1. Bevezetés
   1. A vizsgaremek témája

A vizsgaremeknek a témája egy közösségi fórum. A fórum fő célja, hogy a felhasználók megvitathatnak egyes kérdéseket vagy segítségeket kérhetnek különböző témákban. A másik fő eleme a piactér, ahol a felhasználók a már nem használt számítástechnikai eszközeiket árusíthatják. Az fő ihletet a hardverapro.hu, és az itcafe.hu fórum része adta.

* 1. A program funkciói

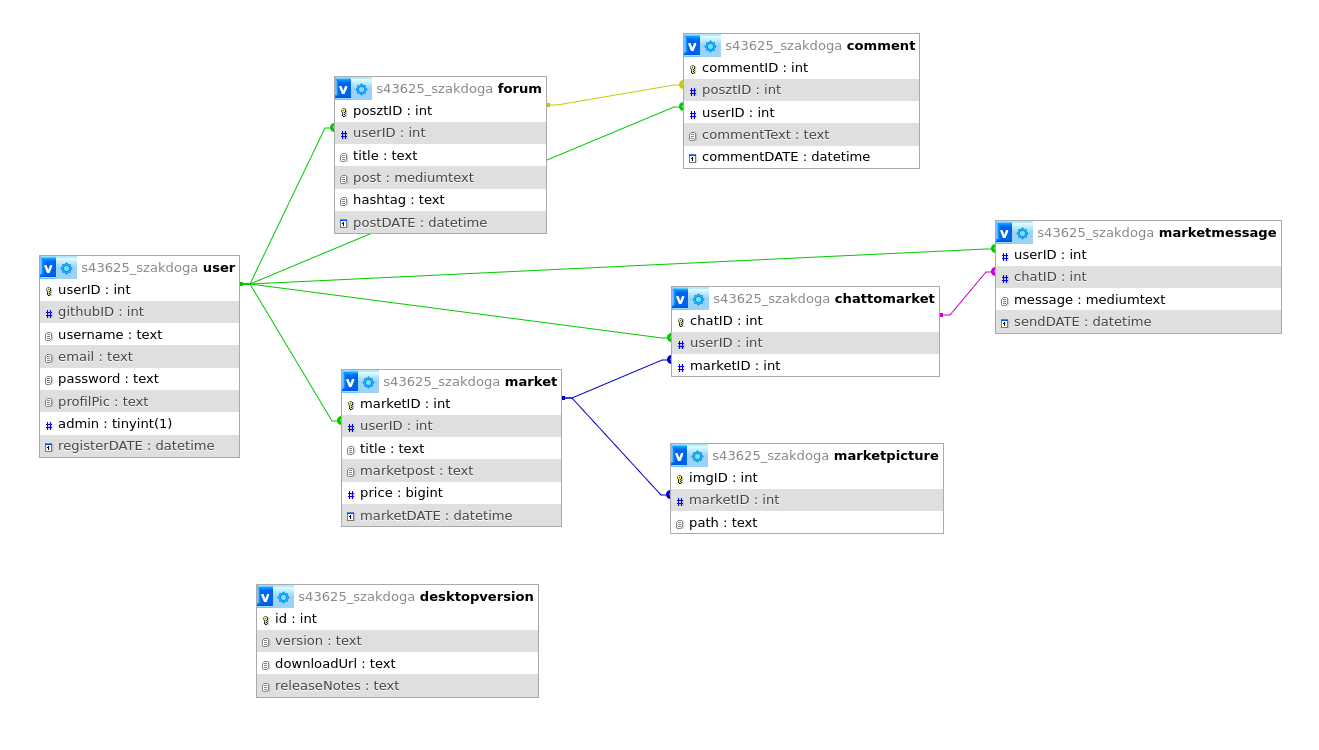
Az első funkció, amivel az átlagos felhasználó találkozni fog a weboldalon az a bejelentkezés és a regisztráció oldala. Bejelentkezve a főoldalon találjuk magunkat, ahol a fórum szekció kapott helyet. Itt egyből látni lehet a saját és a mások által létrehozott posztokat. Egy posztra rá nézve láthatjuk, hogy ki tette közzé és hogy milyen időpontban tette azt. A posztok között lehetőségünk van keresni is felhasználó, cím vagy tartalom alapján. Így olvashatunk egy adott témáról, vagy egy adott felhasználótól. Abban az esetben ha nekünk is kérdésünk van, vagy egy adott témát szeretnénk kifejteni más felhasználók társaságával, az erre megalkotott gomb segítségével megalkothatjuk saját posztunkat is. A poszt megnyitásával elérhetővé válik a komment szekció,itt láthatjuk az esetleges hozzászólásokat. Ugyan itt tudjuk törölni a posztunkat és szerkeszteni is pár kattintással. A főoldalon található a saját profilunkhoz vezető gomb is , itt kapunk lehetőséget a jelszavunk megváltoztatására. Ha tovább lépünk a piactérre megtaláljuk a hirdetéseket, vagy saját eszközöket árusíthatunk. Az eszközeink árusításakor képeket és leírást csatolhatunk a hirdetéshez. Egy már meglévő hirdetés alatt akár privát beszélgetést is kezdhetünk a hirdetővel, így lehetőségünk adódik kérdéseket feltenni vagy tárgyalni az adásvételről. Továbbiakban elfelejtett fiók-jelszó esetén rendszerünk bérelt Mail szerverünkön keresztül továbbít egy emailt a megadott címre egy 6 számjegyű hitelesítő kódot, mely segítségével vissza tudjuk állítani elfelejtett jelszavunkat, melyre weboldalunkon külön oldal részt alakítottunk ki a felhasználók számára. Ezt a bejelentkezésnél találhatjuk a regisztráció hivatkozása alatt.

Az asztali alkalmazás az adminisztrációs feladatkört fedi le. Átfogóan, hasonlóan egy bejelentkezési felülettel találkozunk, mely során a program ellenőrzi, a felhasználó rendelkezik-e a megfelelő felhatalmazással. Amennyiben igen, a program tovább engedi az adminisztrációs felületre. Ellenkező esetben hibaüzenettel megakad a bejelentkezés során. Az adminisztrációs felületen több füllel találkozhatunk a posztok-, hirdetések-, felhasználók-, adminisztrátorok listájával. A felhasználók fülön tilthatunk - figyelmeztethetünk felhasználót esetleg véglegesen is. A fellebbezéseket is innen kezelhetjük, melyeket a figyelmeztetések után tehetnek meg a felhasználók. Innen értesíthetőek a felhasználók, esetlegesen privát chat-ben beszélhetik meg a problémát egy adminisztrátorral. A felhasználók által lefoglalt tárgyak foglalását is feloldhatjuk - azon eladó kérésére, amely létrehozta a hirdetést. A poszt fülön rendelkezünk törléssel - annak módosításával, jelentéseket kezelhetünk, melyeket a weboldalon a felhasználók tehetnek meg. Hasonlóan működik a hirdetések fül is. Törölhetünk, módosíthatunk, figyelmeztetéseket oszthatunk ki. Az adminisztrátorok fülön láthatjuk az összes adminisztrátor nevét. Ebben az esetben teljeskörű hatalmat kap a felhasználó a posztok, hirdetések és felhasználók kezelésére. Továbbá a program rendelkezik egy élő admin chat-el, melyen az adminisztrátorok ki értékelhetik egymás honorálását. Ezt az átlagos felhasználó persze nem láthatja. Fontos, az adminisztrátorok között nincsen hierarchia, minden adminisztrátor hozzá tehet vagy elvehet a csapat munkájából.

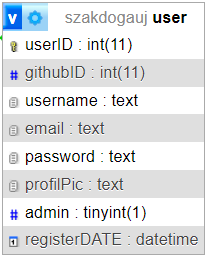
* 1. Célközönség

Ez az oldal hasznára válhat annak aki a legjobb ajánlatokat keresi, túladna régi eszközein, vagy különböző témákban szeretne beszélni másokkal korosztálytól függetlenül.

