Tervezés I.

# Tartalomjegyzék

# Bevezetés

Mivel a személyes vagyon nyilvántartása pénzügyi alkalmazások nélkül hosszú és rengeteg odafigyelést igénylő folyamat, így az emberek többsége nem is veszi a fáradságot e nyomon követések elkészítésére. Tovább bonyolítja a helyzetet, ha egy adott személy, aki szeretné nyomon követni a pénzügyeit, de tegyük fel, hogy több banknál is rendelkezik fiókkal, vagy egy család tagjai nem ugyan annál a banknál rendelkeznek számlával mégis egyszerűen és gyorsan szeretnék megtekinteni az összesített pénzügyi kimutatásaikat. Az effajta problémák áthidalására jöttek létre a személyes pénzügyi alkalmazások. Természeten, ha pénzügyeink nyomon követésről beszélünk, akkor egy ilyen alkalmazás segítségével a portfólióinkat is könnyen nyomon tudunk követni.

Tehát személyes pénzügyi szoftverek elsősorban a célra lettek kifejlesztve hogy egyszerűbbé tegyék a felhasználók számára pénzügyeik nyomon követését és bizonyos funkciók nyújtásával segítsék beosztani, esetleg jobbá tenni azt. Ezek a funkciók többek között lehetnek:

* Ha a banki tranzakciókat tekintjük, például számon tarthatjuk mennyi pénzünk van jelenleg a bankszámlánkon, mennyi pénzt tudtunk félretenni bizonyos hónapokban, milyen kategóriájú termékekre költöttünk a legtöbbet, hány tranzakció volt a számlánkon bizonyos hónapokban, napokban.
* Ha tőzsdei tranzakciók nyomon követéséről beszélünk az alkalmazás segítségével meg tudjuk jeleníteni a jelenleg birtokolt részvényeket és azokhoz tartozó darabszámokat illetve a jelenlegi részvényárfolyamot is. Le tudjuk kérni a birtokolt részvényekhez tartozó jelenlegi és egy bizonyos intervallumra visszamenő legmagasabb, legalacsonyabb, nyitó és záró árakat. Valamint a felhasználó által eladott részvényekhez nyereség-veszteség kiszámolására is van lehetőség.

Ezekből az adatokból olyan következtetéseket tud levonni a felhasználó, amelyek segítik a sikeres pénzbeosztását, esetleges spórolást illetve a tőzsdei nyereségek esélyét. De elsősorban segítik a pénzügyi tudatosság kialakulását.

Bizonyos applikációk felajánlanak pénzspórolási tanácsokat is, amiket a felhasználó tranzakcióinak statisztikai elemzéseiből nyernek ki. Habár e tanácsok többsége fizetős funkció.

A legnépszerűbb pénzügyi szoftverek legfőbb tulajdonsága hogy a felhasználók egy helyen láthatják az összes bankszámlájuk tranzakcióját, költési szokásukat és egyéb kimutatásokat. Így egy átfogó képet kapva a személyes pénzügyeikről, ezáltal segítve annak nyomon követését és esetleges jobbá tételét is.

Fontos megjegyezni, hogy manapság az alkalmazások nagy része elérhető telefonos applikáció formájában is. Mivel a mai emberek azonnali elérhetőséget kívánnak a nap bármely pontjában, főként, ha a személyes pénzügyeik alakulásáról van szó. Tehát a sikeres pénzügyi alkalmazások mind elérhetőek telefonos készülékre is.

Az első fejezetben a feladat rövid leírása található, leírva a fő problémát, annak okát, illetve tisztázva azokat a funkciókat, amivel a programnak rendelkeznie kell.

A második fejezetben bemutatásra fog kerülni pár elterjedtebb pénzügyi alkalmazás, azok funkcióival, pozitív illetve negatív tulajdonságukkal egyaránt. Illetve ezen alkalmazások összehasonlítása táblázat formájában.

A harmadik fejezetben bemutatásra kerülnek a szoftver képernyőtervei, illetve az azokhoz kötött funkcionális követelményeiket is, figyelembe véve a második fejezetben olvasottakkal.

A negyedik fejezetben leírásra kerülnek a program megírása alatt használt szoftvereket és e szoftverek ismertetése, illetve hogy miért ezeket az alkalmazásokat használtam a szakdolgozatok megírásához.

Az ötödik fejezetben felsorolásra kerülnek a jövőbeli tervek, fejlesztési irányok a jelenleg meglévő szoftvert illetően.

TÖBB

# Feladat leírása

A jelenleg piacon lévő pénzügyi alkalmazások mind nagyon különbözőek, mind megjelenítésben, mint nyújtott funkciók terén. De egy közös tulajdonsággal mind rendelkeznek, méghozzá azzal hogy mind képesek feldolgozni és megjeleníteni a felhasználó által beimportált tranzakciókat. A legtöbb személyes pénzügyi alkalmazás Amerikában van jelen, így csak amerikai bankkártyák adatainak feldolgozására lettek kifejlesztve. Ennek az az oka, hogy ott minden banktól letöltött tranzakciós fájl formátuma egységes felülettel rendelkezik. Egy számlaaggregátor applikáció kifejlesztését megkönnyíti az a tény, hogy a felhasználó által beimportált fájlok banktól függetlenül ugyan azzal a formátummal rendelkeznek. Sajnos a Magyarországon jelenlévő bankoktól és tőzsdei programoktól letölthető tranzakciós fájlok felépítése nincs egységesítve. Tehát minden fájl felépítése, struktúrája, elrendezése más és más.

Célunk egy olyan importáló szoftver írása, amely képes feldolgozni bármely magyar banktól, illetve tőzsdei programból exportált számlakivonatot tartalmazó fájl adatait. Majd ezen adatokat eltárolni egy egységes formátumban lévő adatbázisba. Ezáltal kiküszöbölve a különböző formátumú, struktúrájú export fájlok problémáját. Majd ezen egységesített adatokkal dolgozva megvalósítani a személyes pénzügyi szoftverek egyéb funkcióit.

Tehát a programnak képesnek kell lennie megjeleníteni az importált adatokat egységesített táblázat formájában, valamint a felhasználónak legyen lehetősége a táblázatban megjelenített adatokon való rászűrésre bizonyos kategóriák szerint, például bank neve, tranzakció időpontja, jövedelem vagy költés, bizonyos összeg feletti költés vagy jövedelem stb..

Valamint az applikációnak fel kell tudnia dolgozni mind a vásárolt, mind az eladott részvények adatait is. Valamint egy táblázatban megjeleníteni a nyereség illetve veszteség jelenlegi állását az importált tőzsdei tranzakciókra vonatkozóan. Továbbá a szoftvernek tőzsdei tranzakciók esetén fel kell ajánlania több fajta nyereség-veszteség kiszámolási módszert is a felhasználó számára. Illetve legyen lehetősége a felhasználónak lekérni a bizonyos tőzsdei szimbólumokhoz tartozó árakat is.

## FinTech

A FinTech (pénzügyi technológia) egy fejlődésben lévő iparág, melyek hatékonyabbá tehetik a pénzügyi szektort a digitális kor innovációinak felhasználásával. Ennek következményeképpen kiszorítva azon pénzügyi szektorban jelenlévő, úgymond kevésbé informatikai szemléletben gondolkozó szereplőket.

Egyszerűen, a FinTech egy pénzügyi technológia, egy digitális forradalom. Ez nagymértékű változásról szól a vagyonkezelésről, a személyi és az üzleti kölcsönökről, a tőkegyűjtésről, a pénzátutalásokról valamint a befektetés módjáról. A FinTech magában foglalja az összes pénzügyi vállalkozás működésének megzavarását, valamint személyes pénzügyeinket. Miért kellene fizetni egy pénzügyi tanácsadónak, ha pénzügyi tanácsadást kaphatunk online, egy sokkal alacsonyabb árért?

A fintech cégek között megkülönböztetnünk két alkategóriát.

* Enabler: a jelenlegi pénzügyi rendszer résztvevőit (bankok, biztosítok stb.) támogatják, segítik annak érdekében, hogy a FinTech cégek által támasztott kihívásoknak meg tudjanak felelni, ezáltal tartani tudják a versenyt a versenytársaikkal.
* Disruptor: hagyományos pénzügyi intézmények (bankok, biztosítok stb.) versenytársainak tekinthetők.

A bankszektor alapvetően konzervatív jellege miatt sokáig ellenkezett a technológiai újításokkal szemben, de manapság már nem tudják, illetve nem is áll szándékukban ellenezni a fejlődést mivel az gyors és elsöpri azokat, akik nem tudják tartani az ütemet.

Habár, nem okoz nagy meglepetést hogy a banki szektor a leginkább szabályozott intézmények közé tartozik, tehát egy-egy újítás nehezen építhető be a fennálló jogi környezetbe. Viszont új típusú fogyasztói igények megjelenése és a gyorsan fejlődő technológia azt eredményezi, hogy a fintech iparág az egyik legnépszerűbb új technológia lett a befektetők körében (főként külföldön).

A bankok ellen szól, hogy továbbra is kézi fizikai azonosítási ellenőrzésekre van szükség a számlák megnyitásához. A FinTech cégek teljesen online nyomon követési folyamatot kínálnak, és személyes befektetési tanácsadást nyújt interneten keresztül. A nagy bankoknak nem áll szándékában online személyes tanácsadást nyújtani, mivel ez tanácsadók és az értékesítők szándékában áll.

Tehát a bankok most azzal a választással szembesülnek, hogy vagy alkalmazkodnak a digitális forradalomhoz, vagy elveszítik a piaci részesedésüket. A FinTech-ektől eltérően a bankok nem képesek megfelelni a gyorsaságnak, amit a FinTech cégek képesek nyújtani. Ennek ellenére a vállalatokkal való együttműködés lehetővé teszi a bankok számára, hogy bővüljenek és növekedjenek.

A FinTech előnyei annyira ígéretesek hogy szinte elfeledkezünk az esetleges hátrányaikról, ami nagy hiba lenne.

* Az online elérhető adatok könnyen ellophatók harmadik féltől. Más jövedelmező célokra vagy akár személyazonosság-lopáshoz is felhasználhatók

A FinTech cégek évtizedes kriptográfiai és biztonsági kutatások csúcspontjára épülnek és bár az SHA-256 titkosítást jelenleg nagyon erősnek tartják, még mindig vannak bizonyos sebezhető területek. Ami egy nagyon nagy hátrány, mivel az emberek nagyon szkeptikusak kezdetben a technológiai újításokkal szemben, főleg ha a pénzügyeikről van szó. De mentségükre szóljon, hogy veszélymentes rendszer nem létezik.

### Blokklánc technológia

Az úgy nevezett blokklánc-technológia (blockchain) elsősorban a pénzügyi válság után vált ismerté. Nagy előnye a nyílt forráskódú fejlesztés, ami által valóra váltható a tranzakciók elszámolása közvetítők bevonása nélkül. Aminek legfőbb előnye, hogy töredékére csökkennek az elszámolási idők. Tehát ami eddig napokba telt, ezt követően közel valósidőben teljesülne. Nem mellesleg költségcsökkenést is eredményez.

## FinTech Magyarországon

Magyarországon a tavalyi év (2017) szeptemberében a parlamentnek be lett nyújtva egy törvényjavaslat a bankok és a fintech-cégek számára vonatkozóan. Ha a PSD2 nevű irányelv elfogadásra kerül az a két fő dolgot jelent.

* Nagyobb versenyt a bankok számára.
* A bankolást megújító innovatív cégeknek pedig könnyebb piacra lépést.

A PSD2 irányelv elsődleges célja a bankok és fintech cégek ügyfeleinek hatékonyabb védelme valamit nem utolsó szempontban a belső piac további erősítése. A belső piac erősítése előnyös, mint a fogyasztók, mind a kereskedők és a vállalkozók számára is.

Az irányelv új típusú szolgáltatók megjelenését is lehetővé teszi. Az úgynevezett külső szolgáltatók (TTP-k) kétféle új szolgáltatást nyújthatnak: számlainformációs és fizetéskezdeményezési szolgáltatókká válhatnak. Ezekhez az ügyfelek engedélyével banki adatokat lehet lekérdezni, vagyis a fintech szolgáltatók is hozzáférést kapnak a banki rendszerekhez.

A gyakorlatban ez azt jelenti, hogy új, például a kártyatársaságokat az értékláncból kihagyó fizetési szolgáltatások jöhetnek, illetve olyan számlaaggregátor szolgáltatók, akiknél egyszerre akár több számlát is kezelni lehet. Ezáltal rengeteg felhasználót segítve számláik és egyéb tranzakció átlátásaiban és nyomon követésében.

A hazánkban jelenlévő bankoknak, biztosítóknak, vagyonvédelemi stb. szervezeteknek az MNB (Magyar Nemzeti Bank) szerint célszerű lenne minél előbb kialakítaniuk a fintech cégekkel kapcsolatos álláspontjukat. Bár a fintech cégek számtalan újdonságot próbálnak meghonosítani a pénzügyi szektorban, kiforratlan üzleti modelljeik jelentős veszélyeket is hordozhatnak a pénzügyi szektor és az ügyfelek számára is. A túlzóan megengedő szabályozói megközelítés azt idézheti elő, hogy a pénzügyi közvetítés egyre növekvő részére a hatóságoknak csak közvetett ráhatásuk lesz. Emellett olyan, rendszerszintű kockázatokkal szembesülhetnek, amelyek kezelésére nincsen kiforrott jogszabályi környezet (pl. betétbiztosítás hiánya). Ezzel szemben azokban az országokban, ahol a szabályozó hatóságok túlzottan szigorúak, a pénzügyi közvetítőrendszer könnyen veszíthet a hatékonyságából, és ezzel végső soron a versenyképességéből.

# Irodalmi áttekintés

Mivel a személyes pénzügyek a legtöbb ember számára fontosak így nagy igény van ilyen szoftverekre, főleg napjainkban, ahol minden alkalmazásnak van telefonos változata és az emberek kényelemben szeretik élni mindennapjaikat. Tehát csak pár újmozdulat segítségével láthatjuk pénzügyeinket.

Természetesen jó-pár pénzügyeket beosztó, segítő és nyomon követő applikáció létezik. Nem szokatlan az sem hogy egy felhasználó egyszerre használ több ilyen alkalmazást, mivel amit az egyik változat tud azt nem biztos, hogy a másik is. Vagy az is előfordulhat, hogy az egyik alkalmazás használata könnyen elsajátítható vagy esetleg jobban személyre szabható, jobb grafikus felülettel rendelkezik, míg a másik nem rendelkezik e tulajdonságokkal így nem olyan megnyerő a felhasználók körében.

A pénzügyi applikációk mindegyike rendelkezik egy Tranzakciók Importálása menüponttal, ahol pár kattintás segítségével új tranzakciókat tudunk felvinni a rendszerbe, valamint Táblázat menüponttal, ami tartalmazza a beimportált, már az adatbázisban lévő tranzakciókat, aminek segítségével könnyűszerrel átláthatjuk a bizonyos időközben végbement változásokat a bankszámlán(in)kon. Kijelenthetjük, hogy e két funkciói alapkövetelményének számít a pénzügyi programok számára. Illetve könnyedén előfordulhat, hogy egy család használja ugyan azt az alkalmazást és a közös kasszájukat követik nyomon programban. Tegyük fel hogy, a család minden tagja külön banknál rendelkezik számlával (ami nagyon könnyen előfordulhat). Így egy szoftvernek képesnek kell lennie több bank adatait is feldolgozni és megjeleníteni, hogy eredményesen tudjon működni. A mai felhasználóknak nagy erénye a kényelmesség és ehhez hozzátartozik a kimutatások könnyű átlátása, ezalatt azt kell érteni, hogy látványos grafikonokkal illetve színekkel várják el a fontos adatok megjelenítését. Ha egy alkalmazás nem rendelkezik e funkciókkal, nem lesz túl népszerű a felhasználók körében.

Természetesen egy pénzügyi szoftvert sikeressé tehet az általa nyújtott funkciók sokasága is. Például hogy nem csak bankoktól származó adatok, hanem tőzsdei tranzakciók beolvasására, megjelenítésére, eltárolására is képes. Valamint nyereség-veszteség kiszámolására is használható, jelenlegi árfolyamok megtekintésére. Nagyon gyakori a Pénztárca funkció is, ami segítségével a felhasználók saját kedvűkre megszabhatnak egy úgymond képzeletbeli korlátot bizonyos terméktípusokra, amit egy adott ideig (felhasználó állítja ezt is be) nem szabad túllépniük.

Habár ezen applikációk többségének kipróbálása ingyenes, természetesen léteznek csak fizetős változatok is. Fizetős funkciók között szerepelhet a számlabefizetés, szöveges értesítés telefonra, ha bizonyos tranzakciók történnek, több számla nyilvántartása stb..

Manapság a felhasználók körében a kényelmesség illetve az azonnali elérhetőség nagy erény. Tehát nem ritka az, ha egy alkalmazásnak nem is készül úgymond asztali változata, hanem csak telefonos applikációként létezik. Mivel úgy könnyebben és kényelmesebben elérhető a felhasználó számára.

Tehát rengeteg apró dolog tehet egy pénzügyi alkalmazás sikeressé, avagy kevésbé népszerűvé. A következőekben négy darab szoftvert mutatok be azok pozitív illetve negatív tulajdonságait is leírva.

