



Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Idősor vizualizációs webalkalmazás

Dr. Pataki Norbert

Adjunktus, PhD

Ovád Nóra

Programtervező Informatikus Bsc

Budapest, 2020

EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Hallgató adatai:

Név: Ovád Nóra

Neptun kód: ISBN55

Képzési adatok:

Szak: programtervező informatikus, alapképzés (BA/BSc)

Tagozat: Nappali

Belső témavezetővel rendelkezem

Témavezető neve: Pataki Norbert

munkahelyének neve: ELTE IK, Prog. Nyelvek és Fordprog. Tanszék

munkahelyének címe: 1117. Bp. Pázmány Péter sétány 1/C

beosztás és iskolai végzettsége: Adjunktus, PhD

A szakdolgozat címe: Idősor vizualizációs webalkalmazás

A szakdolgozat témája:

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

A szakdolgozat célja egy webes környezetben futó alkalmazás, melynek fő funkciója, hogy különféle adatforrásokat (pl. egérmozgatást, telefon giroszkóp adatokat) különböző grafikonok segítségével vizuálisan ábrázoljon (pl. sebesség grafikon). Ezek az adatforrások időszorként reprezentálhatóak, valamint különféle vizuális ábrázolásuk megadható. Az adatforrások file-ban kerülnek elmentésre a szerveren későbbi újraindítás céljából. A webes alkalmazás különálló backendből és frontendből áll, a kettő között hálózati kommunikáció zajlik. A webes felületet a felhasználó saját ízlésének megfelelően alakíthatja. A felületen megtalálható többek között az adatforrás, a konfiguráció, a metaadatok és a különféle grafikonok. Az alkalmazás Törteli Olivér Máté közreműködésével készül.

Budapest, 2019.11.09.

Tartalom

| | |
|---|-------|
| 1. Bevezetés..... | III |
| 1.1 Témaválasztás..... | III |
| 1.2 Célok..... | III |
| 1.3 Ismert problémák..... | III |
| 2. Felhasználói dokumentáció..... | V |
| 2.1 Rendszerkövetelmények, telepítés..... | V |
| 2.1.1 Szerver..... | V |
| 2.1.2 Kliens..... | V |
| 2.2 Futtatás..... | VI |
| 2.2.1 Szerver..... | VI |
| 2.2.2 Kliens..... | VII |
| 2.3 Használat..... | VII |
| 2.3.1 Főoldal (index.html)..... | VII |
| 2.3.2 Adatmegjelenítő oldal (TimeseriesClient.html)..... | VIII |
| 2.3.2.1 Konfiguráció..... | IX |
| 2.3.2.2 Idősorok metaadatai..... | XI |
| 2.3.2.3 Vonaldiagramok tulajdonságai..... | XI |
| 2.3.2.4 Vonaldiagramok kezelése..... | XII |
| 2.3.2.5 Animációk..... | XIII |
| 3. Fejlesztői dokumentáció..... | XV |
| 3.1 Használt fogalmak..... | XV |
| 3.2 Funkcionalitás – elm..... | XV |
| 3.2.1 Az elm nyelv..... | XV |
| 3.2.2 Kommunikáció javascript-kód és elm-kód között (portok)..... | XVII |
| 3.2.3 Kommunikáció a szerverrel..... | XVIII |
| 3.2.4 Főoldal (Index.elm)..... | XIX |
| 3.2.5 Adatmegjelenítő oldal (TimeseriesClient.elm)..... | XIX |
| 3.2.5.1 Típusok..... | XIX |
| 3.2.5.2 Inicializálás..... | XXII |
| 3.2.5.3 Update..... | XXII |

| | |
|---|--------|
| 3.2.5.3.1 Szerverről érkezett adat kezelése..... | XXIII |
| 3.2.5.3.2 Külső események (idő, képernyőméret, süti)..... | XXIII |
| 3.2.5.3.3 Konfiguráció frissítése..... | XXIV |
| 3.2.5.3.4 Diagramonkénti konfiguráció frissítése..... | XXV |
| 3.2.5.3.5 Diagramok egéreseeményeinek kezelése..... | XXVI |
| 3.2.5.4 View..... | XXVII |
| 3.2.5.4.1 Konfigurációs rész megjelenítése..... | XXIX |
| 3.2.5.4.2 Összefoglaló táblázat megjelenítése..... | XXX |
| 3.2.5.4.3 Vonaldiagramok megjelenítése..... | XXX |
| 3.3 Stílus – css..... | XXXII |
| 3.4 Makefile..... | XXXII |
| 4. Tesztelés..... | XXXIII |
| 4.1 Elm-test..... | XXXIII |
| 4.2 Kézi tesztelés..... | XXXV |
| 4.2.1 Konfiguráció..... | XXXV |
| 4.2.2 Diagramonkénti konfiguráció..... | XXXVI |
| 4.2.3 Animáció..... | XXXVII |
| 4.2.4 Összefoglaló táblázat..... | XXXVII |
| 5. Összefoglalás..... | XXXIX |
| 6. Köszönetnyilvánítás..... | XL |
| 7. Hivatkozások..... | XLI |

1. Bevezetés

1.1 Témaválasztás

A napjainkban is zajló digitális forradalom következtében egyre több adatot gyártunk, rögzítünk és használunk fel. Ezek az adatok igen sokfélék lehetnek, ide tartoznak például a rögzített GPS-koordináták, valutaárfolyamok, vagy akár a webes böngészési előzmények. Ezen adatok jelentős része időbeliséget is tartalmazó skaláris értékekből áll. Ezek az idősorok rengeteg területen felhasználhatóak, legyen szó akár irányított marketingről, tőzsdei elemzésekről vagy időjárás-előrejelzésről. A felhasználás egyik fontos lépése az adatok elemzése, amihez szükség van az adatok vizualizációjára. A vizualizáció minősége jelentős mértékben segítheti vagy gátolhatja a kutatók munkáját. Például ha nem lehet belenagyítani a vonaldiagramokba, az szinte ellehetetleníti az elemzést. Ha pedig bizonyos funkciók el vannak rejtve vagy körülményes a használatuk, az ha nem is lehetetlenné, de nehezebbé teszi az elemzést. Ezzel szemben itt is igaz az "egy kép többet ér ezer szónál". Egy átlátható, kényelmesen állítható ábra felhívhatja a figyelmet az idősor olyan tulajdonságaira, amelyek felett könnyedén el is lehetne siklani.

Hallgatótársammal egy általános idősor esetén kényelmesen használható vizualizációs webalkalmazást kívántunk létrehozni.

1.2 Célok

Elsődleges célom az volt, hogy a webalkalmazás minél több olyan funkciót tartalmazzon, ami segíti egy általános idősor elemzését vagy idősorok összehasonlítását. Ugyanakkor csak ténylegesen felhasználható, a legtöbb idősornál valóban további, kiegészítő információkat tartalmazó lehetőségeket szerettem volna beletenni, hogy a felhasználói felület letisztult maradjon. Emellett célom volt az intuíciót segítő egyszerű animációk megjelenítése, amelyek a legtöbb hasonló megoldásban nem kerülnek előtérbe.

1.3 Ismert problémák

Kezdetől fogva bonyolult kérdés, hogy melyek ezek a hasznos funkciók. Például a megjelenítendő dimenziók kényelmes változtatása biztosan fontos, de a deriváltak van-e információtartalma minden idősor esetén? A nehézséget a feladat általánossága, az idősorok

sokfélesége jelenti. Ezeket a döntéseket igyekeztem átgondoltan, többfajta idősort figyelembe véve, illetve saját tapasztalataim alapján meghozni.

2. Felhasználói dokumentáció

2.1 Rendszerkövetelmények, telepítés

2.1.1 Szerver

A szervert Ubuntu (18.04.1) operációs rendszeren használtam, ennek vagy egyéb Linux-alapú operációs rendszernek a használatát ajánlom.

A szerver futtatásához szükségesek:

- Erlang (OTP 22.3)
- rebar3 (3.9.1)

A szervert a parancssorban a `timeseries` mappában állva a `make build` paranccsal lehet telepíteni. Ekkor a szerver mellett a kliens is telepítve lesz.

2.1.2 Kliens

A kliens telepítéséhez szükséges:

- elm (0.19.1) [1][2]
- Linux-alapú operációs rendszer

Külön a klienst a parancssorban az `apps/timeseries/priv/monitor` mappában állva a `make` paranccsal lehet telepíteni, ennek az eredménye látható az 1. ábrán. A telepítés folyamán generált fájlokat, mappákat ugyanitt a `make clean` paranccsal lehet törölni, melynek eredményét a 2. ábra mutatja be.

```

Creating directories...
mkdir -p dist/assets
Copying html files...
cp src/index.html dist/index.html
cp src/TimeseriesClient.html dist/TimeseriesClient.html
Copying css file...
cp assets/TimeseriesClient.css dist/assets/TimeseriesClient.css
Compiling Index.elm to index.js...
elm make src/Index.elm --output dist/assets/Index.js
Success!

Index —> dist/assets/Index.js

Compiling TimeseriesClient.elm to TimeseriesClient.js...
elm make src/TimeseriesClient.elm --output dist/assets/TimeseriesClient.js
Success!

TimeseriesClient —> dist/assets/TimeseriesClient.js

```

1. ábra: make parancs futtatásának eredménye

```

Deleting generated files, directories...
rm -rf dist

```

2. ábra make clean parancs futtatásának eredménye

A kliens használatához egy böngészőre lesz szükség. Az alkalmazást a következő böngészőkkel teszteltem, ezek bármelyike használható:

- Ubuntu (18.04.1) operációs rendszer alatt:
 - Mozilla Firefox (75.0)
 - Google Chrome (81.0.4044.122)
- Microsoft Windows 10 Education (10.0.17.134) operációs rendszer alatt:
 - Mozilla Firefox (75.0 (64-bit))
 - Google Chrome (81.0.4044.122 (64-bit))

2.2 Futtatás

2.2.1 Szerver

A webszerver a command line – ban a `timeseries` mappában állva a következő paranccsal indítható:

```
make shell
```


Ha elindul a konzol, akkor fut a szerver. Ha a 8080-as port foglalt volt, akkor Protocol 'inet_tcp': the name timeseries@localhost seems to be in use by another Erlang node hibaüzenetet kapunk, a szerver nem tud elindulni.

A szerverre a CLI-n keresztül lehet adatokat feltölteni. Ehhez a command line – ban a timeseries mappában állva a szerver elindítása után a következő parancsot adjuk ki:

```
./timeseries_cli --upload <feltöltendő fájl elérési útvonala>
```

A szerver a parancssorban állva a Ctrl + c, majd a billentyűvel állítható le.

2.2.2 Kliens

A felhasználói felület a választott böngészőben, a szerveret futtató gép IP-címén, a 8080-as porton érhető el. A felület két oldalból áll, az index.html-ből és a TimeseriesClient.html-ből. Így például az index oldal elérése a böngészőből:

<IP-cím>:8080/index.html

2.3 Használat

2.3.1 Főoldal (index.html)

A főoldal felsorolja a szerveren található idősorokat és azok hosszát, ezt illusztrálja a 3. ábra. Ezen információk alapján választhatunk, hogy melyiket szeretnénk vizsgálni. Az adatmegjelenítő oldalon lesz lehetőségünk más idősorra váltani vagy egyszerre több idősor megjeleníteni.

