(Fedlap)

Storage Manager

Kirsch Ádám Péter – Sztrelcsik Zoltán

2021-2022

Tartalomjegyzék

[Témaválasztás 4](#_Toc99389837)

[Használt technológiák, környezetek, nyelvek 4](#_Toc99389838)

[WPF (asztali frontend) 4](#_Toc99389839)

[REACT (webes frontend) 18](#_Toc99389840)

[A frontend weboldal felépítése 20](#_Toc99389841)

[WCF (backend) 25](#_Toc99389842)

[A raktár programunk WCF rendszere 26](#_Toc99389843)

[További fejlesztési lehetőségek 27](#_Toc99389844)

[Összefoglalás 28](#_Toc99389845)

[Forrásjegyzék 28](#_Toc99389846)

### Témaválasztás

Vajon létezik-e olyan alkalmazás, amelyre világszerte széleskörűen szükség van, és ami nélkül a gazdaság működése minden bizonnyal összeomlana?

A gazdaság működésének alapját gazdasági szervezetek alkotják, legyenek azok nemzetközi multinacionális cégek, vagy kisebb helyi vállalkozások. A hatékony, fenntartható gazdálkodáshoz a termelő és kereskedelmi egységeknek minden esetben szükségük van egy központi adatbázisra, amely az erőforrásokat nyilvántartja, egy annak kezelésére hivatott alkalmazásra (backend), valamint egy felhasználói felületre (frontend), amely segítségével a munkatársak a teljes rendszert célirányosan és kényelmesen irányíthatják.

A szakdolgozatunk keretében egy ilyen raktárkezelő alkalmazást készítettünk, amely különböző termékek kezelését teszi lehetővé bármilyen távolságból, nemcsak helyhez kötött irodákból, hanem útközben, mobil eszközökön keresztül is.

### Használt technológiák, környezetek, nyelvek

### WPF (asztali frontend)

A vállalkozások irányítása jellemzően telephelyről, irodából történik, ezért magától értetődő igény, hogy egy raktárkezelő program asztali alkalmazásként is elérhető legyen. Az irodai munka napjainkban már szinte elképzelhetetlen egy nagyfelbontású monitor nélkül, amely akár több egymás melletti ablak használatával, azok átméretezésével hatékonyabb és kényelmesebb munkavégzést biztosít, mint a kisebb hordozható eszközök. Az asztali alkalmazások esetében kiemelt szempont a felhasználóbarát és rugalmas megjelenítés, ezért a választásunk az ezen a téren zászlóshajónak számító WPF (Windows Presentation Foundation) technológiára esett.

A WPF a Microsoft által fejlesztett .NET keretrendszer része, amely gazdag felhasználói felülettel rendelkező asztali alkalmazások létrehozására szolgál. A WPF egyik fő újítása a felület és az üzleti logika szétválasztása. Az **ablakok deklaratív kialakítása** a XAML (Extensible Application Markup Language) jelölőnyelvvel történik, a mögöttes kóddal pedig a megjelenítés dinamikusan változtatható. A megjelenés és vezérlés szétválasztása jelentősen javítja az alkalmazás felépítését.

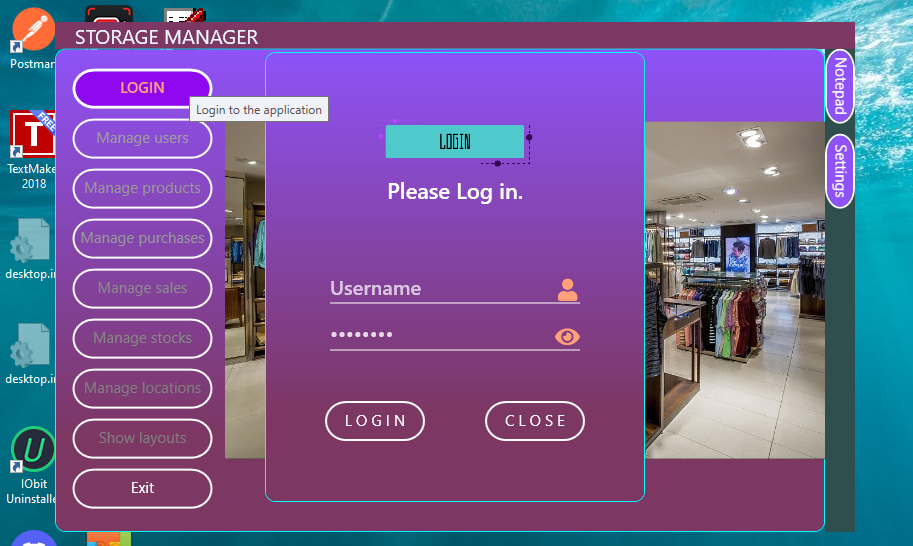
A WPF alkalmazások grafikai elemei már nem raszteresek, hanem **vektor alapúak** (felbontás-függetlenek), így az egyes elemek tetszőlegesen átméretezhetőek. A vektorgrafikus képek kevesebb helyet foglalnak a raszteres elemekhez képest, miközben a WPF továbbra is támogatja a raszter grafikát. A WPF a megjelenítéshez a GDI mellőzésével DirectX-et használ. A DirectX tehermentesíti a processzort, a videókártya (GPU) bevonásával gyorsabbá válik többek között az animációk lejátszása.

Az asztali alkalmazás indítását követően a felhasználót a kezdőképernyő fogadja, ahol a telephelyéről készült **képekből álló diavetítés** fogadja. A képek cserélhetőek és tetszőleges számban bővíthetőek, mivel az alkalmazás a képeket a forrásmappából indításkor kiolvassa.



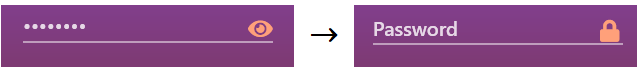
1. ábra: kezdőképernyő

Ekkor csak a LOGIN és az Exit gombok aktívak, és a kiválasztott gombnál eszközleírás jelenik meg. A LOGIN gombra kattintva megjelenik a **bejelentkezési ablak**:



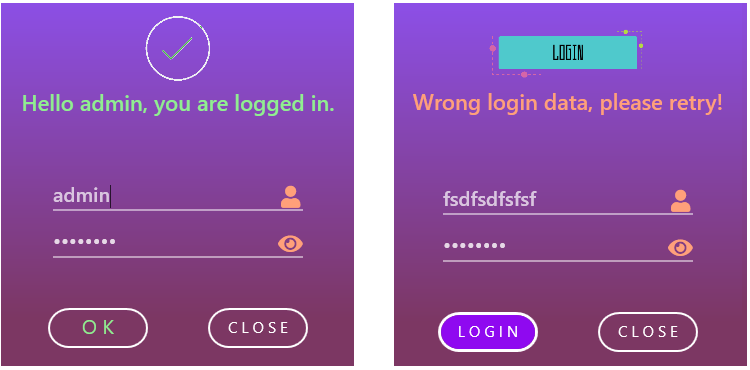
2. ábra: bejelentkezési ablak

A jelszó karakterei ki vannak takarva, de a mellette lévő szem-ikonra kattintva ez feloldható:

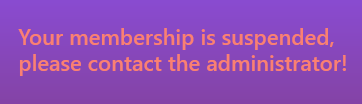


3. ábra: jelszó felfedési funkció

Az **adatbiztonság** egyik garanciája, hogy a program a jelszót azonnal titkosítja, és az már így kerül továbbításra. A bejelentkezést az is megkönnyíti, hogy nem szükséges a LOGIN gombra kattintani, mert az Enter billentyű megnyomása is a bejelentkezési folyamatot váltja ki. Sikeres bejelentkezést követően a program üdvözli a felhasználót, míg hibás adatok megadásakor erre figyelmezteti.



4. ábra: bejelentkezési folyamat

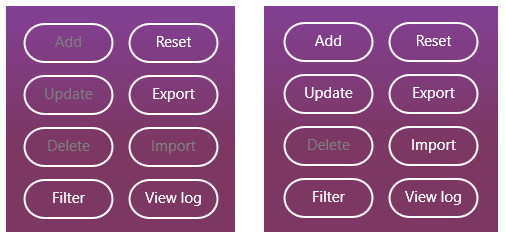


5. ábra: felfüggesztett felhasználó figyelmeztetése

Ha a felhasználó fiókja nem aktív (fel van függesztve), akkor a meghiúsult bejelentkezést követően egy erre vonatkozó hibaüzenet jelenik meg, és a felhasználó nem kap hozzáférést a program funkcióihoz:

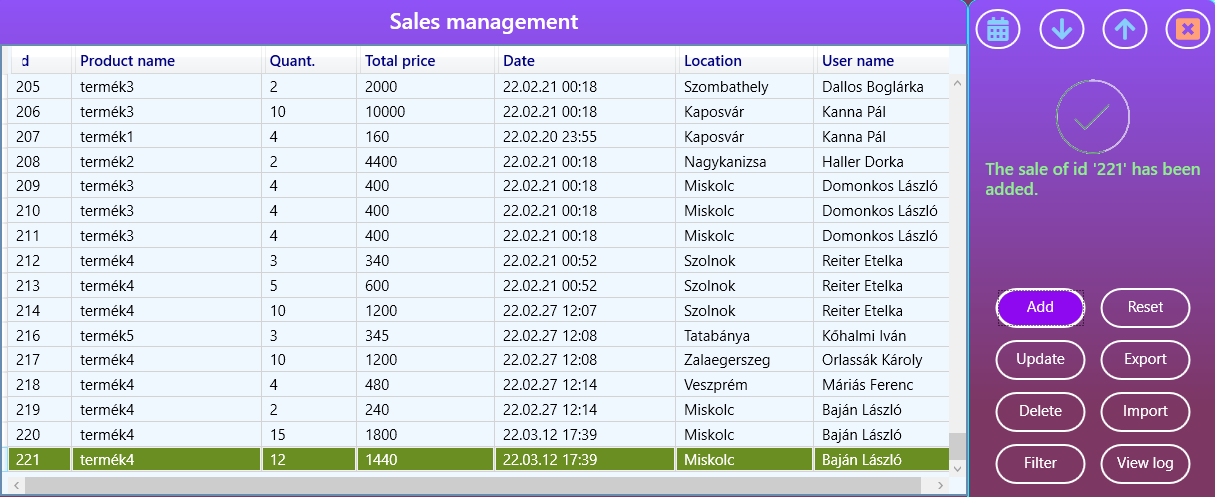
Az alkalmazás egy **integrált felhasználói és admin felületet** valósít meg azáltal, hogy a felhasználó jogainak függvényében enged hozzáférést a különböző funkciókhoz. A 0-2 jogosultsági csoportban kizárólag megtekintési joga van a felhasználónak. A 3-5 közötti sávhoz ezen kívül hozzáadási és frissítési jog tartozik, a 6-8 közötti tartományban pedig már törlési joggal is rendelkezik a felhasználó. A 9-es szint teljes adminisztrátori jogosultságot jelent, vagyis a felsoroltakon kívül még hozzáférhet a felhasználók és a telephelyek kezeléséhez. A **0-9 közötti jogosultsági tartomány** lehetővé teszi, hogy a fenti három besoroláson kívül további, kisebb tartományú csoportok létrehozásával még finomabb megkülönböztetést lehessen tenni az egyes jogosultságok között.

