ZÁRÓDOLGOZAT

**Rátkai Martin**

**5/13SzOFT**

**BAJAI SZC TÜRR ISTVÁN TECHNIKUM**

**Szoftverfejlesztő**

ZÁRÓDOLGOZAT

autóalibi alkatrészek

**Rátkai Martin**

2024

**NYILATKOZAT A ZÁRÓDOLGOZATRÓL**

Alulírott (név) tanuló

**kijelentem, hogy** című záródolgozatomat (nyomtatott és elektronikus formában) a Bajai SZC Türr István Technikumának pedagógusai és tanulói:

* + felhasználhatják (pl. hivatkozás alapjául, olvasótermi használatra) későbbi munkájukhoz a szerzői jogok tiszteletben tartása mellett).
  + nem használhatják fel (titoktartási nyilatkozat csatolása mellett).

Ugyanakkor kijelentem, hogy a záródolgozat ***saját munkám eredménye*.**

Baja, 2024.

aláírás

1. Bevezetés 1

1.1. Probléma ismertetése 1

2. Felhasználói dokumentáció 2

2.1. Általános specifikáció 2

2.2. Webalkalmazás használata 2

2.3. Rendszerigény 2

2.4. Első indítás 2

2.5. Felhasználói felület elemei, funkciói 2

3. Fejlesztői dokumentáció 8

3.1. Témamegjelölés 8

3.2. Adatbázis terv 8

3.3. Felhasznált eszközök, technológiák 15

3.4. Függőségek 19

3.5. Forráskód 20

3.6. Tesztelési dokumentáció 43

4. Összegzés 44

4.1. Fejlesztési lehetőségek 44

4.2. Személyes tapasztalatok 44

5. Felhasznált források 45

6. Ábrajegyzék 46

# Bevezetés

## Probléma ismertetése

Az autószerelőknek otthonról alkatrészeket rendelni számos előnnyel jár, különösen a hagyományos bolti vásárláshoz képest. Az online platformok lehetővé teszik az árak és a termékek közötti összehasonlítást, ami segíti az autószerelőket abban, hogy a legjobb áron és minőségben szerezzék be a szükséges árucikkeket. Sok esetben az autószerelőknek speciális vagy ritka alkatrészekre van szükségük, amelyek nem feltétlenül találhatók meg a helyi üzletekben. Az online platformok gyakran kínálnak ilyen speciális alkatrészeket is, így az autószerelők könnyen hozzájuthatnak ezekhez. A hagyományos bolti vásárlás során időt kell szánniuk az üzletek keresésére, utazásra és végül az alkatrészek kiválasztására. Az otthonról történő rendelés egyszerű és gyors folyamat, amely megszünteti az időveszteséget. Az autóalkatrészek gyorsabb beszerzését segíti az „AutóAlibi Alkatrészek”. Egy olyan weboldal, ahol regisztrált felhasználóként tudunk online vásárlást végezni. Mindez HTML, CSS, JavaScript, AngularJS és Bootstrap segítségével készült.

Az oldal célja, hogy megkönnyítse az autószerelő dolgát és időt spóroljon neki. Ez egy olyan rendszer, amit csak regisztrált felhasználók tudnak használni. A regisztráció bárki számára nyitott.

A képen szöveg, képernyőkép, tervezés látható

Automatikusan generált leírás

1. ábra - Bejelentkezés az oldalra

# Felhasználói dokumentáció

Gyorsabb és egyszerűbb megoldást kínál azoknak az autószerelőknek, akik a hatékonyság, kényelem és időmegtakarítás céljából nem a hagyományos bolti vásárlási módszereket választják.

## Általános specifikáció

A „webshop” egy olyan felületet biztosít, ahol a felhasználók gyorsan és könnyen tudják elvégezni az alkatrészek rendelését (több különböző rendelés leadása egyszerre, jobb átláthatóság, rendezhetőség)**.**

## Webalkalmazás használata

Kizárólag regisztrált felhasználók tudják használni a weboldalt. Az oldal több részre van bontva. A weboldal megnyitása után először egy bejelentkezés fogad minket. Ezen a bejelentkezés részen fel van tüntetve, hogy csak regisztrált felhasználók tudják használni. Miután megtörtént a regisztráció, az oldal automatikusan visszairányít minket a bejelentkezés részre. Itt, ha bejelentkezünk, már magát a főoldalt fogjuk látni. Ezen a felületen a következő műveleteket tudjuk elvégezni: Meg tudjuk nézni magát a rendeléseinket, az üzlet elérhetőségét és magát a webshopot. Egyféleképpen lehet alkatrészre keresni a weboldalon. A keresés egy kereső mező segítségével történik.

## Rendszerigény

Egy olyan számítógép vagy okoseszköz, ami képes webböngészőt kezelni. (például: Firefox, Chrome)

## Első indítás

Az első indítást követően, a felhasználónak létre kell hozni egy saját fiókot, majd ezt követően ez a felhasználó használni tudja a webshopot, leadni és megtekinteni a rendeléseit.

## Felhasználói felület elemei, funkciói

A jobb felső sarok elemei:

* Kosár
* Felhasználónév
* Kijelentkezés

A bal felső sarok elemei:

* Főoldal
* Kapcsolat
* Webshop

**Menüpontok:**

**Főoldal:** A bejelentkezés után az oldal automatikusan a Főoldalra irányít minket. Ezen az oldalon találhatóak meg az eddigi leadott rendeléseink. Ha új felhasználóként lépünk be akkor szinte biztos, hogy a rendeléseink fül üres lesz.

A képen képernyőkép, szöveg, szoftver, multimédia látható

Automatikusan generált leírás

2. ábra - Főoldal

**Kapcsolat:** A "Kapcsolat" elnevezésű menüpont alatt megtalálhatók az AutóAlibi vállalat két fő helyszínének elérhetőségei. Az első helyszín a cég központi irodájának, információi. A második helyszín a logisztikai központ információi.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Automatikusan generált leírás

3. ábra - Kapcsolat fül

Ezen a menüponton keresztül könnyen elérhetők és megismerhetők a vállalat fontosabb elérhetőségei, így az ügyfelek és a partnerek könnyedén kapcsolatba léphetnek az Alibi vállalattal. A címek, telefonszámok, e-mail címek és faxszámok segítenek a hatékony kommunikációban és kapcsolattartásban. (Az itt felsorolt összes adat kitalált, nem léteznek.)

**Webshop:** A "Webshop" fül alatt a látogatók számára elérhetővé válik az AutóAlibi vállalat által kínált nyolc különböző alkatrész kategória. Ezek a kategóriák a következők:

* Szűrők: Ide tartoznak az olajszűrők, légszűrők, üzemanyagszűrők és egyéb gépjármű szűrők, melyek fontosak a jármű rendszeres karbantartásában és működésének fenntartásában.
* Fékberendezés: Ez a kategória tartalmazza a féktárcsákat, fékbetéteket és egyéb fékalkatrészeket, melyek elengedhetetlenek a jármű biztonságos működéséhez.
* Tömítések: Ide sorolhatók az olajtömítések, dugattyúgyűrűk, szeleptömítések és egyéb gépjármű motorjának és rendszerének tömítései, melyek fontosak a szivárgások megelőzésében és a motor hatékony működésében.
* Olajok és folyadékok: Ebben a kategóriában találhatók az olajok, hűtőfolyadékok, fékfolyadékok, sebességváltó olajok és egyéb járműfolyadékok, melyek szükségesek a jármű motorjának és rendszerének megfelelő működéséhez.
* Kuplung: Ide tartoznak a kuplungtárcsák és egyéb kuplungalkatrészek, melyek fontosak a jármű sebességváltójának működéséhez és az autó váltórendszerének hatékonyságához.
* Belső felszerelés: Ebben a kategóriában megtalálhatóak a padlószőnyegek és egyéb belső kiegészítők, melyek növelik a jármű komfortját és esztétikai megjelenését.
* Karosszéria: Ez a kategória tartalmazza a lökhárítók, sárvédők, motorháztetők és egyéb karosszériaelemeket, melyek fontosak a jármű külső védelmében és esztétikai megjelenésében.
* Világítás: Ide sorolhatók a fényszórók, lámpák, irányjelzők és egyéb világítótestek, melyek biztosítják a jármű láthatóságát és biztonságos közlekedését rossz fényviszonyok között.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

4. ábra - Webshop fül

Ezen a webshop felületen a felhasználók könnyedén böngészhetnek és vásárolhatnak a különböző alkatrész kategóriák között, hogy biztosítsák járművek hatékony működését és karbantartását.

**Kosár:** Az AutóAlibi weboldalán található kosár funkció lehetővé teszi a felhasználók számára, hogy egyszerűen és hatékonyan kezeljék a vásárlási folyamatot. A kosárban a felhasználók láthatják az általuk választott termékek részleteit, valamint könnyedén módosíthatják azokat, mielőtt folytatnák a fizetési folyamatot. A kosárban megjelennek a következő információk:

* Termék nevek: Az általuk választott termékek nevei, melyek alapján azonosítható, hogy melyik terméket adták hozzá a kosárhoz.
* Képek: A termékek képei, amelyek segítenek azonosítani és felismerni a kívánt termékeket.
* Mennyiség: A kosárban szereplő egyes termékek mennyisége, hogy lássák, hogy hány darabot szeretnének vásárolni belőlük.
* Ár: Minden termékhez tartozó ár, amelyek összege a kiválasztott mennyiségtől függően változhat.
* Összesített ár: Az összes termék végösszege, amely magában foglalja az összes kiválasztott termék árát és mennyiségét.