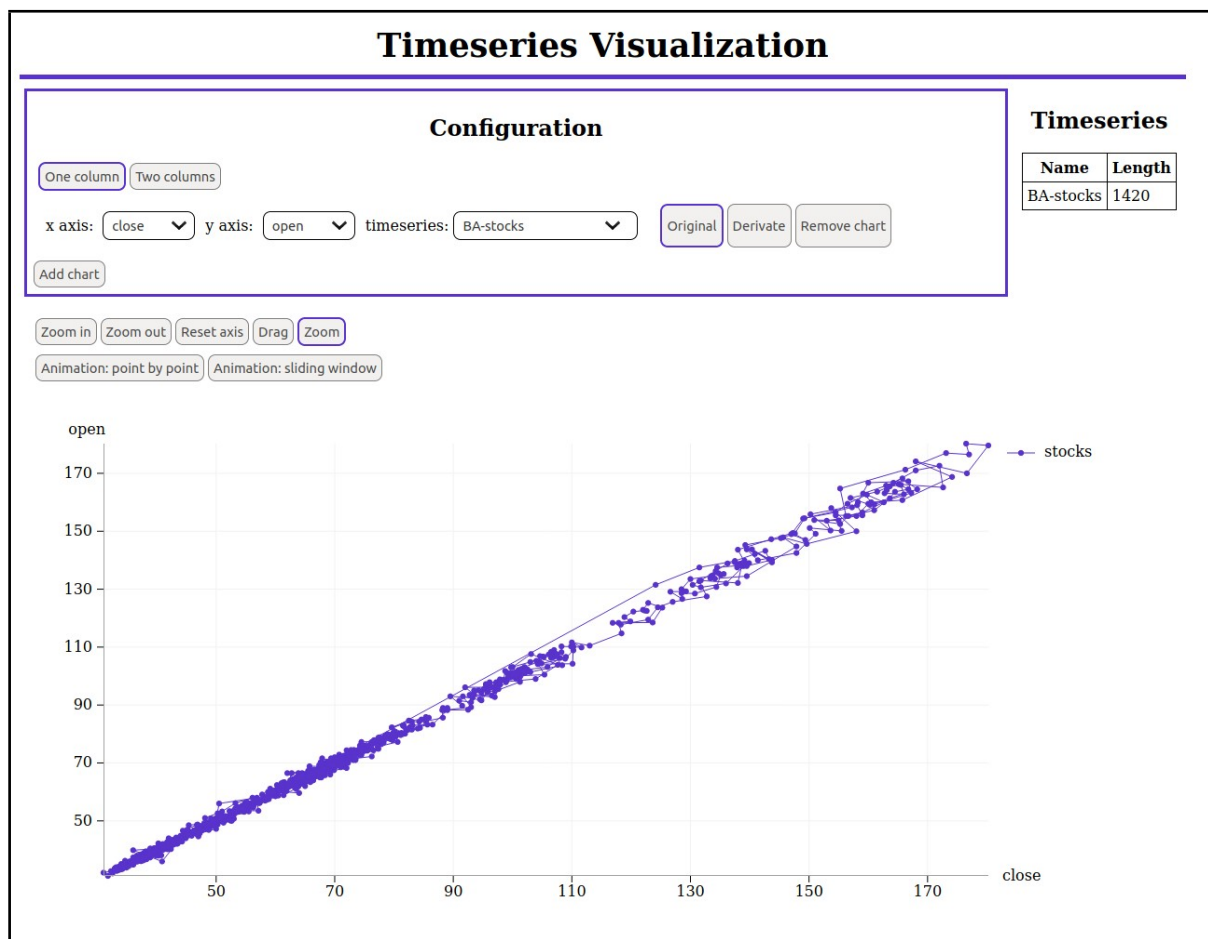
| Timeseries Visualization | |
|--------------------------|--------|
| Choose a timeseries! | |
| Name | Length |
| BA-stocks | 1420 |
| london-humidity-Jan | 24 |
| london-humidity-Jun | 24 |

3. ábra: főoldal

A kiválasztott idősorra kattintva átjutunk az adatmegjelenítő oldalra, ahol ki is rajzolódik egy vonaldiagram a kiválasztott idősorról.

2.3.2 Adatmegjelenítő oldal (TimeseriesClient.html)

Az adatmegjelenítő oldal elérhető a főoldalról vagy közvetlenül is. Előbbi esetben a kiválasztott idősorról jelenik meg egy vonaldiagram, utóbbi esetben pedig az első szerveren található idősorról. Kezdetben mindkét esetben egyoszlopos az elrendezés (egy diagram kitölti a teljes szélességet), a megjelenítendő dimenziók pedig az idősor első, illetve második dimenziói, ezt illusztrálja a 4. ábra. Betöltéskor az eredeti adatok kerülnek megjelenítésre, később át lehet váltani a deriváltra.



4. ábra: adatmegjelenítő oldal betöltéskor

Az oldal három fő részből áll: a konfigurációs rész (kék keretben), az idősorok metaadataiból (kék keret jobb oldalán) és a vonaldiagramokból. Minden vonaldiagramhoz tartozik egy kezelőpanel (a vonaldiagram felett). Míg a konfigurációs ablakban az választható ki, hogy milyen adatot szeretnénk látni, itt a megjelenítés módját (például nagyítás, animációk) tudjuk beállítani.

Ahol két mód közül lehet választani (oszlopok száma, deriválás, animáció típusa), ott az éppen aktív mód gombjának kék kerete van.

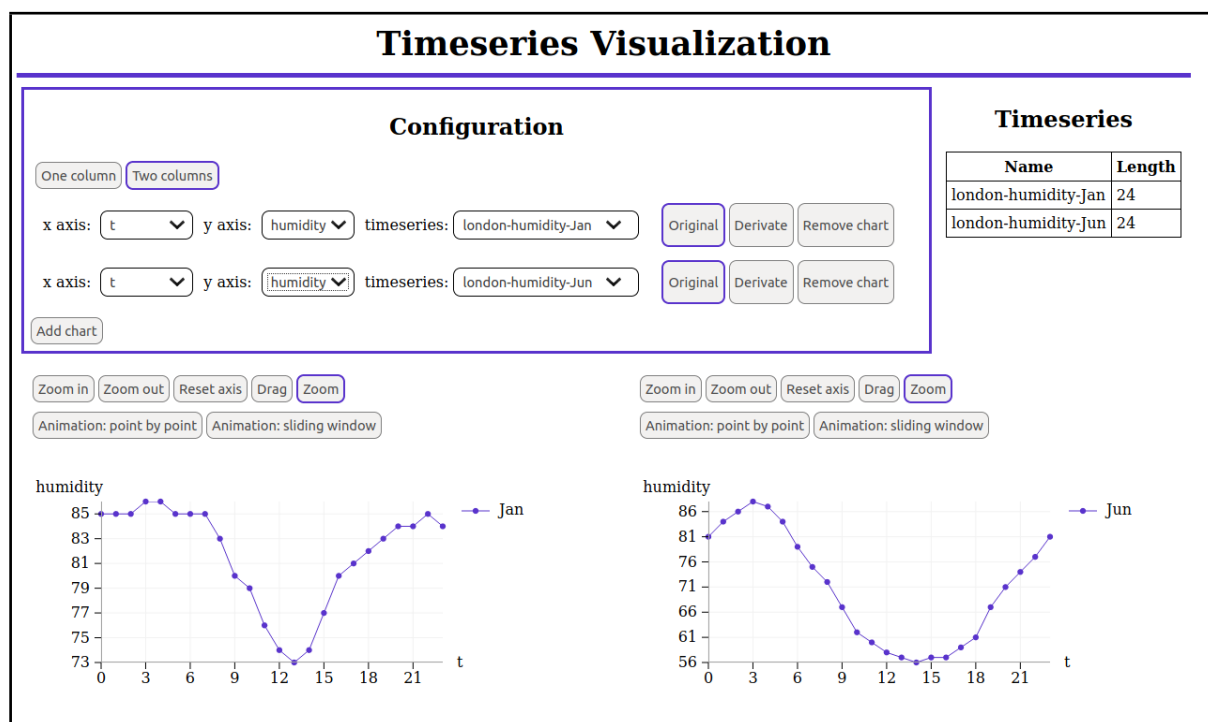
2.3.2.1 Konfiguráció

A konfigurációs részben állíthatjuk be a megjelenítendő adatokat. A lehetőségeken fentről lefelé, balról jobbra haladok végig.

`One column / Two columns`: Kiválaszthatjuk, hogy a vonaldiagramok egy vagy két oszlopban helyezkedjenek el. Utóbbi esetben a vonaldiagramok felsorolása a konfigurációs részben sorfolytonos. A diagramok mindkét esetben kitöltik a rendelkezésre álló szélességet, így két oszlop választásakor a diagramok mérete a felére csökken.

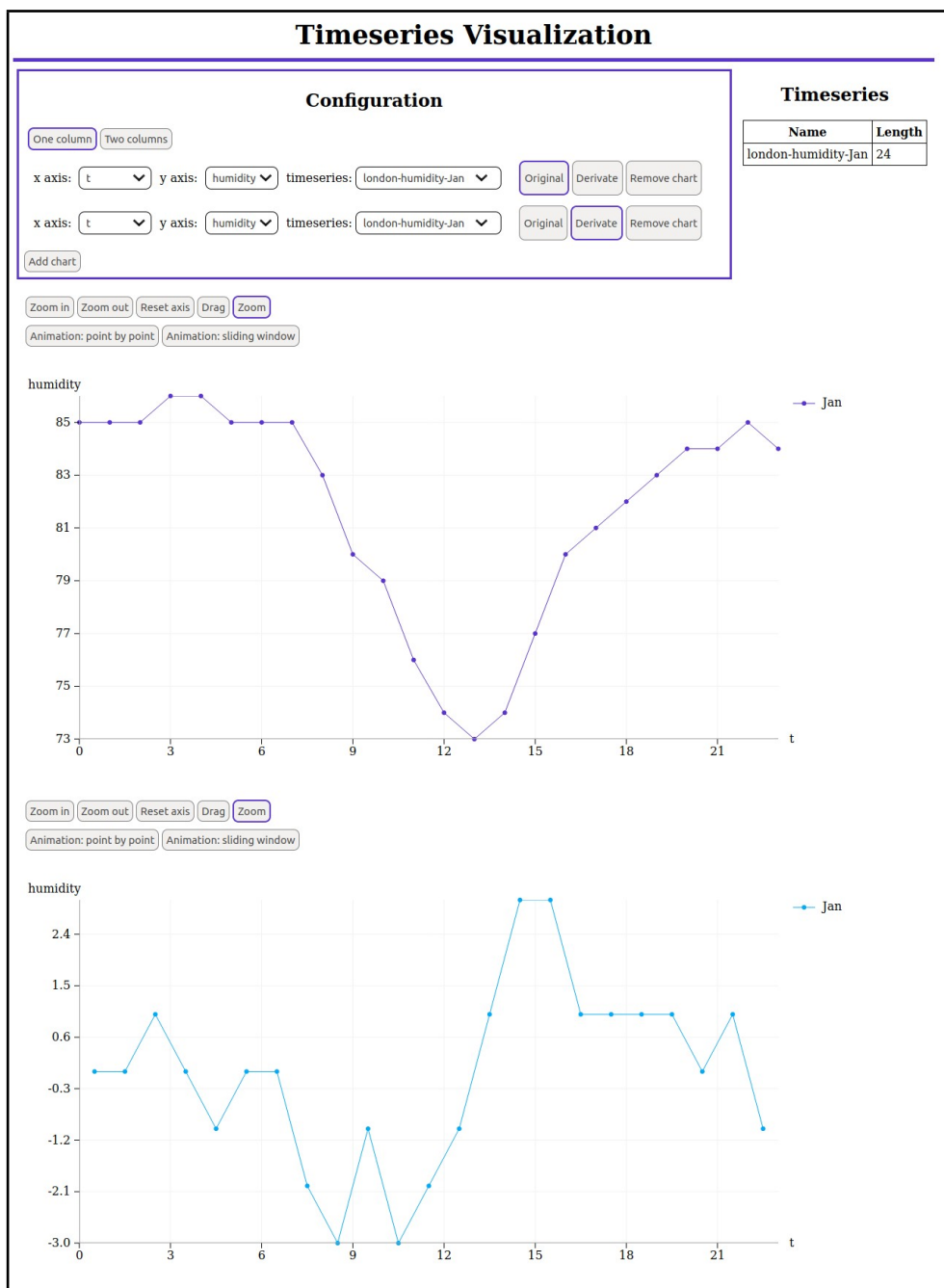
`x/y axis`: Diagramonként változtathatók a megjelenítendő dimenziók (x és y tengely). A lehetőségek között mindig megfelelő idősorok dimenziói szerepelnek, hiszen ezek idősortól függőek lehetnek. A dimenziók bármilyen párosa megjeleníthető.

`timeseries`: Minden diagramnál külön választhatunk a szerveren szereplő összes idősor közül. Ha olyan idősorra váltunk, amelyben megtalálhatóak az addig megjelenített dimenziók, akkor ezek beállítása megmarad, míg új dimenziók esetén az első kettő kerül kiválasztásra.



5. ábra - különböző idősorok azonos dimenzióinak megjelenítése egymás mellett

Original / Derivate: Szintén diagramonként lehet választani a deriválást. Ilyenkor a kiválasztott dimenziók szerint történik a deriválás. Ennek a módnak a használatát a gombon megjelenő kereten kívül az is jelzi, hogy az eredeti értékek sötétkéssel vannak megjelenítve, míg a deriváltak világoskékkel. Deriváláskor az x tengely nem változik, az y tengely pedig igazodik a derivált értékekhez. Az eredeti és a derivált értékek megjelenítését a 6. ábra illusztrálja.



