1. Bevezetés
   1. A vizsgaremek témája

Az online áruházunk, melynek neve Pc-Max kiváló minőségű számítógép alkatrészek széles választékát kínálja az ügyfelek számára. Legyen szó otthoni számítógépről, irodai munkáról vagy éppen játékról, nálunk mindent megtalál, hogy kielégítse informatikai igényeit. Böngéssze áruházunkat, és találja meg a legjobb PC alkatrészeket az Ön számára!

* 1. A program funkciói

**Termékek és kategóriák**: Az oldalon széles választékban elérhető PC alkatrészek,valamint ezeket kategóriákba rendezett módon jelennek meg.

**Kereső**: Keresési funkció segít a látogatóknak gyorsan megtalálni a kívánt termékeket vagy információkat.

**Kosár és pénztár**: A vásárlók virtuális kosarat használhatnak a termékek összegyűjtésére és a vásárlás befejezésére.

**Regisztráció és fiókkezelés**: A vásárlók létrehozhatnak fiókokat, ahol megtekinthetik rendeléseiket és következő vásárlásaikat könnyebben teljesíthetik.

* 1. Célközönség

**Otthoni felhasználók**: Azok, akik számítógépet vásárolnak otthoni használatra, például szórakozás, munka vagy oktatás céljából.

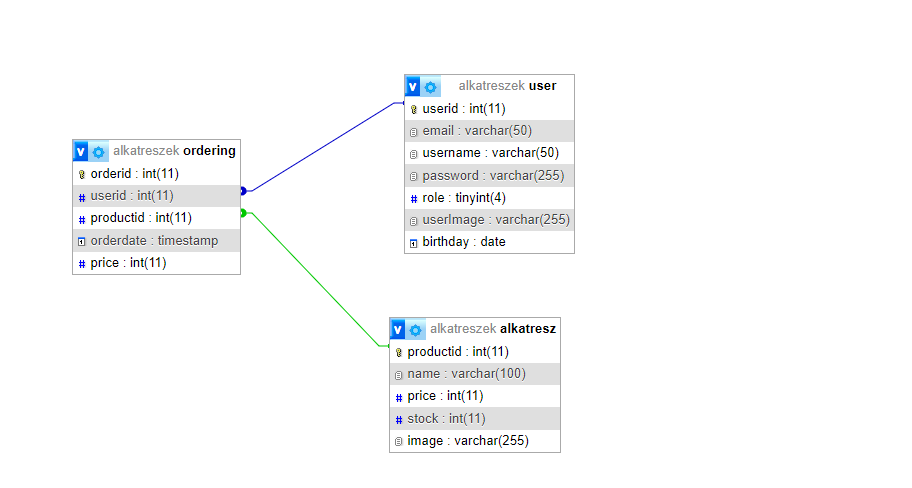
**Gamer közösség**: Azok, akik gaming számítógépeket és kiegészítőket keresnek, hogy kiváló játékélményt éljenek át.

**Irodai felhasználók**: Vállalatok és irodák, amelyek informatikai eszközöket és szolgáltatásokat keresnek munka céljából.

**PC építők és tuningolók**: Azok, akik saját számítógépüket építik vagy frissítik, és különböző alkatrészeket vásárolnak hozzá.

**IT szakemberek és fejlesztők**: Olyan személyek, akik számítógépes alkatrészeket és perifériákat vásárolnak professzionális projektekhez és fejlesztésekhez.

1. Fejlesztői dokumentáció



* 1. Adatbázis
     1. Az adatbázisok nélkülözhetetlen eszközei a modern webalkalmazásoknak, mivel segítik az adatok hatékony tárolását, kezelését és lekérdezését. A webshopunk számára kialakított adatbázis oldalunk tervezésekor a MySQL-t választottuk, mivel az iskolában ezt tanultuk.
     2. User tábla

userid(Elsődleges int(11) AUTO\_INCREMENT): Egyedi azonosító, amely automatikusan növekszik minden új felhasználó hozzáadásakor. Az elsődleges kulcsként szolgál a táblában.

email (varchar(50) utf8mb4\_general\_ci): Felhasználó e-mail címe, maximális hossza 255 karakter. Az "utf8mb4\_general\_ci" karakterkészletet választottuk, ami támogatja a nemzetközi karaktereket is.

password (varchar(255) utf8mb4\_general\_ci): A felhasználó jelszavát tárolja. A "utf8mb4\_general\_ci" karakterkészletet választottuk a karakterek széles skálájának támogatásához. A felhasználók jelszavainak biztonságos tárolása kiemelt fontosságú, ehhez a bcrypt Node.js csomagot használtuk, amely hatékony jelszóhashelést és számos biztonsági intézkedést kínál.

username (varchar(50) utf8mb4\_general\_ci): Felhasználó beceneve vagy felhasználónév, maximális hossza 50 karakter.

role(tinyint(4):a felhasználó adatköre:0 user ,1 admin.

userImage(varchar(255)utf8mb4\_general\_ci):kép a felhasználóról.

Birthday(date):a felhasználó születési dátuma.



* + 1. ordering tábla

A "ordering" táblában rögzítjük a felhasználók által leadott rendeléseket és azok részleteit.

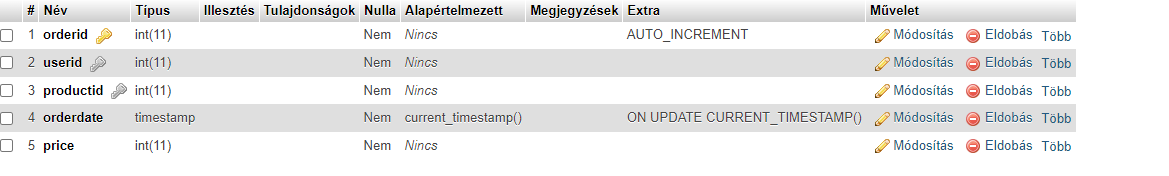
orderid (Elsődleges int(11) AUTO\_INCREMENT): Egyedi azonosító, amely minden új rendelés hozzáadásakor automatikusan növekszik. Az elsődleges kulcsként szolgál a táblában.

userid (int(11)): Azonosító, amely a rendelést leadó felhasználóhoz kapcsolódik. Ez egy külső kulcs, amely a "user" tábla "userid" mezőjére hivatkozik.

orderdate (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP): A rendelés időpontját rögzíti. Alapértelmezett értékként a rendszer aktuális időpontját veszi fel.

productid (int(11)): A rendelt termék azonosítóját rögzíti.

price (int(11)): A termék ára.



* + 1. alkatresz tábla

A "alkatresz" tábla egy kulcsfontosságú része a webshopunk adatbázisának, amelyben rögzítjük a különböző termekeket és azok részleteit.

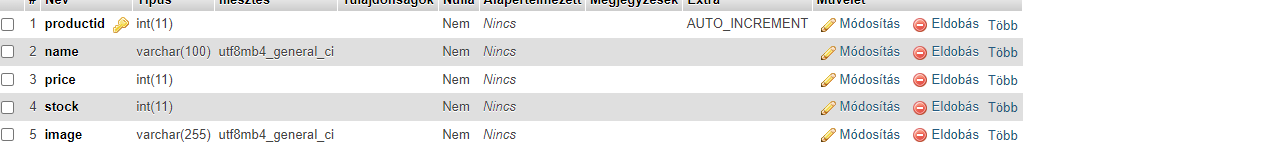
productid (Elsődleges int(11) AUTO\_INCREMENT): Egyedi azonosító, amely minden új termék hozzáadásakor automatikusan növekszik. Az elsődleges kulcsként szolgál a táblában.

name (varchar(50) utf8mb4\_general\_ci): A termék nevét tárolja.

