JEGYZŐKÖNYV

Web technológiák 1 gyakorlat

Féléves feladat

Autókereskedés

Készítette: **Kelemen Ádám**

Neptun Kód: **DBO8MH**

Dátum: **2025. november**

Miskolc, 2025

1. Tartalomjegyzék

2. Bevezetés

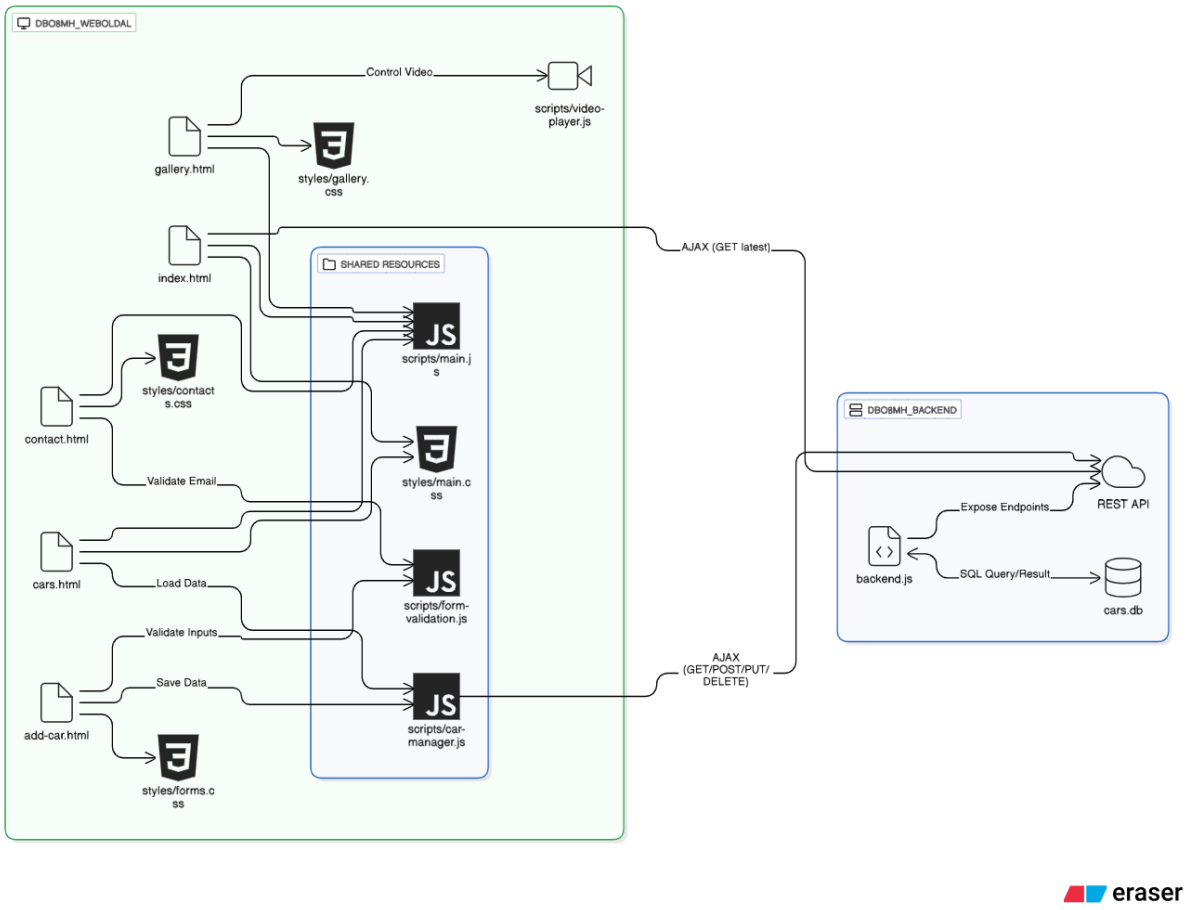
A féléves feladat keretében egy modern, online gépjármű-hirdetési portált terveztem és valósítottam meg. A projekt elsődleges célja egy olyan dinamikus webalkalmazás létrehozása volt, amely digitális piacteret valósít meg az autók meghirdetéséhez, lehetővé téve a járművek részletes adatainak (márka, típus, teljesítmény, ár) strukturált megjelenítését és kezelését. A rendszer a klasszikus apróhirdetési oldalak (mint például a Használtautó.hu vagy Mobile.de) alapvető funkcionalitását modellezi, modern webes technológiák segítségével integrálva a hirdetések böngészését és az új ajánlatok feltöltését.

