Budapesti Műszaki és Gazdaságtudományi Egyetem
Témalabor (BMEVIAUAL00)

Webes alkalmazás készítése

Témalabor

Tőzsde

Konzulens:

Albert István

Csapattagok:

Szabó Egon Róbert	DEQGWW
Szabó Viktor Ákos	VMPJLJ
Varga Ádám Marcell	E22H8P

Tartalomjegyzék

1. Feladatleírás

A félév során egy webalkalmazást próbálunk majd elkészíteni, melyben a felhasználónak (avagy befektetőnek) lehetősége lesz különböző részvényeket valamint crypto-kat vásárolni és eladni.

2. Feladatspecifikáció

2.2. Alkalmazás felépítése

Az alkalmazás különböző nézetekből (2.4. Wireframek) fog állni, melyekhez (többnyire) eltérő funkciók fognak társulni felhasználói típus specifikusan (a felhasználói típusokról részletesebben 2.2.1. Felhasználók).

Publikus használat: Ekkor a felhasználó még nincs bejelentkezve. Ilyenkor hozzáférhet a fő (nyitó) oldalhoz, továbbá a különböző részvények és crypto-k listájához, viszont nem vásárolhat belőlük, csupán az árfolyam alakulását figyelheti meg.

Befektetői használat: Ekkor a felhasználó bejelentkezett befektetői fiókjába. A bejelentkezést a login nézeten tudja majd megtenni a kért adatok megadásával. Amennyiben még nincs fiókja, akkor a register nézeten lesz lehetősége létrehozni egyet a kért adatok megadásával. Számára éppúgy elérhetőek azon nézetek, mint a publikus használatban, ugyanakkor így már lehetősége lesz a különböző tételek vásárlására, valamint nyomonkövetni azok alakulását a befektető portfóliójában.

A befektetőnek lehetősége lesz egyes részvényeket és crypto-kat megjelölni **kedvenc**ként, majd egy checkbox segítségével megjeleníteni azokat.

Admin használat: Ekkor adminisztrátorként van a felhasználó bejelentkezve. Számára csak a felhasználók listája lesz elérhető.

2.2.1. Felhasználók

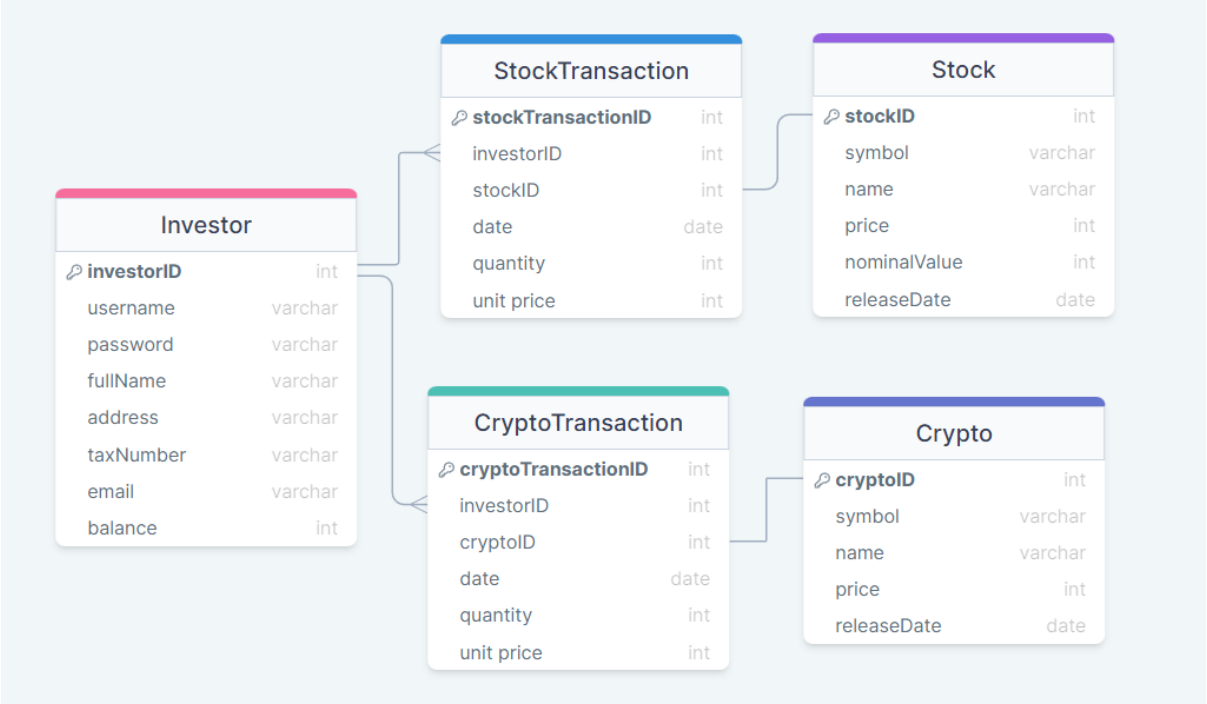
Az alkalmazásban kétféle felhasználó típus lesz:

- **befektető (investor)**: A befektető képes különböző részvényeket valamint crypto-kat vásárolni és eladni, továbbá egyenlegét tetszőleges mennyiségben feltölteni.
- **admin**: Az admin képes lesz különböző befektetőket eltávolítani.

2.2.2. Technikai megfontolások

A tőzsdén elérhető tételek alapjául valódi részvények és crypto-k szolgálnak majd. A szükséges információkat egy tőzsdei API fogja szolgáltatni.

2.3. Adatbázis



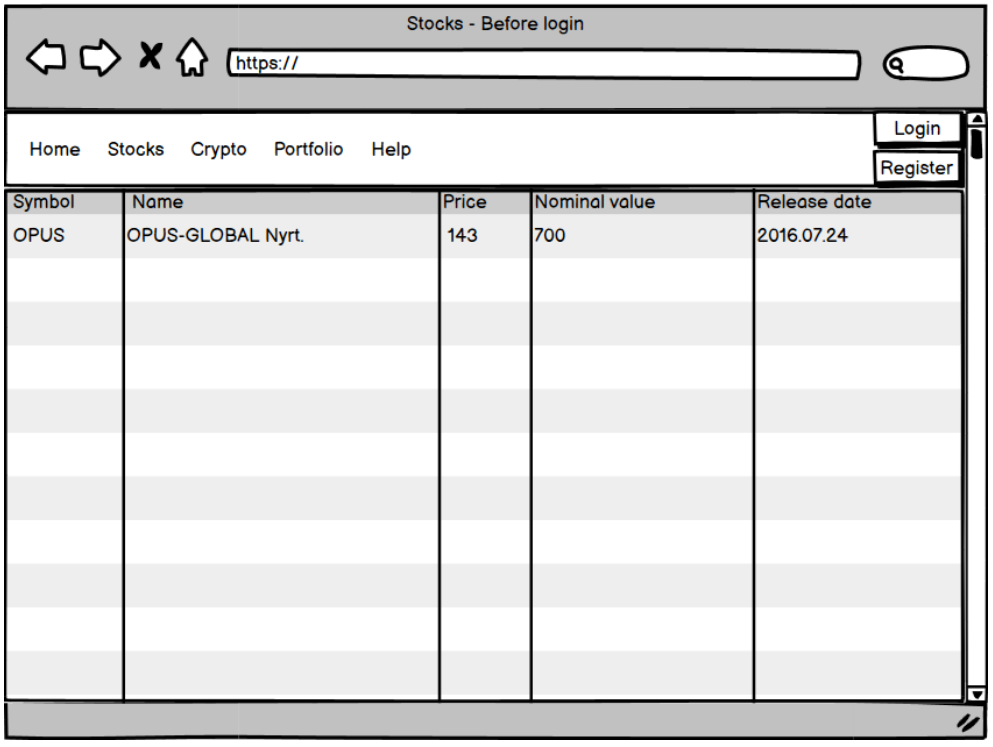
Adatbázis diagram

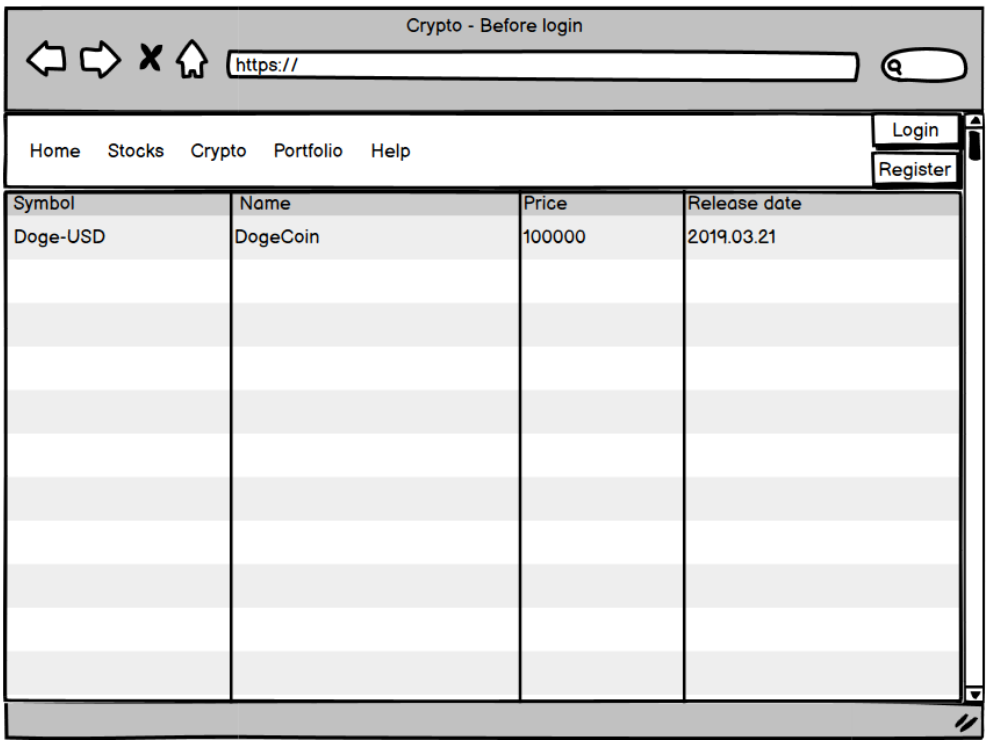
Entitások az adatbázisban:

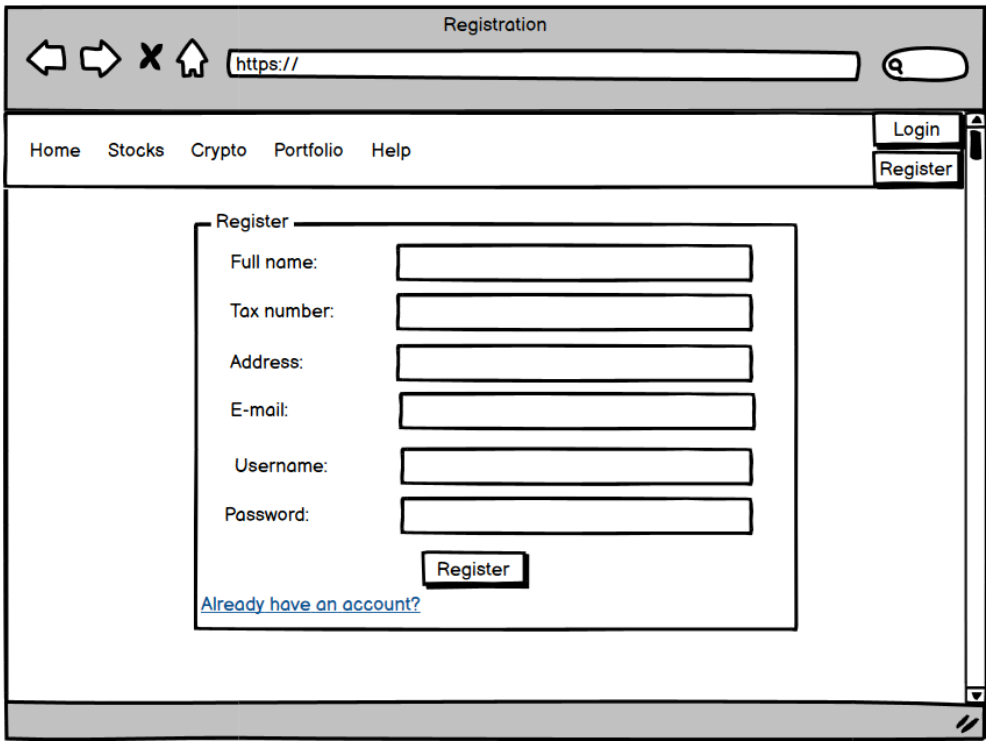
- **Investor**: A befektető személyes adatait tárolja.
  Attribútumai:
  - investorID: generált elsődleges kulcs az Investor táblához.
  - userName: A befektető felhasználóneve. A bejelentkezéshez szükséges.
  - password: A befektető jelszava. A bejelentkezéshez szükséges.
  - fullName: A befektető teljes neve.
  - address: A befektető lakcíme.
  - taxNumber: A befektető adószáma.
  - email: A befektető email címe.
  - balance: A befektető egyenlege. Ezt bármikor feltöltheti majd.
- **Stock**: A részvény adatait tárolja.
  Attribútumai:
  - stockId: generált elsődleges kulcs a Stock táblához.
  - symbol: A kibocsátó cég rövidítése.
  - name: A kibocsátó cég neve.
  - price: A részvény jelenlegi ára.
  - nominalValue: A részvény névleges (első kibocsátáskor jegyzett) értéke.
  - releaseDate: A részvény kibocsátásának dátuma.

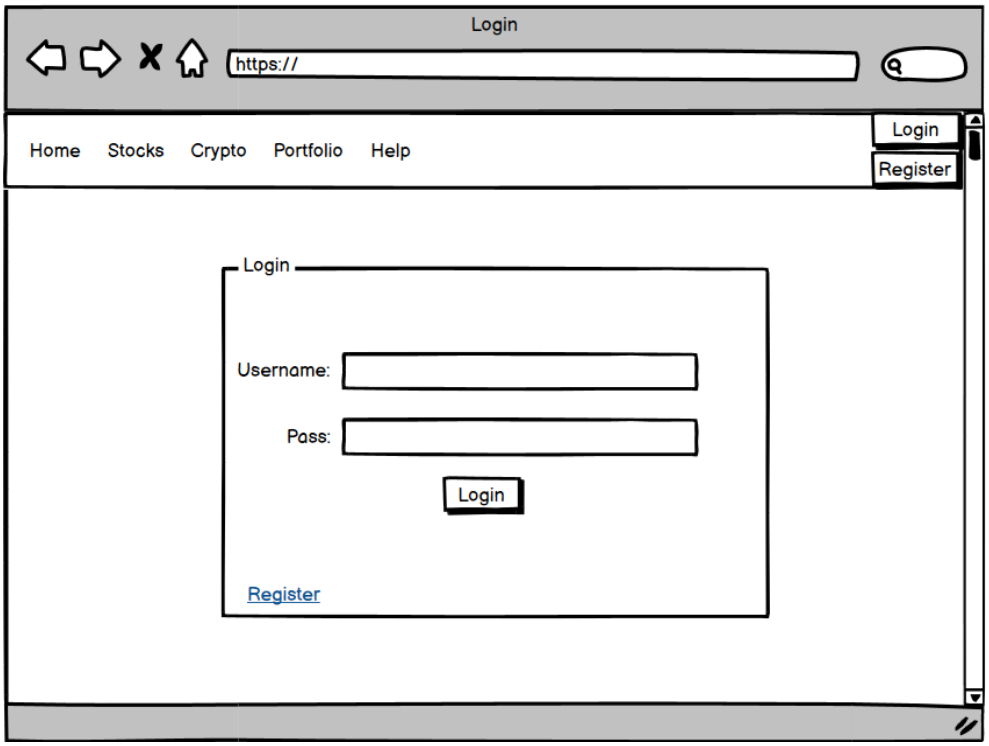
- **StockTransaction**: A részvény tranzakciók adatait tárolja.
  Attribútumai:
    - stockTransactionId: generált elsődleges kulcs a StockTransaction táblához.
    - investorID: idegen kulcs az Investor táblából.
    - stockID: idegen kulcs a Stock táblából.
    - date: A vásárlás dátuma.
    - quantity: A vásárolt mennyiség.
    - unitPrice: A vásárolt részvény egységára (vásárláskor).
- **Crypto**: A crypto adatait tárolja.
  Attribútumai:
    - cryptoId: generált elsődleges kulcs a Crypto táblához.
    - symbol: A kibocsátó cég rövidítése.
    - name: A kibocsátó cég neve.
    - price: A crypto jelenlegi ára.
    - nominalValue: A crypto névleges (első kibocsátáskor jegyzett) értéke.
    - releaseDate: A crypto kibocsátásának dátuma.
- **CryptoTransaction**: A crypto tranzakciók adatait tárolja.
  Attribútumai:
    - cryptoTransactionId: generált elsődleges kulcs a CryptoTransaction táblához.
    - investorID: idegen kulcs az Investor táblából.
    - cryptoID: idegen kulcs a Crypto táblából.
    - date: A vásárlás dátuma.
    - quantity: A vásárolt mennyiség.
    - unitPrice: A vásárolt crypto egységára (vásárláskor).