## Mint.com

A Mint.com egy olyan alkalmazás, amely többek között segít beosztani a pénzünket, emlékeztet a számlák befizetésére, sőt egy régi frissítésnek köszönhetően, szigorúan csak a felhasználó engedélyével, már be is fizetheti helyettünk is azokat. Több bankkártya adatait is tudja tárolni és nyomon követni. Több lépcsős beléptető rendszerrel rendelkezik, tehát a biztonságra természetesen nagyon ügyelnek. Csak a számlaforgalmat látja, azt kezelni nem tudja (számlabefizetésen kívül, ha a felhasználó igény tart erre a funkcióra).

Mikor regisztrál egy új ügyfél és megadja a bankkártyája adatait a szoftver beolvassa a tranzakciókat és egy algoritmus segítségével megvizsgálja a felhasználó korábbi tranzakcióit, ezáltal megtudva költési szokásait. Ezáltal adva esetleges spórolási illetve költési tanácsokat a felhasználó számára. Tehát a felhasználó könnyen nyomon tudja követni a fejlődését, illetve a szoftver pontozza is azt, ezzel buzdítva a felhasználót.

A Mint.com elérhető telefonos változatként is. Sajnos a magyar bankok kezelésére nem képes, csak amerikai bankkártyákat ismer fel, viszont a PayPal fiókokat támogatja, ami Magyarországon is elérhető. A Mint.com rendelkezik az egyik legjobb osztályzással a felhasználók körében. Amit a barátságos kezelőfelületének, könnyen használhatóságának és főként a sokrétű pénzügyi funkciók nyújtásáért érdemelt ki. Valamint az interneten rengetek oktató videót találhatunk a program kezeléséről. A Mint.com képes tőzsdei tranzakciók megjelenítésére, nyomon követésére is. Valamint összehasonlíthatjuk a jelenleg birtokolt részvényein vételi árát a jelenlegi tőzsdei árfolyammal. Így segítve az esetleges nyereséges eladást.



Ábra: Mint.com kezdőoldala.

Az ábrán látható menüpont a Mint.com kezdőoldala, ahogy láthatjuk a szoftver ezen menüpontja tartalmazza az éppen aktuális Pénztárca állását figyeljük meg ahogy a program kiírja hogy mennyi pénzt költhetünk még az adott kategóriában, illetve ha túlköltöttünk akkor pirosra színezve azt, így felhívva a felhasználó figyelmét a költési korlát átlépésére, természetesen láthatjuk azt is hogy mennyivel haladtuk túl a képzeletbeli keretünket.

Továbbá láthatjuk az esedékes számlabefizetések dátumát (négy hétre előretekintve) illetve azok pénzösszegeit is megjelenítve. Valamint az elmúlt fél év statisztikáit jövedelem szempontjából. Nem utolsó sorban láthatjuk az osztályzást, amit a rendszer ad nekünk. A kapott osztályzat függ a többi felhasználó által elért eredményektől, mivel a rendszer átlagolja a spórolást, megtakarást, pénztárca keretek betartását is.

**Pozitív tulajdonságok**

* Könnyen személyre szabható.
* Könnyen átlátható, rengeteg oktató videó található az interneten.
* Automatizált rendszer, pár kattintás segítségével importálhatunk, láthatjuk tranzakcióinkat.
* Többlépcsős beléptető rendszerrel rendelkezik.
* Ingyenes.
* Az osztályzások alapján a Mint.com az egyik legjobb és legnépszerűbb személyes pénzügyi alkalmazás.
* Rendelkezik tőzsdei tranzakció importálás funkcióval is, megnézhetjük a jelenlegi portfóliónk helyzetét is
* Létezik telefonos alkalmazás formájában is.

**Negatív tulajdonságok**

* Csak az Amerikai bankkártyák adatait tudja kezelni.
* Ha fizetünk bizonyos összeget, akkor tanácsokat kaphatunk, hogyan spóroljunk több pénzt.

## YNAB (You Need A Budget)

A YNAB a nevéhez hűen, „csak” egy pénztárca létrehozásához és kezeléséhez jó. De arra tökéletesen megfelel feltéve, ha az ember elég időt szán az alkalmazás megismerésére. A program regisztrálástól számítva 34 napig ingyenes ezen idő letelte után havi 5 dollár a felhasználási díja. Barátságos, könnyen átlátható kezelőfelülettel rendelkezik. Főként telefonos applikációként tervezték meg, de rendelkezik asztali alkalmazásként is.



Ábra: a YNAB kezdőoldala.

Az ábrán láthatjuk a felhasználó által létrehozott Pénztárca kategóriákat. Vegyük, észre hogy bal oldalt láthatjuk a fontosabb adatokat:

* Jelenleg rendelkezünk 3,362.01 $-al
  + Amiből jövedelem 1,740.00 $
  + A spórolt pénzösszeg összesen 5,247.23 $
  + A tartozásunk (hitelkártyán) összesen $3,625.22$

Illetve láthatjuk a pénztárcánk maximális korlátait, az ebből már elköltött pénzösszeget illetve kategóriánként az aktivitásunkat az elmúlt hónapban. A rendszer első induláskor már rendelkezik kategóriákkal, de természetesen ezeket lehet bővíteni is

Észrevehetjük, hogy a szoftver felajánlja a több bankszámla kezelését is, ami például egy családi kassza kezeléséhez elengedhetetlen funkció.

Sajnos, ez az alkalmazás inkább csak a költésre fokuszál és nem jelenik meg a jövedelmek kezelése ilyen mértékben. Habár a felhasználók véleménye szerint a szoftver hatékony használatának megtanulása sok időt emészthet fel, segítségünkre lehetnek az útmutató videók, amiből az interneten rengeteg található. A tranzakciókat a felhasználók importálják, majd később kategorizálhatják is azokat. A tranzakciók megfelelő kategóriába besorolása után a rendszer kiszámolja a pénztárca és azon korlátainak aktuális állapotát. Amikor először használjuk az applikációt meg kell adnunk mennyi pénzel rendelkezünk jelenleg, hogy a szoftvernek legyen egy alapösszeg amiből tud számolni megtakarítást. A YNAB a pénzünk jelenlegi helyzetével foglalkozik, és nem tekint előre esetleges jövőbeli tranzakciókra (a Mint.com-al ellentétben). Valamint nem rendelkezik tőzsdei tranzakciók importálásának lehetőségével.

**Pozitív tulajdonságok**

* Személyes „pénztárca” alkalmazás.
* Jó felhasználói felület.
* Sok oktató irodalom található róla.

**Negatív tulajdonságok**

* Sok idő míg a felhasználó meg tanulja használni a funkciókat.
* Nincs többlépcsős belépés.
* Drágább, mint a konkurens alkalmazások (34 napig ingyenes).
* Nincs nyomtatás funkció.
* Főként költésre van kiélezve.
* Nem lehet tőzsdei tranzakciók áttekintésére használni

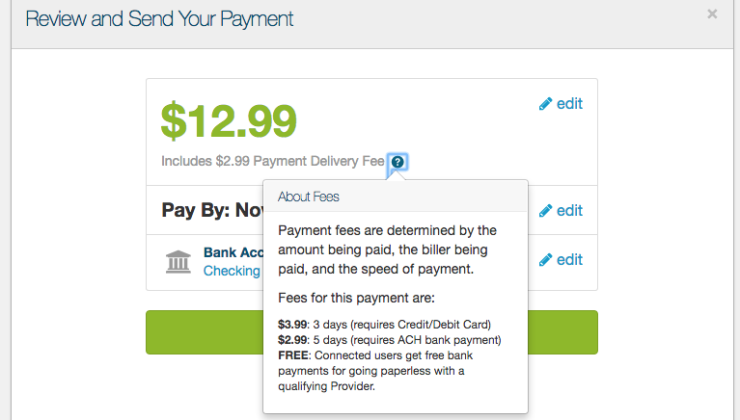
## Doxo

A Doxo egy olyan pénzügyi alkalmazás, ami főként a számla befizetésre és a banki számlával kapcsolatos dolgok tárolására van kiélezve. Egy kifejezetten jó alkalmazás azon emberek számára, akik nem szeretnének többé papír alapú tranzakciós dokumentumokat, számlákat tárolni.

Habár az alkalmazás használata ingyenes, rendelkezik fizetős funkciókkal is pl. számlabefizetés. A felhasználó szoftverben tárolt adatai automatikusan frissülnek számlaforgalom esetén. Automatikusan importál tranzakciókat és egyéb dokumentumokat, amik kapcsolódnak bankszámláinkhoz.

Az következő ábrán láthatjuk a Doxo egyik meghatározó funkcióját a számla befizetést, ami egy nagyon hasznos funkció, képzeljük el milyen kényelmes így csekket befizetni, kattintunk párat és már kész is. Míg az alkalmazás használata ingyenes, a számlabefizetés funkcióért befizetésenként különböző összeget kell fizetni.

Sajnos amilyen jól hangzik a számlabefizetés funkció, annyira rosszul is működik a Doxo alkalmazáson keresztül. Mivel nagyon lassan megy végbe a tranzakció, esetleges lefagyások is bekövetkezhetnek ami azt eredményezheti hogy nem megy végbe az átutalás. Illetve az alkalmazáson keresztül való befizetésért számláktól függően külön kezelési költséget számol fel az alkalmazás. Valamint a kezelési költség a számla befizetésének végbemenetele után jelenik meg.



Ábra: Doxo számla befizetés.

**Pozitív tulajdonságok**

* Rendelkezik számlafizetési opcióval.
* A használata nagy részben ingyenes.
* Nyomtatás opcióval is rendelkezik.
* Tud tárolni blokkokat és egyéb dokumentumokat amik a bankszámláinkhoz tartoznak.

**Negatív Tulajdonságok**

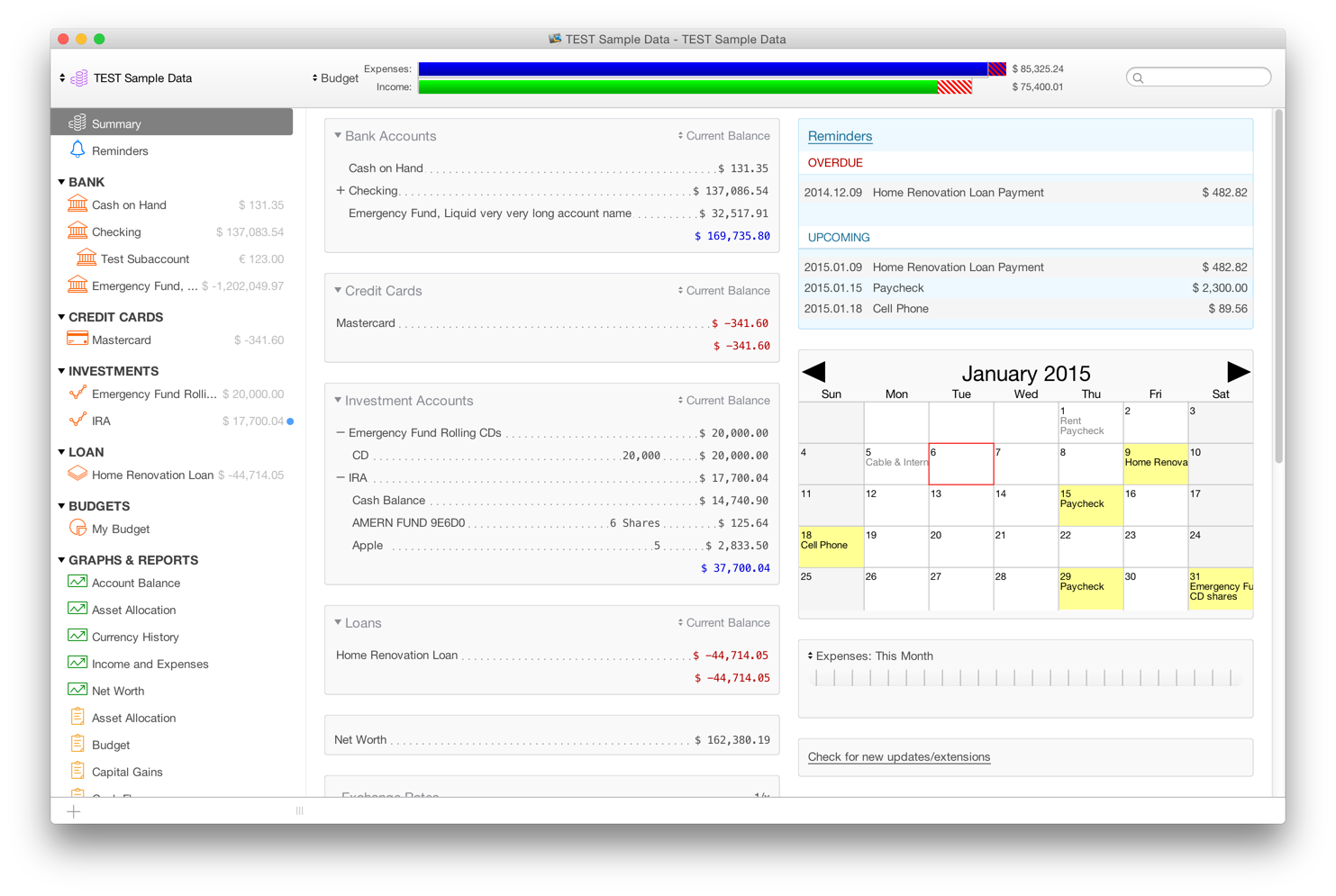
* Lassú lehet bizonyos funkciók esetén például számla befizetés, tranzakciók importálás (bank függő).
* Sok negatív vélemény (felhasználók befizették a számláikat, de a pénz sosem érkezett meg).
* Nem ismer fel bizonyos bankkártyákat.

## Quicken Premier

A Quicken Premier az egyik legrégebb óta piacon lévő pénzügyi alkalmazás. Több mint harminc öt éve jelent meg. Legfőképp azon felhasználók számára ajánlott, akik már ismeretesek egyéb számla menedzselős programokkal és tudják mit keresnek. Az alkalmazásnak nagy előnye az automatizálás, például automatikusan letölti tranzakcióinkat, melyeket utána automatikusan kategorizál. A többi alkalmazáshoz hasonlóan mutatja a befizetéseket, a számla jelenlegi állását, pénztárca és azon korlátainak jelenlegi helyzetét.

Habár a felület talán kicsit barátságtalanabbnak tűnhet a többi piacon lévő alkalmazáshoz képest, több funkcióit, megjelenítési módot is tartalmaz, mint konkurens társai. Sajnos ennek az alkalmazásnak nincs ingyenes változata, éves díja 80 $. Az alkalmazás különböző diagramokkal segíti az átláthatóságot.

Valamint rendelkezik kiforrott befektetés követéssel is, ami csak nagyon ritkán található meg személyes pénzügyi alkalmazásokban mint funkció. Eredetileg csak személyes számítógépre tervezték. De ma természetesen már elérhető telefonra is.



Ábra: Quicken Premier kezdőoldala.

A fenti ábrán láthatjuk a Quicken Premier kezdőoldalát és annak sokrétű információ megjelenítő részeit. Ahol többek közt láthatjuk az összes kiadásainkat illetve bevételeinket, befektetéseink jelenlegi helyzetét. A kiválasztott hónapban várható számlabefizetéseket, valamint ezen számlák árait. A fizetésünk utalásának várható időpontját és egyéb számlánkat érintő eseményeket.

(Az ábrán látható adatok szerint 15.-én fog megjönni a fizetésünk és 18.-án kell befizetnünk a telefonszámlánkat. Valamint baloldalon láthatjuk milyen sokrétű grafikont megjelenítését kínálja fel a rendszer a felhasználó számára, például jelenlegi számla egyenleg, bevétel és kiadások, valutatörténet, importált számlák nettó értéke).

A Quicken Premier felülete kezdetben kicsit barátságtalannak tűnhet és a felhasználónak és jó pár órát rá kell szánnia az alkalmazás megismerésére, illetve használatának megtanulására.

A következő ábrán láthatjuk a Quicken Premier egyik fő funkcióját a befektetés kezelés terén. Láthatjuk hogy a felhasználó rendelkezik 5 darab Apple részvénnyel, amit 2012.03.31.-én vásárolt 42,5$ értékben, összesen 212,50$ ért.

A jelenleg birtokolt részvény mellet jobbra pedig láthatjuk az Apple részvények aktuális eladási árát. Valamint egy felhasználót segítő kimutatást, hogy 1233,41%-al többet ér jelenleg az öt darab Apple részvényünk, mint mikor vásároltuk őket.



Ábra: Quicken Premier befektetés követő funkció.

**Pozitív tulajdonságok**

* Rendelkezik befektetés követő funkcióval.
* Gyakran jönnek ki új verziói.
* Rendelkezik az összes személyes pénzügyeket segítő és nyomon követő funkcióval. Mind a banki és tőzsdei tranzakciók szempontjából.
* Elsősorban asztali alkalmazásnak tervezték, de tudjuk irányítani a fiókunkat telefonról is.
* Az egyik legrégebb óta piacon lévő személyes pénzügyi alkalmazás.