Emellett a kosárban lehetőség van a következő műveletek végrehajtására:

* Tétel törlése: Ha a felhasználó úgy dönt, hogy nem kíván egy adott tételt megvásárolni, lehetősége van azt egyszerűen törölni a kosárból.
* Fizetés: Miután a felhasználó elégedett a kosár tartalmával, lehetősége van a fizetési folyamat folytatására, ahol kiválaszthatja a fizetési módokat és befejezheti a tranzakciót.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

5. ábra – Kosár

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

6. ábra - Vásárlói adatok

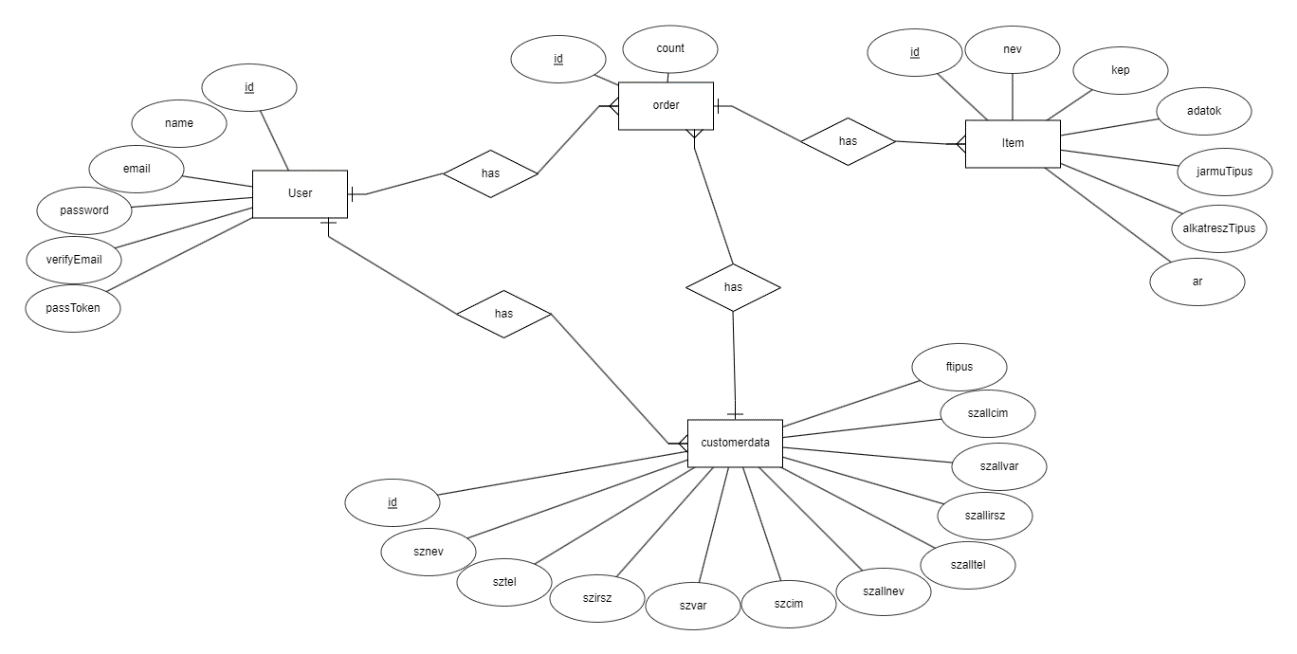
Ez a kosár funkció lehetővé teszi a felhasználók számára, hogy könnyedén és gördülékenyen vásároljanak az AutóAlibi webshopjából, és biztosítsák, hogy a kiválasztott termékek és mennyiségek megfeleljenek az igényeiknek és elvárásaiknak.

# Fejlesztői dokumentáció

## Témamegjelölés

Célom, egy olyan weboldal elkészítése, ami képes megkönnyíteni az autószerelők munkáját.

## Adatbázis terv

Mindenek előtt készítettem egy ERmodellt, hogy könnyebb legyen a dolgom mikor létrehozom a rendes adatbázist.

7. ábra - ERmodell látványterv

Az adatbázis modellt azért készítettem, hogy strukturált és hatékony módon tudjam tárolni és kezelni az alkalmazásban lévő felhasználók, rendelések és termékek adatait. Ezáltal könnyebben lehet keresni, frissíteni és lekérdezni az adatokat az alkalmazás működése során. Az elkészült adatbázisban minden tábla pontosan meghatározott struktúrával és kapcsolatokkal rendelkezik, ami lehetővé teszi az adatok hatékony tárolását és kezelését.

**A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás**

8. ábra - Adatbázis

**Az adatbázis neve:** autoalibi\_adatbazis

**Az adatbázis a következő táblákból áll: users, orders, items, customerdata**

„**users**” ebben a táblában a felhasználó adatai vannak:

`id` int(11): Ez az egyedi azonosító mező, amely egyedi számokat tárol minden felhasználó számára. Ez a mező az adatbázisban való könnyű azonosítás és hivatkozás céljára szolgál. Az int(11) a mező hosszát jelzi, ami annyit tesz, hogy az adott integer értékek 11 karakter hosszúságúak lehetnek.

`name` varchar(50): Ez a mező tárolja a felhasználó nevét. A varchar(50) típusú mező, ami azt jelenti, hogy legfeljebb 50 karakter hosszúságú karakterláncot lehet tárolni benne.

`email` varchar(100): Ez a mező tárolja a felhasználó e-mail címét. Szintén varchar(100) típusú, tehát maximum 100 karakter hosszúságú e-mail címeket lehet tárolni benne.

`password` varchar(255): Ez a mező tárolja a felhasználó jelszavát. A jelszót titkosított formában tárolom a biztonság érdekében. Ez varchar(255) típusú, hogy elegendő hely legyen a titkosított (hashelt) jelszavak tárolására.

`verifyEmail` varchar(100): Ez a mező tárolja a felhasználó e-mail címének ellenőrzésére szolgáló tokent, amely akkor használatos, ha a felhasználó új fiókot hoz létre, és meg kell erősítenie az e-mail címét.

`passToken` varchar(100): Ez a mező tárolja a felhasználó jelszóvisszaállításához használt tokent. Ha a felhasználó elfelejti a jelszavát, akkor egy ilyen token segítségével kérhet jelszóvisszaállítást.

„**orders**” ebben a táblában a felhasználó rendelésének adatai kerülnek tárolásra:

`id` int(11): Ez az egyedi azonosító mező, amely egyedi számokat tárol minden rendelés számára. Az int(11) a mező hosszát jelzi, ami annyit tesz, hogy az adott egész értékek 11 karakter hosszúságúak lehetnek.

`userId` int(11): Ez a mező tárolja azon felhasználók azonosítóját, akik a rendelést leadták. Az int(11) típusú mező, ami egész számokat tárol, 11 karakter hosszúságúak lehetnek.

`customerDataId` int(11): Ez a mező tárolja azon ügyfél adatok azonosítóját, amelyek a rendeléshez kapcsolódnak. Az int(11) típusú mező, ami egész számokat tárol, 11 karakter hosszúságúak lehetnek.

`itemId` int(11): Ez a mező tárolja azon tételek azonosítóját, amelyeket a rendelés tartalmaz. Az int(11) típusú mező, ami egész számokat tárol, 11 karakter hosszúságúak lehetnek.

`count` int(11): Ez a mező tárolja a rendelt tételek darabszámát. Az int(11) típusú mező, ami egész számokat tárol, 11 karakter hosszúságúak lehetnek.

„**items**” ebben a táblában a termékek adatai kerülnek tárolásra:

`id` int(11): Ez az egyedi azonosító mező, amely egyedi számokat tárol minden tétel számára. Az int(11) a mező hosszát jelzi, ami annyit tesz, hogy az adott egész értékek 11 karakter hosszúságúak lehetnek.

`nev` varchar(100): Ez a mező tárolja az alkatrész vagy jármű nevét. A varchar(100) típusú mező, ami azt jelenti, hogy legfeljebb 100 karakter hosszúságú karakterláncot lehet tárolni benne.

`kep` varchar(255): Ez a mező tárolja az alkatrész vagy jármű képének elérési útját vagy. A varchar(255) típusú mező, ami azt jelenti, hogy legfeljebb 255 karakter hosszúságú karakterláncot lehet tárolni benne.

`adatok` varchar(255): Ez a mező tárolja az alkatrész vagy jármű egyéb adatokat. A varchar(255) típusú mező, ami azt jelenti, hogy legfeljebb 255 karakter hosszúságú karakterláncot lehet tárolni benne.

`jarmuTipus` varchar(100): Ez a mező tárolja az alkatrész vagy jármű típusát, ha az adott tétel egy jármű. A varchar(100) típusú mező, ami azt jelenti, hogy legfeljebb 100 karakter hosszúságú karakterláncot lehet tárolni benne.

`alkatreszTipus` varchar(100): Ez a mező tárolja az alkatrész típusát, ha az adott tétel egy alkatrész. A varchar(100) típusú mező, ami azt jelenti, hogy legfeljebb 100 karakter hosszúságú karakterláncot lehet tárolni benne.

