FELADATKIÍRÁS

A feladat egy olyan full-stack alkalmazás megvalósítása, amelyben a felhasználónak lehetősége van egy tőzsdei kereskedő robot használatára. Az alkalmazás webes felületén nyomon tudja követni a robot tevékenységét, valamint annak paramétereit szabadon állíthatja. Ezekkel befolyásolhatja, hogy például mekkora kockázatot szeretne vállalni, vagy éppen milyen devizákkal, CFD-kkel kíván kereskedni.

Az alkalmazás tartalmazni fog egy beépített historikus tesztert, amelyen minden beállítással ki lehet próbálni, hogy az algoritmus milyen eredményeket ért volna el egy adott időszakban a múltban.

A hallgató feladatának a következőkre kell kiterjednie:

* Mutassa be az algoritmikus tőzsdei kereskedés alapjait, annak buktatóival és nehézségeivel együtt!
* Elemezze a tőzsdei folyamatokat algoritmizálási szempontból!
* Készítsen egy back-end szoftvert, amely futtatni fogja az algoritmust és a tesztkörnyezetet, valamint API-t szolgáltat, amelyen keresztül beállítható és megfigyelhető a robot működése!
* Készítsen egy reszponzív webes front-end felületet, amely a back-en API-ját használva kényelmes felhasználói élményt nyújt!
* Tesztelje az alkalmazást!



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Boros Bence

Tőzsdei kereskedést segítő webalkalmazás fejlesztése

Kövesdán Gábor

BUDAPEST, 2020

Tartalomjegyzék

[Összefoglaló 6](#_Toc58277719)

[Abstract 7](#_Toc58277720)

[1 Bevezetés 8](#_Toc58277721)

[1.1 Motiváció 8](#_Toc58277722)

[1.2 Tőzsdei kereskedés 8](#_Toc58277723)

[1.2.1 A tőzsde múltja 8](#_Toc58277724)

[1.2.2 A devizakereskedés 10](#_Toc58277725)

[1.2.3 Magánemberként a devizapiacon 10](#_Toc58277726)

[1.3 Webes technológiák 11](#_Toc58277727)

[1.3.1 Történelem 11](#_Toc58277728)

[1.3.2 Modern webalkalmazások 11](#_Toc58277729)

[2 Felhasznált technológiák 13](#_Toc58277730)

[2.1 Algoritmus 13](#_Toc58277731)

[2.1.1 MetaQuotes Language 4 (MQL4) 13](#_Toc58277732)

[2.2 Backend 13](#_Toc58277733)

[2.2.1 MongoDB 13](#_Toc58277734)

[2.2.2 Node.js 14](#_Toc58277735)

[2.2.3 Express.js 15](#_Toc58277736)

[2.2.4 Passport.js 15](#_Toc58277737)

[2.2.5 Oanda API / simple-fxtrade 15](#_Toc58277738)

[2.3 Frontend 16](#_Toc58277739)

[2.3.1 React 16](#_Toc58277740)

[2.3.2 React Bootstrap 17](#_Toc58277741)

[3 Tervezés 18](#_Toc58277742)

[3.1 Algoritmus 18](#_Toc58277743)

[3.2 Backend 18](#_Toc58277744)

[3.2.1 Komponens diagram 18](#_Toc58277745)

[3.2.2 API végpontok 18](#_Toc58277746)

[3.3 Frontend 18](#_Toc58277747)

[4 Implementáció 19](#_Toc58277748)

[4.1 Algoritmus 19](#_Toc58277749)

[4.2 Backend 19](#_Toc58277750)

[4.3 Frontend 19](#_Toc58277751)

[5 Tesztelés 20](#_Toc58277752)

[5.1 Algoritmus 20](#_Toc58277753)

[5.2 Backend 20](#_Toc58277754)

[5.3 Frontend 20](#_Toc58277755)

[6 Összegzés 21](#_Toc58277756)

[7 Irodalomjegyzék 22](#_Toc58277757)

[Függelék 23](#_Toc58277758)

Hallgatói nyilatkozat

Alulírott **Boros Bence**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2020. 12. 03.

...…………………………………………….

Boros Bence

Összefoglaló

A szakdolgozatom témájaként mindenképpen szerettem volna valami olyat választani, ami nem egy átlagos, könnyen letudható projekt, hanem beleesik az érdeklődési körömbe és még gyakorlati haszna is lehet. Édesapám révén már kezdetleges ismeretekkel rendelkeztem a devizakereskedés terén, és úgy gondoltam, hogy ez egy kellően sok irányból megközelíthető téma ahhoz, hogy ezen dolgozatom alapjául szolgáljon.