2.4. Wireframek

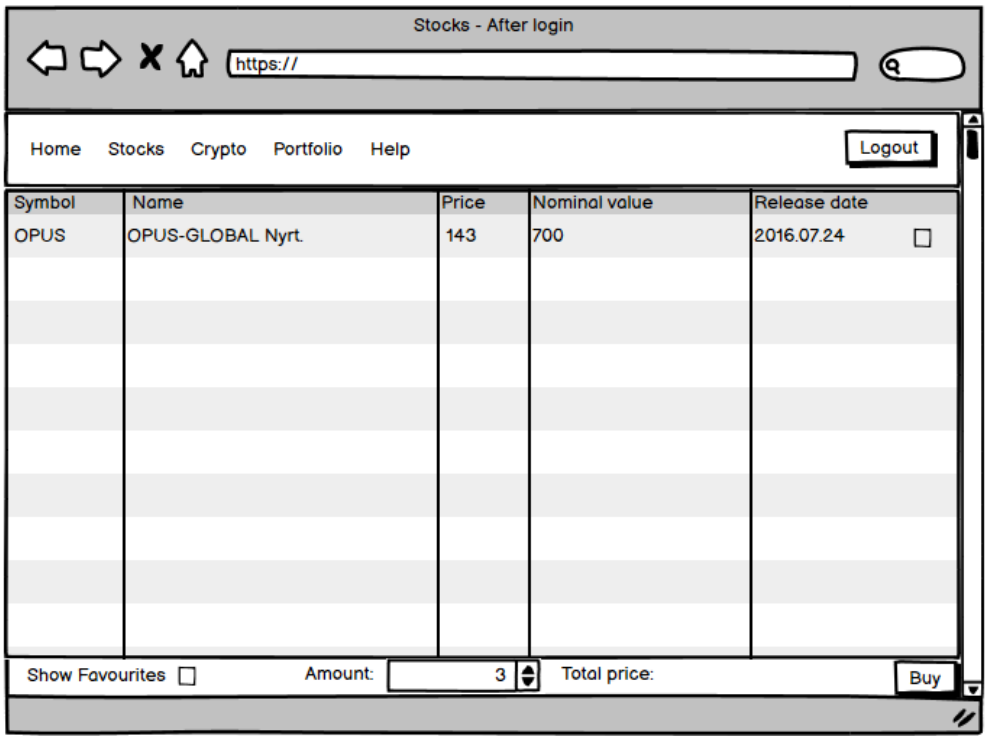


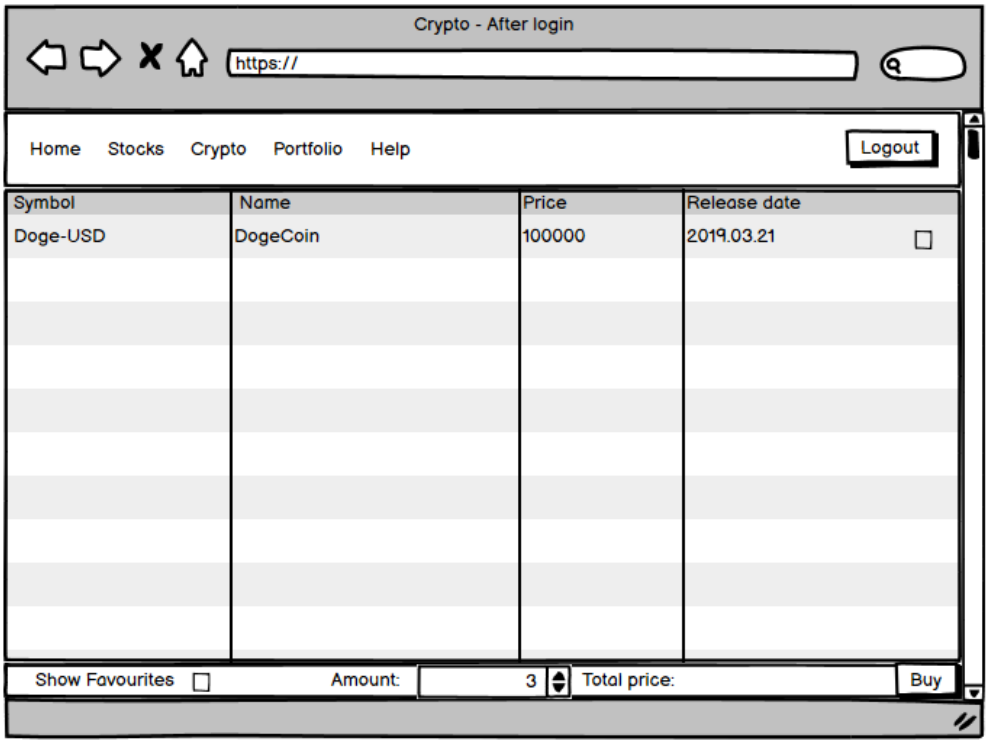

Crypto - Before login
https://
Home
Stocks
Crypto
Portfolio
Help
Login
Register
Symbol
Name
Price
Release date
Doge-USD
DogeCoin
100000
2019.03.21

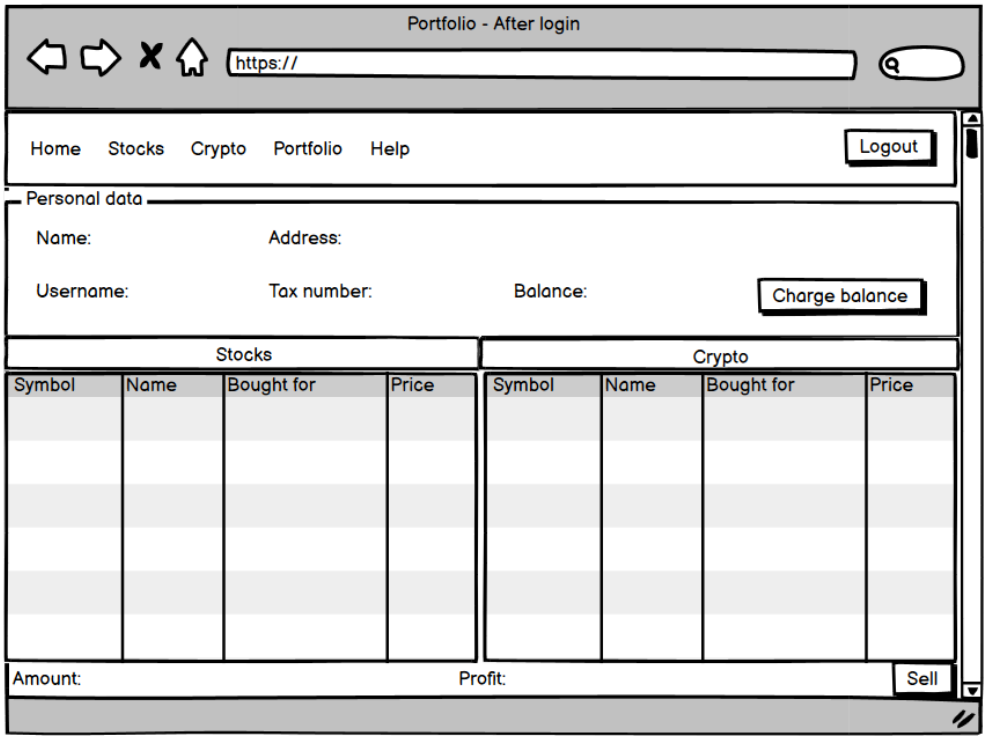

Registration
https://
Home
Stocks
Crypto
Portfolio
Help
Login
Register
Register
Full name:
Tax number:
Address:
E-mail:
Username:
Password:
Register
Already have an account?

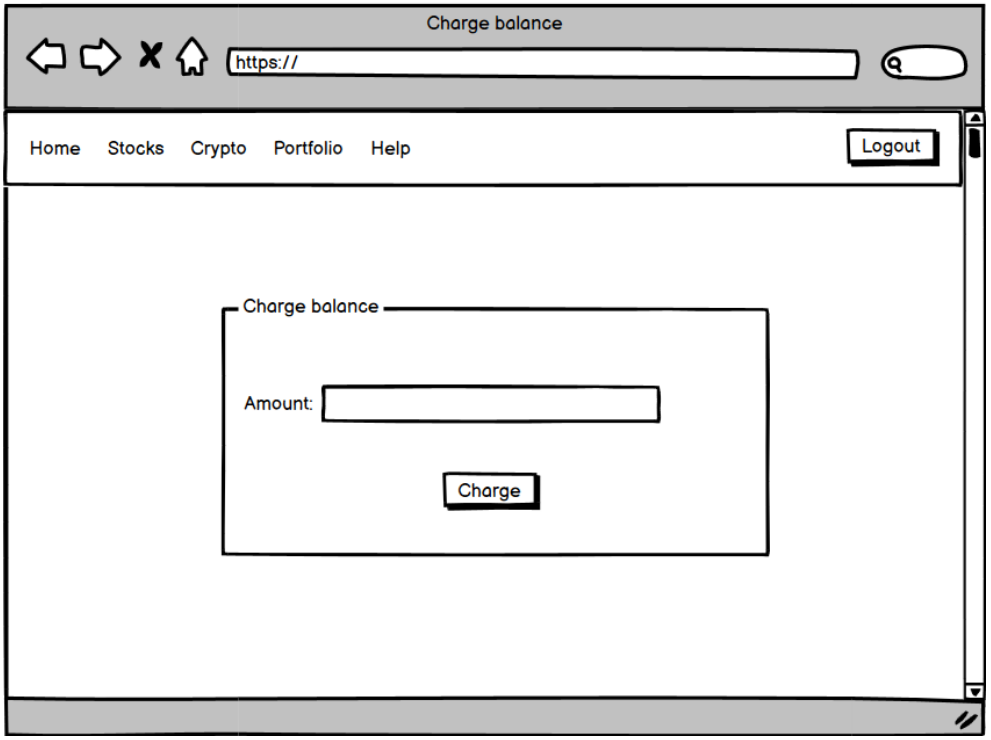

Login
https://
Home
Stocks
Crypto
Portfolio
Help
Login
Register
Login
Username:
Pass:
Login
Register

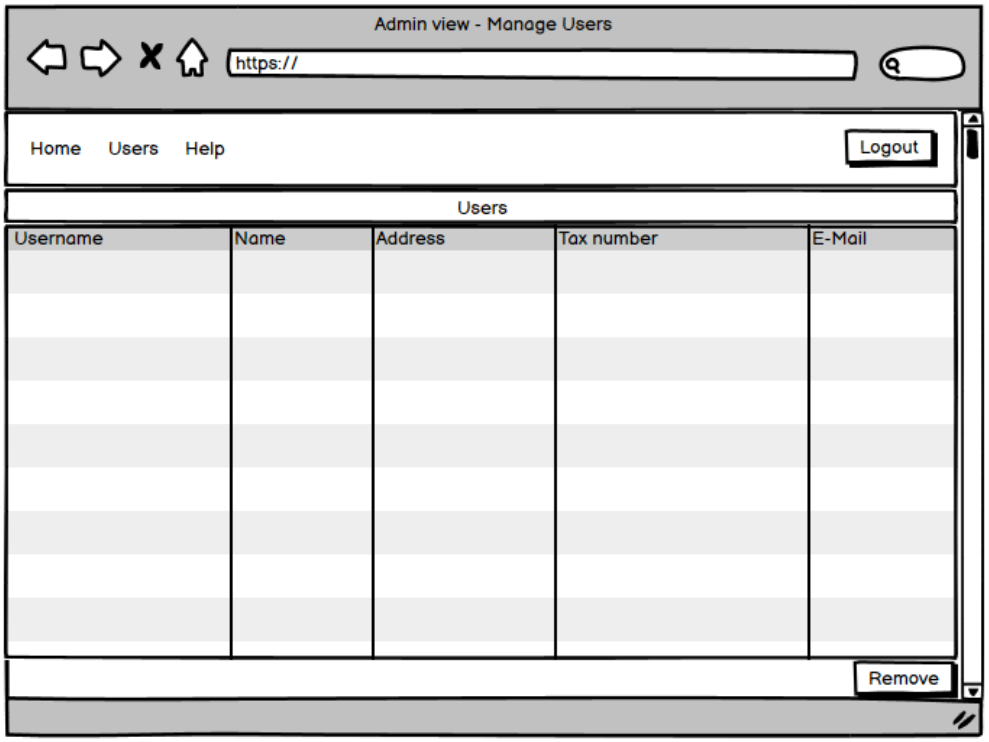

Főoldal - bejelentkezés után
https://
Home
Stocks
Crypto
Portfolio
Help
Logout
Kép a háttérben
Szöveg a képen


Stocks - After login
https://
Home
Stocks
Crypto
Portfolio
Help
Logout
Symbol
Name
Price
Nominal value
Release date
OPUS
OPUS-GLOBAL Nyrt.
143
700
2016.07.24
Show Favourites
Amount:
3
Total price:
Buy


Crypto - After login
https://
Home
Stocks
Crypto
Portfolio
Help
Logout
Symbol
Name
Price
Release date
Doge-USD
DogeCoin
100000
2019.03.21
Show Favourites
Amount:
3
Total price:
Buy


Portfolio - After login
https://
Home
Stocks
Crypto
Portfolio
Help
Logout
Personal data
Name:
Address:
Username:
Tax number:
Balance:
Charge balance
Stocks
Crypto
Symbol
Name
Bought for
Price
Symbol
Name
Bought for
Price
Amount:
Profit:
Sell


Charge balance
https://
Home
Stocks
Crypto
Portfolio
Help
Logout
Charge balance
Amount:
Charge


Admin view - Manage Users
https://
Home
Users
Help
Logout
Users
Username
Name
Address
Tax number
E-Mail
Remove


Help - Before/after login
https://
Home
Stocks
Crypto
Portfolio
Help
Logout
Sablon szöveg

4. Fejlesztői dokumentáció

Jelen blokkban kerülnek kifejtésre az alkalmazást felépítő elemek működése valamint azok felelősségei.
Az alkalmazás felépítését tekintve két nagyobb részre bontható:

- **backend (szerver oldal)**: Az alkalmazás logikájáért valamint az adatok tárolásáért felelős.
- **frontend (kliens oldal)**: A felhasználóval történő kapcsolattartásért felelős.