Az alábbi ábrán például az látszik, hogy a bal oldali esetben egy 0-2 jogosultságú felhasználó van bejelentkezve, és az adott tábla vonatkozásában csak olvasási joga van, a hozzáadással és frissítéssel kapcsolatos gombok nem aktívak. A jobb oldalon pedig az látható, hogy csak a törlési jog van korlátozva, vagyis a felhasználó a 3-5 közötti jogosultsági csoportba tartozik:



6. ábra: hozzáférés korlátozása jogosultsági szint szerint

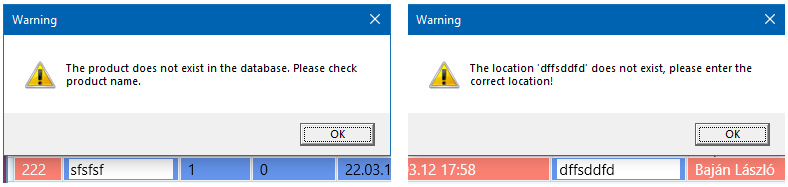
A program fejlesztésénél az egyik elsődleges szempont volt a felhasználóbarát kialakítás volt, különösen az adatok megtekintése, rögzítése, módosítása és törlése tekintetében. Minden tábla adatai egyetlen kattintással megnyithatóak. Az **adatok hozzáadását** az alkalmazás úgy segíti elő, hogy az Add gombra történő kattintáskor egy új beviteli sor jelenik meg, a kurzor megjelenik a szerkesztésre kijelölt első cellában, az adat bevételét követően a kurzort átmozgatja a következő cellába, a mennyiség alapján (a termék árat használva) automatikusan kiszámolja és beírja a teljes összeget, automatikusan beírja az aktuális dátumot, a felhasználó neve alapján automatikusan beírja az adatokat rögzítő felhasználó nevét, és sikeres rögzítés esetén az adott sort zöldre színezi és a tranzakciót szövegesen is visszaigazolja.



7. ábra: rekord hozzáadása az értékesítések (sales) adattáblához

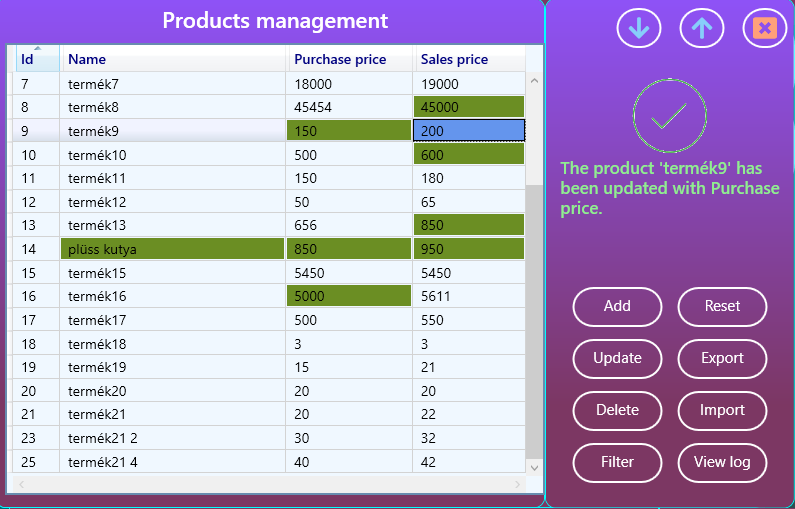
Egy további rekord bevitelekor a program az összes mezőt kitölti az előző adatok alapján, automatikusan új azonosítót (Id-t) rendel a rekordhoz, így a felhasználónak csupán az eltérő adatot kell módosítania. Ennek köszönhetően egy új rekord néhány másodpercen belül hatékonyan és kényelmesen rögzíthető.

Az adatrögzítésnél azonban nemcsak a hatékonyság és a kényelem, hanem a pontosság és a hibák kiszűrése is fontos szempont. Az alkalmazás ezért **minden bevitt adat vonatkozásában ellenőrzi, hogy az helytálló-e**. Ha a felhasználó esetleg téves adatot ad meg, akkor az adatrögzítést a program megállítja és figyelmeztető üzenetet ad, ahogyan az a következő ábrán látható egy téves terméknév és egy téves telephely megadásakor:



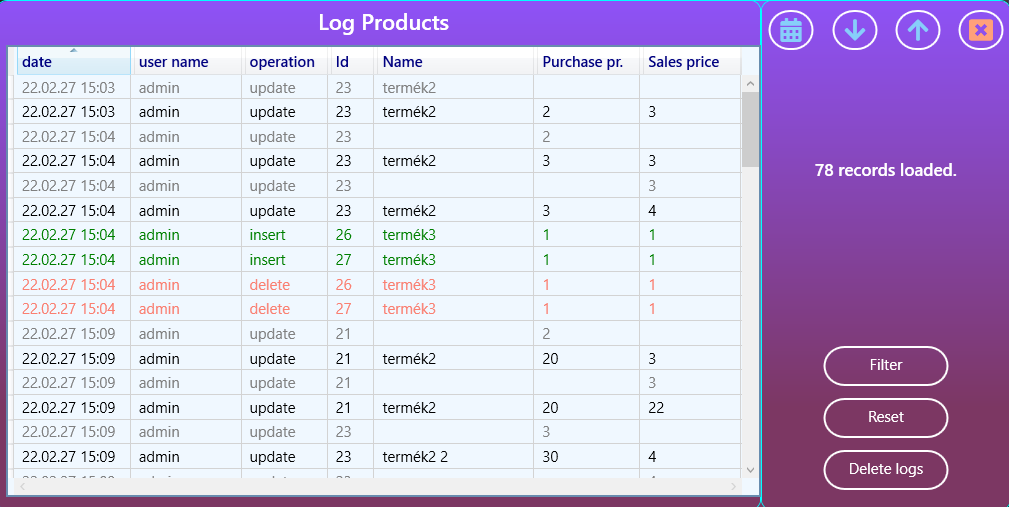
8. ábra: helytelen adatok rögzítésének megakadályozása

Az **adatok frissítésekor** is hasonló ellenőrzés történik. A módosítás folyamatát továbbá nagymértékben segíti, hogy a módosított mezőket az alkalmazás zöldre színezi, így több adat módosításakor a felhasználó minden pillanatban láthatja azokat az adatokat, amelyeket már módosított, és amelyeket még nem:



9. ábra: egyes adatbázis-értékek módosítása, amit zöldre színezett cellák jeleznek vissza a felhasználónak

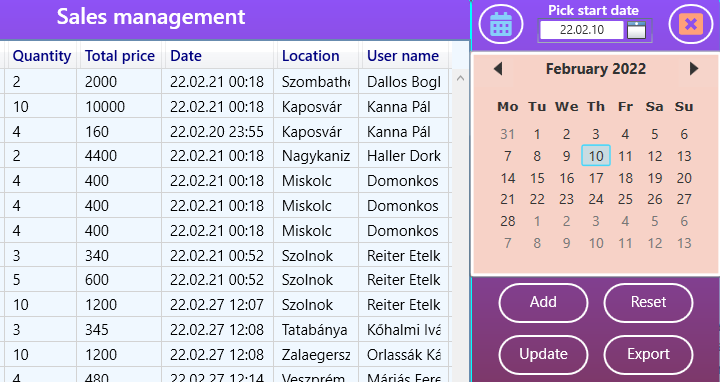
Nemcsak egy multinacionális vállalatnál, de egy kisebb cégnél is fontos, hogy az adatbázist érintő változásokat nyomon tudja követni. Hiszen egy téves adatrögzítés, frissítés vagy törlés történhet, de ennek hiányában is fontos lehet megtudni, hogy bármely módosítás mikor és ki által történt. Ennek visszakereshetőségét biztosítja az alkalmazás **naplózási funkciója**, amely az adatbázist érő minden módosítást azonnal rögzít, és ez bármikor megjeleníthető. Nemcsak az összes táblázat, hanem a naplózási adatok is sorba rendezhetőek bármely oszlop alapján. A különböző műveletek eltérő színekkel vannak jelölni, megkönnyítve a sorok értelmezését. A naplózott rekordok az utolsó 30 napos időszakra kerülnek betöltésre, de ez módosítható, valamint a naplózási fájl tartalma szükség esetén törölhető.