A manuális devizakereskedés egy sok tanulást és tapasztalatot igénylő tevékenység, így nekem programozóként magától értetődő volt a gondolat, hogy célszerű lenne azt automatizálni. Ehhez kell egy algoritmus, amit múltbéli adatok alapján, gyakorlatilag mintafelismerésre optimalizálok. A kényelmes használathoz szükség van egy grafikus felületre is. A modern kor követelményeihez igazodva ezt egy reszponzív webalkalmazás formájában valósítom meg.

TODO – Megoldás

TODO - Konklúzió

Abstract

# Bevezetés

## Motiváció

A szakdolgozatom témájaként mindenképpen szerettem volna valami olyat választani, ami nem egy átlagos, könnyen letudható projekt, hanem beleesik az érdeklődési körömbe és még gyakorlati haszna is lehet. Édesapám révén már kezdetleges ismeretekkel rendelkeztem a devizakereskedés terén, és úgy gondoltam, hogy ez egy kellően sok irányból megközelíthető téma ahhoz, hogy ezen dolgozatom alapjául szolgáljon.

## Tőzsdei kereskedés

A tőzsde olyan intézményként született, amelynek helyzetét ugyanúgy, mint egy szabályalkotási, közigazgatási és bíráskodási joggal felruházott testületét, nem egy törvény, hanem a fejlődés különböző időpontjaiban alkotott különböző jogszabályok, illetve a szokásjog szabályozta. A tőzsde a piacok piaca. Központosítja a kereskedelmet nem csak egy ország, hanem az egész világpiac számára. A modern gazdaságban ugyanis a kereslet és a kínálat központosítása létszükséglet. [1]

### A tőzsde múltja

Maga az elnevezés az 1300-as ével elején Brüggében élő van der Burse patríciuscsalád nevéből ered. A Burse-k fogadót tartottak fen Brügge központjában, ahol az itáliai kereskedők találkoztak északi üzletfeleikkel, és ide telepítették váltóikat. Brügge szerepét 1460-tól Atwerpen vette át, itt a XVI. Század elején már nemzetközi árutőzsde működött. Érdekesség, hogy a tőzsdét a szeszélyessége miatt nőnemű szóval jelölik több idegen nyelvben is (pl. Die Börse).

A tőzsde a vásárból alakult ki. A különbség persze nagy: a vásárokon mindig jelen volt az áru, a tőzsdén soha. A vásárból tőzsde akkor lett, amikor az üzletkötések tömegessé váltak, és mind az áru, mind a fizetési eszköz oldalán a helyettesíthetőség vált uralkodóvá. Az első tőzsdét Antwerpenben hozták létre. Itt történt meg először, hogy különböző nemzetiségű kereskedők jöttek össze rendszeresen üzleteket kötni. Addig ugyanis a különböző fejedelmi privilégiumokra vigyázó és mindenkire féltékeny más–más nemzetiségű kereskedőket mesterségesen elválasztották egymástól.

Az amszterdami tőzsde a Holland Kelet–Indiai és a Nyugat–Indiai Társaság részvényeinek adásvételével vált a legjelentősebbé. Rövid időn belül még az antwerpeni tőzsde forgalmát is túlszárnyalta. A 17. század folyamán itt már lényegében kialakult a modern tőzsdei üzlet technikája. Ekkor vált ketté az áru– és az értéktőzsde.

Az 1602-ben alapított Holland Kelet-Indiai Társaság először 1605-ben fizetett osztalékot, a 15%-os osztalékfizetés azonban nem készpénzben, hanem borsban történt. A társaság részvényeit kezdetben az utcán és kávéházakban árusították, de hamarosan az üzletkötés az 1611-ben alapított amszterdami tőzsdére tevődött át. A részvények kezdetben névre szóltak, tulajdonosuk felelőssége korlátlan volt. 1672-ben a holland-francia háború idején az Egyesült Tartományok dobtak piacra hosszú lejáratú, egységes címletű adósságleveleket, és a példát még ugyanabban az évben követte az angol, majd hét év múlva a francia kincstár.

A részvényforgalom a múlt század közepén az első vasúttársaságok megjelenésével indult gyors növekedésnek. A párizsi tőzsdén 1800-ban még csak hét, 1869-ben már négyszáznál több értékpapírt, zömmel részvényt jegyeztek.