Az ezekhez tartozó technikai információk a továbbiakban kifejtésre kerülnek.

4.1. Adatbázis

4.1.1. Technikai részletek

Backend oldalon az adatok tárolásához a *Microsoft SQL Server (Database File)* -re esett a választás.
Főbb okok a választás mellett:

- Megszokott relációs adatbázis.
- SQL nyelv.
- Könnyű összehangolás és kezelés a szerveroldali keretrendszerrel.

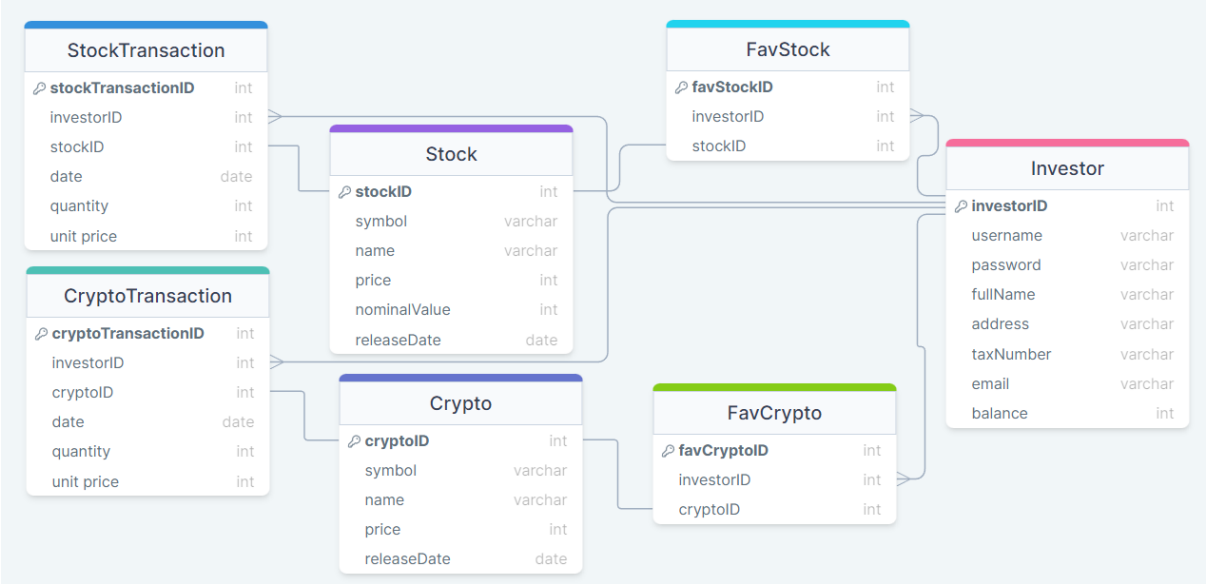
Továbbá felhasználásra került még az ugyancsak Microsoft által biztosított *Entity Framework* megnevezésű objektum-relációs (ORM) leképező keretrendszer is, amely sokat egyszerűsít az adatbázisban történő lekérdezéseknél a már meglévő model elemekre építve.

4.1.2. Új táblák

Az alábbiakban csak az (2.3. Adatbázis részhez képest) új, valamint módosított entitások kerülnek kifejtésre.
Új / módosított entitások:

- **Stock**: A *nominalValue* attribútum kikerült a táblából.
- **Crypto**: A *nominalValue* attribútum kikerült a táblából.
- **FavStock**: Az egyes felhasználókhoz tartozó kedvenc részvényeket tárolja.
  Attribútumai:
  - favStockID: generált elsődleges kulcs a FavStock táblához.
  - investorID: idegen kulcs az Investor táblából.
  - stockID: idegen kulcs az Stock táblából.
- **FavCrypto**: Az egyes felhasználókhoz tartozó kedvenc cryptokat tárolja.
  Attribútumai:
  - favCryptoID: generált elsődleges kulcs a FavCrypto táblához.
  - investorID: idegen kulcs az Investor táblából.
  - cryptoID: idegen kulcs az Crypto táblából.

4.1.3. Új adatbázis diagram



4.2. Szerver oldal

Az alábbiakban kerülnek kifejtésre a különböző szerver oldalon lévő osztályok felelőssége és egyes metódusainak leírása, valamint a backend technikai részletei.

4.2.1. Technikai részletek

Szerver oldalon a Microsoft által biztosított *ASP.NET* webalkalmazás kertrendszerre került a választás.
Főbb okok a választás mellett:

- Megszokott, objektumorientált szemlélet.
- C# nyelv.

4.2.2. Osztályok leírása

Könyvtárak:

- **API**: A külső (egyéb külső oldalról származó) Api hívások kezeléséért felelős osztályok találhatóak meg benne.
- **Controllers**: A belső (a backend és frontend közötti kommunikáció) Api hívások kezeléséért felelős osztályok találhatóak meg benne.
- **DAOs**: Az adatbázisban történő lekérdezésekért és módosításokért felelős osztályok találhatóak meg benne.
- **Database**: Az adatbázis fájl (database.mdf és) és log fájl (database_log.ldf) található meg benne. Ezekben kerülnek tárolásra az adatok.
- **Models**: Az adatbázisból Entity Framework használatával leképezett modellek találhatóak meg benne.

A továbbiakban az egyes könyvtárak alá tartozó osztályok az alábbi formátumban lesznek láthatóak:

<könyvtár neve>/<osztály neve>
Ahol nincs megadva könyvtár név ott az adott osztály a projekt gyökérkönyvtárában található meg.

4.2.2.1. API/TwelveDataApiHandler

Felelősség:
Ez az osztály felelős a TwelveData-ról való valós idejű részvény, valamint crypto árak lekérdezésére.

Attribútumok:
- **apiKey: string:** A TwelveData-ról való adat lekéréshez szükséges kulcs.

Metódusok:
- **+GetApiPrice(string symbol): ApiPrice:** Lekérdezi a paraméterként megadott részvény vagy crypto valós idejű értékét, majd egy ApiPrice objektummal tér vissza (amely tartalmazza ezt az adatot).

4.2.2.2. API/ApiPrice

Felelősség:
A TwelveData-ról származó adat deszerializációjához szükséges osztály.

Attribútumok:
- **+Price: string:** Az Api kérést követően a real-time ár tárolására szolgáló mező.
- **+Code: int:** Az Api kérés sikerességének tárolására szolgáló mező. Ez egy számkód amely jelzi a kérés állapotát.

Metódusok:
-

4.2.2.3. PriceUpdater

Felelősség:
A valós idejű részvény, valamint crypto árak folytonos (periodikus) lekérdezésére szolgáló osztály.

Attribútumok:
- **+isComplete: bool:** Egy flag, ami jelzi, hogy befejeződött-e a taszk.

Metódusok:
- **+async Start(List<Stock> stocks, List<Crypto> cryptos):** Elindítja az árak folytonos lekérdezésére szolgáló taszkot.
- **UpdatePrices(List<Stock> stocks, List<Crypto> cryptos): Task:** Állandó időközönként generál egy Api hívást a TwelveData felé, majd frissíti az adatbázisban szereplő részvények és cryptok árát.

Megjegyzés:
Mivel a TwelveData kreditrendszert használ az egyes kérések szabályozására (1 perc alatt max 8 db 1 kredit értékű kérés engedélyezett), ezért az árfolyamok frissítése csak meghatározott időközönként történik (jelenleg 65 másodpercenként).

4.2.2.4. Models/Crypto

Felelősség:
A crypto adatok tárolására szolgáló osztály.

Attribútumok:

- **CryptoId: int:** A Crypto táblából származó elsődleges kulcs.
- **Symbol: string:** A kibocsátó cég rövidítése.
- **Name: string:** A kibocsátó cég neve.
- **Price: double:** A crypto jelenlegi ára.
- **ReleaseDate: string:** A crypto kibocsátásának dátuma.

Metódusok:

-

4.2.2.5. Models/CryptoTransaction

Felelősség:
Az egyes crypto tranzakciók tárolására szolgáló osztály.

Attribútumok:

- **CryptoTransactionId: int:** A CryptoTransaction táblából származó elsődleges kulcs.
- **InvestorId: int:** Az Investor táblából származó idegen kulcs.
- **CryptoId: int:** A Crypto táblából származó idegen kulcs.
- **Date: string:** A vásárlás dátuma.
- **Quantity: int:** A vásárolt mennyiség.
- **UnitPrice: double:** A vásárolt crypto egységára (vásárláskor).

Metódusok:

-

4.2.2.6. Models/Stock

Felelősség:
A részvény adatok tárolására szolgáló osztály.

Attribútumok:

- **StockId: int:** A Stock táblából származó elsődleges kulcs.
- **Symbol: string:** A kibocsátó cég rövidítése.
- **Name: string:** A kibocsátó cég neve.
- **Price: double:** A részvény jelenlegi ára.
- **ReleaseDate: string:** A részvény kibocsátásának dátuma.

Metódusok:

-

4.2.2.7. Models/StockTransaction

Felelősség:
Az egyes részvény tranzakciók tárolására szolgáló osztály.

Attribútumok:

- **StockTransactionId: int:** A StockTransaction táblából származó elsődleges kulcs.