`ar` int(11): Ez a mező tárolja az alkatrész vagy jármű árát. Az int(11) a mező hosszát jelzi, ami annyit tesz, hogy az adott egész értékek 11 karakter hosszúságúak lehetnek.

„**customerdata**” ebben a táblában a vásárló személyes adatai kerülnek tárolásra:

`id` int(11): Ez az egyedi azonosító mező, amely egyedi számokat tárol minden ügyfél számára. Ez a mező az adatbázisban való könnyű azonosítás és hivatkozás céljára szolgál. Az int(11) a mező hosszát jelzi, ami annyit tesz, hogy az adott egész értékek 11 karakter hosszúságúak lehetnek.

`userId` int(11): Ez az egyedi azonosító mező, amely egyedi számokat tárol minden felhasználó számára. Ez a mező az adatbázisban való könnyű azonosítás és hivatkozás céljára szolgál. Az int(11) a mező hosszát jelzi, ami annyit tesz, hogy az adott egész értékek 11 karakter hosszúságúak lehetnek.

`sznev` varchar(255): Ez a mező tárolja az ügyfél számlázási nevét. A varchar(255) típusú mező, ami azt jelenti, hogy legfeljebb 255 karakter hosszúságú karakterláncot lehet tárolni benne.

`sztel` varchar(255): Ez a mező tárolja az ügyfél számlázási telefonszámát. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú telefonszámokat lehet tárolni benne.

`szirsz` varchar(255): Ez a mező tárolja az ügyfél számlázási lakcímének irányítószámát. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú irányítószámokat lehet tárolni benne.

`szvar` varchar(255): Ez a mező tárolja az ügyfél számlázási lakcímének városát. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú városneveket lehet tárolni benne.

`szcim` varchar(255): Ez a mező tárolja az ügyfél lakcímét. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú címeket lehet tárolni benne.

`szallnev` varchar(255): Ez a mező tárolja az ügyfél szállítási címének nevét. Szintén `varchar(255)` típusú, tehát maximum 255 karakter hosszúságú neveket lehet tárolni benne.

`szalltel` varchar(255): Ez a mező tárolja az ügyfél szállítási címének telefonszámát. Szintén `varchar(255)` típusú, tehát maximum 255 karakter hosszúságú telefonszámokat lehet tárolni benne.

`szallirsz` varchar(255): Ez a mező tárolja az ügyfél szállítási címének irányítószámát. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú irányítószámokat lehet tárolni benne.

`szallvar` varchar(255): Ez a mező tárolja az ügyfél szállítási címének városát. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú városneveket lehet tárolni benne.

`szallcim` varchar(255): Ez a mező tárolja az ügyfél szállítási címét. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú címeket lehet tárolni benne.

`ftipus` varchar(255): Ez a mező tárolja az ügyfél fizetési típusát. Szintén varchar(255) típusú, tehát maximum 255 karakter hosszúságú fizetési adatot lehet tárolni benne.

Az Orders, Customerdata és Users táblát a felhasználó adataival töltöm fel (alapértelmezetten üres), de az Items táblához saját, valós adatokat készítettem.

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

9. ábra - Items tábla adatok

## Felhasznált eszközök, technológiák

**Eszközök:**

**Visual Studio Code[[1]](#footnote-1):** A Visual Studio Code (VS Code) egy ingyenesen elérhető, könnyen testreszabható és kiterjeszthető szövegszerkesztő és fejlesztőkörnyezet, amelyet a Microsoft fejlesztett ki. A VS Code kiválóan alkalmas különféle programozási nyelvekkel való munkára, például JavaScript, Python, C++, HTML, CSS és sok más nyelv esetén. Ez a szoftver számos hasznos funkciót kínál, például kódszerkesztés közbeni szintaxiskiemelést, automatikus kódbehelyettesítést, hibakeresést és beépített Git integrációt a verziókezeléshez. A Visual Studio Code könnyen testreszabható a felhasználó preferenciái szerint, és széles körű kiterjesztéseket kínál a fejlesztők számára, hogy még hatékonyabban dolgozhassanak a saját projektjeiken.

**Chrome:** A Google által fejlesztett ingyenes webböngésző.

**Technológiák:**

**XAMPP[[2]](#footnote-2):**

Az XAMPP egy ingyenesen elérhető, nyílt forráskódú szoftvercsomag, amely teljes körű fejlesztőkörnyezetet biztosít webszerverek és adatbázisok létrehozásához és kezeléséhez. Az XAMPP az Apache HTTP szerverrel, MySQL adatbázisszerverrel, PHP programozási nyelvvel és Perl interpretterrel rendelkezik, és a legtöbb esetben tartalmazza a phpMyAdmin adatbázis-kezelőt is.

Az XAMPP könnyen telepíthető és használható, és ideális környezetet biztosít a webalkalmazások és weboldalak fejlesztéséhez és teszteléséhez a saját számítógépén, anélkül, hogy szükség lenne valódi szerverre vagy internetkapcsolatra. Az XAMPP alkalmas például PHP alapú CMS-ek (Content Management System), például a WordPress, Joomla vagy Drupal telepítéséhez és teszteléséhez, valamint egyéb webalkalmazások és szkriptek fejlesztéséhez és futtatásához. A fejlesztőknek lehetőségük van lokális környezetben dolgozni és tesztelni a projekteket, mielőtt azokat valóságos szerverre telepítenék vagy közzétennék az interneten. Az XAMPP széles körű funkcionalitása és könnyű használata miatt népszerű választás a webfejlesztők körében.

**Git[[3]](#footnote-3):**

A Git egy elosztott verziókezelő rendszer, amelyet szoftverfejlesztők használnak a kódváltoztatások nyomon követésére és kezelésére. A Git lehetővé teszi, hogy fejlesztők együtt dolgozzanak ugyanazon projektünkön, könnyen összehasonlítsák és egyesítsék a különböző változtatásokat, valamint visszavonják vagy visszaállítsák korábbi verziókat.

**HTML[[4]](#footnote-4):**

Az HTML (HyperText Markup Language) egy olyan szöveg alapú jelölőnyelv, amelyet a weboldalak struktúrájának és tartalmának leírására használnak. Az HTML egy alapvető építőeleme a világháló oldalainak, és lehetővé teszi a szövegek, képek, linkek, táblázatok és más tartalmak strukturált megjelenítését a böngészőkben.

JavaScript: A JavaScript egy dinamikus szkriptnyelv, amelyet a webfejlesztéshez használnak. Az egyik legelterjedtebb programozási nyelv a világhálón, amely lehetővé teszi az interaktív weboldalak létrehozását. A JavaScript segítségével a fejlesztők különféle funkcionalitásokat adhatnak hozzá a weboldalakhoz, például animációkat, formakezelést, adatellenőrzést, dinamikus tartalomgenerálást és sok más dolgot. A böngészőben fut, ami azt jelenti, hogy a felhasználók nem kell külön szoftvert telepíteniük ahhoz, hogy élvezhessék a JavaScript által nyújtott funkciókat.

**CSS[[5]](#footnote-5):**

A CSS (Cascading Style Sheets) egy stíluslapnyelv, amelyet a weboldalak kinézetének és elrendezésének meghatározására használnak. A CSS segítségével a fejlesztők megadhatják, hogy hogyan jelenjenek meg a HTML dokumentumokban található különböző elemek, például szövegek, képek, linkek és táblázatok. A CSS segítségével lehetőség van az elemek színeinek, betűtípusainak, betűméreteinek, margóinak, paddingjének és egyéb vizuális tulajdonságainak meghatározására. A CSS-t gyakran használják a HTML és a JavaScript mellett, hogy interaktív és esztétikus weboldalakat hozzanak létre. A CSS egyszerű nyelv, amely könnyen megtanulható, és lehetővé teszi a fejlesztők számára, hogy különböző stílusokat és elrendezéseket hozzanak létre a weboldalaikon.

**AngularJS[[6]](#footnote-6):**

Az AngularJS egy nyílt forráskódú JavaScript keretrendszer, amelyet elsősorban dinamikus webalkalmazások készítésére használnak. Az AngularJS lehetővé teszi a fejlesztők számára, hogy együttműködő és adatvezérelt felhasználói felületeket hozzanak létre, amelyek könnyen skálázhatók és karbantarthatók.

Ez a keretrendszer egyesíti a HTML kód, a szokványos JavaScript és a szerveroldali adatokat kezelő funkciók egy egyszerű és hatékony platformon. Az AngularJS segítségével a fejlesztők könnyen követik a model-view-controller (MVC) tervezési mintát, amely felosztja az alkalmazást a megjelenítési, adatmodellezési és vezérlési rétegekre, és így javítja a fejlesztés hatékonyságát és karbantarthatóságát.

Az AngularJS funkciói közé tartoznak az adatkötés, az űrlapkezelés, az eseménykezelés, a routolás, a komponensalapú fejlesztés és az integrált tesztelési funkciók. Ezek az eszközök és funkciók lehetővé teszik a fejlesztők számára, hogy gyorsan és hatékonyan fejlesszenek modern webalkalmazásokat, amelyek dinamikus és felhasználóbarát felhasználói élményt nyújtanak.

