Informatikai rendszerterv sablon

Tartalom

[1. Informatikai rendszerterv sablon 4](#_Toc129941717)

[1.1 Bevezető, a probléma rövid ismertetése 4](#_Toc129941718)

[1.2 Választott téma indoklása 4](#_Toc129941719)

[1.3 Téma kifejtése 4](#_Toc129941720)

[2. Fogalmak és rövidítések #(megmagyarázni) 4](#_Toc129941721)

[3. Architektúra 5](#_Toc129941722)

[3.1 Áttekintés 5](#_Toc129941723)

[3.2 Rétegek és felelősségek 6](#_Toc129941724)

[3.3 Választott technológiák 16](#_Toc129941725)

[3.4 Asztali alkalmazás: 16](#_Toc129941726)

[3.5 Web-es app: 17](#_Toc129941727)

[3.6 Külső könyvtárak: 17](#_Toc129941728)

[3.7 Függőségek 18](#_Toc129941729)

[3.7.1 Minimális hardver követelmény: 18](#_Toc129941730)

[3.7.2 Ajánlott hardver követelmény: 18](#_Toc129941731)

[3.7.3 Desktop app: 19](#_Toc129941732)

[3.7.4 Web App: 19](#_Toc129941733)

[4. Modulok 19](#_Toc129941734)

[4.1 Áttekintés 19](#_Toc129941735)

[4.2 Modulok részletei 19](#_Toc129941736)

[5. Perzisztencia 20](#_Toc129941737)

[5.1 A tárolás választott módja 20](#_Toc129941738)

[5.2 Koncepcionális adatmodell 20](#_Toc129941739)

[6. Folyamatok 20](#_Toc129941740)

[7. Interfészek 21](#_Toc129941741)

[8. Biztonság 21](#_Toc129941742)

[9. Üzemeltetést támogató képességek 21](#_Toc129941743)

[9.1 Konfiguráció 21](#_Toc129941744)

[9.2 Monitorozás 22](#_Toc129941745)

[9.3 Logolás 22](#_Toc129941746)

[9.4 Hibaelhárítás 22](#_Toc129941747)

# Informatikai rendszerterv sablon

Verzió: 1.0

## Bevezető, a probléma rövid ismertetése

Az autókereskedés eddig csupán facebookon működött, mert alig 100 autó volt eladó és hirdetett. 2022-ben az eddigi profit felhasználásával bővült a kínálat, több mint 300 autóra, amire már nem volt alkalmas a Facebook Marketplace. Ez okból kellett egy weboldal, és egy hozzátartozó adatbázis, mellyel könnyedén vezetni lehet az eladó autókat, és minden hozzájuk tartozó tulajdonságot.

## Választott téma indoklása

Gyerekkorom óta közel állnak hozzám az autók. Apukámmal mindig autót szereltünk, és így sok kellemes emlék köt össze a kocsikkal. Érdekel minden tulajdonságuk, felszerelésük, teljesítményük stb. Gyakran szoktam eladó autókat nézegetni, és kíváncsi voltam, hogy én magam milyen weboldalt tudnék erre létrehozni, és hogy milyen tudás áll egy ilyen weboldal mögött

## Téma kifejtése

A témám egy autókereskedés weboldala, ahol számon van tartva egy cég minden egyes eladott autója. Az oldalon megtalálható a cég elérhetősége, és az eladó autókról minden féle információ (ár, megtett kilométer, hengerfejek száma, szín, kivitel, évjárat, gyártó, modell, meghajtás, váltó). Az autókra ezek a tulajdonságok alapján szűrni is lehet. Be lehet állítani kedvenc autókat, hogy később ne kelljen keresgélni, hanem egy kattintással elérhető. Jelen pillanatban több mint 400 autó található meg az adatbázisban. Az oldalra regisztrálni is lehet, mint átlag felhasználó. Viszont vannak felhasználók, akik több jogosultsággal rendelkeznek, mint egy vendég, például az adminisztrátor, aki a regisztrált felhasználókat is meg tudja nézni. A weboldalnak van egy asztali alkalmazása is, ahol az adatbázis kezelésére szolgáló menük találhatóak meg. Ide is bejelentkezés alapján lehet csak bejutni különböző felhasználói szintek szerint.

# Fogalmak és rövidítések #(megmagyarázni)

* HTML5: A HTML (Hypertext Markup Language) átdolgozott változata, célja, hogy a webes alkalmazásokhoz ne kelljenek pluginok.
* MySQL: Adatbázis kezelő szerver.
* CSS: Olyan programnyelv, mellyel HTML weboldalakat dizájnolhatunk.
* EJS: Egy sablon nyelv, mellyel HTML oldalt lehet generálni egyszerű JavaScripttel
* JavaScript: Weboldalak programozási nyelve.
* .NET framework: Microsoft által létrehozott szoftver keretrendszer.

