|  |  |
| --- | --- |
| A képen embléma, szimbólum, címerpajzs, jelvény látható  Automatikusan generált leírás | EÖTVÖS LORÁND TUDOMÁNYEGYETEM  Informatikai Kar  Média- és Oktatásinformatika Tanszék |

Kriptovaluták árfolyamának elemzése és korrelációk keresése SAP –

Capireban

|  |  |
| --- | --- |
| Témavezető: Dr Vincellér Zoltán  egyetemi mesteroktató | Szerző: Téglás Tamás  programtervező informatikus BSc. |

Budapest, 2023

Tartalomjegyzék

[Bevezetés 6](#_Toc135758535)

[Témaválasztás indoklása 6](#_Toc135758536)

[Megoldandó feladat 6](#_Toc135758537)

[Felhasználói dokumentáció 7](#_Toc135758538)

[Megoldott probléma rövid leírása 7](#_Toc135758539)

[Felhasznált módszerek rövid leírása 7](#_Toc135758540)

[Program indításához szükséges információk 7](#_Toc135758541)

[Program telepítéséhez szükséges előfeltételek és ismeretek 7](#_Toc135758542)

[Program telepítése 7](#_Toc135758543)

[Program futtatása 7](#_Toc135758544)

[Webes felület használata menüpontokra lebontva 8](#_Toc135758545)

[Home – kezdőképernyő 8](#_Toc135758546)

[Charts –grafikonok megtekintése 8](#_Toc135758547)

[Custom Charts – saját grafikonok 8](#_Toc135758548)

[Bookmarked Charts – könyvjelzőzött grafikonok 8](#_Toc135758549)

[Hidden Charts – rejtett grafikonok 8](#_Toc135758550)

[Grafikonok kinézete, típusai 9](#_Toc135758551)

[Chart operations - grafikonokkal kapcsolatos műveletek 9](#_Toc135758552)

[Create UI5 chart – UI5 grafikon készítése 9](#_Toc135758553)

[Create Chart-js chart – Chart-js grafikon készítése 10](#_Toc135758554)

[Update or Delete chart – grafikon módosítása vagy törlése 11](#_Toc135758555)

[Run script – adatok manipulálása 12](#_Toc135758556)

[Fejlesztői dokumentáció 14](#_Toc135758557)

[Fejlesztői környezet 15](#_Toc135758558)

[Frontend fejlesztése 15](#_Toc135758559)

[Backend fejlesztése 15](#_Toc135758560)

[Szkriptek fejlesztése 16](#_Toc135758561)

[Backend kialakítása, és működése 17](#_Toc135758562)

[Backend technológai oldala – Core Data Services 17](#_Toc135758563)

[Adatmodellek 18](#_Toc135758564)

[Szolgáltatások 22](#_Toc135758565)

[Szoftvert támogató szkriptek működése 27](#_Toc135758566)

[add\_charts.py 27](#_Toc135758567)

[add\_data.py 27](#_Toc135758568)

[analysis.py 29](#_Toc135758569)

[monthly\_charts.py 30](#_Toc135758570)

**EÖTVÖS LORÁND TUDOMÁNYEGYETEM**

**INFORMATIKAI KAR**

**SZAKDOLGOZAT TÉMABEJELENTŐ**

**Hallgató adatai:**

**Név:** Téglás Tamás

**Neptun kód:** BQMR6I

**Képzési adatok:**

**Szak:** programtervező informatikus, alapképzés (BA/BSc/BProf)

**Tagozat** : Nappali

Belső témavezetővel rendelkezem

***Témavezető neve:*** *dr. Vincellér Zoltán*

*munkahelyének neve, tanszéke:* ***ELTE IK, Információs rendszerek Tanszék***

*munkahelyének címe:* ***1117, Budapest, Pázmány Péter sétány 1/C.***

*beosztás és iskolai végzettsége:* ***egyetemi mesteroktató***

**A szakdolgozat címe:** Kriptovaluták árfolyamának elemzése és korrelációk keresése SAP –

Capireban

**A szakdolgozat témája:**

*(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását )*

**A legtöbb kriptovalutákban érdeklődő ember számára bonyolult lehet korszerű, és átlátható tudást szerezni, amely ingyenesen elérhető,**

**vagy nem kell hozzá bármiféle gazdasági háttérrel rendelkeznie. Sok esetben az oldalak melyek átfogó információt nyújtanak egy-egy**

**kriptovalutáról, nem térnek ki, vagy nem nyújtanak olyan funkciókat mellyel többet egyszerre össze lehetne hasonlítani. Gyakran a**

**zöldfülűeknek csak néhány lényegre törő grafikon és ábra kell mely kompakt módon mindent elárul kezdő szinten a kriptovaluta jelenlegi**

**helyzetéről.**

**Szakdolgozatom témája az efféle problémákra próbál megoldást nyújtani, melyben a legtöbb kriptovalutáról valós időben gyors átfogó,**

**egyszerűen olvasható, és személyre szabható grafikonokat és ábrákat biztosít minden felhasználó számára. A felhasználó válogathat az**

**elérhető kriptovaluták árfolyamának megtekintése között és az előre definiált ábrák közt, melyek egy komplexebb képet nyújtanak a piac**

**mélyebb megértéséhez. A felhasználók képesek lesznek a kedvenc árfolyamaikat és ábráikat könyvjelzőzni, mely gyorsabb elérést biztosít**

**azoknak, akik már türelmetlenül várják, hogy befektethessenek. A profiknak testreszabható grafikonokat és ábrákat nyújtunk melyeken**

**különböző valuták korrelációját vizsgálhatják a múltba tekintve. Akik pedig a jövőbe szeretnének látni, egyedi szolgáltatásunk képes**

**bizonyos esetekben megjósolni egy kriptovaluta árfolyamának változását különböző paraméterek figyelembevételével.**

**A fejlesztéshez SAP Capire és CDS technológiákat használok, mellyel egy full-stack webalkalmazást készítek, mely adattároláshoz SAP**

**Hana adatbázist fogok készíteni.**

Budapest, 2022. 11. 30.

# Bevezetés

## Témaválasztás indoklása

A mai világban minden nap találkozhatunk kriptovalutákkal különböző szentimentalitású hírekben. Szemünkbe tűnhetnek dollármilliárdos csalások, átverések, de néha megjelennek az interjúk emberekkel, akik hatalmas vagyonra tettek szert mert 10-15 éve vásároltak fillérekért egy teljesen ismeretlen kriptovalutát, ami mai napra többet ér, mint az arany. Az átlagember nem tudja nyomon követni, hogy melyik kripto mennyit ér a millió közül, a legtöbb pénzügyi weboldal pedig túlságosan sok felesleges információt nyújt.

## Megoldandó feladat

A probléma, hogy az ingyenesen elérhető felületek nagyrésze túl sok információt nyújt a felhasználónak, vagy nem a megfelelő információkat prezentálja, hogy letisztult átfogó képet nyerjen a valuta jelenlegi állapotáról. Célunk, hogy egy egyszerűen használható felületet biztosítsunk a felhasználó részére mely különböző kriptovaluták mozgásának követését, és összehasonlítását teszi lehetővé egy webes felületen keresztül, melyen csak azok az információk jelennek meg melyekre a felhasználó kíváncsi.

# Felhasználói dokumentáció

## Megoldott probléma rövid leírása

A szoftver egy kényelmes letisztult megoldást ad különböző kriptovaluták árfolyamának követésére és összehasonlítására grafikonok segítségével egy webes felületen keresztül. Itt a felhasználó saját maga hozzáadhatja a kriptovalutákat melyeket követni kíván, és hozzáadhat grafikonokat melyek segítségével a felhasználó maga adja meg a követés/elemzés módját.

## Felhasznált módszerek rövid leírása

A felhasználói élményt egy React-ben fejlesztett webes felület segítségével nyújtjuk UI5 és Material UI csomagok segítségével, melyek az irányítási elemek nagyrészét adják, illetve a Chart-js csomag segítségével mely a komplex grafikonokat nyújtja. A szoftver üzleti logikájának egy részét egy capire alkalmazás nyújtja mely alatt egy node.js backend fut. A programhoz tartozik még kettő python script melyek az adatok frissítéséhez, hozzáadásához, és az előrejelzésekhez kellenek.

## Program indításához szükséges információk

### Program telepítéséhez szükséges előfeltételek és ismeretek

Node - ^16.15 (18.15)

cds-dk

### Program telepítése

A szoftvert letöltés után, terminálban a `crypto-gaze` nevű mappába lépve kell futtatni az `npm i` parancsot, mely telepíti a szoftver futásához szükséges csomagokat. Ezután több telepítési teendő nincs.

