



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.....	IV
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 TODO idősor hozzáadása.....	VI
2.2.2 Kliens.....	VI
2.3 Használat.....	VI
2.3.1 Főoldal (index.html).....	VI
2.3.2 Adatmegjelenítő oldal (TimeseriesClient.html).....	VII
2.3.2.1 Konfiguráció.....	VIII
2.3.2.2 Idősorok metaadatai.....	IX
2.3.2.3 Vonaldiagramok tulajdonságai.....	IX
2.3.2.4 Vonaldiagramok kezelése.....	X
2.3.2.5 Animációk.....	X
3. Fejlesztői dokumentáció.....	XII
3.1 Használt fogalmak.....	XII
3.2 Funkcionalitás – elm.....	XII
3.2.1 Az elm nyelv.....	XII
3.2.2 Kommunikáció javascript-kód és elm-kód között (portok).....	XIII
3.2.3 Kommunikáció a szerverrel.....	XIV
3.2.4 Főoldal (Index.elm).....	XV
3.2.5 Adatmegjelenítő oldal (TimeseriesClient.elm).....	XV
3.2.5.1 Típusok.....	XV
3.2.5.2 Inicializálás.....	XVIII
3.2.5.3 Update.....	XIX

3.2.5.3.1 Szerverről érkezett adat kezelése.....	XIX
3.2.5.3.2 Külső események (idő, képernyőméret, süti).....	XX
3.2.5.3.3 Konfiguráció frissítése.....	XX
3.2.5.3.4 Diagramonkénti konfiguráció frissítése.....	XXI
3.2.5.3.5 Diagramok egéreseeményeinek kezelése.....	XXII
3.2.5.4 View.....	XXIII
3.2.5.4.1 Konfigurációs rész megjelenítése.....	XXV
3.2.5.4.2 Összefoglaló táblázat megjelenítése.....	XXV
3.2.5.4.3 Vonaldiagramok megjelenítése.....	XXV
3.3 Stílus - css.....	XXV
4. Tesztelés.....	XXVI
4.1 Elm-test.....	XXVI
4.2 Kézi tesztelés.....	XXVIII
4.2.1 Konfiguráció.....	XXVIII
4.2.2 Diagramonkénti konfiguráció.....	XXIX
4.2.3 Animáció.....	XXX
4.2.4 Összefoglaló táblázat.....	XXX
5. Hivatkozások.....	XXXII

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 deriválnak 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)
- rebar3 (3.13.1)

2.1.2 Kliens

A kliens telepítéséhez szükséges:

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

A klienst a parancssorban a `apps/timeseries/priv/monitor` mappában állva a `make` paranccsal lehet telepíteni. A telepítés folyamán generált fájlokat, mappákat ugyanitt a `make clean` paranccsal lehet törölni.

TODO ábra `make` parancs futtatásának eredménye

TODO 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 TODO idősor hozzáadása

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

```
rebar3 shell
```

A szerver a parancssorban állva a `Ctrl + c` 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. Ezen információk alapján választhatunk, hogy melyiket szeretnénk vizsgálni. Az adatmegjelenítő oldalon lesz lehetőségünk más idősorra váltani vagy egyszerre több idősort megjeleníteni.

Timeseries Visualization

Choose a timeseries!