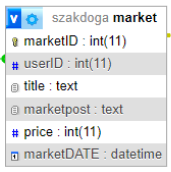
1. Fejlesztői dokumentáció
   1. Adatbázis



Itt látható relációs adatbázis tárolja programunk adatait, mely phpMyAdmin-ban lett létrehozva, szerkesztve a táblák közötti kapcsolat létrehozva . A phpMyAdmin xampp-ban lett futtatva Apache, MySQL szerver segítségével lett futtatva Localhoston a tesztelés idején. Később az ikola által biztosított nodejs.dszcbaross.edu.hu weboldalon keresztül lett futtatva a Backend-el együtt. Az adatbázis típusa InnoDB. Az InnoDB egy általános célú tárolómotor, amely egyformán tartja a nagy megbízhatóságot és teljesítményt. A MySQL 8.0-ban az InnoDB az alapértelmezett MySQL tárolómotor . Az adatbázis teljes egészében a user tábla köré lett felépítve, mivel weboldalunk csak bejelentkezés útján elérhető. Az adatbázis kiinduláskor 3 alap táblát tartalmazott mely az idő folyamán további táblákkal egészült ki.

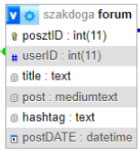
* + 1. A User tábla  
       

A user tábla felhasználói adatok tárolására szolgál. Az user id mező automatikusan generálódik regisztráláskor , típusa int, max értéke 11, autoincrement. Elsődleges kulcs, továbbá kapcsolatban áll a Market, Forum, Comment, MarketMessage és ChatToMarket táblában található userID mezővel.  
A githubID a GitHub-os bejelentkezés által létrehozott fiókok azonosítója, tipusa int. Az username mező a felhasználónév , regisztráláskor kerül megadásra , típusa text. Az  Email mező a felhasználó email címe , regisztráláskor kerül megadásra , típusa text. A jelszó mező a felhasználó hesh-el titkosított jelszava , regisztráláskor kerül megadásra , típusa text. A profilPic mező a felhasználó profilhoz kapcsolt kép , nem regisztráláskor adják meg , típusa blob. Az admin mező regisztráláskor alapértelmezett értéke nulla , ez a mező határozza meg hogy az adott felhasználó admin e vagy sem , az asztali alkalmazásból módosítható az értéke , típusa tinyint. A registerDATE a regisztrálás dátumát rögzíti , tipusa DateTime.

* + 1. A Market tábla  
       

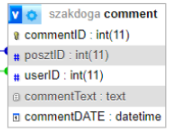
A market tábla a piactér adatainak tárolására szolgál.

A marketID mező az adott poszt egyedi azonosítója , automatikusan generálódik, Int típusú és elsődleges kulcs. A userID mező a felhasználó egyedi azonosítója , automatikusan kerül megadásra.  Típusa Int és indexelt. A title a poszt címe , text típusú ,regisztráláskor kerül megadásra. A marketpost a poszthoz kapcsolt szöveg , text típusú ,regisztráláskor kerül megadásra. A price a meghirdetett tárgy ára , Int típusú , regisztráláskor kerül megadásra. 11-es karakter limittel rendelkezik. A marketDATE a regisztrálás dátumát rögzíti , tipusa DateTime.

* + 1. Forum tábla  
         
       A forum tábla a forum adatainak tárolására szolgál.

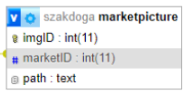
A posztID mező az adott poszt egyedi azonosítója , automatikusan generálódik, Int típusú és elsődleges kulcs. A userID mező a felhasználó egyedi azonosítója , automatikusan kerül megadásra. Típusa Int és indexelt. A title mező a poszt címe , text típusú ,regisztráláskor kerül megadásra. A post mező a poszthoz kapcsolt szöveg , mediumtext típusú ,regisztráláskor kerül megadásra. A hashtag mező különböző TAG-ek hozzáadását teszi lehetővé, text típusú. A postDATE a regisztrálás dátumát rögzíti , tipusa DateTime.

* + 1. Comment tábla



A comment tábla a comment atainak tárolására szolgál.

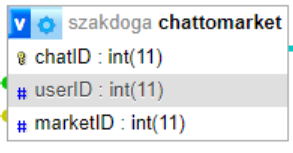
A commentID mező a comment egyedi azonosítója , automatikusan generálódik, Int típusú és elsődleges kulcs. A posztID mező az adott poszt egyedi azonosítója , automatikusan generálódik, Int típusú és indexelt. A userID mező a felhasználó egyedi azonosítója , automatikusan kerül megadásra . Típusa Int és indexelt. A commentText mező a commenthez csatolt szöveg , text típusú ,regisztráláskor kerül megadásra. A commentDATE a regisztrálás dátumát rögzíti , tipusa DateTime.

* + 1. MarketPicture tábla  
       

A marketpicture tábla a piactérhez feltöltött képek  adatainak tárolására szolgál.

A imgID a feltöltött kép egyedi azonosítója, automatikusan generálódik, Int típusú és elsődleges kulcs. A marketID mező az adott poszt egyedi azonosítója, automatikusan generálódik, Int típusú és indexelt. Kapcsolatban áll a Market táblával. A path a feltöltött kép nevét tárolja el, a program generál egy 10-15 karakter hosszúságú egyedi nevet és ez továbbításra kerül az adatbázis felé. Text típusú.

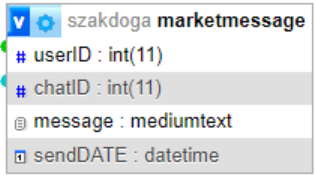
* + 1. ChatToMarket tábla



A chattomarket tábla a piactéren található hirdetésekhez tartozó privát beszélgetések adatait tárolja.

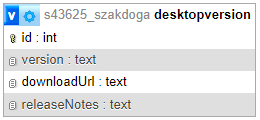
A chatID mező a privát chat azonosítója, Típusa Int. A userID mező a felhasználó egyedi azonosítója , automatikusan kerül megadásra. Típusa Int és idegen kulcs. A marketID mező az adott poszt egyedi azonosítója , Típusa Int és idegen kulcs.

* + 1. MarketMessage tábla



A MarketMessage az üzenetek tartalmát és adatait tárolja.

A userID mező a felhasználó egyedi azonosítója , automatikusan kerül megadásra. Típusa Int és idegen kulcs. A chatID mező a privát chat azonosítója. Típusa Int. Kapcsolatban áll a chattomarket táblával. A message mező magát az üzenetet tárolja , típusa mediumtext. A commentDATE a regisztrálás dátumát rögzíti , típusa DateTime.

* + 1. DesktopVersion tábla

A DesktopVersion tábla a telepítő adatait kezeli.

Az ID mező a program működését és az adatok elrendezését segíti, típusa int. A version mező a verzió számot tárolja (pl.:\_1.2.4.0. ) , típusa text. A downloadUrl tárolja a fájlnevet amivel a program dolgozik, típusa text. A releaseNotes-ban a fejlesztők által megírt változások dokumentálása van eltárolva, típusa text.

1. Fejlesztői dokumentáció

WEBOLDAL

* 1. Index.js
     1. Az index.js fájl a webáruház backend részének a főbeállító fájlja. Ez a fájl importál különböző modulokat, beállítja a middleware-eket, definiálja az útvonalakat és a logikájukat a HTTP metódusokhoz (GET, POST, PUT, DELETE).
     2. A kód egyik legkiemelkedőbb eleme a github bejelentkezés.

Ez a kód a Passport modult használja a GitHub-on keresztüli felhasználói bejelentkezéshez. A felhasználó a GitHub bejelentkezési gombra kattintva indíthatja el a folyamatot. A GitHub átirányítja a felhasználót a weboldalra, ahol a szerver a code paraméterrel lekéri a profilját. A szerver ellenőrzi, hogy a felhasználó már szerepel-e az adatbázisban. Ha igen, bejelentkezteti, ha nem, új bejegyzést hoz létre a profil adataival.

* + 1. Az index.js másik kiemelkedő része a kereső mező.

A Piactéren belüli keresési funkció (/searchmar) a megadott kifejezés alapján szűr termékeket. A keresés a title, marketpost és username mezőkben történik LIKE operátorral. A lekérdezés egyszerre lekéri a termék adatait, a tulajdonos felhasználónevét és a termékhez tartozó első képet.

* 1. Login.js
     1. A login.js fájl felelős a felhasználói interakciók kezeléséért, az űrlapok adatkezeléséért, a felhasználók bejelentkezési folyamatának és regisztrációjának kezeléséért, valamint a GitHub bejelentkezési lehetőség biztosításáért.
     2. Bejelentkezés

A kód kiválasztja a login-form űrlapot. Az adatok kitöltése után az űrlap elküldésre kerül a szervernek. Ezután a kód lekérdezi a bejelentkezési űrlapon megadott adatokat: a felhasználónevet az #loginusername azonosítóval rendelkező input mezőből, a jelszót az #loginpassword azonosítóval rendelkező input mezőből, és a "Bejelentkezve tartás" jelölőnégyzet állapotát az #rememberme azonosítóval rendelkező jelölőnégyzetből.A beírt adatok biztonságossá tétele érdekében a kód a setToSafe nevű függvényt használja.Ha a bejelentkezés sikeres, a kód átirányítja a felhasználót a /home oldalra.