- **InvestorId: int:** Az Investor táblából származó idegen kulcs.
- **StockId: int:** A Stock táblából származó idegen kulcs.
- **Date: string:** A vásárlás dátuma.
- **Quantity: int:** A vásárolt mennyiség.
- **UnitPrice: double:** A vásárolt részvény egységára (vásárláskor).

Metódusok:

-

4.2.2.8. Models/Investor

Felelősség:

Az egyes felhasználók (befektetők) eltárolására szolgáló osztály.

Attribútumok:

- **InvestorId: int:** Az Investor táblából származó elsődleges kulcs.
- **UserName: string:** A befektető felhasználóneve. A bejelentkezéshez szükséges.
- **Password: string:** A befektető jelszava. A bejelentkezéshez szükséges.
- **FullName: string:** A befektető teljes neve.
- **Address: string:** A befektető lakcíme.
- **TaxNumber:** string: A befektető adószáma.
- **Email: string:** A befektető email címe.
- **Balance: double:** A befektető aktuális egyenlege.

Metódusok:

-

4.2.2.9. Models/FavStock

Felelősség:

Az egyes felhasználókhoz tartozó kedvenc részvények eltárolására szolgáló osztály.

Attribútumok:

- **FavStockId: int:** A FavStock táblából származó elsődleges kulcs.
- **InvestorId: int:** Az Investor táblából származó idegen kulcs.
- **StockId: int:** A Stock táblából származó idegen kulcs.

Metódusok:

-

4.2.2.10. Models/FavCrypto

Felelősség:

Az egyes felhasználókhoz tartozó kedvenc cryptok eltárolására szolgáló osztály.

Attribútumok:

- **FavCryptoId: int:** A FavCrypto táblából származó elsődleges kulcs.
- **InvestorId: int:** Az Investor táblából származó idegen kulcs.
- **CryptoId: int:** A Crypto táblából származó idegen kulcs.

Metódusok:

-

4.2.2.11. Models/DatabaseContext

Felelősség:
A modellek és az adatbázis ORM leképezésért felelős osztály.
Attribútumok:
EntityFramwork által generáltak.
Metódusok:
EntityFramwork által generáltak.

4.2.2.12. DAOs/CryptoDAO

Felelősség:
A Crypto táblában történő lekéredzésekre és módosításokra szolgáló osztály.
Attribútumok:
-
Metódusok:
- **+UpdatePrice(int cryptoID, double newPrice):** A megadott cryptoID-ra illeszkedő cryptonak változtatja meg az árát a newPrice alapján.
- **+GetCryptos(): List<Crypto>:** Visszatér a táblában szereplő összes cryptoval.
- **+GetUnitPrice(int cryptoID):** double: Visszatér a megadott cryptoID-ra illeszkedő cryptonak az egység árával.
- **+GetCryptoBySymbol(string symbol):** Crypto: Visszatér a meghatározott symbol-al rendelkező Crypto objektummal.

4.2.2.13. DAOs/CryptoTransactionDAO

Felelősség:
A CryptoTransaction táblában történő lekéredzésekre és módosításokra szolgáló osztály.
Attribútumok:
-
Metódusok:
- **+GetCryptoTransactions(int investorID): List<CryptoTransaction>:** Visszatér az adatbázisban lévő összes crypto tranzakcióval.
- **+InsertCryptoTransaction(CryptoTransaction cryptoTransaction):** Beilleszti a táblába a megadott cryptoTransaction objektumot.
- **+RemoveCryptoTransaction(CryptoTransaction cryptoTransaction):** Eltávolítja a táblából az adott cryptoTransaction objektumot.
- **+RemoveCryptoTransactions(int investorId):** Eltávolítja a táblából az adott invesorId-re illeszkedő objektumot (rekordot).
- **+GetCryptoTransactionById(int transactionId): CryptoTransaction:** Visszatér a megadott transactionId-val rendelkező tranzakcióval.
- **+GetPortfolioTransactionData(int investorId): List<PortfolioTransactionData>:** Visszatér az investorId-ra illeszkedő felhasználó összes (crypto) tranzakciójával a PortfolioTransactionData objektumon keresztül.

4.2.2.14. DAOs/StockDAO

Felelősség:
A Stock táblában történő lekéredzésekre és módosításokra szolgáló osztály.

Attribútumok:
-

Metódusok:
- **+UpdatePrice(int stockID, double newPrice):** A megadott stockID-ra illeszkedő részvénynek változtatja meg az árát a newPrice alapján.
- **+GetStocks(): List<Stock>:** Visszatér a táblában szereplő összes részvénnyel.
- **+GetUnitPrice(int stockID):** double: Visszatér a megadott stockID-ra illeszkedő részvénynek az egység árával.
- **+GetStockBySymbol(string symbol):** Stock: Visszatér a meghatározott symbol-al rendelkező Stock objektummal.

4.2.2.15. DAOs/StockTransactionDAO

Felelősség:
A StockTransaction táblában történő lekéredzésekre és módosításokra szolgáló osztály.

Attribútumok:
-

Metódusok:
- **+GetStockTransactions(int investorID): List<StockTransaction>:** Visszatér az adatbázisban lévő összes részvény tranzakcióval.
- **+InsertStockTransaction(StockTransaction stockTransaction):** Beilleszti a táblába a megadott stockTransaction objektumot.
- **+RemoveStockTransaction(StockTransaction stockTransaction):** Eltávolítja a táblából az adott stockTransaction objektumot.
- **+RemoveStockTransactions(int investorId):** Eltávolítja a táblából az adott invesorId-re illeszkedő objektumot (rekordot).
- **+GetStockTransactionById(int transactionId): StockTransaction:** Visszatér a megadott transactionId-val rendelkező tranzakcióval.
- **+GetPortfolioTransactionData(int investorId): List<PortfolioTransactionData>:** Visszatér az investorId-ra illeszkedő felhasználó összes (részvény) tranzakciójával a PortfolioTransactionData objektumon keresztül.

4.2.2.16. DAOs/InvestorDAO

Felelősség:
Az Investor táblában történő lekéredzésekre és módosításokra szolgáló osztály.

Attribútumok:
-

Metódusok:
- **+GetInvestors(): List<Investor>:** Visszatér a táblában szereplő összes felhasználóval (befektetővel).

- **+GetInvestor(string userName, string password): Investor:** Visszatér az adott felhasználónévvel és jelszóval rendelkező felhasználóval.
- **+GetInvestorById(int investorId): Investor:** Visszatér a megadott Id-val rendelkező felhasználóval.
- **+InsertInvestor(Investor investor):** Beilleszt egy új felhasználót a táblába.
- **+RemoveInvestor(Investor investor):** Eltávolítja a megadott felhasználót a táblából.
- **+UpdateBalance(int investorID, double balanceAddition):** Frissíti a megadtott Id-val rendelkező felhasználó egyenlegét. A balanceAddition értéke hozzáadódik az aktuális egyenleg értékéhez. Az előjel ennek megfelelően választandó (pénz hozzáadás esetén +, levonás esetén -)

4.2.2.17. DAOs/FavStockDAO

Felelősség:
A FavStock táblában történő lekéredzésekre és módosításokra szolgáló osztály.
Attribútumok:
-
Metódusok:
- **+GetFavStocks(int investorID): List<Stock>:** Visszatér a táblában lévő összes bejegyzéssel.
- **+InsertFavStockEntry(FavStock favStockEntry):** Beilleszt a táblába egy új kedvenc részvényt.
- **+RemoveFavStockEntry(FavStock favStockEntry):** Eltávolítja a favStockEntry objektumot a táblából.
- **+RemoveFavStockEntries(int investorId):** Eltávolítja a táblából az adott invesorId-re illeszkedő objektumot (rekordot).
- **+GetFavStockEntry(int investorId, int stockId): FavStock:** Visszatér az adott investorId-ra és stockId-ra illeszkedő bejegyzéssel.

4.2.2.18. DAOs/FavCryptoDAO

Felelősség:
A FavCrypto táblában történő lekéredzésekre és módosításokra szolgáló osztály.
Attribútumok:
-
Metódusok:
- **+GetFavCryptos(int investorID): List<Crypto>:** Visszatér a táblában lévő összes bejegyzéssel.
- **+InsertFavCryptoEntry(FavCrypto favCryptoEntry):** Beilleszt a táblába egy új kedvenc crypto-t.
- **+RemoveFavCryptoEntry(FavCrypto favCryptoEntry):** Eltávolítja a favCryptoEntry objektumot a táblából.
- **+RemoveFavCryptoEntries(int investorId):** Eltávolítja a táblából az adott invesorId-re illeszkedő objektumot (rekordot).

- **+GetFavCryptoEntry(int investorId, int cryptoId): FavCrypto:** Visszatér az adott investorId-ra és cryptoId-ra illeszkedő bejegyzéssel.

4.2.2.19. Controllers/CryptosController

Felelősség:
A crypto oldalhoz (frontend) tartozó api hívásokért felelős osztály.
Attribútumok:
-
Metódusok:
- **+GetCryptos(int investorID): List<Crypto>:** Visszatér az összes crypto-val a frontend felé.
- **+BuyStock(CryptoTransactionData data): double:** A data paraméter alapján a befektető vásárol egy megadott cryptoból valamennyi darabot.

4.2.2.20. Controllers/CryptoTransactionData

Felelősség:
A frontend felől érkező vásárlási adatok deszerializálására szolgáló adat osztály.
Attribútumok:
- **+InvestorId: int:** A befektető Id-ja.
- **+Symbol: string:** A vásárolni kívánt crypto symbol-ja.
- **+UnitPrice: double:** Az aktuális egységár a vásárláskor.
- **+Quantity: int :** A vásárolni kívánt mennyiség.

Metódusok:
-

4.2.2.21. Controllers/FavCryptosController

Felelősség:
A crypto oldalhoz (frontend) tartozó a kedvenc cryptok kezeléséhez szükséges api hívásokért felelős osztály.
Attribútumok:
-
Metódusok:
- **+GetFavCryptos(int investorId): List<Crypto>:** Visszatér a megadott felhasználóhoz tartozó kedvenc cryptok listájával.
- **+ManageFavCrypto(FavCryptoData data):** Egy megadott crypto (a data paraméterből) felvétele a kedvencek közé vagy eltávolítása onnan, függően attól hogy aktuálisan kedvenc-e.