**Bootstrap[[7]](#footnote-7):**

A Bootstrap egy ingyenesen elérhető és nyílt forráskódú front-end keretrendszer, amelyet a webfejlesztők használnak a gyors és hatékony weboldalak és webalkalmazások létrehozásához. A Bootstrap előre definiált stílusokat, sablonokat és komponenseket kínál, amelyek segítségével könnyen elindíthatja és testre szabhatja a projektjét. Ezek a komponensek tartalmazzák a gombokat, navigációs sávokat, űrlapokat, kártyákat, modális ablakokat és sok más felületi elemet, amelyek felgyorsítják a fejlesztési folyamatot és a következetes, modern megjelenést biztosítanak a webalkalmazásoknak. A Bootstrap egyszerűen használható, dokumentált és támogatott közösség által, amely további kiegészítőket és segítséget nyújt a fejlesztőknek.

**Katalon Studio[[8]](#footnote-8):**

A Katalon Studio egy ingyenesen elérhető, teljes körűen automatizált tesztelési platform, amely kifejezetten webes alkalmazásokhoz készült. A platform lehetővé teszi webes alkalmazások tesztelését különböző szinteken, beleértve a felhasználói felület (UI) tesztelését, API tesztelést, adatbázis-tesztelést és sok más tesztelési funkciót. A Katalon Studio könnyen használható grafikus felhasználói felülettel rendelkezik, ami lehetővé teszi a tesztek könnyű készítését és futtatását akár a kevésbé tapasztalt felhasználók számára is. A Katalon Studio részletes dokumentációval és online közösséggel rendelkezik, így könnyen megoldhatja a felmerülő problémákat és kérdéseket, valamint segítséget kaphat az alkalmazás teljes körű kihasználásához. A Katalon Studio kiterjeszthető és testre szabható, így alkalmazkodhat az adott projektek és követelmények egyedi igényeihez. Összességében a Katalon Studio kiváló választás a webes környezet teszteléséhez, mivel könnyen használható, számos funkciót kínál, jól dokumentált és támogatott, valamint lehetővé teszi a teljes körű és hatékony tesztelést.

## Függőségek

**Github:** <https://github.com/HunMartin2024/AutoAlibi-Alkatreszek>

**Fejlesztői környezet:**

* Visual Studio Code v1.88 - Kódolás
* XAMPP v3.3.0 – Adatbázis szerver
* Figma v116.15.4 – Frontend tervezés
* JIRA – Projekt menedzsment
* Github – Verzió kezelés, projekt menedzsment

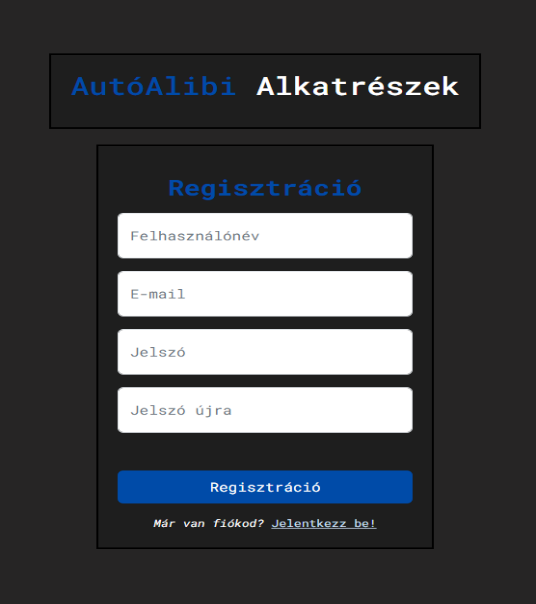
**Frontend függőségek:**

* Axios v1.6.2 – Backend requestek küldése
* Bootstrap v5.3 – Frontend elemek
* AngularJS v1.8.3 – A Frontend szíve

**Backend függőségek:**

* Express v4.18.2 – HTTP szerver
* Cors v2.8.5 – Cross-Origin engedélyek
* Dotenv v16.3.1 – Környezeti változók kezelése
* Bcrypt v5.1.1 – Jelszó titkosítás
* NodeJs v2.18.1 – JavaScript alkalmazások futtatása szerveroldalon
* Nodemon v3.1.0 – Automatikusan újraindítja a node alkalmazást
* Utils v0.3.1 – Segédfüggvények
* Mailer v0.6.1 – E-mail üzenetek küldését lehetővé teszi
* NPM v10.5.2 – Könyvtárak keresése, telepítése

## Forráskód

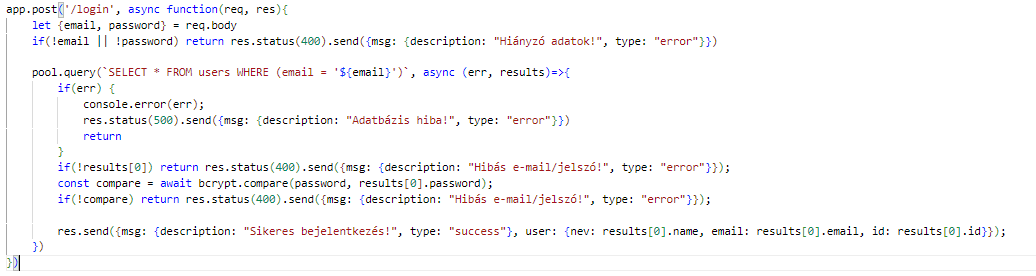
**Belépés és Regisztráció:** Amikor először nyitjuk meg a weboldalt, akkor a bejelentkezés oldal fogad minket. Itt egy figyelem felhívásra alkalmas ikont és színt választottam, hogy a weboldal csak regisztrált felhasználóknak érhető el. Ha nincs felhasználói fiókunk, akkor a „Regisztrálj” linkre kattintva az oldal átirányít, minket a Regisztráció fülre, ahol létrehozhatjuk a felhasználónkat. Miután regisztráltunk, az oldal automatikusan visszairányít minket a bejelentkezés részre és kiírja, hogy sikeres volt a regisztráció. Ebben az esetben küld nekünk egy e-mailt a weboldal, amiben megtudjuk erősíteni a regisztrációnkat.

10. ábra Bejelentkezés és Regisztráció oldal



11. ábra - Regisztráció kód

Amikor valaki regisztrálni szeretne, kitölti az űrlapot a nevével, email címével és két jelszóval. A kód ellenőrzi, hogy minden szükséges információt megadott-e a felhasználó, és hogy a két jelszó, amit megadott, megegyezik-e. Ha minden rendben van, akkor a kód elmenti az adatokat egy adatbázisba, és elküld egy megerősítő emailt a felhasználónak, hogy megerősítse az email címét. Ha valami hiba történik, például ha az email cím már regisztrálva van, akkor a kód visszaküld egy hibaüzenetet a felhasználónak. Összességében ez a kód segít a felhasználóknak regisztrálni az oldalra, biztosítva, hogy csak érvényes adatokkal lehessen regisztrálni, és megerősítést kér az email címükre.



12. ábra Belépés kód

Amikor valaki be szeretne jelentkezni, megadja az email címét és a jelszavát egy űrlapon keresztül. Ez a kód megkapja ezeket az adatokat, majd ellenőrzi, hogy mindkettőt megadta-e a felhasználó. Ha valami hiányzik, mondjuk az email cím vagy a jelszó, akkor a kód hibát jelez vissza, és közli a felhasználóval, hogy hiányzó adatok vannak. Ha minden szükséges adat rendelkezésre áll, a kód megpróbálja megtalálni az adott email címet az adatbázisban. Ha hiba történik ebben a folyamatban, például ha valami nem működik az adatbázissal, akkor a kód jelzi a problémát. Ha az email cím nem található az adatbázisban, akkor a kód azt jelzi, hogy hibás email cím vagy jelszó, mert nem talál egyezést az adatbázisban. Ha az email címet megtalálta az adatbázisban, akkor a kód összehasonlítja a megadott jelszót azzal, amit korábban a felhasználó regisztrációkor megadott. Ha a jelszavak egyeznek, akkor a kód jelzi, hogy sikeres volt a bejelentkezés, és visszaküldi a felhasználó nevét, email címét és egy azonosítóját. Ha megadott a jelszó mégsem egyezik az adatbázisban eltárolttal, akkor is hibát ad vissza.



13. ábra - Belépés Controller

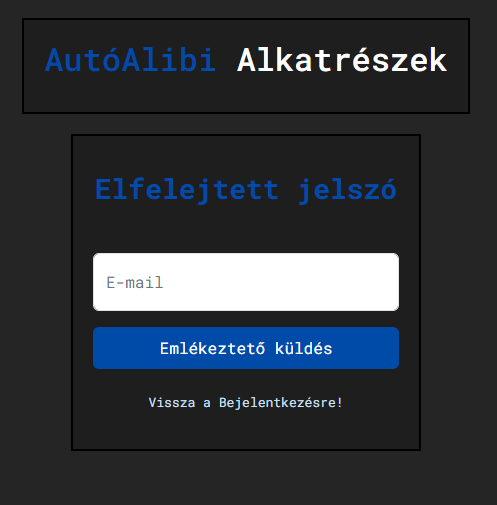
Ez a kód egy AngularJS kontrollert definiál, amely felelős a bejelentkezésért. A kontroller tartalmaz különböző változókat és funkciókat, amelyek segítségével kezeli a bejelentkezési űrlap működését és a felhasználói felületet. Ezek a következők:

`$scope.messages`, amelyben tárolom a különböző üzeneteket, amiket a bejelentkezés során megjelenítek a felhasználónak, például sikeres bejelentkezés vagy hibaüzenetek. A `$scope.loguser` változó tárolja a felhasználó bejelentkezési adatait, mint az email cím és a jelszó. A `$scope.passToggle` funkció egy gombnyomásra váltogatja a jelszó mező láthatóságát. Ha a felhasználó kattint a gombra, akkor a jelszó láthatóvá válik vagy visszatér az eredeti állapotába, attól függően, hogy előtte látható vagy rejtett volt. A `$scope.login` funkció kezeli a bejelentkezés folyamatát. Először törli az előző üzeneteket, majd megpróbál bejelentkezni a felhasználó által megadott adatokkal. Ha sikeres a bejelentkezés, akkor elmenti a felhasználói adatokat a `sessionStorage`-ben és átirányítja a felhasználót egy másik oldalra. Ha hiba történik, akkor hibaüzenetet jelenít meg a felhasználónak.