### Program futtatása

A szoftver futtatásához a `crypto-gaze` mappába lépve terminálban először ki kell adni a `cds deploy –to sqlite` parancsot, amely létrehozz a lokális perzitsztens adatbázisfájlt. Majd ki kell adni a `cds serve` parancsot, mely elindítja a node.js webszervert, és a React frontend alkalmazást a `http://localhost:4004` címen, amelyre böngészővel lépve elérhető a webes felület.

## Webes felület használata menüpontokra lebontva

### Home – kezdőképernyő

Ez a felület jelenik meg elsőnek az alkalmazás indításánál. Itt találhatóak az automatikusan generált, és frissített grafikonok az adatbázisban szereplő kriptovaluták ehavi árfolyamváltozásáról. Kriptovaluta-adatok frissítése, törlése, hozzáadása során, a szoftver indításakor, és hóvégén éjfélkor is változhatnak az itt megjelent grafikonok, amennyiben viszont a felhasználó úgy gondolja, nem adnak hű képet, a `Run Scripts`, menüpontban manuálisan is el lehet indítani a frissítését a grafikonoknak.

### Charts –grafikonok megtekintése

Ebben a menüpontban találhatóak a saját magunk által készített grafikonok megtekintési alpontjai, ahol megtekinthetjük a könyvjelzőzött grafikonjainkat, az elrejtetteket, és az összeset, egy bizonyos rendezés keretein belül.

### Custom Charts – saját grafikonok

Erre a menüpontra kattintás után megjelennek az általunk készített grafikonok alap sorrend szerint: az elsődleges adathalmaz szimbóluma ábécé sorrendben, a könvjelzőzött grafikonokkal kezdve.

Itt meg lehet adni hogy mi más alapján rendezzük a grafikonokat, milyen irányba, illetve, hogy a könyvjelzőzött grafikonokat, soron kívül külön rendezve előrébb jelenjenek meg, mint a több.

Amikor a rendezési szempontok közül valamelyiket módosítjuk, a grafikonok eltűnnek, de a `Show Charts` gom megnyomása után megjelennek.

A rejtett grafikonok itt nem jelennek meg.

### Bookmarked Charts – könyvjelzőzött grafikonok

Ebben a menüpontban találhatóak azok a grafikonok melyeket könyvjelzőztünk a gyorsabb elérés kedvéért.

A rejtett grafikonok itt nem jelennek meg.

### Hidden Charts – rejtett grafikonok

Ebben a menüpontban találhatóak meg a rejtett grafikonok, melyeket nem kívántunk látni a többi közt.

### Grafikonok kinézete, típusai

A program kétfajta grafikont kezel, egy egyszerűbbet a `UI5` csomagból, és egy komplexebbet a `Chart-js` csomagból. Ezekre a típusokra mostantól a csomagok alapján fogunk hivatkozni. A UI5 grafikonok csak egy adathalmazt képesek megjeleníteni, viszont rajtuk találhatóak különböző előrejelzési adatok. A Chart-js grafikonok általában kettő adathalmazt jelenítenek meg, különböző színekkel és különböző léptékekkel. Rajtuk nem lehet előrejelzéseket lekérni.

Mindkét grafikon alján található 2 gomb:

* Hide Chart / Unhide Chart
* Bookmark / Remove Bookmark

Itt az előbbi gomb felel a grafikon elrejtéséért, rákattintva vagy elrejti a program a grafikont, és átnavigál a `Hidden Charts` menüpontra, vagy visszavonja a rejtett státuszát a grafikonnak, és átnavigál a `Custom Charts` menüpontra.

Hasonlóan, a másik gomb könyvjelzőzi a grafikont, és átnavigál a `Bookmarked Charts` menüpontra, vagy leveszi a könvjelzőt a grafikonról és átnavigál a `Custom Charts` menüpontra.

### Chart operations - grafikonokkal kapcsolatos műveletek

Eme menüpont alatt találhatóak a saját magunk által létrehozott grafikonok létrehozásához, módosításához és törléséhez szükséges almenüpontok.

### Create UI5 chart – UI5 grafikon készítése

Ebben a menüpontban vannak a megfelelő mezők melyekbe meg tudjuk adni a grafikon paramétereit.

* Title: A grafikon címe
* Ticker: Itt ki tudjuk választani, hogy az adatbázisban lélvő kriptovaluták közül melyiknek szeretnénk az értékeit figyelni.
* Field: Itt a kritpovaluta különböző értékét választhatjuk ki mely az alábbiak lehetnek:
  + Open: Kriptovalut árfolyama tözsdenyitáskor
  + High: Kriptovaluta árfolyamának aznapi legmagasabb értéke
  + Low: Kriptovaluta árfolyamának aznapi legalacsonyabb értéke
  + Close: Kriptovaluta árfolyamának aznapi zárási eredménye
  + Adjusted Close: A kriptovalutáknál ritka hogy ez az érték más legyen mint a Close érték, ám lehetséges, ha „forkolják” a kriptovalutát, vagyis új verziót kezdenek használni belőle, ami nem kompatibilis az előző tranzakciókkal
  + Volume: A kriptovaluának forgalmának aznapi naagysága.
* Label: itt az adathalmaz címkéjét adhatjuk meg ami meg fog jelenni a grafikonon
* Start date: itt a grafikon kezdődátumát adhatjuk meg
* End date: itt a grafikon végsődátumát adhatjuk meg
* Forecast: itt amennyiben szeretnénk előrejelzéseket nézni a grafikonon, megadhatjuk. Az előrejelzések aznaphoz 60 napra elérhetőek.
  + None: Nincs előrejelzés
  + Forecast with 50% of the data: A rendelkezésre álló adatok második feléből generált előrejelzéseket mutatja.
  + Forecast with 25% of the data: A rendelkezésre álló adatok negyedik negyedéből generált előrejelzéseket mutatja.
  + Forecast with 10% of the data: A rendelkezésre álló adatok tizedik tizedéből generált előrejelzéseket mutatja.

A `Preview Chart` gombra kattintva meg tudjuk tekinteni hogy nézne ki a grafikon ha létrehoznánk. Amennyiben valamelyik beviteli mezőt módosítjuk, az előnézet eltűnik és őjra erre a gombra kell kattintani, hogy megjelenjen.

Az `Add chart` gomb megnyomására a beviteli adatok alapján létrejön egy UI5 grafikon, és a program átnavigál minket a `Custom Charts` menübe.

### Create Chart-js chart – Chart-js grafikon készítése

Ebben a menüpontban vannak a megfelelő mezők melyekbe meg tudjuk adni a grafikon paramétereit.

* Ticker: Itt ki tudjuk választani, hogy az adatbázisban lélvő kriptovaluták közül melyiknek szeretnénk az értékeit figyelni.
* Field: Itt a kritpovaluta különböző értékét választhatjuk ki mely az alábbiak lehetnek:
  + Open: Kriptovalut árfolyama tözsdenyitáskor
  + High: Kriptovaluta árfolyamának aznapi legmagasabb értéke
  + Low: Kriptovaluta árfolyamának aznapi legalacsonyabb értéke
  + Close: Kriptovaluta árfolyamának aznapi zárási eredménye
  + Adjusted Close: A kriptovalutáknál ritka hogy ez az érték más legyen mint a Close érték, ám lehetséges, ha „forkolják” a kriptovalutát, vagyis új verziót kezdenek használni belőle, ami nem kompatibilis az előző tranzakciókkal
  + Volume: A kriptovaluának forgalmának aznapi naagysága.
* Label: itt az adathalmaz címkéjét adhatjuk meg ami meg fog jelenni a grafikonon
* Second Ticker: Itt ki tudjuk választani hogy a második adathalmaz melyik kriptovaluta értékeit mutassa
* Second Field: Itt ki tudjuk választani, hogy a második adathalmaz melyik értékeit nézzük.
* Second Label: Itt meg tudjuk adni a második adathalmaz címkéjét
* Start date: itt a grafikon kezdődátumát adhatjuk meg
* End date: itt a grafikon végsődátumát adhatjuk meg
* Chart Title: A grafikon címe

A `Preview Chart` gombra kattintva meg tudjuk tekinteni hogy nézne ki a grafikon ha létrehoznánk. Amennyiben valamelyik beviteli mezőt módosítjuk, az előnézet eltűnik és őjra erre a gombra kell kattintani, hogy megjelenjen.

Az `Add chart` gomb megnyomására a beviteli adatok alapján létrejön egy Chart-js grafikon, és a program átnavigál minket a `Custom Charts` menübe.

A második adathalmazt nem kötelező megadni.

### Update or Delete chart – grafikon módosítása vagy törlése

E menüpontban tudjuk kezelni a saját magunk által létrehozott grafikonokat.