* + 1. Regisztráció

A felhasználó először kitölti a regisztrációs űrlapot, megadva a felhasználónevét, e-mail címét és jelszavát. Ha a "Regisztráció" gombra kattint, az űrlap elküldésre kerül a szervernek.Ezután a kód lekérdezi a regisztrációs űrlapon megadott adatokat: a felhasználónevet az #regusername azonosítóval rendelkező input mezőből, az e-mail címet az #regemail azonosítóval rendelkező input mezőből, és a jelszót az #regpassword azonosítóval rendelkező input mezőből.Ha a regisztráció sikeres, a kód visszaigazolást jelenít meg a felhasználónak.

* 1. Home.js
     1. A home.js fájl felelős a felhasználói főoldal működéséért és a posztok kezeléséért. A kód elején több változót definiál, melyek a bejelentkezett felhasználó adatait fogják tárolni (id, név, email, regisztráció dátuma).
     2. Posztok betöltése

A posztok betöltése a következőképp működik.   
A ” window.addEventListener("load", (event) => { ... })” rész a weboldal betöltődése után fut le. A getpost függvény a szervertől a /getpost címen kéri le a bejegyzéseket. A válaszban kapott JSON adatokat feldolgozza, és HTML kóddá alakítja át. Minden bejegyzéshez létrehoz egy linket a cím alapján, mely a bejegyzés oldalára (/post/[id]) navigál. A megjelenített információk a bejegyzés címe, rövidített tartalma, posztoló neve és a létrehozás dátuma.

* + 1. Posztok létrehozása

A weboldal egy modal ablakot (felugró ablakot) használ új bejegyzések létrehozásához. Ezen az ablakon egy űrlap található, ahol a felhasználó megadhatja a bejegyzés címét és tartalmát. A modal ablak betöltésekor a kód lekéri a gombokat és mezőket azonosító változókba. Az "openModalButton" gombra kattintva megjelenik a modal ablak, a "cancelButton" gombra kattintva pedig bezáródik. Az űrlap beküldésekor a kód először megakadályozza az alapértelmezett form submit eseményt, hogy teljes mértékben irányíthassa a folyamatot. Ezután lekéri a megadott címet és tartalmat a mezőkből. A kód elküldi a bejegyzés adatait a szervernek a sendpost függvény segítségével. Sikeres küldés esetén a kód megjelenít egy üzenetet, hogy a bejegyzés létrehozása sikeres volt, majd automatikusan a kezdőoldalra navigál a felhasználót.

* 1. Post.js
     1. A post.js fájl a home.js által megjelenített bejegyzések kezeléséért felelős.  A fájl több funkciót tartalmaz, melyek a bejegyzések betöltését, megjelenítését, frissítését, törlését és a hozzájuk tartozó kommenteket kezelik.
     2. Aktuális bejegyzés betöltése és megjelenítése.

A setCurrentUser függvény meghatározza az aktuálisan megjelenített bejegyzés ID-jét (currentPosztID) a modifyBTN gomb data-posztid attribútumából, valamint a bejegyzést létrehozó felhasználó ID-jét (userPosztID) ugyanannak a gombnak a data-posztuserid attribútumából. Ezzel a két információval a fájl lekéri a bejegyzés adatait a szervertől, és megjeleníti a felhasználó számára a bejegyzés fejlécét (cím, felhasználónév, dátum), a bejegyzés tartalmát, a hozzá tartozó képeket, és a hozzászólásokat.

A setCurrentUser függvényben a fájl ellenőrzi, hogy a bejelentkezett felhasználó rendelkezik-e adminisztrátori jogosultsággal. Ha igen, az isAdmin változó értéke true lesz. Ez a jogosultság befolyásolja a későbbi funkciók elérhetőségét (pl. bejegyzés szerkesztése, törlése).

* 1. Market.js
     1. A market.js a weboldalon található piactér működését kezeli. A kód a felhasználói munkamenetek kezelését, a piactéren hirdetett termékek megjelenítését és a felhasználói profilok kezelését végzi.
     2. Bejegyzések lekérdezése

A getpost függvény lekéri a piactéren lévő bejegyzéseket a szervertől a fetch('/get-market-ad') paranccsal. A bejegyzések lekérdezése után felépíti a bejegyzések megjelenítéséhez szükséges HTML-struktúrát. A HTML a következőket tartalmazza cím, tartalom, felhasználónév, ár és dátum.

* 1. MarketPost.js
     1. A marketpost.js fájl a market.js által megjelenített bejegyzések kezeléséért felelős.
     2. Képek betöltése

Először lekéri az aktuális bejegyzés azonosítóját (currentPosztID) a "modifyBTN" azonosítójú elem dataset.posztid tulajdonságának használatával. Ez feltételezi, hogy az elemnek van egy "posztid" nevű adat attribútuma, amely tartalmazza a bejegyzés azonosítóját.  
A függvény letölti a képinformációkat egy adott bejegyzéshez, és ezen információk alapján dinamikusan hozzáadja vagy frissíti a karusszel tartalmát a releváns képekkel.

* + 1. Chat megnyitása

A sendchat gomb kattintására aktiválódó eseménykezelő. Az eseménykezelő a click eseményre van csatolva a "sendchat" azonosítójú elemhez.  
A kód ezután ellenőrzi a chatid változó értékét. Ha a chatid nagyobb, mint 0, ez azt jelenti, hogy már létezik chat a jelenlegi bejegyzéshez, és a felhasználót átirányítja a chat oldalra. Ha a chatid 0-val egyenlő, ez azt jelenti, hogy új chatet kell létrehozni a jelenlegi bejegyzéshez.

* 1. Chat.js
     1. A Chat.js felelős a piactér bejegyzéseihez kapcsolt felhasználók közötti beszélgetés kezeléséért.
     2. A Chat.js működése

A kód a bejelentkezett felhasználó csevegési élményét kezeli. Kezdetben inicializálja a szükséges változókat, mint a bejelentkezett felhasználó azonosítója, neve, e-mail címe, regisztrációs dátuma és az aktuális csevegés azonosítója. Ezt követi a munkamenet adatainak lekérdezése a szervertől, ahol ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha igen, lekéri a csevegő ikonokat a kiválasztott mód (vásárlás vagy eladás) alapján, és frissíti a csevegést automatikusan 2 másodpercenként. A frissítés során lekéri az üzeneteket az aktuális csevegésből, HTML elemeket hoz létre az üzenetekhez, és hozzáfűzi azokat a csevegőablakhoz.

* 1. Az update.js működése
     1. A főoldalról elérhető profil menüben lévő verziók linkre kattintva átjutunk az admin alkalmazás verzió kezelőjére. Itt elérhető az összes verzió és köztük a legfrissebb is. Az asztali alkalmazás az ehhez köthető backend részben innen szolgálja ki hogy mindig a legfrissebb verzión tudjon futni és a esetleges patcheket könnyen el tudjuk juttatni a felhasználóknak. Az oldal eléréséhez admin jogosultság szükséges

1. Fejlesztői dokumentáció

ASZTALI ALKALMAZÁS

* 1. AfterLogin

A Backend mappában található AfterLogin almappa két fájlt tartalmaz a LoginResponseClass-t és a LoginResponse-t

* + 1. LoginResponseClass

Ez az osztály tárolja a bejelentkezési válaszban kapott információkat. A következő adatokat tárolja:   
”hitelesitve” ez tárolja az admin jogosultság állapotát   
”userID” ez a felhasználó egyedi azonosítóját tárolja  
”userName” ez a felhasználónevet tárolva  
”Email” ez a felhasználó email címét tárolja

A kód ezután az kinyert adatokat egy tömbben helyezi el ahol minden adat típus előre megadott helyet kap.

* + 1. LoginResponse

A LoginResponse osztály egy struktúrát biztosít a bejelentkezési válaszok tárolására és kezelésére, lehetővé téve a LoginResponseClass objektumok hatékony kezelését.