**Negatív tulajdonságok**

* Nincs ingyenes változata
* Nem kimondottan kezdők számára fejlesztették ki.
* Az új felhasználók számára kezdetben nehezen átlátható

## Összehasonlító táblázat

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Alkalmazás neve | Univerzális importálás | Telefonos változat | Nyomtatás funkció | Fizetős funkció | Befektetések kezelése |
| Mint.com | + | + | + | Tanácsok | + |
| YNAB | + | + | - | csak 34 napig ingyenes használata | - |
| Doxo | + | + | - | számla befizetés | - |
| Quicken Premier. | + | + | + | éves díja 80$ | + |

# Személyes pénzügyi importáló tervezése

A szoftver tervezési fázisának elején fontos döntéseket kellet meghoznom. Természetesen elsősorban hogy milyen platformra tervezem létrehozni a szoftvert, illetve hogy milyen programozási nyelvet válasszak a fejlesztés közbeni bizonyos problémák hatásos megoldásához. Mivel célom volt a platformfüggetlenség. Valamint egy könnyen változtatható, látványos grafikus felület létrehozása ezért a C# programnyelven való fejlesztést választottam. Így csupán NET keretrendszerre van szükség a célszámítógépen, ami pedig rendelkezésre áll a felhasználók által leggyakrabban használt operációs rendszerek mindegyikén (Windows, Linux, Mac).

Illetve a C# platform előnyei között szerepel:

* Egy magas szintű programozási nyelv, amely megkönnyíti a programozó dolgát bizonyos problémák megoldásában, illetve program struktúrájának átlátásában.
* Mivel adatbázist szerettem volna használni mindenképp szükség volt egy programozási nyelvre, amely támogatja a többszálúságot.
* Népszerű programozási nyelv lévén, a C# rengeteg ingyenes ,,könyvtárral" rendelkezik, amik szintén megegyszerűsíthetik a programozói munkát.
* Ugyancsak a népszerűséghez tartozik, hogy rengeteg oktató illetve ismeretterjesztő videó található az interneten.

Valamint az sem volt az utolsó szempont hogy egy eddig ismeretlen programozási nyelvet tanuljak és sajátítsak el a szakdolgozatom megírása által.

## Használt technológiák

### .NET keretrendszer

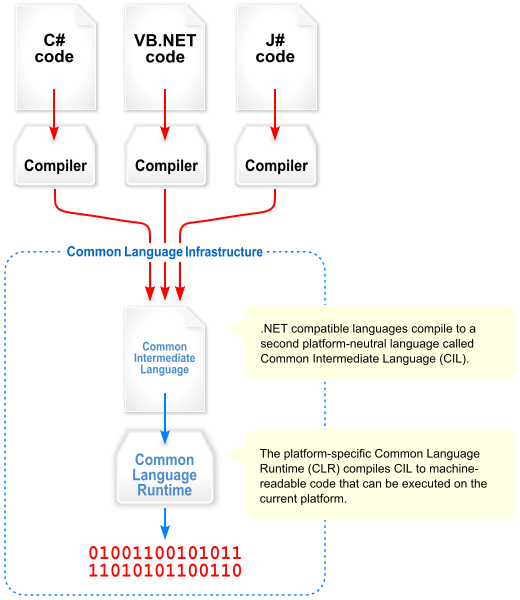
Habár a .NET keretrendszer nem új technológia, azonban a felhasználók közül nem sokan tudják, mire jó, vagy mire használatos. Ezen rövid leírásomban bemutatom, hogy mire is jó. Ezen felül szót fogok ejteni olyan témákról is, amelyek a járatosabb felhasználók számára sem egyértelműek.

A Microsoft .NET rendszere többféle programozási nyelven is programozható - feltéve, ha az adott programozási nyelvhez van olyan fordítóprogram, amely alkalmazza a Common Language Specification elveit és ajánlásait, valamint képes a .NET keretrendszer virtuális kódjára fordítani. Sok ilyen programozási nyelv létezik. Például a Microsoft által direkt támogatott nyelvek a Visual Basic, J# , C#, Visual C++.

(A Common Language Specification olyan szabályok halmaza, melyek leírják azokat az alapvető normákat, amelyre a programozási nyelveknek illeszkedniük kell.)

Mivel a .NET keretrendszerben mindegy, hogy milyen nyelven programozunk, az egyik programozási nyelven megírt eljárást a másik nyelvből is meg lehet hívni és lehet használni. Fontos hogy ez a funkció csak abban az esetben valósítható meg, ha a nyelvek egymással is megegyeznek bizonyos alapvető elvekben. Ilyen elvek közé tartozik a tömbök és rekordok, ábrázolása és használata és az alapvető típusok (Integer, Double, String, stb.) reprezentációja a memóriában.

A Common Language Runtime (CLR) egy végrehajtási környezet. Ami egy rétegként működik az operációs rendszer és a .NET nyelveken írt alkalmazások között, (azon nyelvek, melyek megfelelnek a Common Language Specification (CLS) szabványnak). A Common Language Runtime (CLR) fő funkciója a kezelt kód natív kóddá való átalakítása, majd a program végrehajtása. A kezelt kód csak akkor lesz összeállítva, amikor szükség van rá, vagyis az egyes funkciók hívásakor átalakítja a kellő utasításokat. A CLR tartalmaz egy úgynevezett Just In Time Compilert is, amely a futó programot ,,röptében" a memóriában átfordítja az adott számítógép mikroprocesszorának gépi kódjára A program végrehajtása során a CLR a memóriát, szálkezelés, memória felszabadítást (Garbage Collection), kivételkezelést, jogosultsági rendszert kezelést és egyéb rendszerszolgáltatásokat kezel.



Ábra: a .NET programkódok fordítási folyamata.

### C# programozási nyelv

A C# az a programozási nyelv, ami a legközvetlenebb módon tükrözi az alatta működő, minden .NET keretrendszeren futtatható programot. A nyelv adattípusai az objektumok. Automatikus szemétgyűjtést használ, valamint az absztrakcióinak többsége (osztályok, interfészek, kivételek stb.) a .NET futtatórendszert használja közvetlen módon.

A C# több korlátozást és továbbfejlesztést is tartalmaz a C és C++ nyelvekhez képest. Például:

* A mutatók és a nem ellenőrzött aritmetika csak egy speciális, nem biztonságos módban használhatók. A legtöbb objektum-hozzáférés csak biztonságos hivatkozásokon keresztül tehető meg.
* Az objektumokat a programozó nem szabadíthatja fel közvetlen módon, ehelyett az automatikus szemétgyűjtő szabadítja fel őket, mikor már nincs az adott objektumokra hivatkozás (C, C++ nyelvek egyik legvitatottabb hiányossága). Ez a módszer által teljesen megszűnik a nem létező objektumokra való hivatkozás lehetősége. Amivel programlefagyások illetve egyéb visszafordíthatatlan hibák bekövetkezése is elkerülhető.
* A C++ hoz képest a C# csak egyszeres öröklődést támogat. Viszont egy osztály több interfészt is megvalósíthat
* A C# sokkal típus biztosabb mint a C++. Az egyetlen implicit konverzió a biztonságos konverzió. Úgy, mint az egészek tágabb intervallumba konvertálása vagy a leszármazott osztályok alaposztályba konvertálása. Nem lehetséges az implicit konverzió a logikai (boolean) és az egész típusok között, vagy a felsorolás tagok és az egészek között. Valamint a C# nem rendelkezik void mutatókkal (habár az Object osztályra mutató mutatók elég közel állnak hozzájuk), valamint bármely, felhasználó által definiált implicit konverziót explicit módon meg kell jelölni.

Mivel a C# kétféle grafikus felhasználói felület létrehozását is támogatja:

* WPF (Windows Presentation Foundation)
* WinForm (Windows From Designer

Itt is megkellet vizsgálnom hogy melyik felületet választásával tudnám elérni a legjobb grafikus felületet létrehozását a szakdolgozatom írása közben. Mivel a felhasználói felület az egyik legfontosabb dolog egy szoftver tekintetében.

### WPF (Windows Presentation Foundation) vagy WinForm (Windows From Designer)

C# nyelvi környezetben használhatunk Windows Form Designer (röviden WinForm) vagy Windows Presentation Foundation (röviden WPF) dizájnereket hogy gyorsan és könnyedén készíthessünk grafikus felületet.

A fő különbség a két felület között hogy a WinForm csak egy felületet biztosít az alapvető képernyő elemek fölé (például egy Szövegdoboz). Viszont a WPF et használva felépíthetjük akár a semmiből saját készítésű képernyőeleminket. és csak nagyon kevés helyzetben függ az alapvető képernyőelemektől. Egy jó példa erre a különbségre egy Gomb elem készítése, amiben van egy kép és egy TextBox is, ez nem egy beépített képernyőelem, így a WinForm nem ajánlja fel nekünk ezt a lehetőséget egyből. Tehát nekünk kell implementálnunk az ezt megvalósító metódust, ami a következő lépésekből áll:

* Meg kell szerkesztenünk a képet.
* Létre kell hoznunk egy olyan gombot, ami támogatja a képek megjelenítését.
* Vagy importálunk kell egy harmadik féltől származó kiegészítést.

Ellenben a WPF felületen egy Gomb komponens tartalmazhat bármit, mivel ez az elem végülis egy keret, ami bizonyos állapotokkal rendelkezhet (például érintetlen, kattintott, letiltott, egér alatt helyezkedik el).

Ez a tulajdonság nem csak a Gombról mondható el, hanem az összes WPF-ben létrehozható elemről is. Ha az akarjuk megoldani WPF felületen, hogy egy Gomb tartalmazzon egy képet és egy TextBoxot nincs más dolgunk, mint a Gomb tartalmába beilleszteni ezeket az elemeket. A WinForm ilyen szintű kényelmet nem nyújt a fejlesztők számára. Viszont ennek a kötetlenségnek és a semmiből való ,,felépítésnek" az a hátránya hogy bizonyos esetekben több időt kell fordítanunk bizonyos elemek létrehozására, hogy ugyan azt az eredményt elérjük, amit esetleg könnyű megvalósítani a WinForm alapértelmezett képernyőelemek által korlátozott elemein.

#### XAML

Az XAML (eXtensible Application Markup Language) az XML nyelv egy változata, amit a Microsoft fejlesztett ki a GUI (Grafikus Felhasználó Felület) leírására WPF megjelenítési felületre. A XAML-t jelentős mértékben használják a .NET keretrendszerben, ezen belül különösképpen a WPF esetén. A WinForm vagy korábbi GUI részei ugyan azon a nyelven íródtak, mint maga a program, ami a megjelenítés mögött van például C#,VB.NET és a dizájnerek felügyelik. Viszont az XAML esetén külön van választva a megjelenítés és a programkód, ami megkönnyíti a programozók munkáját.

Mivel lehetőség van elkülöníteni a grafikus felület kódját a program belső kódjától akár munka elosztásra is van lehetőség dizájnerek és programozók között. Fontos megjegyezni programban lévő adatkötéseket, adatelemeket, eseményeket és más szolgáltatásokat XAML-len keresztül is lehet definiálni.

#### Databinding

A Databinding egy olyan technológia, ami veszi az tulajdonság forrását az adat küldőjétől, valamint a vevőjétől és szinkronizálja őket (egy ilyen tulajdonság lehet a Szövegdoboz képernyőelem Szöveg tulajdonsága, vagy a Legördülő lista aktuálisan kiválasztott eleme). A WPF (Windows Presentation Foundation) felületen a Databinding egyszerű és konzisztens útja az adatok megjelenítésének és változtatásának.



Ábra: Databinding fajtái WPF esetén.

* Egyszeri: (Az ábrán nincs illusztrálva) Az adat a forrás irányból megváltozhat, de a későbbi változtatások nem jutnak vissza a forráshoz. Ami azt jelenti, hogy ha a cél adat megváltozik, arról a forrás nem fog tudomást szerezni. Ez a módszer akkor használatos, amikor szükségünk van a jelenlegi állapotra vagy az adatok ténylegesen statikusak.
* Egyszeri (forrás felé): Az Egyszeri összekötés fordítottja, tehát akkor frissíti a forrás értéket, amikor a ,,cél" érték megváltozik. Akkor használatos, ha egyszer kell frissítenünk a meglévő cél értékét a felhasználói felületről egy új értékre.
* Egy irányú: (Az ábrán OneWay) Az egy irányú összerendelés a cél érték automatikus megváltoztatását eredményezi, abban az esetben, ha megváltoztatjuk a forrás adatot, de a célobjektum tulajdonságainak módosítása nem hat vissza a forrás értékre. Ez a fajta adat összekötés akkor használatos, ha a változtatott adat a felhasználói felületen csak olvasható. Például kötődhet egy olyan forráshoz, mint például egy tőzsdei figyelő vagy esetleg a célobjektumhoz vagy például az asztal háttérszínét vezérlő adatok. Ha nem szükséges figyelni a céladat változásait, akkor az Egyszeri kötési mód használatával elkerülhető a Kétirányú kötési mód túlterhelése.
* Kétirányú: (Az ábrán TwoWay) a Kétirányú összerendelés megváltoztatja a forrásadatot vagy a céladatot a másik frissítése esetén. Tehát ha a kódban változtatunk egy tulajdonságot, az kihatással lesz a felhasználói felületen megjelenő elem tulajdonságára is, valamint ugyan ez fordítva is igaz. Ez a fajta kötés akkor használatos, ha szerkeszthető formák vagy egyéb, teljesen interaktív felhasználói felületen elhelyezkedő elemek tulajdonságait szeretnénk kezelni. A legtöbb tulajdonsághoz alapértelmezetten az Egyirányú kötést rendeljük, de bizonyos függőségi tulajdonságok (jellemzően a felhasználó által szerkeszthető vezérlők, például a Szövegdoboz szöveg tulajdonsága és a CheckBox, IsChecked" tulajdonsága) alapértelmezetten Kétirányú összerendelést követel meg

#### Összehasonlítás

WPF előnyei:

* Újabb, tehát jobban szinkronban van az új technológiákkal.
* Rugalmas, több dolgot csinálhatunk anélkül, hogy egyéb, harmadik féltől származó kiegészítőket implementálnunk kellene.
* Ha harmadik féltől származó elemet importálunk, nagy valószínűséggel támogatni fogja mivel újabb technológia, mint a WinForm.
* XAML segítségével könnyen el tudjuk különíteni a megjelenítését és a programozás részét a programunknak.
* Rendelkezik Dabinding-el, ami segítségével össze tudjuk kötni a programkódunkban lévő változókat a grafikus felületen megjelenített adatokkal.
* Hardveres gyorsítást használ a GUI kirajzolására, jobb teljesítmény.
* Lehetővé teszi a Felhasználó Felületek készítését Windowsos szoftverekre és Web applikációkra is (Silverlight/XBAP).

WinForm előnyei:

* Régebbi technológia, ezáltal többet tesztelték és próbálták.
* Sok harmadik féltől származó kiegészítés létezik, amit ingyen letölthetünk.
* A Visual Studio jelenleg is a WinForm-ot támogatja inkább. Tehát WPF-ben többet kell dolgozni bizonyos esetekben ugyan azért az eredményért.

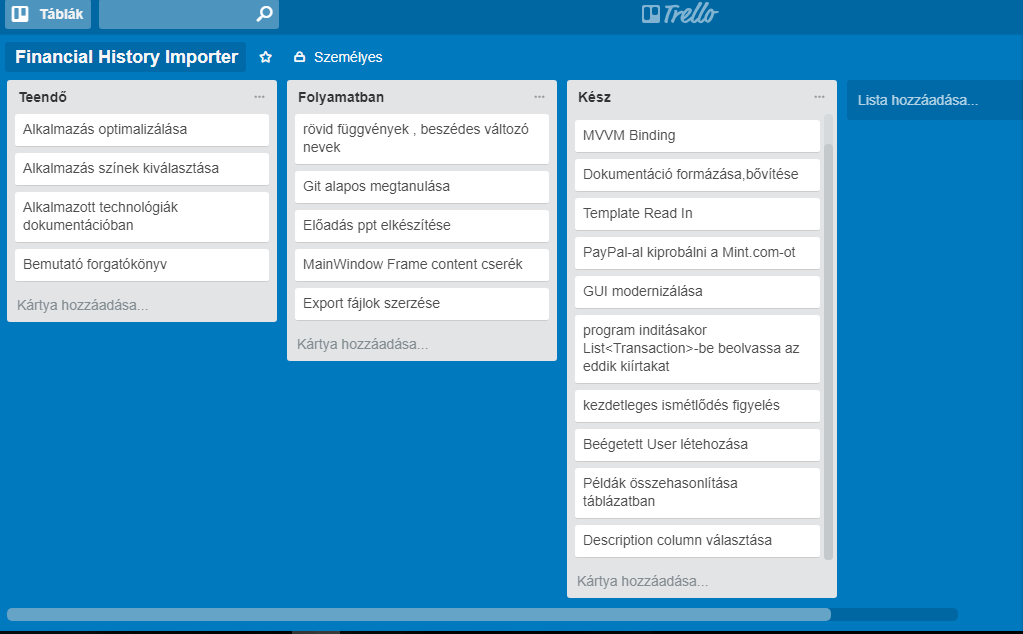
### Trello

A Trello egy web alapú projektmenedzsment alkalmazás, aminek segítségével nyomon tudjuk követni, hogy állunk a számunkra kitűzött célok elérésében. A Trello alapvetően egy weboldal, amely az oldalon vízszintesen felsorolt ​​listákat tartalmaz, így egy madártávlatos képet kaphatunk a projektről. A listákon belüli, kártyákkal ellátott elemek húzhatók és más listákra áthúzhatók vagy listákon belül rendezhetők.

Az egyes kártyák magukban tartalmazhatnak ellenőrző listákat, képeket, csatolmányokat, határidő dátumokat, színes címkéket és vitafelvételeket másoktól, akik osztoznak a projecten. Annyi sajátkészítésű táblát csinálhatunk, amennyit csak szeretnénk. Az egyik például "Otthoni teendők" a másik "Szakdolgozat" és így tovább.