Először is a lenyíló menüből válasszuk ki a grafikont melyet módosítani vagy törölni szeretnénk, a lenyíló menüben a grafikonoknak a neveit láthatjuk, és rájuk kattintva egyet kiválaszthatunk.

Az `Edit Chart` gombra kattintva, előjön a grafikon típusától a létrehozáshoz használt menü, egy extra mezővel kiegészítve.

Itt ugyanazokat a beállításokat tudjuk módosítani, megadni amiket létrehozáskor is, viszont a bevietli mezők alapértelemzett értéke a grafikon értékeire vannak beállítva, így nem kell minden adatot újra felvinni.

Két új checkbox típusú mező is található, melyekkel a grafikon elrejtettségét, és könvjelzőzöttségét lehet ki/be kapcsolni.

Az `Update Chart` gombra kattintva, a beviteli adatok alapján frissül a grafikon és a program átnavigál a `Custom Charts` menübe.

A `Delete Chart` gombra kattintva az adott grafikon törlődik és a program átnavigál a `Custom Charts` menübe.

### Run script – adatok manipulálása

Ebben a menübenlehet különböző szkripteket futtatni, illetve manipulálni bizonyos mértékben az adatbázisban lévő adatokat.

A `Script` mezőben négy dolgot választhatunk ki:

* Monthy Chart refresh: Ezen szkript futtatása törli az összes generált grafikont és a rendelkezésre álló kriptovaluták adataiból generál egy-egy grafikont mely az ehavi `close` értékeit mutatja.
* Refresh ticker data: Ezen szkript kiválasztásakor megjelenik egy mező ahol az adatbázisban lévő kriptovalutákból lehet választani egyet, és egy időpontot megadni. Ez a szkript törli a kiválasztott kriptovaluta értékeit és a megadott dátum kezdetétől frissíti az értékeket
* Add new ticker data: Ezen szkript kiválasztásakor megjelenik egy mező ahova a kriptovaluta szimbólumát kell beírni, és egy dátumot mely kezdetétől a kriptovaluta értékei bekerülnek az adatbázisba
* Remove ticker data: Egy adatbázisban lévő kriptovaluta eltávolítása
* Analyze tcier data: Ez a funkció elemzi kriptovaluta értékeit a megadott időintervallumon belül, és megadja mellé, hogy a trend melyik irányba fog mozogni, Lináris Regresszió alapján.

A kriptovaluták hozzáadása vagy törlése során automatikusan hozzá tartozó generált grafikon is megjelenik.

A `Run Script` gomb megnyomására a választott szkript a megadott paraméterekkel lefut. A futás során a szöveg ` Selected script is running...` jelenik meg ami indikálja a szkript futását a háttérben.

Amint a szkript lefut, ez a szöveg lecserélődik vagy arra hogy ` Script ran successfully` amennyiben sikeresen futott le, vagy arra hogy ` Script ran with errors` amennyiben hibákkal futott le

Ez a szöveg alatt a szkriptnek futása alatt generálódott üzenetek jelennek meg.

# Fejlesztői dokumentáció

A szoftver fejlesztéséhez szükséges néhány különböző eszköz, mely nem biztos hogy mindenkinek kéznél van.

* npm – csomag kezelő, mely a frondend és backend működéséhez szükséges csomagokat telepíti
* node.js – a backend és az üzleti logika futtatásáért felel
* cds – az adatstruktúrákért felel
* sqlite – a perszisztenciáért felel
* python – az adatok importálásához, és egyéb funkciók biztosításához szükséges szkriptek futtatásához és teszteléséhez

A projekt kódbázisa kettő nagyobb és egy kisebb részből áll, melyek külön mappába vannak szervezve:

* React frontend - `react-ui`
* NodeJS backend - `crypto-gaze`
* Python szkriptek - `python\_scripts`

A szoftver legfontosabb feladata, hogy az adatbázisba betöltött adatokat képes legyen megjeleníteni a böngészőben grafikon formában, és ezekkel a megjenelített grafikonokkal lehetőségünk legyen különböző műveletek végezni, mint például elmenteni őket, testreszabni őket, különböző módon rendezni és megjeleníteni őket.

A szoftver az adatokat a yahoo.com/finance oldalról tölti le, mely egy CSV formátumú táblázatot eredményez, amit megformál és betölti az adatbázisba egy szkript meghívásával. Ez a szkript felel a különböző előrejelzések generálásáért is. A backend több végpontot nyújt a frontend részére is, ahonnan le tudjuk kérni az adatokat, és meg tudjuk jeleníteni. Ezen végpontok teszik lehetővé a felhasználó számára, hogy különböző szkripteket meg tudjon hívni a webes felületen keresztül. A frontend az elmenteni kívánt grafikonok adatait elküldi a backendnek, ami pedig egy rekordként az adatbázisba menti, így innen bármikor le tudjuk kérdezni, és meg tudjuk jeleníteni.

A szoftver felhasználói felületéhez a React keretrendszert használtam, melyben főleg a `ui5-webcomponents-react` és a `material-ui` csomagokat használtam a legtöbb elem megjelenítéséhez. Az oldalsó menühöz a `proprogressbar` csomagot, illetve a komplex, vagyis chart-js grafikonokhoz pedig a `chart-js` csomagot. A backendhez a NodeJS keretrendszert használtam, illetve az SAP Core Data Services eszközeit és definícióit. Az egyéb feladatok ellátásához pedig python szkriptekkel egészítettem ki a kódbázist, melyet a program hív meg.

## Fejlesztői környezet

A fejlesztői környezet a szoftver készítése során nem állt sokmindenből, csak egy windows terminálból, egy böngészőből, és egy kódszerkesztő alkalmazásból.

Kódszerkesztőnek a Visual Studio Code nevű alkalmazást használtam, és böngészőnek a Brave nevű chromium alapú szoftvert.

### Frontend fejlesztése

A frontend fejlesztéséhez elég megnyitni a `react-ui` mappát, a frontend összes eleme ott található. Mielőtt a fejlesztést megkezdenénk, ajánlott a terminálban, ugyanebben a mappában lefuttatni az `npm install` parancsot, ami az internetről leszedi az frontend fordításához és futtatásához szükséges összes csomagot. Amint ez megvolt, kiadhatjuk az `npm start` parancsot mely elindítja a React fejlesztői szerverét a `http://localhost:3000` címen. Itt meg fog jelenni a frontend jelenlegi kinézete, és minden alkalommal amikor frissítjük és elmentjük valamelyik forrásfájlt, az frontend újra lefordul, és frissül is a megadott címen, ahol böngészővel meg tudjuk tekinteni a fordítás eredményét. Amennyiben meg vagyunk elégedve a frontenddel, az `npm run build` parancs lefordítja, és becsomagolja nekünk a kódunkat, melyet az alkalmazás futáskor fel tud használni. Ahhoz hogy teljes mértékben tudjuk tesztelni a frontend kódját, a backendet is el kell hozzá indítani, hiszen nélküle nem tudunk kommunikálni az adatbázissal, és nem tudunk adatokat vagy grafikonokat lekérni megjelenítésre.

### Backend fejlesztése

A backend fejlesztéséhez elég megnyitni a `crypto-gaze` nevű mappát, a backend összes eleme ott található. Hasonlóan ahogy a frontendnél, itt is ajánlott a terminálban, ugyanebben a könyvtárba futtatni az `npm install` parancsot, mely a backend által használt csomagokat letölti az internetről. A backendet fejlesztés során tudjuk perzisztens, és nem perzisztens módon futtatni. Amennyiben nem perzisztens módon szeretnénk, a `cds watch` parancsot kell kiadni a terminálban. Ez a parancs elindítja a backendet egy memóriába töltött sqlite adatbázissal, ami addig őrzi meg az adatokat amíg le nem állítjuk. Ide bekerülnek kezdőadatként a `\db\csv` mappában található CSV fájlok. Hasonlóan, mint a React fejlesztői szervernél, itt is minden forrásfájl módosítására az backend újraindul, de a backend mindig a `http://localhost:4004` címen fog elindulni és futni. Amennyiben perzisztenciával szeretnénk fejleszteni, először is, amennyiben nincs még létrehozva adatbázisfájl, le kell futtatnunk a `cds deploy –to sqlite` parancsot, ami létrehozza az adatbázisfájlt. és beletölti a `\db\csv` mappában található CSV fájlokat. Ezután a `cds serve` vagy a `cds run` parancsot kell kiadni, és a backend el is indul. Itt viszont minden módosítás életbeléptetéséhez szükséges a backendet újraindítani.