# Architektúra

## Áttekintés

A rendszer két fő modulból áll:

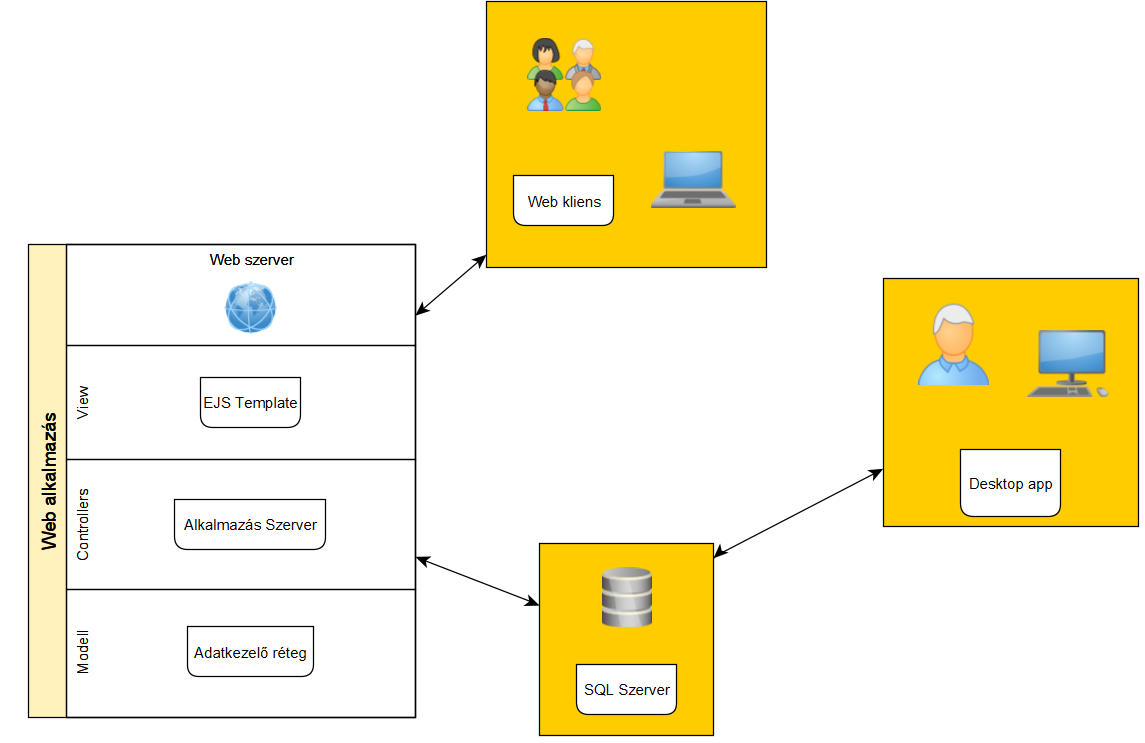
* Asztali alkalmazás:
* WEBkiszolgáló
* **Müködési környezet**: Windows környezet
* Windows .NET app: Microsoft által készített keretrendszer, mely a gyors alkalmazás fejlesztést szolgálja.
* alkalmazásszerver – **NodeJS** v16.17.1: Webszerverek készítését szolgáló szoftverrendszer.
* **MySQL** adatbázisszerver 8.0.31: Adatbázis kezelésre szolgál
* **Connector/NET** 8.0.28
* Támogatott böngésző (Chrome, Firefox, Brave)
* Használat protokollok (http,https)
* **Portok**:
  + MySQL: 3306
  + http: 80 (Jelenleg a webkliens https protokollal működik)
  + https: **443**
* **SSL** tanúsítvány
  + Teszteléshez használt: OpenSSL programmal generált önaláírt tanúsítvány
  + Az átadott verzióhoz ajánljuk valamelyik tanúsítványkiszolgálótól megvásárolni a használt domainhez (pl.: carmarket.hu) tartozó tanúsítványt.
  + Tanúsítvány tároló mappa helye
  + [telepítési mappa]\cert\localhost\
  + [telepítési mappa]\bin\www.js tartalmazza a használt tanúsítvány elérhetőségét

const options ={

key:fs.readFileSync(path.join(\_\_dirname,'../cert/localhost/localhost.decrypted.key')),

cert:fs.readFileSync(path.join(\_\_dirname,'../cert/localhost/localhost.crt'))

}



1. ábra

## Rétegek és felelősségek

**Router Layer:**

A Router Layer tartalmazza az alkalmazás programozási felületének (API) útvonalait. Feladata, hogy meghívja az alkalmazás szerver megfelelő metódusát és várja a válaszát. Választ küldjön a szervertől a webes böngésző felé.