4.2.2.22. Controllers/FavCryptoData

Felelősség:
A frontend felől érkező kedvenc crypto kezeléshez szükséges adatok deszerializálására szolgáló adat osztály.

Attribútumok:

- **+InvestorId: int:** A befektető Id-ja.
- **+Symbol: string:** A vásárolni kívánt crypto symbol-ja.
- **+IsFavourite: bool:** Flag, ami jelzi hogy adott crypto jelenleg kedvenc-e.

Metódusok:

-

4.2.2.23. Controllers/StocksController

Felelősség:

A stock oldalhoz (frontend) tartozó api hívásokért felelős osztály.

Attribútumok:

-

Metódusok:

- **+GetStocks(int investorID): List<Stock>:** Visszatér az összes részvénnyel a frontend felé.
- **+BuyStock(StockTransactionData data): double:** A data paraméter alapján a befektető vásárol egy megadott részvényből valamennyi darabot.

4.2.2.24. Controllers/StockTransactionData

Felelősség:

A frontend felől érkező vásárlási adatok deszerializálására szolgáló adat osztály.

Attribútumok:

- **+InvestorId: int:** A befektető Id-ja.
- **+Symbol: string:** A vásárolni kívánt részvény symbol-ja.
- **+UnitPrice: double:** Az aktuális egységár a vásárláskor.
- **+Quantity: int :** A vásárolni kívánt mennyiség.

Metódusok:

-

4.2.2.25. Controllers/FavStocksController

Felelősség:

A stock oldalhoz (frontend) tartozó a kedvenc részvények kezeléséhez szükséges api hívásokért felelős osztály.

Attribútumok:

-

Metódusok:

- **+GetFavStocks(int investorId): List<Stock>:** Visszatér a megadott felhasználóhoz tartozó kedvenc részvények listájával.
- **+ManageFavStock(FavStockData data):** Egy megadott részvény (a data paraméterből) felvétele a kedvencek közé vagy eltávolítása onnan, függően attól hogy aktuálisan kedvenc-e.

4.2.2.26. Controllers/FavStockData

Felelősség:
A frontend felől érkező kedvenc részvény kezeléshez szükséges adatok deszerializálására szolgáló adat osztály.

Attribútumok:
- **+InvestorId: int:** A befektető Id-ja.
- **+Symbol: string:** A vásárolni kívánt részvény symbol-ja.
- **+IsFavourite: bool:** Flag, ami jelzi hogy adott részvény jelenleg kedvenc-e.

Metódusok:
-

4.2.2.27. Controllers/InvestorsController

Felelősség:
A manageusers oldalhoz (frontend) tartozó api hívásokért felelős osztály.

Attribútumok:
-

Metódusok:
- **+GetInvestors(): List<Investor>:** Visszatér az összes befektető listájával a frontend felé.
- **+RemoveInvestor(Investor investor):** A frontendről érkező kérés alapján eltávolítja a meghatározott befektetőt az adatbázisból.

4.2.2.28. Controllers/LoginController

Felelősség:
A login oldalhoz (frontend) tartozó api hívásokért felelős osztály.

Attribútumok:
-

Metódusok:
- **+Login(LoginCredentials loginCredentials): Investor:** A loginCredentials paraméter alapján megkísérel bejelentkezni, és amennyiben sikerül visszatér az adott felhasználóhoz tartozó Investor objektummal. Ellenkező esetben pedig null-al.

4.2.2.29. Controllers/LoginCredentials

Felelősség:
A frontend felől érkező bejelentkezéshez szükséges adatok deszerializálására szolgáló adat osztály.

Attribútumok:
- **+UserName: string:** A bejelentkezéshez szükséges felhasználónév.
- **+Password: string:** A bejelentkezéshez szükséges jelszó.

Metódusok:
-

4.2.2.30. Controllers/RegistrationController

Felelősség:
A register oldalhoz (frontend) tartozó api hívásokért felelős osztály.

Attribútumok:
-

Metódusok:
- **Registration(RegistrationForm registrationForm): Investor:** A registrationForm paraméter alapján megkísérli létrehozni a felhasználót, és amennyiben sikerül visszatér az adott felhasználóhoz tartozó Investor objektummal. Ellenkező esetben pedig null-al.

4.2.2.31. Controllers/RegistrationForm

Felelősség:
A frontend felől érkező regisztrációhoz szükséges adatok deszerializálására szolgáló adat osztály.

Attribútumok:
- **+UserName: string:** Felhasználónév.
- **+Password: string:** Jelszó.
- **+FullName: string:** A felhasználó teljes neve.
- **+Address: string:** A felhasználó lakcíme.
- **+TaxNumber: string:** A felhasználó adószáma.
- **+Email: string:** A felhasználó email címe.

Metódusok:
-

4.2.2.32. Controllers/PortfolioController

Felelősség:
A portfolio oldalhoz (frontend) tartozó api hívásokért felelős osztály.

Attribútumok:
-

Metódusok:
- **+UpdateBalance(BalanceUpdateForm balanceUpdateForm):** A frontendről érkező kérés alapján növeli a befektető egyenlegét.
- **+GetStockTransactions(int investorId): List<PortfolioTransactionData>:** Visszatér a befektetőhöz tartozó összes (részvény) tranzakcióval a frontend felé.
- **+GetCryptoTransactions(int investorId): List<PortfolioTransactionData>:** Visszatér a befektetőhöz tartozó összes (crypto) tranzakcióval a frontend felé.
- **+SellStockTransaction(PortfolioTransactionData data): double:** A frontendről érkező kérés alapján eladja az adott tranzakcióban szereplő részvényeket, majd visszatér azzal az értékkel, amennyivel nőtt vagy csökkent az egyenlege.
- **+SellCryptoTransaction(PortfolioTransactionData data): double:** A frontendről érkező kérés alapján eladja az adott tranzakcióban szereplő cryptokat, majd visszatér azzal az értékkel, amennyivel nőtt vagy csökkent az egyenlege.

4.2.2.33. Controllers/BalanceUpdateForm

Felelősség:
A frontend felől érkező egyenleg növeléséhez szükséges adatok deszerializálására szolgáló adat osztály.

Attribútumok:
- **+InvestorId: int:** A befektető Id-ja.
- **+BalanceAddition: double:** A plusz összeg amellyel a befektető növelni kívánja egyenlegét.

Metódusok:
-

4.2.2.34. Controllers/PortfolioTransactionData

Felelősség:
A frontend felől érkező eladáshoz szükséges adatok deszerializálására szolgáló adat osztály.

Attribútumok:
- **+TransactionId: int:** A tranzakció Id-ja.
- **+Symbol: string:** A részvény/crypto symbol-ja.
- **+Name: string:** A részvény/crypto neve.
- **+Date: string:** A vásárlás dátuma.
- **+Quantity: int:** A vásárolt mennyiség.
- **+OldPrice: double:** Az ár, amin vásárolva lett.
- **+CurrentPrice: double:** A jelenlegi ára.

Metódusok:
-

4.2.2.35. Controllers/Program

Felelősség:
Az alkalmazás indításáért / futásáért felelős osztály.

Attribútumok:
-

Metódusok:
- **+Main(string[] args):** Elindítja a PriceUpdater-t, majd a back és frontendet.

4.3. Kliens oldal

4.3.1. Technikai részletek

Az alkalmazás kliens oldalon a React könyvtárat használja TypeScript-tel és a Bootstrap framework React-es változatával kiegészítve. A TypeScriptre a típusos jellege miatt esett a választás, hiszen ez sok debugolást kiküszöböl. A Bootstrap framework segítségével pedig egyszerűen építhetünk stílusos weboldalt a CSS túlzott ismerete nélkül is.

4.3.2. Komponensek leírása

Könyvtárak:

- **css-files**: Itt találhatóak a weblapok formázásáért felelős fájlok (jelenleg ez egyedül a styles.css)
- **images**: Ez a könyvtár tartalmazza a weboldalon megjelenő képeket.
- **models**: Itt találhatóak azon interfészek melyek a backanden lévő modelleket, illetve egyéb csak a frontenden szükséges modelleket írnak le.
- **pages**: Itt találhatóak azon komponensek, melyek a felhasználó által látható oldalak megjelenítéséért és egyéb frontend logika (api hívások, ütemezés…) megvalósításáért felelősek.
- **components**: Minden egyéb, az egyes oldalakon megjelenő rész komponensek ebben a könyvtárban találhatóak.

A továbbiakban az egyes könyvtárak alá tartozó komponensek az alábbi formátumban lesznek láthatóak:

<könyvtár neve>/<komponens neve>

Ahol nincs megadva könyvtár név ott az adott komponens a projekt gyökérkönyvtárában található meg, ami jelen kontextusban a **ClientApp/src/** könyvtárnak felel meg.
Az **attribútumok** alatt a komponensek által használt useHook-ok (pl. useState, useContext…), illetve egyéb a komponenshez tartozó globális változók fognak szerepelni.
A metódusokban az adott komponensekben szereplő függvények, illetve a useEffect (ez egy callback függvény ami az első render után, valamint a dependency listában megadott elemek változása esetén tüzel) hookok fognak szerepelni.
A továbbiakban a html kódok nem kerülnek kifizetésre (az egyes komponensek return ága).

4.3.2.1. models/balanceUpdateForm.model

Felelősség:

A backend felél küldött, egyenleg növeléséhez szükséges adatok szerializálására szolgáló interfész.

Attribútumok:

- **investorId:** number: A befektető Id-ja.
- **balanceAddition:** number: A plusz összeg amellyel a befektető növelni kívánja egyenlegét.

Metódusok:

-

4.3.2.2. models/crypto.model

Felelősség:

A frontenden történő crypto adatok megjelenítéséhez szükséges interfész.

Attribútumok:

- **symbol: string:** A kibocsátó cég rövidítése.
- **name: string:** A kibocsátó cég neve.
- **price: number:** A crypto jelenlegi ára.
- **releaseDate: string:** A crypto kibocsátásának dátuma.

Metódusok:

-

4.3.2.3. models/investor.model

Felelősség:

Az egyes felhasználók adatait leíró interfész.

Attribútumok:

- **investorId: number:** A befektető Id-ja.
- **userName: string:** A befektető felhasználóneve.
- **password: string:** A befektető jelszava.
- **fullName: string:** A befektető teljes neve.
- **address: string:** A befektető lakcíme.
- **taxNumber:** string: A befektető adószáma.
- **email: string:** A befektető email címe.
- **balance: number:** A befektető egyenlege.

