token) titkosítás segítségével tárolja az éppen bejelentkezett felhasználó adatait. Ezt a szerver generálja le a bejelentkezés során, SHA 256 segítségével titkosítja, és ezt küldi a kliensnek. Sokkal biztosabb megoldás, mintha a fontos adatokat egyszerű változókban tárolnánk.

### Composer

A Composer egy alkalmazás szintű PHP nyelvhez készített függőség és könyvtárkezelő. Ennek segítségével könnyedén tudunk mások által készített könyvtárakat illetve a szerverünk működését elősegítő praktikákat használni.

### CodeSnap

Ezzel a VSC kiegészítővel egyszerűen készíthetünk képernyőképeket a kódunkról. Kijelöljük a megörökíteni kívánt kódrészletet, majd egy jobb klikk segítségével elindítjuk a CodeSnap kiegészítőt, beállítjuk a megfelelő méretet, majd a kép feletti logóval elmentjük az elkészült képet a meghajtónkra.

## Adatbázis

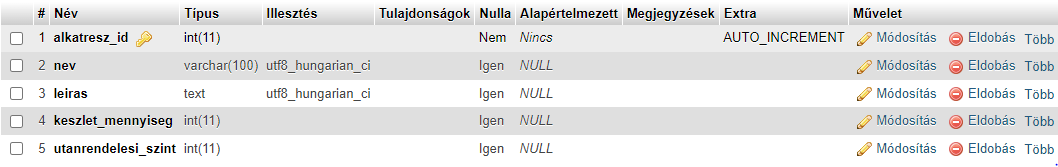
Adatbázis szerver: MariaDB 10.4.32 kiszolgáló verzió  
Adatbázis neve: automuhely  
Illesztés: utf8mb4\_general\_ci  
Használt adatbázis motor: InnoDB

Adatbázist létrehozó SQL parancs:

CREATE DATABASE IF NOT EXISTS automuhely DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

### Az „alkatreszek” tábla

Ez a tábla tartalmazza az alkatrész fajtákat.



*alkatresz\_id:* Az alkatrész azonosítója, tábla elsődleges kulcsa

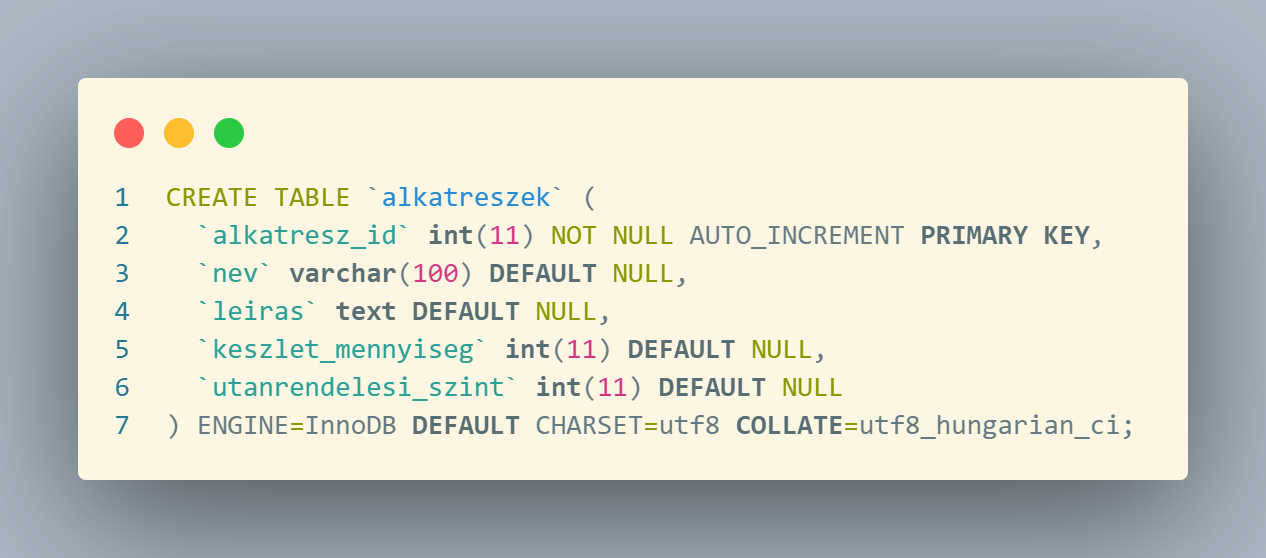
*nev:* Az alkatrészek elnevezése

leiras: Alkatrészek részletes leírása

keszlet\_mennyiseg: Készleten levők száma az adott alkatrészből

utanrendelesi\_szint: Alkatrészek utánrendelési mennyisége.

Tábla létrehozása:



### A „felhasznalok” tábla

Ez a tábla tartalmazza a felhasználók adatait



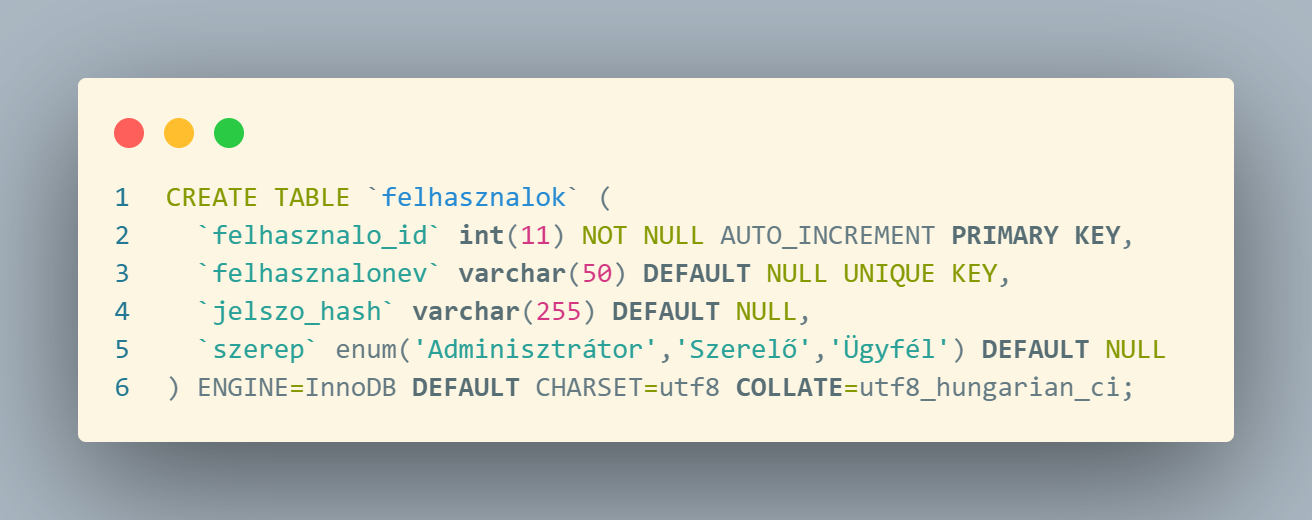
*felhasznalo\_id:* A felhasználók azonosítója, tábla elsődleges kulcsa

*felhasznalonev:* A felhasználók bejelentekezési neve, a tábla egyedi kulcsa

jelszo\_hash: felhasználók titkosított jelszava

szerep: (enum) a felhasználó jogosultságát beállító paraméter

Tábla létrehozása:



### A „felhasznalok\_ugyfelek” tábla

Ez a tábla kapcsolja össze a felhasználók és ügyfelek táblákat



*felhasznalo\_id:* a felhasználók azonosítója, tábla elsődleges kulcsa, idegen kulcs

*ugyfel\_id:* az ügyfelek azonosítója, tábla elsődleges kulcsa, idegen kulcs

Tábla létrehozása:



### A „hibakodok” tábla

Ez a tábla tartalmazza a hibakódok tartozó elemek adatait

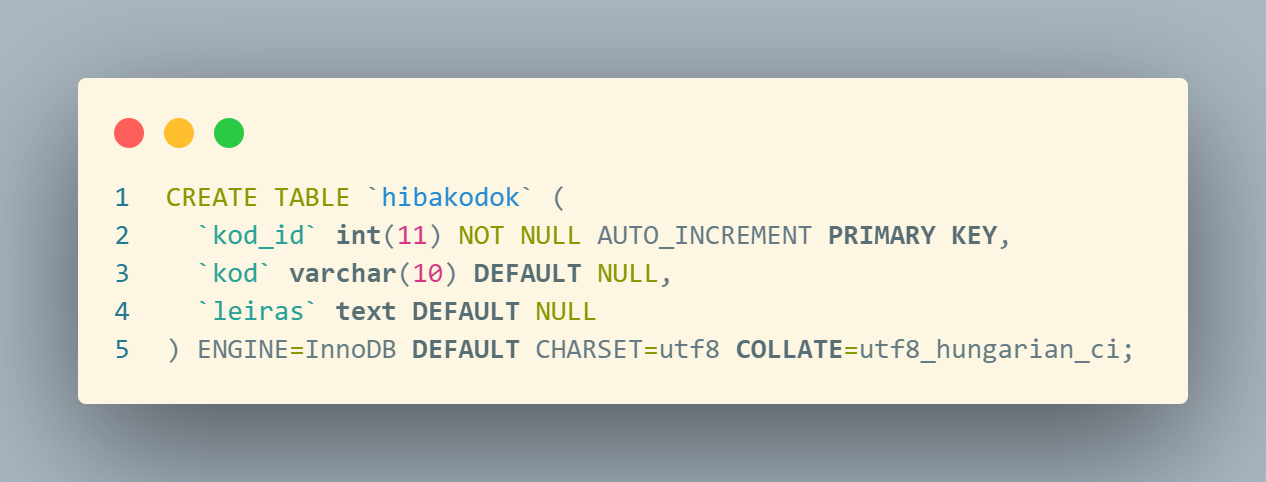


*kod\_id:* a hibakodok azonosítója, tábla elsődleges kulcsa

*kod:* maga a hibakód

*leiras:* a hibakód leírása

Tábla létrehozása:



### Az „idopontfoglalasok” tábla

Ez a tábla tartalmazza az időpont foglalások adatait



*idopont\_id:* az időpont azonosítója, tábla elsődleges kulcsa

*jarmu\_id:* a jármű azonosítója adott elemhez, a tábla idegen kulcsa

*csomag\_id:* a szervízcsomag azonosítója adott elemhez, idegen kulcs

*idopnt:* Az időpont

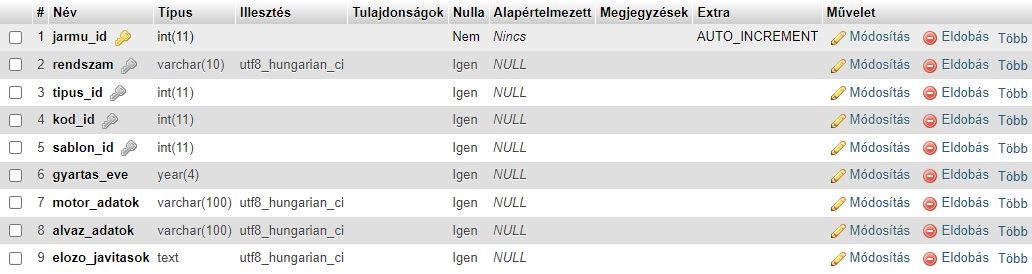
allapot: Az munka időpont állapota

Tábla létrehozása:



### A „jarmuvek” tábla

Ez a tábla tartalmazza a járművekhez tartozó adatokat



*jarmu\_id:* a járművekhez tartozó azonosító, tábla elsődleges kulcsa

*rendszam:* a járművek rendszáma egyedi kulcs

*tipus\_id:* A jármű típusának az azonsoítója idegen kulcs

kod\_id: A hibakód azonosítója idegen kulcs

sablon\_id: A munkafolyamat sablon azonosítója idegen kulcs

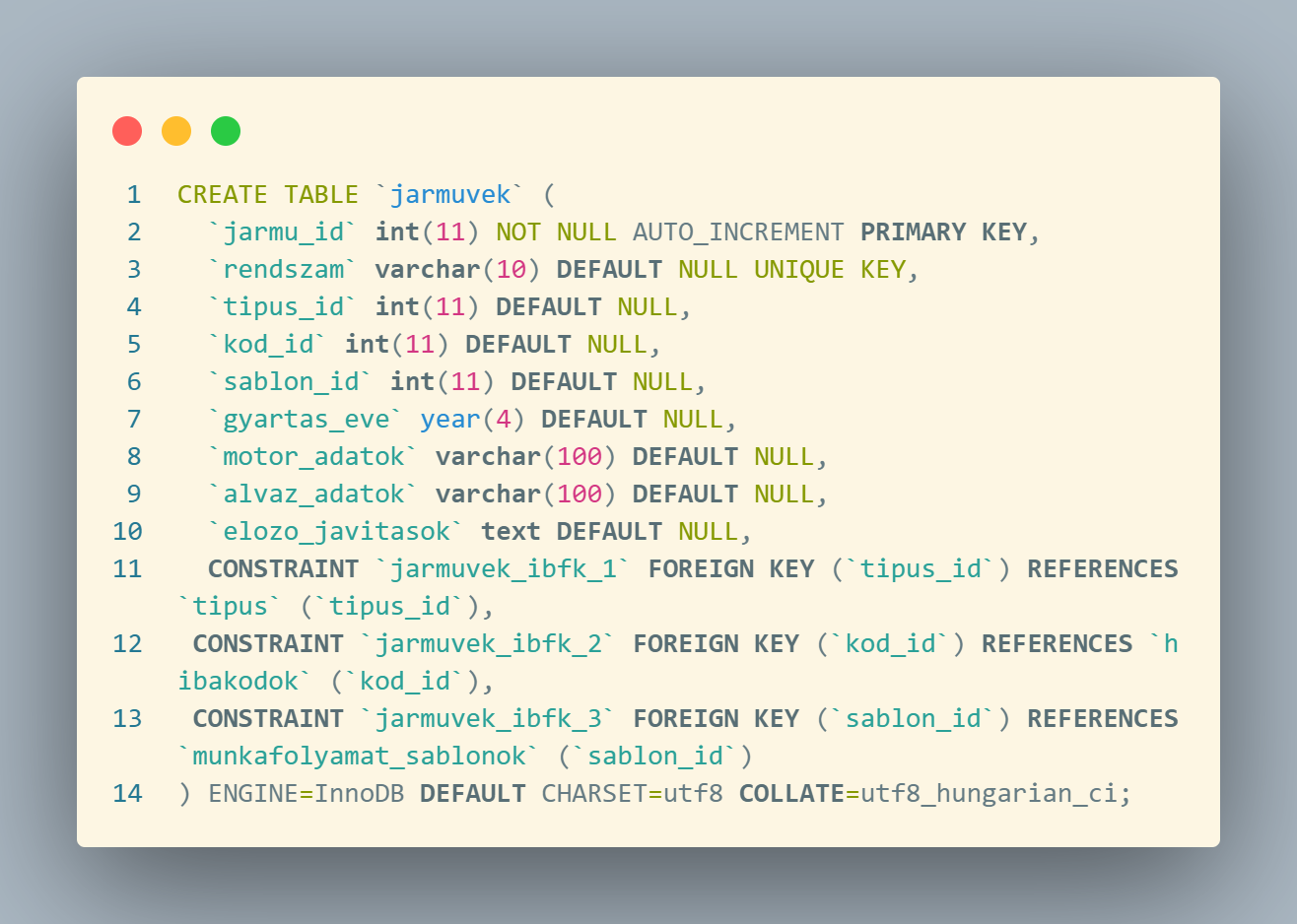
gyartas\_eve: A jármű gyártásának éve

motor\_adatok: A motor adatai

alvaz\_adatok: Az alváz adatai

elozo\_javitasok: Előzetesen elvégzett javítások az autón

Tábla létrehozása:



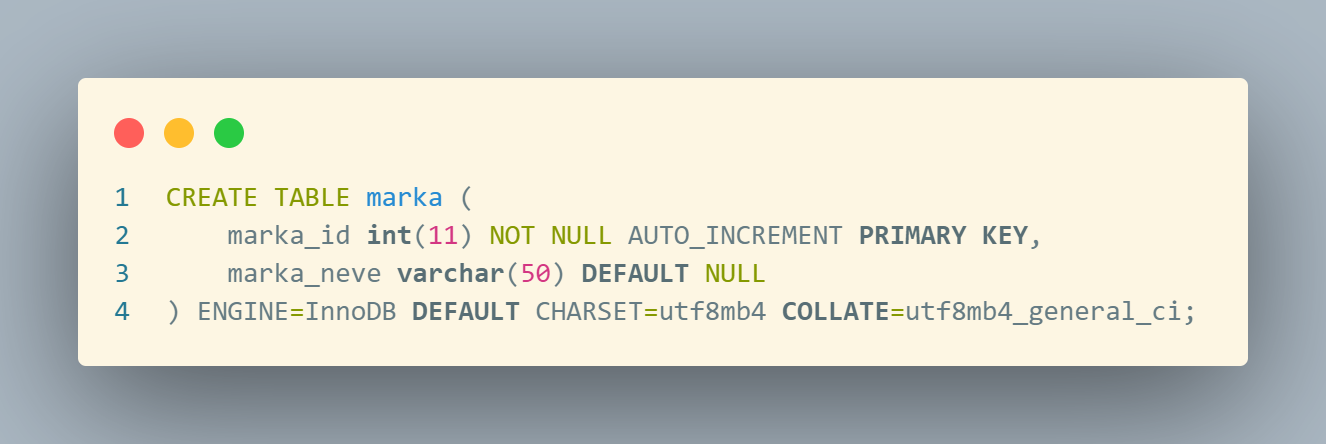
### A “marka” tábla

A tábla tartalmazza az autó márkákat

*marka\_id:* Az autó márkához tartozó azonosító, a tábla elsődleges kulcsa

*marka\_neve:* Az autó márka hivatalos megnevezése.