A részvénytársaságok alapítását is kontinentális joggyakorlat (először Napóleon 1808-as kereskedelmi törvénye, majd az 1843-as, az előbbinél szigorúbb porosz részvénytársasági törvény) állami engedélyhez kötötte, szigorú felügyeletet írt elő, ugyanakkor megengedte a bemutatóra szóló részvények kibocsátását. Angliában – és az Egyesült Államokban – a társasággá alakulás soha nem volt állami engedélyhez kötve, de itt a részvények a múlt század végéig szigorúan névre szóltak.

Minden elővigyázatosság ellenére a gazdasági–pénzügyi válságok és a túlzott spekuláció időnként így is tőzsdekrachokat idézett elő. Az első tőzsdeválság 1557-ben alakult ki, amikor is az Amerikából behozott aranyra túlzott spekulációt folytattak. Az 1873-as ún. Gründerkrachot a porosz–francia háború utáni nagy alapítási láz okozta. Ebből a válságból már Magyarország is kivette a részét, jelezve, hogy bekapcsolódott a világkereskedelembe. A következő hírhedt tőzsdeválság az 1895-ben a dél–afrikai aranybányákkal kapcsolatos részvényspekulációk miatt rázkódtatta meg a nemzetközi pénzpiacokat. Az 1929. október 14-i New York-i pedig a világválság kezdetét jelezte. A 19. század végétől a kormányok és a nagy bankok jelentősen beavatkoztak a tőzsdei üzletekbe, a krachok megakadályozása céljából.

### A devizakereskedés

A devizakereskedés nagyban hasonlít a részvényekkel való kereskedésre, a különbség, hogy itt a szóban forgó termék nem egy társaság által kibocsájtott részvény, hanem devizák keresztárfolyama. Sok tekintetben a devizapiacok állnak a legközelebb a tökéletes piac elméletéhez, aminek alapja a hatalmas forgalom, a piac rendkívüli likviditása, a szereplők magas száma, földrajzi szétszórtsága, a kereskedési idő (hétvégék kivételével napi 24 óra), az árfolyamokat befolyásoló tényezők sokfélesége és a magas elérhető haszonkulcs.

A devizapiac legfontosabb szereplői a kereskedelmi bankok, a külkereskedelemben részt vevő vállalatok, a nem banki pénzügyi intézmények (mint a befektetési alapok vagy a biztosító társaságok) és a központi bankok. Magánemberek is beléphetnek a devizapiacra – egy turista például, amikor a szálloda recepciójánál pénzt vált –, ám ezek a készpénzes tranzakciók a teljes devizakereskedelem töredékét teszik csak ki. [2]

A bankok a devizapiac mindennapos szereplői, hiszen ügyfeleik – főként a vállalatok – devizaigényeinek kielégítése szokásos napi munkamenetük részét képezi. Ezen túl a bankok más bankok számára is jegyeznek árfolyamot, amelyen hajlandók valutát venni, illetve eladni. A bankok közötti devizakereskedelem – amelyet bankközi kereskedelemnek nevezünk – teszi ki a devizapiaci forgalom legnagyobb részét.

### Magánemberként a devizapiacon

Átlagos állampolgárként is bekapcsolódhatunk a forex piac (foreign exchange market – nemzetközi devizapiac) körforgásába. Ehhez szükségünk van egy bróker cégre. Ezen cégek nemzetközi ellenőrzés alá esnek, így nagyon szigorú szabályok szerint kell működniük.

Rajtuk keresztül intézhetjük az ügyleteinket, valamint a legtöbb esetben tőkeáttételen keresztül extra tőkét nyújtanak. Ez azt jelenti, hogy ha van $1000-om, akkor pl. 1:15-ös tőkeáttétel esetén, $15000 értékben kereskedhetek, vagyis sikeres kötés esetén 15-szörösére nő a profitom. Ugyanakkor egy rosszul sikerült ügylet után a veszteség is 15-ször nagyobb lesz.

## Webes technológiák

Napjainkban a legtöbb hétköznapi alkalmazáshoz tartozik egy webes felület, ugyanis az gyakorlatilag platformfüggetlen, ami azt jelenti, hogy a lehető legszélesebb rétegekhez juttathatjuk így el a szolgáltatásainkat.

### Történelem

A webfejlesztés hőskorában egy weboldal nem volt más, mint egy statikus dokumentum. Ezen dokumentumok HTML (HyperText Markup Language) nyelven íródtak, amely a fájl strukturálását hivatott segíteni. Akkoriban a különböző böngészők egyedi módon vizualizálták a szövegesen megadott strukturáltságot.