6. ábra: azonos idősor eredeti és derivált értékeinek megjelenítése egymás alatt

Remove chart : Minden diagram kitörölhető a sorból.

Add chart : Új diagram létrehozásakor annak beállítása az addigi utolsó diagramével fog megegyezni. Ha nincs előző diagram, akkor a szerveren megtalálható első idősről jelenik meg egy vonaldiagram.

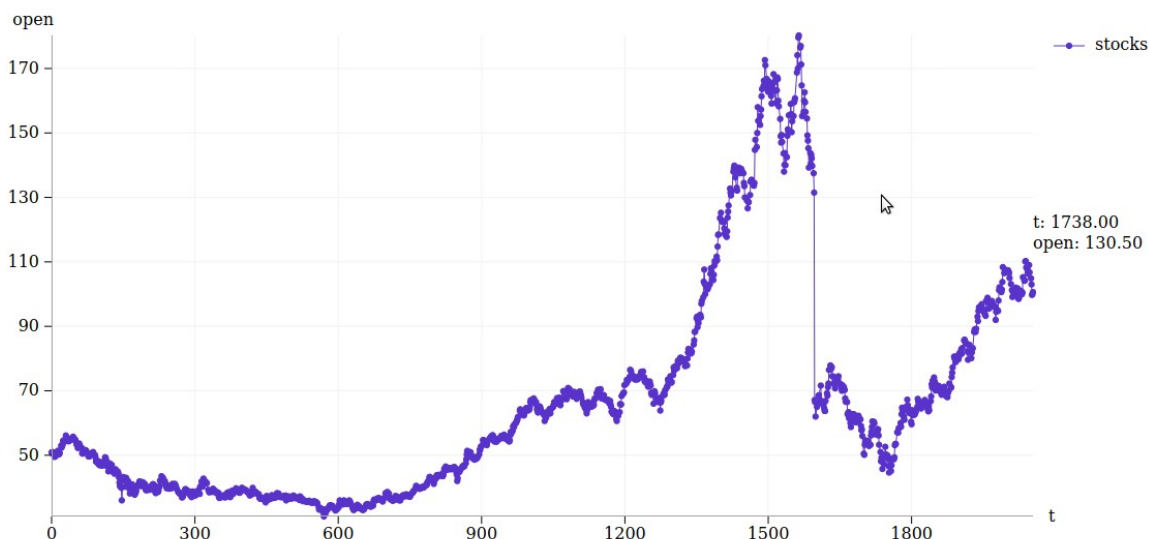
2.3.2.2 Idősorok metaadatai

A konfigurációs résztől jobbra helyezkedik el egy táblázat, ami a megjelenített idősorokat tartalmazza azok hosszával. Itt csak a megjelenített idősorok szerepelnek, nem az összes, amely megtalálható a szerveren, mint a főoldalon. Minden idősor csak egyszer szerepel, akkor is, ha több vonaldiagram is használja az adatait. Ez a rész nem interaktív.

2.3.2.3 Vonaldiagramok tulajdonságai

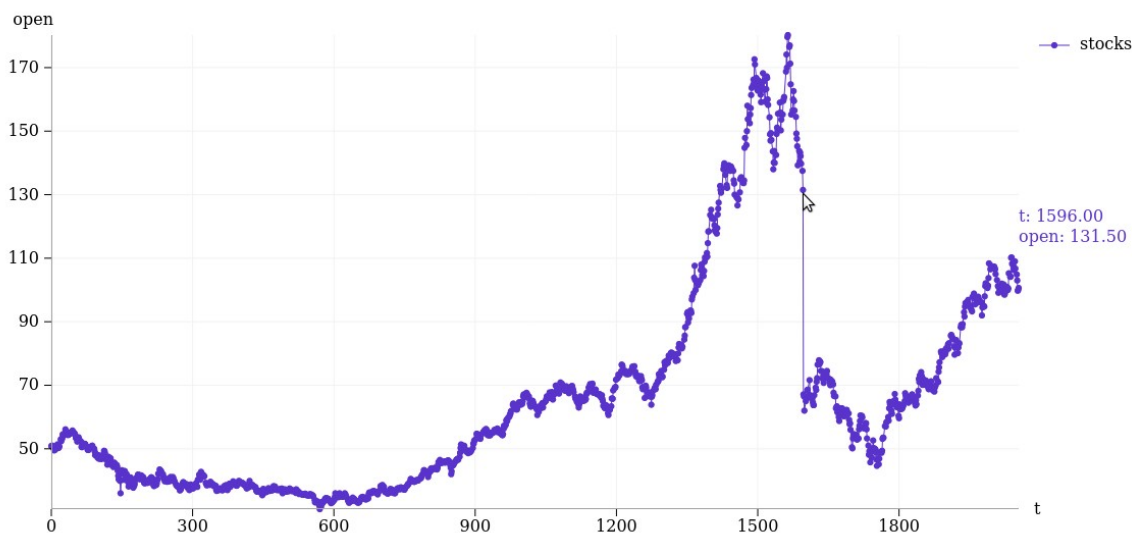
A vonaldiagramoknak vannak passzív, az adatelemzést segítő tulajdonságai.

A kurzort a diagramban mozgatva annak pontos helyének koordinátái megjelennek a diagram jobb oldalán fekete színnel, ahogyan a 7. ábrán látható. Ez eltűnik, ha a kurzor elhagyja a diagram területét.



7. ábra: kurzor koordinátáinak megjelenítése

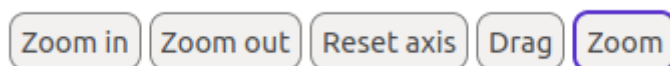
A 8. ábra bemutatja, hogy ha elég közel vagyunk a megjelenített pontok egyikéhez, akkor a diagram jobb oldalán a pontos koordináták helyett a közeli pont koordinátái jelennek meg kék színnel.



8. ábra: mérési pont koordinátáinak megjelenítése

2.3.2.4 Vonaldiagramok kezelése

Lehetőség van diagramonként nagyításra, illetve a tengelyek elmozdításra. Erre a diagramok feletti kezelőpanelek első sora szolgál, melyet a 9. ábra mutat be. A lehetőségeken balról jobbra haladok végig.



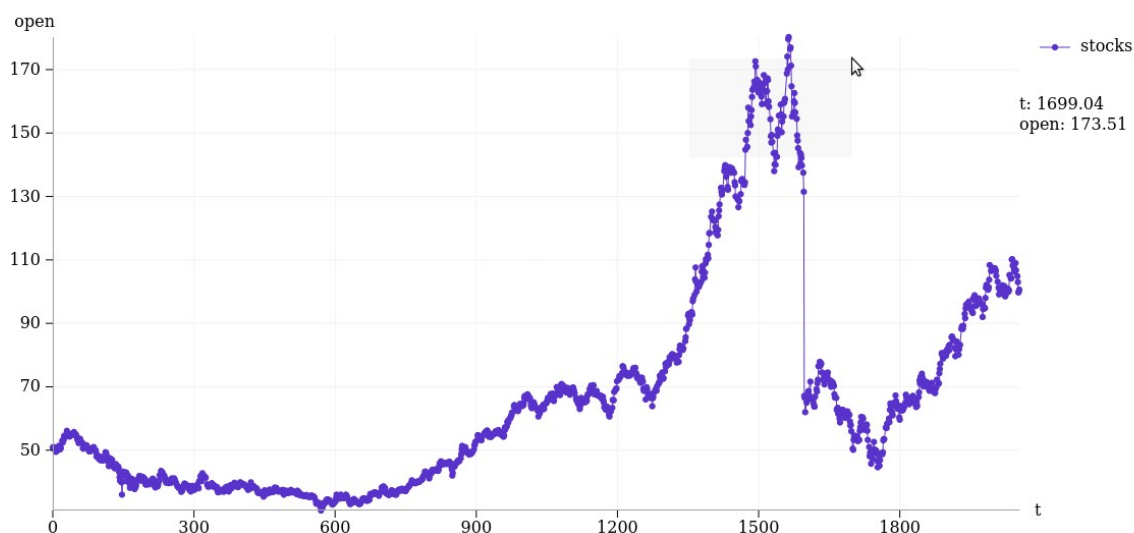
9. ábra: diagramonként kezelőpanel első sora

Zoom in / Zoom out: Lehet gombnyomással nagyítani, illetve kicsinyíteni a vonaldiagramban. Ezek mértéke olyan, hogy éppen kiegyenlítsék egymást. Tehát egy nagyítás – kicsinyítés pár után az eredeti diagramot kapjuk vissza.

Reset axis: Vissza lehet állítani a tengelyeket az eredeti beosztásukra. Ilyenkor a legkisebb olyan intervallumok kerülnek kiválasztásra, amelyekkel minden pont láthatóvá válik. Animáció esetén ez az összes pontra vonatkozik, nem csak az éppen megjelenítettek.

Drag: Ezzel a gombbal válthatunk a tengelyek kézzel történő elmozdítására.

Zoom: Ezzel a gombbal válthatunk a kézi nagyításra. Az egér bal gombját lenyomva tartva kijelölhető egy téglalap (szürkével jelezve), ez lesz széthúzva a diagram teljes méretére az egérgomb felengedésekor. Ha nagyon kicsi területet jelölünk ki (valamelyik tengely mentén a teljes hossz 5%-ánál kisebbet), akkor a nagyítás nem történik meg.



10. ábra: kézi nagyításkor megjelenő szürke területkijelölő téglalap

2.3.2.5 Animációk

Az animációk kezelésére a diagramonkénti kezelőpanel második sora szolgál, ami a 12. ábrán látható. Ha éppen nincs aktív animáció, akkor ez csak két gombból áll, amelyek az animációk elindítására szolgálnak. Ezt a 11. ábra mutatja be.

Animation: point by point Animation: sliding window

11. ábra: diagramonként kezelőpanel második sora animáció nélkül

Animation: point by point Animation: sliding window Pause Stop animation

12. ábra: diagramonként kezelőpanel második sora animáció esetén

Aktív animáció esetén a panel elemei balról jobbra:

Animation: point by point: Ezzel a gombbal indíthatunk el pontonként megjelenítést. Az új pontok másodpercenként jelennek meg.

Animation: sliding window: Egyszerre az idősor egy fix hosszú szakasza jelenik meg, ez másodpercenként "vándorol".

Pause / Continue / Start over: A következő gomb funkciója az animáció állásától függ. Ha az animáció aktív, de még nem ért az idősor végére, akkor a Pause gombbal szüneteltethető. Szüneteltetés esetén az animáció a Continue gombbal folytatható.

Ha pedig az animáció elérte az idősor végét, akkor a `Start over` gombbal indítható újra ugyanez az animáció.

`Stop`: Ezzel a gombbal állíthatjuk meg az animációt, az első két gomb inaktívvá válik és az idősor összes pontja megjelenítésre kerül.

Animációk közben is lehetőség van a kezelőpanel első sorának használatára, ezek értelemszerűen ugyanúgy működnek, mint animáció nélkül.

3. Fejlesztői dokumentáció

3.1 Használt fogalmak

Kigyűjtöttem néhány fogalmat, konvenciót, amit a fejlesztői dokumentációban (és a kódban) az olvashatóság kedvéért alkalmazok.

- idősor/timeseries: A megjeleníteni kívánt adatsor.
- data/adatpont: Az idősor egy pontja.
- pont/point: Megjelenítendő pont x és y koordinátákkal.
- pont-sorozat: Megjelenítendő pontok listája.
- tengelyintervallum: A diagramok tengelyein megjelenő intervallum. Az ezekbe eső pontok kerülnek megjelenítésre.
- m előtag: Az elm nyelv `Maybe <XY>` típust használ olyan esetekben, ahol egy műveletnek nem biztos, hogy van eredménye. Ilyen művelet például egy `<XY>` típusú lista első elemének kinyerése. Ennek eredménye vagy egy `<XY>` típusú változó vagy üres lista esetén `Nothing`. Ennek a leírására szolgál a `Maybe <XY>` típus, melyet explicit típuskonverzióval kell `<XY>` típussá tenni. A még nem konvertált változó rövid jelölésére használtam az m előtagot a kódban.
- elm-kód: Az elm-nyelven írt kódok, illetve az ezekből fordított javascript-kódok.
- javascript-kód: A html-kódokba beépített pársoros javascript kód. Ez köti be az elm-kódot az oldalba.
- húzogatás: A megjelenített terület kézzel való mozgatása.