**A modul helye**: [telepítési mappa]\routes\ mappa

**A réteg komponensei:**

* *index.js* – általános útvonalak kezelése

|  |  |  |  |
| --- | --- | --- | --- |
| **URL** | **Paraméterek** | **Metódus** | **Funkció** |
| / | - | GET | Betölti a kezdőoldalt |
| /cars | - | GET | Betölti az összes autót |
| /cars/:page | page: oldalszám | GET | Egy sablont tölt be 30 autóval, a megadott oldalszámnak megfelelően |
| /search | - | GET | Betölti a keresési oldalt |
| /search | - | POST | A bodyból kiveszi a szükséges paramétereket/feltételeket, és aszerint visszaadja a keresésnek megfelelő elemeket |
| /car/:id | id: autó azonosítója | GET | Azonosító alapján betölti az autó adatlapját |

* *users.js* – a felhasználó kezeléshez tartozó útvonalak kezelése

|  |  |  |  |
| --- | --- | --- | --- |
| **URL** | **Paraméterek** | **Metódus** | **Funkció** |
| /user/login | - | GET | Betölti a bejelentkezési oldalt |
| /user/login | - | POST | A bodyból kiveszi a szükséges paramétereket/feltételeket, és aszerint bejelentkezteti a felhasználót |
| /user/registration | - | GET | Betölti a regisztrációs oldalt |
| /user/registration | - | POST | A bodyból lekért paraméterek alapján regisztrálja a felhasználót |
| /user/accounts | - | GET | Betölti a felhasználók listáját |
| /user/profile | - | GET | Betölti a felhasználó adatait |
| /user/favourites | - | GET | Betölti a felhasználó kedvenc autóit |
| /user/edit | - | GET | Betölti az oldalt, ahol a felhasználó szerkesztheti az adatait |
| /user/edit | - | POST | Betölti az oldalt, ahol a felhasználó szerkesztheti az adatait |
| /user/addFavourute | - | POST | A bodyból kiszedi az autó azonosítóját, és hozzáadja a felhasználó kedvenceihez |
| /user/removeFavourite | - | POST | A bodyból kiszedi az autó azonosítóját, és eltávolítja azt a felhasználó kedvenceiből |
| /user/reset-password | - | GET | Megjeleníti az új jelszó generálására szolgáló weblapot. |
| /user/reset-password | - | POST | Leellenőrzi, hogy a megadott email cím megtalálható-e az adatbázisban, ha nem akkor megjeleníti az oldalt újra, egy hibaüzenettel, ha igen, akkor generál egy 10 karakter hosszú véletlenszerű jelszót, és azt a emailjs fetch API segítségével, kiküldi a felhasználó email címére. |
| /user/logout | - | GET | Kijelentkezteti a felhasználót |

**Controller réteg:**

A Controller Layer kezeli az alkalmazás üzleti logikáját. Ez azt jelenti, hogy az adatokat átalakítják vagy kiszámítják, hogy megfeleljenek az adatbázismodell követelményeiknek, mielőtt elküldök őket a kiszolgálóra. A router réteg része.

**Adatkezelő réteg:**

Az adatkezelő réteg (Modells) hozzáféréssel rendelkezik az adatbázishoz, hogy adatokat hozzon létre, töröljön vagy szerkesszen.

Az adatbázishoz szükségek paramétereket a **[telepítési mappa]\db\dbconfig.js** fájl tartalmazza:

const config = {

    connectionLimit: 10,

    password: 'root',

    user: 'root',

    database: 'auto',

    host: 'localhost',

    port: 3306,

    dateStrings: true

  };

A telepítést követően a paramétereket a helyi sajátosságoknak megfelelően kell beállítani.

A telepített verziónál ajánlatos egy korlátozott jogosultságokkal rendelkező felhasználó létrehozása a MySQL szerveren és ezzel a felhasználóval csatlakozni.

**connectionLimit:** Megadja mennyien csatlakozhatnak egyszerre a webkliensen keresztül az adatbázishoz.

**password**: a felhasználóhoz tartozó jelszó.

**user**: a MySQL szerverhez kapcsolódó felhasználó neve.

**database:** Az adatbázis neve.

**host**: amennyiben az SQL szerver és a webkiszolgáló külön gépen kerül telepítésre, itt kell megadni a MySQL szervergép hostnevét

**port:** A port amin a MySQL szerver fut.

**dateStrings**: Dátumok könnyebb kezelése.

**A réteg felelősségei:**

Ez az a réteg, ahol a szerver logikájából érkező összes kérést és választ kezelik. Az adatelérési (Data Access Layer) metódusok változókat kapnak a controller rétegtől. Ezek a változók szükségesek az SQL lekérdezésekhez. Az SQL szervertől kapott válasz alapján ígéretet (Promise) küld, amely egy objektum. Az objektum a szerver válaszának rendelkezésre állása után kerül meghatározásra. Ha a kérés sikeres, a DAL ígéretet egy objektum oldja fel, amely visszatér a szolgáltatási réteghez, amely maga is visszatér az útválasztási réteghez. Amikor az útválasztási réteg megkapja a szolgáltatási réteg által visszaadott objektumot, az útválasztási réteg JSON formátumban küldi el az objektumot a View rétegnek.