A webalkalmazás architektúrája a "Full-Stack" fejlesztési elveket követi, élesen szétválasztva a felhasználói felületet (Frontend) és az adattárolás logikáját, vagyis a Backendet. A kliensoldal célja a maximális felhasználói élmény biztosítása: a látogatók egy reszponzív, Bootstrap 5 keretrendszerre épülő felületen böngészhetnek a hirdetések között. A felület interaktivitását JavaScript és jQuery biztosítja, amely lehetővé teszi az autók aszinkron betöltését (AJAX), a valós idejű szűrést és rendezést anélkül, hogy az oldalnak újra kellene töltődnie. A rendszer támogatja a multimédiás tartalmakat is, beleértve a hirdetésekhez tartozó képek megjelenítését és egy egyedi videólejátszóval is el van látva a weboldal.

A projekt szerveroldali megvalósítása Node.js környezetben, Express.js keretrendszerrel készült, amely egy REST API interfészen keresztül kommunikál a frontenddel. Ez a réteg felelős a hirdetések adatainak fogadásáért, validálásáért és tartós tárolásáért egy SQLite adatbázisban, amely közvetlen a backend mappájában tárolódik A hirdetésfeladás folyamata kritikus része a rendszernek: a felhasználók egy űrlapon keresztül rögzíthetnek új autókat, ahol a rendszer előre definiált szabálykészletek és reguláris kifejezések (Regex) segítségével ellenőrzi a bevitt adatokat (pl. évjárat intervallum, negatív ár szűrése, email formátum), ezzel biztosítva a hirdetések minőségét.

3. A rendszer architektúrája és fájlszerkezete

A projekt moduláris felépítésű, különválasztva a Backend és a Frontend komponenseket a jobb átláthatóság érdekében.



1. ábra: Fájlszerkezet és függőségek

A diagram szemlélteti a teljes alkalmazás fájlarchitektúráját és a komponensek közötti kapcsolatokat. A struktúra két fő tárolóra oszlik:

- **Backend:** Tartalmazza a szerveroldali logikát (backend.js), a csomagkezelő konfigurációit (package.json), valamint az adatbázist (cars.db). Látható, hogy a szerver közvetlen kapcsolatban áll az adatbázissal.
- **Frontend:** Magába foglalja a kliensoldali elemeket. A diagramon látható nyilak jelölik az erőforrás-hivatkozásokat azaz, hogy melyik HTML nézet mely JavaScript és CSS fájlokat tölti be.

4. A felhasználói felület és a funkciók bemutatása

4.1 Főoldal (index.html)

Az alkalmazás belépési pontja. A felső navigációs sáv (Navbar) minden oldalon elérhető, és biztosítja a könnyű átjárást a menüpontok között. A "hero" szekció alatt dinamikusan betöltődnek a legújabb autók kártyái, amelyeket JavaScript segítségével kérünk le a szerverről.

4.2 Autólista és szűrés (cars.html)

Ez az oldal jeleníti meg a teljes kínálatot.

- **Keresés és Rendezés:** A felhasználó szűrhet autókra név vagy márka alapján, valamint rendezheti a találatokat ár (növekvő/csökkenő), név vagy évjárat szerint.
- **Statisztika:** Az oldal alján dinamikus kártyák mutatják az aktuális készlet statisztikáit (Összes autó, Átlagár, Átlag lóerő).
- **Részletek:** A "Részletek" gombra kattintva egy modális ablak (Modal) ugrik fel az autó minden adatával, ahol lehetőség van a szerkesztésre vagy törlésre is.

4.3 Adminisztráció és új autó felvétele (add-car.html)

Ezen a felületen lehet új hirdetést rögzíteni.

- **Űrlap:** A beviteli mezők (Márka, Típus, Évjárat, Lóerő, Ár, Szín) validációval vannak ellátva.
- **Képfeltöltés:** A rendszer támogatja képek feltöltését, amelyeket Base64 formátummá alakít át a küldés előtt.
- **Előnézet:** A "Előnézet" gomb megnyomásával a felhasználó láthatja, hogyan fog kinézni a hirdetés, mielőtt véglegesítené.