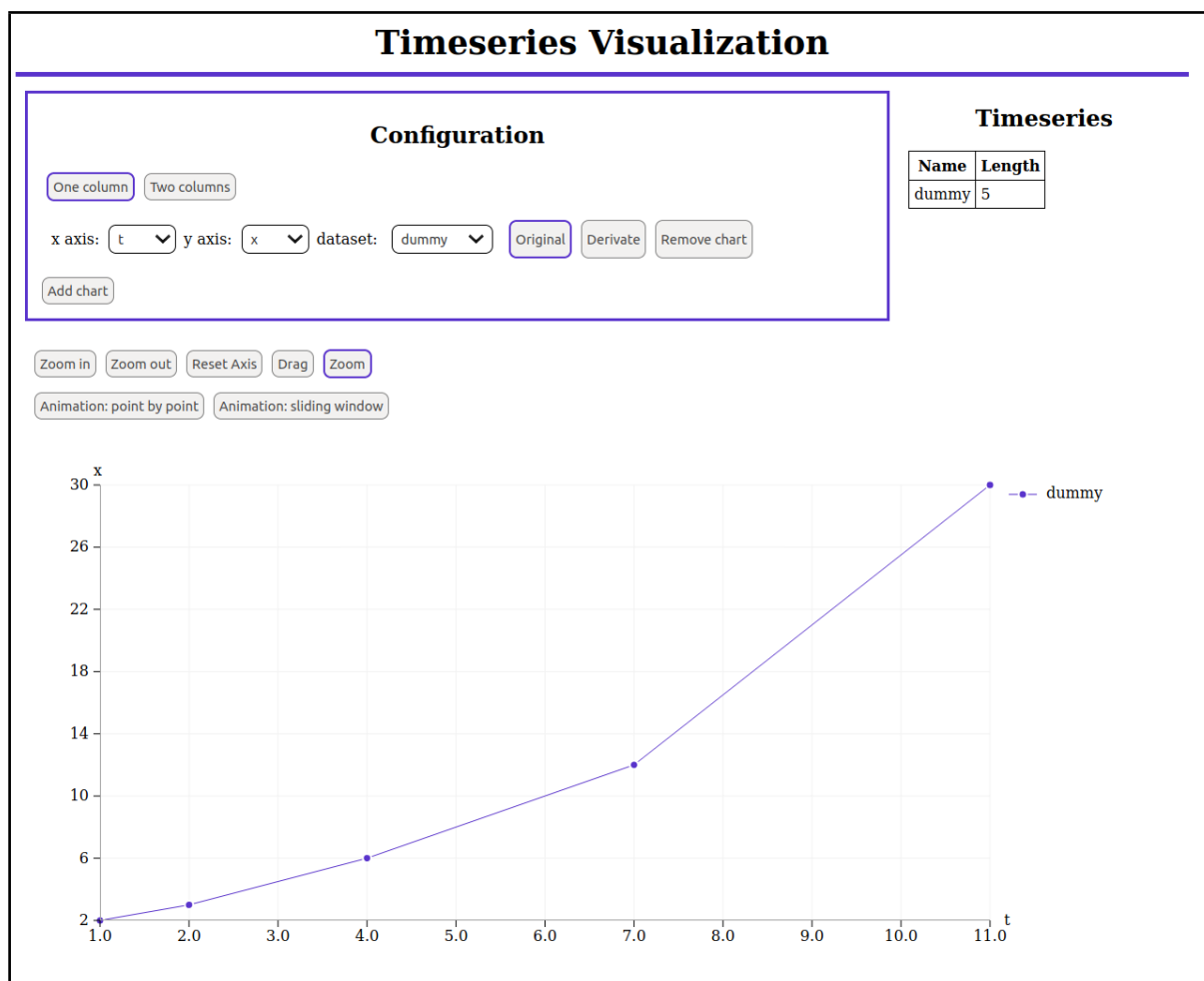
Name	Length
<input type="button" value="dummy"/>	5
<input type="button" value="dummy2"/>	100

1. 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. Betöltéskor az eredeti adatok kerülnek megjelenítésre, később át lehet váltani a deriváltra.



2. 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 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 serveren szereplő összes idősor közül. Ha olyan idősorra váltunk, amelyben megtalálhatóak az eddigi megjelenített dimenziók, akkor ezek beállítása megmarad, míg új dimenziók esetén az első kettő kerül kiválasztásra.

TODO 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 világoskéssel. Deriváláskor az x tengely nem változik, az y tengely pedig igazodik a derivált értékekhez.

TODO 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 eddigi utolsó diagraméval fog megegyezni. Ha nincs előző diagram, akkor a serveren megtalálható első idősorról 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 serveren, mint a főoldalon. Minden idősor csak egyszer szerepel, akkor is, ha több vonaldiagram is használja az adatait. Az 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. Ez eltűnik, ha a kurzor elhagyja a diagram területét.