10. ábra: módosított adatok naplózása minden adattábla esetében

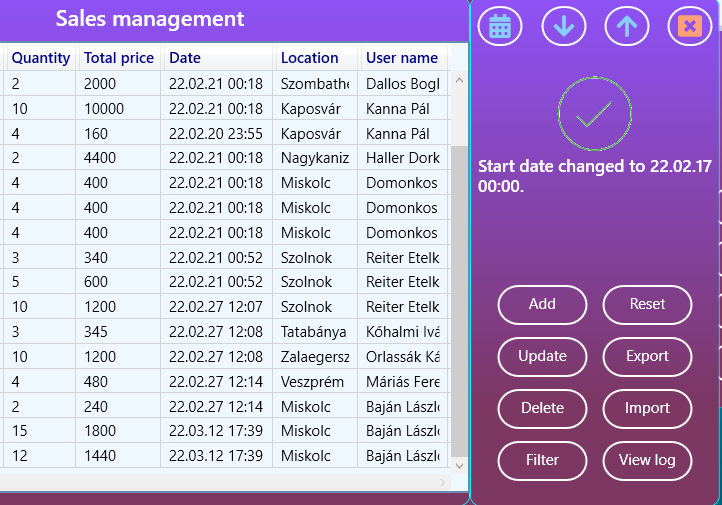
A jobb felső sarokban lévő piros gomb az ablak bezárására szolgál. A felfelé mutató nyíl segítségével **bármelyik ablakot maximalizálhatjuk** a képernyő teljes méretére. Ennek során az oszlopok szélessége és a betűméret is automatikusan kiigazításra kerül. A lefelé mutató nyílra történő kattintáskor az ablakot az **eredeti méretére állíthatjuk vissza**.

A **naptár ikonnal** az alapértelmezetten betöltött 30 napos időszakot növelhetjük, vagy csökkenthetjük. Az ikonra történő kattintáskor megjelenik egy naptár, mutatva az aktuálisan beállított kezdő vagy zárónapot:



11. ábra: lekért adatok kezdő- vagy záródátumának beállítása dátumkiválasztó ablak segítségével

Ily módon csupán egy kattintásra van szükség a kívánt dátum beállítására. Az alább mintán az új kezdődátum február 17-e lett, amiről az alkalmazás vizuális visszajelzést adott:

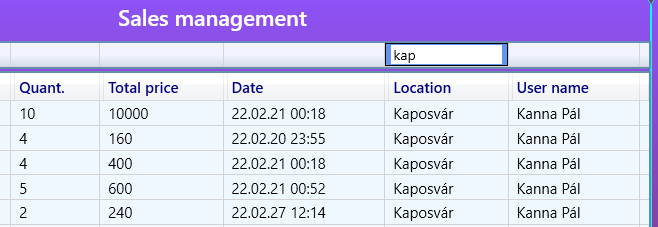


12. ábra: megjelenített adatok kezdődátumának beállítása

Az alapértelmezett 30 napos intervallum lehetővé teszi, hogy egy jelentős méretű, akár több évre visszamenő adatbázis esetén az erőforrásokat ne vegyük feleslegesen igénybe, miközben szükség esetén akár több évre is vissza lehet nyúlni.

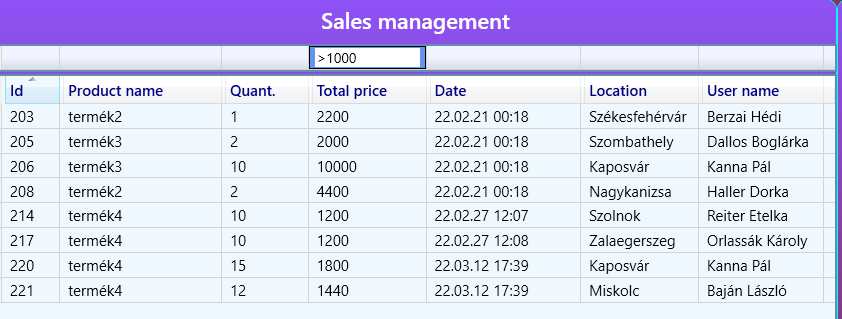
A betöltött adatokat több tekintetben is lehet manipulálni, az adatbázis újabb lekérdezése nélkül. **Az adatok minden mező tekintetében sorba rendezhetőek** az oszlop fejlécre történő kattintással, az ismételt kattintás a rendezési sorrendet megváltoztatja.

A Filter gombra kattintással bármelyik mező vonatkozásában **szűrést végezhetünk**. A funkció felhasználóbarát kialakítását mutatja, hogy egy adott karakter begépelését követően már megtörténik a szűrés, így akár egy szórészlet alapján is eredményt kapunk:



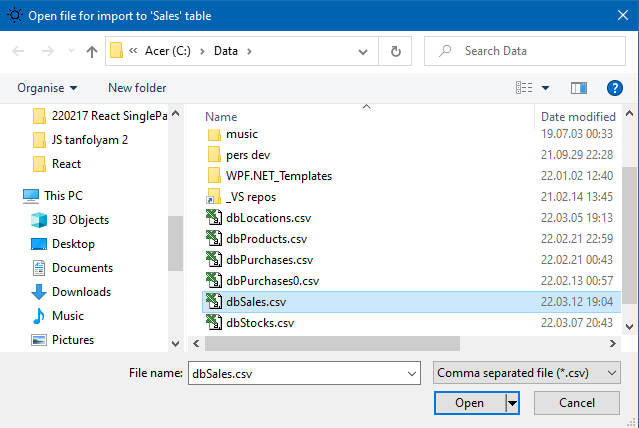
13. ábra: rekordok szűrése telephely-név szerint

A felhasználói élményt tovább növeli, hogy szám- és dátumadatok esetén **relációs keresést is lehet végezni**. A lenti példán az 1000 forintnál nagyobb árbevételű eladásokra történt szűrés:



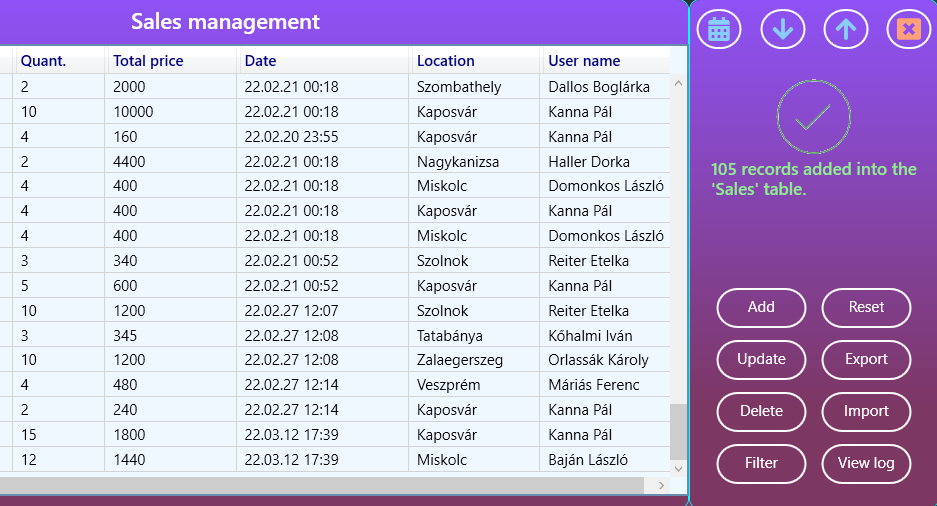
14. ábra: értékesítési adatok relációs szűrése, ahol a bevételnek 1000 front felett kell lennie

Egyes (adminisztrátori jogosultsággal rendelkező) felhasználók úgy is dönthetnek hogy az egyes rekordokat nem kézzel rögzítik, hanem **fájlokban található adatok tömeges beolvasása** révén. Az Import gomb megnyomásakor egy fájl megnyitási dialógus ablak jelenik meg, a csv (vesszővel elválasztott mezőjű szöveges) fájlokat megjelenítve:



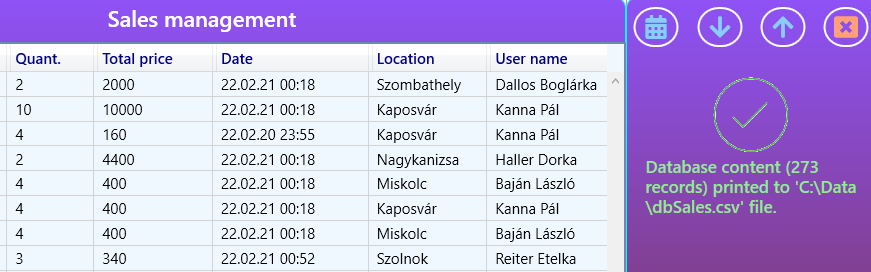
15. ábra: fájlban tárolt értékesítési adatok kiválasztása

A megfelelő dbSales.csv fájl kiválasztása után a program soronként ellenőrzi a fájl tartalmának helyességét, és a helyes adatokat beolvassa és menti az adatbázisba, és a művelet eredményéről – mind a helytálló rekordok beolvasása, mind az esetlegesen hibás rekordok beolvasásának meghiúsulása vonatkozásában – szöveges visszajelzést ad. Az alábbi példában 105 értékesítési rekord került rögzítésre egyetlen import művelet során:



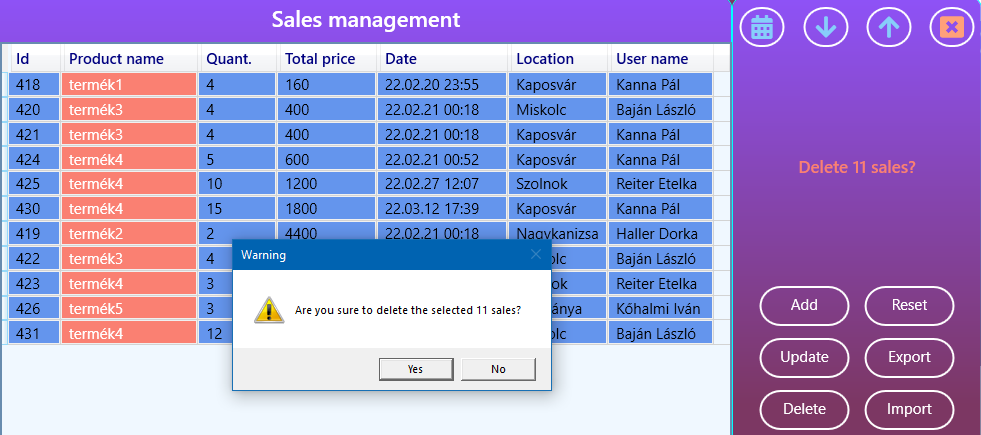
16. ábra: értékesítési adatok tömeges beolvasása fájlból

Egy ilyen alkalmazás esetében az adatbiztonságra is kiemelt hangsúlyt kell fektetni. Ennek jegyében hoztuk létre az Export funkciót. Ennek segítségével **az adatbázis bármely táblája bármikor kiexportálható**, és szükség esetén visszatölthető. A következő képernyőfelvétel 273 értékesítési rekord exportálása utáni állapotot tükrözi:



17. ábra: adatbázis adatok kimentése fájlba

Az **adatok törlése** adminisztrátori jogosultságot igényel. Nagyszámú rekord törlése jelentős időráfordítást igényelne az ilyen munkatársaktól, ha ezt a műveletet egyesével kellene elvégezniük. A felhasználóbarát alkalmazásunk azonban tetszőleges számú rekord törlését teszi lehetővé egyetlen művelet keretében. A törlés előtt a program egy ablakban összesítve megjelenítni a törlésre kijelölt rekordokat, piros színnel kiemelve, és csak egy megerősítést követően törli a kiválasztott rekordokat. Az alábbi képernyőfelvételen 11 értékesítési rekord törlésének megerősítését kéri az alkalmazás.