Ahogy egyre több ember és egyre szélesebb körök kezdték el használni az internetet, úgy már nem volt elég a tartalomra koncentrálni puritán megjelenéssel. Nőtt az igény színvonalas webdizájnra és ezt a változást a nyelvnek is le kellett követnie.

A HTML 1.0 -nál csak az alapvető szerkezet kialakítása volt megvalósítható, olyan elemeké, mint a bekezdések, hivatkozások, fejlécek és felsorolások.

A 2-es verzió sem hozott sok újítást, de lehetőség volt például félkövér vagy dőlt betűk használatára, illetve képek beillesztésére. Ezen verzió kiegészítésében már megjelentek az űrlapok, illetve az ezen belüli többsoros szövegbeviteli és a kiválasztható opciók lehetőségei.

A nagyobb újítást az 1996-ban elfogadott HTML 3.0 és 3.2 jelentette, ahol már lehetőség volt java appletek, valamint scriptek beágyazására, melyek dinamikussá tudták tenni az addig statikus HTML kódot. Ezekben az új verziókban jelent meg először a „style” elem is, továbbá olyan tartalom- és struktúraformázási megoldások, mint a táblázatok vagy a betűtípusok változtatása.

1998-ban aztán megérkezett a HTML 4.0, melyben már megjelentek a nemzetközi karakterkészletek, támogatták a jobbról balra olvasást, továbbfejlesztették az űrlapok és táblázatok használhatóságát, valamint itt vált hivatalossá a frame-ek (keretek) használata is. [3]

### Modern webalkalmazások

2008-ban mutatták be a HTML5-öt, amely aztán 2014-ben szerzett hivatalos ajánlást. Az újítás elsődleges célja a beépülő modulok (Pl. Adobe Flash, Microsoft Silverlight) használatának megszüntetése volt. Ennek érdekében széles körben támogatja a natív multimédiás és grafikus elemeket. Ezen felül az új szabvány lehetőséget ad arra, hogy szemantikailag kifejezőbb strukturális elemeket használjunk. Ezzel a keresőmotorok könnyebben tudják értelmezni az oldalunk felépítését és tartalmát, amiért előrébb sorolják azt a találatok között.

Manapság a weboldalak túlnyomó többsége a HTML5 szabványt követi, azonban a webfejlesztők egyre kevesebb nyers HTML kódot írnak. Előtérbe kerültek ugyanis a reszponzív megjelenést és könnyebb fejlesztést biztosító keretrendszerek (pl. Angular), és osztálykönyvtárak (pl. React).

# Felhasznált technológiák

## Algoritmus

Az alkalmazás alapja egy forex kereskedő algoritmus, amely előre meghatározott mintákat figyelve reagál a piaci mozgásokra. Ennek a fejlesztésére a MetaTrader 4 kereskedő platform az MQL4 nyelvet szolgáltatja.

### MetaQuotes Language 4 (MQL4)

Az MQL4 alapvetően egy integrált programozási nyelv, amely segítségével kereskedő robotokat, indikátorokat, szkripteket és függvénykönyvtárakat lehet készíteni.

Az MQL4 egy C++ koncepciókat követve született, magas szintű, objektumorientált nyelv, amely rengeteg, az adatok elemzéséhez szükséges beépített függvényt tartalmaz. Tartalmaz továbbá beépített indikátorokat, így a leggyakrabban használt eszközök rendelkezésre állnak.

A nyelv felhasználásának két iránya van. Az egyik az adatok elemzése és ezzel a manuális kereskedés professzionálisabb szintre emelése, a másik a kereskedés teljes automatizálása. Én az utóbbi irányba indultam el, ami ugyan leveszi az emberről a manuális kereskedés terhét, azonban a robotokat karban kell tartani, mivel az idő múlásával és a piac változásával azok elavulhatnak.

## Backend

Az alkalmazás lelke a kiszolgáló szerver, amely kezeli az adatbázist, a több-felhasználós működést, figyeli a piaci mozgásokat, végrehajtja a szükséges piaci tevékenységeket, futtatja a historikus tesztert, valamint adatokat szolgáltat a grafikus felületnek. Ezen feladatok ellátása érdekében az alábbi technológiákat használtam fel.