* 1. Osztályok
     1. A GetAdminClass mappa, amely 3 különböző fájlt tartalmaz.   
         A három kód egymáshoz kapcsolódó osztályokat definiál, amelyek adminisztrátorok adatainak tárolására és kezelésére szolgálnak. Az AdminDatas osztály az adminisztrátorok alapvető adatait tárolja, a GetAdmin osztály további információkat tárol a regisztrációról és az adminisztrátori jogosultságokról, az AdminResponse osztály pedig egy listát tárol a GetAdmin objektumokból, lehetővé téve több adminisztrátor adatainak kezelését.
     2. A GetUserClass mappa szintén 3 különböző fájlt tartalmaz. Ezek a kódok osztályokat definiálnak a felhasználók adatainak reprezentálására és továbbítására . A UserDatas osztályt a felhasználó adatainak tárolására használják, a GetUser osztályt a felhasználók adatainak reprezentálására a rendszerben, a UserResponse osztályt pedig a felhasználók adatait tartalmazó JSON válasz reprezentálására.
     3. A GetPostClass mappa szintén 3 különböző fájlt tartalmaz. Ezek a kódok osztályokat definiálnak a posztok adatainak reprezentálására és továbbítására Az első kód (PostDatas) egyetlen bejegyzést reprezentál tulajdonságokkal, a második kód (GetPosts) szintén egy bejegyzést definiál, de tartalmaz egy konstruktort a létrehozás dátuma/időpontja kezelésére. A harmadik kód (PostResponse) pedig egy listát tárol az objektumokról.

A GetMarketPostClass mappában található fájlok ugyan azon az elven működnek mint a GetPostClass-ban találhatóak. Egyetlen eltérés hogy a GetMarketPostClass a piactéri posztok adatait kezeli.

* 1. baseURL
     1. A kód egy baseURL nevű osztályt definiál, amely kezeli az alkalmazás globális URL-jét. A Singleton tervezési mintát használva. Elsőnek megalkottunk egy privát statikus változót az osztály egyetlen példányának tárolására. A privát konstruktor megakadályozza a közvetlen példányosítást az osztályon kívülről. Aztán a kód hozzáférést biztosít az egyetlen példányhoz.     
        “GlobalURLString = ConfigManager.GetConfigValue("baseURL")”   
        Ez a sor a konfiguráció kezelőtől a "baseURL" kulcs használatával lekéri az alap URL értékét, és hozzárendeli a “GlobalURLString” tulajdonsághoz.
  2. ConfigManager
     1. Ez a kód egy ConfigManager nevű statikus osztályt definiál, amely kezeli a konfigurációs beállítások olvasását és írását egy JSON fájlban.  
        “appDataRoamingPath” - Az “Environment.GetFolderPath” segítségével lekéri az ApplicationData mappájának elérési útvonalát.  
        “byteBagDirectoryPath” - Az “appDataRoamingPath”-t összevonja a "ByteBag" könyvtárnevével.  
        “filePath” - A “byteBagDirectoryPath”-t összevonja a "ByteBagConfig.json" fájlnevével. Ezek az elérési útvonalak határozzák meg a konfigurációs fájl tárolási helyét.
     2. A SetConfigValue(string key, string value) metódus egy adott kulcs ("key") értékét állítja be a konfigurációs fájlban. A metódus a következő lépéseket követi:  
        -Ellenőrzi a fájl létezését: Először ellenőrzi, hogy a konfigurációs fájl ("ByteBagConfig.json") létezik-e a megadott elérési útvonalon. Ha nem, akkor a ” CreateDefaultConfig()” metódust hívja meg a fájl létrehozásához és az alapértelmezett beállítások beállításához.  
        -Beolvassa a konfigurációt: Ha a fájl létezik, a ”JsonConvert.DeserializeObject” metódust használja a fájl tartalmának JSON formátumból szótárrá konvertálására. Ez a szótár a konfigurációs beállításokat kulcs-érték párok formájában tárolja.  
        -Frissíti a kulcs értékét: A metódus megkeresi a megadott kulcsot ("key") a szótárban. Ha a kulcs megtalálható, frissíti az értékét a megadott "value" értékkel. Ha a kulcs nem létezik, új kulcs-érték párt hoz létre a szótárban.  
        -Visszaserializálja a konfigurációt: A frissített szótárt a ”JsonConvert.SerializeObject” metódussal JSON formátumba konvertálja.  
        -Visszaírja a fájlba: A JSON tartalmat az ” File.WriteAllText” metódussal visszaírja a konfigurációs fájlba.
     3. “CreateDefaultConfig()” - Ez a metódus létrehoz egy alapértelmezett konfigurációs fájlt, ha az nem létezik. A “Directory.CreateDirectory” használatával létrehozza a “byteBagDirectoryPath” könyvtárat. Meghatároz egy alapértelmezett szótárt két kulcs-érték párral a lehetséges alap URL-ekhez. A “JsonConvert.SerializeObject” metódust használja az alapértelmezett szótár JSON formátumba konvertálására.
     4. A GetConfigValue(string key) metódus egy adott kulcs ("key") értékét lekéri a konfigurációs fájlban. Először ellenőrzi a fájl létezését, majd beolvassa és deserializálja a konfigurációt. Ha a kulcs megtalálható, a metódus visszaadja az értékét.
  3. Internet Connection Check
     1. Ez a kód egy egyszerű módszert biztosít az internetkapcsolat ellenőrzésére. A google.com webhely pingelésével ellenőrzi, hogy a számítógép képes-e kommunikálni a külső világgal.
     2. A “System.Net.NetworkInformation”  könyvtár hálózati kommunikációhoz szükséges osztályokat tartalmaz, mint például a Ping osztályt. A kód a google.com webhely pingelésével ellenőrzi az internetkapcsolatot. Sikeres ping esetén true értékkel jelzi az internet elérhetőségét, hiba esetén pedig false értékkel a kapcsolat hiányát.
  4. LoginWindow
     1. Ez a kód egy bejelentkezési felületet valósít meg egy szoftverhez. A felhasználó beírhatja a felhasználónevét és jelszavát, majd a program ellenőrzi a szerveren a hitelesítést. Ha a bejelentkezés sikeres, a program megnyitja a főablakot.
     2. LoginMethod()

Ez a metódus felelős a bejelentkezési folyamatért. Elküldi a felhasználónevet és jelszót a szervernek, és ellenőrzi a választ. Ha a bejelentkezés sikeres, a metódus true értékkel tér vissza, és megnyitja a főablakot. Ellenkező esetben false értékkel tér vissza, és megjelenít egy hibaüzenetet.

Felhasználói adatok lekérése:  
A metódus először lekéri a felhasználónevet a usernameTB Textbox mezőből, és tárolja a username változóban.Ezután lekéri a jelszót a passwordPB PasswordBox mezőből, és tárolja a password változóban.

Mezők ellenőrzése:  
A metódus ellenőrzi, hogy a username és a password változók értéke üres-e. Ha bármelyik változó értéke üres, a metódus létrehoz egy ErrorMessageWindow ablakot, és a következő üzenetet jeleníti meg a "LabelContent" tulajdonságban: "Minden mezőt ki kell tölteni!".

Bejelentkezési kérelem küldése:  
Ha a username és a password változók nem üresen, a metódus POST kérést küld a szervernek a NetworkHelper.Backend.POST() metódus használatával.  
A kérés URL-je a baseURL.Instance.GlobalURLString értékéhez concatenálva a "/wpflogin" utat tartalmazza.  
A kérés testében a loginusername és a loginpassword kulcsokhoz rendelve megadja a username és a password változók értékét.  
A metódus aszinkron módon várja a szerver válaszát a Task.Run() metódus használatával.

Sikeres bejelentkezés:  
Ha a responseData változó értéke "hitelesitve" szöveggel kezdődik, a metódus a következőket teszi:  
Létrehoz egy MainAdmin objektumot a logindatas lista átadásával. A logindatas lista a bejelentkezési adatokat tartalmazza.  
Bezárja a bejelentkezési ablakot a this.Close() metódus használatával.  
Megnyitja a főablakot a mainWindow.Show() metódus használatával.

* 1. Az asztali alkalmazás telepítője.

Az .msi telepítő, Microsoft szerverről teszi lehetővé a futtatáshoz szükséges .NET Framework illetve .NET Core eszközöket, futtató környezeteket. A telepítők „Microsoft Visual Studio Installer 2022”-ben készültek.

A „File System”-ben létrehoztunk egy OutPut-ot a program futtatásához szükséges parancsikonok létrehozása végett, melyeket a(z) „User’s Desktop” illetve a „User’s Programs Menu” mappákban helyeztünk el. Az előbbi mappa a felhasználó asztalára, másik pedig a Start Menübe helyezi el a parancsikonokat.