3.2 Funkcionalitás – elm

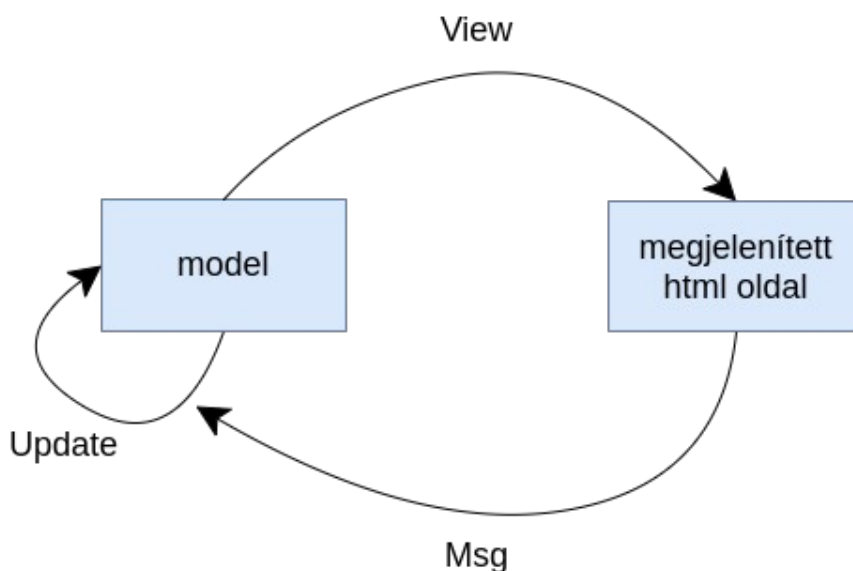
3.2.1 Az elm nyelv

A kliens fejlesztéséhez az elm funkcionális nyelvet választottam. Ennek előnye, hogy nem lehetnek futási idejű hibák és a fordítási hibák többsége is könnyen értelmezhető. [3] Az elm kódolási konvenciói pedig úgy lettek kialakítva, hogy jól használhatóak legyenek

verziókövető rendszerekkel. [4] Ezenkívül, bár a nyelv nem követeli meg, én minden függvényhez kiírtam a típus annotálást, ezzel is segítve a hibaüzenetek értelmezését.

Egy elm program három nagy egységből áll, ezek láthatóak a 13. ábrán:

- `model`: Tárolja az alkalmazás jelenlegi állapotát.
- `View`: Megadja, hogy a jelenlegi állapotból hogyan készüljön megjeleníthető html oldal.
- `Update`: Az oldal felől érkező üzenetek (`Msg`) alapján frissíti a `model`-t.



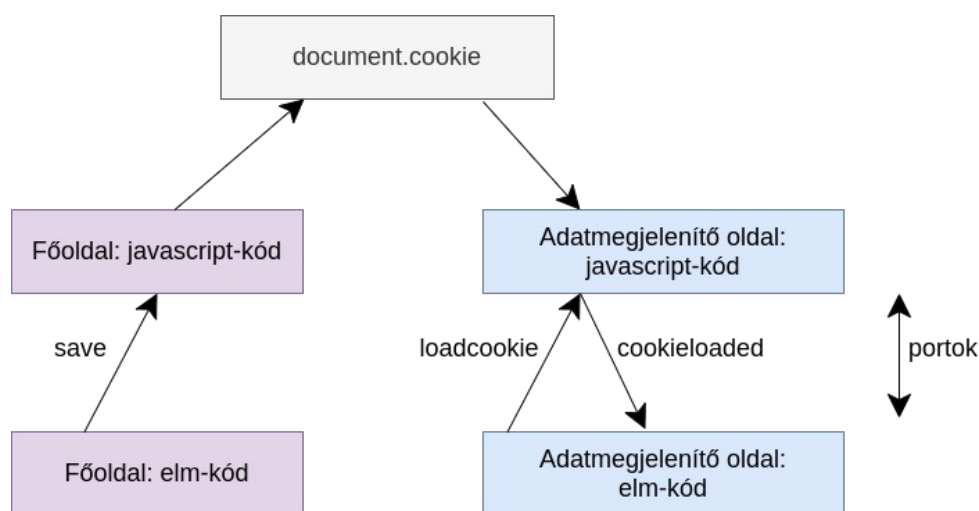
13. ábra: az elm alkalmazás felépítése

A vonaldiagramok megjelenítéséhez a `terezka/line-charts` csomagot használtam. [5] Ezzel a csomaggal kényelmesen hozhatóak létre diagramok (`Svg Msg` típus), ugyanakkor van lehetőség egyedi megoldások alkalmazására is.

3.2.2 Kommunikáció javascript-kód és elm-kód között (portok)

A főoldalról egy gombnyomással lehet átjutni az adatmegjelenítő oldalra. Eközben továbbítani kell, hogy melyik gomb lett megnyomva, azaz melyik idősor töltődjön le a szerverről és jelenjen meg róla egy diagram. Ennek az információnak a továbbítására több lehetőség is van. Az egyik elképzelés az volt, hogy jelenítődjön meg az idősor neve az adatmegjelenítő oldal url címében. Ekkor azonban kérdéses, hogy hogyan legyen kezelve, ha később egyszerre több idősorról szeretnénk diagramokat megjeleníteni. Ezért inkább egy süti (cookie) alkalmazása

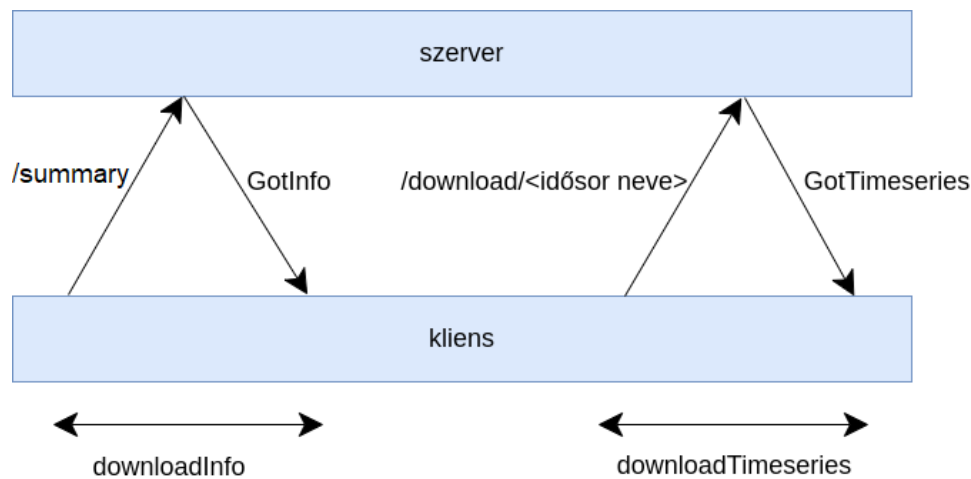
mellett döntöttem. Ezt viszont csak a html oldalba közvetlenül írt javascript-kód éri el, az elm-kód nem. Ezért szükséges a kettő közti kommunikáció, amely portok segítségével oldható meg. A főoldalon egy gomb megnyomásakor az elm-kód a `save` porton keresztül elküldi a kiválasztott idősor nevét a javascript-kódnak, ami elmenti ezt a `document.cookie` változóba. Ezután az adatmegjelenítő oldal elm-kódja a betöltésekor a `loadcookie` porton keresztül kérdést indít az adatmegjelenítő oldal javascript-kódja felé, ami erre válaszul visszaküldi a `cookieloaded` porton a `document.cookie` tartalmát. Ezt a folyamatot illusztrálja a 14. ábra.



14. ábra: süti használata

3.2.3 Kommunikáció a szerverrel

A szerverrel való kommunikáció http lekérdezéseken keresztül történik, ezeket mutatja be a 15. ábra. Az idősorok neveinek és hosszainak letöltését a `downloadInfo` végzi. A lekérdezés a `"/summary"` url-en keresztül történik, eredménye egy dictionary (`Dict`), mely alapján befejezésekor frissül a `model` (`GotInfo` üzenet). Teljes idősor letöltését a `downloadTimeseries` végzi. Ekkor a lekérdezés a `"/download/<idősor neve>"` url-en történik, melynek eredménye egy karaktersorozat (`String`). A `model` frissítése a `GotTimeseries` üzenet alapján történik. A karaktersorozat megfelelő típusú (`Timeseries`) alakítása a frissítés folyamán történik.



15. ábra: szerver-kliens kommunikáció

3.2.4 Főoldal (Index.elm)

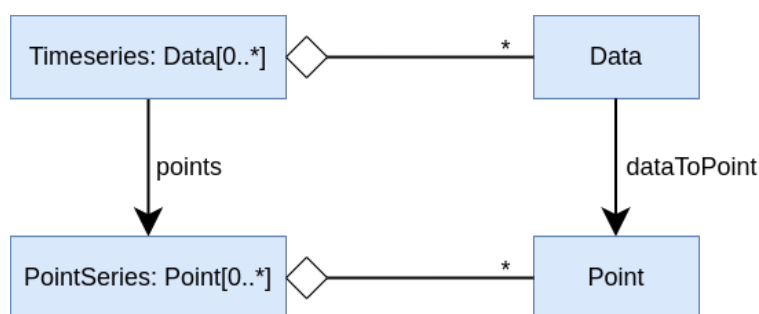
A főoldal szerkezete jóval egyszerűbb, mint az adatmegjelenítőé. Annyi a feladata, hogy felsorolja a szerveren található idősorokat, és át lehessen navigálni róla az adatmegjelenítő oldalra (a megfelelő süti elmentése után). A modelben csak a `timeseriesInfo` szerepel, ami a szerveren található idősorok neveit és hosszait tartalmazza. Inicializáláskor ez üres, majd elindítok egy lekérdezést a szerver felé (`downloadInfo`). A válasz megérkezésekor (`GotInfo`) frissítem a model-t. Ezen kívül csak egyfajta üzenet (`Msg`) lehetséges, a `TimeseriesChosen`. Ez akkor váltódik ki, ha a felhasználó rákattint az egyik idősorhoz tartozó gombra. Ilyenkor a kiválasztott idősor nevét továbbítom a javascript-kód felé (`save port`), amivel elmentem a `document.cookie` változóba. A név továbbítása után betöltésre kerül a `TimeseriesClient.html`.

3.2.5 Adatmegjelenítő oldal (TimeseriesClient.elm)

3.2.5.1 Típusok

A megjelenítendő adatpontok, idősorok, pontok, pont-sorozatok könnyebb követésének érdekében `type alias`-okat hoztam létre hozzájuk, ezeket jeleníti meg a 16. ábra. A megfelelő párok között pedig átalakító függvényeket definiáltam. A szerveren tárolt idősor egy pontjának (adatpont) felel meg a `Data` típus, amely egy `String`-kulcsokkal (dimenziók) rendelkező, `Float` értékeket tartalmazó `dictionary`. Ilyen adatpontoknak a sorozata a `Timeseries` (idősor). Megjelenítéskor azonban valójában nem adatpontokat

vagy idősorokat használtam, hanem `x` és `y` koordinátákkal rendelkező `Point` típusok sorozatát, a `PointSeries` típust. Az idősor pont-sorozattá alakításához két függvényt használtam. Az első az adatpontot ponttá alakító `dataToPoint` függvény, a második pedig az ezt felhasználó, idősort pont-sorozattá alakító `points` függvény. A megfelelő átalakításhoz mindkét függvénynek szüksége van további információkra. A `dataToPoint` függvény várja a megjelenítendő dimenziókat, míg a `points` függvény a megfelelő diagram konfigurációját (`ChartConfig` típus), melyből felhasználja a megjelenítendő dimenziókat, illetve azt, hogy a deriváltat vagy az eredeti pontokat kell-e megjeleníteni.