TODO ábra – kurzor koordinátáinak megjelenítése

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.

TODO 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. A lehetőségeken balról jobbra haladok végig.

TODO ábra – diagramonként kezelőpanel első sora

`Zoom in / Zoom out`: Lehet gombnyomással nagyítani, illetve kicsinyíteni a vonaldiagramban. Ezek mértéke olyan, hogy pont 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 megjelenítettekre.

`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.

TODO ábra - kézi nagyításkor megjelenő szürke területkijelölő téglalap

2.3.2.5 Animációk

Az animációk kezelésére a diagramonkénti kezelőpanel második sora szolgál. Ha éppen nincs aktív animáció, akkor ez csak két gombból áll, amelyek az animációk elindítására szolgálnak.

TODO ábra – diagramonként kezelőpanel második sora animáció nélkül

TODO 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 azonos időközönként jelennek meg.

Animation: sliding window: Egyszerre az idősor egy fix hosszú szakasza jelenik meg, ez azonos időközönként "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 felhasználói 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 gyakran használ `Maybe <XY>` típust, amit explicit típuskonverzióval kell `<XY>` típusá 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.

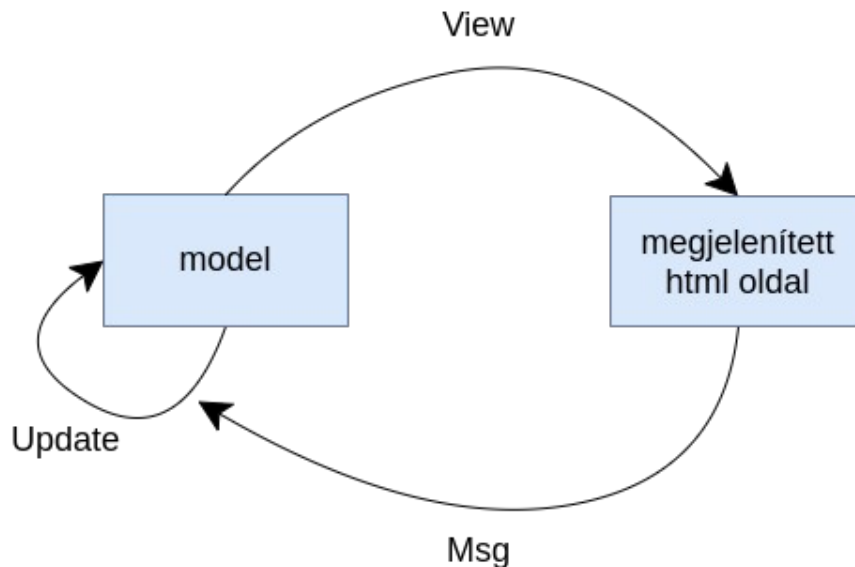
3.2 Funkcionalitás – elm

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 ki, én minden függvényhez kiírtam a típus annotálást, ezzel is segítve a hibaüzenetek értelmezését.

Egy elm program három nagy egységből áll:

- `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.