Az „Application Folder” a felhasználó által meghatározott program telepítési mappájának felépítését reprezentálja. A mappában találhatóak a futtatáshoz szükséges „Assembly”-k ún. „dll”-ek. A dll-ek alapértelmezetten „Hidden” tulajdonságuk „True” értékkel rendelkeznek. Ebben a mappában található még egy „Licence.txt” fájl, mely a lincenszet tartalmazza. Itt található továbbá az ikon és telepítő banner képe is. Minden tulajdonság a felhasználók számára biztosítja a könnyű mappaszerkezet átláthatóságot, valamint a könnyű kezelést. Az ikonok megjelenése mindenhol a programunk ikonja. Mind a vezérlőpultban, (Windows 10 esetén gépházban), mind az asztalon-tálcán stb.

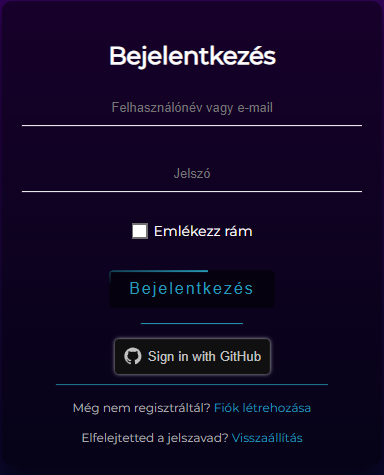
A felhasználók számára .msi formátumot tettük lehetővé a könnyebb telepítési folyamat végett. Verziószámokat és egyéb információkat folyamatosan frissítettük, viszont a produktID-t meghagytuk egy ponton, az előző verzió sikeres detektálása végett. „SupportUrl” értékként beállítottuk a készítő Github url-jét kapcsolatfelvétel céljából. A „TargetPlatform” pedig a x64 es processzorok támogatása lett beállítva.

* 1. Az asztali alkalmazás frissítése
     1. A fejlesztők számára egy egyszerű rendszert próbáltunk felépíteni. Az alkalmazás mappaszerkezetében az osztályt és a backend felőli json értelmezésére szolgáló osztályt szeparálva találhatjuk meg a Backend/UpdateChecker mappán belül.  
          
        Az UpdateInfo osztály előre fel lett készítve esetleges json fájl szerkezet változásaira.

Az „id” a szerveren található frissítés sorszámát takarja.  
A „version” tulajdonság a verziószámot takarja, melyet a szervertől kap meg „1.0.0.0” formátumban.  
A „downloadUrl”-t szintén a szervertől kapott információ, melyet tulajdonságként kezelünk. Ez tartalmazza az útvonalat a fájlhoz. Pl.: https://bytebag.hu/update/ ---> „downloadUrl”  
A „releaseNote” az update változásainak leírására szolgál.

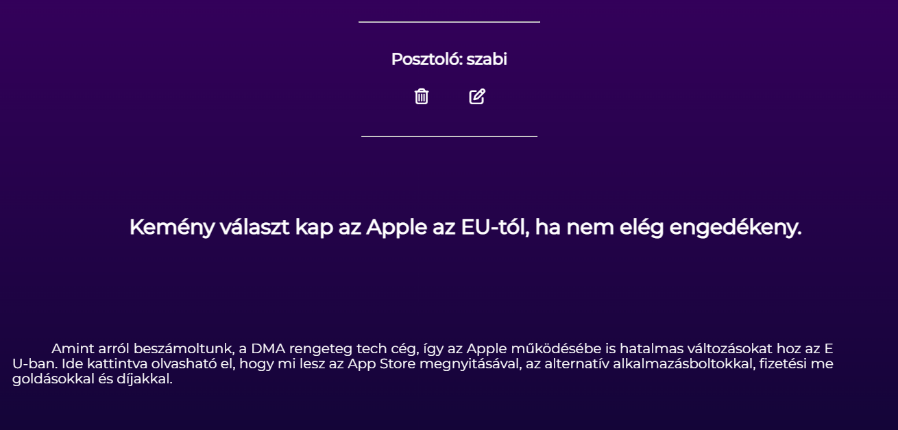
* + 1. A „checkUpdate” osztály a fő mechanizmusa a frissítési rendszerünknek. Itt található több függvény is. A „CheckForUpdatesAsync” függvény ellenőrzi a frissítés aktualitását. Egy szerverkérésen keresztül lekérdezi a „/update/latest” útvonalon az utolsó elérhető frissítést, melyben a szervertől GET metódus segítségével egy verziószámot, letöltési URL-t és egy frissítési megjegyzést kap vissza. Következőekben pedig ezek az infók listába kerülnek, mely „UpdateInfo” típusú lesz.
    2. Amennyiben a szerver verzió nem egyezik a telepített alkalmazás verziójával, abban az esetben meghívódik a „DownloadAndUpdate” függvény a „downloadUrl” paraméterrel, melyet előzőleg a szervertől kaptunk és tároltunk.  
       Függvény meghívás után egy ideiglenes fájl jön létre egy mappával a felhasználó ideiglenes mappájában, mely helyre letöltést indít a szerverről a program, majd amint a letöltés befejeződött, megnyitja a telepítőt és a jelenlegi alkalmazást bezárja, így elkerülve a hibákat.

1. Felhasználói dokumentáció
   1. weblap

Az első funkció amivel találkozunk az oldal betöltésekor a bejelentkező panel, itt lehetőségünk van bejelentkezni saját fiókunkba, új fiókot regisztrálni vagy visszaállítani elfelejtett jelszavak.   
Új fiók regisztrálásakor meg kell adni egy felhasználónevet , egy érvényes email címet illetve a jelszót.  
A bejelentkező panelnél még lehetőségünk adódik visszaállítani elfeledett jelszavunk az email cím megadásával.   
Továbbá lehetőségünk van GitHub fiókkal is bejelentkezni   


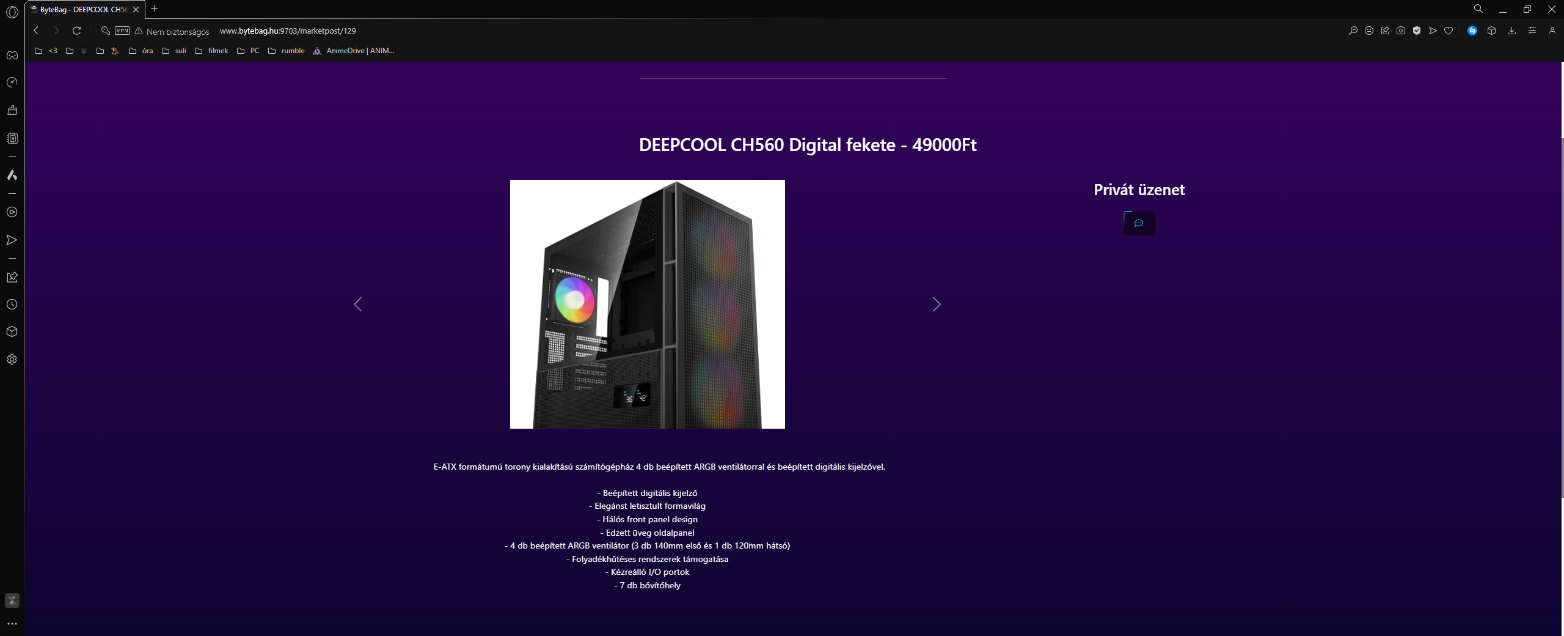
Sikeres bejelentkezés után a Fórumot látjuk magunk előtt , itt találjuk a bejegyzéseket . A címsoron érhetjük el a kereső mezőt , a fórum és a piactér főoldalára irányító két gombot, a piactér privát chat oldalát, a saját fiókunkat és a kijelentkező gombot. Az oldal alján találjuk meg azt a gombot amivel új bejegyzéseket hozhatunk létre.