16. ábra: adattípusok és átalakításuk

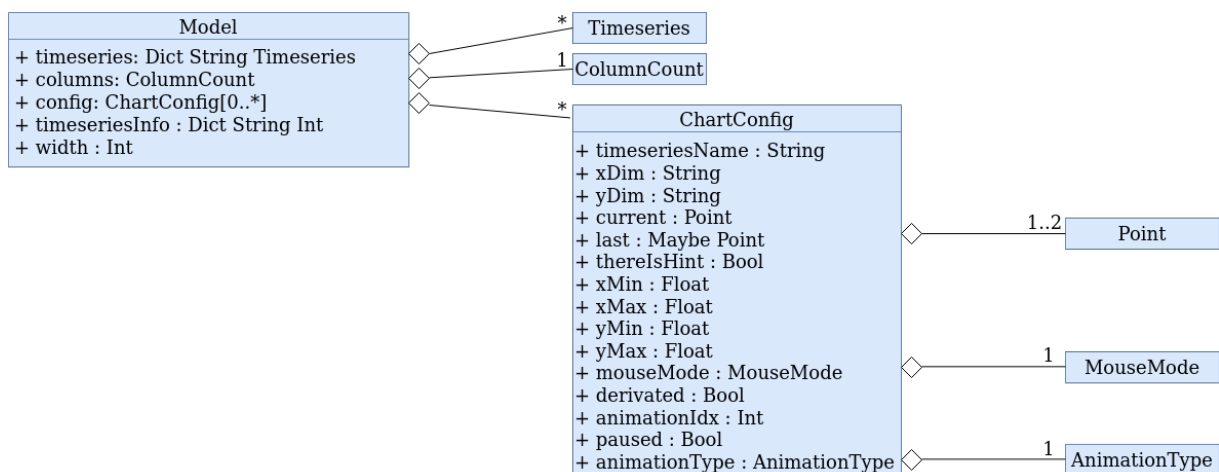
Az alkalmazás állapotát tároló model és annak építőelemeinek reprezentálásához `type alias`-okat (`Model`, `ChartConfig`) és `record`-okat (`ColumnCount`, `MouseMode`, `AnimationType`) készítettem, ezeket a 17. ábra mutatja be. A `record`-ok előnye, hogy `case`-elágazások esetén egyértelműen lefedhető minden lehetséges értékük, nincs szükség alapértelmezett ágra.

A `Model` típus mezői:

- `timeseries`: Tárolja a szerverről letöltött idősorokat.
- `columns`: Megadja, hogy a diagramokat egy- vagy kétszlopos elrendezésben kell-e megjeleníteni.
- `config`: Felsorolja a vonaldiagramok egyenkénti konfigurációját.
- `timeseriesInfo`: Tárolja a serveren található idősorok neveit és hosszait.
- `width`: Tárolja az ablak szélességét. Ehhez idomulnak a vonaldiagramok méretei.

A `ChartConfig` típus írja le egy diagram külön konfigurációját. Ennek mezői (csoportosítva):

- `timeseriesName`: A megjelenítendő idősor neve.
- `xDim`, `yDim`: A megjelenítendő dimenziók.
- `current`: Az egér legutóbbi (diagram területére eső) helyzete.
- `last`: Az egérgomb lenyomásának helye a diagramon belül. Ha az egérgomb nincs lenyomva, akkor az értéke `Nothing`.
- `thereIsHint`: Azt adja meg, hogy ki kell-e írni az egér helyének koordinátáit a diagram mellé (hint), tehát az egér a diagram területén található-e.
- `xMin`, `xMax`, `yMin`, `yMax`: A tengelyintervallumok kezdő- és végpontjai.
- `mouseMode`: Megadja, hogy az egérrel nagyítani vagy húzogatni lehet-e.
- `derivated`: Megadja, hogy az eredeti pont-sorozatot vagy az adatok deriváltját kell-e megjeleníteni.
- `animationIdx`: Tárolja, hogy hol tart az animáció. Pontonkénti animációnál a megjelenítendő pontok száma (az utolsó indexe), csúszóablaknál pedig a megjelenítendő pontok előtti pont indexe.
- `paused`: Megadja, hogy éppen szüneteltetve van-e az animáció.
- `animationType`: Megadja az aktuális animáció típusát. Ha nincs aktív animáció, akkor az értéke `None`.



17. ábra: az alkalmazás állapotának reprezentálása

Az eddig felsorolt típusokon kívül még létrehoztam egy `Msg` típust, ami a lehetséges üzeneteket tartalmazza. Ezeket fogja kezelni az `update` függvény.

A olvashatóbb kód kedvéért még definiáltam egy `toMaybe` függvényt, mellyel bármely `<XY>` típusú változó átalakítható `Maybe <XY>` típusúvá.

3.2.5.2 Inicializálás

A diagramonkénti konfiguráció inicializálására szolgál az `initChartConfig` függvény, amely a megjelenítendő dimenziók, az idősor neve és az idősor alapján hozza létre a konfigurációt. A megjelenítendő dimenziók értéke lehet `Nothing`, ha nincsen erről korábbi információ. Ekkor (vagy ha a megadott dimenzió nem szerepel az idősorban) az idősor első, illetve második dimenzióját választom ki. A megjelenítendő dimenziók figyelembe vételére szükség van például, amikor új idősor kerül kiválasztásra, de a régi dimenziók az új idősorban is szerepelnek. A tengelyintervallumokat inicializáláskor a legszűkebb, minden pontot tartalmazó intervallumokra állítom be.

Az első inicializáláskor is létre kell hozni egy ilyen diagramonkénti konfigurációt, de ekkor még egy idősor sem lett letöltve a szerverről. Ezért létrehoztam egy térkitöltő konfigurációt `emptyChartConfig` néven. Ezt azonban le is cserélem, amint letöltöttem az első idősort a szerverről. Ugyanezt a konstans függvényt tudtam használni máshol a kód olvashatóbbá tételéhez is.

Az oldal betöltésekor az `init` függvény létrehoz egy alapértelmezett `model`-t. Ebben az oszlopok száma egyre van állítva, a `config`-ban pedig egyetlen, csak térkitöltőnek szolgáló diagramonkénti konfiguráció szerepel. Az idősorokra, illetve ablakszélességre vonatkozó mezőket üres `dictionary`-kkal és a 0 értékkel inicializálom térkitöltő jelleggel az első frissítésekig. Ezek után rögtön el is indítok három folyamatot: a süti betöltését (`loadcookie` port), a serveren található idősorok listájának a letöltését (`downloadInfo`) és az ablak méreteinek a lekérdezését (`getWidthAndHeight`).

3.2.5.3 Update

Több frissítésnél is csak a diagramonkénti konfigurációk egyikét kell változtatni. Ez a helyzet például a diagramok területéről érkező eseményekkel. Az ilyen változtatások kényelmes elvégzéséhez hoztam létre az `updateConfig` függvényt, amely a `model`-ben csak a megadott indexű diagramonkénti konfigurációt frissíti a megadott függvény szerint.

A `model` frissítése folyamán figyelembe kell venni, hogy csak a `NewTimeseriesName` üzenet esetén indulhat el egy idősor letöltése a szerverről. Ezért minden esetben, mikor változtatni akarjuk a megjelenítendő idősort, meg kell hívni az `update` függvényt ezzel az üzenettel. Ellenkező esetben, ha eddig nem volt elmentve az adott idősor a `model`-be, nem fogjuk tudni megjeleníteni azt.

A frissítés az üzenet (`Msg`) típusa alapján történik. Az alábbiakban ezeken a lehetséges üzeneteken és kezelésükön haladok végig a funkcionalitásuk szerint csoportosítva őket.

3.2.5.3.1 Szerverről érkezett adat kezelése

`GotInfo` üzenet érkezik, amikor befejeződik a szerveren található idősorok neveinek és hosszainak letöltése. Ekkor elmentem ezt az információt a `model timeseriesInfo` mezőjébe és ha az első diagramonkénti konfiguráció csak térkitöltő volt (tehát az idősor neve üres), valamint a szerveren legalább egy idősor található, akkor egy újabb `update` meghívásával lecserélem ezt a térkitöltőt egy, a szerveren található első idősornak megfelelő kezdőkonfigurációra.

`GotTimeseries` üzenet akkor érkezik, amikor egy idősor letöltése fejeződött be. A kapott karaktersorozatot itt alakítom idősorrá és kiegészítem vele a `model timeseries` mezőjét. Ezen kívül minden eddigi térkitöltő diagramonkénti konfigurációt lecserélek egy ennek az idősornak megfelelő kezdőkonfigurációra, illetve minden erre az idősorra vonatkozó konfigurációt (ha az idősor neve azonos) frissítek a valódi idősor alapján.

3.2.5.3.2 Külső események (idő, képernyőméret, süti)

Másodpercenként érkezik egy `Tick` üzenet, ezt az animációk léptetéséhez használom fel. Ha egy diagramnál éppen nincs aktív animáció vagy szüneteltetve van az éppen futó, akkor nincs teendő. Ellenkező esetben az animáció típusa és a teljes idősor hossza alapján kiszámolom az utolsó lehetséges `animationIdx`-t (`lastIdx`). Ha az animáció még nem érte el ezt, akkor egyszerűen léptetem az indexet. Ha elérte, akkor szüneteltetem az animációt.

`GetViewport` és `Resized` üzenetek esetén frissítem a `model width` mezőjét. `GetViewport` üzenet csak egyszer, az inicializálás után fog érkezni, `Resized` pedig minden alkalommal, amikor a felhasználó átméretezi az ablakot.

`CookieLoaded` üzenettel érkezik meg a süti a javascript-kód felől. A kapott "timeseries=<idősor neve>" szövegből kivágom az idősor nevét. Ha ez sikeres volt, és az így

kapott idősor szerepel a `model timeseriesInfo` mezőjében (vagy az a mező még üres), akkor frissítem az első diagramonkénti konfigurációt ennek a névnek megfelelően.

3.2.5.3.3 Konfiguráció frissítése

Ha a felhasználó a konfigurációs ablakban új idősort választ, akkor `NewTimeseriesName` üzenet érkezik. Ekkor az új idősor adatpontjai alapján kiszámítom az új idősor dimenzióit (`newDims`). Ha az eddig kiválasztott dimenziók szerepelnek az újak között, akkor ezeket megtartva, egyébként ezek nélkül hozok létre egy, az új idősorra vonatkozó kezdőkonfigurációt. Ezenkívül ellenőrzöm, hogy az új idősor megtalálható-e a `model timeseries` mezőjében, és ha nem, akkor a diagramonkénti konfigurációban csak az idősor nevét frissítem és elindítom az idősor szerverről való letöltését (`downloadTimeseries`). Ekkor a diagramonkénti konfiguráció újra frissítve lesz az idősor letöltése után (`GotTimeseries` üzenet kezelésekor).

`NewXAxis` / `NewYAxis` üzenet érkezik, ha a felhasználó megváltoztatja az egyik megjelenítendő dimenziót. Ekkor kiszámolom az új megjelenítendő pont-sorozatot (`newPoints`), és ezek alapján frissítem a diagramonkénti konfigurációban a megjelenítendő dimenziót, illetve a hozzá tartozó tengelyintervallumot.

Új diagram hozzáadásakor (`AddChart` üzenet), ha eddig nem volt egy vonaldiagram sem, akkor létrehozok egy térkitöltő konfigurációt, majd a `model timeseriesInfo` mezőjében szereplő első idősor nevével frissítem azt (`update` meghívása `NewTimeseriesName` üzenettel). Azért van szükség erre a két lépésre, hogy az idősor nevének változtatása itt is a `NewTimeseriesName` üzeneten keresztül történjen, és le lehessen tölteni az idősort a szerverről, ha szükséges. Ha volt már legalább egy diagramonkénti konfiguráció a `model`-ben, akkor az utolsóval megegyező konfigurációt adok hozzá a diagramonkénti konfigurációk listájához.

Diagram törlésekor (`RemoveChart` üzenet) kitörölöm a megfelelő indexű elemet a `model config` mezőjéből (tehát a diagramonkénti konfigurációk listájából).

Az eredeti pontok és a derivált megjelenítése közötti váltásra szolgál a `PlotOriginal` és a `PlotDerivate` üzenet. Ezek kezelésekor a diagramonkénti konfigurációban a `derivated` mező mellett frissítem az y tengelyintervallumot is. Az x tengelyintervallumot nem változtatom, mert bár a megjelenítendő pontok x koordinátái változnak, a váltás így lesz szemmel jobban követhető.

Az egy-, illetve kétoszlopos elrendezés közötti váltásnál (`OneColumn` és `TwoColumns` üzenetek) csupán a `model.columns` mezőjét kell frissítenem. Az elrendezés megvalósítása a `view`-ban fog történni.