x. ábra: az elm alkalmazás felépítése

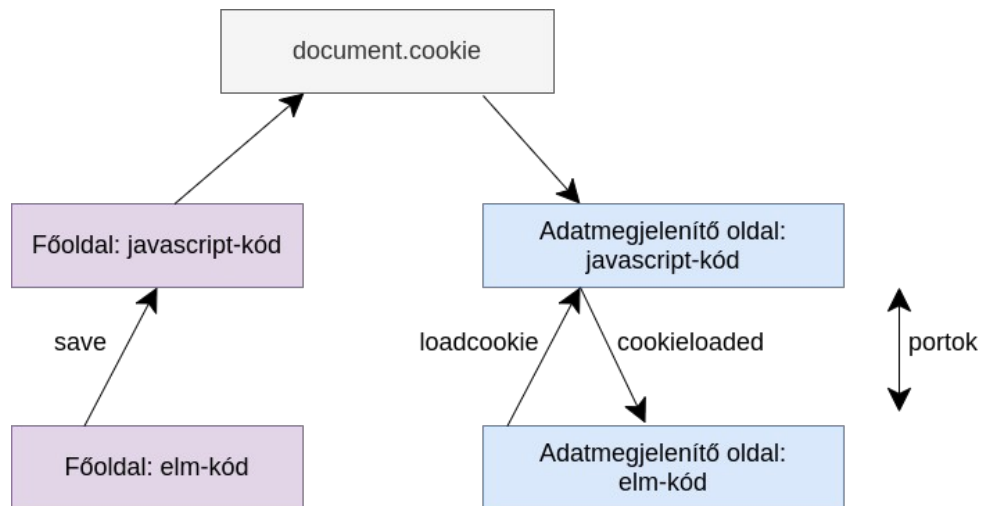
TODO – ábraszám

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.

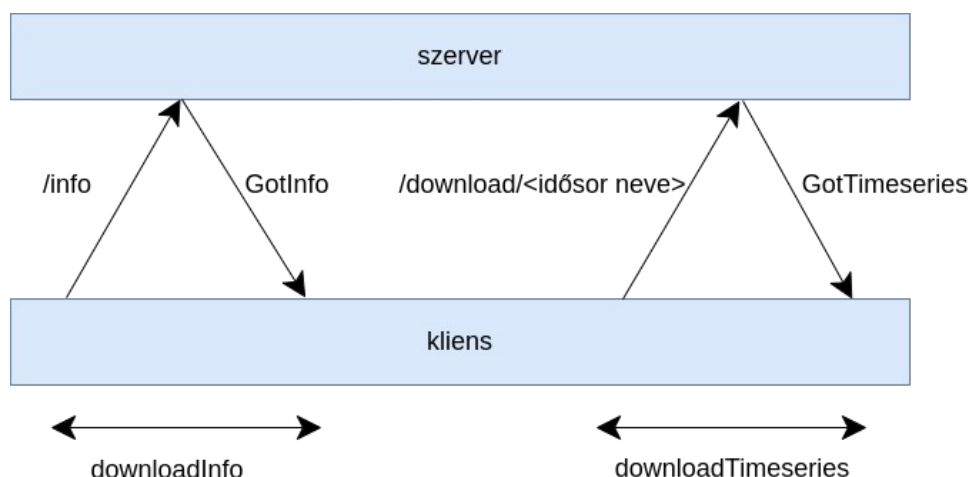


x. ábra: süti használata

TODO ábraszám

3.2.3 Kommunikáció a szerverrel

A szerverrel való kommunikáció Http lekérdezéseken keresztül történik. Az idősorok neveinek és hosszainak letöltését a `downloadInfo` végzi. A lekérdezés az `"/info"` 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.



x.ábra: szerver-kliens kommunikáció

TODO ábraszám

3.2.4 Főoldal (Index.elm)

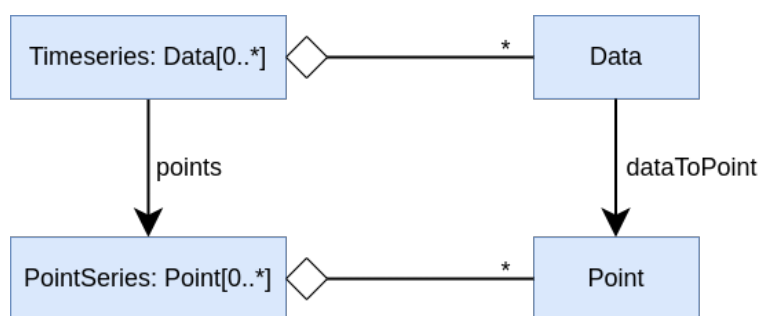
A főoldal szerkezete jóval egyszerűbb, mint az adatmegjelenítőé. Annyi a feladata, hogy felsorolja a szerveren található idősorokat és át lehessen navigálni róla az adatmegjelenítő oldalra (a megfelelő süti elmentése után). A modelben csak a `timeseriesInfo` szerepel, ami a szerveren található idősorok neveit és hosszait tartalmazza. Inicializáláskor ez üres és 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. 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). Itt a

szervertől érkező idősoroktól elvárható, hogy minden adatpontjában ugyanazok a dimenziók szerepeljenek. 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.