Trello kártyák olyanok, mint a ragadós jegyzetek, amelyeket egy parafa fedélzeten rendeznek. Azaz olyan digitális ragadós jegyzetek, amelyek kereshetők, megoszthatók, és emlékeztetők.

Ha több fős a project csapattal rendelkezünk, akkor különböző emberekhez rendelhetünk bizonyos részfeladatokat, erről ők is értesítést kapnak email-en. Ha kész vannak a rájuk kiosztott részfeladatokkal áthúzhatják a kártyát és áthúzhatják azt a ,,Kész" oszlopba, így jelezve a többiekkel hogy elkészültek a rájuk kibaszott feladattal..



Ábra: Trello folyamatok állapotának áttekintése.

### Verziókövetés - GitHub

A GitHub megértéséhez először meg kell értenünk a Git-et. A Git egy nyílt forrású verzióvezérlő rendszer, amelyet Linus Trovalds indított. Ugyanaz a személy, aki a Linux operációs rendszer létrehozását is elindította. A Git hasonlít a többi verzióvezérlő rendszerhez - a Subversion, a CVS és a Mercurial, hogy csak néhányat említsünk.

Tehát a Git egy verziókezelő rendszer, de mit jelent ez? Amikor a fejlesztők valaminek a létrehozásán dolgoznak (például egy alkalmazás), folyamatosan módosítják a programkódot, új verziók kiadásától az első hivatalos kiadásáig és természetesen a kiadás utána is.

A verzióellenőrző rendszerek ezeket a módosításokat nyomon követik, és ezen módosításokat egy központi adattárban tárolják. Ez lehetővé teszi a fejlesztők számára, hogy könnyedén együtt működjenek. Letölthetik a szoftver új verzióját az módosíthatják azt, majd feltölthetik a legújabb verziót. Minden fejlesztő láthatja ezeket a változásokat, letöltheti őket és esetleg hozzá is járulhat a kód megváltoztatásához.

Hasonlóképpen azok az emberek, akiknek semmi köze a projekt fejlesztéséhez tehát úgymond kívülállónak számítanak, de joguk van megtekinteni a projectet, letölthetik és felhasználhatják a fájlokat. A legtöbb Linux-felhasználónak ismernie kell ezt a folyamatot, mivel a Git, a Subversion vagy más hasonló módszer használata elég gyakori a szükséges fájlok letöltéséhez. Különösen egy program forráskódból történő összeállításának előkészítésében.

A Git a legtöbb fejlesztő számára előnyben részesített verzióvezérlő rendszer, mivel számos előnye van a többi rendelkezésre álló rendszerhez képest. Fájlváltozásokat hatékonyabban tárolja, és jobb fájlintegritást biztosít a felhasználói számára.

Megállapítottuk, hogy a Git verziókezelő rendszer hasonló, de jobb, mint a rendelkezésre álló alternatívák. Tehát mi teszi a GitHub-ot annyira különlegessé? A Git egy parancssori eszköz, de a központ, amely körül minden olyan dolgot, amely a Git-et forgatja, az a hub-GitHub.com, ahol a fejlesztők hasonlóan gondolkodó emberekkel tárolják projektjeiket és hálózatukat.

### Pencil Project

A Pencil nevű mockup lehetővé teszi, hogy gyorsan és egyszerűen hozzunk létre szórakoztató és interaktív grafikus felhasználói felületet, maketteket. Az alkalmazás rengeteg beépített elemmel rendelkezik és nagyon könnyű kezelni. Valamint egy egyszerű ,,behúzós" felülettel rendelkezik.

Mivel cross-platform és teljes offline támogatást nyújt, bárhol létrehozhatunk maketteket. Nagy előnye ennek a szoftvernek, hogy teljesen ingyenes. A Pencil természetesen engedi, hogy saját képeinket illetve egyéb elemeinket is beimportáljuk. Letölthető asztali felületre, de használható böngészőn keresztül is. Habár a telepített verziósokkal több képernyő elemmel rendelkezik mint a böngészőn keresztül használt változata.

## Követelmények

A következő követelményeket állítottam fel a programmal szemben

### Alapvető követelmények

A program támogassa mind az Excel, valamint a CSV kiterjesztésű fájlok importálását is. A felhasználóknak legyen lehetősége fájlok egyszeri illetve többszörös importálására. Ezalatt az értendő, hogy az import fájl kiválasztásra során a control billentyű lenyomásával több fájl is kijelölhető egyszerre, majd a szoftver azokat sorrendben beolvassa egyesével. Illetve a felhasználóknak legyen lehetősége a grafikus felületre való ,,behúzással" (Drag and Drop) is importálni a fájlokat.

A felhasználó számára bejelentkezés után jelenjen meg egy statisztikát tartalmazó rész ahol láthatja az általa beimportált fájlok számát, importálás történetét, importált tranzakciók bankokra, bankszámlaszámra lebontva stb.

Egy felhasználóhoz tartozhasson több számlaszám is a banki tranzakciók esetén. Így segítve azon felhasználókat, akik több banknál is rendelkeznek fiókkal.

Banki tranzakciók esetén egy beolvasott tranzakció csak abban az esetben lesz eltárolva és megfelelő formájú adatnak tekintve, hogyha legalább az alábbiakban felsorolt információk szerepelnek benne:

* Tranzakció dátuma
* Tranzakció ára
* Tranzakciós fájlhoz tartozó bankszámlaszám

Ezen felül az alábbi adatok kiolvasására és eltárolására fog lehetőséget kínálni az alkalmazás:

* Tranzakcióhoz tartozó leírás
* Számlaegyenleg

Fontos megjegyezni, hogy több cella tartalma is megfelelhet a tranzakció leírásaként. Ezt orvosolva, ha több leírás cellával is rendelkezünk, azokat mind el kell tárolni a rendszerben, az adott cella tartalmát pedig vesszővel elválasztva a többi cella tartalmától.

A tőzsdei tranzakciók esetén egy beolvasott tranzakció csak abban az esetben lesz eltárolva és megfelelő formájú adatnak tekintve, hogyha legalább az alábbiakban felsorolt információk szerepelnek benne:

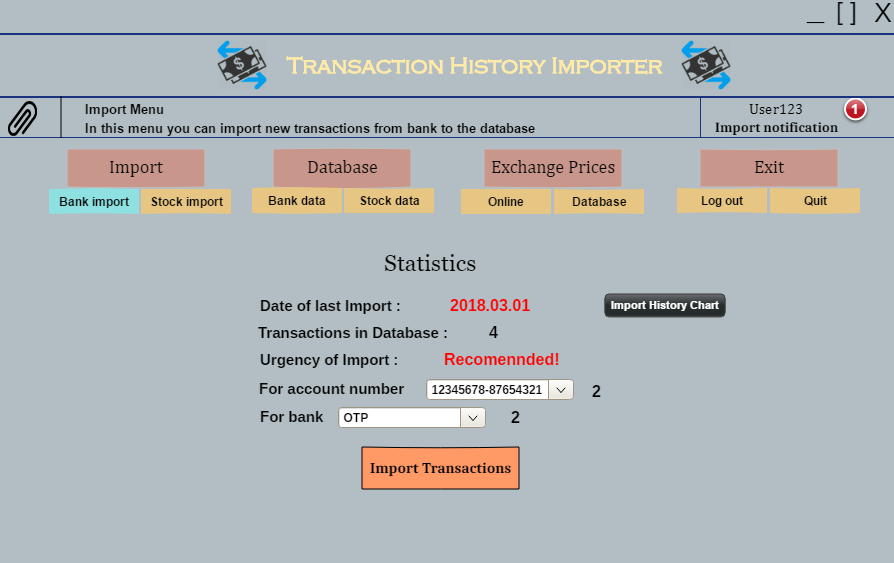
* Tranzakció dátuma
* Termék neve (Vásárolt/Eladott részvény neve)
* Tranzakció ára
* Tranzakció típusa (Eladás vagy Vásárlás)
* Vásárolt/eladott mennyiség

Az alkalmazás során a felhasználó bizonyos menük között fog váltani és a program futása során valószínűleg sokszor megfogja nyitni ugyan azt a menüt, tehát minden menüből a hatékonyság érdekében csak egy példány létezhessen. Ezt a Singleton tervezési minta segítségével oldom majd meg.

### Grafikus felület

Kezdetben a grafikus felület megtervezésére szántam a legtöbb időt, mivel a felület az alkalmazás egyik legmeghatározóbb része. Mivel befolyásolja a szoftver használhatóságát, átláthatóságát és egyszerűségét, avagy bonyolultságát is. Tehát, az alábbi követelményeket állítottam fel a grafikus felülettel szemben:

* A C# WPF (Windows Presentation Form) felületét mind kód szinten, illetve XAML nyelv segítségével lehet szerkeszteni. De mivel célom volt, hogy elkülönítsem, a megjelenítéshez tartozó kódot a tényleges programkódtól a könnyebb szerkeszthetőségért, ezért a grafikus felület megvalósítását ahol csak lehet ott XAML szinten oldom meg.
* Az aktuálisan belépett felhasználónév jelenjen meg az program jobb felső sarkában, illetve ugyan ezen a részen jelenjenek majd meg a felhasználói fiókot érintő esetleges értesítések (az értesítések figyelem felhívó színnel rendelkezzen például piros, sötét sárga).
* A rendszer bizonyos funkciói eléréséhez tartalmazzon gombokat, amikre rákattintva a felhasználó meg tudja jeleníteni a kattintott menü tartalmát. Ezen gombok helyezkedjenek el egy menüben. Mivel az alkalmazás képes lesz banki és tőzsdei adatokat is kezelni ezért a menük nevei legyenek az általuk ellátott funkciók és helyezzük el bennük almenüket a banki és a tőzsdei funkciók elkülönítése érdekében. Az aktuálisan megjelenített menü gombjának háttérszíne vegyen fel egy megkülönböztető színt a többi gombhoz képest. Valamint, fontos hogy a menü ne zavarja a felhasználót a szoftverrel való iteráció során.
* A grafikus felület felső részén, az aktuálisan megjelenő felhasználónévvel egy sorban legyen egy rövid leírás az aktuálisan megjelenített menüről és annak céljáról (mire szolgál), ez természetesen frissüljön minden menüváltáskor.
* A felhasználó a fontosabb eseményekről az oldal közepére kapjon egy felugró szövegdoboz ( Message Box) értesítést.
* Ha a felhasználónak csak az eltárolt adatokból van lehetősége választani, akkor jelenjen meg a számára egy legördülő lista amely tartalmazza a lehetséges elemeket. Ellenben ha új adatot visz fel a felhasználó, akkor sima szövegdoboz jelenjen meg a számára.



Ábra: korai grafikus felület képernyőterve.

### Tranzakciók importálása

Mivel az új tranzakciók importálása a fő funkciója a szakdolgozatomnak ezért annak tervezésével töltöttem el a legtöbb időt. Az alábbi követelményeket állítottam fel az importálás funkcióval szemben:

* Csak abban az esetben fog eltárolásra kerülni egy importált banki tranzakció, ha annak adatai még nem szerepelnek az eltárolt adatok között. Vagy a felhasználó úgy dönt, hogy mindenképpen importálni szeretné a tranzakciót.
* .A tőzsdei tranzakciók esetében a felhasználó válasza ki az általa kívánt nyereség - veszteség kiszámolási módszert.
* A tranzakciók importálása után a felhasználó statisztikái, illetve a tranzakciókat tartalmazó adatok, táblázatok automatikusan frissüljenek egyből mind tőzsdei illetve banki importálás során.
* A felhasználó kapjon egy felugró szöveges értesítőt mind banki, mind tőzsdei fájl importálásának végén, hogy melyik fájlból hány darab tranzakciót sikerült felvinnie. Ha szerepelt már az adatbázisban a fájlból valamennyi tranzakció, akkor az egyezések számát is írja ki.
* Mind tőzsdei, illetve baki tranzakciók esetén a rendszer adatbázisában ugyan abban a sorrendben kerüljenek eltárolásra a tranzakciók, ahogyan az importált fájlban voltak.

Az importálás folyamata alatt a kívánt fájl kiválasztása előtt a felhasználónak legyen, lehetősége kéttípusú importálás között választani egy felugró szöveges ablak segítségével ahol választhat az Automatikus importálás és a Felhasználó által deklarált importálás között.

#### Automatikus importálás

Ebben az esetben a szoftver olyan algoritmusokat használ, ahol a fájlt megvizsgálva megpróbálja beazonosítani azon oszlopokat illetve cellákat, amelyek egy érvényes tranzakció beolvasásához szükségesek. Ha az algoritmusok által nem sikerül beolvasni az importált fájl adatait, akkor a felhasználó kap egy értesítést, hogy használja a Felhasználó által deklarált importálást. A rendszer sikeres importálás végén elmenti a kiolvasott tranzakciókat.

#### Felhasználó által deklarált importálás

Ha a felhasználó az ő általa vezérelt importálási folyamatot választja, akkor megjelenik számára egy új oldal ahol látni fog szövegdobozokat ahova neki kell beírnia a bizonyos oszlopokat illetve cellákat, majd ha kitöltötte az összes szövegdoboz, egy gombra kattintva jelzést ad a szoftvernek hogy az kezdheti az adatok kiolvasását.

A rendszer a sikeres beolvasás végén elmenti a felhasználó által beírt oszlopokat és cellákat, amiket beírt a szövegdobozokba, majd azokat elmenti a rendszer adatbázisába. Abból a célból, ha a felhasználó legközelebb ilyen típusú fájlt szeretne importálni ismételten, a rendszer automatikusan töltse ki számára a kellő szövegdobozokat automatikusan

### Tárolt adatok megtekintése

* A tárolt adatok megjelenítésé esetén minden esetben táblázatot fogok használni ahol a táblázat szélessége egyenlő lesz az oldal szélességével mínusz a menü szélessége. A táblázatokat oszlopokra osztom fel ahol az oszlopok a tranzakcióhoz tartozó tulajdonságok.
* Ha egy adott cellába nem fér bele a megjelenítendő adat, tehát a cella mérete kisebb, mint a benne szereplő adaté akkor, ha a kurzort a cellára visszük egy kis ablakban meg fog jelenni a kurzorunk alatt a cella teljes tartalma.
* Tőzsdei tranzakciók esetén, ha nyereséges volt az adott részvény eladása, akkor azt jelezzük a felhasználó számára zöld színnel, ellenkező esetben piros színnel adjuk a tudtára hogy veszteséges volt az eladás.
* Ha szükséges (a táblázat elég adatot tartalmaz ahhoz, hogy a hossza nagyobb legyen mint a teljes grafikus felület hossza) a táblázat jobb oldalán megfog jelenni egy csúszka, amit használva (görgetéssel vagy rákattintással is tudunk irányítani) megtekinthetjük az összes táblázatban lévő adatot.
* A táblázatok banki és tőzsdei adatok megjelenítése esetén tartalmazzon minden tranzakciós adatot, amit az importálás alatt olvastunk ki, illetve tartalmazza még a rendszerbe való beimportálás dátumát is.

### Részvényadatok lekérése Internetről

* A felhasználónak legyen lehetősége beírni az általa lekért részvény szimbólumát egy szöveges dobozba. Majd megadnia, hogy milyen dátumra visszamenőleg szeretné lekérni az adatokat.
* Majd a kapott adatokat jelenítse meg a felhasználó számára egy gráf segítségével ahol az X tengely fogja tartalmazni a dátumokat, az Y tengely pedig az aktuális árakat.
* Illetve ha a felhasználó ráviszi a kurzort a diagramon található pontokra, akkor láthassa a részvényre vonatkozó konkrét árat is.
* Mivel erre az adatok lekérésére valószínűleg egy külső API-t fogok használni ezért a felhasználónak két adatlekérés között várnia kell legalább fél percet, így a lekérés gomb e harminc másodperci időintervallumra letiltott állapotban lesz a felhasználó számára.
* A lekért adatokat a megjelenítéssel egyhuzamban mentse el a program az adatbázisba a későbbi megjelenítés érdekében.

### Binding és EvenHandler használata

Ahol lehet ott Binding segítségével kell megoldani a bizonyos események bekövetkezését, adatok lekérését, változtatását. Például a gombokhoz rendelt parancsok, táblázatokhoz tartozó adatok, legördülő lista elemei, aktuálisan kiválasztott elem, gomb kattinthatósága (boolean).

Ha nem lehetséges megoldani a Bindingot, akkor EventHandlert fogok használni.

A Bindingot részesítem az EventHandlerrel szemben előnyben, mivel a Binding használtával könnyebb elkülöníteni a kód azon részét, ami a grafikus felületen lévő interakcióktól függ, a kód azon részétől, ami a belső működésért felel. Ha EventHandlert használok egy bizonyos elemre, akkor az EventHandler függvénynek mindenképpen abban az osztályban kell lennie ahol az elem létre van hozva (Az elemek az osztály adattagjai képezik), így egy olyan kód keletkezik ahol keveredik az üzleti, és a megjelenítésért felelő kód.

A különböző kódok elkülönítése érdekében egy úgynevezett MVVM mintát fogok használni a kódban lévő fájlokra, ami azt jelenti hogy külön osztályban fognak elhelyezkedni a megjelenítésért felelő és az üzleti logikát tartalmazó kódok. Így biztosítva a könnyű átláthatóságot és elkülönítést.