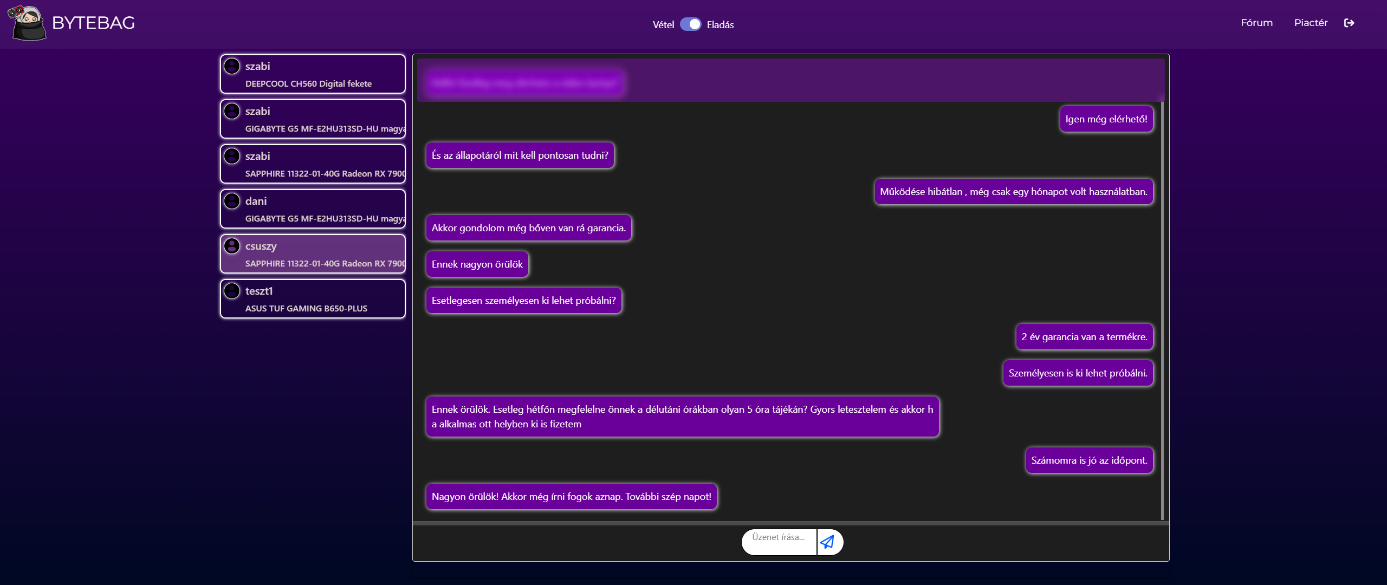
Egy adott posztra kattintva elolvashatjuk a teljes bejegyzést , és kommenteket fűzhetünk hozzá. A bejegyzés megírójának vagy admin jogosultsággal rendelkezőknek lehetősége van törölni vagy szerkeszteni a posztot. A törlés gombra kattintva egy megerősítő kérdés ugrik fel , ezzel elkerülve azt hogy véletlen töröljünk egy bejegyzést , mivel a folyamat nem visszafordítható. A szerkesztés gombra kattintva lehetőségünk adódik átírni a bejegyzés címét és tartalmát.

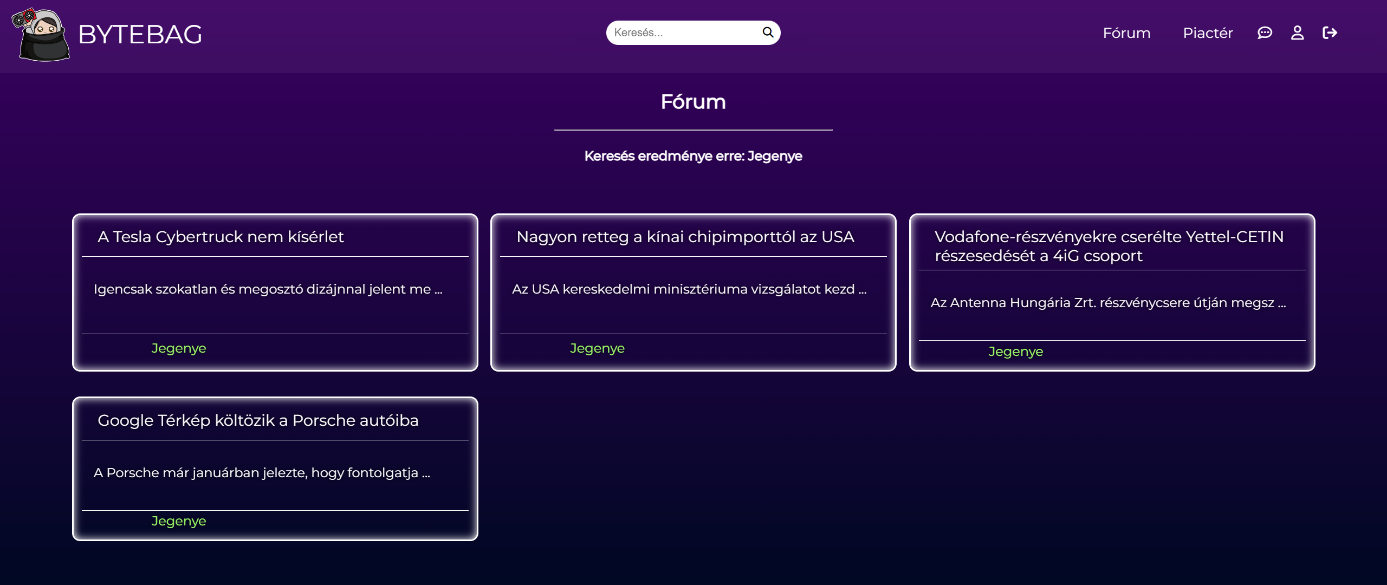


A címsoron található következő fő gombbal átléphetünk a piactérre , ahol láthatóak a hirdetések és egy gomb ami új hirdetések feladására szolgál, továbbá a már említett címsor.

Egy adott hirdetésre kattintva láthatjuk a termék címét, árát, leírását és a hozzá csatolt képeket. És privát üzenetváltásba kezdhetünk a hirdetővel. A hirdetés megírójának vagy admin jogosultsággal rendelkezőknek lehetősége van törölni vagy szerkeszteni a bejegyzést. A hirdetés címe fölött látható két gomb, a törlés és a módosítás. Ezt a két gombot azok használhatják, akik feladták a hirdetést vagy amin jogosultsággal rendelkeznek. A ”Privát üzenet” felirat alatt található gombbal vehetjük fel a kapcsolatot az eladóval , ez az üzenetváltást csak a vevő és a hirdető láthatja. A gombra kattintva a privát chat meg is nyitható, bár a chat a címsoról is elérhető.   
Ezen az oldalon a következők láthatók, a chat fölött egy kapcsoló, ennek a segítségével válthatunk vevő és eladó mód közt így könnyebben kezelhetjük az általunk megvásárolni illetve eladni szánt termékeket. A chat bal oldalán egy nevekből álló lista található, itt látszódnak azok a vevők és eladók, akikkel már üzenetváltást indítottunk. 

Ha kiválasztjuk az egyik kontaktok akkor láthatóvá válnak az esetleges eddigi üzenetek, és bármikor írhatunk új üzenetet is, az új üzenetet a chat alján látható fehér mezőbe kell beírni majd a mellette lévő gomb megnyomásával elküldhetjük. Az üzenet a másik felhasználó nézetéből már 2 másodperc alatt láthatóvá válik.