3.2.5.3.4 Diagramonkénti konfiguráció frissítése

Gombbal való nagyításkor (`ZoomIn` üzenet) mindkét tengelyintervallumot a 0.8-adára csökkentem úgy, hogy mindkét végpontot 10%-kal beljebb húzom. Gombbal való kicsinyítéskor (`ZoomOut` üzenet) a jelenlegi tengelyintervallumot tekintem a 0.8-adnak és így növelem 10-10%-kal a tengelyintervallumok végét. Így a nagyítás és kicsinyítés éppen egymás fordítottja.

A kézi nagyítás és húzogatás közötti váltáskor (`DragMode` és `ZoomMode` üzenet) csak a diagramonkénti konfiguráció `mouseMode` mezőjét kell frissíteni.

Az eredeti tengelyintervallumok visszaállításkor (`ResetAxis`) kiszámítom a megjelenítendő pont-sorozatot (`currentPoints`) és ennek alapján állítom be a tengelyintervallumokat a lehető legszűkebb olyan intervallumokra, amelyek minden pontot tartalmaznak.

`StartSlidingWindow` és `StartPointByPoint` üzenetek esetén ell kell indítani a megfelelő animációt. Ehhez a diagramonkénti konfiguráció `animationType` mezőjét `SlidingWindow`-ra vagy `PointByPoint`-ra állítom, a `paused` mezőjét `False`-ra, az `animationIdx`-t pedig `-1`-re. Azért használok itt `-1`-et, mert az érkező `Tick`-ek nem függenek az animáció indításának időpontjától és így legalább egy másodperc eltelik az animáció első változásáig.

`PauseContinue` üzenet érkezik, ha a felhasználó a `Continue/Start over/Pause` feliratú gombra kattint. Ekkor a `model` frissítése az animáció állapotától függ. Ha az animáció már elérte az utolsó lehetséges indexet, akkor újra kell indítani a `paused` és `animationIdx` mezők megfelelő beállításával. Egyéb esetben a `paused` mezőt az ellenkezőjére állítom, ezzel szüneteltetem vagy folytatom az animációt.

`StopAnimation` üzenet esetén az `animationType` mezőt `None`-ra állítom, és ezzel befejezem az animációt.

3.2.5.3.5 Diagramok egéreseeményeinek kezelése

Az alábbiakban kifejtett üzenetek minden diagram esetén csak a saját területükről érkehetnek. A diagram indexével követem, hogy az adott esemény melyik diagram területén történt. Minden eseménynél csak azt a diagramonkénti konfigurációt kell frissíteni, amelyikről az esemény érkezett.

Hold üzenet érkezik, ha a felhasználó lenyomja az egér bal gombját. Lenyomva tartáskor nem érkezik újabb Hold üzenet. Ekkor a kézi vezérlés mindkét állapotában (nagyítás vagy húzogató) a diagramonkénti konfiguráció `last` mezőjét állítom be az adott pontra.

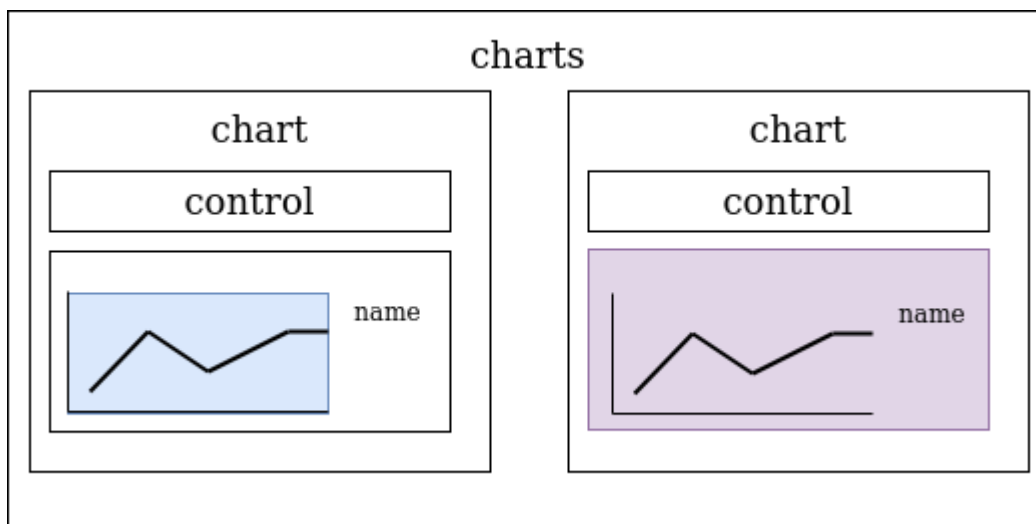
Move üzenet érkezik, ha a felhasználó elmozdítja az egeret. Ekkor nagyítás esetén, akár le van nyomva az egérgomb, akár nem, a `current` mezőt kell az adott pontra frissíteni, a `thereIsHint`-et pedig `True`-ra állítani. Ezek alapján fognak megjelenni a vonaldiagram mellett az egér koordinátái. Nagyítás esetén, ha az egér gombja le van nyomva (tehát a `last` mező értéke nem `Nothing`), el kell tolni a tengelyintervallumokat. Az eltolás mértékét a jelenlegi pont és a `last` pont közötti különbség adja meg. Ilyenkor nem kell frissíteni a `current` mezőt, mivel a húzogató lényege éppen az, hogy az egér koordinátái ne változzanak, hiába mozdítja el azt a felhasználó.

Drop üzenet érkezik, amikor a felhasználó felengedi az egérgombot. Ekkor, ha a `last` mező értéke eddig is `Nothing` volt, nincs teendő. Ellenkező esetben húzogató esetén csupán a `last` mezőt kell `Nothing`-ra állítani. Így a további egérmozgatásoknál nem fognak elmozdulni a tengelyintervallumok. Nagyítás esetén a jelenlegi pont és a `last` pont különbségéből kiszámolom a kijelölt téglalap oldalainak a hosszát. Ha mindkettő nagyobb a hozzá tartozó tengelyintervallum hosszának 5%-ánál (tehát valószínűsíthető, hogy a felhasználó tényleg nagyítani akart), akkor a tengelyintervallumokat a kijelölt téglalapnak megfelelően frissítem. Ha a téglalap kicsi, akkor csak a `last` mezőt állítom `Nothing`-ra.

LeaveChart üzenet érkezik, amikor az egér elhagyja a diagram területét. Ekkor a `thereIsHint`-et `False`-ra állítom, így a továbbiakban nem jelenítem meg az egér koordinátáit.

LeaveContainer üzenet érkezik, amikor az egér a diagramot körbeölelő területet hagyja el. Ekkor a `last` mezőt `Nothing`-ra állítom, tehát úgy veszem, mintha a felhasználó felengedte volna az egér gombját, de kézi nagyítás esetén se változtatok a

tengelyintervallumokon. A diagram területe és a diagramot körbeölelő terület közötti különbség a 18. ábrán látható.

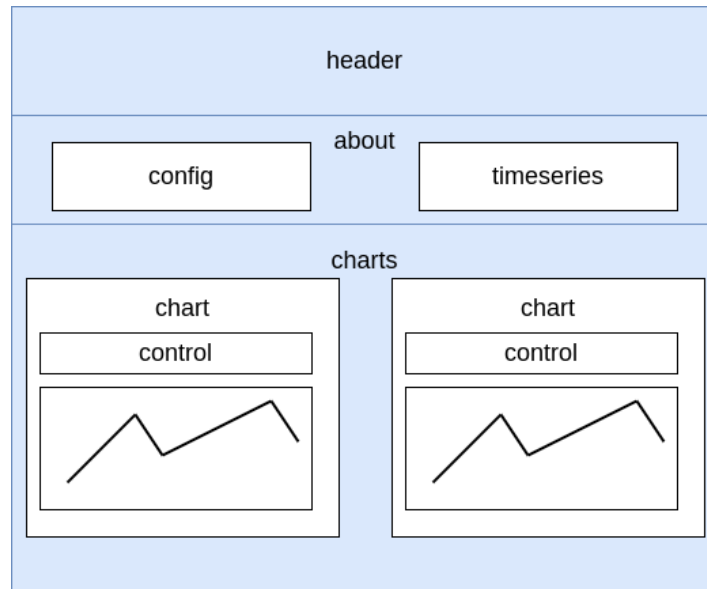


18. ábra: diagram területe (*chart*, kék színnel) és a diagramot körbeölelő terület (*container*, lila színnel) közti különbség

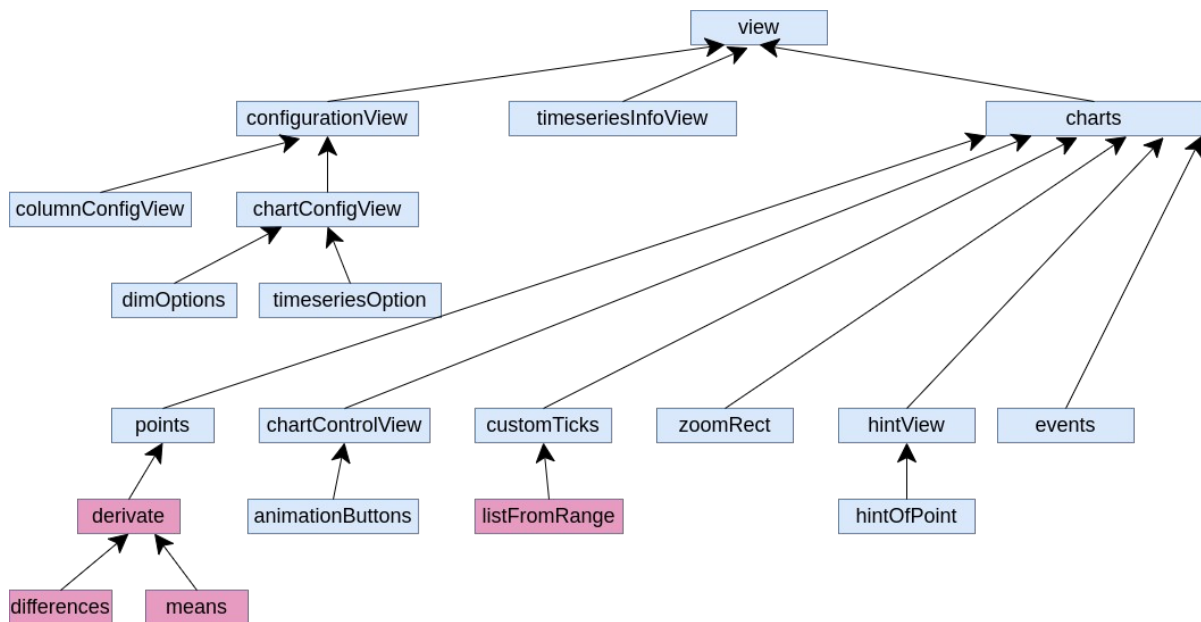
3.2.5.4 View

Az elm-kód által létrehozott html oldalon megjelenő azonosítókat (*id*) és osztályokat (*class*) a stílus beállításánál, a css-kódban használom fel. Az oldal három, egymás alatt elhelyezkedő nagy egységből áll, ezek elrendezését illusztrálja a 19. ábra.

- `header` azonosítójú `div`: Ez tartalmazza a weboldal címét (`Timeseries Visualization`).
- `about` azonosítójú `div`: Magába foglalja a konfigurációs részt (`config` azonosítóval), valamint a megjelenített idősorokat tartalmazó táblázatot (`timeseries` azonosítóval).
- `charts` azonosítójú `div`: Itt jelennek meg vonaldiagramok. Diagramonként egy `chart` osztályú `div` jön létre, ami tartalmazza a kezelőpanelt (`control` osztályú `div`), valamint magát a diagramot.



19. ábra: az adatmegjelenítő oldal fő részei



20. ábra: a megjelenítéshez használt függvények. A lilával jelzett függvényeket, mivel ezek csak áttételesen kapcsolódnak a megjelenítéshez, külön CALCULATIONS blokkba szerveztem.

A view függvény létrehozza a header azonosítójú div-et, valamint az about azonosítójú div-et. Utóbbiba beleágyazza a konfigurációs rész megjelenítését a configurationView függvénnyel és a megjelenített idősorokat összefoglaló táblázatot a timeseriesInfoView függvénnyel. A view függvényben állítom össze azoknak az idősoroknak a listját, amelyeknek szerepelniük kell az összefoglaló táblázatban. A timeseriesInfoView már

csak ezt a listát és a `model timeseriesInfo` mezőjét kapja meg paraméterként. Végül a `view` függvény meghívja a `charts` függvényt, amely a vonaldiagramok megjelenítését végzi.