### Importált adatok elmentése

Az alkalmazás egy belső SQL adatbázisba tárolja el az importált adatokat. Külön adatbázis kell a Felhasználók, a Tranzakciókhoz tartozó információkhoz, Illetve az internetről lekért részvényadatokhoz.

Fontos alapkövetelménye az adatok mentésnek hogy ne érzékeljen semmit a felhasználó az elmentés folyamatából. Ezen feltétel alatt azt értem, hogy külön szálon fusson az adatbázisba való írás, hogy ne akadályozza a felhasználó tevékenységét eseteleges ,,megfagyásokkal" .

Az adatbázisba való írás SQL query-ken keresztül történjen meg.

## Személyes pénzügyi importáló megvalósítása

Ebben a fejezetben befogom mutatni az elkészült Személyes pénzügyi importáló programot, illetve a bizonyos részekhez tartozó implementációkat.

### Grafikus felület

Mint ahogy a tervezés fejezetében írtam, kezdetben a grafikus felület megvalósítására fókuszáltam. A grafikus felület animációs megvalósításához csak XAML nyelvet használtam. A programkódban csak a bizonyos részek láthatóságát változtatom, illetve bizonyos elemek színét

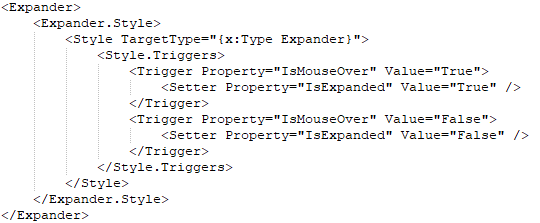
Fontosnak tartottam a modern megjelenést, így egy olyan előugró menüre gondoltam, ami abban az esetben lesz teljesen látható, ha a felhasználó ráviszi az kurzort. Szerencsére erre remeg lehetőséget ad az XAML-ben lévő Tag-ek sokasága.

Tehát az oldal bal oldalára helyeztem el a menü sort, és alapértelmezetten a menükhöz tartozó kép látszódik, amik egy StackPanelben helyezkednek el, ha a felhasználó ráviszi az egeret e StackPanel-re akkor ,,csúszik" elő a menü addig elrejtett része.

Természetesen gondoltam az almenükre is, az almenük abban az esetben jelennek meg automatikusan egy főmenü alatt, ha a felhasználó ráviszi az egerét a főmenüre, viszont ha egy új menüpontra húzza át azt, akkor automatikusan bezáródik a régi almenü és megnyílik az új.

Az almenü abban az esetben is bezáródik, ha a felhasználó elviszi a kurzor fókuszát a StackPanelről, ami a menü elemeit tartalmazza.

Elsőként az almenük megjelenését, illetve a bizonyos események esetén történő eltűnését implementáltam le.



Ábra: Az almenük megjelenése és eltűnéséért felelős XAML kód

Az Expander egy olyan XAML Tag, ami által bizonyos elemek tartalmazhatnak magukban több elemet is, és alapértelmezetten kattintás segítségével nyithatjuk le őket. De mivel én nem ezt a megoldást szemléletem ki, ezért egy egyedi stílust (Style) kellet létrehoznom, aminek meg kell adni egy célt, hogy mire hajtsa végre a stílusváltozást (TargetType), aminek megadtam magát az Expandert (hogy az egész elemre kihatással legyen).

Ezek után meg kell adni, hogy a stílusváltozás egy esemény bekövetkezésekor lépjen életbe (Style.Trigger). Ez az esemény nem más, mint az IsMouseOver állapot, ami akkor lesz igaz, ha az adott elem, amiben az expander van, a kurzor alatt helyezkedik el.

Ha az IsMouserOver igaz, tehát a kurzor alatt az elem helyezkedik el, automatikusan kibontjuk az elemet, ahol később az almenük fognak elhelyezkedni, úgy hogy az elem IsExpanded tulajdonságát (Property) expliciten Igazra (True) állítjuk.

Ha az IsMouseOver hamis, tehát a kurzor alatt már nem az aktuális elem helyezkedik el így annak az IsExpanded tulajdonságát expliciten Hamisra (False) állítjuk, ezzel bezárjuk az aktuális Expandert.

A következő feladatom volt az előugró menü kódjának leimplementálása, aminél ismét használnom kellet többek között a Trigger, Style Tageket.



Ábra: A főmenü előugrásának animációja.

Amint látható a fenti ábrán, a menü (StackPanel) alapértelmezett helyzete -90 pixel távolságra helyezkedik el a bal margótól, tehát a felhasználó számára láthatatlan. Először is a menü elemeit beletettem egy StackPanelbe, ami a nevéből adódóan egymásra teszi a benne lévő elemeket. Fontos megjegyezni, ha a StackPanelben elhelyezkedik elem, akkor annak az elemnek a szülő objektuma a StackPanel, és ha a szülő objektum valamelyik tulajdonságát megváltoztatjuk, úgy változik a benne lévő gyerek elemek tulajdonsága is. Ezt a tulajdonság változást kihasználva elég a StackPanelhez hozzárendelni egy stílust, ami kihatással lesz a benne lévő elemekre is.

Tehát a StackPanelhez hozzárendelt stílus (Style) változtatásának célpontja (TargetType) maga a StackPanel. Ezen belül elhelyezkedik két eseményfigyelő (EventTrigger):

* MouseEnter, ami akkor lép életbe, ha a jelen esetben a StackPanelre történik egy kurzor belépés. Ilyenkor elkezdődik egy Storyboard. A Storyboard egy olyan Tag, amelyben eseményeknek időkorlátot, késleltetést állíthatunk be. A Storyboard Tagen belül a ThicknessAnimation Tag-et használtam, ami lehetővé teszi elemek eltolását, jelen esetben az eltolás cél tulajdonsága a Margó (Storyboard.TargetProperty) amit -90 pixeltől eltolunk 0 pixelig, tehát a végülis a StackPanel bal oldali margóját eltoljuk jobbra 90 pixellel, illetve ehhez beállítottam egy fél másodperces lefutási időt (Duration). Így a StackPanel azon része is látszódni fog, ami eddig a felhasználó számára eddig láthatatlan volt.
* MouseLeave, ami akkor lép életbe, ha a jelen esetben a StackPanel felületről ,,lehúzzuk" a kurzort. Ilyenkor elkezdődik egy Storyboard. A Storyboard Tagen belül a ThicknessAnimation Tag-et használtam, ami lehetővé teszi elemek eltolását, jelen esetben az eltolás cél tulajdonsága a Margó (Storyboard.TargetProperty) amit 0 pixeltől eltolunk -90 pixelig, illetve ehhez beállítottam egy fél másodperces lefutási időt (Duration). Tehát fogjuk a StackPanel bal oldali margóját és azt eltoljuk balra 90 pixellel.

Mivel a különböző menükhöz különböző oldalak tartoznak, de természetesen a program csak egy ablakkal fog rendelkezni, ezért a főoldalban kijelöltem egy részt, ami a tartalmi részét jelenítni majd meg az oldalnak és ezt a területet bele tettem egy Frame Tag-be amit elneveztem MainFrame-nek (Balról 150, felülről 70 pixel kivételével a maradék rész a MainFrame-hez tartozik).

<Frame x:Name="MainFrame" Margin="150,70,0,0"/>

Így amikor megnyitunk, egy új menüpontot csak meghívjuk a főoldal objektumának MainFrame adattagját és átállítjuk a Content tulajdonságát az új oldal tartalmára.

mainWindow.MainFrame.Content = new BankImportPage();

A fent szereplő kód azt próbálja szemléltetni, hogy a MainFrame Content adattagja legyen egyenlő egy új BankImportPage objektummal. Fontos hogy ez csak akkor működik, ha a BankImportPage öröklődik a Page osztályból (System.Windows.Controls névtér).

A Gombokhoz tartozó parancsokat Binding-al oldottam meg. Ami a következőképpen néz ki XAML felöl.

<Button Content="Import" Command="{Binding Path=import}"/>

Látható Command tulajdonságához az Import nevű Binding tartozik. A kódba ezt úgy kell lekezelni, hogy írni kell hozzá egy osztályt, ami öröklődik az ICommand osztályból.

public ICommand importPushed{

get{

string v = sender();

return new CommandHandler(v);

}

}

private string sender([CallerMemberName]string memberName = ""){

return memberName;

}

public class CommandHandler : ICommand{

private string commandName;

public CommandHandler(string value){commandName=value}

public void Execute(){

if(commandName=="import"){

//do something

}

}

public bool CanExecute(object parameter){

return true;

}

}

A fenti kódból láthatjuk, hogy van egy ICommand típusú objektumunk, amit ha lekérünk, akkor két dolog történik.

* Meghívódik egy sender nevű függvény, aminek a visszatérési értéket egyenlővé tesszük egy lokális változó értékével. A sender függvény visszatéríti annak a Binding parancsnak a nevét, amely elemtől érkezett a kérés, jelen esetben a függvény az ,,import " stringel fog visszatérni.
* A sender függvény meghívása után létrehozunk egy példányt a CommandHandler osztályból, ami egyben a visszatérési érték lesz. A CommandHandler objektum létrehozásánál a konstruktort a lokális string változó értékével hívjuk meg, amit a konstruktor létrehozásánál elment az objektum. Majd automatikusan meghívódik az Execute() függvény ahol megvizsgálja az objektum commanName adattagját.

Láthatjuk, hogy a CommandHandler osztálynak még van egy CanExecute függvénye, ami mindig Igazat térít vissza, ezt köthetjük logikához természetesen, de a személtetéshez erre nem volt szükség.

# Elért eredmények

## Kezdőlap



Ábra: Az alkalmazás bejelentkező felülete

A bejelentkező oldalon láthatunk két szövegmezőt amelyek segítségével a felhasználó be tud jelentkezni a rendszerbe. A felső szövegmező a felhasználónév azonosításért felel, míg az alatte lévő a jelszó azonosításért, a jelszót azonosító szövegdobozban minden karakter egy pont formájában jelenik meg, megvédve a felhasználót az esetleges jelszó lopástól. A sikeres bejelentkezéshez ezen mezők kitöltése természetesen kötelező Illetve a Kezdőlapon láthatunk még két gombot is, amik a Bejelentkezés végrehajtásáért, valamint a Regisztrációért felelősek. A billentyűzet fókusza az applikáció indulásánál alapértelmezetten a felhasználónév mezőben van.

Ha a felhasználó úgy kattint rá a Bejelentkezés gombra, hogy a felhasználónév mezőt és/vagy a jelszó mezőt üresen hagyta akkor az oldal, értesítést (Messagebox) küld az érvénytelen próbálkozás után.(A felhasználónév és a jelszó mezők üresek!).

Ha a felhasználó kitöltötte a felhasználónév a jelszó mezőt, de a felhasználónév vagy a jelszó nem egyezik az adatbázisban szereplő adatokkal a rendszer, értesítést (Messagebox) küld a bejelentkezni kívánó személynek a hibáról (Nem megfelelő felhasználónév vagy jelszó). Ezután a felhasználó újrapróbálkozhat, de maximum három lehetősége van hibázásra, utána nem próbálkozhat bejelentkezhetni két percig.

A bejelentkezés folyamata (felhasználónév illetve jelszó ellenőrzés) elkezdődik a Login gombra kattintva vagy egy Enter gomb lenyomásával is, feltéve, ha e mezők nem üresek. A rendszer küld egy kérést a belső adatbázis számára a felhasználónév és a hozzá tartozó jelszóval. A bejelentkezés akkor lesz sikeres, ha a felhasználónév és jelszó kombináció megegyezik a tárolt adattal a felhasználók adatbázisában. Ha az adott személy még nem regisztrált, akkor a Regisztrálás gombra kattintva létrehozhat egy új fiókot.

Ha a felhasználó rákattint a Regisztrálás gombra, egy új ablak jelenik meg. Ahol látni fog három szövegmezőt ahova be kell írnia a leendő felhasználónevét, amivel később be tud jelentkezni illetve a bejelentkezéshez szükséges jelszavát is.



Ábra: Az alkalmazás bejelentkező felülete

A jelszó beírásra kétszer van szükség, természetesen e jelszavaknak egyezniük kell különben a regisztráció nem lesz sikeres.

A regisztráció abban az esetben sem sikeres, ha az adatbázisban már szerepel ilyen nevű felhasználó. Ilyen esetben az oldal értesítést (Messagebox) küld a regisztrálni kívánó felhasználónak, hogy az a felhasználónév, amivel regisztrálni kíván már foglalt és újat kell választania.

A leendő felhasználónévnek legalább négy karakter hosszúnak kell lennie, míg a jelszó hossza legalább hat karakter és tartalmaznia kell számot is. A felhasználónév és a jelszó formájára (bizonyos karakterek megkövetelt használatára) nincsenek egyéb megkötések (például nagybetűk használata a jelszóban illetve speciális karakterek használata).

Ha a felhasználó kitöltötte a három szövegmezőt és rákattint a Regisztrálás gombra hat lehetséges esemény következhet be, amiről a regisztráló fél minden esetben értesítést fog kapni (Messagebox).

* A regisztráció sikeres volt, ezután a felhasználó már be tud lépni az új fiókjával.
* A regisztráció nem volt sikeres, mivel a felhasználónév már foglalt.
* A regisztráció nem volt sikeres, mivel a jelszómezők tartalma nem egyezik.
* A regisztráció nem volt sikeres mivel a felhasználónév nem elég hosszú (legalább 4 karakter).
* A regisztráció nem volt sikeres mivel a jelszó nem elég hosszú (legalább 6 karakter).
* A regisztráció nem volt sikeres mivel a jelszó nem tartalmaz legalább egy számot.

Ha a regisztrálás sikeres volt, akkor a felhasználó újra a kezdőoldalt fogja látni.

Ha a bejelentkezés sikeres a felhasználó az alkalmazás főmenüjét fogja látni, ahol a jobb felső sarokban láthatja az értesítéseit és az aktuális felhasználó nevét. Egy felhasználó értesítései attól függenek, hogy az aktuális dátumhoz mérten mikor importált legalább egy új tranzakciót. Az új tranzakció alatt jelen esetben egy olyan vagy banki, vagy tőzsdei tranzakciót kell érteni, ami még nem szerepelt az importálás előtt az alkalmazás adatbázisában.

A felhasználónak szóló értesítésnek három színe lehet:

* Piros: A felhasználó nem importált új tranzakciót több mint egy hónapja.
* Sárga: A felhasználó legalább tizenöt, de legfeljebb egy hónapja nem importált
* Zöld: Ha az értesítések száma nulla, tehát a felhasználó importált új legalább egy új tranzakciót az elmúlt tizenöt napban egy zöld pipa fog megjelenni az értesítések helyén ezzel jelezve a felhasználónak, hogy napra készen áll a személyes pénzügyeivel.

## Tranzakciók Importálása

Amint sikeresen belépett a felhasználó az alapértelmezett oldal, ami automatikusan megjelenik neki az a Tranzakciók importálása menüpont. Amely tartalmaz két almenüt:

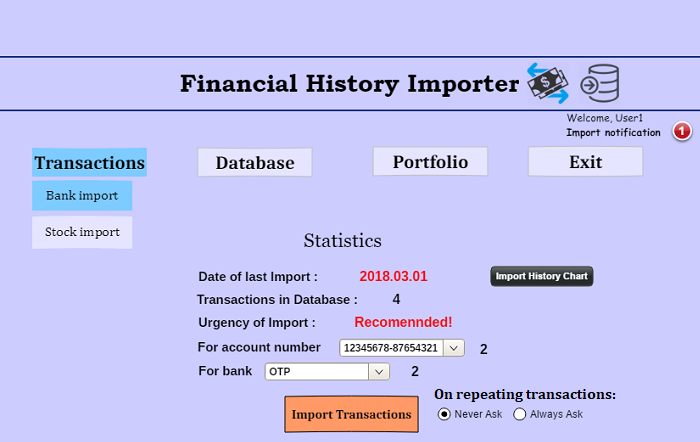
* Banki Tranzakciók importálása
* Tőzsdei Tranzakciók Importálása

A két almenü között az almenük nevét tartalmazó gombra való kattintás segítségével tudunk váltani. Az aktuálisan kiválasztott menü gombja különböző háttérszínnel rendelkezik, mint a többi gomb, ennek segítségével a felhasználó könnyen tudja, melyik menü van kiválasztva jelenleg. A sikeres bejelentkezést követően az alapértelmezetten kiválasztott almenü a Bank Tranzakciók importálása.

### Banki Tranzakciók importálása

A felhasználó a Banki Tranzakciók importálása menüre rákattintva láthat egy rövid áttekintést az általa már beimportált és a szoftver adatbázisában lévő banki tranzakciókról.

* Utolsó importálás ideje
* Tárolt banki tranzakciók az adatbázisban összesen
* Importálás szükségessége
* Különböző számlákhoz tartozó tranzakciók száma (számlaszám vagy különböző bankok száma egy legördülő listában).
* Különböző bankokhoz tartozó tranzakciók száma



Ábra: Banki Tranzakciók importálása

Az utolsó importálás ideje azt a dátumot jeleníti, meg amikor a felhasználó sikeresen importált banki tranzakciókat. A sikeres importálás jelen esetben azt jelenti, hogy a felhasználó egy olyan tranzakciót vitt fel a rendszer adatbázisába amit eddig az nem tartalmazott, tehát ha egy olyan fájlt importálunk, amiben csak olyan adatok vannak, amik már szerepelnek az adatbázisban akkor ez a dátum nem fog frissülni. (Kivétel erre az esetre, ha a felhasználó ettől függetlenül is importálni szeretné az adott tranzakciókat.).