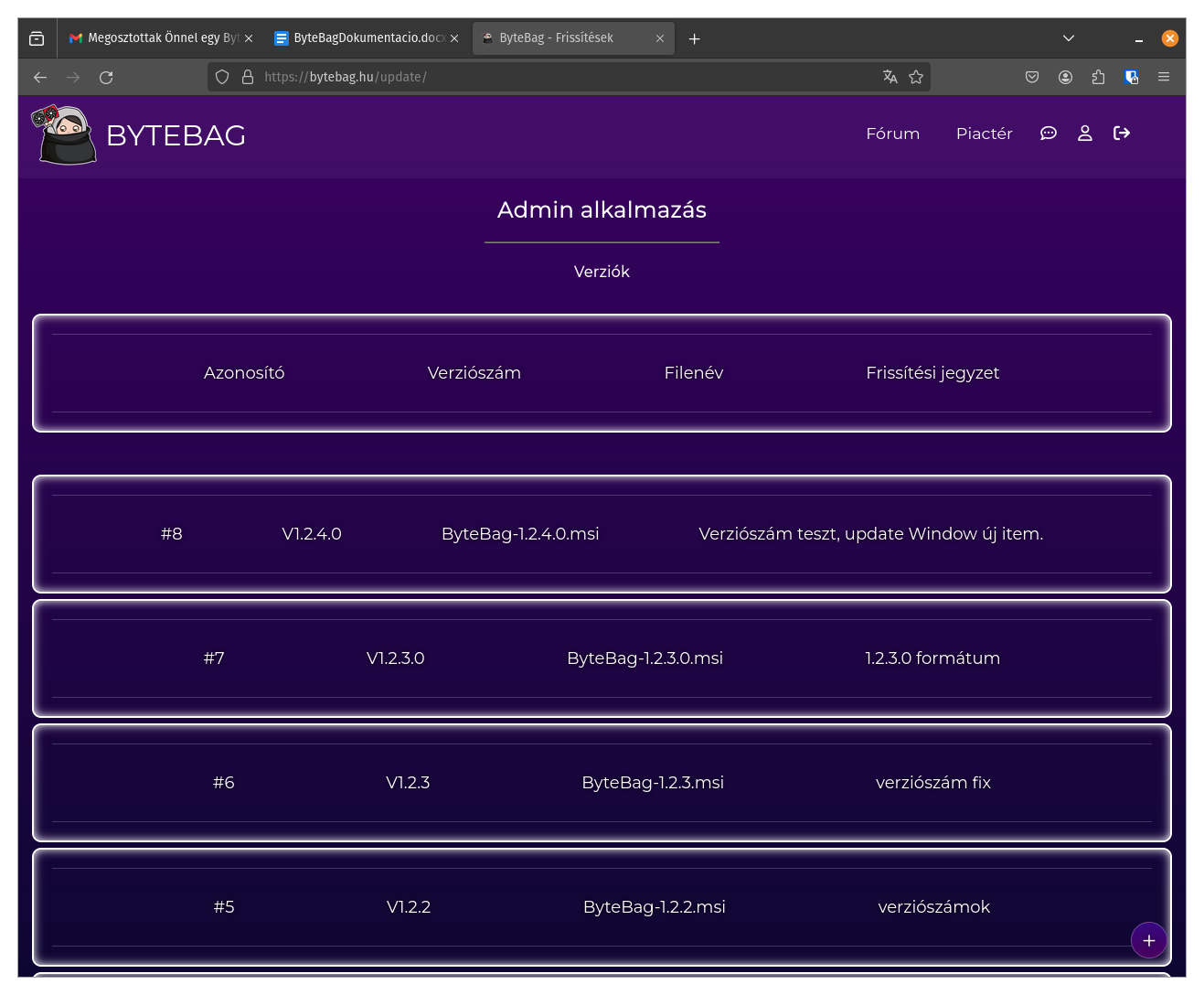


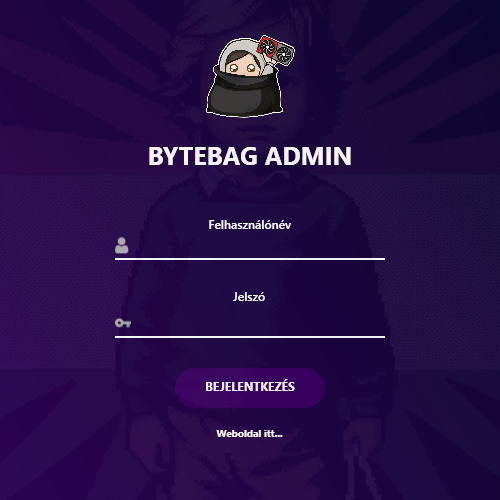
A következő funkció amit a fórum illetve a piactér címsoráról lehet elérni a kereső, itt az eddig posztolt bejegyzések illetve eladások közt lehet keresni. Felhasználónév , cím illetve tartalom alapján kereshetünk. Az oldal minden olyan találatot kifog nekünk adni amiben szerepel valamilyen formában a keresett szó miután rányomtunk a gombra. 

Az utolsó címsoron elhelyezett funkció a saját fiókunkhoz vezet, azt a karakter ikonnal ellátott gomb segítségével érhetjük el. Itt láthatjuk saját adatainkat. Lehetőségünk adódik a jelszó módosítására is, amennyiben megszeretnénk változtatni jelszavunk csak be kell írni a megfelelő mezőbe az új jelszót és a ”Kész” gombra rányomva el is fogadhatjuk a módosítást. Továbbá ha admin jogosultsággal rendelkező fiókkal lépünk be a ”Verziók” gomb segítségével megtalálhatunk egy telepítőt az asztali alkalmazáshoz.   
Asztali alkalmazásunk egyik fő eleme a verziókezelés. A verziókezelést szerveren keresztül kezeltük. A felhasználók számára semmi egyéb dolguk nincsen, csak használni az alkalmazást és amennyiben elérhető frissítés, követni a telepítő utasításait.

A képen szöveg, képernyőkép, multimédia, Mobiltelefon látható

Automatikusan generált leírás

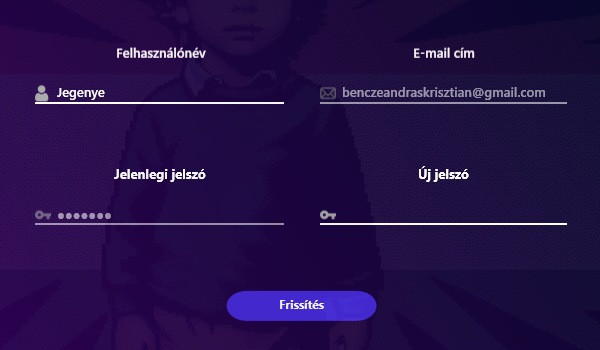


* 1. Admin applikáció-WPF
     1. Az asztali alkalmazás célja hogy a ByteBag adminisztrátorai egy külön felületen moderálhassák az oldalon található bejegyzéseket , eladásokat és felhasználókat .

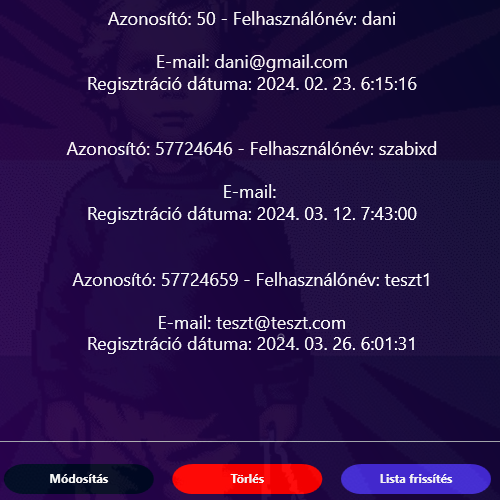
Az alkalmazás megnyitásakor az első funkció amivel találkozunk az a bejelentkezés.A weboldallal ellentétben itt már nincs lehetőségünk regisztrálni , és ide csak azok jelentkezhetnek be akik rendelkeznek admin jogosultsággal. A felhasználónév és jelszó beírása után beléphetünk az admin felületbe vagy átléphetünk a weboldalra .

Sikeres belépés után az oldal üdvözli és eligazítja

a felhasználókat.