### Szkriptek fejlesztése

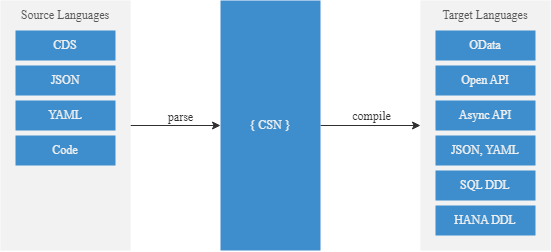
Az alkalmazást támogató szkriptek fejlesztéséhez elég megnyitni a `python\_scripts` mappát, mivel ott található az össze szkript amit az alkalmazás felhasznál. Ezek a rövid kódok mind szoros kapcsolatban állnak a backend végpontjaival, és az adatbázissal is, így futtatásukhoz, és tesztelésükhöz is egyaránt szükséges, hogy a backend fusson.

A szkripteket lehet kézzel futtatni a terminálon keresztül, illetve a frontenden keresztül, a `Run script` menüpontban. Amennyiben a terminálon keresztül futtatjuk, a szkript kimenete a standard outputon megtalálható, ahová, ha hibába ütközne, azt is kiírja. Amikor a szkriptet a frontenden keresztül futtatjuk, ott viszont a kimenetel látszódni fog a böngészőablakban, illetve a terminálon ahol a backendet indítottuk is.

## Backend kialakítása, és működése

### Backend technológai oldala – Core Data Services

A szoftver ezen része, az SAP CAPire, vagyis az SAP felhő alkalmazási programozási modelljének a gerince. Egy infrastruktúra mely segít létrehozni perszisztens adat modelleket, melyet a felhasználói felület számára szogláltatásokkal, és végpontok elérhetővé tételével biztosít. A CDS segítségével lehetőségünk van különböző definiált modelleket adatbázisban táblákként, nézetekként tárolni, illetve meghatározni a többi alkalmazás a modellek mely részeihez hogyan képesek hozzáférni, azokat módosítani.



A CDS modellek egyszerű javascript objektumok, melyek megfelelnek a Core Schema Notation specifikációinak. Ezek a modellek dinamikusa futási idő alatt kerülnek feldolgozásra, és futási idő alatt is létrehozhatóak.

A CDS segítségével meg tudjuk oldani, hogy a React frontend egyszerű HTTP kérésekkel egy REST api-n keresztül kommunikáljon a backenddel, és a backenden keresztül az adatbázissal.

A rekordokat lekérni, és manipulálni a GET POST PATCH és DELETE kérésekkel lehetséges, melyek az alábbi módon felelnek meg a különböző adatbázis operációknak:

|  |  |
| --- | --- |
| HTTP Query | SQL |
| GET | Select |
| POST | Insert into |
| PATCH | Update |
| DELETE | Delete |

A REST hívásoknál, hogy melyik táblára hivatkozunk, azt a definiált végpont dötni el, ahol a modellünk helyezkedik el. Például egy egyszerű szolgáltatásban mely `Catalog Service` néven fut, definiálunk egy `Books` modellt, akkor a SQL lekérdezés mellyel lekérnénk az összes elemét a `Books` táblának az az alábbi módon nézne ki:

SELECT \* FROM Books

Míg a CDS segítségével, elég nekünk egy GET kérést elküldeni a `http://localhost:4004/catalog/Books` címre, melyet ha a böngésző URL sávjába beírnánk, JSON formátumban vissza is kapnánk az összes rekordot a Books táblából. Különböző szűrési és aggregációs műveletekhez tudjuk használni az OASIS által szabványosított OData protokoll URL szabványait, mely a következőképpen épül fel:

A képen szöveg, képernyőkép, sor, Betűtípus látható

Automatikusan generált leírás

Az alkalmazásban használt modelleket több ehhez hasonló szolgáltatásba, modellbe, és végpontba szerveztük.

A definiált szolgáltatásokban nem csak entitásokat találhatunk, hanem leírhatunk cselekményeket (actions), és függvényeket (functions), melyekhez a háttérben személyre szabható kezeléselet adhatunk hozzá, ezzel kibővítve a backend funkcionalitását, és hasonló módon saját kezű eseményeket is hozzáadhatunk.

A szoftverben mi több kötetlen cselekményt is definiáltunk, a különböző adatmanipulációs funkciók biztosításáért.

### Adatmodellek

A backend 3-4 adatmodellel dolgozik összesen, melyek nem kapcsolódnak kifejezetten egymáshoz. Ezek a modellek szimplán az adatok tárolására, és megjelenítésére kellenek.

#### Crypto

A Crypto enitátsban található az összes kriptovaluta napi rekordja. Egy táblában helyezkedik el minden érték valuta összes értéke, így kevesebb endpoint hívással lehet nagyobb terjedelmű adatot lekérni.

Mezői:

* date: Rekord feljegyzésének dátuma, Edm.Date formátumban
* open: Kriptovaluta egy értéke Edm.Double formátumban, null mezővel ellátva
* high: Kriptovaluta egy értéke Edm.Double formátumban, null mezővel ellátva
* low: Kriptovaluta egy értéke Edm.Double formátumban, null mezővel ellátva
* close: Kriptovaluta egy értéke Edm.Double formátumban, null mezővel ellátva
* adj\_close: Kriptovaluta egy értéke Edm.Double formátumban, null mezővel ellátva
* volume: Kriptovaluta egy értéke Edm.Double formátumban, null mezővel ellátva
* ticker: Kriptovaluta szimbluma Edm.String formátumban
* type: Kriptovaluta rekordjának típusa Edm.String formátumban

A rekord bizonyos mennyiségű mezőjének lehetővé kellett tenni, hogy null értéket is felvehessenek, amennyiben feltöltéskor valamilyen probléma lenne, vagy maga a forrásállomány melyből az adatokat töltenénk fel, hibás értékkel rendelkezne, null értéket táplálhassunk be az adatbázisba, így a grafikonok melyek megjelenítik a lekért adathalmazt, igazsághoz hű képet adjanak.

A kriptovaluta rekordjának típusa 4 küönböző érték lehet:

* real
* forecast\_05
* forecast\_075
* forecast\_09

Ezek azt jelzik, hogy a rekord valóságos értéket tartalmaz, vagy valamelyiket a különböző előrejelzésekból. (Az előrejelzések mivoltjáról a Szkriptek részletei közt olvashatunk majd). Amennyiben nem valóságos a rekord, hanem előrejelzés, a `close` érték lesz az egyetlen a kriptovaluta értékei közül mely szám lesz, a többi érték (dátum, típus, tickeren kívül) mind null.

#### PredefinedCharts

Ebben a modellben találhatóak az automatikusan generált havi zárási értékeket megjelenítő grafikonok különböző tulajdonságai.

Mezői:

* key id: a grafikon egyedi azonosítója UUID formában
* ticker: a kriptovaluta szimbóluma melyet a grafikon ábrázol Edm.String formátumban
* start\_date: a grafikon kezdődátuma Edm.Date formátumban
* end\_date: a grafikon végső dátma Edm.Date formátumban
* label: a megjelenített adathalmaz címkéje Edm.String formátumban
* title: a grafikon neve / címe Edm.String formátumban

A ticker, start\_date és end\_date mezők mind azért kellenek, hogy a React frontend, ezekből az adatokból konstruálhasson egy lekérdezést az kriptovalutákról. Ezekben a grafikonokban mindig csak valós értékek jelennek meg, és ezeknek is csak a zárási vagyis a `close` mezőjük.

A title és label mezők szimplán a felhasználói élményhez, és a kinézethez szükséges

#### CustomCharts.

Ebben a modellben találhatóak a felhasználó által létrehozott grafikonok. Ez a modell több tulajdonságot tartalmaz, mint a PreDefinedCharts, mivel több fajta grafikont tárolhat egy ilyen rekord, illetve a grafikonok megjelenítéséről is több információt tartalmaz, hiszen a PreDefinedCharts modell példányai csak a `Home` felületen jelennek meg, míg a `CustomCharts` példányai megjelennek számos oldalon.

Mezői:

* key id: a grafikon egyedi azonosítója UUID formában
* ticker: a kriptovaluta szimbóluma melyet a grafikon ábrázol Edm.String formátumban
* start\_date: a grafikon kezdődátuma Edm.Date formátumban
* end\_date: a grafikon végső dátma Edm.Date formátumban
* label: a megjelenített adathalmaz címkéje Edm.String formátumban
* title: a grafikon neve / címe Edm.String formátumban
* field: a megnevezett kriptovaluta mely értékét jelenítse meg a grafikon Edm.String formátumban
* chart\_type: a grafikon típusa Edm.String fromátumban
* ticker2: amennyiben meg van adva, a grafikon második adathalmazának szimbóluma Edm.String formátumban
* field2: amennyiben meg van adva, a grafikon második adathalmazának megjelenített értéke Edm.String formátumban
* label2: amennyiben meg van adva, a grafikon második adathalmazának címkéje Edm.String formátumban
* forecast: milyen fajta előrejelzést jelenítsen meg a grafikon Edm.String formátumban
* bookmarked: a grafikon könyvjelzőzve van-e vagy sem Edm.String formátumban
* hidden: a grafikon el van-e rejtve Edm.String formátumban

Néhány mező egy az egyben megegyezik a PreDefinedCharts mezőivel, nem csak elnevezésben, hanem funkcionalitásban is. A field, az lehet bármelyik a Crypto modellben található szám típusú értékei közül.