Tábla létrehozása:



### A „munkafolyamat\_sablonok” tábla

Ez a tábla tartalmazza a munkafolyamat sablonok adatait



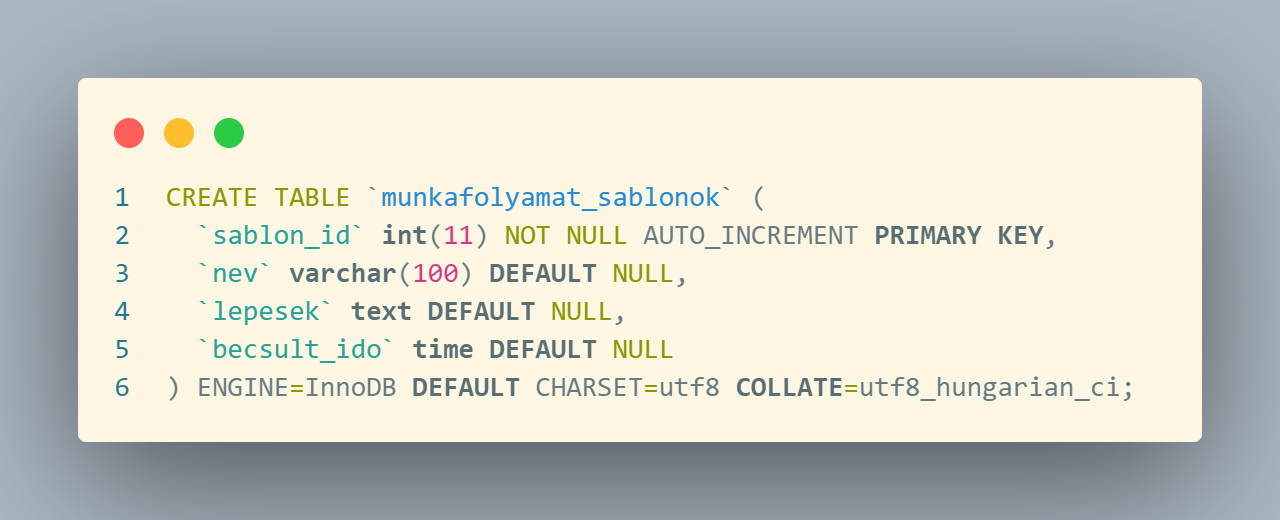
*sablon\_id:* a munkafolyamat sablonokhoz tartozó azonosító, tábla elsődleges kulcsa

*nev:* a sablonok neve

lepesek: a munkafolyamat lepesei

becsult\_ido: a munkafolyamat becsult ideje

Tábla létrehozása:



### A „rendelesek” tábla

Ez a tábla tartalmazza, hogy ki milyen és mennyi alkatrészt rendelt, és a rendelés státuszát

A képen szöveg, Betűtípus, képernyőkép, sor látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

*rendeles\_id:* a rendelés azonosítója

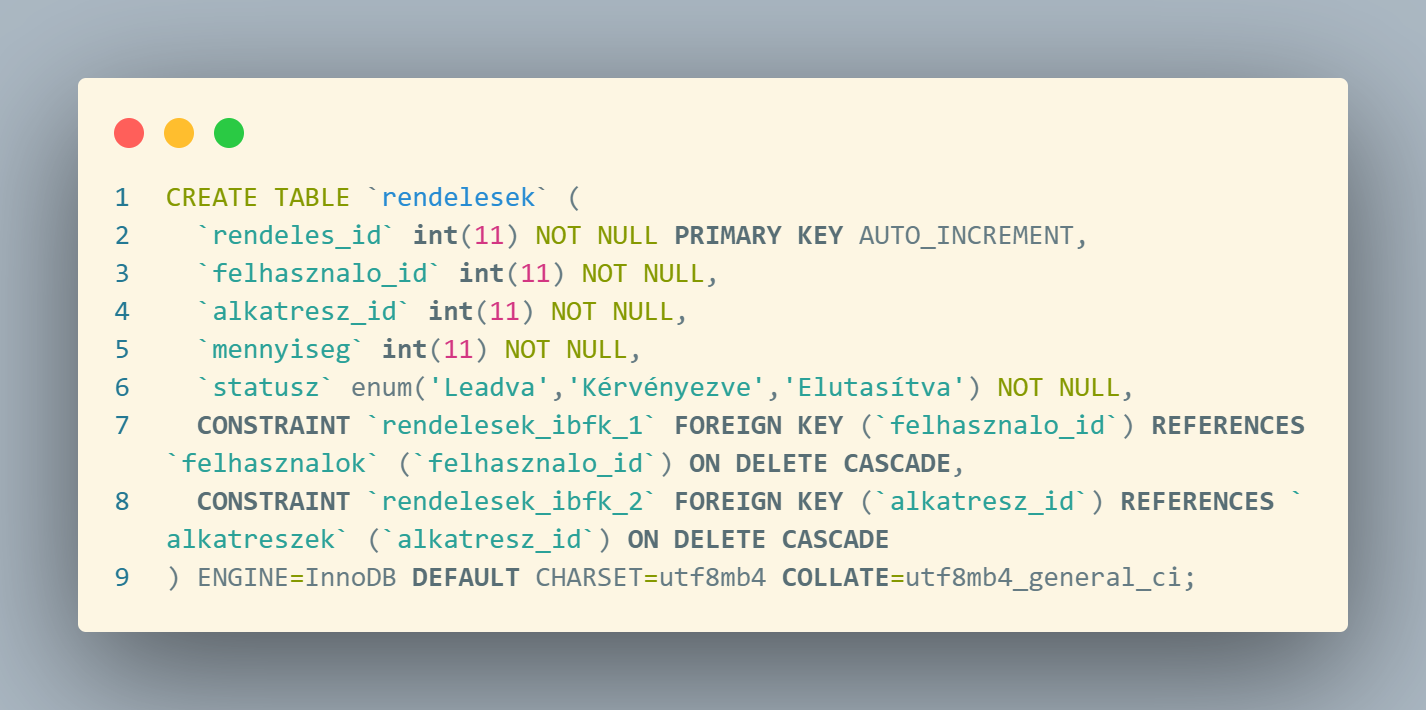
*felhasznalo\_id:* a felhasználó azonosítója, aki a rendelést intézte

*alkatresz\_id:* a rendelt alkatrész azonosítója

*mennyiseg:* a rendelt alkatrész mennyisége

*statusz:* a rendelés státusza

Tábla létrehozása:



### A „szerelesi\_utmutatok” tábla

Ez a tábla tartalmazza a szerelési útmutatók, adatait



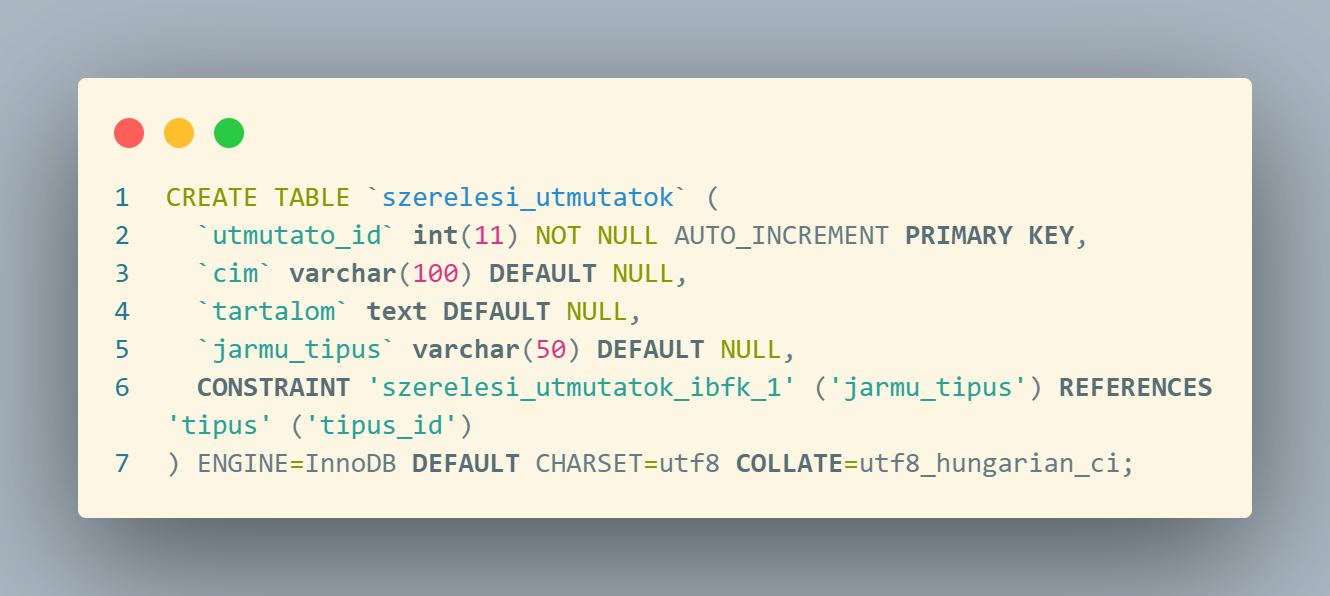
*utmutato\_id:* a szerelési útmutatók azonosítója, tábla elsődleges kulcsa