Price (int(11)): A termék árát tárolja.

Stock(int11):kiírja,hogy hány darab található az adott termékből.

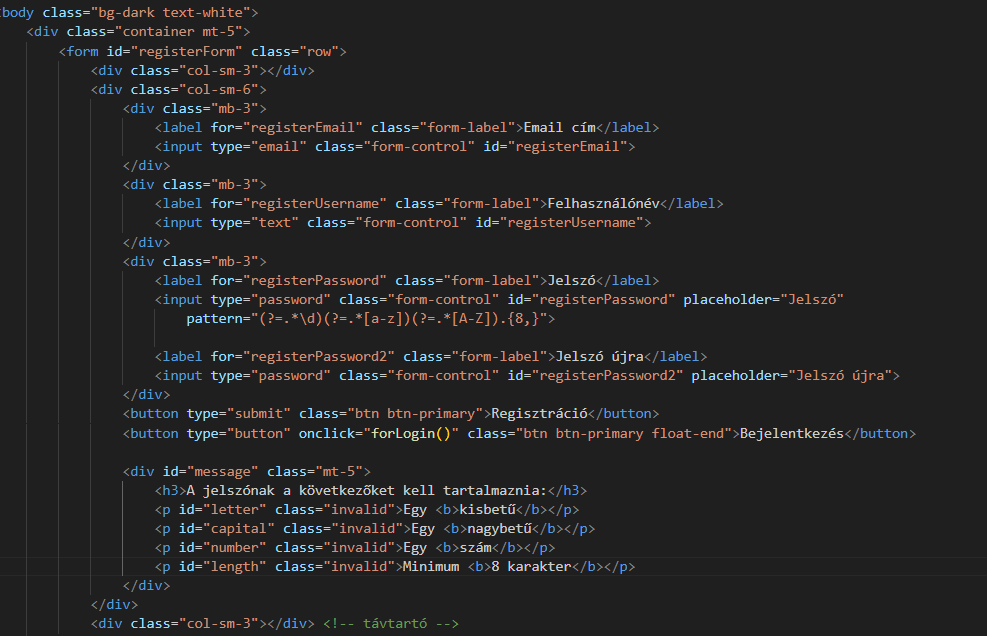
Image(varchar(255)utf8mb4\_general\_ci):A termékekről lehet képet beszúrni ezen keresztül.



1. Regisztráció



A felhasználói regisztráció oldal sokféle webes alkalmazásban és szolgáltatásban kulcsfontosságú szerepet tölt be. Célja, hogy lehetővé tegye az új felhasználók számára, hogy létrehozzák saját fiókjaikat, így hozzáférjenek a weboldal vagy az alkalmazás zárt vagy személyre szabott részeihez. Ez a kód egy regisztrációs űrlap HTML kódját tartalmazza, amely egy webes felületen jelenik meg. Az űrlap lehetővé teszi a felhasználók számára, hogy regisztráljanak egy adott weboldalra vagy alkalmazásba, megadva bizonyos személyes adatokat, mint például email cím, felhasználónév és jelszó.



Az alábbiakban ismertetem, hogy a kód milyen elemekből áll és mit csinál mindegyik:

* 1. Külső konténer és űrlap struktúra: A kód egy div elemmel kezdődik, amely egy container mt-5 osztályokkal rendelkezik, biztosítva a megfelelő pozicionálást és külső margót (mt-5 = margin-top: 3rem). Ezen belül van egy űrlap (form), amely az id attribútummal "registerForm"-ként van azonosítva. Ez az űrlap három oszlopra van osztva a Bootstrap grid rendszerének használatával, ahol a középső oszlop (col-sm-6) tartalmazza a tényleges regisztrációs mezőket, és két üres oszlop (col-sm-3) van mellette két oldalon távtartóként.
  2. Input mezők: Az űrlap három fő input mezőt tartalmaz:

Email cím: Egy email input mező, amelyet az id="registerEmail" azonosít. Ez a mező gyűjti a felhasználó email címét.

Felhasználónév: Egy szöveges input mező, amelyet az id="registerUsername" azonosít. Itt lehet megadni a kívánt felhasználónevet.

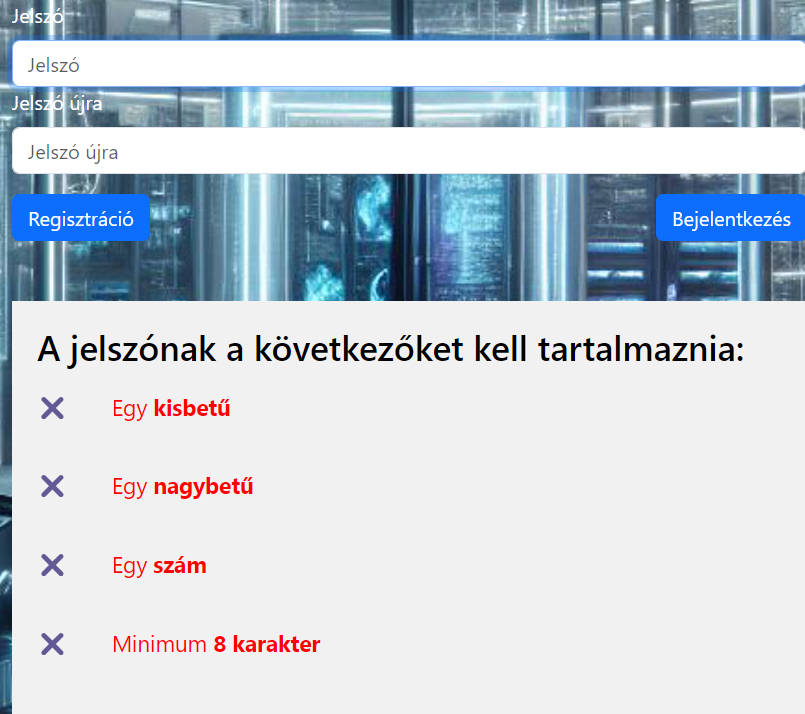
Jelszó és Jelszó újra: Két jelszó input mező, amelyek az id="registerPassword" és id="registerPassword2" azonosítókkal rendelkeznek. A jelszó mező egy biztonsági mintát (pattern) is tartalmaz, ami megköveteli, hogy a jelszónak tartalmaznia kell legalább egy számot, egy kisbetűt, egy nagybetűt és legalább 8 karakter hosszúnak kell lennie.

* 1. Gombok: Két gombot tartalmaz:

Regisztráció: Egy submit típusú gomb, amely elküldi az űrlapot.

Bejelentkezés: Egy button típusú gomb, amely egy onclick eseménykezelőt tartalmaz (forLogin() függvény), egy bejelentkezési űrlap megjelenítésére szolgál.

* 1. Jelszó szabályok megjelenítése: Az id="message" azonosítóval rendelkező div tartalmaz négy p elemet, amelyek a jelszó követelményeit írják le (kisbetű, nagybetű, szám, minimum 8 karakter). Ezek az elemek kezdetben az "invalid" osztállyal rendelkeznek, ami azt jelzi, hogy a jelszó még nem felel meg ezeknek a követelményeknek.



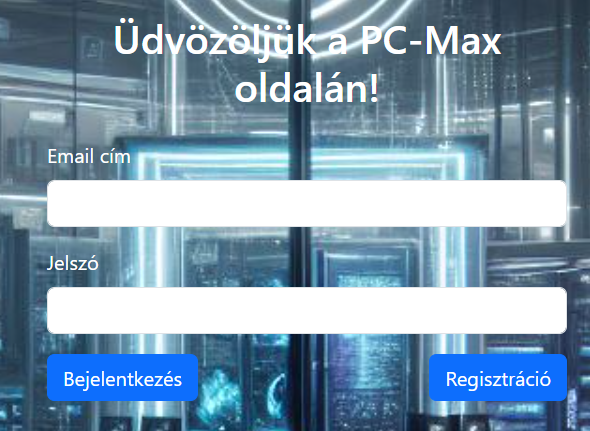
Hogyha a két jelszót nem ugyanúgy írjuk le felugró ablakkal jelzi a program.