4.4 Galéria és Videólejátszó (gallery.html)

A galéria oldalon a multimédiás tartalom is megtalálható.

- **Egyedi Videólejátszó:** A VideoPlayer osztály által vezérelt felület, amely saját gombokkal (Lejátszás, Hangerő, Léptetés, Teljes Képernyő, némítás) és folyamatjelző sávval rendelkezik, felülbírálva a beágyazott https://www.youtube.com alapértelmezett kezelőszerveit.

4.5 Kapcsolat (contact.html)

A kapcsolatfelvételi űrlap lehetőséget ad üzenet küldésére. Sikeres validáció esetén a rendszer egy letölthető „report_aktuálisDátum_aktuálisIdő.txt” fájlt generál, amely tartalmazza az aktuálisan kitöltött űrlap mezőit, E-mail” nézetben, illetve JSON és XML formátumban is.

```
*----------------------------------------------------------------------*
                        AUTÓKERESKEDÉS - KAPCSOLATFELVÉTEL
*----------------------------------------------------------------------*

Feladó: Kelemen Ádám
Email: adamkelemen@gmail.com
Telefon: +36301211233

Tárgy: Hamis hírdetés
Autómodell: Opel
Sürgősség: Közepes

Preferált kapcsolatfelvétel: Email, Telefon

ÜZENET:
Baj van a hirdetéssel

✓ Feliratkozott a hírlevélre

Időbélyeg: 2025. 11. 26. 15:17:10
```

2. ábra: E-mail Példa

```
*------------------------------------------------------------------------*
                                JSON ADATOK
*------------------------------------------------------------------------*

{
  "contactRequest": {
    "personalInfo": {
      "firstName": "Ádám",
      "lastName": "Kelemen",
      "email": "adamkelemen@gmail.com",
      "phone": "+36301211233"
    },
    "requestDetails": {
      "subject": "test_drive",
      "subjectText": "Hamis hírdetés",
      "carModel": "Opel",
      "urgency": "medium",
      "urgencyText": "Közepes",
      "contactMethods": [
        {
          "method": "email",
          "methodText": "Email"
        },
        {
          "method": "phone",
          "methodText": "Telefon"
        }
      ],
      "message": "Baj van a hirdetéssel",
      "newsletter": true
    },
    "metadata": {
      "timestamp": "2025-11-26T14:17:10.782Z",
      "formattedTimestamp": "2025. 11. 26. 15:17:10"
    }
  }
}
```

3. ábra: Kimentett JSON formátumban

```
*------------------------------------------------------------------------*
                                XML ADATOK
*------------------------------------------------------------------------*

<?xml version="1.0" encoding="UTF-8"?>
<contactRequest>
    <personalInfo>
        <firstName>Ádám</firstName>
        <lastName>Kelemen</lastName>
        <email>adamkelemen@gmail.com</email>
        <phone>+36301211233</phone>
    </personalInfo>
    <requestDetails>
        <subject code="test_drive">Hamis hírdetés</subject>
        <carModel>Opel</carModel>
        <urgency level="medium">Közepes</urgency>
        <contactMethods>

            <method type="email">Email</method>

            <method type="phone">Telefon</method>

        </contactMethods>
        <message>Baj van a hirdetéssel</message>
        <newsletter>true</newsletter>
    </requestDetails>
    <metadata>
        <timestamp>2025-11-26T14:17:10.782Z</timestamp>
        <formattedTimestamp>2025. 11. 26. 15:17:10</formattedTimestamp>
    </metadata>
</contactRequest>
```

4. ábra: Kimentett XML formátumban

5. Kódstruktúra és technikai megvalósítás

5.1. Adatkezelés (CarManager osztály)

A scripts/car-manager.js fájlban található CarManager osztály felelős az adatokkal végzett műveletekért.