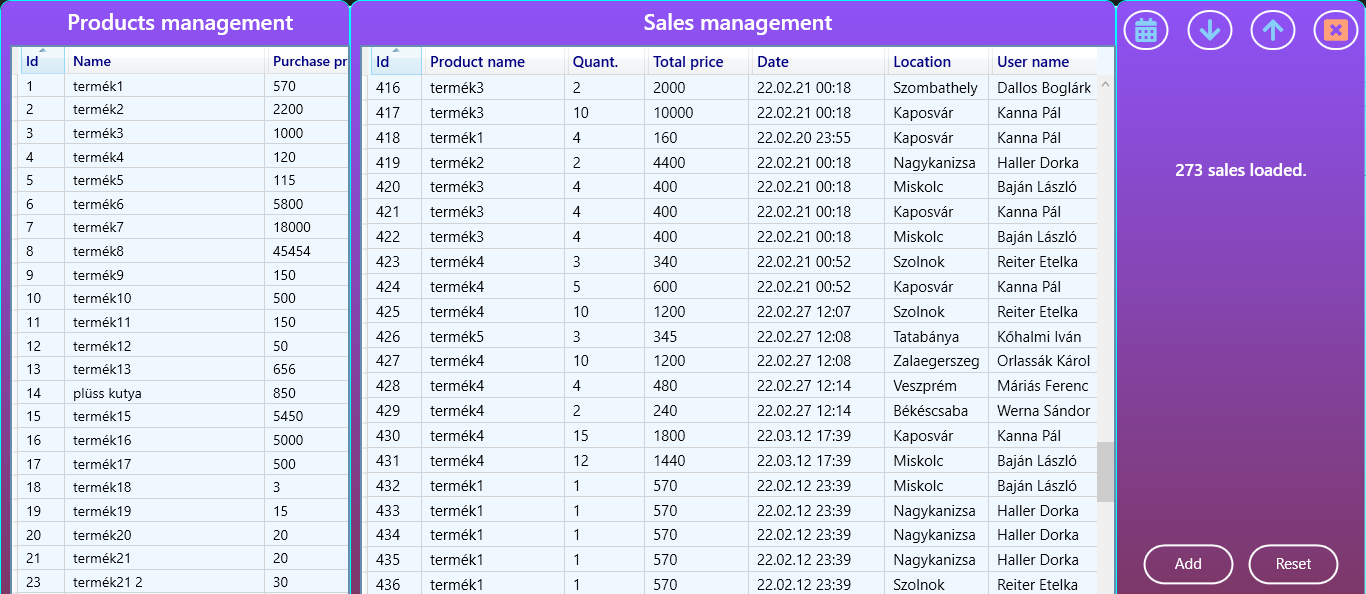
18. ábra: rekordok törlése előtt megerősítő ablak megjelenése

A munkavégzés hatékonyságát növeli, ha a felhasználó egymás mellett több ablakot megjelenítve tud dolgozni. Az alkalmazás támogatja bármely ablak tetszőleges átméretezését, egymás mellé helyezését. Ennek elősegítésére az alkalmazás **előre definiált elrendezés** nézetekkel is rendelkezik, amelyekből a felhasználó tetszés szerint választhat. Sőt, egy elrendezési típushoz két lehetőség is tartozik. A bal oldali elrendezések két ablakot helyeznek egymás mellé, ahol mindkét ablak esetében látszanak a funkció gombok. A jobb oldali elrendezések esetén a csak a jobb oldali (fő) ablak takarja a bal oldali segédablak gombjait, így az nagyobb méretben jeleníthető meg.



19. ábra: előre definiált elrendezési sablonok alapján több ablak megnyitása egy kattintással

A következő ábrán látható elrendezés a termékek és az eladások ablakokat jeleníti meg egymás mellett átfedéssel, egyetlen kattintással, a teljes képernyőt kihasználva. Ez akkor a leginkább előnyös, ha a felhasználó az értékesítések táblában véges módosításokat, a termékek táblában nem, azt csupán megtekintésre nyitotta meg.



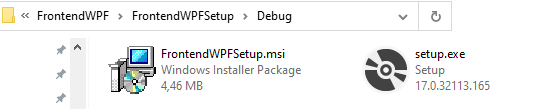
20. ábra: termékek és eladások adatait tartalmazó ablakok megjelenítése a kiválasztott elrendezési sablon szerint

Az alkalmazás tartalmaz egy **jegyzettömböt** is, amelyben .txt vagy .rtf kiterjesztésű fájlok megnyitására és elmentésére használható. A beépített jegyzettömbbel a szöveg tetszőlegesen formázható, így .rtf kiterjesztéssel történő mentés esetén később a dokumentum az alkalmazott kiemelésekkel, aláhúzásokkal és színezésekkel nyitható meg. A kívánt színt a kijelölt szövegre a legördülő színválasztóval lehet kiválasztani.



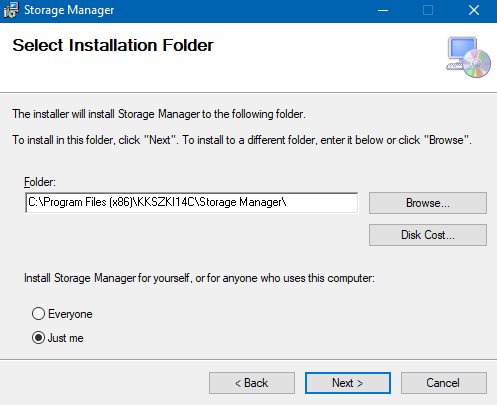
21. ábra: beépített jegyzettömb alkalmazás .txt vagy .rtf fájlok elmentéséhez vagy megnyitásához

Nemcsak a használata, hanem a **program telepítése** is felhasználóbarát. A telepítő fájlok a \FrontendWPF\FrontendWPFSetup\Debug\ mappában találhatóak:



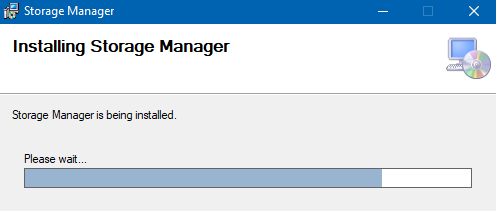
22. ábra: a WPF alkalmazás telepítőkészlete

A telepítő indításakor a felhasználót telepítő-varázsló vezeti lépésről lépésre. Az üdvözlő ablak után a felhasználó kiválaszthatja a telepítés útvonalát és terjedelmét, amihez alapértékeket kap:



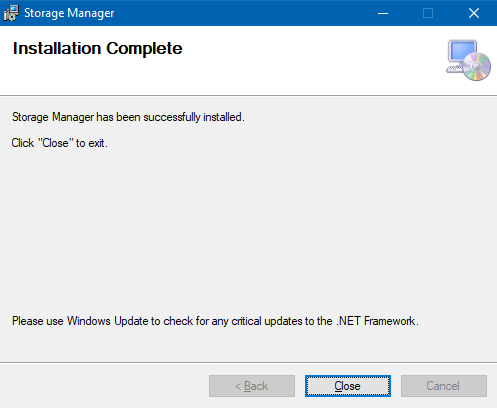
23. ábra: telepítési mappa kiválasztása

Ezt egy megerősítő ablak követi, ahol a telepítés indítható. Utána megkezdődik a telepítés, aminek folyamatáról is tájékoztatást kapunk:



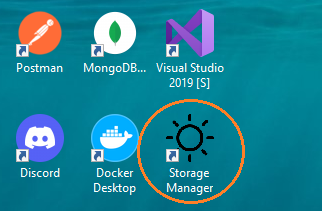
24. ábra: telepítési folyamat kijelzése

A folyamat a sikeres telepítésről szóló üzenettel zárul:



25. ábra: sikeres telepítés visszaigazolása

A telepítő a Windows asztalon automatikusan létrehozza a program parancsikonját, amellyel az alkalmazás kényelmesen indítható:



26. ábra: az alkalmazás parancsikonja

### REACT (webes frontend)

A React (vagy React JS) egy olyan JavaScript nyelven írt nyílt forráskódú könyvtár, ami elősegíti a felhasználói felületek programozását, elsősorban webes alkalmazások létrehozását. A könyvtár alapú megközelítés lehetővé teszi, hogy a fejlesztő teljes szabadsággal rendelkezzen a fejlesztésre használt technológiák kiválasztásában. Vagyis eldöntheti, hogy a React könyvtárból csupán egyetlen komponenst hív segítségül, vagy hogy akár a React legyen a fejlesztői környezetének súlypontja. Szemben egy keretrendszerrel, ami nagymértékben meghatározza és előírja a fejlesztés szabályait, kereteit.