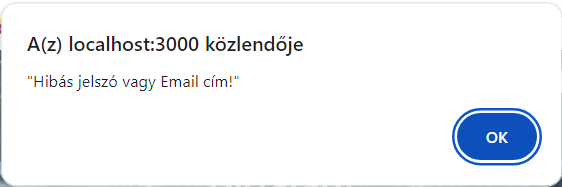
Amennyiben a regisztráció sikeres, ezt a weboldal egy felugró ablakkal jelzi.



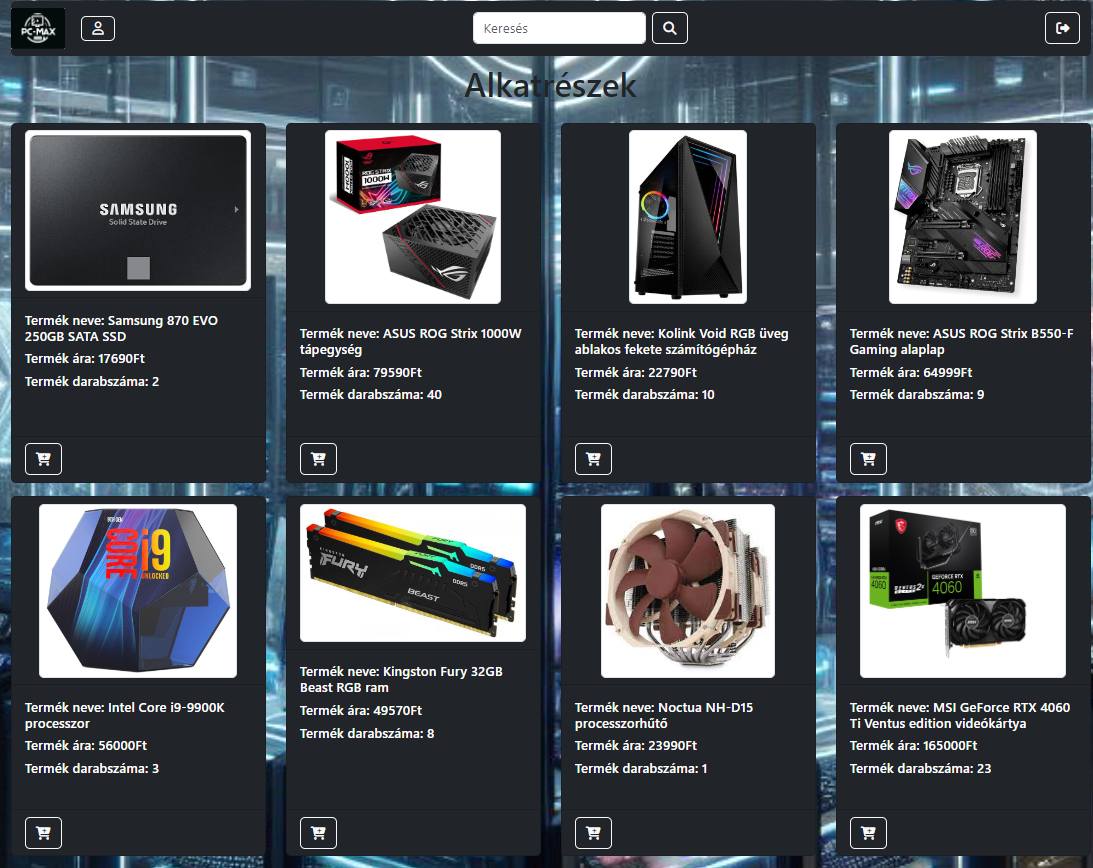
1. Bejelentkezés



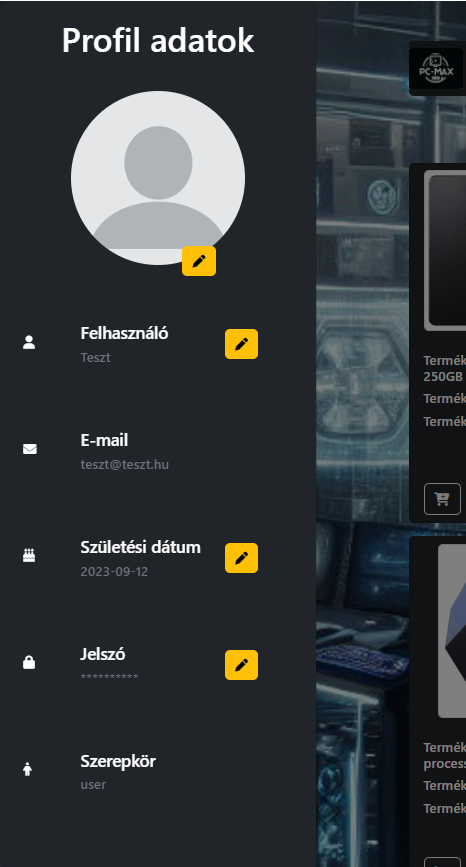
A bejelentkezéshez meg kell adnunk a regisztráció által megadott Email címet és jelszót, és ez után a bejelentkezés gombra kattintva be is lépünk a fiókunkba, viszont ha adat elírás miatt nem enged bejelentkezni ezt jelzi az oldal.



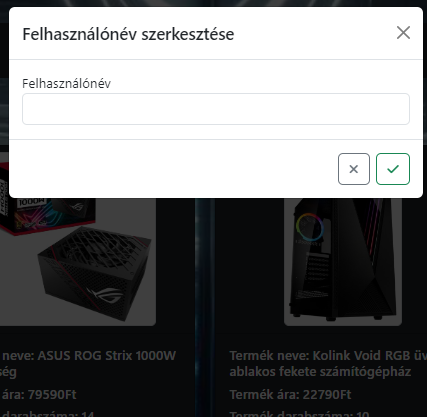
* 1. Weboldal felhasználó részéről



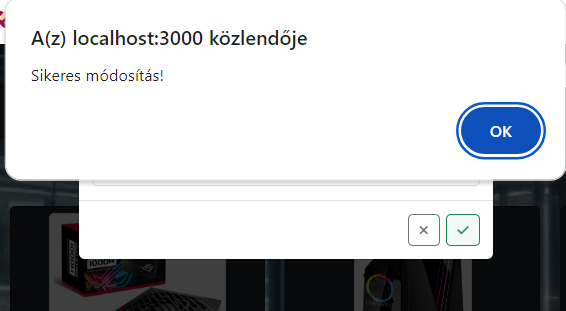
* 1. Profil adatai



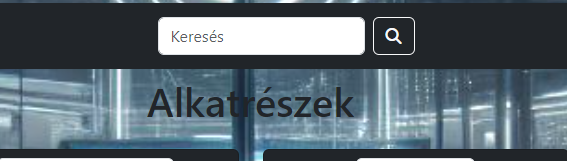
ennél a résznél a felhasználó adatait lehet változtatni a szerkesztés gombra kattintva, amellyel egy felugró ablakot hozunk elő, amiben átírhatjuk a kiválasztott adatot.



Változtatás után a pipára kattintva felugró ablakkal jelzi a sikeres módosítást.