*cim:* a szerelési útmutatók megnevezése

tartalom: a szerelési útmutatók tartalma

jarmu\_tipus: a szerelési útmutatók, hogy milyen jármű tipusra vonatkoznak

Tábla létrehozása:



### A „szervizcsomagok” tábla

Ez a tábla tartalmazza a különböző szervízcsomagokat, és adatait



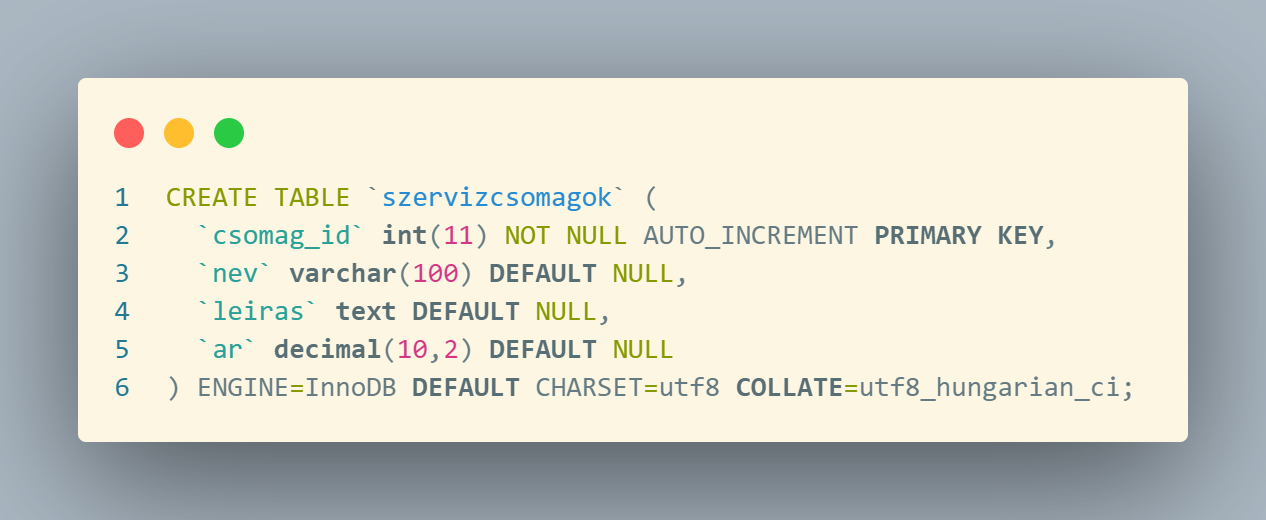
csomag\_id: Az adott szervízcsomag azonosítója

nev: A szervízcsomag neve

leiras: A szervízcsomag leírása

ar: A szervízcsomag ára

Tábla létrehozása:



### A „tipus” tábla

Ez a tábla tartalmazza az egyes járművek típusait

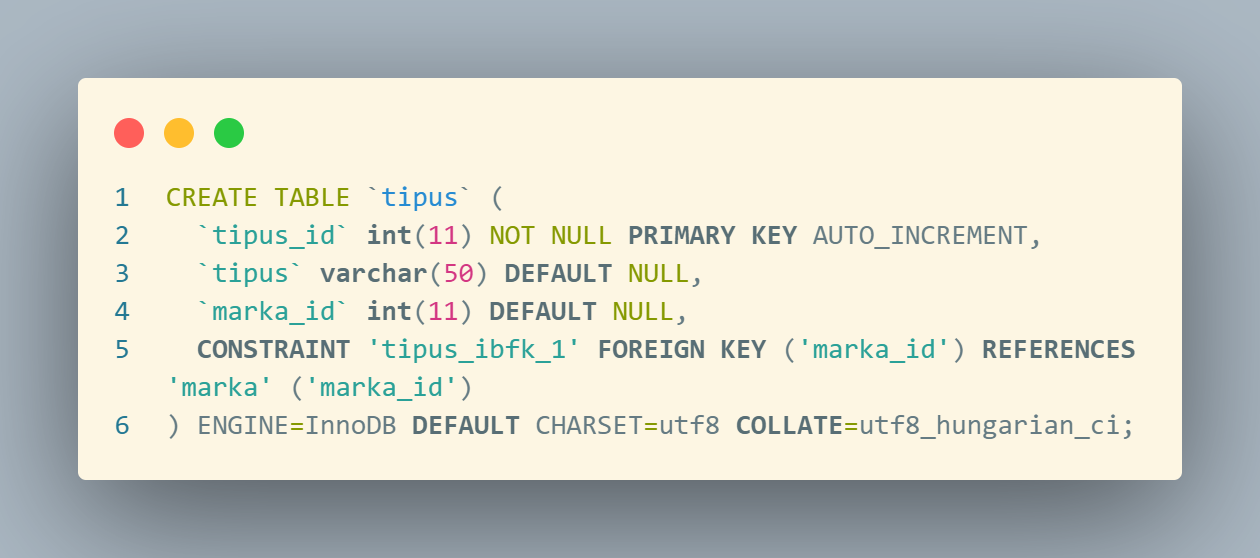


tipus\_id: A jármű típusának azonosítója

tipus: A jármű típusa

utmutato\_id: Az adott típus szereléséhez tartozó útmutató azonosítója

Tábla létrehozása:



### Az „ugyfelek” tábla

Ez a tábla tartalmazza az ügyfelek adatait



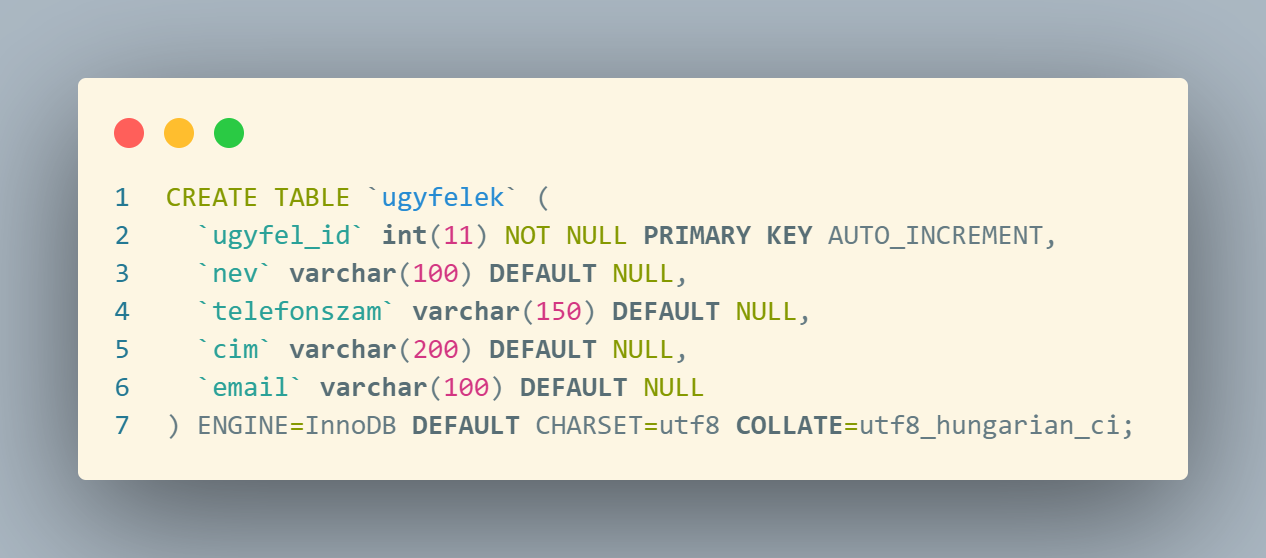
ugyfel\_id: Az ügyfél azonosítója

nev: Az ügyfél teljes neve

elerhetoseg: Az ügyfél elérhetősége

cím: Az ügyfél lakcíme

Tábla létrehozása:



### Az „ugyfel\_jarmuvek” tábla

Ez a tábla tartalmazza az ügyfelek és a járműveik kapcsolatát



ugyfel\_id: Az ügyfél azonosítója

jarmu\_id: Az ügyfél járművének azonosítója

Tábla létrehozása:



## Jogosultságok, jogosultsági szintek

### Admin

(asztali alkalmazás)

### Szerelő

(asztali alkalmazás)

### Ügyfél

A weboldalt teljes szinten kezelni tudja, csak ő fér hozzá. Regisztrálhat, be- és kijelentkezhet. Felveheti autóit az adatbázisba, módosíthatja azok rendszámát, törölheti is autóit a rendszerből, illetve szolgáltatásokat foglalhat le hozzájuk, és le is mondhatja azokat.