A React előnyei:

- megkönnyíti komponensek írását;

- újra felhasználható komponensekkel növeli a hatékonyságot, átláthatóságot

- keresőmotor-barát forráskód megjelenítés

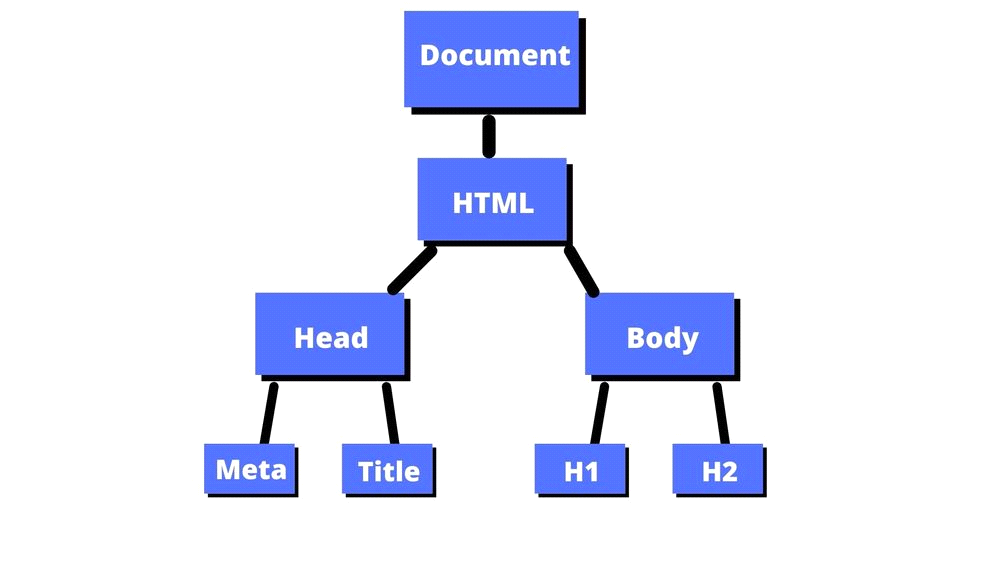
- magas népszerűsége miatt széleskörű fejlesztési támogatás

- további tökéletesítése várható a facebook fejlesztői csapatának köszönhetően

A számítógépes alkalmazáshoz egyik fontos tulajdonsága az interaktivitás. A webes alkalmazások ezt jellemzően a böngészőben megjelenő HTML elemek közvetlen módosításával érik el.

#### DOM \*(1)

A böngésző ablakát képező objektum a "document", ezen jelenik meg az összes HTML elem, az ezek felépítését leíró modell, a DOM (Document Object Model) ennek köszönheti a nevét. A HTML elemek jellemzően egymásba ágyazódnak, egy fához hasonló szerkezetet képezve.



27. ábra: Document Object Model szerkezeti felépítése

A HTML elemek a tartalmat jelenítik meg, a CSS stílusok a megjelenés módját határozzák meg, az ilyen tartalmat és stílust pedig a JavaScript segítségével tudjuk elérni, módosítani.

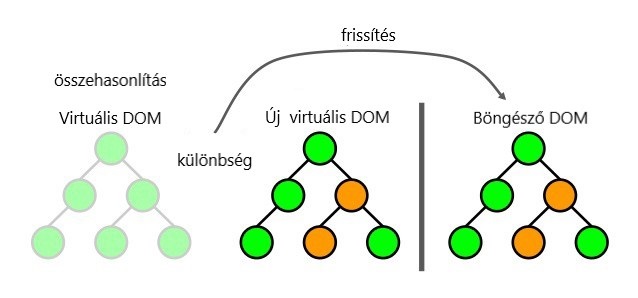
A React előtti technológiák a böngészőben megjelenő elemeket közvetlenül manipulálták a DOM segítségével. Például egy felhasználó bejelentkezésekor egy JavaScript kód a böngésző jobb felső sarkában megjelenítette a felhasználó nevét, a bal oldalon behívta a felhasználó profilját, egy másik oszlopban megjelenítette a barátait is, majd a másik oldalon egy csevegő ablakot is fel kellett dobnia, és felette még ki kellett írnia, hogy hány új üzenetet és értesítést kapott.

A megírt kód a különböző böngészőkben néha eltérően viselkedett, és jelentős programozási ráfordítást jelentett, hogy egy weboldal a legtöbb böngészőben azonos módon jelenjen meg. A jQuery megjelenése erre adott választ, és nemcsak a böngészők közötti kompatibilitást segítette elő, hanem kis JavaScript könyvtárként a HTML elemek elérését és módosítását is egyszerűbbé tette. A fejlesztők idővel egyre nagyobb alkalmazásokat hoztak létre, ahol a JavaScript és az interaktivitás növekvő szerepet kapott. Megjelentek a JavaScript könyvtárak, amelyek az egyre több JS funkciót és fájlt rendszerbe szervezik. A felhasználói élmény növelése, a betöltési idők csökkentése céljából létrejöttek az egyoldalas alkalmazások (SPA), és aszinkron kommunikációt elősegítő technológiák, mint például az AJAX. 2010 körül a Google által kifejlesztett Angular JS keretrendszer vált a webes alkalmazások fejlesztésének egyik meghatározó eszközévé. Ez már valóban nagy és összetett alkalmazások létrehozását tett lehetővé. A HTML, CSS és JavaScript kód továbbra is jellemzően elkülönült, a növekvő interaktivitás miatt ezek összekapcsolása és kezelése egyre több JavaScript kód megírását igényelte. A DOM manipulálása egy összetett alkalmazásban egyre nehezebbé vált, hiszen egyetlen esemény akár több tucatnyi reakciót válthat ki, amelyek szintén további reakciókat eredményezhetnek. Ez megnehezíti az események és a hozzátartozó esetek átláthatóságát, karbantartását.

A DOM manipuláció erőforrásigényes feladat: a böngésző megjeleníti a módosított elemet, és ekkor az oldal elrendezését is gyakran módosítania kell. Rádásul néhány JS keretrendszer felesleges DOM frissítéseket is végez, például egy hosszú lista egyetlen elemének módosulásakor akár a teljes listát újra építheti. A React alkotói erre a problémára hatékony megoldást adtak.

#### Virtuális DOM \*(2)

A Reactben egy DOM objektumhoz egy virtuális DOM objektum kapcsolódik. Ezt a DOM objektum leegyszerűsített másolataként képzelhetjük el, ami nem látható a böngészőben. Ez hátránynak tűnik, de éppen ebből adódnak az előnyei. A módosítása ezáltal sokkal gyorsabb és egyszerűbb, hiszen ilyenkor semmi nem jelenik meg a képernyőn, a tényleges DOM nem módosul. A virtuális DOM változása esetén a React kizárólag azokat a DOM objektumokat frissíti, amelyek virtuális megfelelői módosultak, ami jelentősen jobb teljesítményt eredményez a többi hasonló technológiához képest.



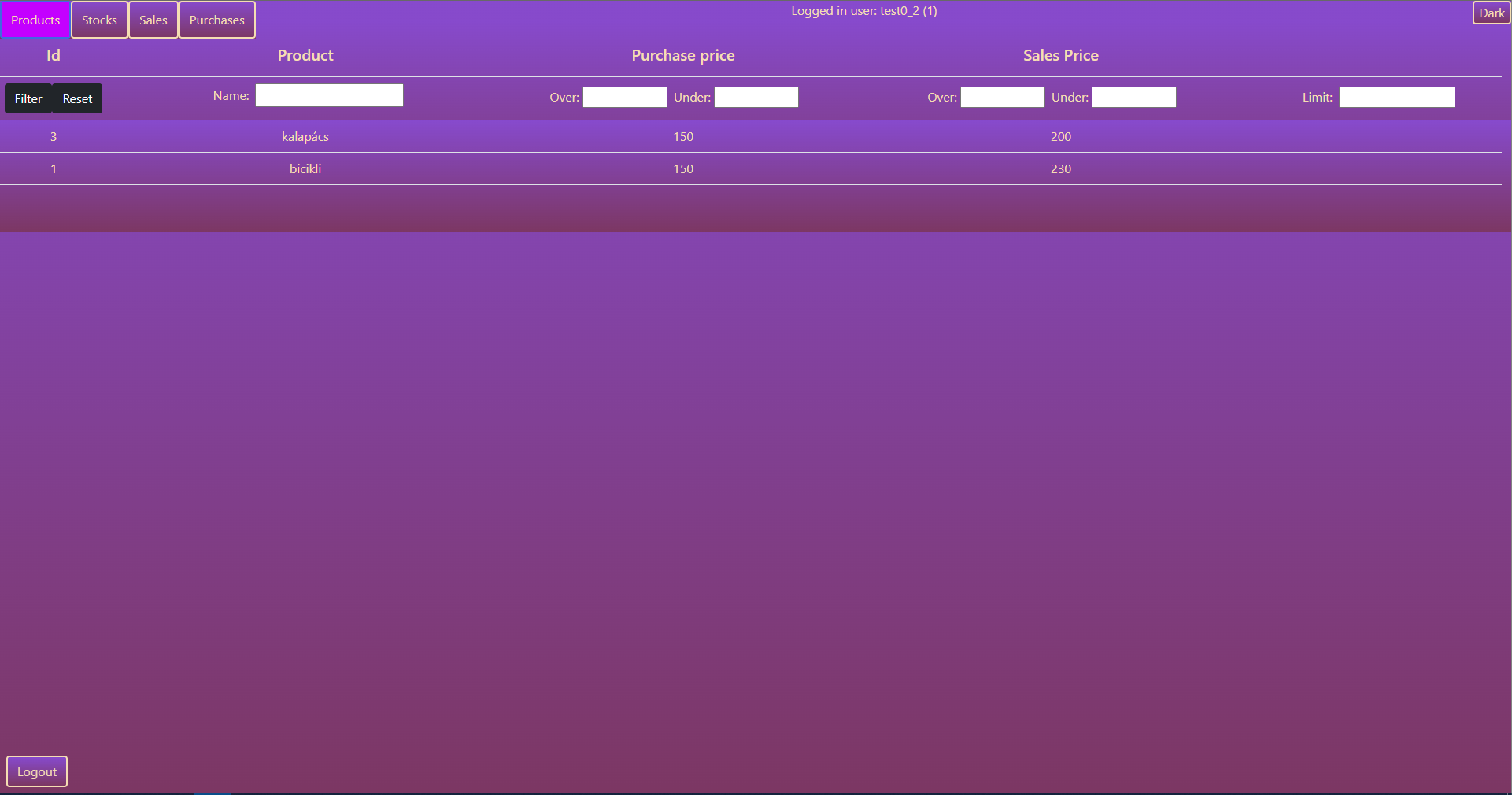
28. ábra: virtuális DOM folyamatábra

### A frontend weboldal felépítése

A webes frontend alkalmazás felépítése hasonló az asztali alkalmazáséhoz, minden táblának külön oldala van, legelőször egy bejelentkező oldallal találkozik a felhasználó, ahol egy kép-diavetítés (slide) is található egyben, bár ennek nagyrészt csak dizájn szerepe van az oldalon, hogy kevesebb legyen az üres tér a bejelentkező felületen.