3.2.5.4.1 Konfigurációs rész megjelenítése

A `configurationView` függvényben létrehozott `config` azonosítójú `div`-ben szerepel az egy-, illetve kétszlopos elrendezés kiválasztására szolgáló két gomb. Ezeket a `columnConfigView` hozza létre az alkalmazás állapotának (a `model columns` mezőjének) megfelelően `active`, illetve `inactive` címkékkel ellátva. Ezt a két gombot követik diagramonkénti konfigurációs lehetőségek, ahol a felhasználó azt választhatja ki, hogy mely pontok legyenek megjelenítve. Ezeket a gombokat, dimenzióválasztási lehetőségeket diagramonként a `chartConfigView` függvénnyel jelenítem meg. Még a `configurationView`-ban kiszámítom a megjeleníthető idősorok listáját (amely minden diagramra közös), valamint diagramonként a választható dimenziókat. Így a `chartConfigView`-nak már csak ezt a két listát, a diagram indexét és a diagram konfigurációját kell továbbítani. A konfigurációs részt az új diagramot létrehozó gomb zárja, ezt a `configurationView`-ban hozom létre.

A `chartConfigView` által létrehozott `chartconfig` osztályú `div` tartalmazza a megjelenítendő dimenziók kiválasztására szolgáló két `select`-et, valamint az idősor kiválasztására szolgáló `select`-et. Az előbbi két esetben a választási lehetőségeket a `dimOption` függvénnyel, míg az utóbbi esetben a `timeseriesOption` függvénnyel hozom létre. Ezekben a függvényekben állítom be a választási lehetőségekre vonatkozó `selected` attribútumot. Ennek az a jelentősége, hogy biztosan a ténylegesen kiválasztott (diagramonkénti konfigurációban szereplő) idősor/dimenzió neve jelenjen meg. Ellenkező esetben betöltéskor például a szerveren szereplő első idősor neve jelenne meg akkor is, ha a süti tartalma miatt nem ez került megjelenítésre. A `select`-eket követik az eredeti pontokat, illetve deriváltak megjelenítő gombok. A gombok aktivitását jelző osztály attribútumokat a `chartConfigView` függvényben számítom ki a diagramonkénti konfiguráció `derivated` mezője alapján. A `chartconfig` osztályú `div`-et a diagram törlésére szolgáló, `chartConfigView` függvényben létrehozott gomb zárja.

3.2.5.4.2 Összefoglaló táblázat megjelenítése

A konfigurációs rész mellett megjelenő összefoglaló táblázatot a `timeseriesInfoView` függvény hozza létre. Ez a függvény a `view`-tól a `model`-ben szereplő diagramonkénti konfigurációk listájából kinyert valódi (tehát nem üres) idősor-neveket kapja meg. A fejléc létrehozása után ezen a listán iterál végig, és minden elemhez létrehoz egy sort az idősor nevével és az idősor `info dictionary`-ből kinyert hosszával.

3.2.5.4.3 Vonaldiagramok megjelenítése

A `charts` függvényben kiszámítom a diagramok szélességét a `model width` és `columns` mezőjének függvényében. Az így kapott érték 0.9-szeresét alkalmazom, mivel a vonaldiagramok megjelenítéséhez használt csomag a szélesség kialakításánál nem veszi megfelelően figyelembe az idősor ábrán megjelenő nevének helyigényét. Ezután végighaladok a `model config` mezőjén, ami a diagramonkénti konfigurációk listáját tartalmazza, és minden konfigurációnak megfelelően kirajzolok egy diagramot. Ehhez a kirajzoláshoz használom a `chart` függvényt, amely a `model`-ben elmentett idősorokat (`timeseries`), egy diagram szélességét (`width`), az oszlopok számát `Bool`-értékként (`twoColumns`), a diagram indexét (`chartIdx`) és a diagram konfigurációját (`chartConfig`) kapja meg paraméterként. A `chart` függvényben a `chartConfig timeseriesName` mezője alapján kiválasztom a `timeseries` változóból a megjelenítendő idősort. Ebből a `points` függvénnyel állítom elő a megjelenítendő pontsorozatot. Az adatok deriválását, ha szükséges, a `points` függvény végzi. Szintén a `chart` függvényben állítom be a megjelenítés színét, az animáció szüneteltetésére szolgáló gomb feliratát az animáció állapota alapján, a vonaldiagram magasságát, valamint az elrendezésért felelős osztály attribútumot. Kétoszlopos esetben minden második diagramot `float_right` osztállyal látok el, a többit pedig `float_left` osztállyal. Ezek alapján fogja a `css`-kód létrehozni az elrendezést. Ezen értékek beállítása után a `chartControlView` függvénnyel kialakítom a diagramonkénti kezelőpanelt és a `terezka/line-charts` csomag `viewCustom` függvényével létrehozom a diagramot.

A `chartControlView` függvényben a `mouseMode` értéke alapján hozzárendelem a `Drag`, illetve `Zoom` feliratú gombokhoz az aktivitásukat jelző osztály attribútumot. Itt hozom létre a kezelőpanel első sorát (a nagyításra, húzogatásra vonatkozó gombokat), a második sorhoz pedig az `animationButtons` függvényt használom. Utóbbiban ha nincs aktív animáció (tehát az `animationType` értéke `None`), akkor csak a két, animáció indítására

szolgáló gombot hozom létre, mindkettőt inaktív állapotban (`inactive` osztállyal). Ha van aktív animáció, akkor az annak megfelelő indítógomb aktív, és megjelennek az animáció szüneteltetésére, illetve befejezésére szolgáló gombok. A szüneteltető gomb feliratát a `chart` függvényről kapom meg paraméterként.

A `chart` függvényben a vonaldiagram kirajzolásakor mindkét tengelyhez egyéni konfigurációt használok, itt állítom be a diagram méreteit, a tengelyintervallumokat és a diagramok beosztását a `customTicks` függvény segítségével. A `customTicks`-ben felhasznált `listFromRange` függvényben úgy választom ki a beosztások helyeit, hogy azok kerek értékekre essenek és számuk legalább 8 legyen. A vonaldiagram konfigurációjának `junk` mezőjében van lehetőség további vizualizációs elemek felvételére. Itt adom hozzá a diagramhoz a nagyításkor kijelölt téglalap kiszínezését (`zoomRect` függvény) és az egér koordinátáinak a megjelenítését (`hintView` függvény). Szintén a vonaldiagram konfigurációjában, az `events` mezőben rögzítem, hogy milyen események hatására milyen üzenettel frissüljön a `model`. Ennek a felsorolásához hoztam létre az `events` függvényt. Bizonyos eseményeket csak a diagram területén figyelek (`Move`, `LeaveChart`), míg a többit a diagramot körbeölelő területen is (`Hold`, `Drop`, `LeaveContainer`).

A `zoomRect` függvényben először eldöntöm, hogy egyáltalán kell-e kirajzolni téglalapot. Ha igen (tehát a `config mouseMode` mezőjének értéke `Zoom` és a `last` mezője nem `Nothing`), akkor a `last` pont és a `current` pont által kialakított téglalapot kiszínezem szürke színnel (`#b6b6b61a`).

A `hintView` függvény ennél összetettebb. Itt is azzal kezdek, hogy eldöntöm, szükség van-e az egér koordinátáinak megjelenítésére, tehát az egér a diagram területén található-e. Ezt a `config thereIsHint` mezője adja meg. Ha szükség van rá, akkor el kell döntenem, hogy az egér pontos koordinátáit jelenítsem meg, vagy a legközelebbi megjelenített pont koordinátáit. Ehhez kiszámítom, hogy milyen közel van az egér a legközelebbi ponthoz, és ha ez kisebb, mint az `x` tengelyintervallum 3%-a, akkor a megjelenített pont koordinátáit írom ki a diagram mellé kék színnel. Ezen kívül ki kell még számolnom a megjelenítés helyét a vonaldiagram koordinátarendszerében (`xLabel`, `yLabel` és `yLabelDown` változók) és a megjelenítendő szövegeket. Utóbbihoz a `hintOfPoint` függvényt használom, ahol a koordinátákat az olvashatóság kedvéért két tizedesjegyre kerekítem.

3.3 Stílus – css

A stílus kialakításánál az egységességre, átláthatóságra törekedtem. A címsor elválasztásához és a konfigurációs rész elkülönítéséhez ugyanazt a színt használtam, ami a vonaldiagramokban is szerepel (`rgb(89, 51, 204)`). Szintén ezzel a színnel kereteztem be az aktív (`active` osztályú) gombokat. A konfigurációs részekenél megjelenő vízszintes elrendezésekhez `display: flex;`-et használtam, míg a kétszlopos elrendezés esetén az oszlopok kialakításához `float: left;`-et, illetve `float: right;`-ot. Utóbbi azzal együtt működik, hogy a diagramok méretét hozzávetőleg az ablak felére állítottam az elm-kódban.

Különböző operációs rendszerek alatt, különböző böngészők esetén a `select`-ek alapértelmezett stílusa jelentősen eltér. Ezért ehelyett saját kinézetet alakítottam ki, ami minden esetben ugyanúgy jelenik meg.

3.4 Makefile

A fordításhoz Makefile-t hoztam létre. Ebben három `target` szerepel:

- `install`: Ez fordítja le az elm-kódokat javascript-kóddá. A fájlrendszert úgy alakítottam ki, hogy minden, a kliens használatához szükséges fájl a `dist` mappába kerüljön. Így a felhasználónak csak ehhez a mappához kell hozzáférnie. Ide kerültek a lefordított kódok, illetve ide másolja a Makefile a html oldalakat és a css fájlt.
- `clean`: Törli a `make install` parancs által létrehozott `dist` mappát.
- `test`: Lefuttatja az elm-kód tesztjeit.

4. Tesztelés

Az egységtesztekhez az `elm-test` csomagot használtam. Így kényelmesen írhatóak, futtathatóak a tesztek, de a programnak csak azon részére alkalmazható, ami nem függ az `elm`-en kívüli kódtól. Ezért szükséges a kézi tesztelés is.

4.1 Elm-test

Az `elm-test` npm csomagként telepíthető, ezért a teszteléshez szükséges az npm használata. A tesztelő csomagot az `npm install elm-test -g` paranccsal lehet telepíteni.

Egységtesztekkel ellenőriztem a számítások helyességét, az idősor pont-sorozattá való alakulását, az inicializációkat és az `update` hatását tipikus használati esetekben. Utóbbinál feltételeztem, hogy a megfelelő `update` funkciók meghívásra kerülnek (például az egér eseményei megérkeznek). Az alábbiakban felsorolom a tesztelt funkciókat. Tesztcsoportoknál zárójelben jelzem a kódban szereplő leírást (idézőjelek között) vagy a tesztfüggvény nevét.

- számítások (calculations)
 - deriválás ("Derivate")
 - beosztás előállítása ("List from range"): Ez a vonaldiagramok tengelybeosztásainak előállításánál kerül felhasználásra.
- inicializációk (inits)
 - vonaldiagramok konfigurációjának inicializálása ("Init chartconfig")
- `update` helyessége (updates)
 - idősorváltás valamely diagramnál ("New timeseries name")
 - új megjelenítendő dimenzió az x tengelyen
 - új diagram hozzáadása
 - diagram törlése
 - megjelenítés kétszlopossá tévése
 - nagyítás ("Zoom")

- kézi mozgatás ("Drag by hand")
- megjelenített tér visszaállítása ("Reset axis")
- váltás a derivált megjelenítésére: Annak az ellenőrzése, hogy a megjelenített tengelyintervallumok megfelelően változnak-e.
- váltás az eredeti pont-sorozatra a derivált megjelenítése után: Annak az ellenőrzése, hogy a megjelenített tengelyintervallumok megfelelően változnak-e.
- az eltelt másodpercek kezelése animáció esetén ("Tick")
- idősor pont-sorozattá alakítása (pointsInChart)
 - adatpont ponttá alakítása ("Data to point")
 - idősor pont-sorozattá alakítása ("Timeseries to list of point")

A tesztek az `apps/timeseries/priv/monitor` mappában állva az `elm-test` vagy a `make test` paranccsal futtathatóak, ennek eredményét mutatja be a 21. ábra. További tesztesetek adhatóak a meglevő `TimeseriesClientTest.elm`-hez vagy létrehozható új elm file a `apps/timeseries/priv/monitor/tests` mappában.

```
elm-test 0.19.1-revision2
-----

Running 44 tests. To reproduce these results, run: elm-test --fuzz 100
--seed 32343170475351 /home/tucsok/Documents/szakdolg/timeseries/priv
/monitor/tests/TimeseriesClientTest.elm

TEST RUN PASSED

Duration: 823 ms
Passed:   44
Failed:   0
```





21. ábra: tesztek futtatásának eredménye

Az `elm-test`-tel nem ellenőrizhetők a következő dolgok:

- a kliens megfelelően kommunikál-e a szerverrel?
- a javascript és elm kód között megfelelő-e az adatátvitel? Ide tartoznak az elm által kezelt események (gombnyomás, egérmozgatás) és a portokon keresztül történő kommunikáció is (süti mentése, betöltése).
- megfelelő és esztétikus-e a megjelenítés?