A chart\_type határozza meg, hogy a Frontend milyen fajta grafikonként kezelje a lekért rekordot, amennyiben az érték `simple` a frontend egy UI5 grafikont fog kirajzolni, amennyiben az értéke `complex` egy chart-js grafikont fog megjeleníteni.

A `simple` grafikonoknál nem kerül figyelembe a ticker2, field2, label2 mezők, és létrehozáskor ezek a mezők NULL értékkel szerepelnek az adatbázisban. A `complex` grafikonoknál, viszont ezek az értékek felhasználásra kerülnek, amennyiben helyes értékkel rendelkeznek, és a Frontend ezek alapján egy második lekérdezést indít, és megjeleníti ugyanazon a grafikonon a kettő adathalmazt. A `complex` grafikonoknál a `forecast` mező viszont nem kerül számításba, mivel összetett grafikon létrehozásánál nincs lehetőség előrejelzéseket hozzátenni a megjelenítendő adatokhoz, így létrehozáskor ez a mező mindig a None értéket veszi fel.

A `forecast` mező hasonló módon, mint a Crypto modellben a `type` mező, az előrejelzésekért felel, és felveheti az alábbi értékeket:

* None
* forecast\_05
* forecast\_075
* forecast\_09

A `None` érték mutatja, hogy a grafikonban nem kívánunk nem valós adatokat megjeleníteni, míg a többi érték mutatja, a nem valós adatok közül melyik típusút szeretnénk hogy megjelenjen.

A bookmarked, és a hidden mezők egyaránt a True vagy False értéket vehetik fel szöveg formájában, melyek jelzik a grafikon könyvejlzőzöttségét, illetve rejtettségét a frontend felé.

#### CommandResult

Ebben az entitásban találhatóak a lefutott szkriptek eredményei, az itt lévő rekordokat a felhasználó csak akkor látja amikor egy bizonyos szkritpet lefuttat, és kivárja a végeredményt. Általában ez a tábla üres, mivel a frontend kiveszi a megjelenített értékeket , így itt csak a kulcs nélkül indított műveletek, és némelyik nem megjelenített műveletnek a lefutási eredményei vannak.

Ez a tábla automatikusan törlődik minden indítás során.

Mezői:

* command: A lefuttatott szkript neve Edm.String formátumban
* data: A lefuttatott szkript kimeneti adatai Edm.String formátumban
* opKey: A szkript futtatásának egyedi azonosítója.

A command mezőbe a lefutott szkript neve kerül bele, hogy a konzolt figyelve egyértelműen meg lehessen határozni, hogy milyen típusú szkript indult.

A data mezőben találhatóak különböző hibaüzenetek, vagy szimplán lefutási eredmények, mint pléldául az analyst szkript futtatásakor, ugyani akkor nem csak a futás során előjött hibaüzenetek vannak benne, hanem értékes adatok megjelenítéshez, és adatok melyeket a frontend felhasznál ideiglenes grafikon létreohzásához is.

Az opKey mezőben egy UUID-nek 4-es verziójú karakterlistája található, melyet a frontend használ, hogy nyilván tudja tartani melyik szkriptnek a kimenetét szükséges megjelenítenie a felhasználó részére.

### Szolgáltatások

#### CryptoService

Ebben a szolgáltatásban található a Crypto entitás, és három `action` mely a kriptovaluták adatainak módosításához fűződik.

* DeleteTicker: Egy kriptovaluta adatait törli az adatbázisból, paraméterei:
  + ticker: a kriptovaluta szimbóluma Edm.String formátumban
  + opKey: a szkript futtatási kulcsa Edm.String formátumban
* AddTicker: Egy kriptovaluta hozzáadása az adatbázishoz, paraméterei:
  + ticker: a kriptovaluta szimbóluma Edm.String formátumban
  + date: a dátum melytől az adatokat letöltjük Edm.String formátumban
  + opKey: a szkript futtatási kulcsa Edm.String formátumban
* RefreshTicker: egy kriptovaluta adatainak frissítése, paraméterei:
  + ticker: a kriptovaluta szimbóluma Edm.String formátumban
  + date: a dátum melytől az adatokat letöltjük Edm.String formátumban
  + opKey: a szkript futtatási kulcsa Edm.String formátumban

Ennek a szolgáltatásnak a három `action` végpontján definiáltunk három különböző függvényt, mely kezeli az ide érkezett hívásokat, és tartalmazza az üzleti logikát.

##### DeleteTicker

A kérésben szerepel, a kriptovaluta szimbóluma, és egy operációs kulcs, amennyiben a kriptovaluta szimbóluma mégsem szerepel, a függvény terminál.

Helyes működéssel, a föggvény törli a Crypto táblának összes rekordját, ahol a szimbólum megyegyezik a kérésben szereplővel, illetve kitörli az ehhez a kriptovalutához generált grafikont is. Ezeket helyben Javascript lekérdezések segítségével teszi meg. Ezután a sikeres futás eredményét betölti a CommandResult táblába.

##### AddTicker

A kérésben szerepel a kriptovaluta szimbóluma, egy kezdődátum, és egy operációs kulcs.

Itt is, hasonlóan, a függvény rögtön terminál, ha nincs érték megadva a `ticker` részének a kérésnek.

A függény elindítja az `add\_data.py` python függvényt, melynek paraméterei a kérésben szereplő argumentumok. Ezután a függvény folyamatosan figyeli, hogy az elindított szkript milyen adatokat ír ki a stdout-ra, és ezeket kiiratja a konzolra, melyben elindítottuk az alkalmazást. Amikor az indított szkript terminál, a konzolra ez is kikerül, illetve amennyiben a bezárási kód 0, vagyis futást megakadályozó hiba nélkül terminált a szkript, a függvény létrehoz egy előre generált grafikont, a kriptovaluta e-havi értékeivel, és ezt belehelyezi az adatbázisba, miután törölte az ugyanehhez a kriptovalutához tartozó grafikont, amennyiben létezett volna.

##### RefreshTicker

A RefreshTicker , a DeleteTicker, és az AddTicker funkciójának egybevonása.

Először törli a kriptovaluta adatait, amennyiben a kérés tartalmazza a szimbólumot. Törlés végbevitele előtt, lekéri az első dátumot mellyel a kriptovaluta szerepel, amennyiben a kérés nem tartalmaz megfelelő dátumot. Ezután a függvény, hasonlóan mint az AddTicker, elindítja a megadott szkriptet, a megfelelő paraméterekkel, és sikeres futás esetén, törli, és létrehozza a megfelelő grafikont.

#### ChartService

Ez a szolgáltatás tartalmazza a kettő grafikonokhoz kapcsolódó entitást, a PreDefinedCharts-ot illetve a CustomCharts-ot is. Mellettük `action` -ként megtalálható a `RefreshCharts` is, paramétere opKey – szkript operációs kulcsa, Edm.String formátumban.

Ez a cselekmény felel a generált grafikonok frissítésének manuális futtatásáért.

##### RefreshCharts

A kérésben adatként csak egy operációs kulcs szerepel.

A függvény nem csinál mást, csak elindítja a `monthly\_charts.py` nevű szkriptet az operációs kulccsal mint argumentum, és kiíratja a konzolra, amit a python szkript írna ki, illetve jelzi a szkript terminálásának eredményét.

#### EndpointService

Ez a szolgáltatás tartalmazza a CommandResult entitást, illetve kettő Action-t mely összefüggésben van az entitás mivoltjával.

* DeleteResult: A megfelelő szkript eredényének törlése, paramétere opKey, a szkript operációs kulcsa, Edm.String formátumban.
* Analyst: Az `analyst.py` szkript futtatásának mehívásához használja a program, paraméterei:
  + ticker: a kriptovaluta szimbóluma Edm.String formátumban
  + start\_date: elemzés kezdeti dátuma Edm.String formátumban
  + end\_date: elemzés végső dátuma Edm.String formátumban
  + opKey: a szkript futásának operációs kulcsa

##### DeleteResult

Ez a függvény nem csinál mást, csak helyben kitörli a CommandResult táblában lévő rekordokat, melynek megyegyezik az operációs kulcsa a kérésben szereplőjével.