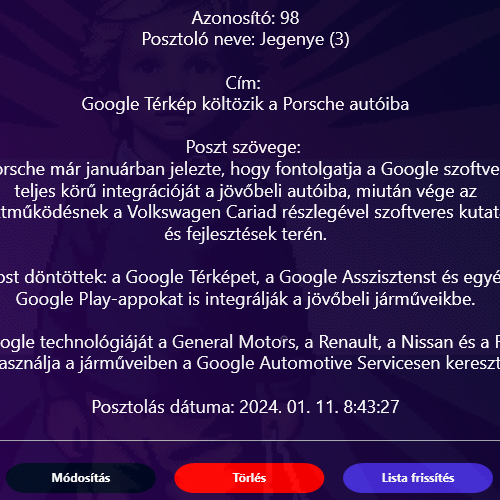
A főoldalon láthatjuk a fiókhoz tartozó felhasználónevet, email címet és a jelszót. Utóbbinak módosítására lehetőségünk is van.A megváltoztatott adatokat a frissítés gombbal tehetjük láthatóvá.

A következő fül alatt láthatjuk a felhasználó listáját , itt minden felhasználó megtalálható. 

Itt kerül megjelenítésre az egyedi azonosító, felhasználónév, email cím és a regisztráció dátuma.

Az adminoknak lehetőségük van módosítani a fiókok adatait , a felhasználók segítése érdekében.

Továbbá lehetőségük adódik véglegesen törölni az adott fiókokat, így akár szankcionálni a felhasználókat felszólítás után.A megváltoztatott adatokat a lista frissítése után láthatjuk.



A következő fül alatt a posztok listáját láthatjuk. Itt elérhető minden olyan bejegyzés ami az oldalon látható. Itt látható az adott posztok azonosítója , a posztoló felhasználóneve , a poszt címe és tartalma továbbá a poszt közzétételének dátuma .  Az adminoknak lehetőségük adódik szükség esetén módosítani egy adott poszt tartalmát vagy címét .

A posztokat szintén tudják törölni is. A módosítások a lista frissítésével láthatóak. A market posztok kezelése ugyanezen a módon történik .



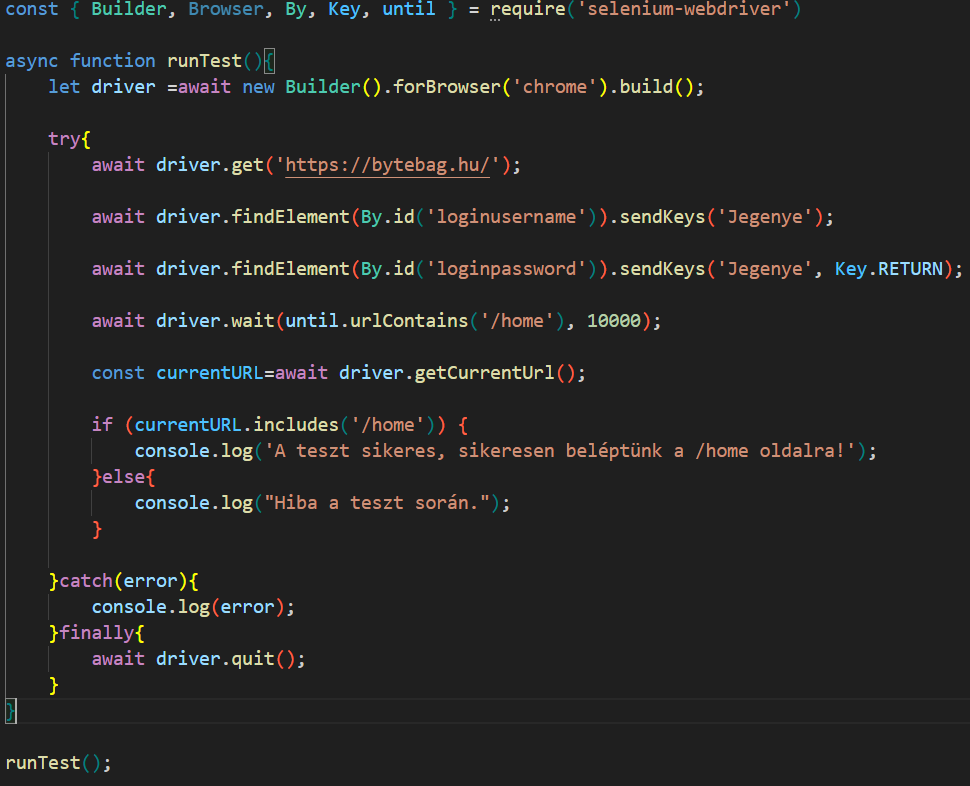
A következő fül alatt az adminok listáját találhatjuk, itt látható az oldal összes adminja . Kiolvasható innen az adminok azonosítója , felhasználóneve , email címe és regisztrálásának dátuma.

Az asztali alkalmazáshoz készült egy telepítő is, amely Windows rendszerekre, 64-bit-es processzorokra nyújt támogatást. A telepítővel garantáljuk a program hibátlan működését, későbbi új verziók támogatásával karöltve. A telepítő csak admin felhasználóknak elérhető, melyet a weboldalról tölthetnek le. A telepítő ajánlott és elérhető közvetlen változata .msi kiterjesztésű, amely könnyen kezelhető, és rendelkezik beépített funkciókkal, mint például hibakezelés és verziókezelés. Az .msi telepítők lényegében tartalmazzák az alkalmazás összes szükséges információját és parancsait, így egyszerűbb és megbízhatóbb telepítést tesznek lehetővé a felhasználók számára. Természetesen .exe telepítő is elérhető, mely hivatalos Microsoft szerverről teszi lehetővé a futtatáshoz szükséges .NET Framework illetve .NET Core eszközöket, futtató környezeteket.   
  
Az telepítő indulását követően az alkalmazás ellenőri, van-e korábbi verzió a rendszerünkön, amennyiben igen, automatikusan eltávolítja, erről értesítve a felhasználót. Amint ezzel végzett a programunk, be is tölt számunkra a telepítő első ablaka. A képen szöveg, elektronika, képernyőkép, képernyő látható

Automatikusan generált leírás

Tovább lépve kiválaszthatjuk a telepítési mappát, illetve, hogy mely felhasználók számára kívánjuk lehetővé tenni a program használatát. Ezek után nincs több dolgunk, mint jóváhagyni a telepítést, mely után már települ is programunk. Ha végzett a telepítés, már használhatjuk is a programunkat. A telepítő által létrehozott parancsikonnal az asztalon.A képen szöveg, elektronika, képernyőkép, képernyő látható

Automatikusan generált leírás

1. Tesztelés
   1. Selenium 

A weboldal automatizált teszteléséhez a selenium-webdriver

4.19-es verzióját használtuk ,ami több böngészőben is sikeresnek bizonyult.

* 1. Manuális tesztelés

A weboldal hosszas és részletes manuális tesztelésen ment keresztül, hogy a lehető legtöbb hibát ki tudjuk szűrni.   
A program teszteléséhez sok ember segítségét kértük így a visszajelzések és vélemények segítségével tudtuk kiszűrni azokat a problémákat melyekkel a felhasználók találkozhattak.

* 1. Automatizált asztali alkalmazás tesztA képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

     Automatikusan generált leírás
     1. Az automatizált tesztet, UnitTest-ben írtuk meg, azon belül is kettő külön teszttel, mely a „ConfigManager” osztály tesztjeit tartalmazza.
     2. Egyik ilyen teszt, a fájl létezésének vizsgálatára szolgál. Ez a teszt ellenőrzi, hogy a függvény helyesen le fog-e futni és nem-e fog hibára vezetni bármilyen esetben. Részletezve, a teszt megpróbálja lekérni az alapértelmezett URL-t a fájból. Amennyiben létezik, vissza adja az URL-t mely a „result” változóban található. Ha hibára fut, abban az esetben „Error” feliratot kapunk.A képen szöveg, képernyőkép, Betűtípus látható

        Automatikusan generált leírás
     3. A másik teszt a „ConfigManager” „SetConfigValue” függvény tesztjét tartalmazza, amely egy tesztben megfogalmazott kulcsal teszteli le a függvény futását és azt, hogy a visszatérési értéke egyenlő-e a tesztben megadottal. Tehát, részletezve. A program beállít egy kulcsot, melyet utána kikér a fájlból. Ha sikeresen lefut, vissza adja a URL-t amely a „value” változóban- és a „key” változóban -található. Amennyiben nem kapja meg ezt az értéket, hibával visszadob.A képen szöveg, képernyőkép, Betűtípus látható

        Automatikusan generált leírás