### MongoDB

A MongoDB egy jól skálázódó, flexibilis dokumentum adatbázis (NOSQL - Not Only SQL), amely felépítéséből adódóan natívan támogatja a JavaScript fejlesztést.

A technológiát 2007-ben kezdték el fejleszteni, így meglehetősen kiforrottnak számít a dokumentum adatbázisok között és ez is a legnépszerűbb választás. Az adatbázis kezelése a mongo shellen keresztül történik, ami a JavaScript V8-as motorját használja. [5]

MongoDB-ben az adatokat JSON-szerű dokumentumokban tároljuk, így dokumentumról dokumentumra változhat azok struktúrája. Ez a felépítés lehetővé teszi struktúrák egymásba ágyazását is, amely a hagyományos adatbázisok sokszor költséges *join* műveletét eliminálja. Ez ugyanakkor hibázásra is lehetőséget ad, hiszen innentől a programozó felelőssége, hogy az adatmentések és a lekérdezések helyesek legyenek, de következetesen írt programkóddal könnyen ki lehet aknázni a technológia előnyeit. Az adatbázis minden dokumentumot egyedi \_id azonosítóval lát el, ezeken keresztül lehet azokat hivatkozni. A dokumentumokat kollekciókba szervezhetjük, ezzel az azonos jelentésű dokumentumokat egy helyről érhetjük el.

Mindezeken felül a MongoDB teljesen ingyenesen használható, ami a neves adatbáziskezelő szoftverekhez képest egy újabb előny.

### Node.js

A Node.js egy szoftverrendszer, melyben JavaScriptben írhatunk szerver oldali alkalmazásokat. Maga a rendszer C/C++-ban íródott, és egy esemény alapú I/O rendszert takar a JavaScript V8 motorja felett.

A JavaScript a világon a jelenleg legnépszerűbb programozási nyelv, többek között ezért esett erre a nyelvre a választás. A Node.js-t úgy írták meg, hogy (szinte) minden esemény aszinkron legyen, ezért az program sosem blokkolódik, azaz nem kell várni, hogy egy művelet befejeződjön, vele „párhuzamosan” futtathatunk további műveleteket. Ez pontosan ugyan úgy működik, mint a böngészőben levő kérések, vagy más események, mint például a *click*. Ez az alkalmazás folyamatosabb futását, valamint az egész rendszer optimálisabb működését teszi lehetővé. [5]

Mindezek mellett a Node.js legnagyobb előnye a Node Package Manager (NPM), amely egy online tárhely nyílt forráskódú Node.js projekteknek, valamint egyben egy parancssoros csomag-, verzió- és függőségkezelő is. Az egyszerű használat és az elérhető könyvtárak mennyisége mellett a fejlesztés alatt álló projektek hordozhatóságát is elősegíti, ugyanis a szükséges függőségeket a *package.json* fájlban tárolja, aminek segítségével egy parancsra bárhol telepíthetőek a szükséges könyvtárak.

### Express.js

Az Express.js egy minimalista, könnyűsúlyú keretrendszer Node.js felett, ami általános megoldást kínál web alkalmazások készítéséhez.

A keretrendszer könnyen kezelhetővé teszi az útvonalválasztást és az adott útvonalon végrehajtandó feladatok strukturálását is. Az útvonalakra middlewareket lehet feliratkoztatni, amelyek három paraméterrel rendelkező függvények. A három paraméter a *request*, a *response* és a *next*.Az első a beérkező http kérést, a második az arra adott választ takarja. Az egy útvonalra feliratkoztatott middlewarek egy láncot alkotnak, amelyek közül az Express.js az elsőt hívja meg. A harmadik paraméter (ami egy függvény) szolgál arra, hogy egy middlewareből továbblépjünk a láncban utána következőre. A next meghívása nélkül a futási lánc megszakad és a keretrendszer elküldi a http választ. Amennyiben előrelátóan fejlesztünk, a middlewarek adott esetben többször is felhasználhatóak, amely csökkenti a kódduplikációt.

Az egyszerűsége és robosztussága miatt széles körben kedvelt alap API-k készítéséhez.

### Passport.js

A Passport.js egy authentikációs middleware Express.js-hez.

Egy cél kiszolgálására tervezték, az pedig http kérések authentikációja. Használatával követjük a funkciók szétválasztása elvet, hiszen egymagában old meg egy feladatot, és így később nekünk ezzel máshol már nem kell foglalkoznunk.