##### Analyst

Hasonlóan, mint a legtöbb szkript indításához használt függvénynél, itt is amennyiben nincs megadva a kérésben a kriptovaluta szimbóluma, a függvény terminál. Amennyiben tartalmazza, a függvény meghívja az `analyst.py` szkriptet, a kérésben szereplő szimbólummal, kezdődátummal, végdátummal, és operációs kulccsal.

Itt is a szkript futásának eredményét, illetve a futása során kiírt üzeneteket jeleníti meg a konzolra.

#### Szerver

A szolgálgatások, entitások, végpontok, és ezeket kezelő függvények mellett található még egy szkript mely a szerver különböző eseményeit figyeli, és ezeknek megfelelően cslekeszik.

##### Listening

Ez az esemény akkor triggerelődik, amikor a szerver elkezdett a megadott porton figyelni a bejövő kapcsolatokra, és képes azokat fogadni, és kiszolgálni, vagyis amikor a program használatra készen fut. Ilyenkor a függvény először is, kiüríti a CommandResult tábla rekordjait, majd kigyűjti egy változóba az elérhető kriptovaluták szimbólumait. Ezután párhuzamosan, minden szimbólummal amit lekérdezett, lefuttatja az `add\_data.py` szkriptet, azzal a dátummal amellyel az első adott szimbólumú rekord található az adatbázisban, illetve az `add\_chart.py` nevű szkriptet, mely egy olyan szkritp amelyhez a felhaszáló nem fér hozzá a webes felületen keresztül. Ez a szkript felelős a grafikon létrehozásáért, és nem egy Javascript query, mivel a for ciklus aszinkron módon megy, és több műveletet is tartalmaz, melynek futására várnia kell a programnak, így egy külön szkriptbe szervezve, nem kerülnek bizonyos változók átírásra kiolvasásuk előtt/alatt.

##### Bootstrap

Ez az esemény a backend indításának elején történik. Itt a függvény, csak instruálja az alkalmazást, hogy a beépített autmoatikusan generált felület helyett, a frontend felületet használja, és szolgáltassa a felhasználónak, mikor a böngészőbe beírja az alkalmazás elérési pontját.

Itt található még egy instrukció, mely megmondja az alkalmazásnak hogy használja a `CORS`nevű csomagot, melynek fő funkciója, hogy különböző címekről is lehessen adatmanipuéációs máveletek végezni. Ez kifejezetten hasznos, ha egy felhőben lévő adatbázishoz csatlakoztatnánk az alkalmazást, vagy elindítanánk a React fejlesztői szervert, mely a localhost-on fut ugyebár, de a 3000-es porton, mely megszefgné a CORS irányelveket.

A konfigutrciós fájlban egy harmadik függvény is található, mely egy CronJob-ot ír le, melyen belül ugyanazon parancsok futnak le, mint a `listening` eseménynek kezelője során. Ez a CronJob minden alkalommal amikor éjfélt üt az óra lefut, és frissíti a kriptovaluták adatait, illetve a hozzájuk tartozó grafikonokat, amennyiben éjfélkor, az alkalmazás fut.

## Szoftvert támogató szkriptek működése

A szoftvernek egyes funkcionaluitása kiszervezésre került python szkriptekbe, mely a könnyebb adatmanipuláció, és különböző előrejelzési, elemzési funkciók ellátása miattt történt.

Ezeknek a szkripteknek nem javasoljuk az elindítását manuálisan, csak a webes felületen keresztül, a legjobb működés elősegítéséhez.

A python szkriptek futtatásához pythonra szüksége van a rendszernek, és azon belül több különböző telepített csomagra is.

### add\_charts.py

Ez a szkript, csak a szoftver számára létezik, a felhasználó nem tudja külön elindítani ezt a szkriptet a webes felületen keresztül.

A szkriptre a szoftvernek azért van szüksége, hogy párhuzamos módon képes legyen több grafikont létrehozni indításkor, vagy éjfélkor.

A szkript argumentumként csak egy kriptovaluta szimbólumot kap, és a CustomChart végpontba küld egy grafikont, melyben ez a kriptovaluta szerepel, és a szkript futtatásakori hónap.

### add\_data.py

Ez a szkript a legfontosabb az egész szoftveren belül, hisz ez tartalmazza a megfelelő funkcionalitást, hogy új adatokat lehessen a szoftverbe betölteni.

A szkript először csekkolja, hogy argumentumként érkezett-e operációs kulcs, amennyiben nem, vagy az operációs kulcs `null` vagy `undefined` a szkript egy hibaüzenetként kiírja az stdout-ra , illetve egy változóba gyűjti, melyet a CommandResult táblájának egyik mezőjébe be fog tölteni.

A szkript ellenőrzi, hogy kriptovaluta szimbólum értkezett-e mint első argumentum, amennyiben nem, vagy az értéke `null` vagy `undefined` a szkript terminál egyes kóddal, és a hibaüzenetet kiírja a stdoutra, illetve elküldi a CommandResult táblájába mint egy rekordot, melyben a parancs az `add\_data`, az adat a hibaüzenet, és az operációs kulcs az amit megadtunk.

Ehhez hasonló módon ellenőrzi hogy a dátum megadásra került-e és átkonvertálható -e UNIX EPOCH időbélyeggé. Amennyiben nem, a szkript kiírja a strdoutra, és a változóba mely a végeredményt közvetíti, a végére hozzáfűzi. Ilyenkor a szkript kiszámol egy öt évvel ezelőtti dátumnak a UNIX időbélyegjét, és azzal folytatja a futást.

Ezután a szkript ellenőrzi, hogy az adatbázisban szerepel-e az adott kriptovaluta, amennyiben igen, a szkript terminál egyes hibakóddal, és a hibát kiírja az stdoutra, és elküldi a CommandResult táblába is.

A szkript ezután megkonstruálja a letöltési URL-t mellyel a yahoo.com-ról lekéri egy CSV fomátumban a kriptovaluta adatait. Amennyiben hiba történik a letöltés alatt, a szkript itt is hasonló módon, terminál egyes kóddal, és a hibát kiírja az stdoutra, és elküldi a CommandResult táblába is.

Letöltés után beolvassa a letöltött CSV állományt, törli a fájlt, majd átnevezi az oszlopokat, és hozzáadja a `ticker`és a `type` nevű oszlopokat. A `ticker` oszlop minden értéke a megadott szimbólum lesz, és a `type`oszlop értékei mind a `real`érték lesz.

Ezek után az adathalmaznak a dátum és `close` értékeiből készít egy sorozatot, melyenek veszi a második felét, a negyedik negyedét, és a tizedik tizedét, mind külön-külön adathalmazokba. Ezeknek a halmazoknak a hiányzó értékeit interpolálással lecseréljük, majd példányát létrehozzuk az XGBModelnek.

Mielőtt a predikciókat lefuttatjuk, létrehozunk egy listát, melyben a prediktált értékekhez tartozó dátumokat legeneráljuk, méghozzá mához a következő 60 nap dátumát.

Eztán mindhárom adatsorozattal melyet létrehoztunk, a következő dolgokat tesszük:

* az eredeti adatszerkezethez hasonlóan, legeneráljuk a `ticker` mezőt, és a `type` mezőt, mely adaathalmaztól függően lehet forecast\_05, forecast\_075, forecast\_09
* az egyik modellt betaníttatjuk a z adathlmazzal, majd 60 értéket előre generálunk, melyeket áttöltünk egy másik formába.
* az így létrejött adatszruktúrának átnevezzük és átrendezzük az oszlopait, hogy megegyezzen a Crypto modellben lévő sémával.

Amennyiben a tanítás, vagy a predkció hibába ütközik, kiírjuk az sdtoutra, és hozáfűzzük a változóhoz mely a szkript lefutási állapotát őrzi.

Tapasztalatink szerint, a predikció, és a tanítás, csak abban az esetben ütközik hibába, amennyiben túl kis mértékű adathalmazt adtunk meg a szkriptnek, így nagyobb adathalmaz elemzése esetén mindegyik predikció sikeresen fog lefutni.

Miután a predikciókat lefuttattuk, a szkript meghatározza a kerekítési értéket. Alapesetben három tizedesjegyre kerekít, amennyiben a kriptovaluta `close` értékei közül a legnagyobb 10-nél kisebb, 5 tizedesjegyre kerekít, viszont ha 1000-nél nagyobb akkor csak 2 tizedesjegyre kerekít.

Miután az adatmanipulációs lépések megvoltak, a szkript az eredeti, és a három előrejelzett adathalmazt, ugyanazon módon próbálja feltölteni az adatbázisba.

Először megpróbálja a rekordot feltölteni, amennyiben hibába ütközik feltöltés közben, megpróbálja a rekordot null adatokkal feltülteni, ha így is hibába ütközik a szkript terminál egyes hibakóddal, és a hibát kiírja az stdoutra, és elküldi a CommandResult táblába is.