**A modul helye: [telepítési mappa]\db\dboperation.js**

**A réteg komponensei:**

|  |  |  |
| --- | --- | --- |
| **Név** | **Paraméterek** | **Funkció** |
| Select30 | -pageNo: oldalszám | Lekérdezi 30 autó összes adatát, annyival eltolva amennyi az oldalszám |
| SelectAll | - | Lekérdezi az összes autó adatát az adatbázisban |
| SelectOne | -carId | Lekérdezi egy autó összes adatát, az autó azonosítója alapján |
| SelectAllSpecial | - | Lekérdezi a különleges ajánlatokban lévő autók azonosítóját, gyártóját, modellét, évjáratát, és a hozzáírt bemutató szöveget |
| CountElements | - | Lekérdezi, hogy hány autó adata szerepel az adatbázisban |
| SelectAllMake | - | Lekérdezi az összes gyártót |
| SelectAllCondition | - | Lekérdezi az összes állapotot |
| SelectAllColor | - | Lekérdezi az összes színt |
| SelectAllFuel | - | Lekérdezi az összes üzemanyag típust |
| SelectAllType | - | Lekérdezi az összes kivitelt |
| SelectAllTransmission | - | Lekérdezi az összes váltótípust |
| SelectAllDrive | - | Lekérdezi az összes meghajtást |
| SelectModelByMake | - | Lekérdezi az összes modellt és gyártót, a gyártó szerint csoportosítva |
| SelectMaxPrice | - | Lekérdezi a legdrágább autó árát |
| SelectMinYear | - | Lekérdezi a legrégebbi autó évjáratát |
| SelectMaxYear | - | Lekérdezi a legújabb autó évjáratát |
| EditUser | -name: felhasználó módosított neve  -email: felhasználó módosított email címe  -username: felhasználó módosított felhasználóneve  -id: felhasználó azonosítója | A megadott azonosítóval rendelkező felhasználóhoz tartozó mezőket módosítja a paramétereknek megfelelően |
| VerifyUser | -username: felhasználónév  -password: jelszó | Megnézi, hogy található-e az adatbázisban a paraméterekben megadott felhasználónév-jelszó páros. A jelszót titkosítani kell a lekérdezés során |
| VerifyUserById | -id: felhasználó azonosítója  -password: felhasználó által megadott jelszó | Megnézi, hogy található-e az adatbázisban a paraméterekben megadott azonosító-jelszó páros. A jelszót titkosítani kell a lekérdezés során. |
| CheckEmail | -email: felhasználó által megadott email cím | Lekérdezi az összes olyan rekordot, ahol az email cím megfelel a paraméterben megadottban. |
| NewPassword | -email: megadott email cím  -password: az újonnan generált jelszó. | Email cím alapján módosítja a jelszót. |
| NewUser | -username: felhasználónév  -password: jelszó  -name: teljes név  -email: email cím  -letters: hírlevelekre feliratkozik-e | Beilleszt az adatbázisba egy új felhasználót a megadott paramétereknek megfelelően. Jogosultsági szintje automatikusan 1 lesz. |
| AllUsers | - | Lekérdezi az összes felhasználó összes adatát |
| SelectUser | -userId: felhasználó azonosítója | Lekérdezi egy felhasználó összes adatát, a megadott azonosító szerint. |
| CheckUsed | -username: felhasználónév  -email: email cím | Lekérdezi, az összes olyan felhasználó adatát, akinél a megadott felhasználónév, vagy email cím szerepel. |
| NewFavorite | -userId: felhasználó azonosítója  -carId: autó azonosítója | A favorites táblába beilleszt egy új sort a paramétereknek megfelelőlen. |
| RemoveFavorite | -userId: felhasználó azonosítója  -carId: autó azonosítója | A favorites táblábl kitörli a paramétereknek megfelelő sort. |
| Favorites | -userId: felhasználó azonosítója | Lekérdezi a megadott felhasnzálóhoz tartozó összes kedvencnek jelölt autót |
| CheckFavorite | -userId: felhasználó azonosítója  -carId: autó azonosítója | Lekérdezi a paramétereknek megfelelő autókat a favorites táblából. |
| Filter | -make: gyártó  -models: tömb, modelleket tartalmazza  -years: tömb, az évjárat alsó és felső határát tartalmazza  -prices: tömb, az ár alsó és felső határát tartalmazza  -conditions: tömb, állapotokat tartalmaz  -colors: tömb, színeket tartalmaz  -fuels: tömb, üzemanyag típusokat tartalmaz  -types: tömb, kiviteleket tartaémaz  -transmissions: tömb, váltókat tartalmaz  -drives: tömb, meghajtásokat tartalmaz | A megadott paramétereknek megfelelően visszaadja az összes olyan autó összes adatát, amelyek megfelelnek a feltételeknek. |

**View réteg:**

A html oldal előállításáért (render) az EJS template engine a felelős.