Metódusok:

-

4.3.2.4. models/loginCredentials.model

Felelősség:

A backend felél küldött, bejelentkezéshez szükséges adatok szerializálására szolgáló interfész.

Attribútumok:

- **+userName: string:** A bejelentkezéshez szükséges felhasználónév.
- **+password: string:** A bejelentkezéshez szükséges jelszó.

Metódusok:

-

4.3.2.5. models/portfolioTransaction.model

Felelősség:

A backend felél küldött, eladáshoz szükséges adatok szerializálására szolgáló interfész.

Attribútumok:

- **+transactionId: number:** A tranzakció Id-ja.
- **+symbol: string:** A részvény/crypto symbol-ja.
- **+name: string:** A részvény/crypto neve.
- **+date: string:** A vásárlás dátuma.
- **+quantity: number:** A vásárolt mennyiség.
- **+oldPrice: number:** Az ár, amin vásárolva lett.
- **+currentPrice: number**: A jelenlegi ára.

Metódusok:

-

4.3.2.6. models/registrationForm.model

Felelősség:
A backend felél küldött, regisztrációhoz szükséges adatok szerializálására szolgáló interfész.

Attribútumok:
- **+userName: string:** Felhasználónév.
- **+password: string:** Jelszó.
- **+fullName: string:** A felhasználó teljes neve.
- **+address: string:** A felhasználó lakcíme.
- **+taxNumber: string:** A felhasználó adószáma.
- **+email: string:** A felhasználó email címe.

Metódusok:
-

4.3.2.7. models/stock.model

Felelősség:
A frontenden történő részvény adatok megjelenítéséhez szükséges interfész.

Attribútumok:
- **symbol: string:** A kibocsátó cég rövidítése.
- **name: string:** A kibocsátó cég neve.
- **price: number:** A részvény jelenlegi ára.
- **releaseDate: string:** A részvény kibocsátásának dátuma.

Metódusok:
-

4.3.2.8. pages/Crypto

Felelősség:
A crypto oldal megjelenítéséért, és api hívások kezeléséért felelős komponens.

Attribútumok:
- **+cryptos: Crypto[]:** Az oldalon megjelenő összes crypto listája.
- **+favCryptos: Crypto[]:** Az oldalon megjelenő kedvenc cryptok listája.
- **+showFavs: boolean:** Flag, ami jelzi az összes, vagy csak a kedvenc cryptok jelenjenek meg.
- **+showBalance: boolean:** Az oldal frissítésére szolgáló kapcsoló.
- **+favChanged: boolean:** Egy flag, ami jelzi hogy változott kedvencek megjelenítésének állapota.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:
- **+handleChange():** Callback function, ami vátoztatja a showFavs és showBalance állapotát.
- **+favChangedListener():** Callback function, ami változtatja a favChanged állapotát.
- **+getCryptos():** Lekéri backendről az összes vagy csak a kedvenc cryptok listáját.

- **+useEffect(dependencies: showFavs, favChanged):** Kedvenc hozzáadása vagy eltávolítása esetén küld egy api kérést a backend felé.
- **+useEffect(dependencies: showFavs):** 65 másodpercenként meghívja a getCryptos() metódust (folyamatosan frissíti az árfolyamot).

4.3.2.9. pages/Help

Felelősség:
A segítség oldalt megjelenítő komponens.
Attribútumok:
-
Metódusok:
-

4.3.2.10. pages/Home

Felelősség:
A kezdőoldal megjelenítéséért felelős komponens.
Attribútumok:
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:
-

4.3.2.11. pages/Login

Felelősség:
A login oldal megjelenítéséért, és api hívások kezeléséért felelős komponens.
Attribútumok:
- **+loginState: LoginCredentials:** A bejelentkezéshez szükséges adatokat tárolja (felhasználónév, jelszó)
- **+loginSuccess: boolean:** A bejelentkezés állapotát jelző flag.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:
- **+handleChange(evt: React.ChangeEvent<HTMLInputElement>):** A textboxokba történő írás alatt folyamatosan frissíti a loginState értékét.
- **+renderLoginResult():** A bejelentkezés állapotától függően újrarendereli az oldalt.
- **+login():** Bejelentkezéshez szükséges api kérést indít a backend felé, majd visszakap egy investort vagy null-t.

4.3.2.12. pages/ManageUsers

Felelősség:
A manageusers oldal megjelenítéséért, és api hívások kezeléséért felelős komponens.
Attribútumok:

- **+investors: Investor[]:** Az összes felhasználót listája.
- **+investorChanged: boolean:** Flag, ami jelzi, hogy eltávolítottak egy felhasználót.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+getInvestors():** Lekéri az összes felhasználót backendről.
- **+useEffect(dependencies: investorChanged):** Felhasználó törlése esetén frissíti a felhasználók listáját.
- **+investorChangedListener():** Változtatja az investorChanged állapotát amennyiben felhasználót töröltek.

4.3.2.13. pages/Portfolio

Felelősség:

A portfolio oldal megjelenítéséért, és api hívások kezeléséért felelős komponens.

Attribútumok:

- **+viewCrypto: boolean:** Kapcsoló, ami jelzi hogy a részvényeket, vagy a cryptokat jelenítse meg az oldal.
- **+stocksChanged: boolean:** A részvények változását jelző flag.
- **+cryptosChanged: boolean:** A cryptok változását jelző flag.
- **+stockPieChartLoaded: boolean:** Flag, ami jelzi, hogy betöltődött-e a részvények PieChart-ja.
- **+cryptoPieChartLoaded: boolean:** Flag, ami jelzi, hogy betöltődött-e a cryptokPieChart-ja.
- **+stockPieChartData: PieChartData[]:** A részvények PieChartjában megjelenítendő adatok (részvények mennyiségének aránya).
- **+cryptoPieChartData: PieChartData[]:** A cryptok PieChartjában megjelenítendő adatok (cryptok mennyiségének aránya).
- **+stockTransactions: PortfolioTransaction[]:** A részvény tranzakciók listája.
- **+cryptoTransactions: PortfolioTransaction[]:** A crypto tranzakciók listája.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+stocksChangedListener():** Részvények törlését követően változtatja stocksChanged flag állapotát.
- **+cryptosChangedListener():** Cryptok törlését követően változtatja cryptosChanged flag állapotát.
- **+handleViewChange():** Az oldalon történő részvények és cryptok megjelenítésének változtatása során módosítja viewCrypto állapotát.
- **+loadCryptoPieChartData():** Betölti az adatokat a cryptokat megjelenítő PieChartba.
- **+loadStockPieChartData():** Betölti az adatokat a részvényeket megjelenítő PieChartba.
- **+getStockTransactions():** Lekéri backendről a felhasználóhoz tartozó összes részvény tranzakcióját.

- **+getCryptoTransactions():** Lekéri backendről a felhasználóhoz tartozó összes crypto tranzakcióját.
- **+useEffect(dependencies: cryptoPieChartLoaded, cryptosChanged):** Nézetváltás esetén betölti a cryptok PieChartját.
- **+useEffect(dependencies: stockPieChartLoaded, stocksChanged):** Nézetváltás esetén betölti a részvények PieChartját.
- **+useEffect(dependencies: stocksChanged):** 65 másodpercenként, vagy törlés esetén frissíti a részvények tranzakcióinak listáját.
- **+useEffect(dependencies: cryptosChanged):** 65 másodpercenként, vagy törlés esetén frissíti a cryptok tranzakcióinak listáját.

4.3.2.14. pages/Register

Felelősség:

A register oldal megjelenítéséért, és api hívások kezeléséért felelős komponens.

Attribútumok:

- **+registrationState: RegistrationForm:** A regisztrációhoz szükséges adatokat tárolja.
- **+registrationSuccess: boolean:** A regisztráció állapotát jelző flag.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+handleChange(evt: React.ChangeEvent<HTMLInputElement>):** A textboxokba történő írás alatt folyamatosan frissíti a registrationState értékét.
- **+emptyFields(): boolean:** Visszatér azzal, hogy üresek-e a mezők.
- **+renderRegistrationResult():** A regisztráció állapotától függően újrarendereli az oldalt.
- **+registration():** Regisztrációhoz szükséges api kérést indít a backend felé, majd visszakap egy investort vagy null-t.

4.3.2.15. pages/Stocks

Felelősség:

A stock oldal megjelenítéséért, és api hívások kezeléséért felelős komponens.

Attribútumok:

- **+stocks: Stock[]:** Az oldalon megjelenő összes részvények listája.
- **+favStocks: Stock[]:** Az oldalon megjelenő kedvenc részvények listája.
- **+showFavs: boolean:** Flag, ami jelzi az összes, vagy csak a kedvenc részvények jelenjenek meg.
- **+showBalance: boolean:** Az oldal frissítésére szolgáló kapcsoló.
- **+favChanged: boolean:** Egy flag, ami jelzi hogy változott kedvencek megjelenítésének állapota.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+handleChange():** Callback function, ami vátoztatja a showFavs és showBalance állapotát.
- **+favChangedListener():** Callback function, ami változtatja a favChanged állapotát.
- **+getStocks():** Lekéri backendről az összes vagy csak a kedvenc részvények listáját.
- **+useEffect(dependencies: showFavs, favChanged):** Kedvenc hozzáadása vagy eltávolítása esetén küld egy api kérést a backend felé.
- **+useEffect(dependencies: showFavs):** 65 másodpercenként meghívja a getStocks() metódust (folyamatosan frissíti az árfolyamot).

4.3.2.16. components/BuyCryptoModal

Felelősség:

Crypto vásárlás során felugró komponens.

Attribútumok:

- **+show: boolean:** A felugró megjelenítéséért felelős flag.
- **+close: any:** A felugró bezárásáért felelős flag.
- **+crypto: Crypto:** A vásárolni kívánt crypto.
- **+amount: number:** A vásárolni kívánt crypto mennyisége (db).
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+handleChange(event: React.ChangeEvent<HTMLInputElement>):** A vásárolandó mennyiség változtatása során folyamatosan frissíti az amount értékét.
- **+renderAlert():** Újra rendereléshez szükséges callback függvény.
- **+buyCrypto():** A vásárláshoz szükséges api kérést indít a backend felé, majd visszakapja az összeget, amivel frontend oldalon frissítenie kell a felhasználó egyenlegét (le kell vonnia az aktuálisból).