Amennyiben a szkript hibába ütközött volna olyan helyen, melyen nem lett volna lekezelve, a szkript ugyanúgy és a hibát kiírja az stdoutra, és elküldi a CommandResult táblába is.

### analysis.py

A szkript egy kriptovaluta időintervallum alatti elemzésért felelős, és ezt jeleníti meg a `Run Script` felületen, miután a meghívásra került a webes felületen.

Megadja a legkisebb, legnagyobb értéket és az átlagot az időintervallumon belül, illetve az utolsó 10 napot nézve, ugyanezeket az értékeket. Eztán LinearRegression modell segítségével előrejelez egy kialakuló trendet.

Ezeket az értékeket mind a CommandResult táblába tölti be, és ezek alapján jeleníti meg a frontend.

A szkript hasonlóan az add\_data.py-hoz először ellenőrzi az operációs kulcs létezését, és értékét, majd a megadott kriptovaluta szimbólum létezését, és értékét.

Ezután a kezdő és végdátumot ellenőrzi, amennyiben nincs kezdődátum 1970-01-01-et veszi kezdődátumnak, amennyiben nincs végdátum, None-nak veszi az értéket és kihagyja a lekérdezésből.

A változóhoz hozzáköti pontoszvessző után a szimbólumot, a kezdődátumot és a végdátumot.

Ezután lekéri a kiválasztott kriptovaluta értékeit, majd beletölti egy adatszerkezetbe. Kiszámítja a kerekítési számot, ugyanazzal a módszerrel, mint az `add\_data.py` szkriptben, majd az alábbi értékeket hozzáteszi a változóhoz:

* Range: a legalacsonyabb érték és a legmagasabb érték egy kötőjellel elválasztva
* Average: az átlaga a `close` értékeknek

Amennyiben 10 napnál nagyobb a megadott időintervallum a következő kettő érték is belekerül:

* 10 day range: ugynaz mint a Range csak az utolsó 10 napra nézve
* 10 day average: ugynaz mint a Average csak az utolsó 10 napra nézve

Ezután egy lineáris regressziót használó model segítségével előrejelzünk 60 napi értéket, majd az első és utolsó előrejelzett nap értékéből megállapítjuk, hogy a trend szerint a kriptovaluta értéke felfelé, vagy lefelé fog mozogni, és ezt is belerakjuk a változóba. Amennyiben itt problémába ütköznénk, azt jelenítjük meg a változóban.

Ezután egy pontosbesszővel elválasztva a változó végére a különböző generált adatokat dátummal együtt belerakjuk, hogy a frontend egy grafikon keretein belül kirajzolhassa a generált értékeket.

Itt is, hasonlóan az `add\_data.py`-hoz, ha bárhol hibába ütközne a szkript, ami nem lenne lekezelve, a szkript terminál egyes hibakóddal, és a hibát kiírja az stdoutra, és elküldi a CommandResult táblába is.

### monthly\_charts.py

Ennek a szkriptnek a feladata, hogy törölje, és újragenerálja a létrehozott grafikonokat, a felhasználó kérésére. Először kitörli a táblában lévő grafikonokat, majd az adatbázisban lévő kriptovaluták szimbólumaiból, havi grafikonokat generál, és feltölti a PreDefinedCharts táblába.

A szkriptnek csak egy argumentuma van, az opKey, így először az `add\_data.py`-hez hasonlóan, ellenőrzi létezik-e az argumentum, és amennyiben nem, vagy undefined, vagy null, ezt kiírja az stdoutra, illetve a változóba, és folytatja a futást.

Ezután a szkript lekéri a kriptovalutákat az adatbázisból és eltárolja.

Ezután a szkript lekéri a generált grafikonokat, majd egyesével kitörl őket az adatbázisból.

Ezután, amennyiben talált kriptovaluta adatokat, mindegyikhez konstruál egy grafikont, mely az ehavi értékeit fogja mutatni a kriptovalutának, majd ezeket feltölti az adatbázisba.

Ezután elküldi a CommandResult táblába, hogy sikeresen frissítette a grafikonokat.

Hasonlóan, mint a többi szkriptnél, amennyiben itt is egy lekezeletlen hiba történne, a szkript terminál egyes hibakóddal, és a hibát kiírja az stdoutra, és elküldi a CommandResult táblába is.

## Frontend kialakítása és működése

A frontend teljes egésze a `reac-ui` mappában található, a hozzá felhaszált technológia pedig a React.

A React egy javascript alapú keretrendszer, melynek segítségévek egy oldalas webalkalmazásokat tudunk létrehozni egymásba ágyazható komponensek segaítségével. Egyoldalas, mivel a teljes frontend egy elérési ponton, url-en található, egy darab html fájl, és bármilyen változás történik az oldalon, vagy az url sávban, az mind kliensoldali logika alapján történik.

A komponensek `jsx` fájlokba külön mappákba vannak szervezve, és ezekből az `npm run build` parancs létrehoz egy index.html-t melyben az összes komponens amit felhasználtunk, javascript fájlokba kerül fordításra, mely a generált index.html fájlba bele van linkelve.

A komponensek, melyeket a frontendhez készítettem, három mappába vannak rendezve. A Basic Components mappában vannak a komponensek, melyeket egy oldalon többször használtam fel, vagy olyan komponensek, melyeketet meg lehet jeleníteni több helyen is.A Layout mappában találhatóak a frontend elrendezéséért felelős komponensek, melyek biztosítják a webes felület kialakítását. A Routes amppában vannak a különböző menüpontokat leíró fájlok. Az frontend belépési pontja a MyApp.jsx nevű fájl. Ebben a fájlban vannak egymásba ágyazva a főbb kopmponensek, mint az Oldalsó navigációs sáv, a React navigációért felelős komponens, a Layout komponens, melyre később kitérünk, és a különböző útvonalak, melyek segítségével különböző menüpontokat jelenítünk meg, látszólag más és más URL-eken, min például a Grafiklonok listázási komponense a `/charts` úton érhető el.

Az alkalmazás felületén az oldalsó sáv, sosem változik, csak lekicsinyíteni lehet. A menüpontokra kattintás során a mellette lévő tartalom változik meg a menüpontokban meghatározott komponensekre.

### Routes

Ebben a mappában találhatóak meg, a különböző menüpontok elemei.

#### Home

Ez a komponens maga a főoldal, amit a felhasználó lát amint az alkalmazás címét beírja az URL sávba. Nem tartalmaz semmi különleges dolgot, csak egy API kérést a PreDefinedCharts táblára, melyből a generált grafikonokat lekéri, és sorban megjeleníti őket a MyChart komponens segítségével.

Fontos megjegyezni, hogy a program kialakítása miatt, mivel a React kliens oldali útvonalakat használ, bármikor a felhasználó az újratöltés gombot nyomja meg a böngészőben, vagy néha az előre vagy a vissza gombot, az alkalmazás erre az oldalra irányítja át a felhasználót.

#### Charts

Ez a komponens felel, a felhasználó által készíttett grafikonok megjelenítéséért. Menüpontban megjelenő neve Custom Charts

Egy fejléc szerű `Card` komponensben található kettő beviteli mező, mely különböző kiválasztási lehetőségeket ad a grafikonok sorrendbe helyezéséhez. Az első a mezőt, a második az irányt határozza meg. Mellettük van egy checkbox mely megmondja a komponensnek, hogy speciális rendezést alkalmazzon-e vagy sem, mellyel a könvyjelzőzött grafikonok, külön sorba rendezésre kerülnek, és a nem könyvjelzőzött grafikonok előtt kerülnek megjelenítésre. Meleltte pedig egy gomb, mellyel a kívánt rendezést lehet megjeleníteni. Az alapértelmezett rendezés, melyet a felhasználó lát amint rákattint a menüpontra, az az elsődelegs kritpotvaluta szimbólumok alapján történik, növekvő sorrendben, mely ábécé sorrendnek megfelelő rendezést jelent a dátum mezők kívételével, és a könyvjelzőzött grafikonok előre kerülnek. Amint a felhasználó változtatna valamelyik beviteli mező értékén, a rendezett grafikonok eltűnnek, és a `Show Charts` gomb megnyomására jelennek meg újra. A rendezést, ugyanaz a függvény teszi meg, mely a grafikonok lekérdezéséért felelős. Az elrejtett grafikonok itt nem jelennek meg.

A fejléc szerű konstrukció alatt pedig egy egyszerű `ChartList` komponens található, mely miután megadtuk neki a grafikonok rekordjait srrendben, azért felelős, hogy ezeket megfelelő formában, adatokkal együtt megjelenítse nekünk.

Legtöbb grafikonnal való művelet után, a szoftver, erre az oldalra irányít át minket.

#### BookmarkedCharts

Ez a komponens felel a könyvejlzőzött grafikonok megjelenítésért.