**A modul helye:** [telepítési mappa]\views mappa

**Réteg felelőssége:**

A JSON formátumban megkapott paraméterek és az EJS fájlban meghatározott sablon alapján előállítja a kimeneti HTML formátumot. A generált HTML oldal kódját visszaadja az útválasztási rétegnek.

**A réteg komponensei:**

|  |  |  |
| --- | --- | --- |
| **Neve** | **Szükséges bemenet** | **Funkció** |
| 30scroll.ejs | -list: tömb, ami tartalmazza az autók adatait | Betölti annyi autó kártyáját, amennyit megkap |
| accounts.ejs | -list: tömb, mely tartalmazza a felhasználók adatait | Felhasználók listájának megtekintése |
| allList.ejs | - | Összes autó listájának a váza, majd a 30scroll.ejs és „/cars/:page” url segítségével, tölti be az autókat harmincasával, ahogyan görget le a felhasználó |
| car.ejs | -images: tömb, mely tartalmazza az autóhoz tartozó képek neveit  -list: tömb, melynek egy eleme van, mely az adott autóhoz tartozó adatokat tartalmazza  -favourite: egy változó mely azt tartalmazza, hogy az autó hozzá van-e adva a felhasználó kedvenceihez, alapesetben hamis | Autóhoz tartozó adatok megjelenítése |
| editProfile.ejs | -lastname: változó, tartalmazza a felhasználó vezetéknevét  -firstname: felhasználó keresztneve  -email: felhasználó e-mail címe  -username: felhasználó felhasználóneve | Itt tudja a felhasználó megtekinteni az adatait |
| emailSent.ejs | -email: az email cím, amire el lett küldve az email. | Itt a felhasználó kap egy megerősítést, hogy az email el lett küldve, melyben szerepel az új jelszó. |
| error.ejs | -msg: objektum, mely tartalmazza a hiba részleteit | Ha a szerver hibára fut, itt megjeleníti |
| favourites.ejs | -list: tömb, mely tartalmazza a felhasználóhoz tartozó kedvenc autókat | Megjeleníti a felhasználó kedvence autóit |
| footer.ejs | - | A láblécet tartalmazza |
| getEmail.ejs | -emailError: alapértelmezett érteke hamis, akkor lesz igaz, ha a felhasználó rossz email címet ad meg, amely nem szerepel az adatbázisban | Egy felület, ahol az új jelszó kéréséhez a felhasználó meg tudja adni a jelszavát. |
| icon.ejs | - | A weboldalhoz tartozó ikon megjelenítéséért felel |
| index.ejs | -specials: tömb, mely a különleges ajánlatok autót tartalmazza | A kezdőlapot jeleníti meg. |
| login.ejs | -loginError: alapesetben hamis, akkor lesz igaz, ha a felhasználó rossz adatokat adott meg bejelentkezésnél | Megjeleníti a bejelentkezési oldalt. |
| navbar.ejs | -allPage: ha az Összes oldalon van a felhasználó, akkor igaz, amúgy hamis  -searchPage: ha a Keresés oldalon van a felhasználó, akkor igaz, amúgy hamis  -loggedIn: Be van-e jelentkezve a felhasználó, ha igen igaz, amúgy hamis  -username: A bejelentkezett felhasználó felhasználóneve  -permission: A bejelentkezett felhasználó jogosultsági szintjét tartalmazza  -loginPage: ha a Bejelentkezés fülön van a felhasználó, akkor igaz, amúgy hamis | A navigációs sáv megjelenítésért felel. |
| profile.ejs | -list: tömb, mely egy felhasználó adatait tartalmazza | Megjeleníti a felhasználóhoz tartozó adatokat. |
| registration.ejs | -usedData: változó, alap esetben hamis, akkor lesz igaz, ha a felhasználónév vagy email cím már foglalt  -lastname: változó, a regisztráló felhasználó vezetéknevét tartalmazza  -firstname: változó, a regisztráló felhasználó keresztnevét tartalmazza  -password: változó, a regisztráló felhasználó jelszavát tartalmazza  -passwordAgain: változó, a regisztráló felhasználó ismételten beírt jelszavát tartalmazza | Megjeleníti a regiszrációhoz szükséges űrlapot. Hibás adatok esetén, a felhasználónak nem kell mégegyszer megadnia minden adatot. |
| search.ejs | -modelsWithMake: egy tömb, mely a kiválasztott márkához tartozó modelleket tartalmazza  -makes: összes gyártót tartalmazó tömb  -minYear: a legkisebb évjáratú autó évjárata  -maxYear: a legkésőbbi évjáratú autó évjárata  -maxPrice: a legdrágább autó ára  -conditions: az összes állapotot tartalmazó tömb  -colors: az összes színt tartalmazó tömb  -fuels: az összes üzemanyagot tartalmazó tömb  -types: az összes kivitelt tartalmazó tömb  -transmissions: az összes váltót tartalmazó tömb  -drives: az összes meghajtást tartalmazó tömb | Az autók szűréséhez szolgáló űrlapot tölti be. |