4.3.2.17. components/BuyStockModal

Felelősség:

Részvény vásárlás során felugró komponens.

Attribútumok:

- **+show: boolean:** A felugró megjelenítéséért felelős flag.
- **+close: any:** A felugró bezárásáért felelős flag.
- **+stock: Stock:** A vásárolni kívánt részvény.
- **+amount: number:** A vásárolni kívánt crypto mennyisége (db).
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+handleChange(event: React.ChangeEvent<HTMLInputElement>):** A vásárolandó mennyiség változtatása során folyamatosan frissíti az amount értékét.
- **+renderAlert():** Újra rendereléshez szükséges callback függvény.
- **+buyStock():** A vásárláshoz szükséges api kérést indít a backend felé, majd visszakapja az összeget, amivel frontend oldalon frissítenie kell a felhasználó egyenlegét (le kell vonnia az aktuálisból).

4.3.2.18. components/ChargeBalanceButton

Felelősség:
Egyenleg feltöltéséhez szolgáló felugró ablak, mely tartalmazza a ChargeBalanceModal komponenst.

Attribútumok:
- **+showChargeBalance: boolean:** A ChargeBalanceModal komponens megjelenítését jelző flag.

Metódusok:
-

4.3.2.19. components/ChargeBalanceModal

Felelősség:
Magát az egyenlegfeltöltést intéző komponens.

Attribútumok:
- **+show: boolean:** A komponens megjelenítéséért felelős flag.
- **+close: any:** A komponens bezárásáért felelős flag.
- **+money: number:** Az aktuális egyenleghez hozzáadandó összeg.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:
- **+updateBalance():** Az egyenleg feltöltéséhez szükséges api kérést indít a backend felé, majd frontend oldalon frissíti a felhasználó egyenlegét (hozzáadja a money értékét az aktuális egyenleghez).

4.3.2.20. components/CryptoList

Felelősség:
A backendről származó cryptok listájának megjelenítéséért felelős komponens. Az egyes cryptok adatait a CryptoListRow rész komponensen keresztül jeleníti meg. A CryptoListRow sorok felelnek meg a CryptoList egyes sorainak.

Attribútumok:
- **+cryptos: Crypto[]:** A megjeleníteni kívánt (összes) cryptok listája.
- **+favCryptos: Crypto[]:** A kedvenc cryptok listája.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:
- **+favChangedListener():** Callback függvény, ami a kedvenc cryptok változása esetén hívódik meg.

4.3.2.21. components/CryptoListRow

Felelősség:
Egy meghatározott crypto (sor) megjelenítéséért és a kedvencek kezeléséhez szükséges api hívásokért felelős komponens.

Attribútumok:

- **+crypto: Crypto:** A megjeleníteni kívánt crypto.
- **+favCryptos: Crypto[]:** A kedvenc cryptok listája.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.
- **+showBuyCrypto: boolean:** A crypto vásárlási szándékot (felugróhoz) jelző flag.
- **+favourite: boolean:** Flag, ami jelzi, hogy az adott crypto kedvenc-e.

Metódusok:

- **+handleChange():** Callback function, ami a kedvenc állapotának változás esetén hívódik meg.
- **+manageFavourite():** Api kérést indít a backend felé, majd hozzáadja, vagy eltávolítja az adott crypto-t a kedvencek közül, annak megelőző állapota alapján.
- **+useEffect(dependencies: favCryptos):** A favCryptos lista változása esetén frissíti az adott crypto állapotát (kedvenc/nem kedvenc).

4.3.2.22. components/CustomNavbar

Felelősség:

A navigációs sáv megjelenítéséért és az egyes oldalak közötti navigációért felelős komponens.

Attribútumok:

- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+routeChange():** Kijelentkezés esetén visszadob a kezdőoldalra.

4.3.2.23. components/InvestorList

Felelősség:

A backendről származó befektetők listájának megjelenítéséért felelős komponens. Az egyes befektetők adatait a InvestorListRow rész komponensen keresztül jeleníti meg. A InvestorListRow sorok felelnek meg a InvestorList egyes sorainak.

Attribútumok:

- **+investors: Investor[]:** A megjeleníteni kívánt (összes) befektető listája.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+investorChangedListener():** Callback függvény, ami a befektetők listájának változása esetén hívódik meg.

4.3.2.24. components/InvestorListRow

Felelősség:

Egy meghatározott befektető (sor) megjelenítéséért felelős komponens.

Attribútumok:

- **+investor: Investor:** A megjeleníteni kívánt befektető.

- **+showRemoveInvestor: boolean:** A RemoveInvestorModal komponens megjelenítéséért felelős flag.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+investorChangedListener():** Callback függvény, ami a befektetők listájának változása esetén hívódik meg.

4.3.2.25. components/PortfolioItemList

Felelősség:

A backendről származó tranzakciók (részvény vagy crypto függően attól, hogy melyik van kiválasztva) listájának megjelenítéséért felelős komponens. Az egyes tranzakciók adatait a PortfolioItemListRow rész komponensen keresztül jeleníti meg. A PortfolioItemListRow sorok felelnek meg a PortfolioItemList egyes sorainak.

Attribútumok:

- **+items: PortfolioTransaction[]:** Az egyes tranzakciók listája (részvény vagy crypto).
- **+viewCrypto: boolean:** A részvény és crypto tranzakciók közötti váltásra szolgáló flag.

Metódusok:

- **+stocksChangedListener():** Callback függvény, ami egy részvény tranzakció eltávolítása (eladása) esetén hívódik meg.
- **+cryptosChangedListener():** Callback függvény, ami egy crypto tranzakció eltávolítása (eladása) esetén hívódik meg.

4.3.2.26. components/PortfolioItemListRow

Felelősség:

Egy meghatározott részvény vagy crypto tranzakció (sor) megjelenítéséért felelős komponens.

Attribútumok:

- **+item: PortfolioTransaction:** A megadott tranzakció.
- **+viewCrypto: boolean:** A részvény és crypto tranzakciók közötti váltásra szolgáló flag.
- **+showSellItem: boolean:** A részvény vagy crypto eladási szándékot (felugróhoz) jelző flag.

Metódusok:

- **+stocksChangedListener():** Callback függvény, ami egy részvény tranzakció eltávolítása (eladása) esetén hívódik meg.
- **+cryptosChangedListener():** Callback függvény, ami egy crypto tranzakció eltávolítása (eladása) esetén hívódik meg.

4.3.2.27. components/RemoveInvestorModal

Felelősség:

Felhasználó (befektető) eltávolításához szolgáló felugró ablak.

Attribútumok:

- **+show: boolean:** A felugró megjelenítéséért felelős flag.
- **+close: any:** A felugró bezárásáért felelős flag.
- **+investor: Investor:** Az eltávolítani kívánt befektető.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+investorChangedListener():** Callback függvény, ami a befektető eltávolításának esetén hívódik meg.
- **+removeInvestor():** Api kérést indít a backend felé az adott befektető eltávolítására.

4.3.2.28. components/SellItemModal

Felelősség:

Részvény vagy crypto eladás során felugró komponens.

Attribútumok:

- **+show: boolean:** A felugró megjelenítéséért felelős flag.
- **+close: any:** A felugró bezárásáért felelős flag.
- **+item: PortfolioTransaction:** A kiválasztott tranzakció.
- **+viewCrypto: boolean:** A részvény és crypto tranzakciók közötti váltásra szolgáló flag.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+stocksChangedListener():** Callback függvény, ami egy részvény tranzakció eltávolítása (eladása) esetén hívódik meg.
- **+cryptosChangedListener():** Callback függvény, ami egy crypto tranzakció eltávolítása (eladása) esetén hívódik meg.
- **+sellStock():** A kiválasztott tranzakción belül vásárolt részvények eladására vonatkozó api kérést indít a backend felé.
- **+sellCrypto():** A kiválasztott tranzakción belül vásárolt cryptok eladására vonatkozó api kérést indít a backend felé.

4.3.2.29. components/StockList

Felelősség:

A backendről származó részvények listájának megjelenítéséért felelős komponens. Az egyes részvények adatait a StockListRow rész komponensen keresztül jeleníti meg. A StockListRow sorok felelnek meg a StockList egyes sorainak.

Attribútumok:

- **+stocks: Stock[]:** A megjeleníteni kívánt (összes) részvények listája.
- **+favStocks: Stock[]:** A kedvenc részvények listája.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.

Metódusok:

- **+favChangedListener():** Callback függvény, ami a kedvenc részvények változása esetén hívódik meg.

4.3.2.30. components/StockListRow

Felelősség:

Egy meghatározott részvény (sor) megjelenítéséért és a kedvencek kezeléséhez szükséges api hívásokért felelős komponens.

Attribútumok:

- **+stock: Stock:** A megjeleníteni kívánt részvény.
- **+favStocks: Stock[]:** A kedvenc részvények listája.
- **+userContext: UserContext:** A teljes frontenden lekérhető usercontext, ami az investor-t tárolja.
- **+showBuyStock: boolean:** A részvény vásárlási szándékot (felugróhoz) jelző flag.
- **+favourite: boolean:** Flag, ami jelzi, hogy az adott részvény kedvenc-e.

Metódusok:

- **+handleChange():** Callback function, ami a kedvenc állapotának változás esetén hívódik meg.
- **+manageFavourite():** Api kérést indít a backend felé, majd hozzáadja, vagy eltávolítja az adott részvényt a kedvencek közül, annak megelőző állapota alapján.
- **+useEffect(dependencies: favCryptos):** A favStocks lista változása esetén frissíti az adott részvény állapotát (kedvenc/nem kedvenc).

4.3.2.31. App

Felelősség:

A frontenden lévő oldalak közötti útvonalak kezeléséért és az egész alkalmazáson belül elérhető UserContext (a bejelentkezett felhasználó) biztosításáért felelős komponens.

Attribútumok:

- **+investor: Investor:** Az egész alkalmazásban elérhető (UserContext-en keresztül) bejelentkezett felhasználó (befektető), vagy admin.

Metódusok:

- **+providerValue():** A felhasználói kontextusváltás esetén frissíti az investor-t.