A Tárolt banki tranzakciók azt a számot jeleníti, meg hogy hány tranzakciót importáltunk összesen, banktól, számlaszámtól, dátumtól függetlenül.

Az importálás szükségessége egy jelzés a felhasználó számára, hogy ha eredményesen szeretné nyomon követni pénzügyeit, akkor ajánlott egy bizonyos időközönként importálnia új tranzakciókat. A "Szükséges" felírat, akkor jelenik meg, ha a legfrissebb újonnan beimportált banki tranzakció dátuma az aktuális dátum előtt van legalább egy hónappal. Különben ha az utolsó banki tranzakció felvitele óta legalább tizenöt, de legfeljebb egy hónap telt el, az "Ajánlott" felirat fog megjelenni. Ha a legfrissebb banki tranzakció importálása óta még nem telt el legfeljebb tizenöt nap a felhasználó a "Naprakész" feliratot láthatja. A feliratoktól függően változik a felhasználó értesítése.

A Különböző tárolt számlák felirat és a mellette lévő legördülő lista csak abban az esetben látható a felhasználó számára, ha már sikeresen importált legalább egy darab tranzakciót. Ebben a sorban a felhasználó láthatja az általa beimportált fájlból kinyert bankszámlaszámo(ka)t, amelyeket egy legördülő lista segítségével választhat ki. Egy elem kiválasztása esetén megjelenik egy szám a lista mellet, amely jelzi az kiválasztott számlaszámtól importált tranzakciók számát. Ezen információ segítségével könnyen láthatjuk melyik bankszámlánkról, hányszor importálunk már banki tranzakció(ka)t a rendszerbe.

A Különböző tárolt bankok felirat és a mellette lévő legördülő lista csak abban az esetben látható a felhasználó számára, ha már sikeresen importált legalább egy darab tranzakciót. Ebben a sorban a felhasználó láthatja az általa beimportált fájlból kinyert bankneveket, amelyeket egy legördülő lista segítségével választhat ki. Egy elem kiválasztása esetén megjelenik egy szám a lista mellet, amely jelzi az kiválasztott banktól importált tranzakciók számát. Ezen információ segítségével könnyen láthatjuk melyik banktól, hányszor importálunk már banki tranzakció(ka)t a rendszerbe.

Ezen négy importálási statisztikát tartalmazó szövegmező mellet láthatunk egy gombot, ami csak abban az esetben jelenik meg, ha a felhasználó legalább egy tranzakciót sikeresen importált az adatbázisba. Ez a gomb Importálási Történet diagram felirattal rendelkezik. Ha erre a gombra rákattint a felhasználó egy új ablak fog megjelenni, ami tartalmaz egy legördülő listát és egy diagramot. A legördülő lista tartalmazza azon banko(ka)t amely fájljait felhasználva a felhasználó sikeresen importált a rendszer adatbázisába legalább egy tranzakciót. A lista aktuálisan kiválasztott elemétől függ a diagram által megjelenített adat. Alapértelmezetten az Összes opció van kiválasztva, ami azt jelenti hogy az összes banktól származó importálások számát fogja megjeleníteni a diagram. A diagram X tengelye tartalmazza a tranzakció importálásának dátumát, növekvő sorrendben. Az Y tengely pedig jelzi az adott napon felvitt tranzakciók számának összegét. Továbbá diagram jobb felső sarkában láthatunk egy gombot, ami a vissza felirattal rendelkezik. Erre rákattintva a felhasználó visszakerül a Banki Tranzakciók Importálása menübe.

Az Importálási Történet diagram gomb alatt láthatunk két rádió gombot, amik felett az alábbi felirat szerepel: "Ismétlődő tranzakció esetén". Ezen rádió gombok változtatása kihatással van az importálás folyamán fellépő esetleges ismétlődő tranzakciók felismerésére és szoftver adatbázisába történő elmentésükre. A rádió gombok két opciót ajánlanak fel:

* Mindig kérdezzen rá
* Soha ne kérdezzen rá

A rádió gomboktól függetlenül, a szoftver egy algoritmust használ a beimportálni kívánt fájlok adataira, ami összehasonlítja a már eltárolt tranzakció adataival és megpróbálja kiszűrni azokat az adatokat, amik nagy valószínűséggel már el vannak tárolva a rendszer adatbázisában, többek közt figyelembe veszi a tranzakció végbemenetelének dátumát, tranzakció pénzösszegét és az akkori számla egyenleget és nem utolsó sorban a tranzakció bankszámlaszámát is.

Ha a Mindig kérdezzen rá rádió gombot választja ki a felhasználó, abban az esetben, az importálás folyamata alatt egy valószínűleg ismétlődő tranzakciót talál a szoftver, értesíti a felhasználót, hogy valószínűleg az aktuális fájlból importált bizonyos tranzakciók már szerepelnek az adatbázisban, kiírva a tranzakció végbemenetelének dátumát és tranzakció pénzösszegét is. Ezek után az alkalmazás két lehetőséget kínál fel a felhasználó számára:

* Figyelmeztetés figyelmen kívül hagyása, a felhasználó tisztában van az esetleges redundanciával de ezt figyelmen kívül hagyja és szeretné beimportálni az adott tranzakciót a rendszerbe.
* Ne importálja az aktuális tranzakciót, ez esetben a felhasználó nem importálja a tranzakciót mivel az már szerepel a rendszer adatbázisában így az redundáns adat lenne.

Soha ne kérdezzen rá rádió gomb kiválasztása esetén a felhasználó átadja az irányítást a szoftvernek ami saját kezűleg dönti el hogy importáljon-e egy bizonyos tranzakciót, figyelembe véve annak a lehetőségét hogy már eltárolt adatról van-e szó.

Alapértelmezetten a Soha ne kérdezzen rá rádió gomb van kiválasztva.

Tranzakciók Importálása gomb: Erre a gombra rákattintva megjelenik a felhasználó számára két választási lehetőség egy Messagebox formájában. Amin az alábbi két lehetőséget kiválasztására van lehetősége a felhasználónak:

* Automatikus importálás
* Felhasználó által deklarált importálás

Ha a felhasználó az Automatikus importálás opciót választja akkor a következőekben az általa kiválasztott fájlra az alkalmazás bizonyos algoritmusokat használva megpróbálja azokat a cellákat és oszlopokat beazonosítani, amelyek szükségesek egy tranzakció egyértelmű azonosítására. Amik a következőek:

* Fájlhoz tartozó bankszámlaszám
* Tranzakció dátuma
* Tranzakció összege

Tehát ha ez a három adat sikeresen kiolvasható egy fájlból akkor már van beolvasható tranzakció amit beilleszthetünk az adatbázisba. Ha ezt a három adatot nem sikerül megtalálnia az algoritmusnak akkor értesítést küld a felhasználónak hogy az adott fájlra alkalmazza a Felhasználó által deklarált importálást. Természetesen egy tranzakcióhoz több adat is tartozhat, ez teljes mértékben a banktól függ ahonnét a tranzakció történetet tartalmazó fájl származik. Ilyen kiegészítő adat lehet:

* Tranzakció leírása
* Jelenlegi egyenleg

Fájl típusától függően több cella is tartalmazhat tranzakcióhoz tartozó leírást. Az alkalmazás által használt beolvasó algoritmus megpróbálja felismerni a legtöbb lehetséges leírásra illeszkedő cellát.

A jelenlegi egyenleget a legtöbb fájl megjeleníti, de természetesen a bankoktól letöltött fájlok mint már korábban is leírtam nagyon különböznek egymástól. Így hát az is megtörténhet hogy bizonyos fájlok nem rendelkeznek ezzel az adattal.

Ha az algoritmus sikeresen felismerte a kellő cellákat és kiolvasta az összes lehetséges tranzakciót, ezek után elkezdődik a kiolvasott tranzakciók adatainak összehasonlítása a már eltárolt tranzakciók adataival.

Ha az ismétlés megtalálásáért felelő algoritmus talál egy valószínűleg már eltárolt tranzakciót a felhasználó korábbi döntéseit figyelembe véve reagál (Ismétlődő tranzakciók esetén rádió gombok).

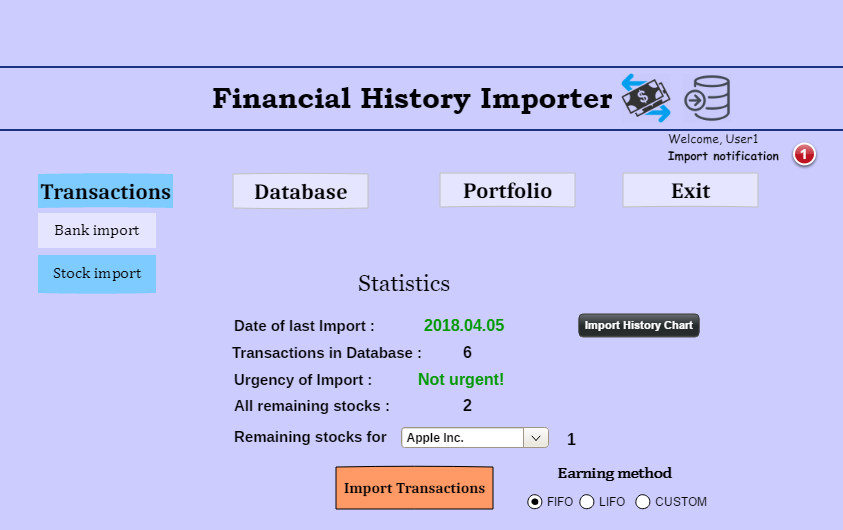
Miután az összes beimportált tranzakció átvizsgálásra került, az alkalmazás exportálja a kellő adatokat az adatbázisba. Így azok már később megtekinthetőek a szoftverben táblázat formájában, illetve frissítésre kerül a felhasználóhoz tartozó statisztika is.

//todo user-specified import

### Tőzsdei Tranzakciók importálása

A felhasználó a Tőzsdei Tranzakciók importálása menüre rákattintva láthat egy rövid áttekintést az általa már beimportált és a szoftver adatbázisában lévő tőzsdei tranzakciókról.

* Utolsó importálás ideje
* Tárolt tőzsdei tranzakciók az adatbázisban összesen
* Importálás szükségessége
* Jelenleg birtokolt részvények összes száma
* Jelenleg birtokolt részvények száma különböző cégekre



Ábra: Tőzsdei Tranzakciók importálása

Az utolsó importálás ideje azt a dátumot jeleníti meg amikor a felhasználó sikeresen importált tőzsdei tranzakció(ka)t. A sikeres importálás jelen esetben azt jelenti, hogy a felhasználó egy olyan tranzakciót vitt fel a rendszer adatbázisába amit eddig az nem tartalmazott, tehát ha egy olyan fájlt importálunk, amiben csak olyan adatok vannak, amik már szerepelnek az adatbázisban akkor ez a dátum nem fog frissülni. (Kivétel erre az esetre, ha a felhasználó ettől függetlenül is importálni szeretné az adott tranzakciókat.).

A Tárolt tőzsdei tranzakciók azt a számot jeleníti, meg hogy hány tranzakciót importáltunk összesen, cégtől, tranzakció típusától, dátumtól függetlenül.

Az importálás szükségessége egy jelzés a felhasználó számára, hogy ha eredményesen szeretné nyomon követni pénzügyeit, akkor ajánlott egy bizonyos időközönként importálnia új tranzakciókat. A "Szükséges" felírat, akkor jelenik meg, ha a legfrissebb újonnan beimportált tőzsdei tranzakció dátuma az aktuális dátum előtt van legalább egy hónappal. Különben ha az utolsó tőzsdei tranzakció felvitele óta legalább tizenöt, de legfeljebb egy hónap telt el, az "Ajánlott" felirat fog megjelenni. Ha a legfrissebb tőzsdei tranzakció importálása óta még nem telt el legfeljebb tizenöt nap, a felhasználó a "Naprakész" feliratot láthatja. A feliratoktól függően változik a felhasználó értesítése.

A Jelenleg birtokolt részvények összes száma felirat mellett egy darabszámot láthat a felhasználó, méghozzá azon már a szoftver adatbázisába beimportált vásárolt részvények darabszámát amelyekből az esetleges eladott tranzakciók darabszámai levonásra kerültek. Így a felhasználó könnyűszerrel láthatja hogy hány darab vásárolt részvénnyel rendelkezik még az adott időszakban összesen.

A Jelenleg birtokolt részvények száma különböző cégekre felirat mellett egy darabszámot láthat a felhasználó, méghozzá azon már a szoftver adatbázisába beimportált vásárolt részvények darabszámát amelyekből az esetleges eladott tranzakciók darabszámai levonásra kerültek azon termékre leszűrve, amit a felhasználó kiválasztott a felirat mellet lévő legördülő listából. Természetesen a legördülő lista azon részvény termékneveket tartalmazza amit a felhasználó már sikeresen importált a rendszer adatbázisába.

Ezen négy importálási statisztikát tartalmazó szövegmező mellet láthatunk egy gombot, ami csak abban az esetben jelenik meg, ha a felhasználó legalább egy tranzakciót sikeresen importált az adatbázisba. Ez a gomb Importálási Történet diagram felirattal rendelkezik. Ha erre a gombra rákattint a felhasználó egy új ablak fog megjelenni, ami tartalmaz egy legördülő listát és egy diagramot. A legördülő lista tartalmazza azon részvény termékneveket, amelyet a felhasználó egy fájl importálása által sikeresen hozzáadott a rendszer adatbázisához. A lista aktuálisan kiválasztott elemétől függ a diagram által megjelenített adat. Alapértelmezetten az "Összes" opció van kiválasztva, ami azt jelenti hogy az összes terméknévtől származó importálások számát fogja megjeleníteni a diagram. A diagram X tengelye tartalmazza a tranzakció importálásának dátumát, növekvő sorrendben. Az Y tengely pedig jelzi az adott napon felvitt tranzakciók számának összegét. Továbbá diagram jobb felső sarkában láthatunk egy gombot, ami a vissza felirattal rendelkezik. Erre rákattintva a felhasználó visszakerül a Tőzsdei Tranzakciók Importálása menübe.

Az Importálási Történet diagram gomb alatt láthatunk három rádió gombot, amik felett az alábbi felirat szerepel: "Nyereség kiszámítási módszer". Ezen rádió gombok változtatása kihatással van az importálás alatt kiszámolt esetleges vásárlás-eladás nyereségek illetve veszteségek összegének kiszámolási módszerére. A három lehetséges opció:

* FIFO
* LIFO
* CUSTOM

A **FIFO (First In-First Out)** azaz amit időrendi sorrendben előbb vásároltunk meg részvényt, azt fogjuk először "felhasználni" tehát eladni. Az alábbi ábrán, egy egyszerű példával szemléltetem a FIFO elv működését.



Ábra: FIFO elv működése

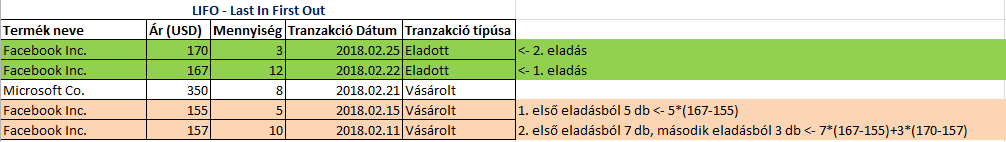
Mivel FIFO elvet alkalmazunk ezért a vásárolt tranzakciókat alulról-felfelé dolgozzuk fel. A fenti ábrából láthatjuk hogy 2018.01.22.-én vásároltunk tíz darab Apple Inc. részvényt, darabonként 170 dollár áron, hívjuk el ezt a tranzakciót Első tranzakciónak. Majd 2018.01.25.-én vásároltunk öt darab Apple Inc. részvényt, darabonként 165 dollár áron hívjuk el ezt a tranzakciót Második tranzakciónak, illetve 2018.02.02.-án vásároltunk nyolc darab Coca Cola Co. részvény darabonként 350 dollár áron, de mivel ezeket a részvényeket nem adtuk el ezért a későbbiekben nem foglalkozunk vele. Majd láthatjuk hogy eladtunk tizenkét darab Apple részvényt 2018.02.02.-án méghozzá 180 dollár értékben. Ha FIFO elvet alkalmazunk a nyereség-veszteség kiszámolására akkor azt a részvényt adtuk el amit legelőször megvettünk. Tehát mivel tizenkettő darab részvényt adtunk el ezért mind az Első tranzakció illetve a Második tranzakcióval számolnunk kell, mivel az Első tranzakcióhoz csak tíz darab részvény tartozik. Mivel FIFO elvet alkalmazunk ezért az Első tranzakcióból adtunk el tíz darab részvényt, aminek a nyeresége a következő: 10\*(180-170) ahol a 10 jelenti az eladott részvények számát, a 180 az eladási árat és 170 a vételi árat. Még maradt két eladott részvényünk az első eladásból amit a Második tranzakcióból adtunk el, mivel az Első tranzakcióból eladtuk minden részvényt. Ennek az eladásnak a nyeresége a következő: 2\*(180-165) ahol kettő az eladott részvények száma, 180 a részvények eladási ára, 165 a Második tranzakcióból származó vételi ár.

Tehát a 12 darab eladott Apple Inc. részvény nyeresége összesen 10\*(180-170)+2\*(180-165) ami egyenlő száz-harminccal.