Támogat a legegyszerűbb jelszó alapú lokálistól kezdve a token alapú és az OAuth authentikáción keresztül egészen a közösségi fiókokkal való belépésig mindent, amire egy webalkalmazás programozójának szüksége lehet.

### Oanda API / simple-fxtrade

Az Oanda egy online brókercég, akiken keresztül bárki bekapcsolódhat a tőzsdepiac működésébe.

A kereskedéshez kínálnak saját webes platformot, más asztali rendszereket kiszolgáló szervereket, valamint API-t is szolgáltatnak. Én az utóbbi lehetőséggel éltem és ennek segítségével írtam meg a kereskedő alkalmazásomat.

Az API-n keresztül elérhető az összes devizapárhoz tartozó grafikon összes adata, amelyeket felhasználva egyedi elemzéseket, vagy akár kereskedő algoritmusokat lehet készíteni. Ugyanitt teljes funkcionalitást kapunk pozíciók kötése terén is.

Az API-hoz a Simple FxTrade NPM csomag kínál wrapper szolgáltatást, ami leegyszerűsíti az API-hívások kezdeményezését.

## Frontend

### React

A React egy deklaratív, effektív, és rugalmas JavaScript könyvtár, felhasználói felületek készítéséhez. Lehetővé teszi komplex felhasználói felületek összeállítását izolált kódrészletekből, amiket komponenseknek hívunk. Mivel a komponens logikáját sémák helyett JavaScriptben írjuk, könnyedén lehet adatot mozgatni az alkalmazásban, és így az állapotok a DOM-on kívül maradnak. [4]

A komponenseket a React saját template nyelvében, JSX-ben írhatjuk. Egy komponenst egy JavaScript osztály, vagy függvény valósít meg. A render metódusban HTML-szerű kódot írhatunk, amelyet JavaScript kifejezéseket használva egészíthetünk ki, vagy módosíthatunk.

A komponensek egymásba ágyazhatók, így a fájlrendszerünk átlátható, szépen strukturált maradhat, valamint egy adott komponenst többször is felhasználhatunk. Éppen ezért érdemes minél kisebb logikai egységeket létrehozni.

A React egy úgynevezett virtual DOM (Document Object Model) koncepciót alkalmaz, amelynek segítségével a tartalom változása esetén újra tudja tölteni kizárólag a módosult részt, nem szükséges az egész oldalt lekérni újra a szerverről. Ezzel teljesítményben képes felülmúlni a vetélytársait, amennyiben ez a funkció kihasználásra kerül.

A virtual DOM-on belüli navigáció a React saját Router komponensével történik. Ezt általában a kiinduló komponensben helyezzük el.

A hátránya egy teljes keretrendszerrel szemben az, hogy itt a kiegészítő szolgáltatásokat nekünk kell összegyűjteni és integrálni az alkalmazásba, ugyanakkor ez egy nagyfokú szabadságot is biztosít a fejlesztőnek.

### React Bootstrap

A React Bootstrap az egyik leggyakrabban alkalmazott dizájn keretrendszer, a Bootstrap átültetése React webalkalmazásokhoz.

A keretrendszer egy egységes dizájnt biztosít az alkalmazáson belül. Mindezt úgy teszi, hogy a fejlesztőnek egy egyszerűbb alkalmazásnál nincs is szüksége arra, hogy CSS kódot írjon.

A React Bootstrap React komponenseket biztosít, amelyet úgy paraméterezhetünk fel, mint azok eredeti HTML megfelelőit. Ezek mellett kiegészítésként egyéb paramétereket is tudnak fogadni a komponensek, például az elrendezéshez, vagy a reszponzív megjelenéshez kapcsolódóan.

# Tervezés

## Algoritmus

## Backend

### Komponens diagram

### API végpontok

## Frontend

# Implementáció

## Algoritmus

## Backend

## Frontend

# Tesztelés

## Algoritmus

## Backend

## Frontend

# Összegzés

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | „econom.hu,” 18. január 2010.. [Online]. Available: http://www.econom.hu/a-tozsde-tortenete/. |
| [2] | M. Sághi, „A devizapiac,” in *Nemzetközi gazdaságtan - Elmélet és gazdaságpolitika*, Panem Könyvkiadó. |
| [3] | „livestudio.eu/,” [Online]. Available: https://livestudio.eu/hu/blog/55-a-html-tortenete-vagyis-a-weboldal-keszites-hoskora. |
| [4] | „hu.reactjs.org,” [Online]. Available: https://hu.reactjs.org/tutorial/tutorial.html. |

Függelék