Az if feltétel azt ellenőrzi, hogy az első üzenet típusa "success", sikeres-e. Ha igen, akkor a `$timeout` segítségével vár két másodpercet, majd az `$location` segítségével átirányítja a felhasználót a "/mainMenu" útvonalra. Ez a késleltetés arra szolgál, hogy a felhasználó láthassa a sikeres bejelentkezés üzenetet, mielőtt átirányítják a főmenüre.

A `$scope.$apply()` segít az alkalmazás állapotának frissítésében a változások alkalmazása után. Ez biztosítja, hogy az összes változás megjelenjen a felhasználói felületen.

**Elfelejtett jelszó:** A felhasználónak van lehetősége új jelszót kérni abban az esetben, ha elfelejtette a regisztrációkor megadottat. A rendszer emailt fog küldeni arra az email címre, amivel az adott felhasználó létrehozta a fiókját. Az emailben kapott linkre kattintva átirányít egy olyan oldalra, ahol megadhatjuk az új jelszót.

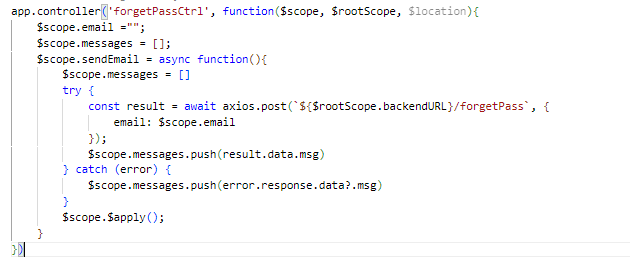


14. ábra - Elfelejtett jelszó és annak megváltoztatása



15. ábra - Elfelejtett jelszó kód

Ez a funkció lehetővé teszi a felhasználók számára, hogy új jelszót kérjenek, ha elfelejtették azt. Először ellenőrzi, hogy valóban megvan-e az email cím, és hogy nem üres-e. Ha nincs megadva vagy üres, akkor hibát küld vissza, amely jelzi, hogy nincs ilyen email cím regisztrálva. Ha van email cím, a kód lekérdezi az adatbázisból a felhasználót az email cím alapján. Ha a lekérdezés során hiba történik, akkor hibát küld vissza. Ha talál találatot az adatbázisban, a kód generál egy új, véletlenszerű token-t, amelyet felhasznál a jelszó helyreállítási folyamat során. Ez a token segíti azonosítani azon felhasználót, aki az elfelejtett jelszó helyreállítását kérelmezte. Ezután a kód frissíti az adatbázisban a felhasználó rekordját, hogy hozzárendelje az új passToken-t az email címhez. Ezután a kód elküld egy emailt a felhasználónak az elfelejtett jelszó helyreállítási folyamatának folytatásához. Az email tartalmazza a jelszó helyreállítási linket, amely a token-t tartalmazza az azonosításhoz. Végül, ha minden folyamat sikeres volt, a kód visszaküldi a választ a kliensnek, hogy jelzi az email elküldését, és visszaadja a generált passToken-t is.



16. ábra - Elfelejtett jelszó Controller

A kontroller tartalmaz különböző változókat és funkciókat, amelyek segítségével kezeli az elfelejtett jelszó űrlapot és a felhasználói felületet.

Az `$scope.email` változó tárolja a felhasználó által megadott email címet. Az `$scope.messages` változó tartalmazza a különböző üzeneteket, amelyeket az elfelejtett jelszó folyamata során kap a felhasználó. A `$scope.sendEmail` funkció felelős az email elküldéséért az elfelejtett jelszó funkció során. Először törli az előző üzeneteket, majd megpróbál elküldeni egy POST kérést a szervernek az elfelejtett jelszó funkció használatával. Ha sikeres az elküldés, a kapott üzenetet hozzáadja a `$scope.messages` tömbhöz. Ha hiba történik az elküldés során, a kapott hibaüzenetet hozzáadja a `$scope.messages` tömbhöz. Végül, a `$scope.$apply()` segít az alkalmazás állapotának frissítésében a változások alkalmazása után.

**Új jelszó kérés:** A felhasználónak lehetősége van új jelszót készítenie, ha elfelejtette a jelszavát, de ehhez tudnia kell az email címét, amivel regisztrált, mivel oda fogja megkapni a linket, amivel megtudja változtatni.

**A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás**

17. ábra - Új jelszó kód

Az új jelszót és egy speciális azonosítót (tokent) vár a kódtól. Ha hiányzik a token, vagyis azonosító, akkor a kód ezt jelezve visszautasítja a kérését. Amennyiben a token helyes, akkor ellenőrzi, hogy ez a token valóban létezik-e az adatbázisban. Ha a token nem található az adatbázisban, akkor azt jelzi, hogy valami nem stimmel. Ha minden adat rendben van, akkor megkéri a felhasználót, hogy adja meg az új jelszót kétszer is, hogy biztosan jól írta-e be. Ellenőrzi, hogy mindkét új jelszó mezőt kitöltötte-e a felhasználó és hogy egyeznek-e egymással. Ha minden rendben van, akkor az alkalmazás biztonságosan tárolja az új jelszót az adatbázisban, hogy a felhasználók később be tudjanak jelentkezni vele. Végül a felhasználót értesíti, hogy az új jelszó sikeresen megváltoztatásra került, vagy hibaüzenetet küld, ha valami nem működött.

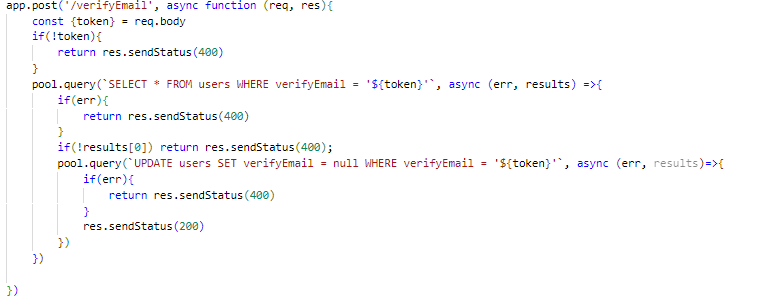
**A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás**

18. ábra - Új jelszó Controller

Először létrehozok egy üres tárolót (`$scope.passwords`), amelyben tárolni fogom az új jelszó adatokat, és egy üres listát (`$scope.messages`), amelyben később üzeneteket tárolok. A `changePwd` nevű függvény arra szolgál, hogy az új jelszó beállítását kezelje. Először üríti a korábban tárolt üzeneteket. Ezután elküld egy kérést a szervernek az új jelszó beállításáért. A kérés tartalmazza az új jelszó adatokat és egy token-t, ami egyfajta azonosító a felhasználóhoz. Ha a kérés sikeres volt, az üzenetet, amit a szerver visszaküldött, hozzáadja a tárolt üzenetek listájához. Ha a kérés sikertelen volt, akkor a hibáról szóló üzenetet (amit a szerver válaszaként küldött) hozzáadja a listához. Végül a kódrészlet frissíti az alkalmazás állapotát, hogy az új üzenetek megjelenjenek a felhasználói felületen.

**Email cím megerősítés regisztráció után:** Ennek a funkciója az, hogy megerősítse a felhasználók email címét egy regisztrációs folyamat során.



19. ábra - Email cím megerősítés kód

Amikor valaki regisztrál az oldalon, egy megerősítő emailt kap, amely tartalmaz egy egyedi token-t a felhasználó email címének megerősítéséhez. A kód a beérkező kérésben megkapja ezt a tokent a `req.body` segítségével. Először ellenőrzi, hogy valóban megvan-e a token, mert ez alapján tudja, hogy a kérés valós-e. Ha nincs megadva token, akkor hibaüzenet helyett csak egy 400-as hibakódot küld vissza. Ha van token, akkor a kód lekérdezi az adatbázisból a felhasználót a token alapján.