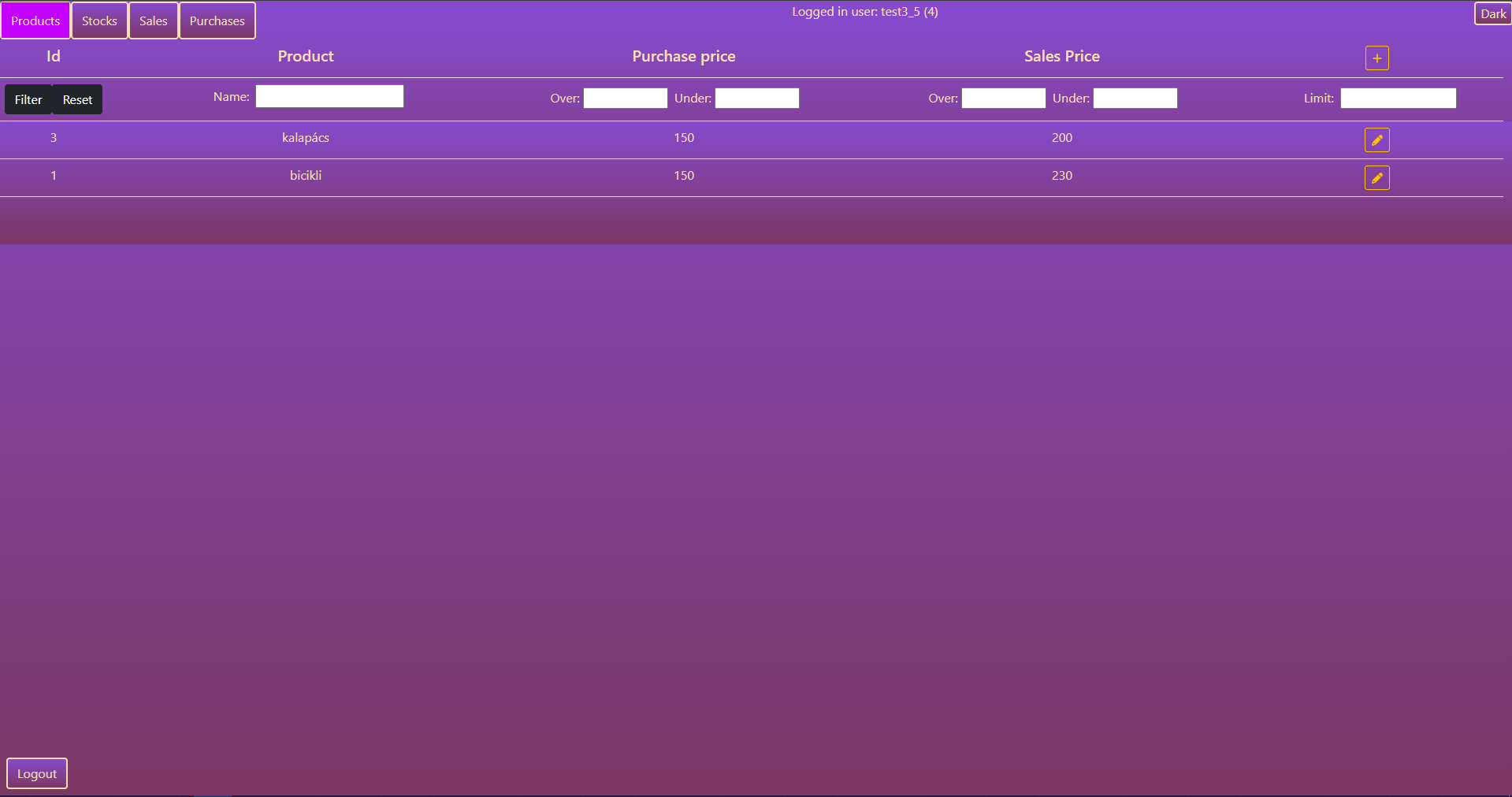
Az egyes táblák módosítási lehetőségei a felhasználó jogosultsági szintjétől függ, a rendszer azonos az asztali alkalmazásban leírtakkal, az egyes szintek a következőképp látják az oldalt belépés után:

0-2-es szintek:



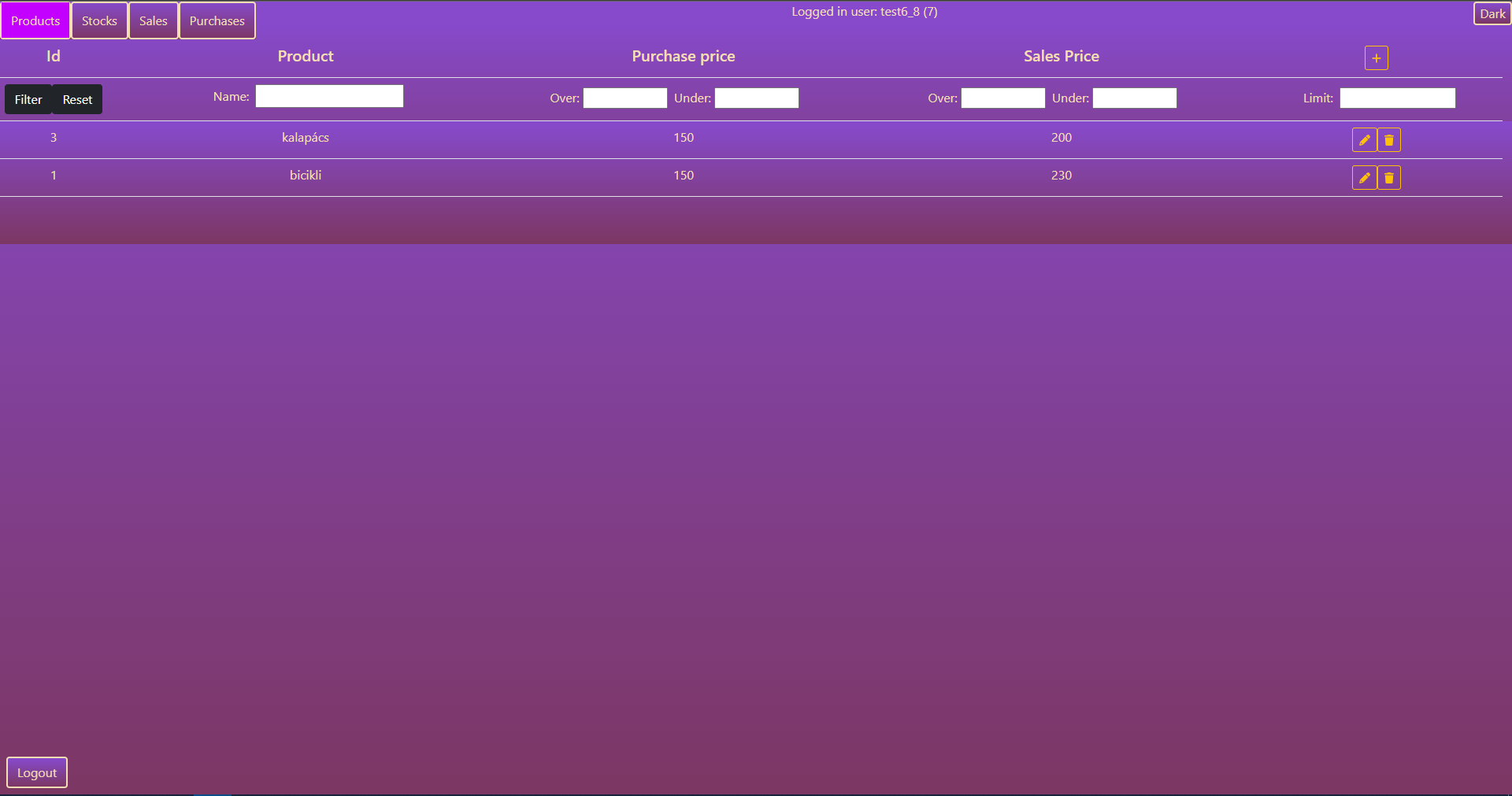
29. ábra: 0-2 jogosultsági szintek közötti felhasználó számára látható webes felület

3-5-ös szintek:



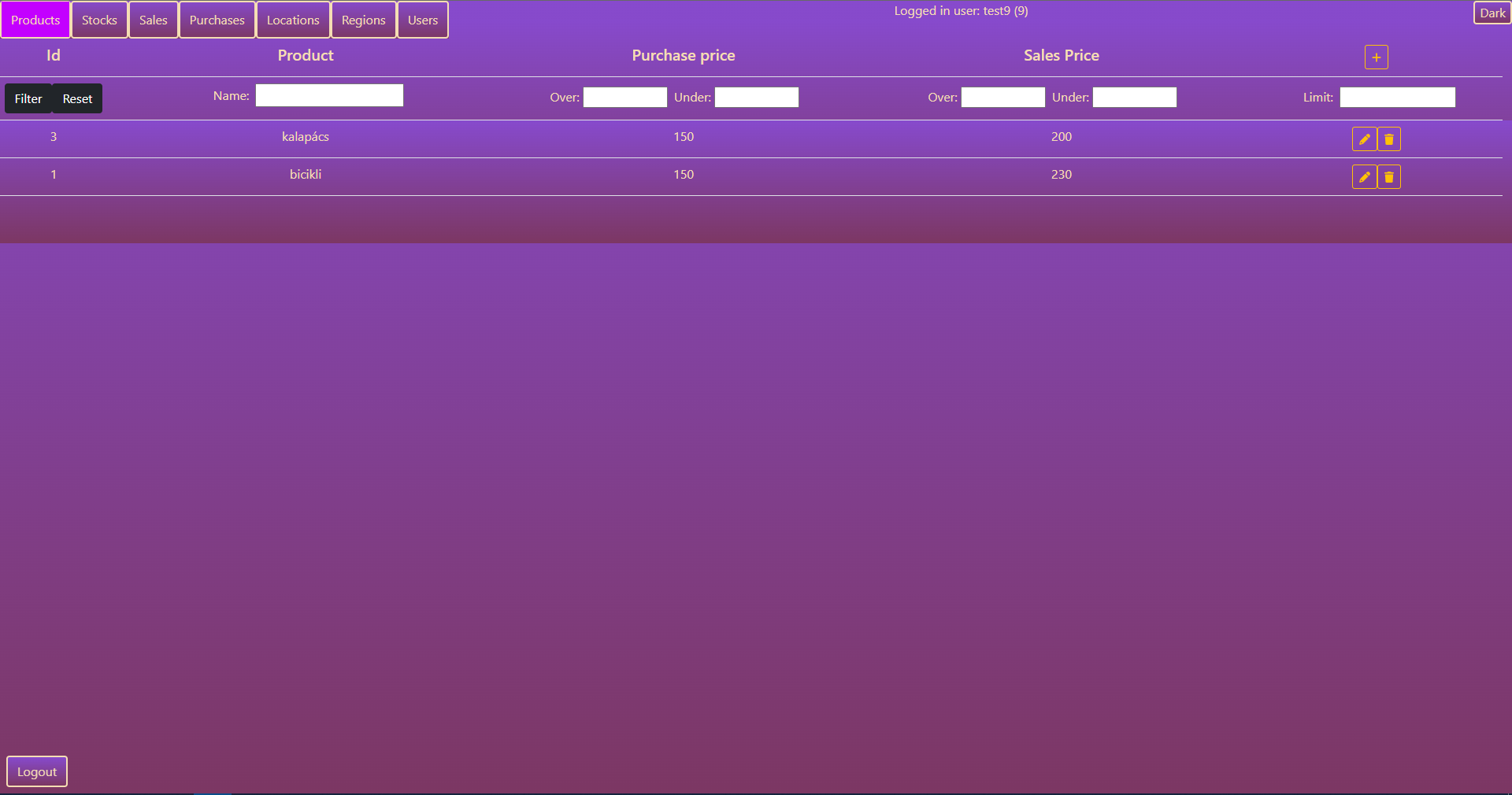
30. ábra: 3-5 jogosultsági szintek közötti felhasználó számára látható webes felület

6-8-as szintek:



31. ábra: 6-8 jogosultsági szintek közötti felhasználó számára látható webes felület

9-es szint:



32. ábra: 9 jogosultsági szinttel rendelkező felhasználó (admin) számára látható webes felület

Bármikor lehet választani 2 színséma közül, az első az asztali alkalmazás mintájára készült, a második pedig nagyrészt sötét színeket használ, így éjszakai használatkor a szemet kíméli. A színséma helyileg van tárolva, így ameddig ugyanazon a gépen vagyunk, a preferencia el van tárolva és mindig az fogja fogadni a felhasználót.

Az alkalmazás mindaddig elmenti a bejelentkezett felhasználót, ameddig az ablakot be nem zárja. Ez főként biztonsági okokból van így. Vagyis a felhasználónak ismét be kell jelentkeznie, ha a böngészőt bezárja vagy a számítógépet leállítja.

##### MySQL adatbázis \*(3)

A MySQL egy relációs adatbázis-kezelő rendszer. A relációs adatbázisok külön táblákban tárolják az adatokat, ahelyett, hogy az összes adatot egyetlen nagy tárhelybe helyeznék el. A logikai modell az adatbázisok, táblák, nézetek, sorok és oszlopok mint objektumok révén rugalmas programozási környezetet kínál. Az adatmezőkhöz számos adattípus közül választhatunk, és a mezők közötti kapcsolatokat is szabályozhatjuk attól függően, hogy egy az egyhez (1:1), egy a többhöz (1:N) vagy több a többhöz (N:M) kapcsolatról van szó. Ezen kívül további tulajdonságokat rendelhetünk az egyes mezőkhöz (pl. egyedi, kötelező vagy nem kötelező, automatikus növelés, stb.). Az adatbázis érvényesíti ezeket a szabályokat, így egy jól megtervezett adatbázis esetén az alkalmazás soha nem lát inkonzisztens, duplikált, árva, elavult vagy hiányzó adatokat.

A MySQL név SQL része a „Structured Query Language” rövidítése. Az SQL az adatbázisok elérésére használt legelterjedtebb szabványosított nyelv. A programozási környezettől függően közvetlenül futathatjuk az SQL utasításokat (például jelentések létrehozásához), egy másik nyelven írt kódba is beágyazhatjuk, vagy nyelvspecifikus API-t használhatunk, amely elrejti az SQL szintaxist.

A MySQL adatbázis szerver nagyon gyors, megbízható, méretezhető és könnyen használható. A MySQL szerver kényelmesen futhat asztali számítógépen vagy laptopon, más alkalmazások, webszerverek mellett, kevés figyelmet igényel. Eredetileg nagy adatbázisok kezelésére fejlesztették ki, és évek óta sikeresen használják nagy igénybevételt jelentő éles környezetben. A funkciók gazdag és hasznos készletét kínálja, miközben folyamatos fejlesztés alatt áll. Kapcsolódása, sebessége és biztonsága miatt a MySQL szerver kiválóan alkalmas az internetes adatbázisok elérésére.

###### Az adatbázis szerkezete \*(4)

Az adatbázis az adatok és a köztük lévő összefüggések rendszere, amelyet egymás mellett tárolunk. A relációs adatmodellben az adatokat egymással logikai kapcsolatban álló táblákba rendszerezzük. Ennek során követelmény, hogy minden oszlopnak egyértelmű neve van, minden sorban ugyanazok az oszlopok vannak, az oszlopokban található adatok meghatározott értéket vehetnek fel, az oszlopok soronként csak egy értéket vehetnek fel, és a táblát a neve egyértelműen azonosítja.

A tábla a logikailag összetartozó adatokat foglalja össze.A tábla oszlopokból és sorokból áll, amelyeket mezőknek, illetve rekordoknak nevezünk. A rekord az adatbázis egy sora. Egy rekordban tároljuk az egymással összefüggő adatokat. Ezzel szemben a mező az adatbázis egy oszlopa, amelyben az egyedek tulajdonságértékeit tároljuk.

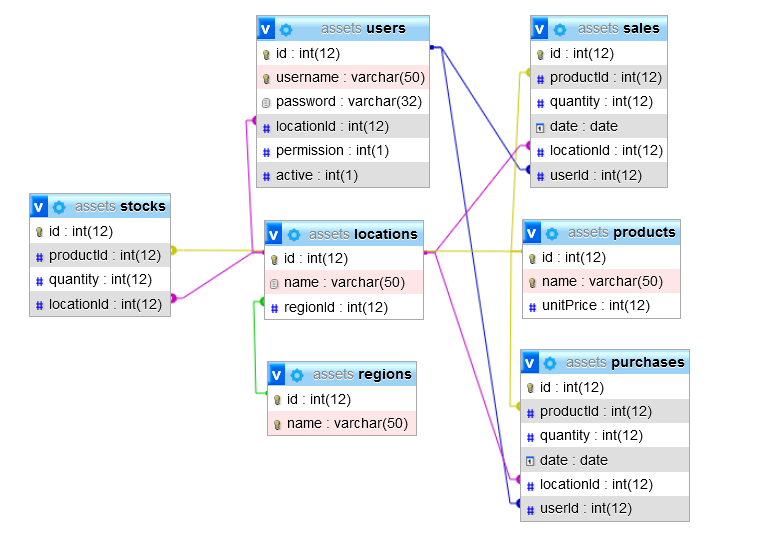
Az egyed az, amit le akarunk írni, amelynek az adatait tároljuk és gyűjtjük az adatbázisban. Az egyedet idegen szóval entitásnak nevezzük. Egyednek tekinthetünk például egy személyt. Az attribútum (vagyis tulajdonság) az egyed valamely jellemzője. Az egyed az attribútumok összességével jellemezhető. Egy személy egy jellemzője lehet például a neve. Az egyedre vonatkozóan megadott tulajdonságok összességét egyedtípusnak nevezzük. Az egyedre vonatkozóan megadott konkrét tulajdonságokat egyed-előfordulásnak nevezzük. Egy egyed-előfordulás például Kis Ede, aki 29 éves, 183 cm magas, kék szemű, barna hajú.

Elsődleges kulcsnak hívjuk a a tábla rekordjainak egyértelmű azonosítóját, ami egyedi értékekkel rendelkezik. Az idegen kulcs olyan azonosító, amelynek segítségével egy másik tábla elsődleges kulcsára hivatkozhatunk.

#### Az alkalmazás mögötti adatbázis

Az adatbázis tervezésekor hatékonysági szempontokat is figyelembe vettünk, ennek fényében a lehető legátláthatóbb szerkezet kialakítására törekedtünk. Elvileg lehetséges lett volna külön tábla létrehozása külön eltérő beosztású munkatársak számára. Azonban több szempontból is optimálisabb megoldásnak tűnt, ha az összes felhasználót egy táblában tartjuk nyilván, és jogosultsági szintekkel különböztetjük meg őket. Ezáltal további típusú felhasználó is könnyen definiálható, mint például az adminisztrátor.