x. ábra: adattípusok és átalakításuk

TODO ábraszám

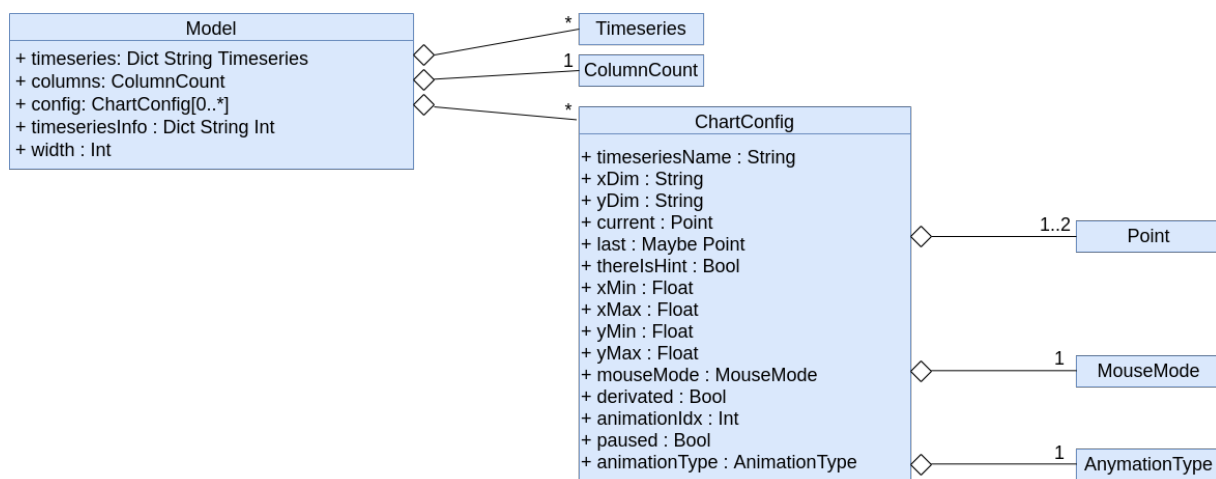
Az alkalmazás állapotát tároló `model` és annak építőelemeinek reprezentálásához `type alias`-okat (`Model`, `ChartConfig`) és `record`-okat (`ColumnCount`, `MouseMode`, `AnimationType`) készítettem. A `record`-ok előnye, hogy `case`-elágazások esetén egyértelműen lefedhető minden lehetséges értékük, 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 (diagramon 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.
- `mouseMode`: Megadja, hogy az egérrel nagyítani vagy húzogatni lehet-e.
- `derivated`: Megadja, hogy az eredeti pontokat vagy azok 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`.



x. ábra: az alkalmazás állapotának reprezentálása

TODO ábraszám

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 függvény 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 ablak szélességre 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 szerveren 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 az 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 időrnak megfelelő kezdőkonfigurációra.

`GotTimeseries` üzenet akkor érkezik, mikor egy idősor letöltése fejeződött be. A kapott karaktersorozatot itt alakítom idősorra 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) frissíték 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 elindítom a szerverről való letöltését (`downloadTimeseries`).

`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). Ezért 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 kétszlopos elrendezés közötti váltásnál (`OneColumn` és `TwoColumns` üzenetek) csupán a `model columns` mezőjét kell frissíteni. 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 pont egymás fordítottja.