Az asztali alkalmazást ő nem használhatja, nincs hozzáférése, nem is ő a célcsoportja.

## Webalkalmazás

### Felépítése

A webalkalmazás két részre bontható, a frontendre és a backendre.

#### Frontend

##### HTML fájlok

HTML, CSS, JavaScript nyelveket és jQuery JavaScript könyvtárat használtunk a weboldal felépítéséhez. A webalkalmazás vázát a HTML kódok adják. Minden oldal külön .html fájllal rendelkezik.

##### CSS fájlok

Egy központi CSS fájl segítségével dolgoztunk alapból, de minden oldal rendelkezik sajáttal is. Reszponzívvá tettük az oldalt így ez megfelelően működik mind mobilon és asztali számítógépen is. Ezt gridek és flex elrendezések, illetve media queryk segítségével valósítottuk meg. Az összes CSS fájl egy helyen található, a public/css mappában.

##### JavaScript fájlok

Ezek a felhasználói felület interaktív működéséért felelnek, a legtöbb oldal így kapott egy hozzá tartozó dedikált JavaScript fájlt is. Ezek egy része module típusú, így exportálható, más helyen is használható metódusokat tartalmaz.

Ahhoz, hogy növeljük a felhasználói élményt, jQuery segítségével Ajax kéréseket végzünk a szerver felé, amelyet egy JSON tömbben kapunk vissza, majd a frontend dolgozza fel az adatok rendszerezését és megjelenítését a felhasználó számára, hogy az élmény még fluidabb legyen. Ezt animációkkal is segítünk jobbá tenni, amiben szintén a jQuery könyvtár segít.

Ezek a public/js/ elérési útvonalon kapnak helyet.

##### Képek

Ahhoz, hogy a felhasználói élményt jobbá tegyük, a public/assets/imgs útvonalon található képeket felhasználva építjük fel az oldalakat hogy színesebb és gazdagabb legyen az egész megjelenése.

#### Backend

A szerver teljes egészében PHP kódon alapul, és MVC architektúrát követ. Ennek elérési útvonala az app, config és routes mappa, ahol a szerver dolgozik.

A routes mappában az API található, ide érkeznek az Ajax kérések. Ezt típusuknak megfelelően feldolgozza a kód, és a megfelelő helyre továbbítja.

Az app mappában található a backend egy részének MVC (Model-View-Controller) megvalósítása, amely az API és az adatbázis között közvetít. Az MVC elemeinek lekérdezéseket intéznek az adatbázis felé, amelyeket JSON tömbbé formálnak, és továbbítják az API-nak.

A config mappában az adatbázissal kommunikáló kód és a környezeti változók beállítása foglal helyet, amely a .env segítségével történik.

Az az adatbázissal a Database.php létesíti a kapcsolatot. Ebben találhatók az ehhez szükséges bejelentkezési és azonosítási adatok is.

### Szerkezete

#### Frontend

Az azonos kiterjesztésű fájlokat egy helyre helyeztük, hogy ezeknek elkülönítése egyszerű legyen. Így mind a HTML, CSS és JavaScript kódok is egy helyen találhatóak meg.

##### HTML

Minden oldal head részében beégetve található meg a gyökér könyvtár az egyszerűbb elérés és navigálás érdekében. Ide lett még elhelyezve a minden oldalon megtalálható jQuery könyvtár, a main.js ami a fő JavaScript kód, illetve a JWT-hez szükséges fájl is.

Minden oldal egy közös style.css-ből dolgozik, illetve kapott egy saját dedikált CSS fájlt is, ahol csak az adott oldalhoz tartozó vizuális beállítások szerepelnek. Egyes oldalak, például az Autóim és a Szolgáltatásuk még tartalmaznak egy maguknak dedikált JavaScript fájlt is car.js és service.js néven.

Az oldalak fej- és láblécét minden betöltés előtt a backend építi fel, hisz például a fejlécben jelzi az oldal, hogy éppen melyik felhasználó van bejelentkezve.

A Regisztráció és Bejelentkezés oldal hasonlóan épül föl, mindegyik egy formot tartalmaz. Ugyanez elmondható az Autóim és a Szolgáltatások oldalról is, ahol kártyán jelennek meg az elemek.

##### CSS

A style.css fájlban vannak definiálva az oldalak alap alkotóelemei. A színeket, betűtípust és méretet is itt, a :root{}-ban adtuk meg, hogy az egységes megjelenítés megmaradjon, illetve a módosítás is könnyen végrehajtható legyen. Az input mezőkhöz, gombokhoz, űrlapokhoz és kártyákhoz mindenhol egységes nézetet használunk. Az oldalak címei és a hibaüzenetek is egységes stílussal rendelkeznek.

A webalkalmazás egésze reszponzívra lett kialakítva, így könnyen használható a felület egyaránt mobilon és asztali számítógépen is. Ezt jellemzően media queryk segítségével oldottuk meg egy @media only screen and (min-width: …px){} kód formájában. Ennek köszönhetően minden oldalon megfelelően látszik a tartalom. Az alábbi felbontásokat használtuk az oldalakon ebben a sorrendben:

* 320px
* 480px
* 544px
* 768px
* 896px
* 1024px
* 1366px

##### JavaScript

A legtöbb .js fájl importálással kezdődik, amivel más fájlokban található funkciókat teszünk helyben, az adott fájlban elérhetővé.

Ezeket aztán a window.függvény\_neve segítségével tesszük használhatóvá, hívhatóvá.

Ugyanazokat a .env-ben található környezeti változókat használjuk itt a frontendben is.

A $(document).ready(function(){}) részbe teszünk minden olyan elemet, amelynek bármikor minden körülménytől függetlenül elérhetőnek kell lennie.

Egy Ajax kérésben a data egy asszociatív tömb, amelyet GET-en kívül minden esetben elküldünk a szervernek.

Routingnál mindig / jelekkel választjuk el az végpontok egyes részeit.

Ha valami hiba történik a szervernél, az abból következő érkező választ az egyedi értesítési ablakban jelenítjük meg.

#### Backend

A szerverben az API felel a be érkező és kifelé menő kérések célba juttatásáért. Ide jönnek az Ajax kérések, amelyek a megelelő kontrollerek felé mennek tovább.

A $routes egy 4 fő csoportot tartalmazó asszociatív tömb, ebben azonosítja a fő függvény a beérkező kérés típusát, majd a kérés által küldött végpontot. Az alábbi kódrészlet a $routes egyik fő, és annak egy elemét mutatja be. A végpont hasonlóan néz ki, mint a JavaScript-ben, csak itt {változó} formában adjuk meg azokat az elemeket, amiket dinamikusan kérdez le a frontend.

A sendResponse metódus segítségével küldjük vissza a választ a frontendnek.

Egy átlagos kontroller neve tartalmazza az osztály nevét, vagy fordítva. A szükséges névtér az App/Controller, és a kontrollernek megfelelő osztályt, és egyéb osztályokat használjuk. Itt építjük össze a választ a frontend számára, vagy csak továbbítjuk a már a Model által összerakott választ.

Egy modellt ugyanúgy nevezünk el, mint egy kontrollert, és itt az adatbázist, illetve más modell szinten lévő osztályokat is használunk.

Válaszként általában egy asszociatív tömböt küldünk vissza a kontrollernek.

Az Database.php felelős az adatbázissal való kapcsolat felvételéért. Az App.php-ban állítjuk be a .env fájlban található környezeti változókat.

### Felületek

A webalkalmazásnak a főoldalán található az autóműhelyünk ismertetője a meglévő és leendő ügyfelek számára.

A Regisztráció oldalon a nevük, felhasználónevük, jelszavuk kétszeri, illetve telefonszámuk opcionális megadásával tudnak regisztrálni. A bejelentkezéshez csak egy felhasználónév és egy jelszó szükséges, ezt a Bejelentkezés oldalon tudják megtenni.

Az Autóim oldalon a felhasználó megtekintheti az autóit illetve azok állapotát és belük kapcsolatos egyéb információkat. Továbbá itt tudja őket módosítani, illetve új autót hozzáadni.

A Szolgáltatások oldalon az éppen elérhető szolgáltatásokat lehet megtekinteni, majd a kiválasztott autóhoz lefoglalni.

A Profilbeállítások oldalon a felhasználó a profiljával kapcsolatos adatokat tekintheti meg, illetve módosíthatja azokat.