Az első eladás nyereség-veszteség kiszámolása után van összesen három darab Apple Inc. részvényünk a Második tranzakcióból. De a második eladás során pont három darabot adunk el. Így a második eladáshoz tartozó nyereségét a következő módon számoljuk ki: 3\*(182-165) ahol három az eladott részvények száma, 182 a részvények eladási ára 165 a részvények vételi ára.

Tehát második eladáshoz tartozó nyereség egyenlő ötveneggyel. Illetve rendelkezünk még nyolc darab Coca Cola Co. részvénnyel.

A **LIFO (Last In-First Out)** azaz amit időrendi sorrendben később vásároltunk meg részvényt, azt fogjuk először "felhasználni" tehát eladni. Az alábbi ábrán, egy egyszerű példával szemléltetem a LIFO elv működését is.



Ábra: LIFO elv működése

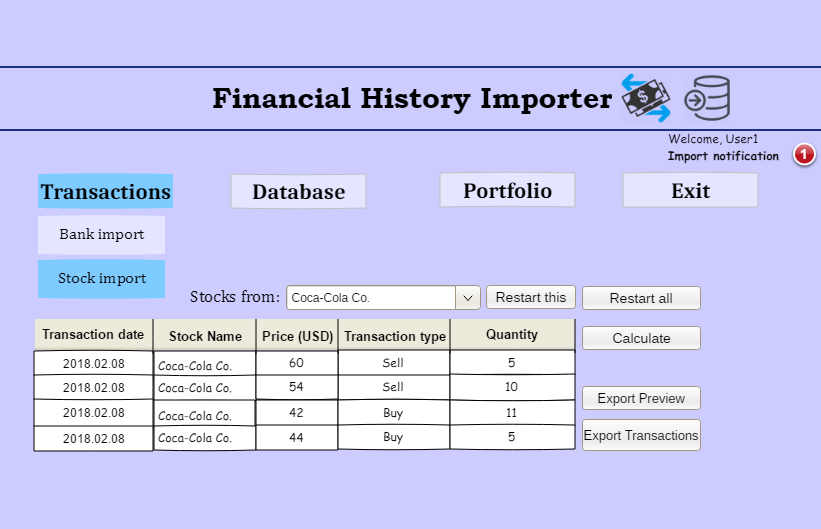
Az ezelőtti példához hasonlóan itt is nevezzük el a vásárolt tranzakciókat. De mivel most LIFO elvet alkalmazunk a vásárolt tranzakciókat fordított időrendi sorrendben fogjuk feldolgozni. Az ábráról leolvashatjuk hogy 2018.02.21.-én vásároltunk nyolc darab Microsoft Co. részvényt háromszáz-ötven dollár értékben, de mivel ezt nem adtuk el, nem foglalkozunk vele a későbbiekben. Láthatjuk 2018.02.15.-én vásároltunk Facebook Inc. részvényt, öt darabot, darabonként száz-ötvenöt dollár áron, nevezzük el Első tranzakciónak. Valamint 2018.02.11.-én is vásároltunk tíz darab Facebook Inc. részvényt, darabonként száz-ötvenhét dollár áron, nevezzük el Második tranzakciónak. Észrevehetjük hogy 2018.02.22.-én eladtunk tizenkét darab Facebook Inc. részvényt, ennek a nyereség-veszteség értékét a következő módon tudjuk kiszámolni LIFO elv használatával: mivel tizenkét darab részvényt adtunk el, és az Első tranzakció csak öt darab részvényt tartalmaz. Ezért az első öt darab eladott részvényhez számolt nyereség a következő: 5\*(167-155) ahol öt az eladott részvények száma, 167 az eladott részvények ára és 155 a vásárolt részvények ára. Mivel tizenkét darabot adtunk el és eddig csak az első ötre számoltuk ki a nyereséget a Második tranzakcióból kell kiszámolnunk maradék hét eladott részvény nyereségét. Az első eladásból vett hét részvény és a Második tranzakcióból számolt nyereség a következő: 7\*(167-157) ahol hét az eladott részvények száma, 167 az eladott részvények ára, 157 a vásárolt részvények ára.

Tehát az első eladáshoz tartozó nyerség egyenlő 5\*(167-155)+7\*(167-157) ami egyenlő százharminccal.

A második eladáshoz tartozó nyereség kiszámolása a következő módon történik: mivel a Második tranzakcióhoz már csak három részvény tartozik, és a második eladás során három részvényt adtunk el így a következő képletet alkalmazhatjuk: 3\*(170-157) ahol három az eladott részvények száma, 170 az eladott részvények ára és 157 a vásárolt részvények ára.

Tehát a második tartozó nyerség egyenlő harminckilenccel. Illetve rendelkezünk még nyolc darab Microsoft Co. részvénnyel.

Ha a felhasználó a **CUSTOM** nyereség-veszteség kiszámítási módot választja az alábbi táblázat és a gombok segítésével a saját íze-kedve szerint köthet össze eladott és vásárolt tranzakciókat.



Ábra: CUSTOM nyereség-veszteség kiszámítás

Az oldalon láthatunk egy táblázatot ami tartalmazza a frissen beimportált tranzakciókat. Méghozzá öt különböző adatot megjelenítve minden tranzakció esetében:

* Tranzakció dátuma: a tranzakció végbemenetelének napja.
* Termék neve: Azonosítja a céget amely a részvények kibocsájtásáért felel.
* Ár: a részvény akkori árát jeleníti meg amikor megvásároltunk vagy eladtuk.
* Tranzakció típusa: a részvény típusa, két értéket vehet fel, eladott vagy vásárolt.
* Mennyiség: megjeleníti az adott áron vett vagy eladott részvény mennyiségét.

A táblázat felett elhelyezkedik egy legördülő lista amely az összes különböző cégekhez tartalmazó részvény nevét tartalmazza, amelyeket a szoftver a felhasználó által beimportált fájlból olvas ki. A táblázat csak attól a cégtől származó részvényt fogja megjeleníteni amit a felhasználó kiválasztott a legördülő listából. Tehát a legördülő lista kiválasztott elemének változtatása hatással van a táblázatban megjelenített adatokra.

A legördülő lista mellet van egy gomb amin az "Újrakezdés erre" feliratot láthatja a felhasználó. Ha rákattint erre a gombra akkor az adott részvényhez tartozó mennyiségek visszakerülnek az eredeti állapotukba. Viszont a legördülő listából az összes többi nem kiválasztott részvénynévhez tartozó adatok maradnak a jelenlegi állapotukban. Tehát erre a gombra akkor ajánlott rákattintania a felhasználónak ha újra szeretné kezdeni a nyereség-veszteség kiszámolását az aktuálisan kiválasztott részvény termékre.

Az "Újrakezdés erre" gomb mellet az "Újrakezdés mindenre" gomb helyezkedik el. Ha erre a gombra kattint rá a felhasználó akkor az összes részvény termék mennyisége a kezdeti állapotba vált vissza. Tehát erre a gombra akkor ajánlott rákattintania a felhasználónak ha újra szeretné kezdeni a nyereség-veszteség kiszámolását az összes beolvasott részvényre.

A részvényeket tartalmazó táblázat mellet láthatunk egy gombot ami a "Kiszámolás" feliratot viseli. Ez a gomb alapértelmezetten kikapcsolt állapotban van, tehát a felhasználó csak bizonyos feltételek teljesítése által tud erre a gombra rákattintani. A gomb engedélyeséhez a felhasználónak ki kell jelölnie a táblázatban egy eladott és egy vásárolt részvényt. Amit az egér kattintása és CONTROL billentyűzet lenyomása segítségével tud megtenni. Ezért arra van szükség mivel így a felhasználó tudja kijelölni melyik eladást melyik vásárlással akarja összekötni. De egy bizonyos részvény nyereség-veszteség kiszámolásához kötelező kijelölni egy vásárolt és egy eladott részvényt. Tehát a gomb addig lesz kikapcsolt állapotban míg ez a feltétel nem teljesül.

A "Kiszámolás" gomb alatt láthatunk egy "Export előnézet" feliratú gombot amire rákattintva a felhasználónak megjelenik egy új táblázat a részvényeket tartalmazó táblázat helyén ami tartalmazni fogja a már kiszámolt részvényeket azon esetleges nyereség illetve veszteségét is egy Profit nevű oszlopban. Ahol a nyereséges eladások profit cellái a felhasználó segítésére zöld hátterű cellával jelennek meg, a veszteséges eladások pedig piros hátterű cellával. Ez a nézet segítség a felhasználó számára a későbbi adatfeldolgozás illetve megjelenítés szempontjából. Továbbá amint a felhasználó rákatiatt "Export előnézet" gombra, annak felirata megváltozik "Vissza"-ra. Ezzel jelezve a felhasználónak hogy az eredeti nézetbe akkor juthat vissza ha újra rákattint a gombra.

Az "Export előnézet" gomb alatt helyezkedik a "Tranzakciók exportálása" gomb amire rákattintva a felhasználó véglegesíti a nyereség-veszteség kiszámolást, és a tranzakciók, azon adataival elmentésre kerülnek a rendszer adatbázisába.

Visszatérve a Tőzsdei Tranzakció importálás menübe, az oldalon elhelyezkedik egy Tranzakciók importálása feliratú gomb, amire ha a felhasználó rákattint. Egy szöveges értesítést (Messagebox) kap az alkalmazástól, ami választás elé állítja a felhasználót. Az alábbi két opcióval

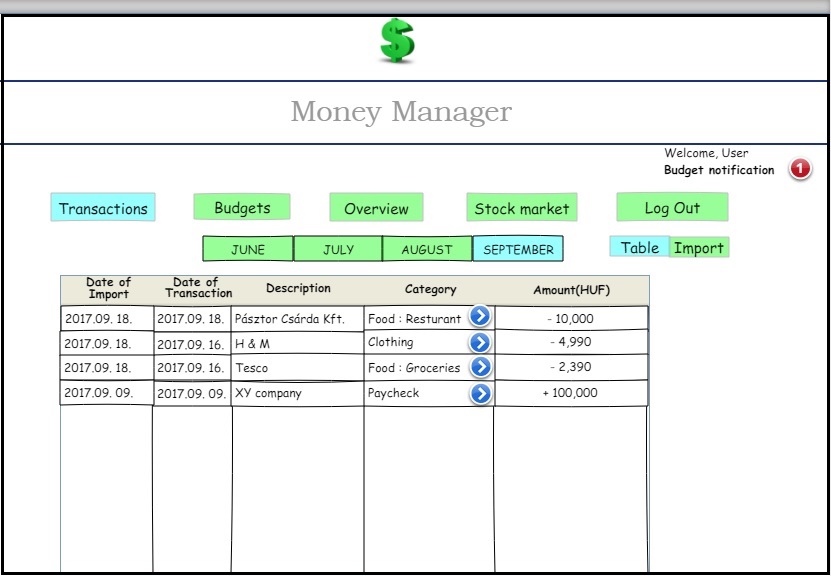
* Automatikus importálás
* Felhasználó által deklarált importálás

Ha a felhasználó az Automatikus importálást opciót választja, a szoftver megpróbálja automatikusan felismerni a

Táblázat és Importálás néven. Alapértelmezetten a Táblázat gomb aktív. Aminek hatására a Tranzakciók menüpont megjelenít egy táblázatot, amely tartalmazza a felhasználó által beimportált fájlból kiolvasott tranzakciókat, a táblázat tartalmaz minden fontos információt a tranzakciókat illetően.

A táblázat öt oszloppal rendelkezik:

* Importálás dátuma: A tranzakció importálásának ideje (ÉÉÉÉ. HH. NN) például 2017.11.11
* Tranzakció dátuma: A tranzakció „végbemenetelének” ideje (ÉÉÉÉ. HH. NN) például 2017.11.11
* Leírás: Tartalmazza a tranzakció közleményét (vásárlás eseten ez a cég neve ahol vásároltunk) vagy az ellenoldali nevet (például fizetés megkapása esetén a cég neve ahol dolgozunk).
* Kategória: Itt a felhasználó beállíthatja milyen típusú volt az adott tranzakció egy legördülő listából. Alapértelmezett érték szerepel ebben a cellában, ha a felhasználó nem állított be értéket.
* Érték: Tartalmazza a tranzakció értékét (Forintban). Ha vásárlunk például -5,000. Ha fizetésünk jön +5,000



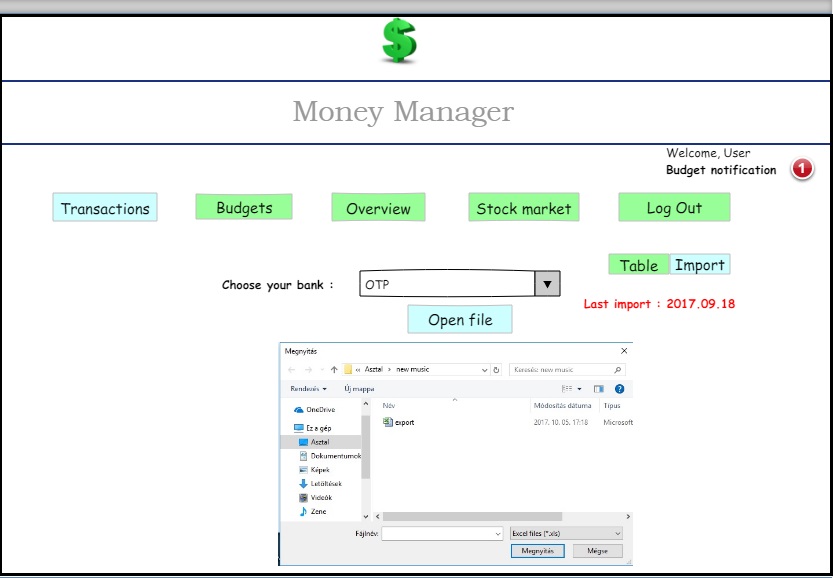
Ábra: Tranzakció menüpont Táblázat menüpontja

Ha a felhasználó be szeretné állítani a Kategória cella értékét az ő általa felvitt „Pénztárca” típusokat választhatja, vagy ha nem szerepel köztük a megfelelő kategória hozzá is adhat új kategóriákat.

Fontos, hogy a bizonyos Kategóriák több alkategóriára oszthatóak. Például az Ételnek lehetnek olyan alkategóriái például:

* Étterem
* Bevásárlás.

Az Importálás gomb konkrétan az alkalmazás fő eleme. Ami azt a célt szolgálja, hogy a felhasználó (saját bankjától) letöltött tranzakciós adatokat tartalmazó Excel fájlt feldolgozza és felvigye a táblázatba illetve a rendszer adatbázisába. Ezáltal tudunk felvinni új tranzakciókat az alkalmazásba. Ha rákattintunk az Importálás gombra. Egy legördülő lista fog megjelenni ahol ki kell választanunk melyik Banktól szeretnénk tranzakciókat importálni.



Ábra: Tranzakció menüpont Importálás funkciója

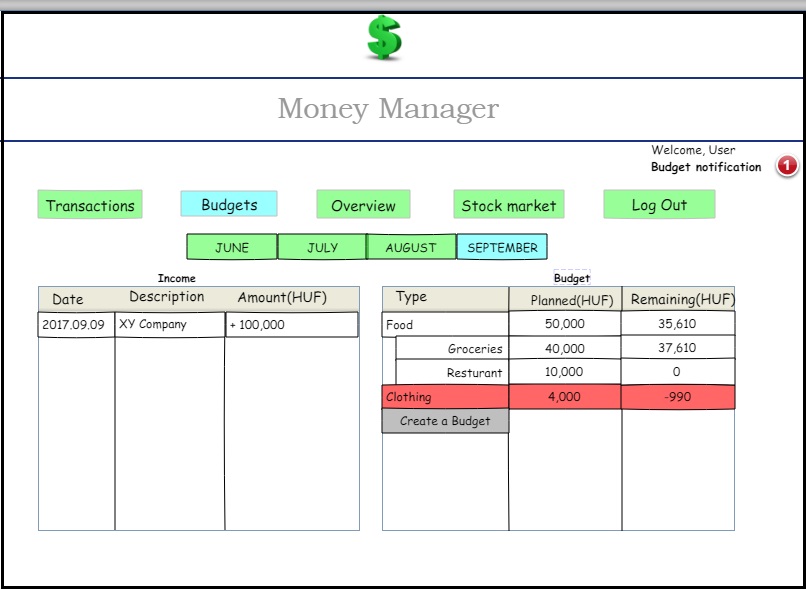
Miután kiválasztottunk bankunkat megjelenik egy Fájl megnyitása gomb, ezt megnyomva kiválaszthatjuk sajátgépünkről a beimportálni kívánt Excel fájlt. A sikeres kiválasztás után a rendszer feldolgozza a fájlban lévő adatokat, amiket elment az adatbázisába is.

A rendszer kiszűri azokat az importálni kívánt tranzakciókat, amik már szerepelnek az adatbázisban, ezzel megoldva a redundáns adattárolás problémáját. Ellenkező esetben a felhasználó hibásan követné nyomon a személyes pénzügyeit. Az ismétlődő tranzakciókról az alkalmazás értesítést is küld a felhasználónak (Messagebox), illetve az értesítésben láthatjuk az új, sikeresen elmentett tranzakciók számát is.

Az Importálás gomb alatt megjelenítésre kerül az utolsó importálás dátuma is, de csak abban az esetben, ha beszúrásra került egy új tranzakció.