4.2 Kézi tesztelés

A kézi tesztelést az alábbi böngészőkben végeztem (a táblázatokban a logójukkal jelölöm őket):

- Ubuntu (18.04.1) operációs rendszer alatt:
 - Mozilla Firefox (75.0) 
 - Google Chrome (81.0.4044.122) 
- Microsoft Windows 10 Education (10.0.17.134) operációs rendszer alatt:
 - Mozilla Firefox (75.0 (64-bit)) 
 - Google Chrome (81.0.4044.122 (64-bit)) 





Teszteltem a megjelenítést nagy (1000 pontból álló) idősorok esetén is.

4.2.1 Konfiguráció

| Teszt eset | Elvárt viselkedés | Ubuntu | | Windows 10 | |
|-------------------------------------|--|---|---|---|---|
| | |  |  |  |  |
| Betöltés főoldalról érkezve | Egy diagram a kiválasztott idősorral | ✓ | ✓ | ✓ | ✓ |
| Betöltés közvetlenül | Egy diagram egy (a szerveren levő) idősorral | ✓ | ✓ | ✓ | ✓ |
| Egyoszlopos elrendezés | Diagramok egymás alatt | ✓ | ✓ | ✓ | ✓ |
| Kétoszlopos elrendezés | Diagramok két oszlopban, konfigurációk sorfolytonosan | ✓ | ✓ | ✓ | ✓ |
| Új x tengely – x tengelyintervallum | Legszűkebb, minden új pontot tartalmazó x tengelyintervallum | ✓ | ✓ | ✓ | ✓ |

| | | | | | |
|-------------------------------------|---|---|---|---|---|
| Új x tengely – y tengelyintervallum | Nem változik | ✓ | ✓ | ✓ | ✓ |
| Új y tengely – y tengelyintervallum | Legszűkebb, minden új pontot tartalmazó y tengelyintervallum | ✓ | ✓ | ✓ | ✓ |
| Új y tengely – x tengelyintervallum | Nem változik | ✓ | ✓ | ✓ | ✓ |
| Új idősor – azonos dimenziók | Megjelenítendő dimenziók megmaradnak | ✓ | ✓ | ✓ | ✓ |
| Új idősor – más dimenziók | Új megjelenítendő dimenziók | ✓ | ✓ | ✓ | ✓ |
| Eredeti pontok deriválás után | Lehető legszűkebb, minden pontot tartalmazó y tengelyintervallum, x tengelyintervallum nem változik | ✓ | ✓ | ✓ | ✓ |
| Derivált pontok | Új y tengelyintervallum, x tengelyintervallum nem változik | ✓ | ✓ | ✓ | ✓ |
| Diagram törlése | | ✓ | ✓ | ✓ | ✓ |
| Diagram hozzáadása | Eddigi utolsóval megegyező konfiguráció | ✓ | ✓ | ✓ | ✓ |

4.2.2 Diagramonkénti konfiguráció





| Teszteset | Elvárt viselkedés | Ubuntu | | Windows 10 | |
|---|--|---|---|---|---|
| | |  |  |  |  |
| Nagyítás gombbal | | ✓ | ✓ | ✓ | ✓ |
| Kicsinyítés gombbal | | ✓ | ✓ | ✓ | ✓ |
| Kicsinyítés, majd nagyítás gombbal | Kiegyenlítik egymást (eredeti tengelyintervallumok) | ✓ | ✓ | ✓ | ✓ |
| Tengelyek visszaállítása – nagyítás után | Lehető legszűkebb, minden pontot tartalmazó tengelyintervallumok | ✓ | ✓ | ✓ | ✓ |
| Tengelyek visszaállítása – húzogatás után | Lehető legszűkebb, minden pontot tartalmazó tengelyintervallumok | ✓ | ✓ | ✓ | ✓ |
| húzogatás - gomb | Drag gomb kék keretben | ✓ | ✓ | ✓ | ✓ |
| húzogatás | A "megfogott" pont nem változik | ✓ | ✓ | ✓ | ✓ |

| | | | | | |
|---|---|---|---|---|---|
| Kézi nagyítás - gomb | Zoom gomb kék keretben | ✓ | ✓ | ✓ | ✓ |
| Kézi nagyítás – téglalap | Szürke téglalap jelzi a kijelölt területet | ✓ | ✓ | ✓ | ✓ |
| Kézi nagyítás – kilógva a diagramból | A szürke téglalap kilóg a diagram területéről, nagyítás a kilógó területre is | ✗ | ✗ | ✗ | ✗ |
| Kézi nagyítás | Kijelölt terület kinagyítva | ✓ | ✓ | ✓ | ✓ |
| Kézi nagyítás – kicsi terület kijelölve | Nincs nagyítás | ✓ | ✓ | ✓ | ✓ |
| Nagyítás gombbal - nagy idősor | Nem lassul le. | ✓ | ✓ | ✓ | ✓ |
| Nagyítás kézzel – nagy idősor | Nem lassul le. | ✓ | ✓ | ✓ | ✓ |
| húzogatas – nagy idősor | Nem akad. | ✓ | ✓ | ✓ | ✓ |

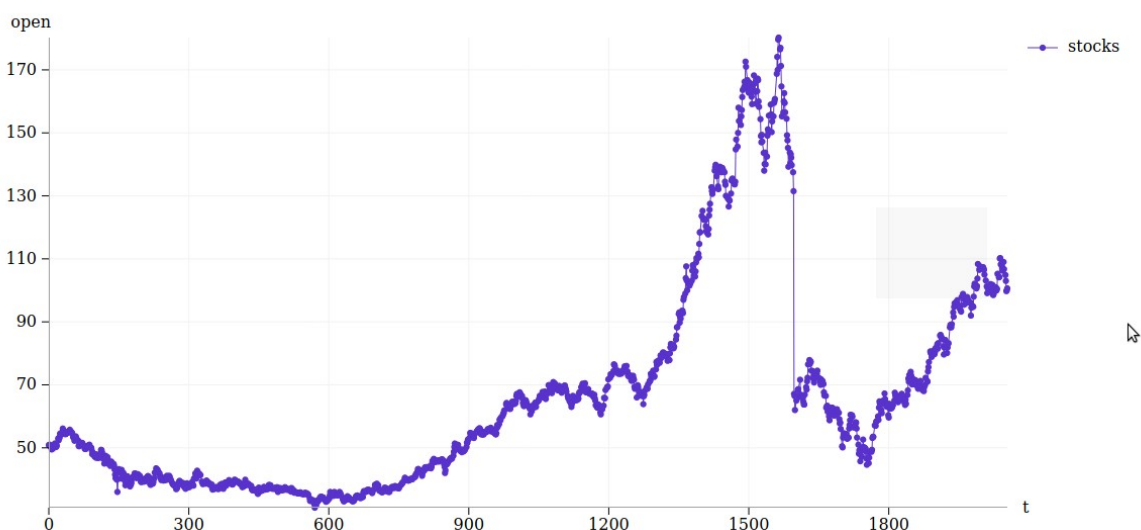
4.2.3 Animáció

| Tesztet | Elvart viselkedés | Ubuntu | | Windows 10 | |
|-------------------------------|---|---|---|---|---|
| | |  |  |  |  |
| Nincs animáció | Második sorban két inaktív gomb | ✓ | ✓ | ✓ | ✓ |
| Pontonként | Egyenlő időközönként megjelennek a pontok | ✓ | ✓ | ✓ | ✓ |
| Futóablakkal | Futóablak végighalad a teljes idősoron | ✓ | ✓ | ✓ | ✓ |
| Szüneteltetés | | ✓ | ✓ | ✓ | ✓ |
| Szüneteltetés, majd folytatás | | ✓ | ✓ | ✓ | ✓ |
| Újrakezdés | Animáció végén van rá lehetőség | ✓ | ✓ | ✓ | ✓ |
| Animáció befejezése | Végig van rá lehetőség | ✓ | ✓ | ✓ | ✓ |

4.2.4 Összefoglaló táblázat

| Teszteset | Elvárt viselkedés | Ubuntu | | Windows 10 | |
|---------------|---|---|---|---|---|
| | |  |  |  |  |
| Üres szerver | Üres táblázat | ✓ | ✓ | ✓ | ✓ |
| Nincs diagram | Üres táblázat | ✓ | ✓ | ✓ | ✓ |
| Van diagram | Csak a diagramokon megjelenített idősorok szerepelnek benne | ✓ | ✓ | ✓ | ✓ |

A hibás tesztetnél a nagyítani kívánt téglalap a 22. ábrán látható módon kinyúlik a vonaldiagram területéről. Ilyenkor eddig nem megjelenített területnek is elő kell kerülnie. A hiba az, hogy a szürke területkijelölő téglalap nem követi az egeret a diagramon kívülre. Ennek oka, hogy az egér mozgatása nincs kezelve a diagramon kívül (de az egérgombok eseményei igen). A nagyítás már helyesen, az egér által kijelölt területre történik.



22. ábra: diagram területéről kilógó nagyítás

5. Összefoglalás

Szakdolgozatommal egy olyan, idősorok vizualizációjára szolgáló webalkalmazást szerettem volna létrehozni, amely megfelelően általános ahhoz, hogy sokfajta idősor esetén használható legyen, ugyanakkor a felhasználói felülete könnyen átlátható. Ezt egyrészt jól átgondolt előretervezéssel sikerült elérnem, másrészt a folyamatos tesztelés közben felmerült új lehetőségekhez, problémákhoz való alkalmazkodással. Megtapasztalhattam, hogy mennyire fontos a tervezés, hogy az ebbe fektetett energia többszörösen megtérül.

A tervezés másik fontos kérdése volt a felhasznált eszközök kiválasztása. Ennek is megtapasztaltam a jelentőségét, hiszen a fejlesztés folyamán végig élvezzük a gyümölcsét egy jó választásnak és küzdünk egy rossz miatt. Úgy érzem, sikerült jó döntéseket hoznom mind az elm nyelv, mind vonaldiagramok megjelenítésére használt csomag kiválasztásakor. Számomra az utóbbi döntés izgalmasabb volt, mivel úgy kellett eszközök között döntenem, hogy egyiket sem használtam korábban. Tapasztalatot szerezhettem abban, hogy ilyen helyzetekben milyen szempontok szerint, milyen technikával, mennyi időbefektetéssel érdemes döntést hozni.

Szintén sok tapasztalatot szereztem a csapatban történő munkával kapcsolatban. A szakdolgozat tervezési-, ötletelési fázisánál tapasztaltam, mennyire motiváló és előrevívő lehet a közös gondolkodás. A fejlesztés folyamán pedig gyakorlatot szerezhettem a csapatmunka eszközeiben (például git) és a szakmai kommunikációban is.

6. Köszönetnyilvánítás

Ezúton köszönöm Pataki Norbert tanár úrnak a témavezetést, támogatását és a szakdolgozat lektorálásában nyújtott segítségét.

Köszönettel tartozom társamnak, Olivérnek, inspiráló gondolataiért és az együttműködésben szerzett tapasztalatokért.

7. Hivatkozások

[1] elm 0.19.1 Linux operációs rendszerre:

<https://github.com/elm/compiler/blob/master/installers/linux/README.md> (elérés: 2020. április 30.)

[2] elm 0.19.1 Windows operációs rendszerre:

<https://github.com/elm/compiler/releases/download/0.19.1/installer-for-windows.exe> (elérés: 2020. április 30.)

[3] Bevezetés az elm nyelv használatába: <https://guide.elm-lang.org/> (elérés: 2020. május 1.)

[4] elm kódolási konvenciók: <https://elm-lang.org/docs/style-guide> (elérés: 2020. május 1.)

[5] terezka/line-charts: <https://package.elm-lang.org/packages/terezka/line-charts/latest> (elérés: 2020. május 1.)