A navigációs sávban az oldal logója és az elérhető oldalak helyezkednek el, illetve egy a felhasználó nevével ellátott gomb is, ez alatt érhető el a profilbeállítások oldal is, illetve itt tud kijelentkezni.

### Funkciók

#### Regisztráció

Ezért az auth.js-ben található register() metódus felel. Ellenőrizzük az űrlapot, hogy minden szükséges mezőt kitöltött e a felhasználó. Pozitív válasz esetén egy asszociatív tömbbe gyűjtjük az űrlapon kitöltött adatokat. Ezek után több biztonsági és formátumi ellenőrzést hajtunk végre.

Ha mindent rendben talál a kód, akkor egy POST Ajax kéréssel továbbítjuk a megfelelő URL-t és az űrlapból összegyűjtött data tömböt az API-nak. A UserController → createUser() létrehozza a felhasználót. Ez a .User → create() metódust hívja meg.

Először a fogadott data tömb elemeit kinyerjük, majd az adatbázishoz való csatlakozás után ellenőrizzük, hogy van e a már kiválasztott névvel rendelkező felhasználó. Nemleges válasz esetén először a felhasznalok táblába illesztjük be a felhasználónevet, az SHA 256-os algoritmust használva előállított jelszó hash-t, és az Ügyfél szerepkört. Illetve lekérjük a beillesztés után létrejött felhasználó azonosítóját.

Az ugyfelek táblába a megadott nevet, telefonszámot és emailt illesztjük be, majd lekérjük a beillesztés után létrejött ügyfél azonosítóját. A lekért felhasználó és ügyfél azonosítót felhasználva ezeket beillesztjük a felhasznalok\_ugyfelek kapcsolótáblába. Sikeres regisztráció esetén bejelentkeztetjük a felhasználót, ellenkező esetben pedig tájékoztatjuk őt az előfordult hibáról, amely foglalt felhasználónév, vagy egyéb hiba lehet.

#### Bejelentkezés

Szintén az auth.js-ben található login() függvény feladata a bejelentkezés. Egy data nevű asszociatív tömbbe összegyűjtjük a felhasználónevet és jelszót. Hibás űrlap kitöltés esetén közöljük a felhasználóval, hogy nem töltött ki megfelelően minden mezőt. Megfelelő űrlap kitöltés esetén egy POST Ajax kérést intézünk az API felé a megfelelő URL-lel és a data tömbbel.

Az API a UserController → loginUser() metódust meghívva jelentkezteti be a felhasználót. Bejelentkezés előtt a User → checkUser() függvény azonosítja, hogy a megadott felhasználónév és jelszó valóban egymáshoz tartozik.

Ha igaz értékkel tér vissza a függvény, akkor a User → find() metódus lekéri a felhasználó összes adatát, így az azonosításhoz szükséges felhasználó azonosítóját is.

Összeállítjuk a JWT-nek szükséges payload tömbböt, amely tartalmazza a felhasználó azonosítóját, a felhasználónevet és a token lejárati idejét. Ezt követi a JWT legenerálása, majd visszaadjuk a tokent a frondendnek.

Sikeres bejelentkezés esetén megadjuk a JWT-nek titkosítva a felhasználónevet és a felhasználó azonosítóját, és a főoldalra irányítjuk a felhasználót. Ha valami hiba történne a bejelentkezés alatt, arról az oldalon tájékoztatjuk a felhasználót.

#### Kijelentkezés

A kijelentkezésért az auth.js-ben megtalálható logoutAuth metódus felel. Az egyedi értesítési ablakban megjelenítjük a „Sikeres kijelentkezés!” üzenetet, majd töröljük az eddig beállított JWT tokent, és a főoldalra irányítjuk a felhasználót.

#### Profilbeállítások oldal betöltése

Az auth.js-ben található kódrészlet tölti be a felhasználó adatait a jelszavát kivéve, ha a profilbeállítások oldalra látogat. GET Ajax hívást intézünk az API felé. A UserController → singleUser metódus kéri le a felhasználó összes adatát a User Model → find() függvényel. A fogadott adatokat betölti a megfelelő input mezőkbe.

#### Profilbeállítások módosítása

Az auth.js-ben a userSettingsSave() metódus módosítja a felhasználó profiljának a beállításait. Az űrlapon szereplő adatok begyűjtése után ellenőrizzük, hogy minden \*-gal jelölt mezőt kitöltött a felhasználó. Ha igen, akkor ellenőrizzük, hogy a felhasználó által kétszer megadott jelszó egyezik e. Üres mező esetén ez természetesen nem módosul.

Az API felé egy PUT Ajax kérést intézünk, amely a UserController → updateUser() frissíti a felhasználó adatait a User →update() metódusával. A kapott asszociatív tömb adatainak kinyerése után egyszerűen módosítja a felhasznalok és ugyfelek táblában található felhasználónak a megadott adatait.

#### Autóim oldal betöltése

A car.js-ben található egy fetchCars() metódus, ez a cars.html oldal body elemének betöltésekor indul el. A kereséshez szükséges #myCars elem alapértelmezett állapotba állítása és az allCars tömb inicializálása után egy GET Ajax kérést intézünk az API felé.

Az autók adatait a CarController → userCars() kéri le, beállított felhasználói azonosító esetén a Car → find() metódusát használja. Amint megérkeztek az adatok a szervertől, és a kapott cars asszociatív tömb hossza nagyobb mint 0, végig iterálunk a tömb összes elemén egy forEach ciklussal, ahol létrehozzuk az autó kártyákat, ezeket feltöltjük a megfelelő adatokkal és gombokkal.

Ellenőrizzük, hogy van e aktív szolgáltatás az adott autóhoz kapcsolva, és ha igen, akkor megjelenítjük a lemondás gombot. Ezek után lezárjuk a kártyát, és megjelenítjük az oldalon. A végére hozzáadjuk az új autó hozzáadás űrlap kártyát is, és az összes autó márkát betöltjük az űrlap márka elemébe. Ha egy nem bejelentkezett felhasználó látogat az oldalra, akkor egy üzenetet jelenítünk meg neki.

#### Új autó hozzáadása

Ezért a car.js-ben található addCar() metódus felel. Az új autó űrlapon elhelyezett adatok begyűjtése és a form helyes kitöltésének ellenőrzése után egy POST Ajax kérést intézünk az API felé.

Ekkor ugyanis a CarController → createCar() metódusa létrehozza az autót a Car →create() függvénnyel. Ha kinyertük a kapott asszociatív tömbből a megfelelő adatokat, ellenőrizzük, hogy nincs e még egy ilyen rendszámú autó az adatbázisban. Ha nincs, akkor lekérjük az ügyfél és az autó típusának azonosítóját, majd felvisszük az autó adatait a jarmuvek táblába, és az ugyfel\_jarmuvek kapcsolótáblában egy új rekordot hozunk létre.

Sikeres autó hozzáadás esetén frissítjük az oldalon megjelenő autók listáját. Ellenkező esetben kijelezzük a hibát.

#### Autó rendszámának módosítása

Ezt a car.js-ben található alábbi kódrészlet hajtja végre. Alapértelmezetten a gomb első állapotában jelenik meg a „Rendszám módosítása” felirattal. Erre kattintva az autó rendszáma helyén egy beviteli mező jelenik meg, benne az autó rendszámával.

A felhasználó számára megfelelő beírt rendszám után ha a Véglegesítés feliratú gombot választjuk, akkor egy GET Ajax kérés indításával lekérjük az adott autó összes adatát. A kérés fogadása után az API a CarController → userCar() metódussal lekéri a felhasználó egy adott renszámú autójának összes adatát a Car → singleCar() függvény használatával. Amint a frontend megkapta az autó adatait, egy PUT Ajax kérést küldünk a szerver felé.

Ezt az API fogadja, a CarController → updateCar() függvény frissíti a felhasználó egy adott autójának rendszámát a Car → update() metódussal. A kapott asszociatív tömb adatainak kinyerése után ellenőrizzük, hogy nincs e már egy az új rendszámmal rendelkező autó az adatbázisban. Ha nincs, akkor módosítjuk az autó rendszámát az újra. Ha a felhasználó a módosítás előtt meggondolná magát a folyamattal kapcsolatban, akkor az autó kártyát az alábbi metódus állítja vissza eredeti állapotába.

#### Autó törlése

A car.js-ben található alábbi kódrészlet megjelenít egy megerősítő ablakot az autó törlését megelőzően. Ha a Törlés gombot választja a felhasználó, akkor egy DELETE Ajax kérést indítunk az API felé.

A CarController → deleteCar() metódus kitörli a felhasználó egy adott autóját az adatbázisból a Car → delete() függvénnyel. Először ellenőrizzük, hogy az autónak vannak e aktív, folyamatban lévő szolgáltatásai, különben nem engedélyezett a törlés. Ha nincsen aktív szolgáltatása, akkor kitörli az autót. A törlés után a fetchCars() metódus meghívásával frissítjük az autók listáját.