Az adatbázisunk EK diagramja a következő:



33. ábra: RESTful alkalmazás adatbázisa

### WCF (backend)

A WCF, vagyis Windows Communication Foundation egy alapvetően szolgáltatás orientált alkalmazásokhoz készült keretrendszer. A segítségével adatokat küldhetsz egyik végponttól a másikba, legyen az egy kliens oldali alkalmazás, vagy egy másik szerver oldali szolgáltatás. Az üzenetek egyszerű karakterektől kezdve XML, JSON, illetve bináris adatfolyamokig bármilyen komplexitásúak lehetnek. A WCF tehát használható üzleti tranzakciók feldolgozására, időjárás, úti forgalom adatok továbbítására vagy akár egy felhasználók közötti üzenetküldő alkalmazásra, és még sok másra. Ezeket természetesen a WCF megjelenése előtt is meg lehetett csinálni, de a WCF sokkal könnyebbé teszi a végpontok közötti kommunikációt. \*(5)

A szolgáltatás orientált keretrendszerek előnye, hogy a kiszolgált végpontokkal elég lazán kapcsolatban lenniük és nem kell semmit vagy szinte semmit változtathatatlanul belekódolni a kliens alkalmazásunkba, mint például egy előre meghatározott cím, ahol el lehet érni a szolgáltatást (pl. http://localhost:3000/), akármi is változik annak kódjában, az egyetlen változtathatatlan dolog ez az IP cím lesz.

Az üzenetváltásra többféle séma is van. A leggyakoribb megoldás a kérés/válasz séma, ahol az egyik végpont adatot kér a második végponttól, a második végpont pedig erre válaszol a kérésnek megfelelően. De van egy irányú séma is, amely esetben egy végpont küld adatot a másiknak anélkül, hogy az válaszra várna ez után. Egy komplexebb séma a két irányú csere, ahol két végpont kapcsolatot létesít, és oda visszaküld adatokat hasonlóan egy üzenetváltó alkalmazáshoz.

A WCF támogatja a szolgáltatás metaadatának kiadását ipari szabványoknak megfelelő formátumokban, mint WSDL, XML vagy WS-Policy. Ezek a metaadatok felhasználhatóak arra, hogy automatikusan előállítsunk és konfiguráljunk klienseket, amik elérhetik a szolgáltatásunkat. http és HTTPS, illetve Web Service Metadata Exchange szabványoknak megfelelve egyaránt kiadható ez a metaadat.

Az üzeneteket titkosítani lehet a bizalmas adatok védelme érdekében, ez esetben a felhasználónak hitelesítenie kell magukat, hogy ezeket az üzeneteket megkaphassak. A titkosítás megoldható több jól ismert szabvánnyal, mint az SSL vagy a WS-SecureConversation.

Az üzeneteket több beépített protokollon és kódoláson keresztül. A leggyakoribb protokoll és kódolás séma kódolt szöveges SOAP üzeneteket küldeni http (HyperText Transfer Protocol) segítségével Interneten keresztüli használatra. A WCF továbbá megengedi a TCP, nevesített csövek, illetve MSMQ segítségével való üzenetküldést is. Az üzenetek vagy szövegként, vagy optimalizált bináris formában kódolva küldhetők. Bináris adatok esetén az MTOM szabvány segítségével hatékonyan küldhetünk adatot. De abban az esetben, ha egyik formátum sem felel meg az elvárásainknak, elkészíthetjük a saját szállítás vagy kódolás formátumunkat is.

A tartós üzenetek olyan üzenetek, melyek soha nem vesznek el valamilyen kommunikációs probléma fellépése esetén. Az üzenetek egy ilyen sémában mindig elmentődnek egy adatbázisba, így abban az esetben, ha valami oknál fogva megszakad a kapcsolat, az adatbázis lehetővé teszi, hogy folytassuk az üzenetváltást onnan, ahol a szolgáltatás megszakadt. A Windows Workflow Foundation segítségével szintén létrehozhatók ilyen tartós üzenetek.

A REST egy példája a fejlődő Web 2.0 technológiáknak. A WCF konfigurálható, hogy fel tudjon dolgozni „egyszerű” XML adatot, ami nincs SOAP csomagba csomagolva. A WCF továbbá kibővíthető, hogy egy egy kifejezett XML formátummal dolgozzon, mint az ATOM, ami egy népszerű RSS szabvány, vagy egyéb nem XML formátumokba, mint a Javascript Object Notation, vagyis JSON.

A WCF egy nagyon rugalmas felület, ebből kifolyólag rengeteg más Microsoft termékkel használható. Az egyik ilyen például a Windows Workflow Foundation(WF). A munkamenetek leegyszerűsítik az alkalmazásfejlesztést azzal, hogy a munkamenet egyes lépéseit úgynevezett „aktivitásokba” foglalják. Az első verzióban még ugyan nem, de az azt követőben beépített része lett a WCF. Ez lehetővé tette, hogy bármilyen munkamenetet könnyedén ki tudjunk szolgálni egy WCF szolgáltatással. További termékek közé tartozik még többek között a:

* Microsoft BizTalk Server R2:

- egy kommunikációs technológia, ami egy szabvány formátumból alakít át adatot egy másikba   
- WCF szolgáltatást használva tud adatokat továbbítani a központi tárolójába, ahol ezek az adatok átalakítódnak

* Microsoft Silverlight:

- média-igényes weboldalak készítésére készült, mint például egy videó streaming szolgáltatás  
- a második verziótól kezdődően a WCF egy beépített része lett a technológiának, ahol a kliens/szerver kommunikáció elvégzésére használható

* Windows Server AppFabric:
* kifejezetten arra lett felépítve, hogy WCF kommunikációt használó alkalmazásokat vezéreljen  
  - kifejezetten WCF alkalmazásokra tervezett konfigurációs beállításokkal és terjedelmes vezérlési lehetőségekkel van ellátva

### A raktár programunk WCF rendszere

A rendszer backendje 3 service-re van szétosztva, amik egyedi portokon működnek, a 3000-es porton a felhasználó kezelés történik, a 3001-es porton a raktárok adatainak kezelése, a 3002-es porton pedig az üzletek és régióik kezelése történik. A portokra nem azonos mennyiségű tábla jut, logikailag lettek elosztva, a felhasználói port csak felhasználókkal kapcsolatos lekéréseket intézi, a raktár port az egyes boltok raktáraival, az azokban történő változásokkal (eladások és vásárlások nyilvántartása) illetve a termékekkel kapcsolatos lekéréseket intézi. Az üzlet port pedig az egyes üzletek és a hozzájuk tartozó logisztikai régiók lekéréseit kezeli.

A be- és kijelentkeztetés a felhasználói porton történik, ahol minden felhasználó egy egyedi globális azonosítóval (UID) tárolódik el, amely kijelentkezésig vagy a következő bejelentkezésig marad eltárolva (ha az ilyen ismételt bejelentkezés 8 órán belül történik az utolsóhoz képest és ezt nem előzte meg ugyanazon felhasználó kilépése). Ez főként azért van így, hogy a felhasználó egy időben bejelentkezve tudjon maradni a webes és az asztali felületen, valamint véletlen ablak vagy alkalmazás bezárások ne okozzanak új UID generálást. A felhasználói adatokat minél kevesebb ideig akarjuk egy adott UID-hoz rögzíteni, így ezt a kompromisszumot választottuk kényelem és biztonság között.

Az adatbázis működését jelenleg egy platformfüggetlen webszerver-szoftvercsomag, a XAMPP biztosítja egy MySQL kiszolgálón keresztül, miközben a kezelését a PhpMyadmin felületen keresztül teszi lehetővé (az Apache webkiszolgáló alkalmazáson keresztül). Az adatbázisnak más MySQL-kompatíbilis környezetben is működnie kell.

A hosztolás érdekében vagy meg kell nyitni ezeket a portokat a tűzfalon, ahol a rendszer fut, vagy adminisztrátorként elindítani a programot, amely esetben a programnak nem szükséges portokat nyitni a működéshez. (Mindkét esetben feltétel, hogy az adott portokat más alkalmazások ne foglalják le.)

### További fejlesztési lehetőségek

A felhasználó igényein túlmenően a további funkciókkal egészíthető ki az alkalmazás.

- A program minden esetben ésszerű alapértelmezett értékekkel indul, de igény szerint elmenthető **egyedi beállítások** is részét képezhetik

- A termék tábla számos **további adatbázis mezővel** kiegészülhet, amely tárolhatja a raktározás közelebbi helyét (épület, polc, szint), vagy a termék további tulajdonságait, azonosítóit (vonalkód, lejárati dátum, stb.)

- a program a termékeken kívül bármilyen **további termék, szolgáltatás vagy erőforrás kezelésére** szolgálhat, legyen az ingatlan, pénzeszköz, vagy szállítási szolgáltatás.

- A program már most is elmenti az eladásból származó bevételt, amivel az adott napon meghatározható az esetleges **akciós eladási ár** mértéke, vagy a **kedvezményes beszerzési ár** mértéke. További mező hozzáadása esetén akár visszamenőlegesen meghatározható, hogy melyik napon vagy melyik időszakban milyen mennyiségű akciós termék került értékesítésre melyik telephelyen, ami az értékesítési stratégiát és üzleti döntéseket támogatja.

- Az adatbázis lekérdezések segítségével **egyedi statisztikai lekérdezések** valósíthatóak meg (például egy adott bolt melyik termékből értékesített legtöbbet, vagy mely termékből származott a legtöbb bevétel egy adott időszakban.

- **beszerzések és eladások egyedi azonosítói alapján** akár az is megállapítható, hogy egy adott időszakban egyes termékekből milyen bruttó nyereséget tudott realizálni a vállalat, vagy egyes kereskedelmi egységek.

- A hoszt alkalmazás adatbázishoz **csatlakozó utasításait kirendezni egy külső konfigurációs fájlba,** ahol az módosítható az üzletlánc igényei szerint.

### Összefoglalás

### Forrásjegyzék

#### \*(1)

What is the DOM? Document Object Model Meaning in JavaScript, [https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript](https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/), letöltés dátuma: 2022.03.28.

#### \*(2)

The Comprehensive Guide to React’s Virtual DOM, <https://javascript.plainenglish.io/react-the-virtual-dom-comprehensive-guide-acd19c5e327a>, letöltés dátuma: 2022.03.28.

##### \*(3)

What is MySQL?, <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>, letöltés dátuma: 2022.03.28.

###### \*(4)

Adatbáziskezelés alapjai,

<https://arato.inf.unideb.hu/bujdoso.gyongyi/kurzusok/adatbazis/ea/adb_1ea_1_Alapok_2016.pdf>, letöltés dátuma: 2022.03.28.

**\*(5)**

What Is Windows Communication Foundation , <https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf> , letöltés dátuma: 2022.03.28.