* 1. Keresés

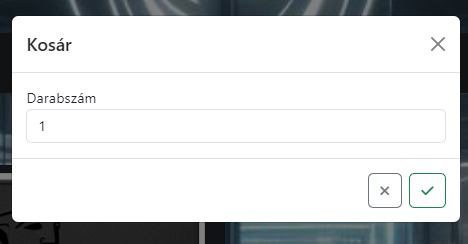


Ide ha beírja a felhasználó a keresett alkatrészt a keresés gomb lenyomására behozza a megtalálható alkatrészeket

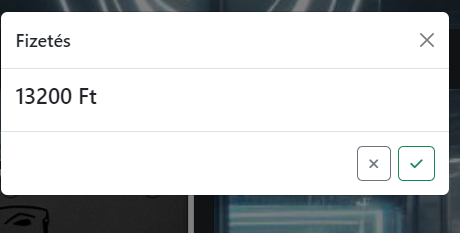


* 1. Vásárlás

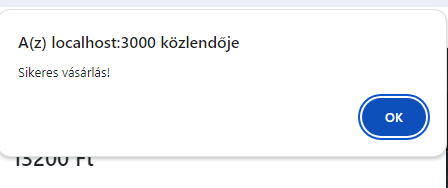
A kosár gombra kattintva lehet kiválasztani,hogy hány darabot szeretnénk vásárolni az adott alkatrészből.



A pipára kattintva kihozza hogy mennyibe kerül a termék.

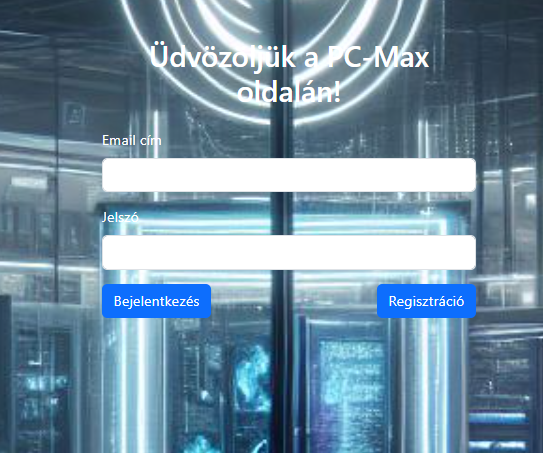


Amint befejeztük a vásárlást a pipára kattintva,a weboldal egy felugró ablakkal jelzi a sikeres vásárlást,és ezután a főoldalra vezet vissza.

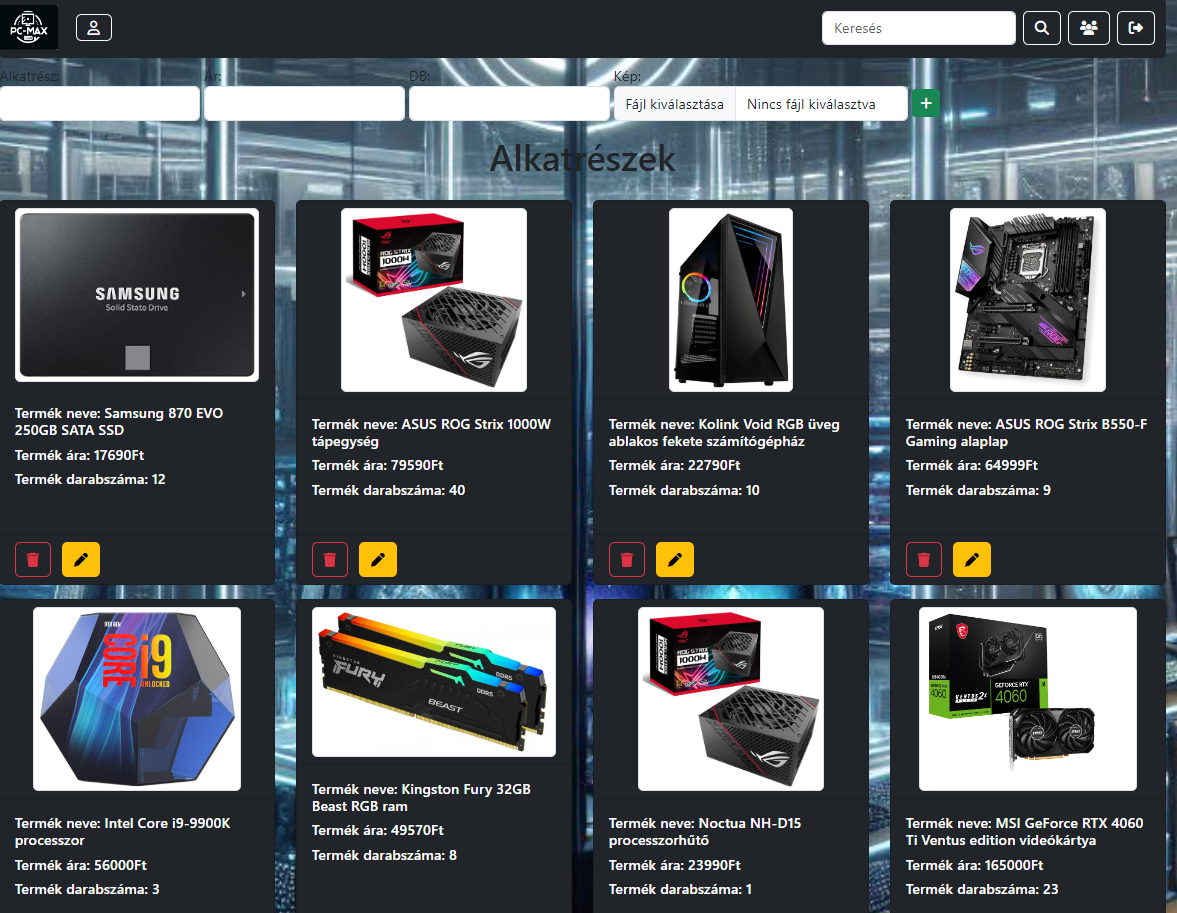


* 1. Kijelentkezés

A kijelentkezés gombra kattintva ki léptet a fiókunkból és újra a bejelentkező/regisztráló oldalon találjuk magunkat.



1. Weboldal admin részéről

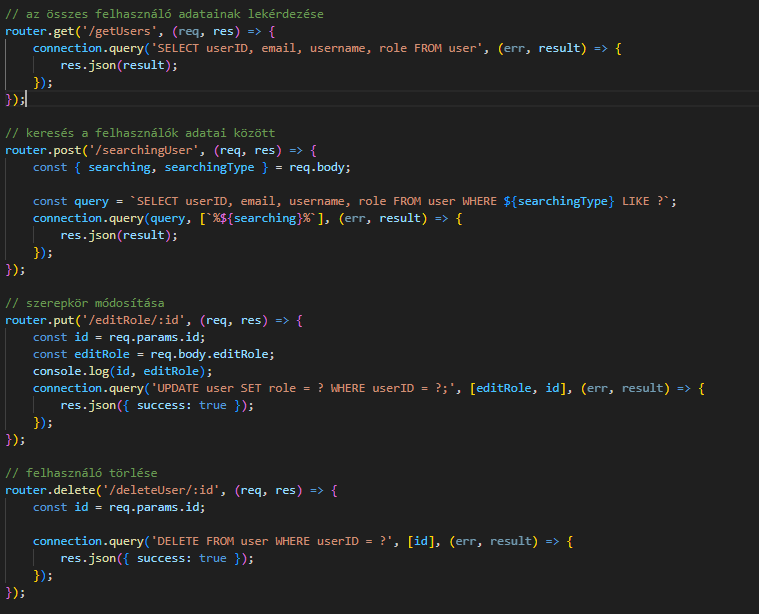


* 1. Felhasználók

Az admin megtudja nézni,hogy hány fiók van regisztrálva a weboldalon a fent található felhasználók gomb segítségével, és hogy milyen szerepkörrel rendelkeznek,illetve ezt módosítani tudják,és a fiókot törölni is tudják.