## Választott technológiák

Miért ezt választottam?

* Web-es technológia: Több felhasználó tudja egyszerre használni (akár több száz). Aszinkron programozási technikát is azért választottuk, mert nagy számú felhasználó esetén nem várja meg az adatbázis válaszát, hanem egyidejűleg futtatja a lekérdezéseket. Másik előnye, hogy az interneten keresztül bárhonnan el lehet érni a szolgáltatást.
* Desktop: Korlátozott számú felhasználó használja csak az adminisztrátori funkciókat (maximum 2-3), de ez is több felhasználós, így egyszerre többen is hozzá tudnak férni az adatbázishoz. Viszont elegendő az, ha a lokális hálózaton elérhető.

## Asztali alkalmazás:

* Fejlesztői környezet:
  + Visual Studio
  + MySQL for Visual Studio
* Programnyelv: C#
* Operációs rendszer: Windows
* Keretrendszer: .Net framework 4.8 Advanced Services
* ADO.NET programkönyvtár
* Connector/NET for MySQL 8.0.28 x86

## Web-es app:

* Fejlesztői környezet:
  + NodeJS
* Programnyelv: Javascript
* Operációs rendszer: Windows/Linux

## Külső könyvtárak:

* **express:** *~4.16.1*, webkiszolgáló keretrendszere
  + Az express egy webalkalmazáskeretrendszer, amely széles körű funkciókat kínál webes és mobilalkalmazások létrehozásához. Segít a szerverek és útvonalak kezelésében.
* **express-session:** *^1.17.3*, session kezelés
  + egy HTTP szerveroldali keretrendszer, amelyet munkamenet köztes szoftver létrehozására és kezelésére használnak.
* **bootstrap:** *^5.2.3*, bootstrap kezelés
  + A bootsrap egy népszerű CSS-keretrendszer
* **cookie-parser**: *~1.4.4*,
  + A cookie-parser egy köztes szoftver, amely elemzi az ügyfél kérési objektumához csatolt cookie-kat.
* **cors:** ^2.8.5, Cross-Origin Resource Sharing
  + Segítségével a front-end tud kéréseket küldeni egy külső back-end szervernek. Az emailjs-hez kell.
* **debug:** *~2.6.9*, fejlesztéshez
  + Olyan folyamat amely azonosítja a hibákat.
* **ejs:** *^3.1.8*,
  + Ez egy egyszerű sablonnyelv/motor, amely lehetővé teszi a felhasználó számára, hogy egyszerű javascripttel HTML-t generáljon.
* **https:** *^1.0.0*,
  + https protokoll használatatát teszi lehetővé
* **http-errors:** *~1.6.3*,
  + Http hibák eredetének kinyomozásában segít.
* **morgan:** *~1.9.1*,
  + A Morgan egy HTTP kérés szintű köztes szoftver. Ez egy nagyszerű eszköz, amely naplózza a kéréseket néhány egyéb információval együtt a konfigurációjától és a használt előre beállított értéktől függően. Nagyon hasznosnak bizonyul a hibakeresés során, és akkor is, ha naplófájlokat szeretne létrehozni.
* **mysql2:** *^3.0.1*,
  + A MySQL adatbázis eléréséhez szolgáló illesztő könyvtár.
* **node-fetch:** ^2.6.9
  + fetch API kérés indítása JavaScriptből.
* **nodemon:** *^2.0.20*,
  + Amikor az éppen futó kódunkon végzünk el változtatásokat, a mentést követően nem lépnek érvénybe ezen változtatások. Minden egyes alkalommal újra kell indítani a teljes programot. Komplexebb applikációk esetén már egy csomó időt elvesz a fejlesztőktől. A probléma megoldására alkották meg a Nodemon modult, mely minden módosítás után automatikusan újraindítja a szervert.
* **path:** *^0.12.7*,
  + olyan karakterlánc, amely egy címtárszerkezetben lévő hely egyedi azonosítására szolgál.
* **fs:** *^0.0.1-security*,
  + mappákból való fájl beolvasást teszi lehetővé