A komponens kifejezetten hasonlít a Home komponenshez működés szempontjából, hiszen semmi más feladata sincs, csak lekéri az adatbázisból azokat a grafikonokat melyek nem rejtettek, és könyvjelzőzött értékkel rendelkeznek, majd ezeket átadja a egy ChartList komponensnek, mely megjeleníti a grafikonokat.

Ha egy grafikont könyvjelzővezünk, az alkalmazás erre az oldalra fog minket átirányítani, amennyiben viszont a könyvjelzőt levesszük egy grafikonról, az alkalmazást a Custom Charts menüpontra fog minket átirányítani.

#### HiddenCharts

Ez a komponens felel az elrejtett grafikonok megjelenítéséért, az elrejtett grafikonok itt jelennek meg, és sehol máshol az alkalmazáson belül.

Működésében, kialakításában szinte teljesen megegyezik a BookmarkedCharts komponensel, annyi kivétellel hogy a lekérdezés, melyet az adatbázisnak küldünk, itt tratalmazza hogy a rejtett mezője a grafikonnak True legyen, mely szűkítés szinte sehol máshol nem található meg a programban.

A lekért grafikonokat, itt is egy ChartList komponens jeleníti meg sorban.

Amint egy grafikont elrejtünk, az alkalmazás erre a menüpontra irányít minket, illetve ha egy grafikonnak megszüntetjük az elrejtettségét, a Custom Charts menübe irányít innnen minket az alkalmazás.

#### AddSimpleCharts

Ez a kopmonens felel az egyszerűbb UI5 grafikonok létrehozásáért.

Nem tartalmaz semmit, csak egy MyChartSettings nevű komponenst, melyet később fogunk tárgyalni.

#### AddComplexChart

Ez a kopmonens felel az komplexebb Chart-js grafikonok létrehozásáért.

Nem tartalmaz semmit, csak egy MyChartSettings2 nevű komponenst, melyet később fogunk tárgyalni.

#### ChartModified

Ez a komponens felel a grafikonokon végzendő műveletek megengedéséért.

Hasonlóan mint a Charts komponensben, itt is találhatunk egy fejléc szzerű elemet, melyen egy kiválasztó menü, és három gomb található meg.

A lenyíló menüben választhatjuk ki, hogy melyik grafikont szeretnénk törölni vagy módosítani. Az `Edit Chart` gomb megnyomására megjelenik egy ChartEditor kopmponens, melyben a grafikon adatai vannak. Az Update Chart gomb megnyomására a grafikon rekordjára a frontend elküld egy PATCH kérést melyben benne vannak a beírt adatai a grafikonnak. Amennyiben nem változtatjuk az adatokat, a kérés az eredeti adatokkal fut le. A Delete Chart gomb megnyomása esetén, pedig egy DELETE kérés indul az adatbázis felé, mellyel a grafikon törlődik. Amennyiben nem választottunk ki grafikont, a gombok nem csinálnak semmit. Amennyiben igen, a művelet befejezése után, a szoftver átnavigál minket a Custom Charts menüpontra.

#### Scripts

Ez a komponens felel a python szkriptek böngészőből való futtatásáért, illetve a szkriptek eredményének kijelzéséért is egyaránt. A menüben ezt a kopmonens-t a Run script néven érjük el.

Ahogy az előző komponensben, itt is találunk egy fejléchez hasonló elemet, melyen különböző beviteli mezők találhatóak, egy lenyíló menü, és egy `Run Script` nevű gomb.

Ez a kettő mező, végig jelen van, bármilyen szkriptet választanánk futtatsára. Mivel néhány szkriptnek egy vagy kettő felhasználó által bevitt paraméter szükséges, így a különbözőő szkriptek kiválasztásakor, különböző beviteli mezők jelennek meg:

* Monthly Chart refresh: semmilyen mező nem jelenik meg
* Refresh ticker data: egy lenyíló menü megjelenik, melyből kiválaszthatjuk melyik kriptovalutának szeretnénk az adatait frissíteni, illetve egy dátumválasztó menü is, mellyel az adathalmaz kezdődátumát tudjuk megadni
* Add new ticker data: itt egy szövegbeviteli mező jelenik meg, melybe a hozzádani kívánt kriptovaluta szimbólumát kell beírni, és az előhőöz hasonlóan egy dátumválasztó menü is, mellyel az adathalmaz kezdődátumát tudjuk megadni
* Remove ticker data: itt egy lenyíló menü jelenik meg, melyben az adatbázisban található kriptovaluták szimbólumai találhatóak, és ezek közül kell kiválasztani az egyet melyet szeretnénk eltávolítani
* Analyze ticker data: itt egy lenyíló menü jelenik meg melyben az adatbázisban tlaálható kriptovaluták szimbólumai van, illetve kettő dátumválasztó menü, melyben meg lehet adni, hogy a kriptovaluta elemzése ,milyen időintervallumon történjen

Miután kiválasztottuk a lefuttatni kívánt szkriptet, és megadtuk a megfelelő adatokat, a Run script gomb megnyomására, a frontend elküldi a megadott adatokat a backend egy végpontjára, mely a megfelelő műveletek futtatásáért felelős.

Míg a szkript fut, egy `Selected script is running…` szöveg megjelenik a fejléc alatt. A szkript futása során a frontend, folyamatosan (1/2 másodpercenként) ellenőrzi a CommandResult táblát, hogy érkezett-e oda adat a szkript indítása során generált operációs kulccssal. Folyamatosan API kéréseket küld egészen addig, ameddig nem jön be rekord, vagy a felhasználó el nem kapcsol a menüpontból. Amennyiben a felhasználó elkapcsol, a szkript futtatási eredményének lekérése, és így megjelenítése is abortál. Miközben a kérések mennek, a Run Script gomb is kiszürkül, így nem lehet rögtön új szkriptet indítani, hasak a felhasználó el nem kapcsol egy másik menüpontra majd vissza.

Amennyibena felhasználó megvárja hogy a szkript termináljon, a lekért értékeket átadja egy CommandResponse nevű komponensnek, elérhetővé teszi a `Run Script` gombot, majd a lekért adatot törli az adatbázisból.

## Layout

Ebben a mappában az elrendezésért felelős komponensek találhatóak.

#### Layout

A layout komponens arra van, hogy szimplán elkülönítse a teret, kódban és böngészőben is egyaránt, ahol a változó tartalom fog megjelenni.

#### SidebarComponent

Ez a komponens felel a navigációért az alkalmazásban, benne találhatóak a különböző menüpontok elrendezése, illetve a pszeudo elérési útvonalak, melyeket a frontend kattintásra lekezel, és megjeleníti a kiválasztott tartalmat.

### Basic Componenst

Ebben a mappában találhatóak a legkisebb egységei a frontendnek, azok a komponensek, melyeket többöször, több formában, vagy több helyen is megjelenít a webes felület.

#### ChartEditor

Ez a komponens a grafikon módosításakor kerül elő, és egyetlen feladata, hogy a paraméterek alapján vagy MyChartSettingsUpdate, vagy MyChart2SettingsUpdate komponens-t jelenítsen meg, illetve a belső változók állapotát közvetítse a szülő és a gyerek komponensei közt.

#### ChartList

Ennek a komponensnek az egyetlen feladata, hogy a bejövő grafikon rekordokon végigiterál, és a grafikon típusa alapján kirajzoljon egy MyChart vagy egy MyChart2 komponenst.

##### CommandResponse

Ez a komponens felel a futtatott szkriptek eredményének megjelenítéséért.

A komponens egyfejléc szerű objektumban kiírja hogy a szkript sikeresen futott, vagy hibákba ütközött futása során, illetve, az eredmény tartalma alapján több dolgot is kijelezhet.

Amikor az eredmény típusa `simple` akkor csa egy egyszerű szövegmezőt jelez ki, hiszen a szkript futása során csak egyszerű szöveges üzenetek keletkeztek, melyeket szimplán meg tud jeleníteni.

Amennyiben az eredmény típusa `complex`, vagyis az eredményszövegben található egy pontosvessző, mely azt indikálja, hogy az `analyst.py` szkript futott le, mely az eredményben nem csak szöveget helyez, hanem adatokat, mellyel garfikont lehet kirajzolni. A válaszban ilyenkor benne lévő adatok:

* Futás közbeni hibák
* Kalkulált értékek
* Garfikont kijejelzéséhez szükséges adatok
* Extra generált adatok a vizualizációhoz

A grafikonhoz tartozó adatokat, és a vizualizációhoz tartozó adatokat, a komponens kiveszi a megjelenítendő adatok közül, és egy MyChartCommandResponse kopmponensnek adja át, mely megjeníti a grafikont, az extra adatokkal együtt.