#### Aktív szolgáltatások megjelenítése

A .carDetails HTML osztályon belül elhelyezkedő „Aktív szolgáltatásaim” feliratú gombra kattintáskor egy GET kérést intézünk az API felé. Az API a CarController →getCarServices() függvényt használva lekéri az autóhoz tartozó aktív szolgáltatásokat a Car →carServices() metódussal.

Ha a kapott tömb nem üres, tehát van aktív szolgáltatás, akkor ezeket az egyedi értesítési ablakban jeleníti meg, ellenkező esetben pedig kiírja, hogy „Nincs aktív szolgáltatás”.

#### Előző javítások

A .carDetails HTML osztályon belül elhelyezkedő „Előző javítások” feliratú gombra kattintáskor egy GET kérést intézünk az API felé. Az API a CarControlleren → userCar() függvénnyel lekéri az autó összes adatát a Car →singleCar() metódussal.

Ha az előző javítások hossza nagyobb mint nulla, akkor azokat jeleníti meg az egyedi értesítési ablakban, ellenkező esetben pedig a „Nincs adat.” szöveg jelenik meg.

#### Szolgáltatás lemondása

Ha a .cancelService osztályú gombra kattintunk, akkor kiválasztja a hozzá tartozó kártyában szereplő autó rendszámát, majd betölti a jármű aktív szolgáltatásait egy GET Ajax kéréssel az API-tól.

Az API a CarControlleren → getCarServices() függvénnyel lekéri az autóhoz tartozó aktív szolgáltatásokat a Car → carServices() metódussal.

Ezek után a kapott szolgáltatásokat egy select elembe tölti be, amelyet az egyedi értesítési ablakban jelenít meg alatta egy gombbal.

Ha ezen a felületen a #cancelService azonosítójú gombra kattint a felhasználó miután kiválasztotta a lemondani kívánt szolgáltatást, akkor egy PUT Ajax kérést intézünk az API felé.

Az API a ServiceController → cancelService() függvénnyel lemondja a kiválasztott szolgáltatást a kiválasztott autóhoz a kapott asszociatív tömbből kinyert adatok után, a Service →cancel() metódus segítségével. Végezetül frissíti az „Autóim” oldalt.

#### Szolgáltatások oldal betöltése

A szolgáltatások betöltése a service.js fájlban található fetchServices() függvény meghívásával kezdődik, amikor a services.html törzse betölt. Egy GET Ajax kéréssel az API felé a ServiceController → allService() metódussal lekéri az összes szolgáltatást, ezeket kártyákba rendezi, mint az „Autóim” oldalon.

#### Szolgáltatás foglalása

A .book osztályú gombra kattintva az megkeresi a hozzá legközelebb álló, szolgáltatás kártyát, amelyben éppen benne van. Egy GET Ajax kérés segítségével az API-tól lekéri a kiválasztott szolgáltatás adatait és a felhasználó autóit is.

Az autók betöltéséhez a CarController → userCars() függvény lekéri a felhasználó összes autóját a Car → find() metódussal. Ha a felhasználó nincsen bejelentkezve, akkor kap egy erre felszólító üzenetet.

A finalize() függvény indítja el a szolgáltatás foglalása folyamatát. Az autó azonosítója és a szolgáltatás azonosítója segítségével összeállított asszociatív tömböt egy POST Ajax kérés keretein belül küldjük el az API-nak, amennyiben a felhasználó választott autót. A ServiceController-bookService() metódus a kapott asszociatív tömbből kinyert adatok után rögzíti a foglalást egy új rekordként az idopontfoglalasok táblában a Service → book() függvénnyel. A foglalás végeztével megjelenik az egyedi értesítési ablak, illetve frissítjük a szolgáltatások oldalt.

### A képen szöveg, képernyőkép, Betűtípus, szám látható Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Osztályok kapcsolati diagramjai

### Input adatok helyességének ellenőrzése és megkötései

#### Regisztráció

Ellenőrizzük, hogy a regisztrációkor kétszer megadott jelszó teljesen karakterről karakterre teljesen megegyezik, legalább 10 karakter hosszúságú, tartalmaz e legalább egy kis- és nagybetűt, számot illetve speciális karaktert az alábbiak közül: „@$!%\*?&\_”.

A telefonszám opcionális, ezt kivéve egyik bemeneti mező sem lehet üres.

A felhasználónév és név nem kezdődhet számmal, illetve a név nem tartalmazhat számot. Továbbá ellenőrizzük, hogy az email cím és a telefonszám megfelelő formátumú-e.

#### Új autó hozzáadása

A rendszám mező nem lehet üres, illetve ennek is ellenőrizzük a formátumát. Az autó márkája és típusa sem lehet az alapértelmezett, ki kell választania a felhasználónak egy márkát és egy típust. Egyedül az előző javítások mező lehet üres.

#### Rendszám módosítása

A rendszám mező itt sem lehet üres, illetve a formátumnak is helyesnek kell lennie.

#### Profil beállítások módosítása

Ugyanazokat az ellenőrzéseket hajtjuk itt végig, illetve ha a két jelszó mező 0 hosszúságú, akkor a jelszó nem változik.

### Biztonsági elemek

A jelszót a felhasznalok táblában tároljuk jelszo\_hash tulajdonságként, ezt mind a regisztrációnál, illetve a profil beállításoknál a MySQL alapértelmezett SHA2(jelszo, 256) függvénnyel egy SHA256 algoritmussal előállított jelszó hash-t készítünk. Ezzel megvédjük a felhasználók fiókját, ha esetleges adatszivárgás történne, ne tudjanak illetéktelenül hozzáférni a felhasználó fiókjához. Adatok bevitelénél a szerveren SLQ Injection elleni védelemmel előkészített MySQL parancsokat állítottunk össze.

### API dokumentáció

Az API elérési útvonala az /automuhely\_web/routes/api.php elérési útvonalon található meg.

Feldolgozzuk a kapott URL-t megfelelő formátumba, hogy ezzel tovább tudjunk dolgozni. A parseRoute() függvény felel, hogy a kapott metódus, URL és a routes asszociatív tömb alapján kiválassza a megfelelő meghívandó elemet.

A sendResponse() metódus küldi a választ a szervernek a meghívott függvény visszatérési értékei alapján. Ez tartalmazza a http válasz kódot, illetve általában egy összeállított asszociatív tömböt.

A routes egy asszociatív tömb, amely alapvetően a metódusok fajtái alapján 4 fő al-asszociatív tömböt tartalmaz. Továbbá ezek tartalmazzák az összes adott AJAX kérés típus alá tartozó elemeket. Ezen elemek általában az elérési URL-ből állnak, amit a szerver kapott, pl: /users/{userId}/cars/{licenseNumber}/services, és egy ehhez tartozó függvényt.

A kód alján létrehozzuk a routeInfo változót, amely a parseRoute függvényt használva megkeresi a megfelelő függvényt, majd meghívja.

# Felhasználói dokumentáció

## Webalkalmazás

### Rendszerkövetelmények

Egy Windows 10, vagy annál újabb operációs rendszert futtató számítógép tökéletesen megfelel erre a célra. Egy modern böngésző telepítése is szükséges, hogy megfelelően megjelenjen és működjen a weboldal.

Telepítendő szoftverek közé tartozik a XAMPP MySQL és Apache támogatottsággal, illetve a Composer PHP csomagkezelő program.

### Telepítése

#### A XAMPP feltelepítése

A képen szöveg, képernyőkép, névjegykártya, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Elsőnek látogassunk el a [www.apachefriends.org](http://www.apachefriends.org) weboldalra, majd szerezzük be a megfelelő verziójú XAMPP telepítőt.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A telepítés folyamán ajánlott, hogy válasszunk ki minden komponenst, amit a telepítő már alapból meg is tett helyettünk, hogy ne ütközzünk semmilyen hibába.

Telepítési helynek válasszuk célszerűen az alapból megadott C:\xampp elérési útvonalat, majd folytassuk a telepítést.

Ha végzett a telepítő, indítsuk el az alkalmazást, és a Control Panelen található Apache és MySQL szolgáltatást is.

Célszerű, hogy ne legyen még egy automuhely nevű adatbázis a szerveren. Ha mégis létezik ilyen, akkor azt nevezzük át ideiglenesen, hogy ne ütközzünk fennakadásokba.

Amint ezt leellenőriztük, futtassuk az asztali alkalmazást, hogy az létrehozza nekünk a teljes adatbázist. Amint ezzel megvagyunk, azt zárjuk be.

#### A Composer feltelepítése

Következőnek látogassunk el a <https://getcomposer.org/download/> weboldalra, innen szerezzük be a legfrissebb Composer verziót.A képen szöveg, képernyőkép, Betűtípus, algebra látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A telepítő indításnál ha van rá módunk, telepítsük ezt fel minden felhasználónak, ha más profilról is szeretnénk használni a weboldalt. Hagyjunk minden beállítást alapértelmezetten a telepítőben, így végezzük el a telepítést.