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

Az eredeti tengelyintervallumok visszaállításakor (`ResetAxis`) kiszámítom a megjelenítendő pont-sorozatot (`currentPoints`) és ez 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 egérgombját. Lenyomva tartáskor nem érkezik újabb `Hold` üzenet. Ekkor a kézi vezérlés mindkét állapotában (nagyítás vagy húzogatás) a diagramonkénti konfiguráció `last` mezőjét állítom be az adott pontra.

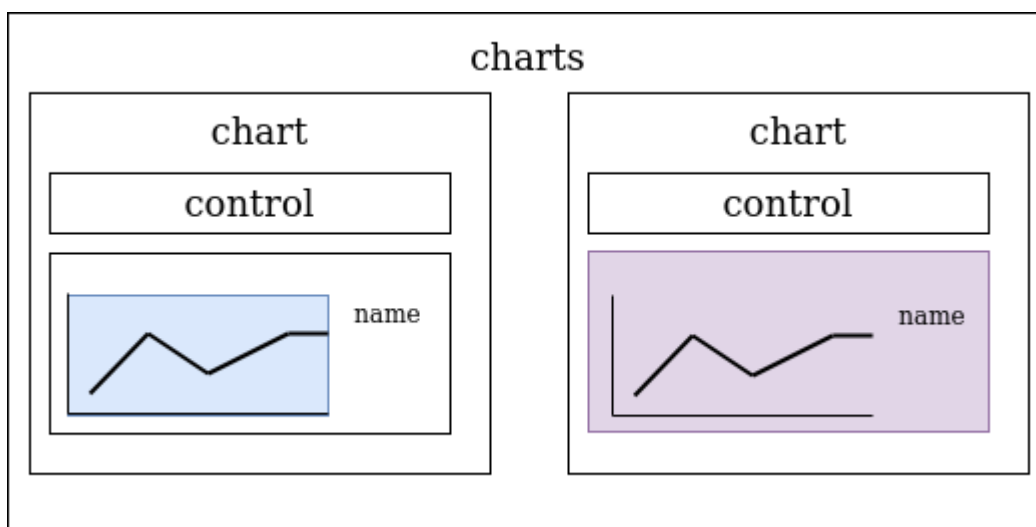
`Move` üzenet érkezik, ha a felhasználó elmozdítja az egeret. Ekkor nagyítás esetén, akár le van nyomva az egérgomb, akár nem, a `current` mezőt kell az adott pontra frissíteni, a `thereIsHint`-et pedig `True`-ra állítani. Ezek alapján fognak megjelenni a vonaldiagram mellett az egér koordinátái. Nagyítás 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 húzogatás 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 húzogatás 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.



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

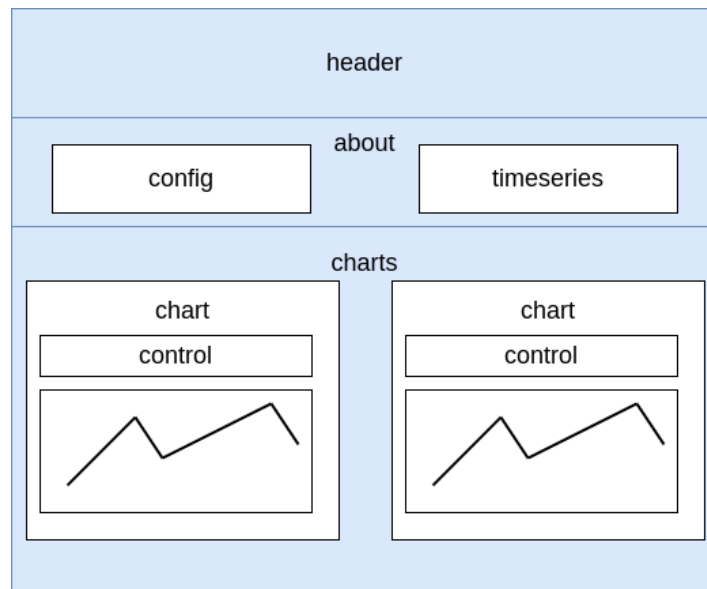
TODO ábraszám

3.2.5.4 View

Az elm-kód által létrehozott html oldalban megjelenő azonosítókat (`id`) és osztályokat (`class`) a stílus beállításánál, a css-kódban használom fel. Az oldal három, egymás alatt elhelyezkedő nagy egységből áll.