## Pénztárca

A Pénztárca menüpont a jobb átláthatóság kedvéért két táblázatot tartalmaz. Míg az egyik táblázat a számlánkra bejövő pénzt tartalmazza (Fizetés táblázat) a másik táblázat a felhasználó által beállított „Pénztárcát” és annak jelenlegi állását tartalmazza (Pénztárca táblázat).



Ábra: Pénztárca menüpont két táblázata

A Fizetés táblázat: tartalmazza azokat a tranzakciókat az aktuálisan kiválasztott hónapból ahol a felhasználónk számlájára pénz érkezett. A táblázat tartalmaz három oszlopot:

* Tranzakció dátuma: A tranzakció „végbemenetelének” ideje (ÉÉÉÉ. HH. NN) például 2017.11.11
* Leírás: Tartalmazza a tranzakció közleményét (vásárlás eseten ez a cég neve ahol vásároltunk) vagy az ellenoldali nevet (például fizetés megkapása esetén a cég neve ahol dolgozunk).
* Érték: Tartalmazza a tranzakció értékét (Forintban). Ha fizetésünk jön például +5,000.

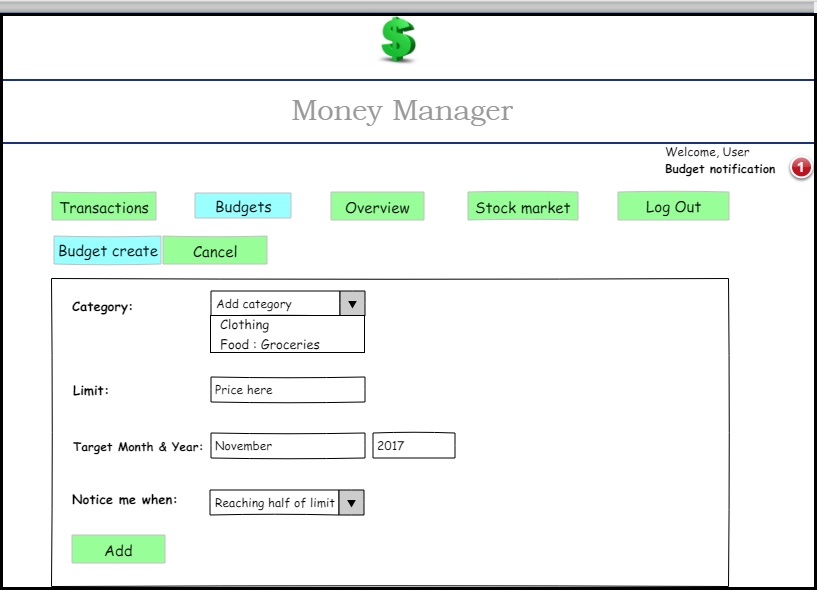
A Pénztárca táblázat: teljes mértékben a felhasználó készíti el és a neki megfelelő értékekkel töltheti fel. A Pénztárca táblázat három oszloppal rendelkezik:

* Típus: A „Pénztárca” típusa, ebben az oszlopban szerepelhetnek főkategóriák és alkategóriák is. Például Autó a főkategória az alkategóriák pedig üzemanyag és tisztítás.
* Tervezett: Ez az oszlop az aktuális kategória tervezett maximális költését tartalmazza forintban. A költés értékét teljes mértékben a felhasználó tölti ki és szerkesztheti kedve szerint. A felhasználó nem tud ebbe az oszlopba negatív értéket beírni, ha megpróbálja az alkalmazásértesítést küld.
* Még rendelkezésre áll: A még rendelkezésre álló oszlop tartalmazza az aktuális kategóriára beállított még rendelkezésre álló pénzösszeget forintban. Ez az oszlop a rendszer által kiszámított értéket tartalmazza, tehát a felhasználó nem szerkesztheti. A felhasználó által beállított Tervezett összegből levonja azoknak a tranzakcióknak az összegét, amelyeknek megegyezik a kategóriája az aktuális kategóriával.

Ha a felhasználó túllépi az általa megszabott korlátot az aktuális sor pirosra színeződik és a jobb felső sarokban frissül a felhasználó értesítéseinek száma, vagy ha közel van a korlát átlépéséhez színváltozás következik be (zöldről-sárgára). A zöld szín azt az esetet jelöli, amikor a felhasználó egyik pénztárca korlátot sem lépte túl és még közel sincs hozzá.

A Pénztárca menüpont még tartalmaz négy gombot a táblázatok felett ahol az aktuális illetve az ezelőtti hónapok szerepelnek, ezen gombok által a felhasználó meg tudja nézni a múlt havi pénztárcáit is. Alapértelmezés szerint az oldal mindig az aktuális hónapot választja ki.

Valamint a táblázat tartalmaz egy Új Pénztárca hozzáadása gombot, amire ha a felhasználó rákattint, egy új oldal jelenik meg ahol új Pénztárca korlátokkal tudja bővíteni a táblázatát. Egy új Pénztárca létrehozásánál a felhasználó megadhatja a tetszőleges kategóriát egy legördülő listából, a limitet, hogy maximálisan mennyit költhetünk az adott kategóriában, illetve az időpontot hogy meddig legyen aktív az adott Pénztárca, valamint egy legördülő listából választhatunk, hogy mikor értesítsen minket a rendszer, például mikor elértük a limitünk felét. Ha a Pénztárca kategóriánál a legördülő listából a kategória hozzáadását választjuk lehetőségünk van új kategóriákkal bővítjük rendszerünket.



Ábra: Új Pénztárca hozzáadása

Ha a felhasználó túllépi az általa megszabott korlátot a Pénztárca táblázat aktuális sora pirosra színeződik és a jobb felső sarokban frissül a felhasználó értesítéseinek száma, vagy ha közel van a korlát átlépéséhez színváltozás következik be az értesítés színe szempontjából zöldről-sárgára. (A zöld szín azt az esetet jelöli, amikor a felhasználó egyik pénztárca korlátot sem lépte túl és még közel sincs hozzá.)

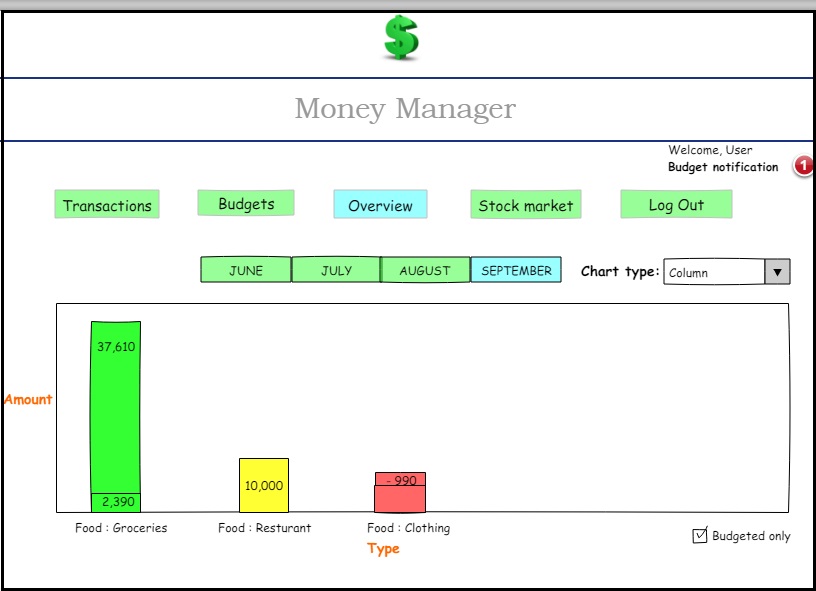
A Pénztárca menüpont még tartalmaz négy gombot a táblázatok felett ahol az aktuális illetve az ezelőtti hónapok szerepelnek, ezen gombok által a felhasználó meg tudja nézni a múlt havi pénztárcáit is. Alapértelmezés szerint az oldal mindig az aktuális hónapot választja ki..

## Áttekintés

Az áttekintés menüpont tartalmaz egy grafikont, amiben szerepel a felhasználó által beállított „Pénztárca” jelenlegi helyzete az adott hónapban.

A felhasználó a grafikon felett láthat négy gombot, amik a hónapokat jelölik. Ezekre rákattintva válthat bizonyos hónapok között és nézheti meg a pénztárcák alakulását. Alapértelmezés szerint az oldal mindig az aktuális hónapot választja ki. Ezzel is segítve a pénzügyeink alakulásainak nyomon követését. Meg tudja nézni mire költött a legtöbbet, milyen kategóriában lépte túl az általa megszabott keretet, mire költött a legkevesebbet, mennyi pénzt spórolt az adott hónapban.

Továbbá a táblázatok fölött láthatunk egy legördülő listát ahol a megjelenített grafikon stílusát választhatjuk ki.



Ábra: Áttekintés menüpont oszlopos grafikon megjelenítés

A fenti ábrán láthatjuk, hogy jelen esetben az oszlopos megjelenítés van kiválasztva, vízszintes tengelyen láthatjuk a bizonyos oszlopokban megjelenített pénztárcák típusát, valamint a függőleges tengelyen láthatjuk a költött pénz mennyiségét.

Továbbá a grafikont tartalmazó táblázat alatt láthatunk egy jelölőnégyzetet, ami jelen esetben be is van pipálva, ezáltal a mi esetünkben csak azok a költések jelennek, meg amikre pénztárca korlát is szabva van. Ha ez a funkció nincs kipipálva az összes tranzakció, költés, jövedelem megjelenik a táblázatunkban.

Az oszlopok négy színt vehetnek fel a pénztárca jelenlegi helyzete alapján:

* Piros: akkor, ha a felhasználó túlköltött az általa létrehozott korláton, ha a felhasználó az oszlopra húzza, a kurzort láthatja mennyivel költött többet a kelleténél.
* Sárga: akkor az azt jelenti, hogy a felhasználó pontosan annyit költött az adott hónapban, mint amennyit beállított magának az adott kategóriára.
* Zöld: akkor a felhasználó az adott kategóriában még nem érte el az általa megszabott korlátot, annak csak egy bizonyos részét használta fel.
* Fehér: akkor a felhasználó az adott kategóriában még nem költött az aktuális hónapban.

## Részvények

A részvények menüpontban a felhasználó két opció közül választhat:

* A rendszer listázza ki a Vásárolt részvényeket.

Vagy

* A rendszer listázza ki az Eladott részvényeket.

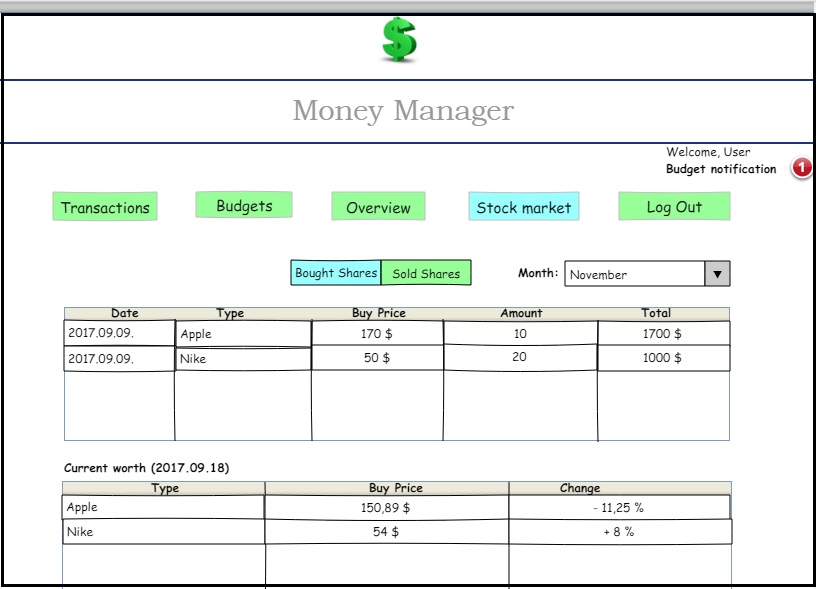
A Vásárolt részvények gombra kattintva a felhasználónak két táblázat fog megjelenni. Az felső táblázat tartalmazni fogja (Vásárolt részvények) a részvényeket, amiket vásárolt. Méghozzá öt oszlopban:

* Dátum: Mikor vásárolta a részvényeket. (ÉÉÉÉ. HH. NN például 2017.09.31)
* Típus: A vásárolt részvény neve, milyen cég részvényei. (például IBM, Apple, Adidas)
* Vételár: A részvény akkori árfolyama, amikor a felhasználó vásárolta őket (dollárban).
* Mennyiség: Hány darab részvényt vásárolt a felhasználó.
* Végösszeg: Az akkori árfolyam és a darabszám szorzata a végösszeg.

Az alsó táblázat tartalmazza az aktuális napi részvényárfolyamokat, méghozzá azon részvényeket, amiket a felhasználó is vásárolt. E táblázat tartalmazni fogja az aktuális árfolyamot és a százalékos változást a felhasználó által vásárolt részvények árfolyama között.

* Típus: A vásárolt részvény neve, milyen cég részvényei. (például IBM, Apple, Adidas)
* Vételár: A részvény akkori árfolyama, amikor a felhasználó vásárolta őket (dollárban).
* Változás: A részvény vásárlási árfolyamához képest milyen százalékos eltérése van az aktuális árfolyamhoz képest.

Nem utolsó sorban láthatunk a Vásárolt részvények táblázat fölött egy legördülő listát, aminek segítségével ki tudjuk, választani melyik hónap adatait szeretnénk kilistázni.



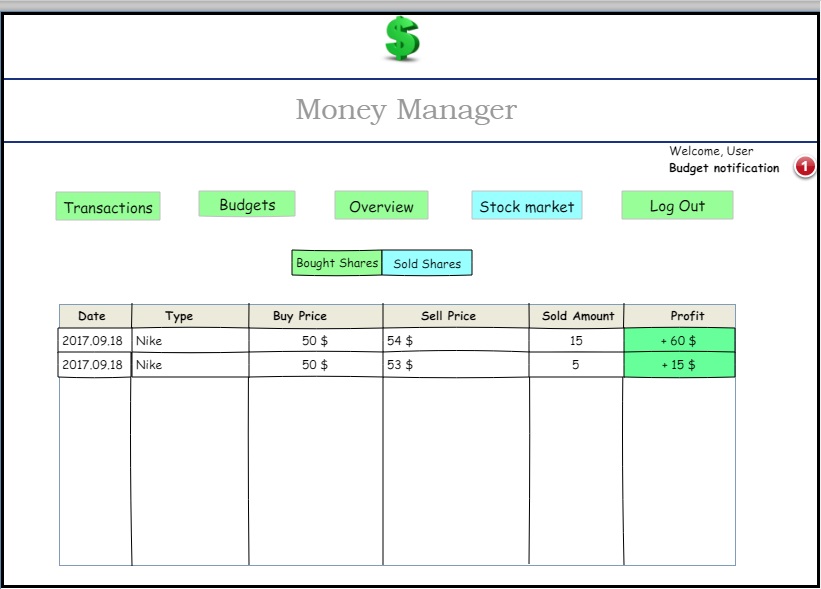
Ábra: Részvények menüpont, Vásárolt részvények

Ha a felhasználó az Eladott részvények gombra kattint egy hatoszlopú táblázat fog megjelenni:

* Dátum: A részvények eladási dátuma. (ÉÉÉÉ. HH. NN például 2017.09.31)
* Típus: Az eladott részvény neve, milyen cég részvényei. (például IBM, Apple, Adidas)
* Vételár: A részvény akkori árfolyama, amikor a felhasználó vásárolta őket (dollárban).
* Eladási ár: A részvény akkori árfolyama, amikor a felhasználó eladta őket (dollárban).
* Darabszám: Azon eladott részvények száma, amelyek ugyan olyan Eladási áron keltek el.
* Profit: A részvény vételárához képest mennyi a nyereség az eladás után (dollárban).

A profit oszlop elemei három színt vehetnek fel:

* Piros: Ha a vételi árnál olcsóbban adtuk el a részvényeket, tehát veszteség keletkezett.
* Sárga: Ha a vételi árral azonos értékben adtuk el a részvényeket.
* Zöld: Ha a vételi árnál drágábban adtuk el a részvényeket, tehát nyereség keletkezett



Ábra: Részvények menüpont, Eladott részvények

## Kilépés

A felhasználó a kilépés menüpontra kattintva kijelentkezik az oldalról és újra a Kezdőlap oldal fog megjelenni.

# Összesítés

A Kiolvasó algoritmus optimalizálása illetve gyorsítása, hatékonyság növelése illetve sok számlakivonat típuson történő tesztelés.

Tőzsdei adatok importálási lehetősége a programba. Ami tartalmazza az eladott illetve vásárolt részvények adatait. Valamint a nyereséget és a veszteséget, illetve a jelenlegi árfolyamokat is láthatják majd a felhasználók.

Tranzakciók dátum szerint történő szűrése illetve a kiválasztott hónap alapján történő megjelenítése az Adatbázis menüpontban a felhasználó számára, amiket gombokkal tud kiválasztani.

Grafikonok készítése a felhasználó pénzügyi szokásairól a kiválasztott hónapban, bizonyos keresési feltételeket figyelembe véve.

**Végső cél:** Mivel rengeteg bank van Magyarországon és az exportált Excel fájlok formátuma mindegyiknél különböző, ezért a fő jövőbeli cél egy olyan beolvasó algoritmus írása, amelyik banktól függetlenül be tudja olvasni az exportált Excel fájlok tartalmát. Ezzel megspórolva azt a hosszadalmas munkát ahol mindegyik Banktól exportált Excel fájlokat különböző algoritmusokat kelljen be olvasni.