Ez a gomb JavaScript kódban lett írva.



**GET /getUsers**: Ez az útvonal a felhasználók adatainak lekérésére szolgál. A szerver a user táblából kiválasztja a felhasználók azonosítóját (userID), e-mail címét (email), felhasználónevét (username) és szerepkörét (role), majd ezeket az adatokat JSON formátumban visszaküldi a kliensnek.

**POST /searchingUser**: Ez az útvonal a felhasználók keresését teszi lehetővé. A kliens által megadott keresési feltételek alapján a szerver lekérdezést hajt végre a user táblában, és visszaadja a talált felhasználók adatait JSON formátumban.

**PUT /editRole/:id**: Ez az útvonal a felhasználó szerepkörének módosítását teszi lehetővé az azonosító alapján. A kliens által megadott új szerepkört az adott felhasználóhoz rendeli a user táblában.

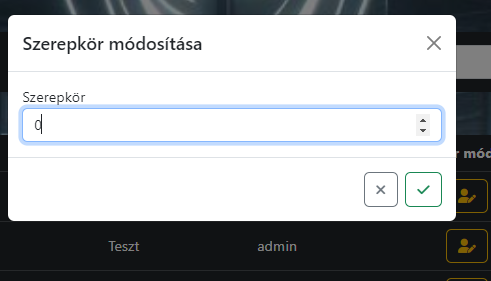
**DELETE /deleteUser/:id**: Ez az útvonal a felhasználó törlését végzi el az azonosító alapján a user táblából.

Az alkalmazás SQL lekérdezéseket használ a MySQL adatbázishoz való hozzáféréshez (connection.query), és az Express.js keretrendszer útvonalvezérlési funkcióit (router.get, router.post, router.put, router.delete) használja a különböző HTTP kérések kezelésére. Az alkalmazás JSON formátumban kommunikál a klienssel az adatok cseréjére. Fontos megjegyezni, hogy a biztonsági és adatkezelési szempontokat is figyelembe kell venni az ilyen típusú alkalmazások fejlesztése során.

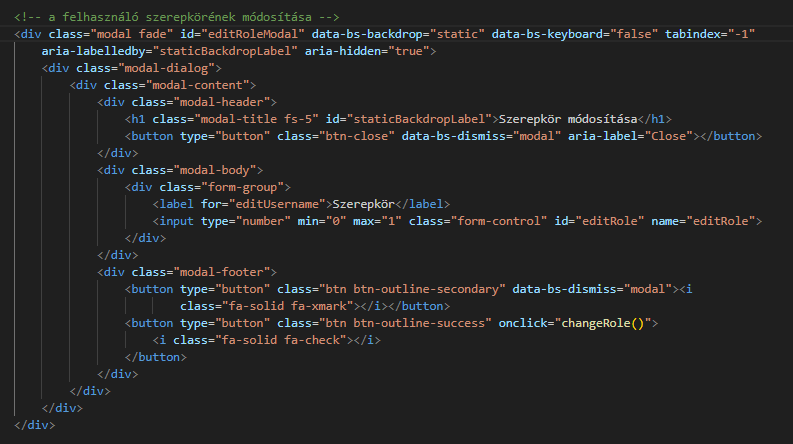
Ezen az oldalon megtalálhatóak a létrehozott felhasználók ID-jei amelyet minden felhasználó kap a regisztráció után,illetve a felhasználó adatai.

* 1. A szerepkör módosítása

A szerepkör módosítása gomb segítségével az admin tudja változtatni a felhasználó szerepkörét.



Ez a gomb HTML kódban lett írva.



Ez a kód egy Bootstrap modális ablakot definiál, amely lehetővé teszi a felhasználó szerepkörének módosítását.

**Modal ID és tulajdonságok**: A modal egyedi azonosítója a "editRoleModal". A **data-bs-backdrop**="static" és data-bs-keyboard="false" tulajdonságok azt jelentik, hogy a háttérre kattintáskor és a billentyűzetre kattintáskor sem zárja be a modális ablakot, ami egy statikus modált eredményez.

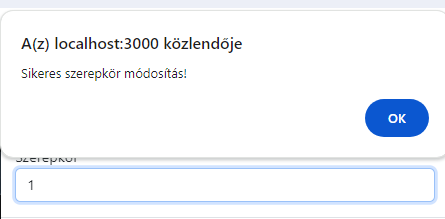
**Modal tartalma**: A modal tartalma a modal-content osztályba van csomagolva. Tartalmaz egy modal-header-t, amelyben a modális cím található, és egy bezárás gomb (btn-close), amely segítségével a felhasználó bezárhatja a modált. A modális test (modal-body) tartalmaz egy űrlapot egyetlen bemeneti mezővel, amely a felhasználó új szerepkörét teszi lehetővé.

**Szerepkör input mező**: Az input mező egy szám típusú mező (type="number"), ahol a felhasználó megadhatja az új szerepkört. A min és max attribútumok segítségével korlátozható, hogy csak bizonyos értékeket lehessen megadni (0 vagy 1 például).

**Modal láb**: A lábban két gomb található. Az egyik gomb a bezárás gomb (btn btn-outline-secondary), amely a modális ablak bezárását végzi el, a másik pedig a módosítást végrehajtó gomb (btn btn-outline-success), amely changeRole() nevű JavaScript függvényt hív meg a szerepkör módosításához.

Hogyha a szerepkör alá 0 azonosítót írunk akkor a felhasználó User szerepkörben van,hogyha 1-es azonosítót írunk akkor pedig a felhasználó az Admin szerepkörben van.

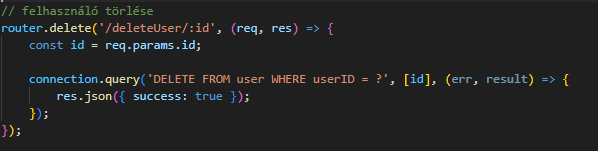
Amint rányomunk a pipára a weboldal felugró ablakkal jelzi.



* 1. A felhasználó törlése

A felhasználó törlése gomb segítségével törölni tudjuk a felhasználót,rányomásra felugró ablakkal rákérdez,hogy biztos törölni szeretnénk-e.

Ez a gomb JavaScript kódban lett írva.

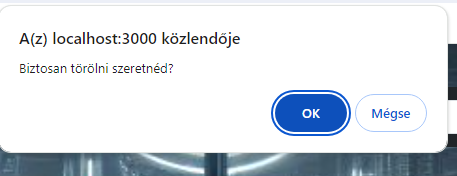


**Útvonal**: Az útvonal definiálása során a router.delete függvény használatával meghatározzuk a HTTP DELETE kérést. Az útvonal elérési útja /deleteUser/:id, ahol :id egy dinamikus útvonal paraméter, ami az adott felhasználó azonosítóját tartalmazza.

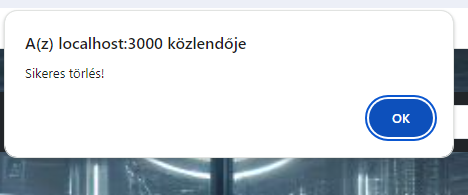
**Request paraméterek**: Az útvonalból a req.params.id segítségével kiolvassuk az azonosítót, amelyet a kliens küldött.

**Adatbázis művelet**: A connection.query segítségével SQL DELETE parancsot hajtunk végre, amely törli az adott felhasználót az adatbázisból. Az id azonosító alapján kerül a törlés végrehajtásra, amelyet a felhasználó a kliens oldalon küldött.