## Függőségek

MySQL Server 8.0.27 adatbázis szerver

### Minimális hardver követelmény:

* Processzor: 1 gigahertzes (GHz) vagy gyorsabb processzor
* RAM: 1 gigabájt (GB) 32 bites rendszerhez vagy 2 GB 64 bites rendszerhez
* Merevlemez-terület: 16 GB 32 bites operációs rendszerhez vagy 20 GB 64 bites operációs rendszerhez
* Videokártya: A DirectX 9-es vagy újabb verziójával kompatibilis, WDDM 1.0 illesztőprogrammal
* Kijelző: 1024x600 vagy nagyobb felbontás

### Ajánlott hardver követelmény:

* Processzor: 2.5 gigahertzes (GHz) vagy gyorsabb processzor
* RAM: 2 gigabájt (GB) 32 bites rendszerhez vagy 4 GB 64 bites rendszerhez
* Merevlemez-terület: 20 GB 32 bites operációs rendszerhez vagy 30 GB 64 bites operációs rendszerhez
* Videokártya: A DirectX 9-es vagy újabb verziójával kompatibilis, WDDM 1.0 illesztőprogrammal
* Kijelző: 1920x1080 vagy nagyobb felbontás

### Desktop app:

* Operációs Rendszer: Windows 10
* Keretrendszer: .NET framework 4.8 Advanced Services

### Web App:

* Windows/Linux
* NodeJS
* Teljes mappa struktúra
* package.json állományban megtalálható függőségek telepítése (npm install vagy a mellékelt node\_modules mappa)
* tanúsítvány, amennyiben van saját

# Modulok

A rendszer két fő modulból áll:

* Desktop alkalmazás
* WEBkiszolgáló

## Áttekintés

A szofvter architektúrája az alábbi modulokra, komponensekre oszlik.

## Modulok részletei

### Desktop alkalmazás – adminisztrációs modul

Feladata: rendszer-adminisztrációs feladatok megvalósítása

* Törzsadatok modul:
  + Állapotok:
  + Autók
  + Gyártók
  + Kivitelek
  + Meghajtások
  + Színek
  + Üzemanyagok
  + Váltók
  + Speciálisok
* Felhasználók modul
  + Felhasználók
  + Kedvencek

### Webes alkalmazás

Feladata: Távoli felhasználók kiszolgálása

* Felhasználókezelés modul:
  + Regisztráció
  + Bejelentkezés
  + Profil
  + Adatok módosítása
  + Kijelentkezés
  + Kedvencek kezelése
* Kereső
* Termék megjelenítő

# Perzisztencia

Adattárolás:

* MySQL adatbázis
* Képek tárolása a fájlrendszer mappáiban

## A tárolás választott módja

**MySQL:**

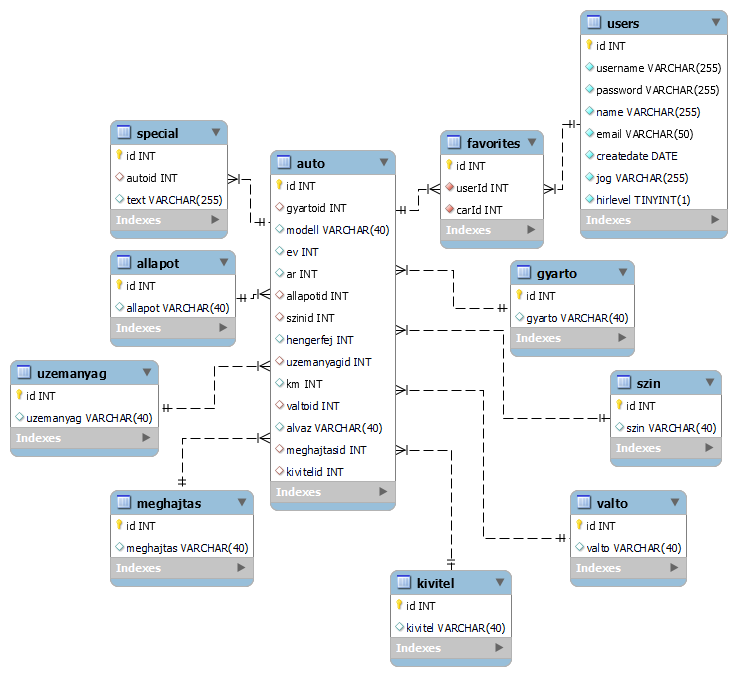
MySQL bemutatása

**Fájlok/képek tárolása:**

Az autók képeit a webes alkalmazás telepített mappájában tároljuk, helye: [telepítési mappa]\public\src\database\[autó azonosítója]\

## Koncepcionális adatmodell

ERD diagram:



Adatkapcsolatok:

Táblák:

auto tábla:

….

…..

# Folyamatok

Autó bemutatása (különlegesek + keresés):

Bejelentkezés:

Regisztráció:

Kijelentkezés:

Felhasználó adatainak megjelenítése:

Kedvencekhez adás:

…..

# Interfészek

*A rendszer külső interfészeit ismertető fejezet. Ebben a fejezetben bemutatjuk a rendszer külső interfészeit, az egyes interfészeken alkalmazott protokollokat, adat és üzenetformátumot (adatstruktúra és reprezentáció (pl. XML, JSON)), az interfészek funkcióit, az alkalmazott integrációs mintát (pl. szinkron kérés-válasz, egyirányú, aszinkron-callback, push/pull), az interfészeken alkalmazott hibakezelési elveket, megismételt küldés módját, többszöri küldés esetén a többszöri feldolgozás elleni védelmet, túlterhelés elleni védelmet (üzenetméret és üzenet volumen esetére is).*

# Biztonság

*A rendszer koncepcionális jogosultsági modelljét és a fő biztonsági elveket ismertető fejezet. Ebben a fejezetben a rendszer biztonsági modelljének műszaki aspektusait ismertetjük. Be kell mutatni a jogosultság kezelés koncepcionális modelljét (pl. funkció alapú jogosultság kezelés, adatkör alapú jogosultság kezelés), ennek műszaki megvalósítási tervét (mely rétegekben milyen jogosultság kikényszerítési pontok vannak). Ismertetni kell a biztonsági funkciójú logolások műszaki megvalósítását, granularitását (pl. képernyő, funkció vagy adat szintű). Ha tárol a rendszer jelszót vagy egyéb érzékeny adatot, akkor ismertetni kell ezek tárolásának technológiai tervét. Amennyiben releváns, az bejövő, kimenő adatok részleges törlésének (data redaction) illetve anonimizálásának elveit, megvalósításának módját is ismertetni kell. Ha releváns a rendszer szempontjából, akkor itt kell bemutatni az alkalmazott speciális biztonsági funkciókat is, pl. négyszem elv vagy pontszámokon alapuló súlyozott jogosultsági megoldások, két faktoros authentikáció.*

# Üzemeltetést támogató képességek

*A rendszer azon képességeit bemutató fejezet, amelyek a hatékony üzemeltetést támogatják. Ebben a fejezetben kell ismertetni az összes üzemeltetést támogató képességet.*

## Konfiguráció

*A rendszer konfiguráció kezelését bemutató fejezet. Be kell mutatni a rendszer konfiguráció kezelésének módját, műszaki megvalósítását. A rendszer konfigurálható paramétereit milyen technológia megoldás(ok)kal lehet kezelni (pl. fájl alapú, adatbázis alapú, rendszerbe fordított paraméterek). A konfigurációs elemek életbe lépésének módját is be kell mutatni (pl. a módosítás mentésétől effektív, a módosítás mentésétől adott időn belül effektív, a rendszer újraindításától effektív, esetleg ezek kombinációja).*

## Monitorozás

*A rendszer monitorozó rendszerekkel való integrációját ismertető fejezet. Ebben a fejezetben kell bemutatni, hogy milyen technológiai megoldás(ok) állnak majd rendelkezésre a rendszer monitorozására (pl. SNMP trap, SOAP vagy REST végpontok, fájl alapú megoldás, adatbázis tábla alapú megoldás). Be kell mutatni a monitorozási képesség granularitását is (pl. csak rendszerszintű paraméterek, vagy modul esetleg még kisebb felbontású paraméterek is monitorozhatóak).*

## Logolás

*A rendszer logolási képességeit bemutató fejezet. Itt kell leírni, hogy a rendszer milyen elvek mentén, milyen műszaki megoldásokkal, milyen strukturájú log üzeneteket generál (pl. stdout-ra, JSON formátumban). Meg kell adni a logolás részletezettségének konfigurálási módját (pl. WARN vagy DEBUG szintek közötti váltás módja), és a részletezettség állításának granularitását is (pl. csak rendszer szinten, vagy modul, esetleg osztály, szolgáltatás, stb. szinten).*

## Hibaelhárítás

*A rendszer hibaelhárításhoz nyújtott támogató funkcióit ismertető fejezet. Ez a fejezet mutatja be, hogy a rendszer milyen képességeket biztosít a hibakeresés és hibaelhárítás támogatása érdekében (pl. képernyőn megjelenő hibajelenségek logbejegyzésekkel való összekapcsolásának képessége, üzleti tranzakció azonosítók hordozása az összes architekturális rétegben az üzenet kontextusban).*