#### A weboldal beüzemelése

Most helyezzük át az “automuhely\_web” névvel ellátott könyvtárat a XAMPP htdocs mappájába. Ha van már egy ilyen nevű mappa eleve a könyvtárban, akkor azt ideiglenesen nevezzük át, hogy elkerüljük az esetleges hibákat.

Ha nem szeretnénk élni ezzel a lehetőséggel, és egy egyedi mappa nevet választunk a projektnek, akkor egy kódszerkesztővel, mint például a fent említett VSCode segítségével nyissuk meg a web könyvtárát a “C:\xampp\htdocs\kiválasztott mappa név\” elérési útvonalon, amelybe már bemásoltuk az automuhely\_web tartalmát, és egy Ctrl+Shift+F billentyű kombinációval elénk tárul egy keresőmező. Ide írjuk be a “/automuhely\_web/” keresendő kifejezést, és helyére gépeljük be a kiválasztott mappa nevét.

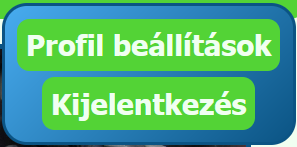
CMD-ben látogassunk el a C:\xampp\automuhely\_web\ vagy a C:\xampp\kiválasztott mappa név\ könyvtárba, majd az alábbi paransokat futtassuk le:

* composer install
* composer dump-autoload

Ha mindent jól végeztünk, akkor a weboldalnak elérhetőnek kell lennie a <http://localhost/automuhely_web/public/> URL-en.

### Felületek

Az fej- és lábléc mindenhol megegyezik oldaltól függetlenül. Az oldalak tetején található az Autóműhely logója, az oldal címe, a továbbá elérhető oldalak, illetve bejelentkezett felhasználóknak a „profil” gomb, amelyben a felhasználónév szerepel.

A „profil” gombra kattintva egy kisebb menü tárul elénk, amelyben a felhasználói profil beállítások oldalt érhetjük el, vagy kijelentkezhetünk.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Mobilon egy hamburger menü fogad minket, amelyre kattintva elérhetővé válik az összes menüpont.

#### Főoldal

A főoldal a <http://localhost/automuhely_web/public/index.html> címen érhető el. Legfelül az oldal címe, alatta egy figyelem felhívó, lentebb egy üdvözlő szöveg látható.

A képen szöveg, képernyőkép, Márka látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A figyelemfelhívó szöveg be- és kijelentkezett felhasználók számára egyaránt változik.

A képen szöveg, képernyőkép, Betűtípus, Acélkék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Aki nincs bejelentkezve, annak számára elérhető az útvonal a regisztrációs és bejelentkező felülethez.

A képen szöveg, képernyőkép, Betűtípus, Acélkék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

#### Regisztráció oldal

Az oldalon egy regisztrációs űrlap található, ahol a felhasználónak kötelezően meg kell adnia a vezetéknevét, keresztnevét, felhasználó nevét, emailcímét, és a jelszavát kétszer megerősítésképpen. A telefonszámát opcionálisan megadhatja. A csillaggal jelölt mezők kötelezőek.



A jelszót alapértelmezetten pontokkal jelölve jeleníti meg az oldal, ám ennek megtekintésére lehetőség van, ha a mezők melletti szem ikonra kattint a felhasználó. Ezzel megtekintheti a jelszót, ilyenkor „kinyílik” a szem, majd el is rejtheti, ilyenkor „becsukódik” a szem.

A képen szöveg, képernyőkép, Betűtípus, Grafika látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az űrlapon található egy „Regisztráció” feliratú gomb, amelyre kattintva regisztrál a felhasználó, majd az oldal automatikusan bejelentkezteti őt. Ha valami hiba történik, erről egy üzenetet kap.

Amennyiben a mezők kitöltése közben a felhasználó valamilyen formátumú hibát vét, az űrlap alján erről figyelmezteti őt a felhasználó.A képen szöveg, képernyőkép, Operációs rendszer, multimédia látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ha a felhasználó már rendelkezik fiókkal, a bejelentkezés oldalt könnyen elérheti az űrlap alján található linken keresztül.

Bejelentkezés után a fő oldalon találja magát a felhasználó.

#### Bejelentkezés oldal

Az oldalon egy bejelentkezési űrlap található, ahol a felhasználó nevét és jelszavát megadva jelentkezhet be.



Ha rossz a párosítás, erről szintén az űrlap alján kap üzenetet a felhasználó.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Az űrlapon található gombbal lehet bejelentkezni, illetve a jelszó mezőhöz tartozó szem ikon ugyanúgy működik, mint a bejelentkezés oldalon.

Amennyiben a felhasználó még nem regisztrált, az oldal alján lehetősége van a regisztrációs felületre ellátogatni az űrlap alján jelzett linken keresztül. Bejelentkezés után a fő oldalon találja magát a felhasználó.

#### Szolgáltatások oldal

A felhasználó ezen az oldalon az elérhető szolgáltatásokat tekintheti meg kártyák formájában. Itt megtalálható az adott szolgáltatás neve, leírása és ára is. Kereshet közöttük, és szűrők alapján rendezheti is őket a felhasználó.

A képen szöveg, képernyőkép, Betűtípus, multimédia látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az adott kártya foglalás gombjára kattintva az oldal alján megjelenik egy kártya „Kiválasztott szolgáltatás” címmel, ahol megtekinthető a szolgáltatás neve, leírása és az ára is. Azonban, ha nincs bejelentkezve a felhasználó, egy üzenet hívja fel erre a figyelmét, hogy a foglalás folytatásához jelentkezzen be.A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Amennyiben már bejelentkezett a felhasználó, de még egy autóját sem adta hozzá az oldalhoz, erről itt tájékoztatást kap.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Azonban, ha bejelentkezett, és újra kiválasztja az egyik szolgáltatást, lent választhat a már hozzáadott autói között, amihez a szolgáltatás szeretné foglalni.

A képen szöveg, képernyőkép, multimédia, Operációs rendszer látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A „Véglegesítés” gombra kattintva elindul a foglalás, ennek sikerességéről egy üzenetet kap a felhasználó, majd újra töltődik az oldal.

#### Autóim oldal

Ezen az oldalon a felhasználó saját, már hozzáadott autóit tekintheti meg szintén kártyák formájában.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Amennyiben még nincs hozzáadott autója, erre fel is hívja figyelmét az oldal, hogy „Adja hozzá első autóját!”.

A képen szöveg, jármű, Szárazföldi jármű, kerék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ezt az „Új autó” címmel ellátott űrlapon teheti meg, ahol megadhatja autója rendszámát, márkáját, típusát, és az előző javításokat is, de az utóbbi opcionális, ha nem volt.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Az autó hozzáadását az „Új autó” feliratú gombbal tudja megtenni.

Ha valami hibát vét az űrlap kitöltése közben a felhasználó, akkor erről tájékoztatást kap.

A képen szöveg, képernyőkép, Betűtípus, okostelefon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A már meglévő autójának az alábbi adatait tekintheti meg:

Előző javításokat

A képen szöveg, Betűtípus, képernyőkép, Acélkék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Aktív szolgáltatásokat

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, Betűtípus, képernyőkép, Acélkék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A „Rendszám módosítása” feliratú gombra kattintva tudja módosítani az autója rendszámát. Ekkor a rendszám helyén egy beviteli mező jelenik meg benne a rendszámmal. Ilyenkor a Lemondás és a Törlés gomb szürke lesz, ezzel jelezve, hogy pillanatnyilag ezek a műveletek nem elérhetőek.

A módosított rendszám beírása után a „Véglegesítés” gombra kattintva módosul a rendszám, ennek sikerességéről üzenetet is kap a felhasználó.

A képen szöveg, Betűtípus, képernyőkép, Acélkék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Ha meggondolná magát a felhasználó, a „Mégsem” feliratú gombra kattintva visszaáll az autó kártyája az eredeti állapotába.

Az autóját törölheti is a felhasználó a „Törlés” gombra kattintva. Ilyenkor kap egy megerősítő kérdést „Törlés” és „Mégsem” feliratú gombokkal, hátha meggondolta magát a felhasználó.

A képen szöveg, képernyőkép, Betűtípus, Acélkék látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A törlés gombra kattintva, ha van foglalt vagy folyamatban lévő szolgáltatása, egy figyelmeztetést kap, hogy amíg ezek nem fejeződtek be, addig nem lehet az autót eltávolítani.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Azonban, ha nem áll fenn ilyen eset, és szintén a törlés gombot választja a felhasználó, akkor az autója törlődik az oldalról.

Ha az igen lehetőségre kattint, akkor törlődik az autója. Ellenkező esetben csak bezáródik az értesítés ablak.