**Válasz küldése**: Miután a törlés megtörtént az adatbázisban, a szerver JSON választ küld vissza a kliensnek. A válasz tartalma { success: true }, ami jelzi, hogy a törlés sikeresen megtörtént.

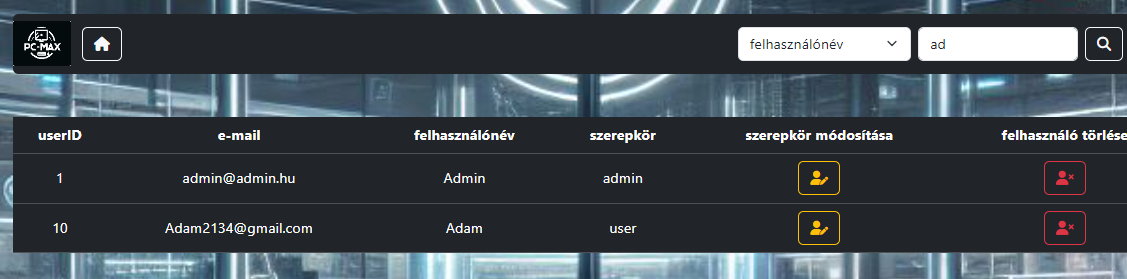


Amennyiben az ok-ra kattintunk a profil törlődni fog,amit egy felugró ablakkal jelez.

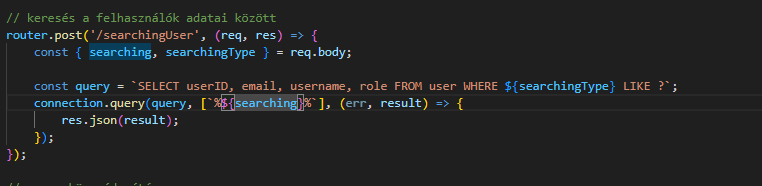


* 1. Felhasználó keresés

A felhasználók fülben még keresni is lehet adatok szerint(e-mail,felhasználónév,és szerepkör szerint)amelyet a keresés gomb segítségével lehet kivitelezni.



Ez JavaScript kódban lett írva.



**Útvonal és metódus:** Az útvonal definiálása során a router.post függvényt használjuk, ami azt jelzi, hogy a keresést csak POST kéréssel lehet elérni. Az útvonal elérési útja /searchingUser.

**Request adatok:** A kliens által küldött keresési paramétereket a req.body objektumból olvassuk ki. A searching változó tartalmazza a keresett kifejezést, míg a searchingType a keresés típusát határozza meg.

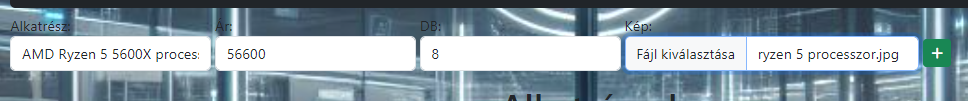
**Lekérdezés összeállítása:** A kódban egy SQL lekérdezést állítunk össze a kapott paraméterek alapján. A lekérdezés a user táblából kiválasztja azokat a rekordokat, ahol a megadott searchingType mező tartalmazza a searching kifejezést (LIKE operátorral).

**Adatbázis művelet:** A lekérdezést végrehajtjuk a connection.query segítségével. A lekérdezésben a ? helyőrzőt helyettesítjük a keresett kifejezéssel és az SQL injekció elkerülése érdekében az % karakterekkel kiegészítjük. Az adatbázis válaszát a result változóba mentjük.

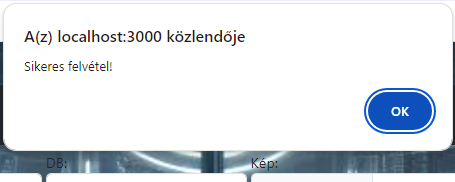
**Válasz küldése**: Miután megkapjuk az adatbázis válaszát, azt JSON formátumban visszaküldjük a kliensnek a res.json(result) segítségével, így a kliensnek lehetősége van megjeleníteni vagy további műveleteket végezni az eredménnyel.

* 1. Alkatrész feltöltés

Alkatrész feltöltésnél meg kell adni az alkatrész nevét,árát,a darabszámát és még képet is lehet csatolni az adott alkatrészről.



A + gombra kattintva fel is tölti az adatot,amelyet felugró ablakkal jelez az oldal.



Ez a gomb HTML kódban lett megírva.



**Form ID**: Az űrlap egyedi azonosítója a "create-alkatresz". Ez az azonosító segíthet azonosítani az űrlapot a JavaScript-ben vagy CSS-ben.

**Input mezők:** Az űrlap négy bemeneti mezőt tartalmaz:

**"name"** - Az alkatrész nevének megadására szolgáló szövegmező.

**"price"** - Az alkatrész árának megadására szolgáló számmező. A "number" típus beállítása biztosítja, hogy csak számokat lehessen megadni.

**"stock"** - Az alkatrész készletmennyiségének megadására szolgáló számmező.

**"image"** - Az alkatrészhez tartozó kép feltöltésére szolgáló fájlbeviteli mező.

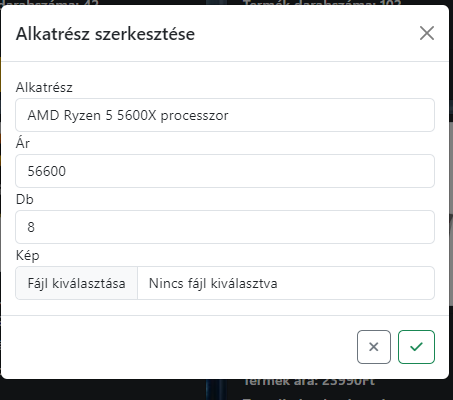
**Gombok**: Az űrlap alatt található egy gomb, amely a <button> elem segítségével van megvalósítva. Ez a gomb típusa "submit", így amikor rákattintnak, az űrlap elküldődik.

**alkatresz-list div**: Ez a <div> elem az újonnan létrehozott vagy szerkesztett alkatrészek listáját fogja megjeleníteni.

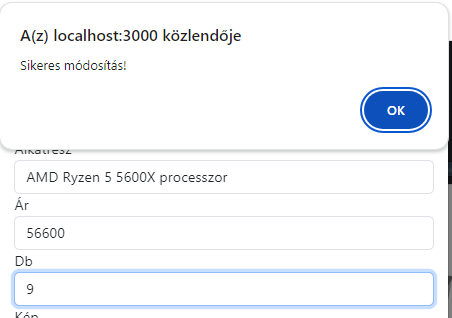
* 1. Alkatrész szerkesztése/törlése

A feltöltött adatokat szerkeszteni is tudjuk a törlés/szerkesztés gomb segítségével.

Szerkesztésnél megjelenik egy ablak,amelyben a beírt adatokat tudjuk szerkeszteni.



A pipára kattintva a változtatott adat frissül, Sikeres módosítás esetén az oldal felugró ablakkal jelzi.



Törlést a törlés gomb segítségével tudjuk törölni a terméket, gombnyomásra felugró ablakkal jelzi a program,hogy biztos törölni szeretnénk-e a terméket.

Ez a gomb JavaScript kódban van megírva



**deleteUser(userID):** Ez a függvény felelős a felhasználó törléséért. A userID paraméter azonosítja meg a törlendő felhasználót.