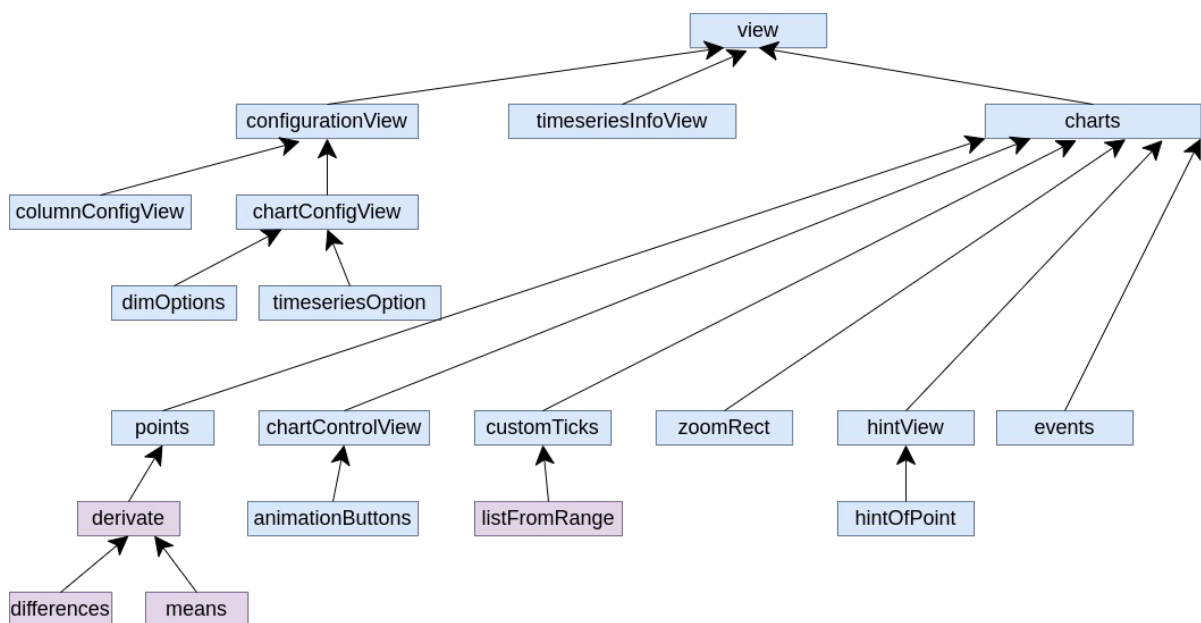
- `header` azonosítójú `div`: Ez tartalmazza a címet.

- `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.



x. ábra: az adatmegjelenítő oldal fő részei

TODO ábraszám



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

TODO ábraszám

3.2.5.4.1 Konfigurációs rész megjelenítése

3.2.5.4.2 Összefoglaló táblázat megjelenítése

3.2.5.4.3 Vonaldiagramok megjelenítése

3.3 Stílus - css

4. Tesztelés

Az egységtesztekhez az `elm-test` csomagot használtam. Ezzel 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

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` paranccsal futtathatóak. 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/tucskok/Documents/szakedolg/timeseries/priv
/monitor/tests/TimeseriesClientTest.elm

TEST RUN PASSED

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

x. ábra: tesztek futtatásának eredménye

TODO ábra szám





Az `elm-test`-tel nem ellenőrizhetőek 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.	✓	✓	✓	✓
húzogató – nagy idősor	Nem akad.	✓	✓	✓	✓

4.2.3 Animáció

Teszteset	Elvárt 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 tesztesetnél a nagyítani kívánt téglalap kinyúlik a vonaldiagram területéről. Ilyenkor eddig nem megjelenített területnek is elő kell kerülnie. A hiba az, hogy 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.

TODO ábra: diagram területéről kilógó nagyítás

5. 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.)