- **Konstruktor:** Inicializálja a cars tömböt és a bázis URL-t (http://localhost:8080).
- **loadCars():** AJAX GET kéréssel lekéri az autók listáját a szerverről, majd meghívja a megjelenítő függvényeket.
- **CRUD műveletek:**
  - **addCar(carData):** POST kérés új autó mentéséhez.
  - **updateCar(carId, carData):** PUT kérés meglévő adat módosításához.
  - **deleteCar(carId):** DELETE kérés autó törléséhez.
- Dinamikus HTML generálás: A createCarCard metódus állítja össze a Bootstrap kártyák HTML kódját a beérkező JSON adatokból.

```
createCarCard(car) {
        const imageSrc = car.image ? `${car.image}` : '...';
        return `
                <div class="col-md-4 mb-4 fade-in">
                        <div class="card car-card h-100">
                         <img src="${imageSrc}" ...>
                                <div class="card-body">
                                        <h5 class="card-title">${car.brand} ${car.name}</h5>
                                        ...
                                </div>
                        </div>
                </div> `;
}
```

5. ábra: Dinamikus kártya generálása (részlet)

5.2. Űrlap validáció (FormValidator osztály)

A scripts/form-validation.js fájl tartalmazza az ős FormValidator és a leszármaztatott CarFormValidator osztályokat.

- **Öröklődés:** A CarFormValidator kiterjeszti az alap ellenőrzést (kötelező mezők, email formátum) specifikus szabályokkal:
- Lóerő: 50 és 2000 között.
- Gyártási év: 1900 és az aktuális év között.
- Ár: Nem lehet negatív és maximum 100,000,000,000 lehet az értéke.
- **Eseménykezelés:** A validáció blur (mező elhagyása) és submit (küldés) eseményekre fut le. Hiba esetén a mező piros keretet kap, és megjelenik a hibaüzenet.

5.3. Média kezelés (VideoPlayer osztály)

A scripts/video-player.js fájl valósítja meg a videó vezérlését.

- A DOM elemeket (gombok, csúszkák) eseményfigyelőkkel látja el.
- A timeupdate esemény segítségével frissíti a folyamatjelző sávot (progressBar) valós időben.

6. Szerveroldali háttérrendszer

A rendszer hátterét egy **Node.js** alapú szerver biztosítja, amely az **Express** keretrendszert használja a HTTP kérések kezelésére. A szerver a 8080-as porton hallgat, és támogatja a **CORS** (Cross-Origin Resource Sharing) mechanizmust, lehetővé téve a böngészőből érkező kérések kiszolgálását.

6.1. Adatbázis (SQLite)

Az adatok tárolását egy szervermentes, fájlalapú **SQLite** adatbázis (**cars.db**) végzi. A cars tábla szerkezete a következő mezőket tartalmazza:

- **id:** Egyedi azonosító (elsődleges kulcs, automatikus növekmény).
- **brand, name:** Márka és típus (szöveges mezők).
- **manufacture_year, horsepower:** Évjárat és teljesítmény (egész számok).
- **price:** Ár (decimális).
- **color:** Szín (szöveg).
- **image:** Kép tárolása bináris (BLOB) formátumban.
- **created_at:** Létrehozás dátuma (automatikus időbélyeg).

6.2. REST API Végpontok

- **GET /api/cars:** Lekéri az összes autót az adatbázisból a létrehozás ideje szerinti csökkenő sorrendben. A BLOB formátumban tárolt képeket Base64 stringgé alakítja át a JSON válaszban való küldéshez.
- **GET /api/cars/:id:** Egy konkrét autó részletes adatait adja vissza ID alapján. Ha az autó nem található, 404-es hibakódot küld.
- **POST /api/cars:** Új autó felvételét végzi. A kérés törzsében (body) érkező adatokat validálja (márka és név kötelező), a Base64 formátumú képet pedig Buffer-ré alakítja az adatbázisba írás előtt.
- **PUT /api/cars/:id:** Meglévő autó adatainak módosítását teszi lehetővé.
- **DELETE /api/cars/:id:** Törli a megadott azonosítójú autót a rendszerből.

7. Források

7.1 Backendhez felhasznált források

- **Node.js hivatalos dokumentáció:** https://nodejs.org/en/docs/
- **Express.js keretrendszer dokumentáció:** https://expressjs.com/
- **SQLite adatbázis dokumentáció:** https://www.sqlite.org/docs.html
- **Node-SQLite3 modul (Github):** https://github.com/TryGhost/node-sqlite3

7.2 Frontendhez felhasznált források

- **Bootstrap 5 dokumentáció és komponensek:**

https://getbootstrap.com/docs/5.0/getting-started/introduction/

- **jQuery API dokumentáció:** https://api.jquery.com/
- **Google Maps Embed API:**

https://developers.google.com/maps/documentation/embed/get-started

- **YouTube IFrame Player API:**

https://developers.google.com/youtube/iframe_api_reference