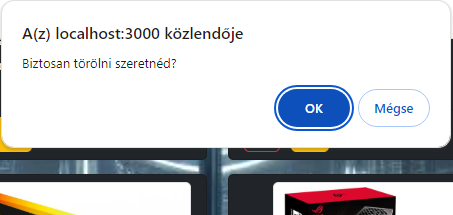
Törlés megerősítése: Először egy megerősítő üzenet jelenik meg a felhasználónak, hogy biztosan törölni szeretné-e a felhasználót. Erre a confirm() függvényt használjuk. Ha a felhasználó megerősíti a törlés szándékát, akkor folytatjuk a törlési folyamatot, egyébként a függvény visszatér és nem történik semmi.

**Fetch API használata**: Ha a felhasználó megerősítette a törlés szándékát, akkor egy fetch kérést indítunk a szerver felé a megfelelő végpont felé (/deleteUser/${userID}). A törlési kérés típusa DELETE, és JSON formátumban küldjük el a kérést, ami a headers objektumban van definiálva.

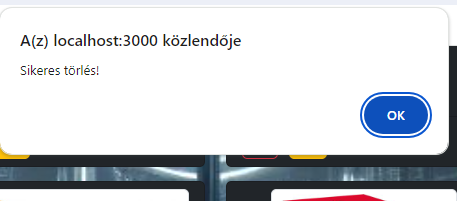
**Válasz feldolgozása:** Miután megkapjuk a választ a szerverről, az adatokat JSON formátumban olvassuk ki (res.json()). Ezután ellenőrizzük, hogy a törlés sikeres volt-e (data.success), és ez alapján jelenítünk meg egy megfelelő üzenetet.

**Sikeres törlés**: Ha a törlés sikeres volt, akkor egy alert() ablakban jelezzük a felhasználónak a sikeres törlést, majd a getUsers() függvényt hívjuk meg a felhasználók frissítéséhez.

**Hiba esetén**: Ha valamilyen hiba történik a törlés folyamata során, akkor egy másik alert() ablakban jelezzük a felhasználónak a hibát.

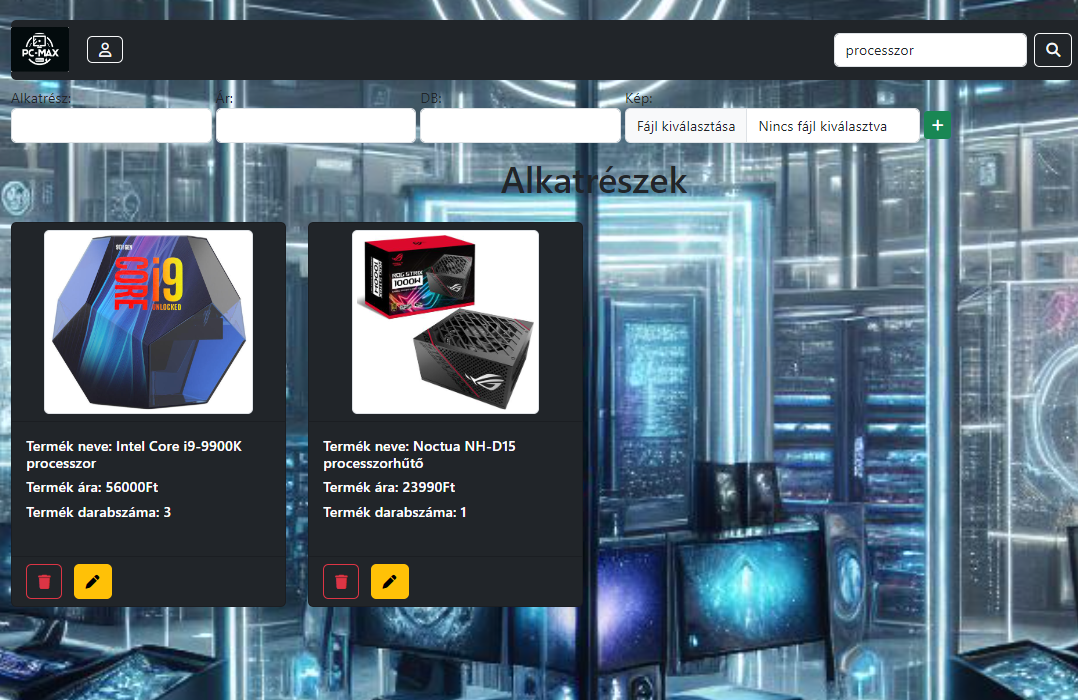


Amennyiben az Ok-ra kattintunk a termék sikeresen törlődik,amit felugró ablakkal jelez az oldal.

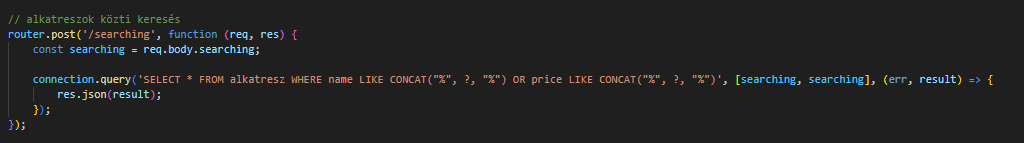


* 1. Alkatrész keresés

Az alkatrészekre ugyanúgy rátudunk keresni mint a User szerepkörű felhasználó,és itt is tudjuk szerkeszteni illetve törölni a talált alkatrészt.



Amennyiben olyan alkatrészt írunk be ami nem található meg a weboldalon,akkor azt a weboldal Nincs találat felirattal jelzi.



**Útvonal és metódus**: Az útvonal definiálása során a router.post függvényt használjuk, ami azt jelzi, hogy a keresést csak POST kéréssel lehet elérni. Az útvonal elérési útja /searching.

**Request adatok**: A kliens által küldött keresési paramétert a req.body objektumból olvassuk ki. A searching változó tartalmazza a keresett kifejezést.

**Lekérdezés összeállítása**: A kódban egy SQL lekérdezést állítunk össze a kapott paraméter alapján. A lekérdezésben a alkatresz táblából kiválasztjuk azokat a rekordokat, ahol az alkatrész neve vagy ára tartalmazza a keresett kifejezést (LIKE operátorral).

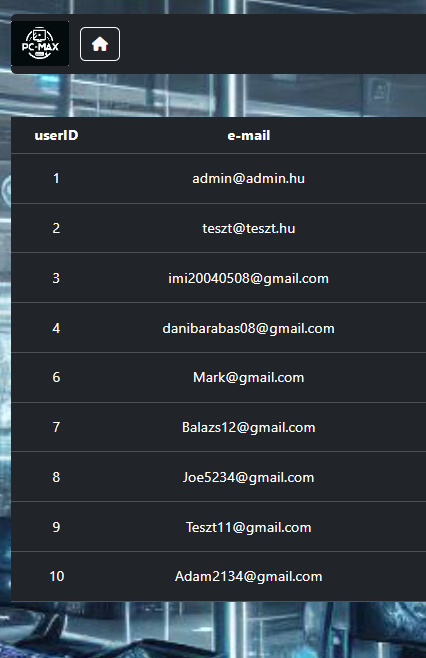
**Adatbázis művelet**: A lekérdezést végrehajtjuk a connection.query segítségével.

**Válasz küldése**: Miután megkapjuk az adatbázis válaszát, azt JSON formátumban visszaküldjük a kliensnek a res.json(result) segítségével, így a kliensnek lehetősége van megjeleníteni vagy további műveleteket végezni az eredménnyel.