Ha a lekérdezés során hiba történik, vagy ha nincs találat a token alapján, akkor szintén csak egy 400-as hibakódot küld vissza. Ha talál találatot és van token, a kód frissíti az adatbázisban a felhasználó rekordját úgy, hogy a megerősítő email token értékét nullára állítja. Ez azt jelzi, hogy az email cím megerősítésre került. Végül, ha minden folyamat sikeres volt, a kód csak egy 200-as visszajelzést küld vissza, hogy jelezze, hogy az email cím megerősítése sikeres volt.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás

20. ábra - Email cím megerősítés Controller

Első lépésként a kódban a `$location` használja a token értékének lekérdezésére a böngésző URL-jét. Ez a token egy lehetőséget kínál az e-mail cím ellenőrzésére. Ha a kérés sikeres volt (`then` ág), akkor a kód 3 másodperc késleltetés után átirányítja a felhasználót a főoldalra (`/` útvonalra). Ha a kérés sikertelen volt (`catch` ág), akkor a kód törli a token-t az URL-ből, majd szintén átirányítja a felhasználót a főoldalra. Mindkét esetben (`then` és `catch`) `$scope.$apply()` hívás történik, amely frissíti az alkalmazás állapotát, hogy a változások megjelenjenek a felhasználói felületen.

**Főoldal:** A sikeres bejelentkezés után a Főoldalra irányít minket a weboldal. Ezen az oldalon tudja megnézni a felhasználó az eddigi rendeléseit.

A képen képernyőkép, szöveg, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

21. ábra Főoldal (Rendelések táblázat)

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Automatikusan generált leírás

22. ábra Főoldal (Rendelések táblázat kód)

Első lépésben a kód figyeli az `/orders/:userid` elérési útvonalat a webalkalmazáson belül. Amikor valaki meglátogatja ezt az útvonalat a webalkalmazásban, a szerver megkapja az útvonal részleteit, például a felhasználó azonosítóját. Ha azonban valamiért nem kapnánk meg a felhasználó azonosítóját, a szerver ezt észreveszi, és hibát küld vissza, hogy segítsen helyrehozni ezt a problémát. Ha minden rendben van az azonosítóval, a szerver folytatja a következő lépéssel, ami az adatbázisban a megfelelő rendelések lekérdezése. Tehát megpróbálja megtalálni azokat a rendeléseket, amelyek ehhez a felhasználóhoz tartoznak. Miután megkapta az adatokat az adatbázistól, a szerver elkezdi feldolgozni ezeket az adatokat, hogy megfelelő formában küldje vissza a kliensnek, azaz azoknak a böngészőknek vagy alkalmazásoknak, amelyek lekérték ezeket az információkat. A szerver minden rendelést egyesével megvizsgál, és összegyűjti azokat az adatokat, amelyeket meg akarunk jeleníteni a felhasználónak. Miután az összes rendelés adatait összegyűjtöttük és előkészítettük, a szerver visszaküldi ezeket az adatokat a böngészőnek vagy az alkalmazásnak, amely kérte őket.

A képen szöveg, képernyőkép, sor, Betűtípus látható

Automatikusan generált leírás

23. ábra - Főoldal Controller

A kontroller három fő dolgot csinál: A `$scope.rendelesek` változóban tárolja a rendeléseket. Ez az a hely, ahol az alkalmazás tárolja és kezeli a megjelenítendő adatokat. Kezdetben ez a változó üres tömbként kerül inicializálásra. Az `axios.get` függvény segítségével HTTP GET kérést küld egy adott URL-re, amely a rendelések adatait tartalmazza. Az URL összeállításakor a `$rootScope.backendURL` és a `$rootScope.user.id` változókat használja. Ezek a változók az alkalmazás globális állapotát tárolják (például a háttér URL-jét és a bejelentkezett felhasználó azonosítóját). A kérés sikeressége esetén, azaz ha a szerver válaszol a kérésre, az alkalmazás frissíti a `$scope.rendelesek` változót a kapott adatokkal (`res.data`). Ezután a `$apply()` függvény hívásával frissítenie kell a megjelenítést a `$scope.rendelesek` változó változása miatt. Ez biztosítja, hogy az új adatok megjelenjenek a felhasználói felületen.

**Kapcsolat:** A weboldalon ez az egyetlen olyan oldal, amelyik nem használ JavaScriptet. Ez az oldal tájékozódás céljából készült, itt csak HTML-t és CSS-t használtam az elkészítéshez.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

24. ábra - Kapcsolat oldal HTML Kódolás

A `<div class="container cont">` elem egy konténer, amely tartalmazza az összes információt és ikonokat. Ez a konténer formázási célokat szolgál, mint a tartalom elrendezésének meghatározására.

A `<div class="containerText">` elem tartalmaz egy szöveges címsort (p), amely a kapcsolatok alcímét jelzi. A `<div class="d-flex p-3 justify-content-evenly flex-wrap mb-5">` elem egy további konténer, amely a kapcsolattartó információk részleteit tartalmazza. Ez a konténer egy flexbox elrendezést használ (`d-flex` osztály), hogy a tartalmat egy sorban helyezze el és az elemeket egyenletesen oszlassa el (`justify-content-evenly` osztály). A `flex-wrap` osztállyal azt jelzi, hogy a tartalom több sorba is elosztható, ha szűk az elrendezés. Az `mb-5` osztály egy margin-bottom értéket állít be a konténer aljának távolságára. A két `<div>` elem a két különböző kapcsolatot tartalmazza: "Központi irodánk" és egy "Logisztikai központ". Mindkét `<div>` elem egy szöveges címsort (`<p class="TitleTags">`), egy vízszintes elválasztó sort (`<hr class="line">`) és szöveges információkat tartalmaz a cím, telefonszám, e-mail cím és faxszám tekintetében. A `<div class="social-icons">` elem az ikonokat tartalmazza a különböző közösségi média platformokhoz, mint például a Facebook, Instagram és Twitter. A `fab` és az `fa` osztályok ikonokat rendelnek a Font Awesome ikonkönyvtárból.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

25. ábra - Kapcsolat oldal formázása

`.cont`:

- `padding`: 0-ra állítja a belső margót, kivéve az alját, ahol 20px padding marad.

- `border-radius`: 10px sugarú kerekítést alkalmaz az elem sarkaira.

- `width`: Az elem szélességét a viewport 60%-ra állítja.

- `background-color`: A háttérszín a #004BA8 (sötétkék) lesz.

- `position`: Relatív pozíciót állít be, ami a pozíció kiszámításánál az eredeti helyzetet veszi figyelembe.

`.cont > div > div`:

- `margin-top`: 5rem távolságot hagy a konténer tetejétől a tartalom megjelenítéséhez.

- `word-wrap`: A hosszú sorok automatikusan tördelésre kerülnek, ha túl hosszúak.

`.containerText`:

- `width`: A szélesség 100%-ra állítva teljes szélességű lesz.

- `background-color`: A háttérszín a #1E1E1E (sötét szürke) lesz.

- `border-radius`: Az elem felső sarkain 10px sugarú kerekítést alkalmaz.

- `text-align`: A szöveget középre igazítja.

- `padding`: 5px paddingot ad a felső és alsó részhez, nulla paddingot a jobb és bal oldalon.

`.containerText p`:

- `margin`: A bekezdéseknek nincs margója.

- `font-size`: A betűméret 30px lesz.

- `color`: A szövegszín fehér lesz (#fff).

`.cont > div > div p`:

- `margin`: A bekezdéseknek nincs margója.

- `font-family`: A betűtípus a "Roboto Mono" lesz, monospaced formátumban.

- `color`: A szövegszín fehér lesz (#fff).

- `margin-bottom`: 20px távolságot hagy a bekezdések alatt.

**Webshop:** A felhasználó 8 mező közül tud választani alkatrészt, minden mező kiválasztásakor az adatok az adott részhez automatikusan töltődnek be adatbázisból.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

26. ábra - Webshop oldal HTML kódolás

A `webshop-sidebar` oszlop: Ez az oszlop tartalmazza a termékkategóriákhoz tartozó linkeket. Minden link egy specifikus termékkategóriára mutat, például "Szűrők", "Fékberendezés", "Tömítések".

A `webshop-content` oszlop: Ez az oszlop tartalmazza a termékek megjelenítését és kezelését. Minden termék egy dobozban jelenik meg, ami tartalmazza a termék nevét, típusát, adatokat, árát, egy mennyiség input mezőt és egy kosárba helyezés gombot. Ezeket a termék adatokat dinamikusan generálja az alkalmazás a backend adatbázisából. A termékek listája szintén dinamikusan változik a keresési lekérdezések vagy a kiválasztott kategória alapján. Az oldal további részei között található egy keresőmező (`input`), amely lehetővé teszi a felhasználók számára, hogy kereséseket végezzenek a rendelkezésre álló termékek között. A keresés eredményei közvetlenül frissítik a terméklistát a felhasználó által megadott szöveg alapján. Az oldal általános felépítése dinamikus és interaktív, lehetővé téve a felhasználók számára, hogy könnyen navigáljanak a termékkategóriák között, kereséseket végezzenek, és hozzáadjanak termékeket a kosárhoz a vásárláshoz.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

27. ábra - Webshop backend

Az útvonal feldolgozása során a kódban lekéri a kérésből az `type` és `search` paramétereket. Ezek a kérésben szereplő query string paraméterek (`req.query`) alapján érkeznek. Ellenőrzi, hogy a `type` paraméter megadásra került-e. Ha nem, akkor visszatér egy hibával (`status 400`), amely jelzi, hogy nincs megadva típus. Ha van `search` paraméter, akkor a kódrészlet egy lekérdezést hajt végre az adatbázisban, hogy a megadott típushoz tartozó termékeket a megadott keresési feltétel alapján kérje le. A keresési feltétel a termék nevének részletét keresi (`LIKE '%${search}%'`). Az eredményeket visszaküldi a kliensnek. Ha nincs `search` paraméter, akkor egyszerűen lekérdezi az adatbázisból a megadott típushoz tartozó összes terméket. Ha bármelyik lekérdezés során hiba történik az adatbázissal, a kód visszatér egy hibaüzenettel (`status 500`), amely jelzi, hogy valami probléma merült fel az adatbáziskapcsolattal.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

28. ábra - Webshop Controller

Inicializálja a `$scope` objektumot, amely tartalmazza az adatokat és a funkciókat a kontrollerben. Ez tartalmazza az `items` tömböt, a `mennyiseg` tömböt (amely a termékek mennyiségét tárolja), és a `search` változót (a keresőmezőbe beírt keresőkifejezés). A kódban definiál egy `debounced` függvényt, amely használja a lodash könyvtár `debounce` funkcióját, hogy késleltessen egy másik függvényhívást. Ez azért van, hogy ne történjen backend hívás minden karakter beírása után a keresőmezőbe. Definiálja a `getData` függvényt, ami egy HTTP GET kérést indít a szerver felé a megfelelő típusú termékek lekérdezésére. A választ kezeli és az adatokat formázza, majd hozzáadja az `items` tömbhöz. A `$scope.kereses` függvény meghívja a `debounced` függvényt, ami késlelteti a `getData` hívást a keresési funkcióhoz. A `$scope.hozzaadas` függvény felelős az adott termék hozzáadásáért a kosárhoz. Frissíti a kosár tartalmát a megadott mennyiség alapján. A kosár tartalmát a `$rootScope` objektumban tárolja, és az adatokat a böngésző helyi tárolójában (`localStorage`) is tárolja. Végül meghívja a `getData` függvényt az oldal betöltődésekor az alapértelmezett adatok lekérdezésére és megjelenítésére.

**Kosár:** A webshop-ban választott alkatrészek jelennek meg itt.

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

29. ábra - Kosár HTML kódolás

A `<div class="order">` elem egy konténert definiál a kosár tartalmának megjelenítésére és kezelésére. A `<div class="orderText">` elem tartalmazza a "Kosár tartalma" című címsort. A `<table id="rendelesek">` elem egy táblázatot definiál a kosárban lévő termékek listázására. A táblázat fejlécében (`<thead>`) meghatározza a következő oszlopokat: "Termék neve", "Termék képe", "Mennyiség", "Ár" és egy üres oszlop. A táblázat törzsében (`<tbody>`) vannak a tényleges termékek listája. Az `ng-repeat` direktívával egy változó (`items`) minden eleméhez egy sor kerül megjelenítésre. A táblázatban minden sorban megjelenik egy termék neve, képe (kis méretben), mennyisége, ára (mennyiséggel szorozva), valamint egy "Törlés" gomb, amelynek segítségével a felhasználó eltávolíthatja a terméket a kosárból. A táblázat alján egy további sorban megjelenik az összesítés, amely tartalmazza az összes termék számát és az összesített árat. Ha a kosár üres (`items.length == 0`), akkor egy üzenet jelenik meg a táblázatban. Ha a kosár nem üres (`items.length != 0`), akkor egy "Tovább a fizetéshez!" gomb jelenik meg, amelyre kattintva a felhasználó továbbléphet a fizetési folyamathoz.

A képen szöveg, Betűtípus, sor, képernyőkép látható

Automatikusan generált leírás

30. ábra - Kosár backend

A kódban meghatározza a `/webshopCart` útvonalon érkező POST kérések kezelését. A kérés (`req.body`) kiolvassa az adatokat, amelyeket a kliens elküldött. Ezek az adatok valószínűleg a kosárban lévő termékek és a hozzájuk tartozó mennyiségek. Az adatokból kinyeri az összes termék azonosítóját (`IDs` változó), amelyeket az ügyfél a kosárba helyezett. Lekérdezi az adatbázisból azokat a termékeket, amelyeknek az azonosítója megegyezik az elküldött azonosítók egyikével. Ehhez egy SQL lekérdezést alkalmaz, amely az `IN` operátort használja, hogy meghatározza a kiválasztott termékek listáját. Amennyiben a lekérdezés során hiba történik az adatbázissal, a kód visszatér egy hibaüzenettel (`status 500`), amely jelzi, hogy valami probléma merült fel az adatbáziskapcsolattal. Ha a lekérdezés sikeresen lefut, akkor az eredményeket visszaküldi a kliensnek, amelyek tartalmazzák az adatbázisból lekért termékek részleteit. Ezáltal a kliens oldalon frissítheti vagy megjelenítheti a kosárban lévő termékeket.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

31. ábra - Kosár Controller

A kosár controller inicializálja a `$scope` objektumot, amely tartalmazza az adatokat és a funkciókat a kontrollerben. Definiálja az `items`, `message` és `osszeg` változókat, amelyek tárolják a kosárban lévő termékek listáját, egy üzenetet a felhasználó számára ("Kosár üres!"), valamint az összesített árat. Ellenőrzi, hogy van-e már tárolt kosár a böngésző helyi tárolójában (`localStorage`). Ha nincs, vagy az üres, akkor beállítja az üzenetet, hogy a kosár üres. Ha van kosár az adatok között, elküldi a kosár tartalmát a szervernek egy POST kérésben a `/webshopCart` útvonalra. A kosár tartalma az `axios.post()` metódus segítségével kerül átadásra. Miután a szerver válaszolt, az adatokat feldolgozza, kiszámolja az összesített árat, majd az `items` tömbhöz adja a termékek részleteit. Meghatározza a `$scope.torles` és `$scope.fizetes` függvényeket. A `torles` függvény felelős egy adott tétel törléséért a kosárból, a `fizetes` függvény pedig az átirányításért a fizetési oldalra. A változások észlelése és frissítése érdekében hívja meg a `$scope.$apply()` függvényt.

**Vásárló adatai:** A rendelés folytatása után irányít át minket a kosárról az oldal. Itt a vásárlói adatokat kell megadni, ha ez megtörtént akkor a főoldalra kerül a rendelések táblázatba.

A képen szöveg, képernyőkép, dokumentum, Betűtípus látható

Automatikusan generált leírás

32. ábra - Vásárló adatai HTML kódolás

A `<div class="container cont">` elem egy konténert definiál, amely a vásárló adatainak megjelenítésére és kezelésére szolgál. A `<div class="containerText">` elem tartalmazza a "Vásárló adatai" címsort. A `<div class="d-flex p-3 justify-content-evenly flex-wrap mb-5 fw-bold text-white">` elem egy flexbox konténert definiál, amelyben három rész található: számlázási adatok, szállítási adatok és fizetési mód. Az első `<div>` tartalmazza a számlázási adatokat, mint például név, telefonszám, irányítószám, város és cím. Itt egy checkbox is található, amely lehetővé teszi a felhasználó számára, hogy a szállítási adatokat ugyanazokra az adatokra állítsa be. A második `<div>` tartalmazza a szállítási adatokat, ugyanúgy, mint az első, de ezeket csak akkor kell megadni, ha különböznek a számlázási adatoktól. Ennek ellenőrzése a fenti checkbox segítségével történik. A harmadik `<div>` a fizetési módot definiálja. Két lehetőség közül választhat a felhasználó: bankkártyás vagy készpénzes fizetés. Végül egy "Megrendelés" gomb található az űrlapon, amelynek segítségével a felhasználó elküldheti a rendelését. A gombra kattintva a `megrendeles()` függvény fut le.

A képen szöveg, sor, Betűtípus, képernyőkép látható

Automatikusan generált leírás

33. ábra - Vásárló adatai backend

A (`req.body`) kinyeri az adatokat, amelyek a számlázási és szállítási adatokat, a kosarat, a fizetési módot és a felhasználó azonosítóját tartalmazzák. Ellenőrzi, hogy minden szükséges adatot megkaptak-e. Ha valamelyik adat hiányzik, akkor visszaküld egy hibaüzenetet (`status 400`), amely jelzi, hogy hiányzó adatok vannak. Ha minden szükséges adat rendelkezésre áll, akkor végrehajt egy adatbázis-beillesztési műveletet (`INSERT INTO`), amely beilleszti az adatokat az "orders" táblába. Az adatok között szerepelnek a számlázási és szállítási adatok, a kosár tartalma, a fizetési mód és a felhasználó azonosítója. Ha a beillesztés sikeres volt, akkor visszaküld egy sikeres rendelésről szóló üzenetet (`status 200`), amely jelzi, hogy a rendelés sikeresen feldolgozásra került. Ha valamilyen hiba történik a folyamat során, akkor visszaküld egy hibaüzenetet (`status 500`), amely jelzi, hogy valamilyen adatbázishiba merült fel.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

34. ábra - Vásárló adatai Controller

A userData Controller inicializálja a `$scope` objektumot, amely tartalmazza a kontrollerben használt változókat és funkciókat. Az inicializált változók közé tartozik a `payment` (fizetési mód), `szallitasi` (szállítási adatok), `szamlazasi` (számlázási adatok) és `same` (logikai változó, amely azt jelzi, hogy a szállítási adatok ugyanazok-e, mint a számlázási adatok). A `megrendeles` függvény felelős a rendelés elküldéséért. Ha a felhasználó kiválasztotta, hogy a szállítási adatok ugyanazok legyenek, mint a számlázási adatok, akkor beállítja a `szallitasi` objektumot a `szamlazasi` objektum értékére. Ezután egy POST kérést küld a szervernek a `/order` útvonalra, amely tartalmazza a felhasználó által megadott adatokat, mint például a felhasználó azonosítóját, szállítási és számlázási adatokat, a kosár tartalmát és a fizetési módot. Ha a rendelés sikeresen elküldésre került (`status 200`), akkor törli a kosár tartalmát, frissíti a helyi tárolót (`localStorage`), majd átirányítja a felhasználót a főmenüre (`/mainMenu` útvonalra). Ez a kontroller összességében felelős a felhasználói adatok kezeléséért, a rendelés elküldéséért és az alkalmazás állapotának frissítéséért a rendelés után.

## Tesztelési dokumentáció

A tesztelés fontos szerepet játszik a hibák felfedezésében és kiküszöbölésében, ezáltal hozzájárulva a szoftver minőségének javításához. Az első lépés általában a kód átvizsgálása, ami lehetővé teszi az egyértelmű hibák, például elhagyott idézőjelek azonosítását. A modern szövegszerkesztők, mint például a Visual Studio Code, segítenek ebben a szintaxis kiemeléssel, ami az egyes elemeket különböző színekkel jelöli meg. Azonban vannak olyan hibák, mint például a pontosvessző elhagyása, amelyeket könnyen elkerülhetünk, de csak futtatáskor veszünk észre, amikor a program nem fut le. A szemantikai hibák nehezebben azonosíthatók, mivel ezek logikai hibák, amelyeket a szintaxis ellenőrzése nem fedez fel. Ezeket általában csak a program tesztelésével, funkcióinak kipróbálásával lehet észrevenni. Lehetőség szerint egy külső személyt is bevethetünk a tesztelésbe, aki friss szemmel vizsgálja meg a programot, és nagyobb valószínűséggel talál hibákat.

A tesztelés célja mindig az, hogy a lehető legmagasabb minőséget érjük el a szoftverünkkel. Ennek érdekében folyamatosan teszteltem a kódokat és funkciókat, hogy biztos legyek benne, hogy minden tökéletesen működik. Minden egyes sor kód mögött állt egy alapos tesztelési folyamat, hogy kiküszöböljem az esetleges hibákat és biztosítsam a stabil működést. Az alkalmazás fejlesztése során folyamatosan változtak az igények és a kihívások, de én mindig arra törekedtem, hogy minden változtatás után újraellenőrizzem a funkcionalitást és a kompatibilitást. A tesztelés nem csupán a fejlesztés kezdeti szakaszában fontos, hanem az egész folyamat során, hogy a végeredmény valóban kielégítse a felhasználók elvárásait. Meghívtam családtagjaimat, hogy segítsenek a tesztelésben. Tudtam, hogy rájuk számíthatok, hogy őszintén végigmennek az alkalmazás minden részletén, és értékes visszajelzéseket adnak. Az egyik legfontosabb funkció, amit teszteltünk, a kosár volt. A felhasználók könnyedén hozzáadhattak termékeket a kosárhoz és törölhették a tételeket, ha szükséges volt. Családtagjaimmal együtt különböző forgatókönyveket játszottunk végig, hogy megbizonyosodjunk róla, hogy a kosár minden esetben megfelelően működik-e. Az ő visszajelzéseik segítettek abban, hogy finomítsam az élményt és optimalizáljam a funkcionalitást. A bejelentkezés és regisztráció is fontos szerepet játszott a tesztelés során. A családtagjaim segítségével ellenőriztük, hogy a felhasználók könnyen és zökkenőmentesen tudnak-e belépni az alkalmazásba, illetve regisztrálni. Összességében a családtagjaim segítsége nélkül nem lett volna ilyen sikeres a tesztelés folyamata.

# Összegzés

## Fejlesztési lehetőségek

Igyekeztem olyan projektet csinálni, amiben rengeteg továbbfejlesztési lehetőség lesz. Fontosnak tartom a továbbfejlesztését, ezért igyekeztem úgy megírni a projektem, hogyha hónapok után előveszem, vagy más kezébe kerül akkor ő is könnyedén megértse és folytatni tudja. Tiszta kód elvre törekedtem a létrehozásától kezdve.

Ilyen fejleszthetőségi opciók lehetnek:

* A vásárlást külön autó kategóriákra lehetne bontani, több termék leírással.
* Bővíteni lehetne az alkatrészek típusait.
* A rendelés táblázatot lehetne bővíteni adatokkal (pontos idő, hogy mikor adták le a rendelést). Lehetne a rendelés táblázatot törölni.

## Személyes tapasztalatok

Rengeteg hibával találkoztam a fejlesztés során, ez volt az első és egyben legnagyobb projektem, amin eddig dolgoztam „egyes egyedül”. Nagy segítségemre volt az internet a forrásokban, mind az általa nyújtott eszközökben (Figma, Jira, Git). Valamint a konzulensek is segítettek, hiszen rengeteg ötletet kaptam. Természetesen okoztak némi fejtörést az elhanyagolt pontosvesszők, a törölt vagy javított kódrészletek utáni félbehagyott zárójelek. A CSS-nél is kellett figyelni, mivel Bootstrapet is használtam, ami nagyon sokszor csak nehezítette a dolgom. Rengeteg olyan Bootstrap elem volt, amire külön CSS-t kellett írnom. Ilyenkor át kellett írni vagy akár eltávolítani ezeket, majd alternatív megoldásokat kellett keresni. Az alapokban sem volt könnyű a dokumentáció megírása, mivel ilyen nagy projekthez szigorú szabályok szerint kell dokumentálni. Összefoglalva, sikernek élem meg hogy egyedül sikerült az oldalt megvalósítani a tervezett módon, így egy működő rendszert létrehozva.

# Felhasznált források

*W3 schools: Learn to code*<https://www.w3schools.com/> [letöltés: 2024. 02.15.]

*XAMPP Apache + MariaDB + PHP + Perl*<https://hu.wikipedia.org/wiki/XAMPP> [letöltés: 2024. 02.15.]

*AngularJS - Superheroic JavaScript MVW Framework*<https://angularjs.org/> [letöltés: 2024. 02.15.]

*Github*<https://github.com/> [letöltés: 2024. 02.15.]

*Fonts Google*<https://fonts.google.com/> [letöltés: 2024. 02.15.]

*Katalon Studio - Writes tests faster*<https://katalon.com/> [letöltés: 2024. 02.15.]

*Visual Studio Code – Code editing*<https://code.visualstudio.com/> [letöltés: 2024. 02.15.]

*Bootstrap – Build fast, responsive sites*<https://getbootstrap.com/> [letöltés: 2024. 02.15.]

# Ábrajegyzék

1. ábra - Bejelentkezés az oldalra 1

2. ábra - Főoldal 3

3. ábra - Kapcsolat fül 4

4. ábra - Webshop fül 5

5. ábra – Kosár 6

6. ábra - Vásárlói adatok 7

7. ábra - ERmodell látványterv 8

8. ábra - Adatbázis 9

9. ábra - Items tábla adatok 14

10. ábra Bejelentkezés és Regisztráció oldal 20

11. ábra - Regisztráció kód 21

12. ábra Belépés kód 21

13. ábra - Belépés Controller 22

14. ábra - Elfelejtett jelszó és annak megváltoztatása 24

15. ábra - Elfelejtett jelszó kód 24

16. ábra - Elfelejtett jelszó Controller 25

17. ábra - Új jelszó kód 26

18. ábra - Új jelszó Controller 27

19. ábra - Email cím megerősítés kód 27

20. ábra - Email cím megerősítés Controller 28

21. ábra Főoldal (Rendelések táblázat) 29

22. ábra Főoldal (Rendelések táblázat kód) 29

23. ábra - Főoldal Controller 30

24. ábra - Kapcsolat oldal HTML Kódolás 31

25. ábra - Kapcsolat oldal formázása 32

26. ábra - Webshop oldal HTML kódolás 34

27. ábra - Webshop backend 35

28. ábra - Webshop Controller 36

29. ábra - Kosár HTML kódolás 37

30. ábra - Kosár backend 38

31. ábra - Kosár Controller 39

32. ábra - Vásárló adatai HTML kódolás 40

33. ábra - Vásárló adatai backend 41

34. ábra - Vásárló adatai Controller 42

1. <https://hu.wikipedia.org/wiki/Visual_Studio_Code> [↑](#footnote-ref-1)
2. <https://hu.wikipedia.org/wiki/XAMPP> [↑](#footnote-ref-2)
3. <https://hu.wikipedia.org/wiki/Git> [↑](#footnote-ref-3)
4. <https://hu.wikipedia.org/wiki/HTML> [↑](#footnote-ref-4)
5. <https://hu.wikipedia.org/wiki/CSS> [↑](#footnote-ref-5)
6. <https://en.wikipedia.org/wiki/AngularJS> [↑](#footnote-ref-6)
7. <https://hu.wikipedia.org/wiki/Bootstrap> [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Katalon_Studio> [↑](#footnote-ref-8)