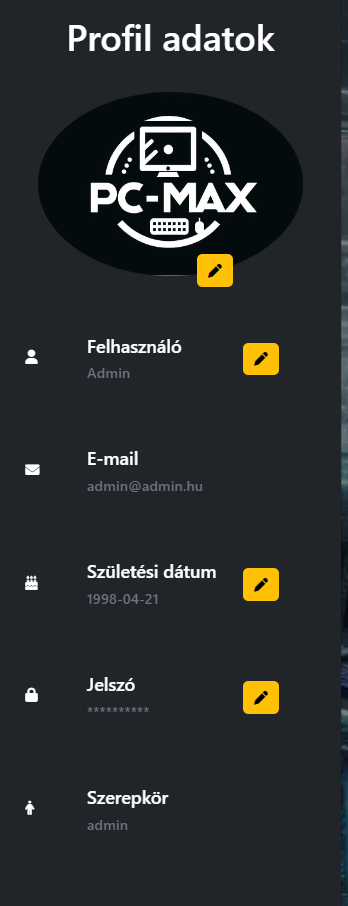
* 1. Visszalépés a főoldalra

Hogyha a keresés után visszaszeretnénk lépni a főoldalra ezt is megtehetjük a bal szélen megtalálható gomb lenyomásával.



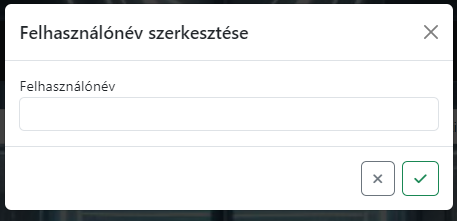
* 1. Admin profil változtatása

Az Admin profil adatait is ugyanúgy lehet változtatni,mint a felhasználói profilban.

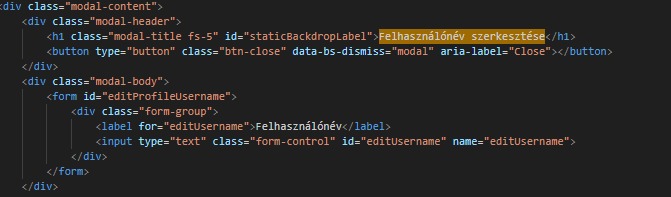


Ez a kód felel a felhasználó adatainak lekérdezéséért és az oldalsó menübe írásáért.

A szerkesztés gombra kattintva, amellyel egy felugró ablakot hozunk elő, amiben átírhatjuk a kiválasztott adatot.



Ez a gomb HTML kódban van megírva

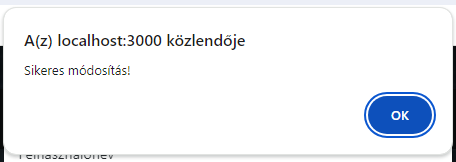


**Modal-header**: Ez a rész tartalmazza a modális ablak fejlécét. A modal-title osztályú címkében jelenik meg a "Felhasználónév szerkesztése" felirat, ami az ablak címe. A btn-close osztályú gomb lehetővé teszi az ablak bezárását.

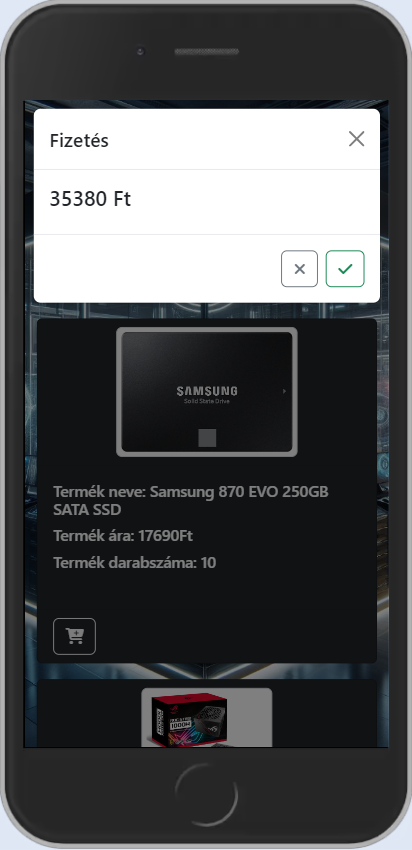
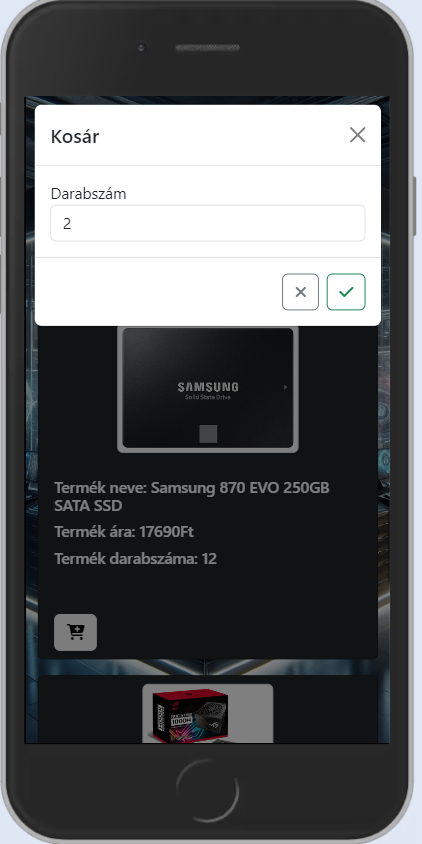
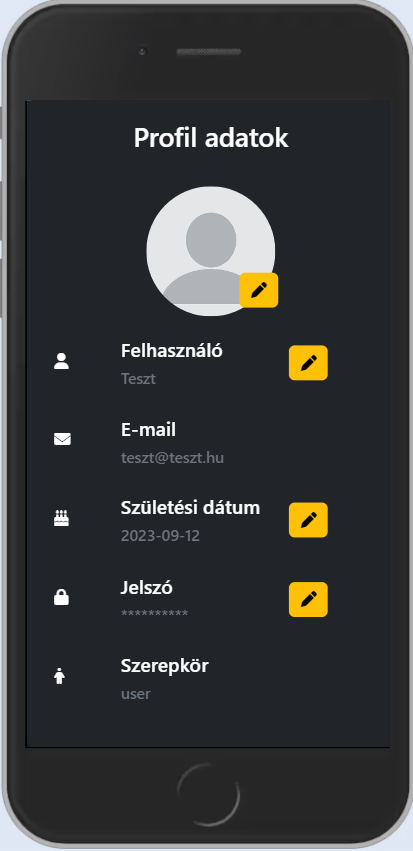
**Modal-body**: Ez a rész tartalmazza a modális ablak tartalmát. Egy űrlapot definiál, amelyben a felhasználó megadhatja az új felhasználónevet. A form-group osztályú div segítségével formázva van a felhasználói felület, és egy címke (label) és egy beviteli mező (input) van a felhasználónév megadásához.

**Input mező**: Az editUsername nevű input mező lehetővé teszi a felhasználó számára, hogy megadja az új felhasználónevet. A form-control osztály biztosítja, hogy a beviteli mező megfelelően formázva legyen a Bootstrap keretrendszer alapján.

Változtatás után a pipára kattintva felugró ablakkal jelzi a sikeres módosítást.

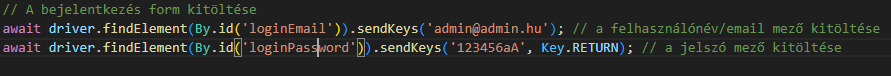


1. A weboldal reszponzív nézetben



1. Tesztelés

a tesztelés Javascript kódban történik a Selenium Webdriver ,amelyet az „npm i selenium-webdirever”segítségével telepítettünk,amelynek a feladata, hogy elindítás után jelentkezzen be a kódban megadott fiókba,és a konzolban írja ki ha a teszt sikeres illetve ha hiba van benne jelezze ki.



Sikeres belépés esetén így írja ki a konzol:



Amennyiben a belépés nem sikeres,így írja ki a konzol:



Selenium webdriver kódolása package.json-ban:

