SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 319

PLATFORMA ZA SIMULIRANJE VISOKOFREKVENTNOG TRGOVANJA DIONICAMA

Željan Juretić

Zagreb, lipanj 2011

Zahvaljujem se svojoj obitelji koja mi je pružala bezuvjetnu potporu tijekom cijelog školovanja.

Posebno se zahvaljujem prof.dr.sc. Branku Jerenu i dr.sc. Zvonku Kostanjčaru koji su me upoznali sa svijetom kvantitativnih financija. Njihov entuzijazam i podrška su uvijek bili nepresušan izvor motivacije.

Sadržaj

Popis slika

Popis Tablica

1 Uvod

Tehnološki napredak je omogućio automatizaciju rada svjetskih burza. S modernizacijom su se pojavili novi načini trgovanja koji su potpuno zasjenili tradicionalne tehnike kao što su tehnička i fundamentalna analiza. Među novim načinima trgovanja najistaknutije mjesto ima visokofrekventno trgovanje, kojim je 2009. godine ostvareno preko 60% volumena trgovanja na svjetskim burzama. Visokofrekventno trgovanje se temelji na velikom broju ostvarenih transakcija s relativno malim prinosima, pri čemu se kupljene dionice zadržavaju svega od nekoliko minuta do najviše nekoliko dana. Algoritmi za visokofrekventno trgovanje konstantno prate tržište s ciljem da uoče nepravilnosti u cijenama i ostvare neposrednu dobit. U poglavljima koja slijede ćemo detaljnije objasniti ovaj način trgovanja, upoznati se s brojnim prednostima koje on donosi te vidjeti koje izazove predstavlja pred nas.

Nažalost, visokofrekventno trgovanje je najčešće dostupno samo velikim investicijskim bankama i fondovima jer oni imaju potrebnu informatičku infrastrukturu i pristup podacima u stvarnom vremenu. Naš cilj je ostvariti sustav za simuliranje visokofrekventnog trgovanja dionicama na Zagrebačkoj burzi koji bi bio namijenjen istraživačima, malim investitorima i entuzijastima.

Ostvareni sustav će predstavljati potpuno funkcionalnu tehničku infrastrukturu za simuliranje visokofrekventnog trgovanja. Sustav će omogućiti korisniku kreiranje svojih algoritama te njihovo testiranje na povijesnim podatcima. Jednom pokrenuti, korisnički algoritmi će imati pristup podacima sa Zagrebačke burze u stvarnom vremenu te će biti u mogućnosti zadavati naloge za kupnju i prodaju dionica. Pri tome, korisnik će u svakom trenutku imati uvid u stanje portfelja, aktivne naloge zadane od strane algoritma te dosada obavljene transakcije. Budući da je izuzetno teško doći do povijesnih cijena dionica unutar dana, omogućit ćemo trajno pohranjivanje podataka o tržištu. Na ovaj način ćemo dobiti visokofrekventne serije cijena dionica koje se mogu iskoristiti za buduća znanstvena istraživanja, čime ovaj sustav dobiva dodatnu vrijednost.

Također, upoznat ćemo čitatelja s nekoliko najpopularnijih algoritama za visokofrekventno trgovanje te ćemo ih implementirati i prilagoditi za rad s našim sustavom. Ove implementacije mogu poslužiti kao predlošci koje korisnici mogu isprobavati i prilagođavati svojim potrebama.

2 Visokofrekventno trgovanje

Visokofrekventno trgovanje (HFT – *High-Frequency Trading*) je zavladalo Wall Street-om. Za procvat ovog načina trgovanja je zaslužna njegova velika profitabilnost. Prema pisanjima uglednog New York Timesa, u 2008. godini investitori koji su se oslanjali na visokofrekventno trgovanje su redom ostvarivali pozitivne prihode, dok je čak 70% investitora koji su zadržali stare načine trgovanja izgubilo svoj novac. Profitabilnost HFT-a se također ogleda i u eksponencijalnom rastu financijske industrije. Naime, izvještaji pokazuju da je početkom 2009. godine preko 60% volumena trgovanja na svjetskim burzama bilo ostvareno pomoću HFT-a. Čak i u najgorim mjesecima krize u 2008. godini, 50% transakcija je ostvareno upravo visokofrekventnim trgovanjem. Unatoč velikom zanimanju za ovu temu, postoji jako malo literature i znanstvenih članaka koji bi pomogli investitorima da se upoznaju s ovim načinom trgovanja te im omogućili da implementiraju vlastita rješenja.

Što je zapravo visokofrekventno trgovanje i zašto su investitori postali očarani njime? Najveća razlika između visokofrekventnog u odnosu na niskofrekventno trgovanje je u velikim obrtajima kapitala kojima upravlja računalo prateći promjene uvjeta koji vladaju na tržištu. Strategiju visokofrekventnog trgovanja karakterizira velik broj transakcija i u prosjeku manja ostvarena dobit po pojedinoj transakciji. Mnogi investitori zadržavaju svoje pozicije tjednima ili mjesecima, ostvarujući tako po transakciji prinos od nekoliko postotaka. Usporedbe radi, investitori koji koriste HFT, svakog dana naprave velik broj transakcija, ostvarujući po transakciji prinos od samo djelića postotka.

2.1 Prednosti visokofrekventnog trgovanja

Visokofrekventno trgovanje donosi brojne prednosti u odnosu na dosada uobičajene načine trgovanja. HFT je malo ili nimalo koreliran s tradicionalnim dugoročnim, kupi i zadrži strategijama, što ga čini vrijednim načinom diversifikacije za dugoročne portfelje. Također, visokofrekventne strategije zahtijevaju kraći period evaluacije zbog njihovih statističkih svojstava. Primjerice, tipična mjesečna strategija zahtijeva podatke u rasponu od šest mjeseci do dvije godine kako bi se utvrdila njena pouzdanost, dok je za većinu visokofrekventnih strategija dovoljno mjesec dana.

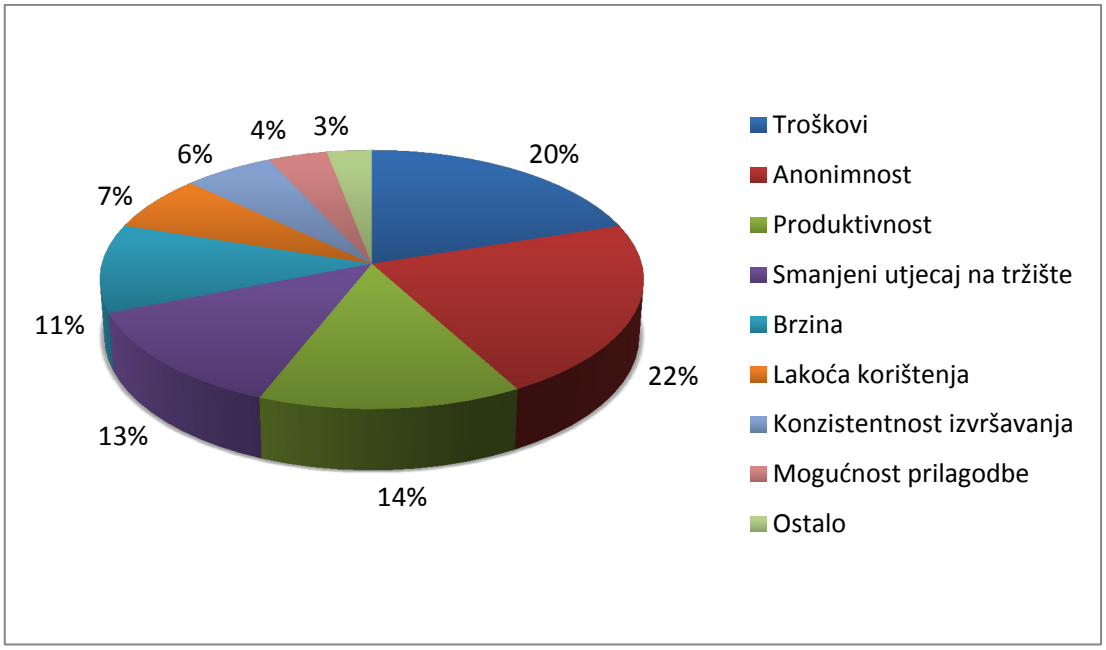
Iz operativne perspektive, automatiziranost HFT-a stvara uštede jer se može reducirati broj osoblja te se može smanjiti broj pogrešaka uzrokovan ljudskim faktorom (oklijevanje, utjecaj emocija i sl.).

Uz navedene prednosti, HFT povoljno utječe i na samo tržište. Izdvojit ćemo četiri najvažnija utjecaja HFT strategija na tržište:

- Povećana efikasnost tržišta
- Dodana likvidnost
- Inovacije u informatičkim tehnologijama
- Stabilizacija tržišnih sustava

HFT strategije pronalaze i iskorištavaju privremene neefikasnosti tržišta i na taj način pomažu brzom ugrađivanju informacija u cijene. Mnoge HFT strategije značajno doprinose likvidnosti tržišta, osiguravajući bolji rad samog tržišta (manje razlike između ponuđene i tražene cijene). Veliki investitori koji trguju pomoću visokofrekventnih strategija potiču inovacije u računarstvu. Najviše se ulaže u razvoj superračunala, tehnologija pohrane podataka te raznih rješenja koja bi rasteretila uska grla komunikacije preko Interneta. Naposljetku, treba dodati da HFT stabilizira tržišne sustave uklanjajući krive procjene cijena dionica.

Uz navedene prednosti postoji još pregršt razloga zašto su velike investicijske banke prigrlile algoritamski način trgovanja. Prema istraživanju kojeg je 2009. godine provela TRADE grupacija neki od važnijih razloga za uvođenje algoritamskog trgovanja su redom: anonimnost, smanjenje troškova, produktivnost, smanjeni utjecaj na tržište, brzina, lakoća korištenja, konzistentnost izvršavanja i mogućnost prilagodbe.



Slika 1. Prednosti visokofrekventnog trgovanja

Iako visokofrekventno trgovanje nudi brojne pogodnosti u odnosu na tradicionalne načine trgovanja, ono istovremeno donosi nove izazove koje ćemo opisati u nastavku

2.2 Izazovi za visokofrekventno trgovanje

Prvi izazov predstavlja ophođenje s izuzetno velikim količinama podataka. Za razliku od mnogih tradicionalnih analiza u kojima se koriste dnevne cijene, za primjenu HFT-a su potrebne cijene unutar dana (engl. *Intra-day data*). Osim što pohrana takvih podataka zahtijeva neizmjerno više memorijskog prostora, takvi podaci mogu biti nepravilno vremenski raspoređeni pa su potrebni novi alati i metodologije.

Drugi izazov se odnosi na preciznost signala za zadavanje naloga za kupnju, odnosno prodaju. Budući da se na razvijenim tržištima kapitala dobitci mogu brzo pretvoriti u gubitke ako signali za zadavanje naloga nisu usklađeni s tržištem, signal mora bit dovoljno precizan da može inicirati transakciju u djeliću sekunde.

Treći izazov je brzina izvršavanja naloga. Naravno, visokofrekventno trgovanje nije moguće na tržištima na kojima se nalozi zadaju telefonski. Kompjuterski automatizirano generiranje i izvršavanje naloga je jedini pouzdan način na koji se može ostvariti potrebna brzina i preciznost. Programiranje sustava za visokofrekventno trgovanje zahtijeva najtalentiranije i najiskusnije programere jer svaka pogreške prilikom izvođenja programa može biti suviše skupa. Stoga je nužno potreban ljudski nadzor za vrijeme rada sustava kako bi se osiguralo da sustav radi unutar unaprijed postavljenih granicama rizika. Kako bilo, ljudska intervencija je ograničena na donošenje samo jedne odluke, da li sustav još uvijek radi u definiranim granicama ili je pak došlo vrijeme da se isključi prekidač.

Konačno, natjecanje u tržišnoj utakmici za ostvarivanje što veće dobiti pomoću HFT-a zahtijeva konstantna održavanja i nadogradnje. Opstanak u utrci za profitom stoga nužno povlači i velika ulaganja u informacijsko komunikacijsku tehnologiju što si najčešće mogu priuštiti samo najveći igrači na tržištu.

2.3 Evolucija visokofrekventnog trgovanja

U svojim počecima, tržišta kapitala su funkcionirala u potpunosti na manualan način. Kad bi klijent htio saznati cijenu određene dionice, trebao bi kontaktirati svog zastupnika za prodaju, osobno ili preko kurira. Tek kasnije je, izumom telefonije, postalo moguće kontaktirati zastupnika telefonom. Kad bi zastupnik za prodaju zaprimio zahtjev otrčao bi do trgovačkog zastupnika zatražiti informacije o cijenama dionica koje zanimaju klijenta. Trgovački zastupnik bi tada dogovorio cijenu s ostalim brokerima i dojavio je zastupniku za prodaju koji bi potom obavijestio svog klijenta. Ako bi nakon toga klijent odlučio zadati nalog za kupnju ili prodaju cijeli ovaj proces bi se morao ponoviti. Naravno da je ovakav proces bio spor, skup i sklon. Troškove trgovanja snosio je klijent. Najviše pogrešaka se događalo iz dva razloga:

- Cijene na tržištu su se mogle značajno promijeniti od trenutka kada je cijena dobivena od trgovačkog zastupnika do trenutka kada je nalog zaprimljen na burzi.
- Višestruko prenošenje poruka između ljudi je unosilo brojne pogreške.

Tek 1980-ih godina se pojavio prvi elektronički sustav na burzama. Iako u vrlo jednostavnom obliku i dostupan samo profesionalcima za unutarnje operacije na burzi, sustav je već tada bio slavljen kao revolucija u

načinu trgovanja. Početkom 21. stoljeća burze diljem svijeta su započele s potpunom kompjuterizacijom svojih sustava. Prema procjenama Aite Group iz Bostona, 25% volumena trgovanja u 2001. godini je obavljeno elektroničkim putem, dok je 2008. godine taj udio narastao na 85%. Tehnološki napredak je omogućio značajan porast dnevnog volumena trgovanja. Tako se je na Newyorškoj burzi 1923. godine dnevno trgovalo s jednim milijunom udjela, a 2003. godine dnevno se trgovalo s preko milijardom udjela, što predstavlja porast od tisuću puta.

2.4 Evolucija metoda trgovanja

2.4.1 Tehnička analiza

Cilj tehničke analize je pronaći ponavljajuće uzorke u serijama cijena dionica. Tehnička analiza je doživjela svoj procvat u prvoj polovici 20-og stoljeća kada se komunikacija s burzom odvijala pomoću telegrafa i kada je kompleksnost trgovanja bila značajno niža od današnje. Nemogućnost brzog prijenosa informacija je ograničavala brzinu i količinu ostvarenih transakcija što je za posljedicu imalo sporije ugrađivanje novih informacija u cijene na tržištu. Trgovanja obavljena prethodnog dana su se pojavljivala u novinama slijedećeg jutra te su ona najčešće bila dovoljna da tehnički analitičari mogu uspješno predvidjeti buduća kretanja cijena. U desetljećima poslije Drugog svjetskog rata, tehnologije trgovanja na burzama su se počele značajno razvijati, što je dovelo do toga da se tehnička analiza pretvorila u samoispunjavajuće proročanstvo.

Primjerice, ako dovoljan broj ljudi vjeruje da nakon pojave uzorka „head-and-shoulders“[1] slijedi nagla rasprodaja određene dionice, tada će svi oni koji vjeruju u to nakon uočenog „head-and-shoulders“ uzorka zadati naloge za prodaju te će se na taj način uistinu i ostvariti predikcija. Poslije su velike investicijske banke započele provoditi tehničku analizu uz pomoć, za to doba, superračunala te su mogle iskoristiti takve prilike na tržištu prije nego su one postale očite svima. Danas je upotreba tehničke analize marginalizirana i koristi se samo pri niskofrekventnom trgovanju najnelikvidnijih dionica kojima se trguje jednom ili dvaput dnevno.

[1] „Head and Shoulders“ je poznati uzorak u cijenama dionica koji je dobio ime po svom obliku. U tehničkoj analizi se smatra da nakon pojave ovog uzorka slijedi promjena trenda cijena dionice.

Tehnička analiza je sredstvo kojim se želi iz povijesnih podataka o kretanju cijena zaključiti kakva će biti ponuda i potražnja na tržištu. Za razliku od tehničke analize koja se bazira na povijesnim podacima, suvremeno visokofrekventno trgovanje se bazira na otkrivanju latentnih informacija o tržištu iz sitnih promjena u najnovijim kretanjima cijena. U visokofrekventnom okružju, predefinirani uzorci tehničke analize nisu konzistentni, stoga se modeli za visokofrekventno trgovanje izgrađuju na stohastičkim ekonometrijskim modelima zaključivanja u koje se često ugrađuju elementi fundamentalne analize.

2.4.2 Fundamentalna analiza

Počeci fundamentalne analize su vezani upravo za trgovanje dionicama, u vrijeme kada su analitičari primijetili da budući tokovi novca, kao što je isplata dividendi, utječu na tržišne cijene dionica. Ti budući tokovi novca su tada bili diskontirani na sadašnju vrijednost kako bi se ostvarila pravedna sadašnja vrijednost dionice na tržištu. Tijekom godina, pojam fundamentalne analize je proširen i na određivanje vrijednosti financijskih instrumenata, koji nemaju jasan tok novca, pomoću očekivanja ekonomskih pokazatelja. Danas, fundamentalna analiza se odnosi na trgovanje vođeno očekivanjem da će se cijene kretati prema onima koje su predviđene odnosima ponude i potražnje. Različiti oblici fundamentalne analize se koriste kao aktivni ulazi u visokofrekventne modele trgovanja. Primjerice, kako bi se poboljšao proces visokofrekventnog trgovanja, fundamentalna analiza se može iskoristiti za procjenu fundamentalne vrijednosti nekog ekonomskog pokazatelja koji tek treba biti objavljen. Tehnička i fundamentalna analiza su koegzistirale kroz 20-to stoljeće, sve dok Wall Street nije preplavila nova vrsta trgovaca, oboružana doktoratima iz područja fizike i statistike.

2.4.3 Kvantitativno trgovanje

Novopridošlice su razvili napredne matematičke modele koji su bili nešto sasvim drukčije od onog što su nudile tehnička i fundamentalna analiza. Trgovanje se sada baziralo na naprednim matematičkim modelima koji skoro da i nisu imali poveznica s tradicionalnim načinima trgovanja. Strategije statističke arbitraže su postali svojevrsni favoriti u utrci za stvaranjem što većih profita. Kako su se vijesti o novim tehnikama širile među

trgovcima, uslijedila je bitka za konstantnim unapređenjima ovih algoritama; ljudi koji su se sa svojim poboljšanjima uspjeli izdignuti iznad ostalih ostvarivali su najveće dobiti.

Jedan aspekt ovog natjecanja je zasigurno bila brzina izvršavanja algoritama. Onaj tko je najbrže mogao izvršavati kvantne modele je mogao prvi uočiti neefikasnosti tržišta, iskoristiti ih i ostvariti najveći dobitak. Kako bi povećali brzinu izvršavanja algoritama, trgovci su se počeli oslanjati na jaka računala te se stvorila nova kultura trgovanja koja je bila usmjerena na tehnologiju. Kompjuterizirano trgovanje je postalo poznato pod nazivom „sistematsko trgovanje“ (engl. *Systematic trading*) zbog kompjuterskih sistema koji su procesirali velike količine podataka na temelju kojih bi donijeli odluku i izvršili naloge za kupnju i prodaju.

2.4.4 Visokofrekventno trgovanje

Visokofrekventno trgovanje se razvilo 1990-ih kao odgovor na tehnološki napredak i prihvaćanje novih tehnologija od strane svjetskih burza. Od svojih početaka kada se ovaj način trgovanja svodio na najjednostavnije sustave za zadavanje naloga pa do današnjih tehnoloških remek djela, sveobuhvatnih sustava za trgovanje, visokofrekventno trgovanje je evoluiralo u industriju vrijednu milijarde američkih dolara. Napredak informatičke tehnologije zadnjih godina, omogućio je potpunu automatizaciju visokofrekventnog trgovanja, generirajući tako još veće prihode za trgovce, ali istodobno povećavajući njihov interes za ulaganje u tehnologiju i istraživanja. Investicijske banke su također uvidjeli još jednu prednost HFT-a. Zbog automatiziranog rada, sada su banke mogle smanjiti broj zaposlenih i tako značajno smanjiti troškove. Neposrednost, točnost izvršavanja i nepostojanje oklijevanja su bile osobine u kojima se ljudi nisu mogli mjeriti sa superračunalima i naprednim algoritmima.

2.4.5 Suvremeni načini trgovanja

Tehnološkim napretkom su se, uz HFT, razvili i razni drugi načini trgovanja. Ukratko ćemo se osvrnuti na elektroničko, algoritamsko i sistematsko trgovanje te ćemo vidjeti kako su oni povezani s visokofrekventnim trgovanjem.

Elektroničko trgovanje se odnosi na mogućnost prenošenja naloga elektroničkim putem, a ne posredno pomoću telefona, pošte ili pak osobno. Budući da se u današnje vrijeme većina naloga na financijskim tržištima prenosi preko računalnih mreža, pojam elektroničkog trgovanja je postao nepotreban jer se on podrazumijeva.

Algoritamsko trgovanje je složenije od elektroničkog i odnosi si se na različite algoritme koji raspoređuju i optimiziraju proces izvršavanja naloga. Ovi algoritmi su namijenjeni optimizaciji izvršavanja trgovanja jednom kad je odluka o kupnji ili prodaji donesena. Algoritamsko izvršavanje donosi odluke o tome koji je najbolji način postavljanja naloga na burzu, što obuhvaća odluku o tome koji je najbolji trenutak za izvršavanje zadanog naloga ako se ne zahtjeva izvršavanje u istom trenutku te kako jedan nalog razlomiti na više manjih, u odnosu na količinu udjela, kako bi ostvarili optimalan efekt. Na primjer, sustav za visokofrekventno trgovanje može generirati signal za kupnju milijun dionica IBM-a. Taj signal se prosljeđuje sustavu za algoritamsko trgovanje koji može odlučiti da je zadani nalog najbolje izvršiti u inkrementima od sto dionica kako bi se spriječio nagli porast cijene na tržištu.

Uspješna implementacija sustava za visokofrekventno trgovanje zahtjeva izradu podsustava za generiranje visokofrekventnih signala za trgovanje i podsustava za optimizaciju izvršavanja naloga. Pri tome treba napomenuti da su algoritmi za generiranje signala u pravilu značajno kompleksniji od algoritama fokusiranih na optimizaciju izvršavanja naloga.

Sistematsko trgovanje se bazira na algoritmima koji donose odluke o kupnji ili prodaji određenih financijskih instrumenata koje ne moraju biti visokofrekventne. Primjer sistematskog trgovanja je računalni program koji se pokreće jednom dnevno, tjedno ili čak mjesečno. Ovakvi programi kao ulaz primaju zaključne dnevne cijene, a kao izlaz daju matricu portfelja te zadaju kupi-prodaj naloge kako bi ostvarili željeni portfelj.

Pravi sustav za visokofrekventno trgovanje donosi čitav niz odluka, od identifikacije podcijenjenih ili precijenjenih dionica, preko optimalnog odabira portfelja pa sve do najboljeg načina izvršenja naloga. No, glavna karakteristika HFT-a je kratko vrijeme zadržavanja pozicije, jedan dan ili kraće. Po svojoj prirodi, većina visokofrekventnih sustava su primjeri sistematskog i algoritamskog trgovanja, ali obratno ne vrijedi.

3 Web servisi

Web servis je način komuniciranja između dva elektronička uređaja preko mreže. W3C [2] konzorcij definira web servis kao softverski sustav dizajniran za podršku interoperabilnosti interakcije računalo-računalo koja se odvija preko mreže. Općenito gledajući, web servis je Web API (engl. *Application Programming Interface*) kojemu se pristupa preko HTTP protokola (engl. *HyperText Transfer Protocol*) i koji se izvršava na nekom udaljenom sustavu, pružatelju usluge. U kontekstu razvoja web aplikacija, Web API je definirani skup HTTP zahtjeva sa točno definiranom strukturom poruka koja se najčešće izražava pomoću XML(engl. *Extensible Markup Language*) ili JSON(engl. *JavaScript Object Notation*) formata.

3.1 XML i JSON formati

XML i JSON su standardi za formatiranje podataka u podatkovnim razmjenama. Oba formata se intenzivno koriste u serijalizaciji i slanju strukturiranih podataka preko mreže. Glavna razlika između ova dva standarda se ogleda u tome što je JSON u ljudima čitljivijem formatu nego što je to XML. Demonstrirat ćemo to kratkim primjerom pomoću kojeg ćemo se upoznati s osnovama oba formata.

Tablica 1. Usporedba XML i JSON formata

Podaci	XML format	JSON format

Ime: Ivan
Prezime: Horvat
Visina: 185
Tezina: 90
	<osoba>
	<ime>Ivan</ime>
	<prezime>Horvat</prezime>
	<visina>185</visina>
	<tezina>90</tezina>
	</osoba>
		{
		"osoba":
		{
		"ime":"Ivan",
		"prezime":Horvat,
		"visina":185,
		"tezina":90
		}
		}

Podaci u XML formatu su omeđeni tagovima, dok su u JSON formatu podaci prikazani kao asocijativni niz. Kako bi došli do tržišnih podataka za Zagrebačku burzu moramo koristiti JSON web servis. Stoga ćemo u nastavku detaljnije pojasniti gramatička pravila kojima se opisuje JSON format.

[2] W3C (engl. *World Wide Web Consortium*), je kratica glavne međunarodne organizacije zadužene za donošenje Internet standarda.

3.2 JSON format

JSON (engl. *JavaScript Object Notation*) je format za razmjenu podataka koji je ljudima razumljiv za čitanje i pisanje. Važna karakteristika ovog formata je jednostavnost pri parsiranju podataka iz formatirane poruke, odnosno jednostavnost pri generiranju same poruke. JSON je tekstualni format temeljen na podskupu programskog jezika JavaScript, međutim ovaj format je u potpunosti neovisan o programskom jeziku programera ili krajnjeg korisnika.

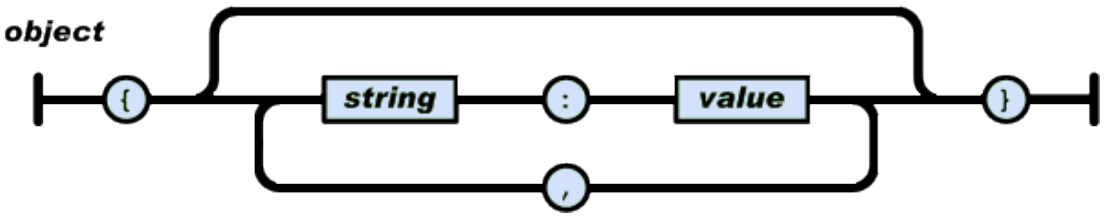
JSON se gradi od dvije strukture:

- Kolekciji parova naziv/vrijednost. Ova struktura je standardno podržana u mnogim programskim jezicima, a najčešće je realizirana kao `object`, `record`, `struct`, `dictionary`, `hash table`, `keyed list` ili `associative array`.
- Uređenoj listi vrijednosti. U većini programskih jezika, uređena lista vrijednosti je realizirana kao `array`, `vector`, `list` ili `sequence`.

Navedene strukture podataka su univerzalne te ih svi moderni programski jezici podržavaju u nekom obliku. Stoga je razumno da se format za razmjenu podataka između raznih programskih jezika temelji upravo na ovim strukturama.

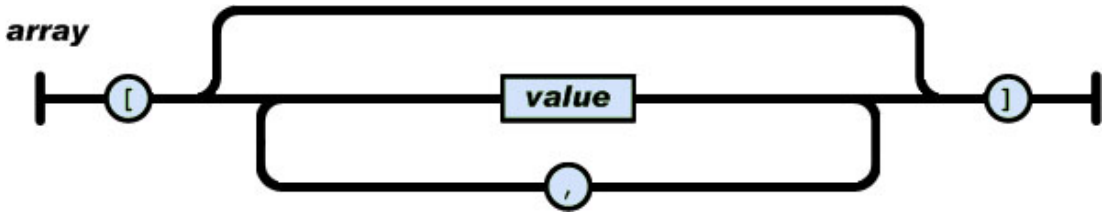
U JSON-u, one mogu poprimiti slijedeće oblike:

1) **Objekt** (engl. *Object*), je neuređeni skup parova naziv/vrijednost. Objekt počinje lijevom vitičastom, a završava desnom vitičastom zagradom. Nakon svakog naziva slijedi znak dvotočke i pripadna vrijednost. Ukoliko postoji više parova naziv/vrijednost, tada ih odvajamo zarezom.



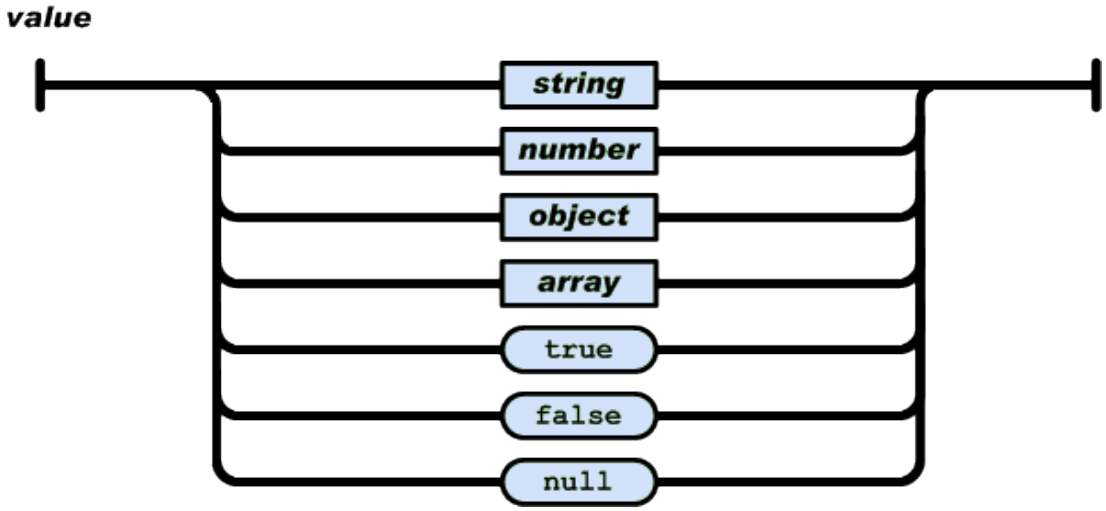
Slika 2. Gramatičko pravilo za stvaranje JSON objekta

2) **Niz** (engl. *Array*), je uređeni skup vrijednosti. Niz počinje lijevom uglatom, a završava desnom uglatom zagradom. Vrijednosti se odvajaju zarezom.



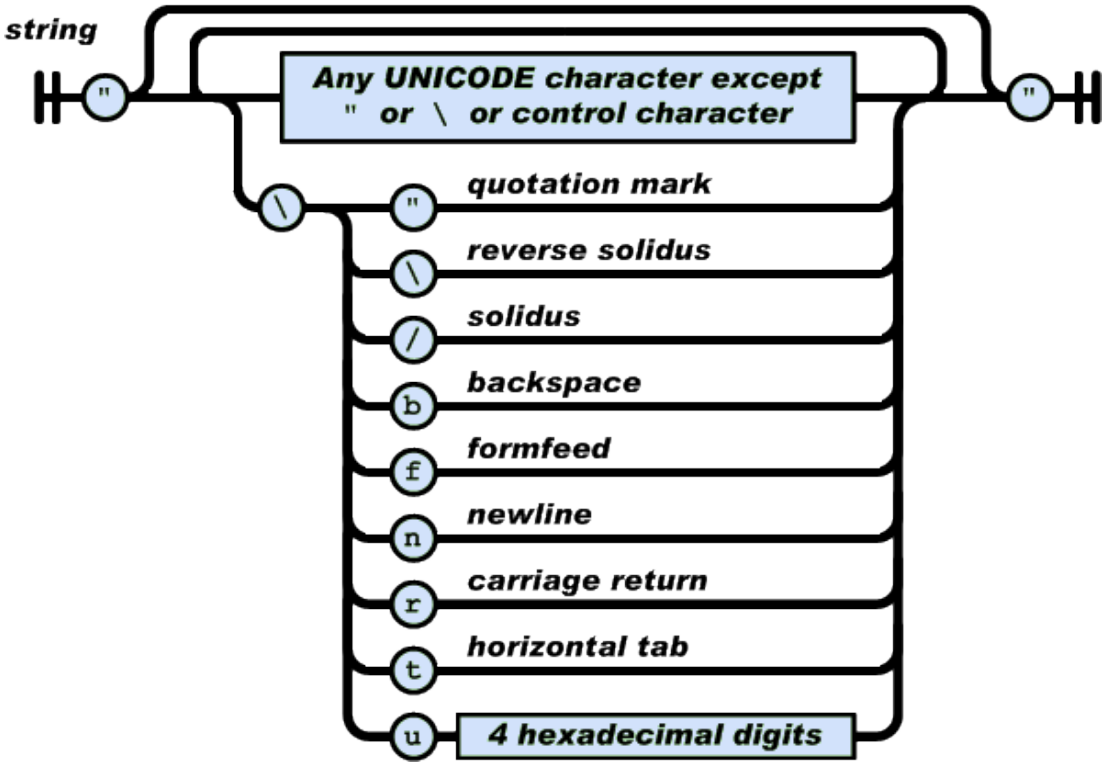
Slika 3. Gramatičko pravilo za stvaranje JSON niza

3) **Vrijednost** (engl. *Value*), može biti niz znakova omeđen dvostrukim navodnicima, broj, true, false, null, objekt ili niz. Navedene strukture se mogu ugnježđivati.



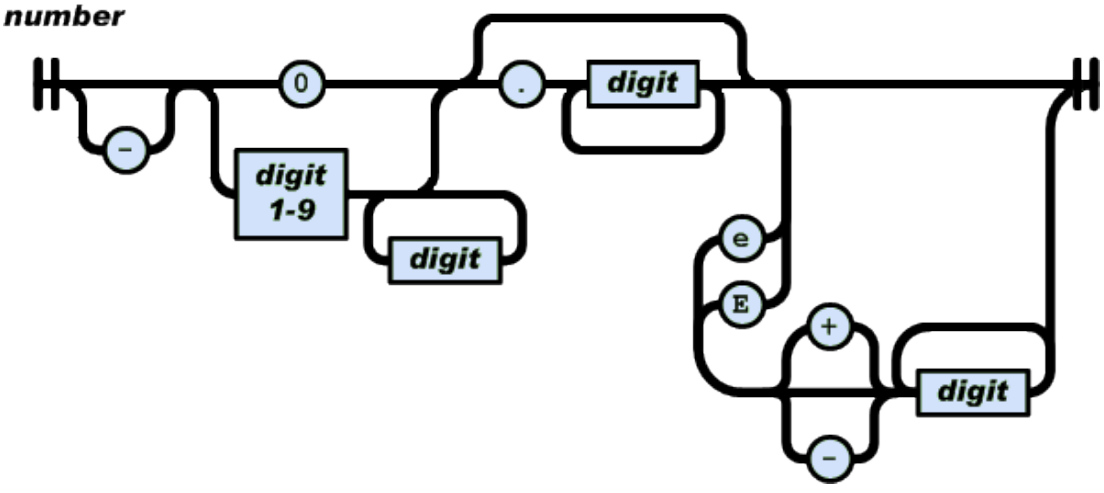
Slika 4. Gramatičko pravilo za stvaranje JSON vrijednosti

4) **Niz znakova** (engl. *String*), je niz od nijednog ili više Unicode znakova, omeđenih dvostrukim navodnicima. Niz znakova u JSON-u slijedi većinu pravila kao niz znakova u programskim jezicima C i Java.



Slika 5. Gramatičko pravilo za stvaranje JSON znakovnog niza

5) **Brojevi** u JSON formatu slijede konvencije brojeva u programskim jezicima C i Java s iznimkom da nisu podržani oktalni i heksadecimalni brojevi.



Slika 6. Gramatičko pravilo za stvaranje JSON broja

Razmaci mogu biti dodani između bilo koja dva simbola. Izuzevši nekoliko detalja o kodiranju, gornjim gramatičkim pravilima smo u potpunosti opisali JSON format.

3.3 Agram Brokeri web servis

Internet adresa web servisa Agram Brokera je `online.agram-brokeri.hr`. Za pristupanje web servisu potrebno je ispravno korisničko ima i lozinka koji se dobiju prilikom sklapanja ugovora s brokerskom kućom Agram Brokeri. Prijavu na web stranicu izvršavamo programski pri čemu se stvara HTTP „kolačić“ (engl. *Cookie*), enkriptirana datoteka koja sadrži informacije potrebne za dobivanje dozvole korištenja web servisa. Ovaj web servis koristi JSON format za razmjenu poruka s korisnicima. Sadržaj zahtijevamo tako da pošaljemo poruku sa četiri parametra u JSON formatu. Opišimo značenje ulaznih parametara servisa:

- detailsTicker: je parametar kojim se određuje za koju dionicu se žele detaljniji podaci, tj. za koju dionicu želimo dobiti trenutno aktivne naloge za kupnju i prodaju te trgovanja obavljena tog radnog dana.
- forceGetDetails: parametar kojim uključujemo, odnosno isključujemo traženje detaljnijih podataka. Ispravne vrijednosti ovog parametra su true i false.
- seqNumSecBoards: parametar koji omogućuje servisu da sazna koje informacije o cijenama dionica nam je poslao posljednji put. Ako prvi puta pristupamo web servisu tada ovaj parametar postavljamo na -1.
- seqNumTrades: parametar koji omogućuje servisu da sazna koje informacije o obavljenim trgovanjima nam je poslao u posljednjoj poruci. Ako prvi puta pristupamo web servisu tada ovaj parametar postavljamo na vrijednost 0.

Prvi poziv web servisu uvijek se upućuje sa slijedećim parametrima:

Tablica 2. Vrijednosti parametara prve JSON poruke

detailsTicker	`null`
forceGetDetails	`false`
seqNumSecBoards	`-1`
seqNumTrades	`0`

Prema gramatičkim pravilima JSON formata, opisanim u prethodnom poglavlju, generiramo JSON poruku:

```
{"seqNumSecBoards":1,"seqNumTrades":0,
"detailsTicker":null,"forceGetDetails":false}
```

Svaka poslana poruka mora imati odgovarajuće zaglavlje kako bi web servis znao kome treba poslati svoj odgovor te u kojem formatu on mora biti. U

slijedećoj tablici se nalazi popis svih varijabli zaglavlja koje se koristi za komunikaciju s Agram Brokeri web servisom te njihov odgovarajući sadržaj.

Tablica 3. Varijable zaglavlja HTTP zahtjeva

Host	online.agram-brokeri.hr
User-Agent	Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	en-us,en;q=0.5
Accept-Encoding	gzip, deflate
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive	115
Connection	keep-alive
Content-Type	application/json; charset=utf-8
Referer	https://online.agram-brokeri.hr/streamer/streamer.aspx
Content-Length	84
Cookie	__utma=1.1221064420.1306320304.1306320304.1306320304.1;__utmz=1.1306320304.1.1.utmcsr=agram-brokeri.hr|utmccn=(referral)|utmcmd=referral|utmcct=/; ChartOptions=ip=0&dp=3&df=21.2.2011 00:00:00&dt=21.5.2011 00:00:00&v=False&btt=3&ct=1&ot=&li=MjY7MTI7OQ==|5|#FF0000|&ui=MTA7MCw1|2|#0000FF|; __utmb=1.2.10.1306320304; __utmc=1; ASP.NET_SessionId=hemsjs45ub0vvz553d43ni25; .ASPXFORMSAUTH=C9608195483E57AB83FE45574F38DAE33FC7639C249CE5FEE47A277F0B931836D34BBC485733B187B8653124950C91A3D355AF01F7CD2343E524CBD1E616BD56CAAEFA3EF78C3DBB49F555289EFA260D

Nakon što primi prvu poruku koja ima sadržaj i zaglavlje kao što je prethodno opisano, web servis nam odgovara porukom koja je također u JSON formatu. Ta poruka sadrži popis imena svih dionica kojima se trguje na Zagrebačkoj burzi te sve informacije o trenutnom stanju njihovih cijena. Svaka slijedeća poruka poslana od web servisa će sadržavati samo informacije o eventualnim promjenama na tržištu od trenutka kada je poslana prethodna poruka. Ovakvim načinom rada web servisa, značajno se smanjuje veličina razmijenjenih poruka te se sprečava zagušenje web servisa. Stoga, web servis pamti svoje korisnike i pridjeljuje im dvije jedinstvene oznake koje mu služe da odredi kad je korisnik zadnji puta poslao zahtjev za podacima te koje su promjene nastupile od tada. Jedna oznaka prati promjene kod dionica, a druga prati promjene kod izvršenih

transakcija i mogu se pronaći u svakom odgovoru dobivenom od servera pod nazivima, SecuritiesSeqNum i TradesSeqNum. Ukoliko je neka od ovih oznaka jednaka nuli, to znači da nije bilo promjena te važeća ostaje oznaka iz prethodnog odgovora kojeg smo dobili od web servisa.

Primjerice ako smo iz poruke koju smo dobili kao odgovor web servisa na naš prvi poslani zahtjev utvrdili da su vrijednosti oznaka SecuritiesSeqNum=29725 i TradesSeqNum=1022 tada prilikom slanja slijedećeg zahtjeva moramo web servisu predati te dvije vrijednosti. Vrijednosti predajemo preko parametara seqNumSecBoards i seqNumTrades kako je prikazano tablicom.

Tablica 4. Primjer vrijednosti parametara slijednih JSON poruka

detailsTicker	null
forceGetDetails	false
seqNumSecBoards	29725
seqNumTrades	1022

Sadržaj JSON poruke se formira na identičan način kao i kod prve poslane poruke.

```
{"seqNumSecBoards":29725,"seqNumTrades":1022,"detailsTicker":
null,"forceGetDetails":false}
```

Dodatno, moguće je od web servisa zatražiti listu naloga za kupnju i prodaju te listu transakcija obavljenih od početka radnog dana za pojedinačnu dionicu. Na primjer, ako bi htjeli dobiti podatke sa web servisa o aktivnim nalozima i obavljenim transakcijama za dionice HT-a moramo u skladu s time promijeniti prva dva parametra zahtjeva kojeg šaljemo web servisu.

Tablica 5. Vrijednosti parametara JSON poruke kojom tražimo detaljnije podatke

detailsTicker	"HT-R-A"
forceGetDetails	true
seqNumSecBoards	29737
seqNumTrades	1023

4 Programski model domene

Kako bi uspješno modelirali sustav, potrebno je koncepte iz domene problema apstrahirati i preslikati u objektno orijentirani model. Koncepti koji se direktno mogu preslikati iz domene se obično preslikavaju u entitete i vrijednosne razrede koji tvore logički sloj aplikacije. Uz koncepte preuzete iz domene problema susrećemo se i s razredima koji spadaju u kategorije tvornica, repozitorija i servisa. U nastavku ćemo opisati ostvareni programski model na kojem se temelji naš sustav.

4.1 Entiteti i vrijednosni razredi

Market razred je modeliran pomoću *singleton* oblikovnog obrasca što znači da tijekom rada aplikacije postoji samo jedan objekt ovog tipa. Ovaj razred opisuje trenutno stanje burze te predstavlja *agregat* (engl. *Agregate*) pomoću kojeg se pristupa ostalim kolekcijama objekata kao što su Stocks, Indexes, StockNames, Orders, Trades.

Stock je razred koji sadrži sve relevantne informacije o pojedinoj dionici koji trenutno vrijede na tržištu kapitala. Ovaj razred je entitet jer je jedinstveno identificiran pomoću svoje oznake(engl. *Ticker*). Navedimo i ostala svojstva ove klase koja predstavljaju informacije koje korisnik može dobiti:

- Ticker: oznaka koja jedinstveno identificira dionicu na Zagrebačkoj burzi
- LastPrice: zadnja cijena po kojoj se trgovalo određenom dionicom
- DayChange: relativna promjena u odnosu na zaključnu cijenu prethodnog radnog dana koja je izražena u postocima.
- BestBid: najviša cijena koja je trenutno ponuđena na tržištu za kupnju određene dionice
- BestAsk: najniža cijena po kojoj netko prodaje određenu dionicu
- HighPrice: najviša cijena koju je neka dionica postigla tijekom radnog dana
- LowPrice: najniža cijena koju je neka dionica postigla tijekom radnog dana
- AveragePrice: prosječna cijena određene dionice za trenutni radni dan
- Volume: količina, odnosno broj udjela određene dionice kojim se trgovalo tog radnog dana
- Value: promet, odnosno vrijednost obavljenih transakcija za određenu dionicu tog radnog dana
- TradeCount: broj obavljenih transakcija za određenu dionicu tog radnog dana
- PreviousPrice: cijena po kojoj je obavljena prethodna transakcija za određenu dionicu
- OpenPrice: cijena za određenu dionicu koja je vrijedila na početku trgovanja tog radnog dana.

- IsActive: zastavica koja nam govori da li je trgovanje određenom dionicom moguće.

Index je razred koji sadrži sve trenutne relevantne informacije o indeksima na Zagrebačkoj burzi. Ova klasa je entitet jer je jednoznačno identificirana pomoću jedinstvenog naziva indeksa. Svojstva koja sadrži ova klasa su slijedeća:

- Code: naziv indeksa. Na Zagrebačkoj burzi postoje dva indeksa koji se odnose na dionice, CROBEX i CROBEX10 te jedan obveznički indeks, CROBIS.
- Value: trenutna vrijednost indeksa.
- DayChange: relativna dnevna promjena indeksa izražena u postocima.

StockName je razred koji sadrži oznaku te puni naziv dioničkog društva. U listi StockNames se mogu pronaći sve oznake i nazivi odgovarajućih dioničkih društva koja su izlistani na Zagrebačkoj burzi.

Trade je razred koji opisuje pojedinačnu transakciju obavljenu na Zagrebačkoj burzi. Razred je određen sa svojim svojstvima:

- Number: jedinstveni identifikator transakcije na Burzi
- Ticker: oznaka dionice kojom se je trgovalo
- Price: cijena za jednu dionicu po kojoj je transakcija provedena
- Quantity: broj dioničkih udjela koji su sudjelovali u transakciji
- Value: ukupna vrijednost transakcije
- DateTime: datum i vrijeme kad se obavila transakcija

Order je razred koji opisuje aktivni nalog zadan na Zagrebačkoj burzi. Svojstva kojima je opisana ova klasa su:

- Ticker: oznaka dionice na koju se odnosi nalog
- OrderType: tip naloga, nalog za kupnju ili prodaju
- Price: cijena za jednu dionicu po kojoj investitor želi izvršiti transakciju
- Quantity: broj dioničkih udjela koje investitor želi kupiti, odnosno prodati
- Value: vrijednost koju bi imala transakcija u slučaju da se nalog izvrši po zadanim uvjetima.

Simulation je razred pomoću kojeg modeliramo pojedinačne simulacije visoko-frekventnog trgovanja unutar naše aplikacije. Ova klasa je entitet, jednoznačno je određena svojim nazivom te datumom i vremenom kreiranja. Osim što je identitet, ova klasa je i agregat preko kojeg pristupamo kolekcijama podataka važnima za izvođenje same simulacije. Svojstva ovog razreda su:

- Name: naziv kojeg je korisnik pridružio simulaciji
- StartDate: datum kreiranja simulacije

- Interval: vremenski interval koji predstavlja frekvenciju pokretanja same simulacije. Interval je izražen u sekundama.
- State: govori nam da li je simulacija aktivna (*active/suspended*)
- StartPortfolio: referenca na objekt tipa Portfolio preko koje možemo doznati kako je izgledao početni portfelj investitora.
- Project: referenca na objekt tipa Project koji sadrži sve informacije o datotekama koje se koriste u simulaciji.
- Orders: kolekcija svih aktivnih naloga koje je zadao simulirani algoritam visokofrekventnog trgovanja
- Trades: kolekcija svih ostvarenih transakcija, odnosno svi nalozi koji su uspješno izvršeni.

Ukratko ćemo opisati i metode ovog razreda.

- Run(): je središnja metoda klase Simulation. Njenim pozivom pripremaju se svi parametri radne okoline, te se pokreće izvršavanje algoritma simulacije kojeg je korisnik ostvario u programskom jeziku MATLAB.
- readOrders(): ako postoje nalozi koje je zadala netom izvršena simulacija, onda se oni učitavaju u program.
- filterOrders(): odbacuje neispravno zadane naloge. Primjerice, odbacit će se nalozi za kupnju za koje investitor nema dovoljno gotovine. Također, odbacit će se nalozi za prodaju dionica koje investitor ne posjeduje.
- ExecuteOrders(): provjerava da li su uvjeti na tržištu takvi da je moguće izvršiti neki od aktivnih naloga. Ako su uvjeti ispunjeni, nalog se briše iz liste naloga i prebacuje se u listu ostvarenih transakcija.
- GetCurrentPortfolio(): ova metoda vraća referencu na objekt tipa Portfolio koji sadržava sve podatke o trenutnom portfelju investitora. Trenutni portfelj se izračunava na temelju početnog portfelja i svih ostvarenih transakcija. Na ovaj način se izbjegavaju redundancija i mogućnost pogreške.

Project razred sadržava podatke o organizaciji direktorija i datoteka simulacije. Svakoj simulaciji je pridružen jedinstveni direktorij koji je imenovan po nazivu simulacije. Unutar tog, glavnog direktorija nalaze se dva poddirektorija, jedan za pohranu podataka o trenutnom stanju tržišta te drugi za pohranu napisanih korisničkih skripta. Putanje do ovih direktorija se mogu zatražiti preko svojstava razreda:

- Folder: putanja do vrhovnog direktorija simulacije.
- DataFolder: putanja do direktorija s podacima o tržištu.
- ScriptsFolder: putanja do direktorija s korisničkim skriptama.

Od metoda ove klase možemo izdvojiti dvije najvažnije:

- WriteDataFiles(): organizira sve relevantne informacije o tržištu te ih na kompaktan način zapisuje u datoteke unutar direktorija namijenjenog za pohranu podataka. Ova metoda se poziva uvijek neposredno prije pokretanja same simulacije.

- CreateScripts(): stvara predefinirane MATLAB skripte unutar direktorija namijenjenog za pohranu istih. Stvaraju se tri vrste skripta. Prva vrsta se sastoji od predefiniranih razreda koji služe kao strukture podataka. Druga vrsta sadrži pomoćne funkcije koje korisniku olakšavaju pisanje skripta te funkcije pomoću kojih simulacija može zadati naloge. U treću skupinu spadaju dvije MATLAB skripte: loadData.m i main.m. LoadData.m je skripta koja učitava podatke o tržištu te ih pohranjuje u odgovarajuće strukture podataka. Main.m je središnja skripta u koju korisnik piše svoj algoritam.

Portfolio razred je namijenjen opisu korisničkog portfelja te sadrži informacije o dioničkim udjelima koje investitor posjeduje te o gotovini koju ima na raspolaganju. Ovim informacijama se pristupa pomoću svojstava Cash i Stocks.

StockInPossesion je razred koji spada u vrstu vrijednosnih (engl. *Value*), objekata što znači da sadrži informacije, ali nema svoje konceptualno značenje. Ovim razredom se jednostavno grupiraju informacije o dioničkim udjelima unutar portfelja. Svojstva razreda su:

- Ticker: oznaka dionice koju posjeduje investitor.
- Volume: količina dioničkih udjela u posjedu investitora.

4.2 Tvornice

Oblikovni obrazac tvornice (engl. *Factory Design Pattern*), je jedan od najkorištenijih oblikovnih obrazaca u modernim programskim jezicima kao što su C# i Java. Dolazi u različitim varijantama i implementacijama. Namjena tvornice je da stvara objekte, a da pritom ne izlaže korisniku inicijalizacijsku logiku. Stoga bi oblikovni obrazac tvornice mogli definirati kao programski element, odgovoran za kreiranje drugih objekata koji u tu svrhu enkapsulira znanje potrebno za kreiranje objekta na jednom mjestu. Ovaj obrazac je najkorisniji pri stvaranju velikih agregata ili objekata s kompleksnom hijerarhijom, stoga ćemo ga pri izgradnji naše aplikacije iskoristiti kod stvaranja glavnih agregata, razreda Stock i Simulation.

StockFactory je razred koji je modeliran pomoću *Factory* oblikovnog obrasca. Njegova namjena je da stvara i inicijalizira objekte tipa Stock. Provjerava se da li su svi podaci koji opisuju trenutno stanje neke dionice na tržištu dobro isparsirani iz JSON formata te ukoliko neku vrijednost nije moguće postaviti baca se iznimka tipa StockFactoryException.

SimulationFactory je razred namijenjen stvaranju i inicijalizaciji objekata tipa Simulation. Osigurava da su svi potrebni parametri ispravno postavljeni. Ukoliko naziv simulacije nije postavljen baca se iznimka tipa SimulationNameUndefined.

Dodatno se provjerava da li je definiran projekt za novu simulaciju te se u slučaju pogreške baca iznimka tipa SimulationProjectUndefined.

4.3 Repozitoriji

Repozitorij je konceptualni okvir koji enkapsulira rješenje za dohvaćanje objekata. On djeluje kao kolekcija objekata sa poboljšanim mogućnostima dohvaćanja i pretraživanja. Također, repozitorij se brine o perzistenciji objekata, sprema ih u baze ili datoteke te osigurava da se u radnoj memoriji uvijek nalazi ispravan objekt. Repozitoriji značajno olakšavaju posao klijentu jer mu daju na raspolaganje jednostavno sučelje iza kojeg se krije kompleksna tehnička infrastruktura. Treba istaknuti i činjenicu da upotrebom repozitorija ostavljamo mogućnost da se naknadno može relativno jednostavno zamijeniti konkretni sustav pohrane podataka (baza podataka, XML datoteke, serijalizacija itd.). Dvije skupine objekata bi nužno trebale imati svoje repozitorije:

- Podskup perzistentnih objekata mora biti globalno dostupan preko pretrage po atributima.
- Korijeni agregata koji nisu dostupni preko asocijacija.

U našoj domeni perzistiramo agregat Simulation sa svim ostalim razredima na koje on sadrži referencu. Štoviše, korijeni objekata tipa Simulation nisu dostupni preko agregata pa razred Simulation mora imati vlastiti repozitorij.

Iako ne perzistiramo ostale razrede iz domene, imamo još jedan agregat, razred Market. Međutim, moramo se prisjetiti da je razred Market izveden pomoću Singleton oblikovnog obrasca što asocijaciju na njegov korijen čini uvijek globalno dostupnom pa stoga ovaj razred ne treba tvoj repozitorij.

Iako smo zaključili da je u našoj aplikaciji potreban samo jedan repozitorij odlučili smo se na implementaciju generičkog repozitorija koji može poslužiti kao repozitorij za bilo koji objekt. Na ovaj korak smo se odlučili zbog mogućih naknadnih nadogradnja i poboljšanja programa. Osim što je generički, repozitorij implementira ISerializable sučelje pa je moguće pozivom jedne funkcije serijalizirati čitavu kolekciju objekata.

Slijedi kratak pregled najvažnijih metoda ostvarenog repozitorija:

- Add(T inObject): dodaje novi objekt u repozitorij pri čemu se provjerava da li objekt već postoji.
- Update(T inObject): ažurira stanje objekta koji se već nalazi u repozitoriju.
- GetByIndex(int index): dohvaća objekt po indeksu kojeg ima unutar kolekcije u repozitoriju. Ova metoda je pogodna za iteriranje po cjelokupnom sadržaju repozitorija.

- Exists(Func<T, bool> filter): vraća logičku istinu ako u repozitoriju postoji barem jedan objekt koji zadovoljava logički uvjet zadan preko ulaznog parametra.
- Find(Func<T, bool> filter): vraća prvi objekt iz repozitorija koji zadovoljava logički uvjet zadan preko ulaznog parametra.
- FindAll(Func<T, bool> filter): vraća listu objekata iz repozitorija koji zadovoljavaju logički uvjet zadan preko ulaznog parametra.
- Remove(Func<T, bool> filter): uklanja iz repozitorija jedan ili više objekata koji zadovoljavaju logički uvjet zadan ulaznim parametrom.
- Remove(T item): uklanja objekt iz repozitorija.
- Clear(): briše sve objekte sadržane u repozitoriju.

4.4 Servisi

Servisi su razredi koji pružaju operacije povezane s konceptom domene koji prirodno ne spada u entitet ili vrijednosni objekt. Dvije osnovne karakteristike servisa su da je njihovo sučelje definirano u terminima drugih elemenata modela domene te da operacija servisa ne mijenja stanje samog servisa (engl. *Stateless*). Kada operacija ne mijenja stanje servisa, tada to u praktičnom smislu znači da svaki klijent može koristiti bilo koju instancu servisa bez obzira na povijest te instance. Zbog svoje prirode, servise imenujemo po aktivnostima koje obavljaju.

Također, treba napomenuti da servisi ne spadaju samo u model domene. Ovisno o operacijama koje omogućuju oni također mogu postojati i u infrastrukturnom te aplikacijskom sloju. Opišimo servise koje smo implementirali u našoj aplikaciji:

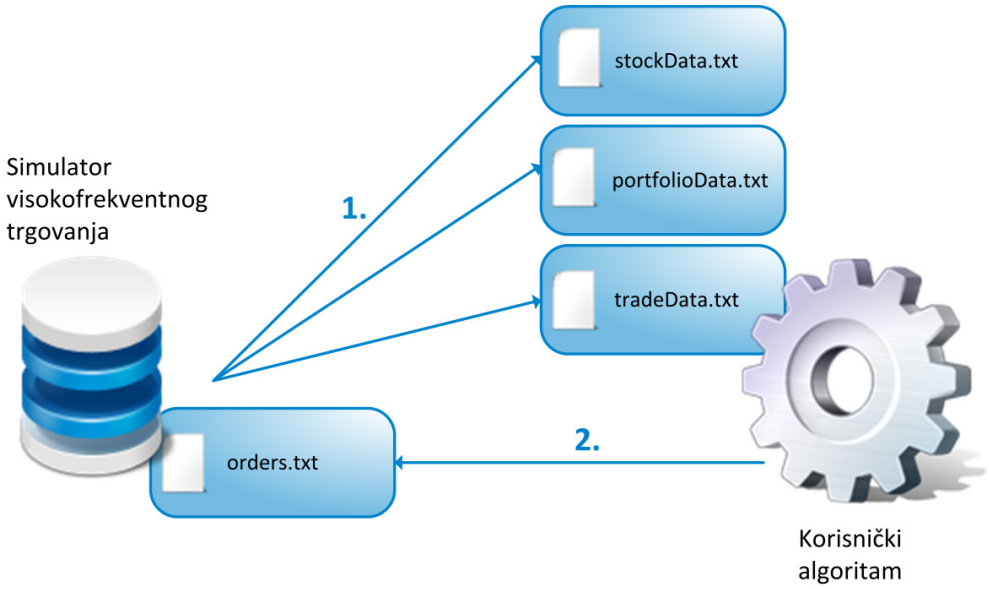
Logger je servis koji nam omogućuje organizaciju podataka o trenutnom stanju na tržištu u kompaktan oblik te spremanje tog sadržaja u odgovarajuće datoteke. Ovaj servis sadrži metode koje omogućuju korisniku da učita povijesne podatke o tržištu koje je naš program prikupio do tog trenutka. Popis metoda je slijedeći:

- Log(Market inMarket, string appPath): je najvažnija metoda ovog servisa. Ona preko ulaznog parametra prima agregat Market koji sadrži sve informacije o trenutnom stanju na tržištu. Te informacije se organiziraju i zapisuju u datoteke na disku.
- GetAvailableDates(string appPath): metoda vraća listu svih datuma za koje postoje pohranjene informacije o tržištu.
- GetAvailableTickers(string appPath, string inDate): metoda vraća listu svih oznaka dionica za koje postoje pohranjene informacije na dan zadat ulaznim parametrom inDate.
- GetTrades(string appPath, string inDate): metoda preko referenci vraća dvije strukture podataka koje sadrže informacije o vrijednostima i

volumenima svih pojedinačnih transakcija koje su se dogodile na dan zadat ulaznim parametrom inDate.
- GetData(string appPath, string inDate, string inTicker): metoda preko referenci vraća nekoliko struktura podataka koje sadrže informacije o cijenama dionice, određene ulaznim parametrom inTicker, tijekom dana određenog ulaznim parametrom inDate.

Serializer je servis koji pruža korisniku sučelje pomoću kojeg je moguće na jednostavan način perzistirati bilo koji objekt koji je izveden iz ISerializable sučelja. Ovaj razred ima dvije metode:
- SerializeObject(string fileName, Object obj): metoda perzistira objekt u datoteku zadanu putanjom fileName.
- DeserializeObject(string fileName): metoda otvara datoteku zadanu putanjom fileName te iz nje čita pohranjeni objekt i prebacuje ga direktno u radnu memoriju.

Interfacer je servis koji se brine o razmjeni podataka između glavnog programa i simulacija napisanih u programskom jeziku MATLAB. Ova komunikacija je dvosmjerna. Glavni program netom prije svakog pokretanja pojedine simulacije priprema tri tekstualne datoteke koje simulacija može koristiti prilikom svog izvođenja. Navedene datoteke redom sadrže sve trenutno dostupne podatke o svim dionicama koje se nalaze unutar indeksa CROBEX, podatke o svim transakcijama izvršenim tog radnog dana te podatke o trenutnom stanju portfelja investitora. Drugi smjer komunikacije se odvija po završetku simulacije. Tada glavni program provjerava da li je simulacija postavila naloge za izvršenje te ako je onda ih učitava u radnu memoriju. Komunikacija između simulatora visokofrekventnog trgovanja i algoritma simulacije je ilustrirana na Slici 7. Metode ovog servisa su slijedeće:
- WriteStockData(List<Stock> inStocks): metoda stvara prvu tekstualnu datoteku koja sadrži podatke o dionicama unutar CROBEX-a koji su trenutno važeći na tržištu.
- WriteTradeData(List<Trade> inTrades): metoda stvara drugu tekstualnu datoteku koja sadrži podatke o svim transakcijama obavljenim od početka radnog dana.
- WritePortfolioData(Portfolio portfolio): metoda stvara treću tekstualnu datoteku koja sadrži podatka o portfelju investitora (gotovina, udjeli u dionicama).
- ReadOrders(string fileName): učitava u radnu memoriju glavnog programa sve naloge koje je zadala simulacija tijekom svog izvođenja.



Slika 7. Komunikacija između glavnog programa i algoritma simulacije

Matlab servis je namijenjen pokretanju MATLAB skripta. On prikriva tehničke detalje poziva od korisnika te mu omogućuje da pozivom jedne metode pokrene skriptu. Također ovaj servis služi i za kreiranje MATLAB skripta koje služe za definiranje struktura podataka te učitavanje podataka u simulaciju. Metode koje korisnik ima na raspolaganju su:

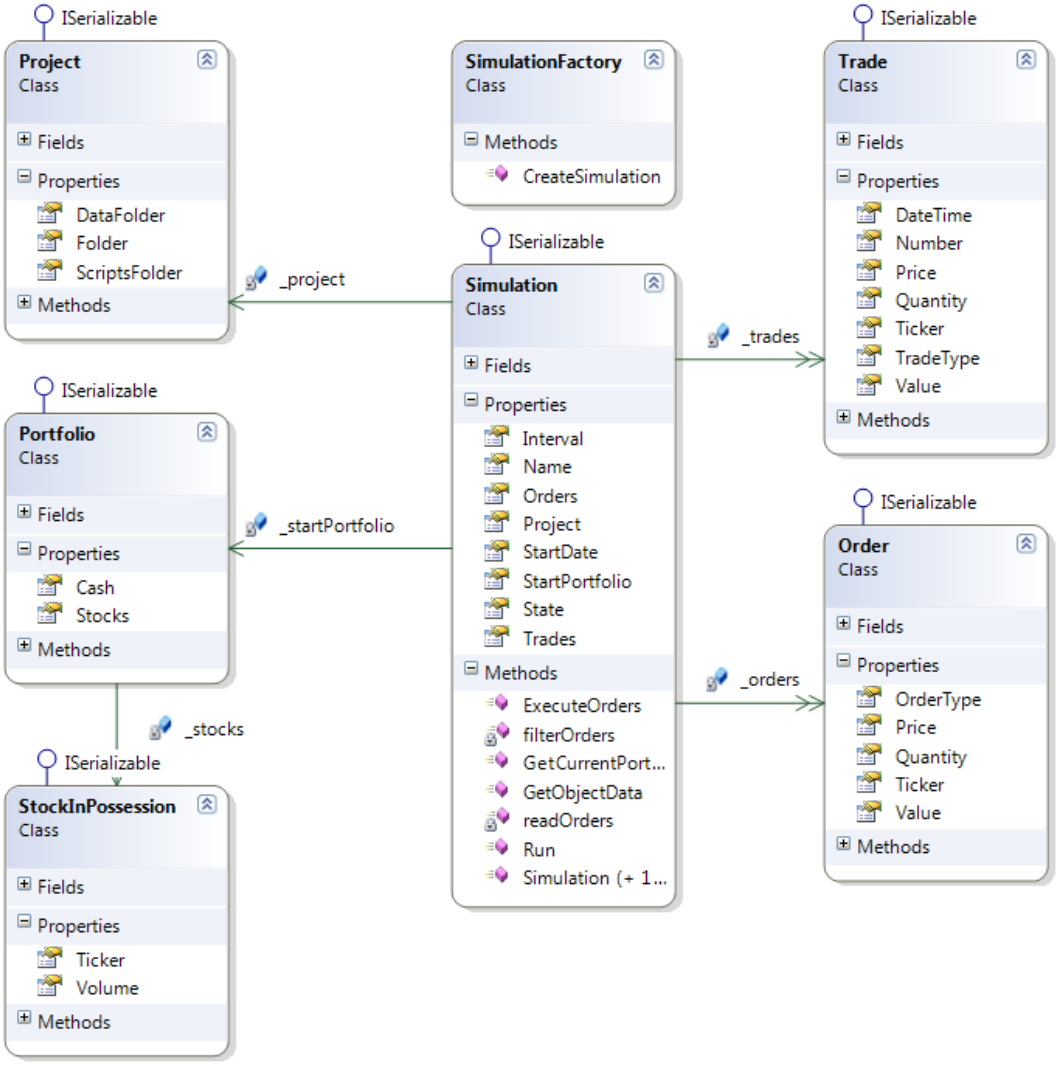
- Run(string scriptsFolder): metoda pokreće MATLAB engine, pozicionira se unutar direktorija u kojem se nalaze skripte te poziva predefiniranu skriptu loadData.m koja učitava podatke o tržištu u radni prostor (engl. *Workspace*), simulacije. Kad su sve predradnje obavljene poziva se centralna skripta simulacije main.m. Ukoliko pri izvođenju ove skripte dođe do pogreške, simulacija se zaustavlja.
- CreateScripts(string scriptsFolder): kreira predefinirane MATLAB skripte unutar direktorija zadanog ulaznim parametrom scriptsFolder.

4.5 Serijalizacija

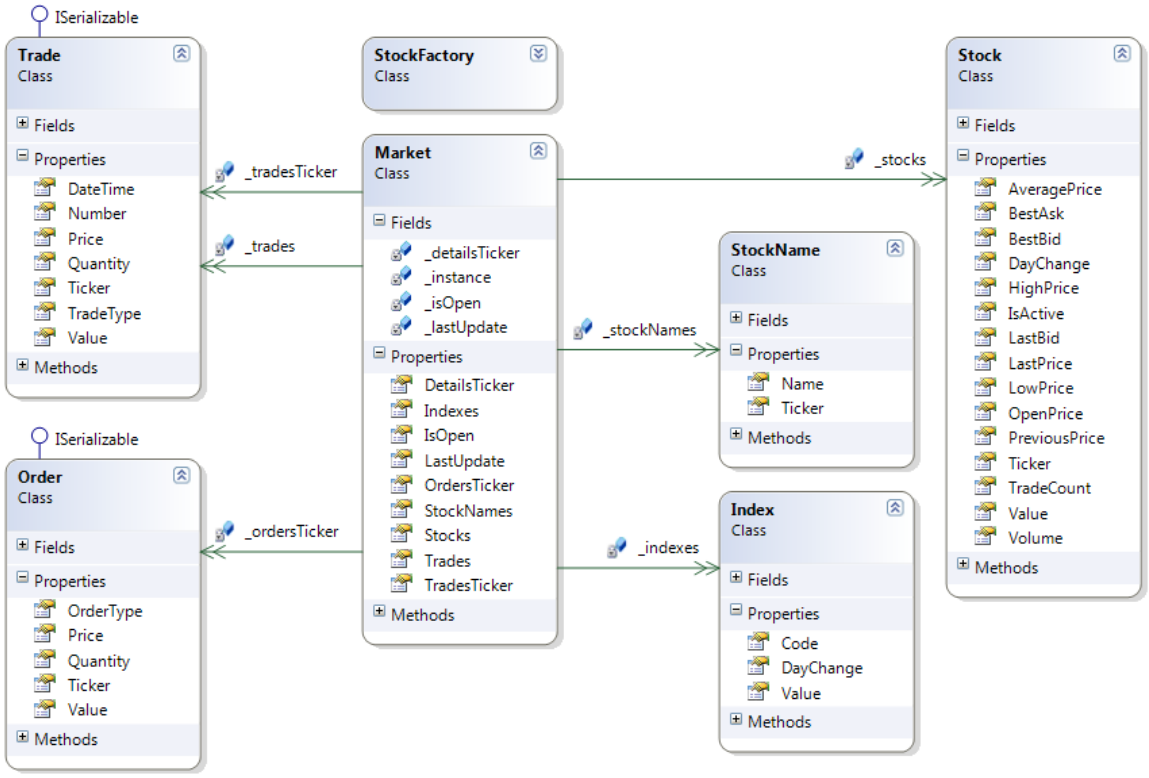
Serijalizacija (engl. *Serialization*), je postupak kojim omogućavamo da strukture podataka te razni drugi objekti budu pretvoreni u format kojeg se može pohraniti na tvrdi disk. Za vrijeme rada programa svi objekti su privremeno pohranjeni u radnoj memoriji. Međutim, završetkom rada programa svi ti objekti se brišu iz radne memorije te se nepovratno gube. Mi želimo osigurati da podaci potrebni za funkcionalan rad programa ostanu sačuvani te da pri svakom novom pokretanju programa ti isti podaci budu dostupni. Primijetimo da mnoštvo objekata u programu inicijaliziramo pomoću podataka koje dohvaćamo u stvarnom vremenu preko web-servisa. Te objekte nije potrebno serijalizirati jer nas ne zanimaju

vrijednosti podataka koje su oni sadržavali u trenutku kada je korisnik zadnji put ugasio program, već želimo da oni sadrže svježe podatke koje su dostupni u svakom trenutku putem web servisa. Stoga, imajući u vidu model domene, zaključujemo da je potrebno serijalizirati sve objekte tipa Simulation te sve objekte koje razred Simulation agregira. Razredi koje moramo trajno pohraniti su: Simulation, Project, Portfolio i StockInPossesion. Kako bi omogućili postupak serijalizacije, nabrojeni razredi moraju implementirati sučelje ISerializable. Dodatno, potrebna nam je i klasa Serializer koja će imati metode za spremanje pojedinačnih objekata na tvrdi disk te za vraćanje istih u radnu memoriju. Detaljnije, o ovoj klasi čitatelj može pronaći u odjeljku 4.4.

U nastavku prilažemo dvije slike na kojima su grafički prikazani dijagrami razreda koji odgovaraju opisanom programskom modelu. Na prvoj slici prikazani su svi razredi asocirani s agregatom Simulation, dok su na drugoj slici prikazani svi razredi asocirani s agregatom Market.



Slika 8. Dijagram razreda asociranih s agregatom Simulation



Slika 9. Dijagram razreda asociranih s agregatom Market

5 Arhitektura i opis razvijenog sustava

Sustav za visokofrekventno trgovanje na Zagrebačkoj burzi se sastoji od glavnog programa i četiri pomoćna modula koji se oslanjaju na programski model domene, opisan u prethodnom poglavlju. Nakon što se korisnik uspješno prijavi na sustav pomoću svog korisničkog imena i lozinke, pokreće se glavni program. Glavni program u pravilnim intervalima od deset sekundi poziva modul za prikupljanje podataka o trenutnom stanju Zagrebačke burze. Modul za prikupljanje podataka šalje HTTP zahtjeve u JSON formatu prema web servisu brokerske kuće Agram Brokeri. Kada dobije povratnu poruku u JSON formatu, parsira je sprema informacije u odgovarajuće razrede iz modela domene. Kako je model domene zajednički za sve module, glavni program sada pristupa svježim podacima te ih prikazuje na zaslonu. Istovremeno, modul za trajnu pohranu podataka, organizira primljene podatke u kompaktan oblik te ih serijalizira na disk.

Korištenjem modula za simulaciju visokofrekventnog trgovanja, korisnik programa može stvoriti novu simulaciju te podesiti njene parametre. Kada je nova simulacija stvorena, potrebno je strategiju trgovanja zapisati u datoteku main.m koristeći programski jezik MATLAB. Prije, nego pokrene simulaciju algoritma visokofrekventnog trgovanja, korisnik može testirati strategiju na povijesnim podacima. Za testiranje je odgovoran poseban modul koji dohvaća povijesne podatke o stanju na tržištu koje je dotada naša aplikacija prikupila te pokreće virtualnu simulaciju nad tim podacima. Nakon što se je uvjerio u ispravnost rada svoje strategije za trgovanje, korisnik može aktivirati simulaciju koja se tada izvršava u pravilnim intervalima, koristeći realne podatke o stanju na Zagrebačkoj burzi.

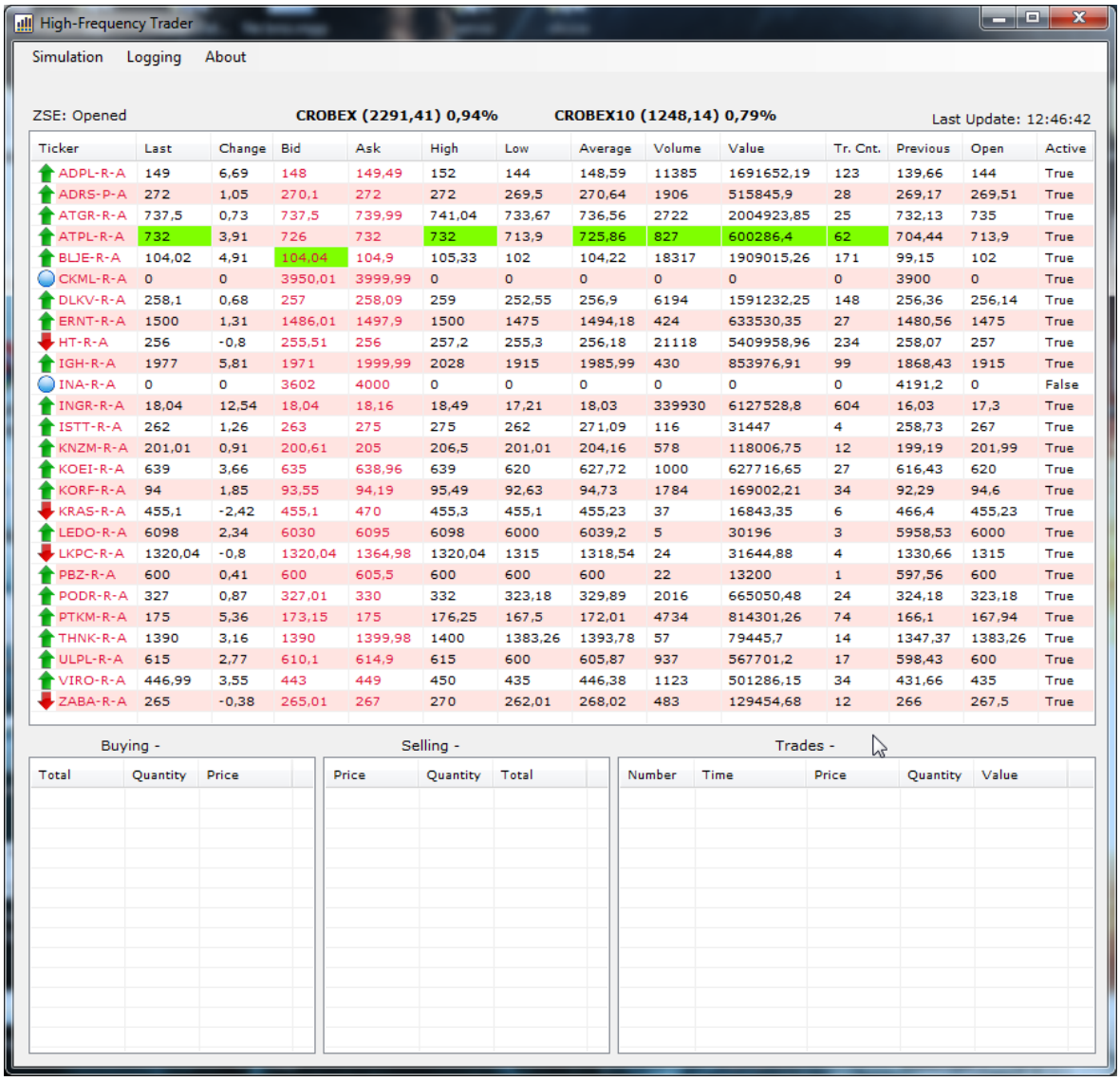
U nastavku ćemo podrobnije opisati glavni program i pripadne pomoćne module čija je međusobna ovisnost prikazana slikom na slijedećoj stranici.

Slika 10. Glavni program i njegovi pomoćni moduli

5.1 Glavni program

Prilikom pokretanja glavnog programa, od korisnika će se zatražiti korisničko ime i lozinka koje je dobio od Agram Brokera. Ako su uneseni podaci ispravni, korisnik dobiva pristup glavnom programu. Glavni program je zadužen za prikaz podataka sa Zagrebačke burze u stvarnom vremenu te za pokretanje i koordinaciju pomoćnih modula sustava.

Glavni program svakih deset sekunda šalje zahtjeve modulu za prikupljanje podataka te prikazuje dobivene podatke. Prikazuju se svi podaci o dionicama koje se nalaze unutar CROBEX indeksa te vrijednosti i dnevne promjene CROBEX i CROBEX10 indeksa. Dodatno se prikazuje vrijeme kada se dogodila posljednja transakcija, odnosno promjena cijene te informacija o tome da li je Zagrebačka burza otvorena ili zatvorena za trgovanje. Sve novonastale promjene se označavaju zelenom bojom zbog lakše uočljivosti. Na slijedećoj slici se može vidjeti kako izgleda prikaz podataka u glavnom programu.



Ticker	Last	Change	Bid	Ask	High	Low	Average	Volume	Value	Tr. Cnt.	Previous	Open	Active
ADPL-R-A	149	6,69	148	149,49	152	144	148,59	11385	1691652,19	123	139,66	144	True
ADRS-P-A	272	1,05	270,1	272	272	269,5	270,64	1906	515845,9	28	269,17	269,51	True
ATGR-R-A	737,5	0,73	737,5	739,99	741,04	733,67	736,56	2722	2004923,85	25	732,13	735	True
ATPL-R-A	732	3,91	726	732	732	713,9	725,86	827	600286,4	62	704,44	713,9	True
BLJE-R-A	104,02	4,91	104,04	104,9	105,33	102	104,22	18317	1909015,26	171	99,15	102	True
CKML-R-A	0	0	3950,01	3999,99	0	0	0	0	0	0	3900	0	True
DLKV-R-A	258,1	0,68	257	258,09	259	252,55	256,9	6194	1591232,25	148	256,36	256,14	True
ERNT-R-A	1500	1,31	1486,01	1497,9	1500	1475	1494,18	424	633530,35	27	1480,56	1475	True
HT-R-A	256	-0,8	255,51	256	257,2	255,3	256,18	21118	5409958,96	234	258,07	257	True
IGH-R-A	1977	5,81	1971	1999,99	2028	1915	1985,99	430	853976,91	99	1868,43	1915	True
INA-R-A	0	0	3602	4000	0	0	0	0	0	0	4191,2	0	False
INGR-R-A	18,04	12,54	18,04	18,16	18,49	17,21	18,03	339930	6127528,8	604	16,03	17,3	True
ISTT-R-A	262	1,26	263	275	275	262	271,09	116	31447	4	258,73	267	True
KNZM-R-A	201,01	0,91	200,61	205	206,5	201,01	204,16	578	118006,75	12	199,19	201,99	True
KOEI-R-A	639	3,66	635	638,96	639	620	627,72	1000	627716,65	27	616,43	620	True
KORF-R-A	94	1,85	93,55	94,19	95,49	92,63	94,73	1784	169002,21	34	92,29	94,6	True
KRAS-R-A	455,1	-2,42	455,1	470	455,3	455,1	455,23	37	16843,35	6	466,4	455,23	True
LEDO-R-A	6098	2,34	6030	6095	6098	6000	6039,2	5	30196	3	5958,53	6000	True
LKPC-R-A	1320,04	-0,8	1320,04	1364,98	1320,04	1315	1318,54	24	31644,88	4	1330,66	1315	True
PBZ-R-A	600	0,41	600	605,5	600	600	600	22	13200	1	597,56	600	True
PODR-R-A	327	0,87	327,01	330	332	323,18	329,89	2016	665050,48	24	324,18	323,18	True
PTKM-R-A	175	5,36	173,15	175	176,25	167,5	172,01	4734	814301,26	74	166,1	167,94	True
THNK-R-A	1390	3,16	1390	1399,98	1400	1383,26	1393,78	57	79445,7	14	1347,37	1383,26	True
ULPL-R-A	615	2,77	610,1	614,9	615	600	605,87	937	567701,2	17	598,43	600	True
VIRO-R-A	446,99	3,55	443	449	450	435	446,38	1123	501286,15	34	431,66	435	True
ZABA-R-A	265	-0,38	265,01	267	270	262,01	268,02	483	129454,68	12	266	267,5	True

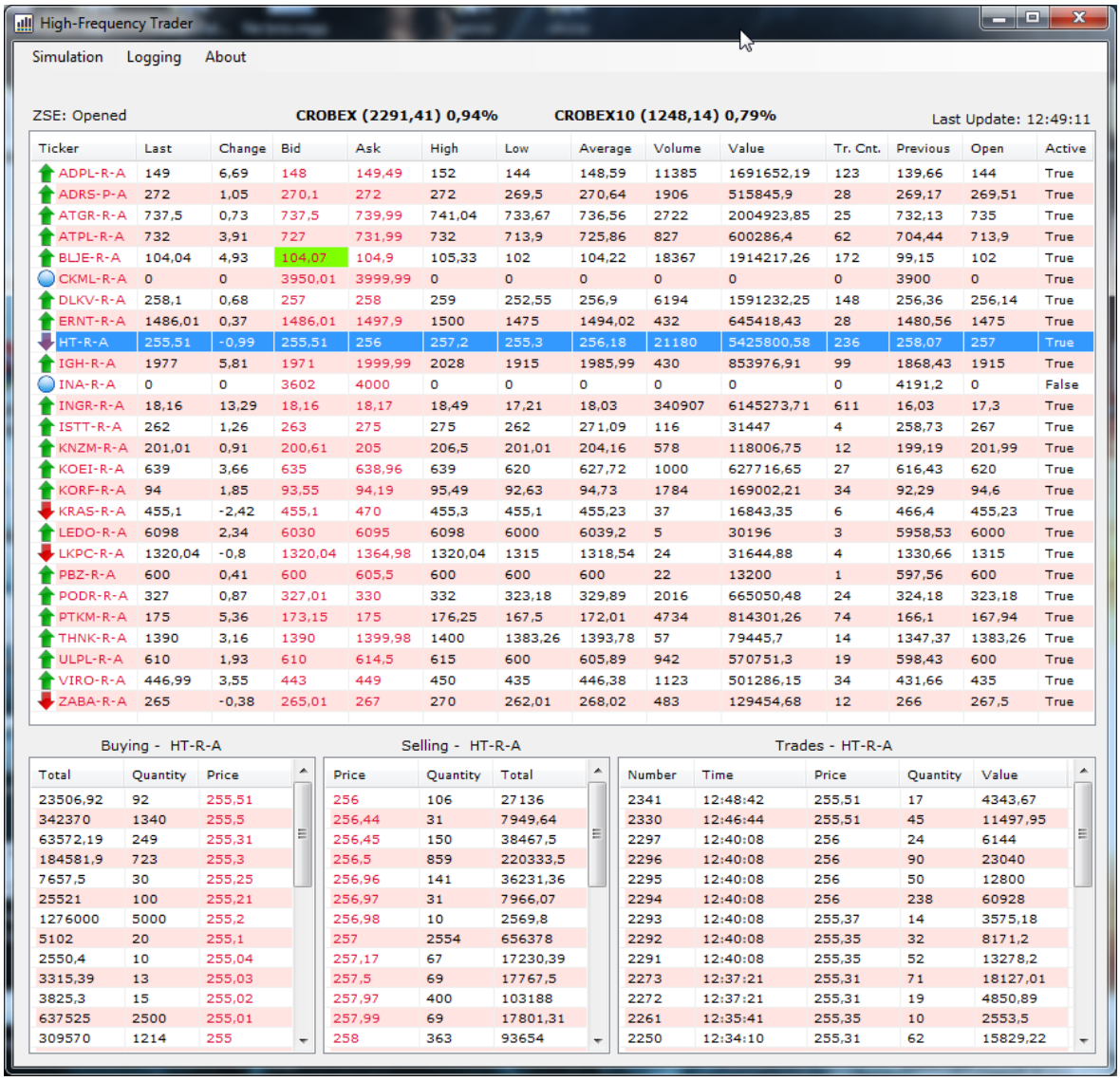
Total	Quantity	Price

Price	Quantity	Total

Number	Time	Price	Quantity	Value

Slika 11. Prikaz podataka u glavnom programu

Klikom na redak pojedine dionice, korisnik može zatražiti i detaljniji prikaz podataka za određenu dionicu. Detaljniji prikaz obuhvaća ispis svih aktivnih naloga za kupnju i prodaju te ispis trgovanja ostvarenih tog dana. U tom trenutku glavni program šalje poseban zahtjev modulu za prikupljanje podataka koji postavlja odgovarajuće parametre zahtjeva u JSON formatu (detaljnije u odjeljku 3.3). Nakon što dobije zatražene podatke, glavni program ih prikazuje u donjem dijelu prozora. Slijedeća slika prikazuje detaljne podatke za dionicu HT-a.



Slika 12. Detaljniji prikaz podataka

5.2 Modul za prikupljanje podataka u stvarnom vremenu

Modul za prikupljanje podataka u stvarnom vremenu je odgovoran za dohvaćanje informacija o stanju na Zagrebačkoj burzi. Ovaj modul, ovisno o vrsti zahtjeva kojeg dobije od glavnog programa, generira odgovarajuću poruku u JSON formatu. Također, on se brine i za postavljanje varijabli zaglavlja HTTP zahtjeva. HTTP zahtjevi se šalju prema web servisima brokerske kuće Agram Brokeri, pri čemu se konstantno brine o autorizaciji korisnika.

Poruke koje modul primi od web servisa variraju po svojoj veličini i kompleksnosti. Međutim, za svaku primljenu poruku ponavlja se identičan postupak. Iz zaglavlja primljene poruke potrebno je odrediti način kodiranja tijela poruke. Tijelo poruke se potom dekodira, čime se dobije poruka u JSON formatu. Poruka u JSON formatu

se mora isparsirati te se dobivenim podacima popune strukture podataka u modelu domene. Jednom kad je model domene osvježen novim podacima, podaci su dostupni glavnom programu i svim pomoćnim modulima.

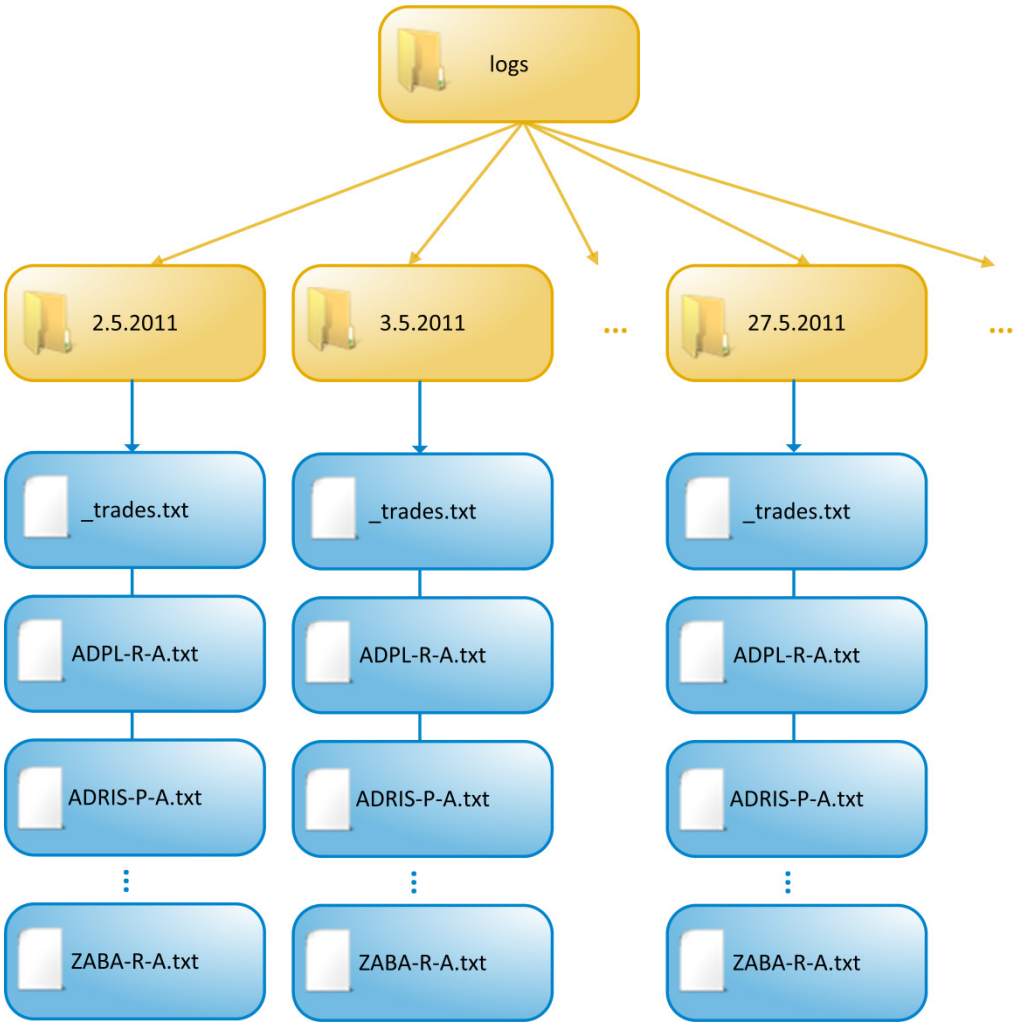
5.3 Modul za trajnu pohranu podataka

Postavlja se pitanje je li potrebno trajno pohranjivati podatke o stanju na burzi? Za najjednostavnije simuliranje rada algoritama visokofrekventnog trgovanja nije potrebno trajno pohranjivati podatke, ali postoji više razloga zbog čega bi to bilo itekako korisno.

Relativno jednostavno je doći do povijesnih podataka o zaključnim dnevnim cijenama dionica, ali kad je riječ o visokofrekventnim signalima, odnosno o cijenama dionica unutar dana stvari se kompliciraju. Naime takvi podaci, nisu dostupni na burzama slabije razvijenih zemalja, a na naprednijim burzama oni se plaćaju, pri čemu im cijena doseže iznose od nekoliko tisuća eura. Ti iznosi nisu preveliki za velike investicijske banke i fondove, ali za male korisnike predstavljaju veliku prepreku. Stoga je ovaj modul izuzetno koristan za male investitore, istraživače i entuzijaste.

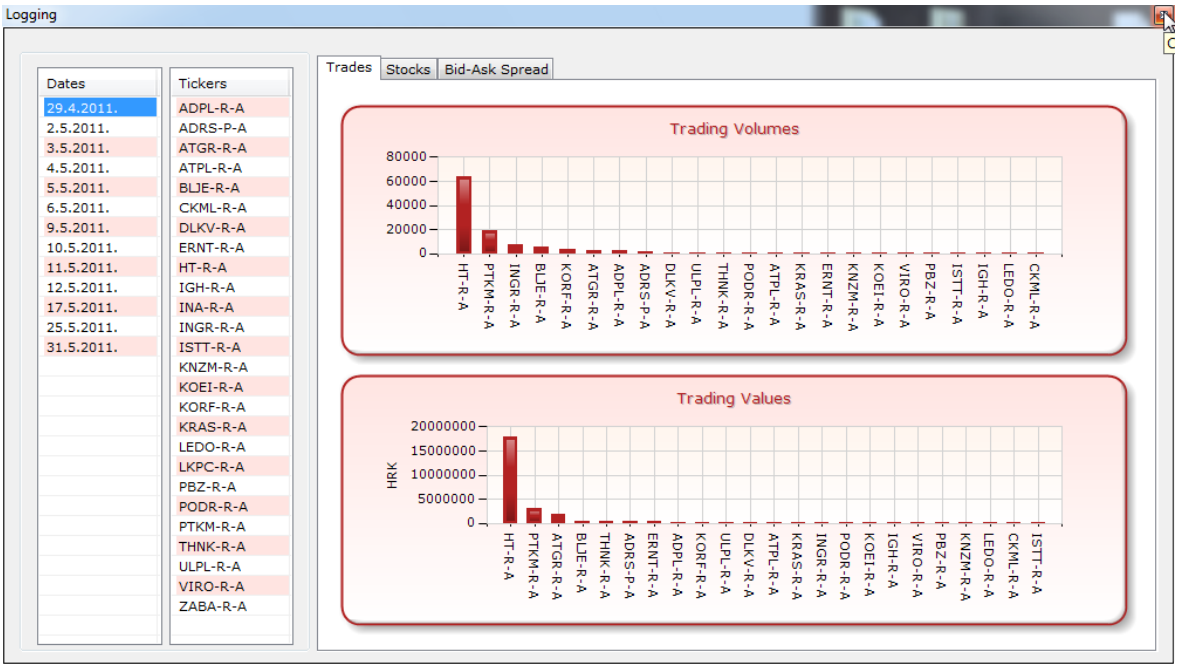
Možemo izdvojiti nekoliko mogućih primjena podataka koji su dobiveni na ovaj način. Visokofrekventne podatke je moguće iskoristiti za testiranje ostvarene strategije visokofrekventnog trgovanja (vidi odjeljak 5.5). Također, ovi podaci se mogu iskoristiti i izvan ovog programa za razna istraživanja. Jednostavan primjer je kad bi se na većim vremenskim periodima izračunali parametri koji bi se poslije mogli iskoristiti kao konstante u algoritmu za visokofrekventno trgovanje.

Pošto korisnik može koristiti prikupljene podatke i izvan samog programa, posebna briga se vodila o načinu organizacije podataka kako bi se omogućio što lakši pristup istima. Korisnik, može napisati jednostavnu skriptu kojom bi brzo došao do podataka koje traži. Stoga ćemo ukratko opisati datotečnu organizaciju prikupljenih podataka. Svi podaci se nalaze u direktoriju logs koji se može pronaći u istom direktoriju gdje se nalazi izvršna aplikacija. Unutar direktorija logs, program stvara direktorije za svaki dan kada se je trgovalo na Burzi te ih naziva po datumu. U direktoriju koji pripada određenom datumu može se pronaći tekstualna datoteka _trades.txt koja u svakom retku sadrži podatke o obavljenim trgovanjima i to vremenskim redom kako su se ona dogodila. Osim ove tekstualne datoteke, za svaku dionicu unutar CROBEXA se može pronaći zasebna datoteka koja je također organizirana po recima, pri čemu svaki slijedeći redak sadrži podatke o dionici s vremenskim odmakom od deset sekunda. Hijerarhija organizacije prikupljenih podataka je ilustrirana slijedećom slikom.



Slika 13. Datotečna organizacija prikupljenih podataka

Dodatno, ovaj modul može vizualno prikazati dotada prikupljene podatke. Moguće je za određeni datum prikazati silazno sortirane dionice prema ukupnom volumenu trgovanja i ukupnoj vrijednosti trgovanja tog dana, (Slika 14). Ovaj prikaz je pogodan kako bi se brzo odredile najlikvidnije dionice na Burzi što možemo iskoristiti pri izradi algoritma statističke arbitraže (vidi odjeljak 6.3). Također, moguće je grafički prikazati apsolutnu i relativnu promjenu cijene (Slika 15), te odnos ponude i potražnje (engl. *Bid-ask spread*) za pojedinu dionicu unutar odabranog dana (Slika 16).



Slika 14. Sortirani prikaz prema ukupnom volumenu i vrijednosti trgovanja

Slika 15. Vizualni prikaz apsolutne i relativne cijene određene dionice

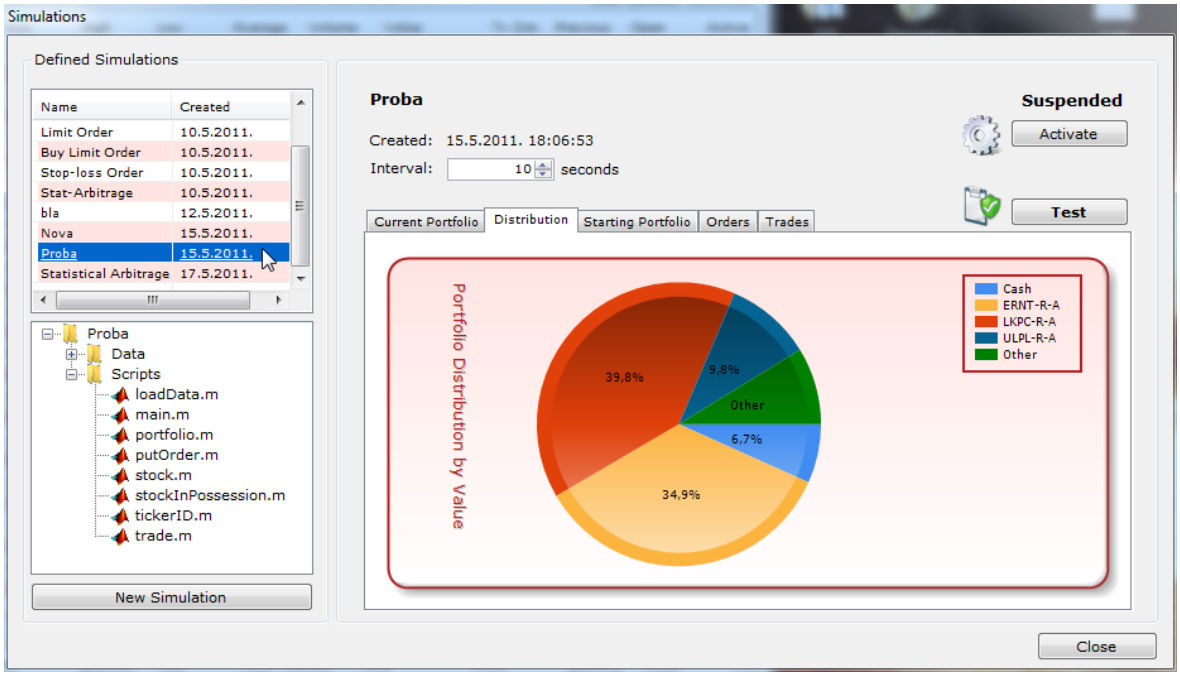
Slika 16. Vizualni prikaz odnosa ponude i potražnje za određenu dionicu

5.4 Modul za simulaciju visokofrekventnog trgovanja

Unutar modula za simulaciju visokofrekventnog trgovanja, korisnik ima mogućnost definirati novu simulaciju. Tom prilikom, korisnik zadaje ime simulacije, direktorij u kojem će biti smještene sve datoteke vezane za simulaciju te interval simulacije. Također, može se zadati i početno stanje portfelja, što uključuje iznos gotovine i dionice koje korisnik posjeduje.

Nakon što je simulacija kreirana, ona se pojavljuje na popisu simulacija. Važno je napomenuti da korisnik može imati više simulacija koje se mogu izvršavati istodobno. Odabirom jedne od simulacija s popisa pojavljuje nam se datotečno stablo s glavnim direktorijem nazvanim po imenu definirane simulacije. Unutar glavnog direktorija, nalaze se poddirektoriji Data i Scripts. U poddirektoriju Data nalaze se tekstualne datoteke pomoću kojih se odvija dvosmjerna razmjena podataka između glavnog program i algoritma simulacije, kao što je prethodno opisano u odjeljku 4.3. U poddirektoriju Scripts nalaze se MATLAB skripte, funkcije i definicije struktura podataka. Skripta load.m se poziva na početku svakog pokretanja simulacije te učitava podatke o stanju tržišta koje je dobila od glavnog programa. Učitani podaci se spremaju u odgovarajuće strukture podataka. Svoju strategiju visokofrekventnog trgovanja korisnik mora ostvariti u programskom jeziku MATLAB te kopirati izvorni kod u datoteku main.m. Ta datoteka se pokreće nakon inicijalnog učitavanja podataka i predstavlja jezgru izvršavanja simulacije. Detaljnije upute i primjere implementacije algoritama u MATLABU čitatelj može pronaći u slijedećem odjeljku 5.4.1. Nakon što implementira svoju strategiju

trgovanja, korisnik može promijeniti interval pokretanja simulacije koji je početno postavljen na deset sekunda. Korisnik, u svakom trenutku može pokrenuti, odnosno zaustaviti simulaciju. Dostupne su informacije o početnom i trenutnom portfelju, grafički prikaz distribucije vrijednosti unutar portfelja, aktivni nalozi koje je zadao algoritam te izvršena trgovanja, Slika 17.



Slika 17. Prikaz modula za simulaciju visokofrekventnog trgovanja

5.4.1 Implementiranje algoritama u MATLABU

Pri implementiranju algoritama u visokofrekventnom trgovanju, korisnik može koristiti sve MATLAB-ove ugrađene funkcije što je zgodno zbog različitih matematičkih izračuna. Izvorni kod algoritma mora biti napisan u datoteci main.m, pri čemu je dozvoljeno da glavna skripta poziva skripte koje su napisane u drugim datotekama. Ono što se od algoritma očekuje kao izlaz su nalozi za trgovanje. Nalozi mogu biti za kupnju ili prodaju, a zadaje ih se isključivo pomoću predefinirane funkcije putOrder.

Prototip funkcije: `putOrder(orderType, ticker, price, quantity)`

- orderType: može poprimiti dvije vrijednosti 'Buy' ako je nalog za kupnju i 'Sell' ako je nalog za prodaju. Za sve ostale vrijednosti, nalog će se zanemariti.
- ticker: oznaka dionice koju želimo kupiti ili prodati
- price: cijena koju nudimo, odnosno tražimo za jednu dionicu
- quantity: cijeli broj udjela koje želimo kupiti ili prodati

Na primjer ako želimo kupiti 20 dionica HT-a po cijeni od 255.5 HRK tada funkciju pozivamo na slijedeći način:

```
putOrder('Buy', 'HT-R-A', 255.5, 20);
```

Algoritam pri svakom izvršavanju ima pristup najsvježijim podacima sa Zagrebačke burze. Moguće je pristupiti podacima o svim dionicama unutar CROBEX-a te svim ostvarenim transakcijama od početka radnog dana. Dodatno, moguće je pristupiti podacima o trenutnom stanju korisničkog portfelja. Ovi podaci se učitavaju u radni prostor prije pokretanja glavne skripte.

U radnom prostoru se nalazi lista stocks koja sadrži objekte istancirane iz razreda stock. Ti objekti sadrže sve informacije o nekoj dionici. Popis svojstava razreda stock može se pronaći na početku datoteke stock.m. Funckija tickerID je pomoćna funkcija pomoću koje pristupamo određenoj dionici unutar liste stocks preko oznake dionice. Ako želimo dobiti objekt s podacima o HT-u, tada ćemo to učiniti ovako:

```
ht = stocks{tickerID('HT-R-A', tickers)};
```

Pristupanje svojstvima objekata je jednostavno, na primjer, ako želimo izračunati razliku između ponude i potražnje za HT, tada to činimo ovako:

```
bidAskSpread = ht.BestAsk - ht.BestBid;
```

Ostvarenim trgovanjima možemo pristupiti preko objekata koji opisuju dionicu jer su trgovanja grupirana po dionicama. Koristimo svojstvo Trades, razreda stock, koje nam daje vremenski sortiranu listu svih trgovanja pojedine dionice. Npr. ako želimo doznati sva trgovanja dionicama HT-a tog radnog dana tada ćemo to učinit ovako:

```
htTrades = ht.Trades;
```

Svaki objekt koji opisuje trgovanje ima sljedeća svojstva:

- Ticker: oznaka dionice kojom se trgovalo.
- Time: vrijeme kada je transakcija obavljena.
- Price: cijena po kojoj se izvršila transakcija.
- Quantity: količina dionica koja je sudjelovala u transakciji.
- Value: ukupna vrijednost transakcije.

Preostaje nam još opisati kako pristupiti informacijama o korisničkom portfelju. Te informacije su dostupne preko objekta myPortfolio koji se nalazi u radnom prostoru. Ovaj objekt ima svojstvo Cash kojim se može saznati iznos gotovine koju korisnik ima na računu te ima svojstvo Stocks kojim se dobija lista svih dionica koje korisnik posjeduje. Dodatno, objekt ima metodu FindStock preko koje

možemo dohvatiti informacije o dionici u posjedu preko njene oznake. Ako dionica s navedenom oznakom ne postoji tada metoda vraća nulu. Pojasnimo to na primjeru u kojem želimo provjeriti da li na računu imamo manje od 1000 kuna te ako je tako, onda se odlučujemo za prodaju svih dionica HT-a koje posjedujemo po cijeni od 255.5 HRK.

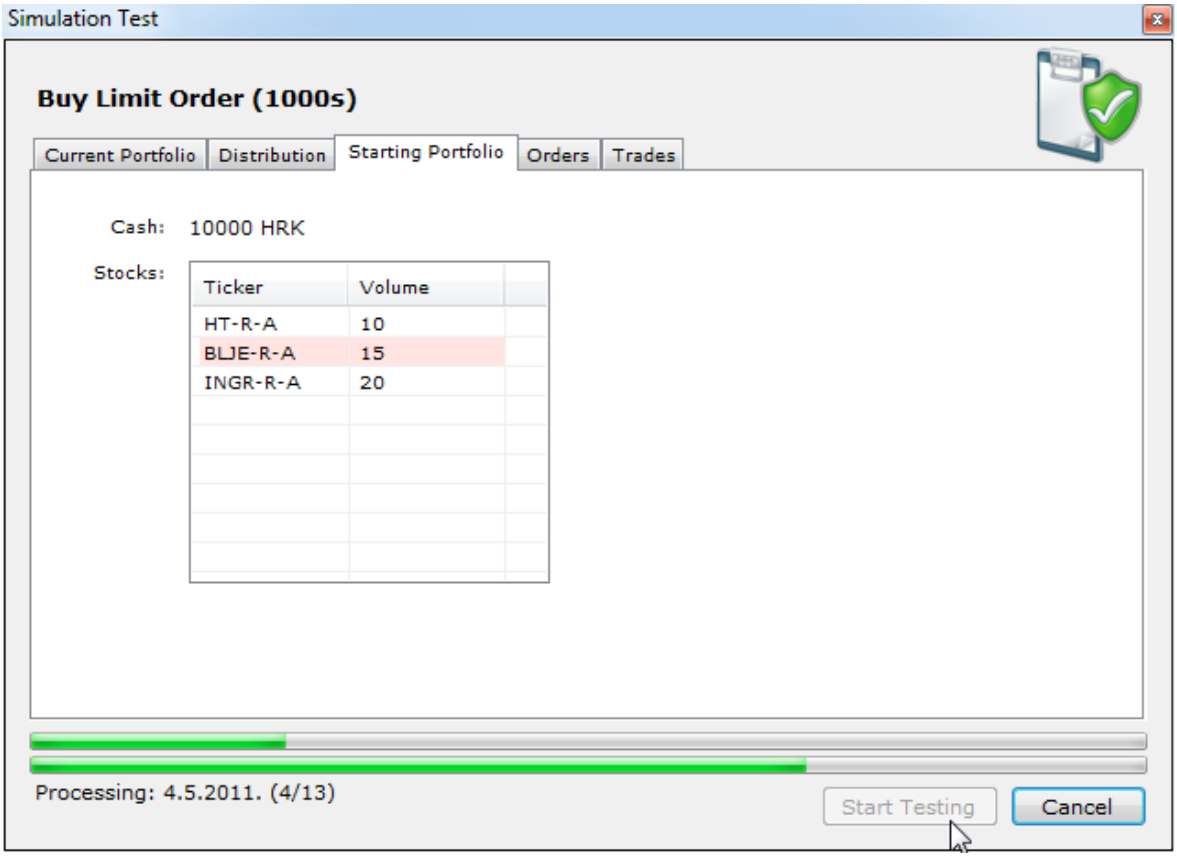
```
if (myPortfolio.Cash < 1000)
   myHT = myPortfolio.FindStock('HT-R-A');

   if (myHT ~= 0)
      putOrder('Sell', myHT.Ticker, 255.5, myHT.Volume);
   end
end
```

Zainteresirani čitatelj može pronaći dodatne primjere funkcionalnih algoritama napisanih u MATLABU i prilagođenih za rad s našim sustavom u poglavlju 6, o algoritmima za visokofrekventno trgovanje.

5.5 Modul za testiranje korisničkih algoritama

Kad korisnik implementira svoju strategiju visokofrekventnog trgovanja u programskom jeziku MATLAB, može je testirati nad povijesnim podacima koje je program dotada prikupio. Testiranje se obavlja na identičan način kao i simulacija u realnom vremenu. Poštuje se vremenski interval izvršavanja koji je zadao korisnik. Korisnik ima uvid u sve naloge koje je algoritam zadao, te sve transakcije koje su se obavile. Također, korisnik ima uvid u početni portfelj te u stanje portfelja koji bi se dobio da se nije radilo o testiranju već o stvarnim podacima. Modul prikazuje koliko testova treba još obaviti, a ukoliko dođe do pogreške u izvođenju korisničke strategije, modul dojavljuje pogrešku i zaustavlja testiranje. Prikaz rada modula za testiranje vidljiv je na Slici 18.



Slika 18. Prikaz rada modula za testiranje

6 Algoritmi visokofrekventnog trgovanja

U ovom poglavlju osvrnut ćemo se na složene vrste naloga te dati njihova konkretna ostvarenja, prilagođena za rad s našim programom. Također, objasnit ćemo matematičku podlogu algoritma statističke arbitraže te ponuditi našu implementaciju.

6.1 Nalozi

Najjednostavniji nalog kojeg broker može zadati je tzv. tržišni nalog, (engl. *Market order*). To je nalog koji zahtijeva da se transakcija izvrši odmah i to po trenutno najpovoljnijoj cijeni na tržištu. Na Zagrebačkoj burzi, moguće je zadavati isključivo tržišne naloge, dok na stranim tržištima postoje razne vrste naloga. Opisat ćemo najčešće korištene vrste naloga.

Nalog s ograničenjem, (engl. *Limit order*), određuje graničnu cijenu. Nalog se izvršava u trenutku kada cijena dosegne graničnu cijenu ili cijenu koja je povoljnija za investitora. Dakle, ako je investitor, koji želi kupiti određenu dionicu, postavio graničnu cijenu na 30 HRK, nalog će biti izvršen po cijeni od 30 HRK ili manje. Naravno, ne postoji nikakva garancija da će se zadani nalog ikad izvršiti jer granična cijena može zauvijek ostati nedosegnuta.

Nalog za zaustavljanje, (engl. *Stop order or stop-loss order*), određuje zaustavnu cijenu. Nalog se izvršava po najboljoj cijeni na tržištu u trenutku kada se kupnja ili prodaja izvrše po zadanoj graničnoj cijeni ili po cijeni koja je još nepovoljnija za investitora. Pretpostavimo da je zadan nalog za zaustavljanje da se dionica proda po cijeni od 30 HRK u trenutku kada je njena tržišna cijena jednaka 35 HRK. Nalog će se izvršiti samo ako cijena dionice padne na 30 HRK ili manje. Svrha ove vrste naloga je da se zatvori pozicija (engl. *Close out a position*), ukoliko dođe do nepovoljnog kretanja cijene dionice, odnosno da se ograniči mogući gubitak.

Nalog za zaustavljanje s ograničenjem, (engl. *Stop-limit order*), je svojevrsna kombinacija naloga za zaustavljanje i naloga s ograničenjem. Ova vrsta naloga se pretvara u nalog s ograničenjem u trenutku kada je kupnja ili prodaja obavljena po cijeni jednakoj ili nepovoljnijoj od zaustavne cijene. Za ovu vrstu naloga moraju biti zadane dvije cijene, zaustavna i granična cijena. Pretpostavimo da je zadan nalog za kupnju određene dionice sa zaustavnom cijenom od 40 HRK i graničnom cijenom od 41 HRK u trenutku kada je tržišna cijena dionice jednaka 35 HRK. Ako cijena dionice u nekom trenutku dođe do 40 HRK, ovakav nalog se pretvara u nalog s ograničenjem s graničnom cijenom od 41 HRK.

MIT nalog (engl. *Market-if-touch*), je vrsta naloga koji se izvršava po najpovoljnijoj cijeni na tržištu u trenutku kada je obavljena transakcija po specificiranoj cijeni ili po cijeni povoljnijoj od specificirane. Ova vrsta naloga je suprotna nalogu za zaustavljanje. Naime, nalogom za zaustavljanje ograničavamo mogući gubitak, dok s MIT nalogom osiguravamo profit ukoliko se dogodi povoljno kretanje cijene dionice.

Također, imamo vrste naloga koje postavljaju uvjete na vrijeme izvršenja naloga. Ukoliko nije navedeno drukčije nalog je dnevni, tj. ističe kada se burza zatvori za trgovanje. Nalozi se mogu zadati tako da se mogu izvršiti jedino u određeno vrijeme tog dana, (engl. *Time-of-day order*). Postoji vrsta naloga koja se izvršava jedino pri samom otvaranju tržišta, (engl. *Open order*). Još jedna popularna vrsta naloga je nalog koji se mora izvršiti odmah po primitku ako su svi uvjeti zadovoljeni, inače se poništava, (engl. *Fill-or-kill*).

Na Zagrebačkoj burzi moguće je zadavati isključivo najjednostavniju vrstu naloga, tržišne naloge, pri čemu je dodatno moguće specificirati da li oni vrijede jedan dan ili više. Pomoću našeg programa investitor je u mogućnosti zadati bilo koju vrstu naloga prethodno navedenih te je u mogućnosti i uvjetovati vrijeme izvršavanja naloga. Štoviše, investitor ima slobodu kreirati proizvoljne vrste naloga sa vlastitim parametrima.

6.2 Primjeri programske izvedbe određenih vrsta naloga

U nastavku ćemo prikazati dva primjera kako investitor, korisnik našeg programa, može programski dizajnirati određene vrste naloga.

U prvom primjeru investitor zadaje nalog za kupnju 20 dionica HT-a s ograničenjem. Granična cijena je postavljena na 257.35 HRK.

```
% Primjer naloga za kupnju s ogranicenjem
ht = stocks{tickerID('HT-R-A', tickers)};

limitPrice = 257.35;
volume = 20;

if (ht.BestAsk <= limitPrice)
      % Ima li dovoljno gotovine za obavljanje kupnje
      if (myPortfolio.Cash >= volume*ht.BestAsk)
            % Zadaj nalog za kupnju dionice HT-a
            putOrder('Buy', ht.Ticker, ht.BestAsk, volume);
      end
end
```

U drugom primjeru investitor zadaje nalog za zaustavljanje sa zaustavnom cijenom od 559.5 HRK. U trenutku kada cijena dionice PBZ-a bude jednaka ili manja od zaustavne cijene zadaje se nalog za prodaju svih dionica PBZ-a koje investitor posjeduje.

```
% Primjer naloga za zaustavljanje
pbz = stocks{tickerID('PBZ-R-A', tickers)};
myPbz = myPortfolio.FindStock(pbz.Ticker);

if (pbz.BestBid <= 559.5)
    % Posjedujem li dionice PBZ-a?
    if (myPbz ~= 0)
        % Zadaj nalog za prodaju svih mojih dionica HT-a
        putOrder('Sell', pbz.Ticker, pbz.BestBid, myPbz.Volume);
    end
end
```

U posljednjem primjeru demonstrirati ćemo primjer naloga s vremenskim uvjetom. Investitor je odlučio rasprodati sve svoje udjele. Ipak, želi pričekati do zadnjeg časa i prodati svoje dionice sat vremena prije zatvaranja burze (napomena: *Zagrebačka burza se zatvara u 16h*).

```
% Primjer naloga s vremenskim uvjetom
[year month day hour minute second] = datevec(now);

numStocks = length(myPortfolio.Stocks);

% Sat prije zatvaranja burze u 16 sati
if (hour >= 15 && hour < 16 && numStocks ~= 0)
    % Prodaj sve dionice koje imam
    for i=1:numStocks
       myStock = myPortfolio.Stocks(i);
       stock = stocks{tickerID(myStock.Ticker, tickers)};
       putOrder('Sell', myStock.Ticker, stock.BestBid, myStock.Volume);
    end
end
```

6.3 Statistička arbitraža

6.3.1 Matematički opis algoritma

Koraci potrebni za razvijanje algoritma za trgovanja pomoću statističke arbitraže se temelje na međusobnim odnosima cijena ili nekih drugih varijabli koje karakteriziraju dvije dionice. Algoritam baziran na međusobnom odnosu cijena za bilo koje dvije dionice i i j se sastoji od sljedećih koraka.

1. Potrebno je odrediti prostor likvidnih dionica – dionica kojima se trguje barem jednom unutar željene frekvencije izvršavanja algoritma. Primjerice, ako želimo trgovati u razmacima od jednog sata, treba izabrati dionice kojima se trguje barem jednom svakog sata.

2. Izračunati razliku između cijena svake dvije promatrane dionice, i i j, koje smo odredili u prvom koraku. Napraviti to za cijeli vremenski interval povijesnih uzoraka.

$$\Delta S_{i,j,t} = S_{i,t} - S_{j,t} \quad t \in [1, T]$$

   gdje je T dovoljno veliki broj uzoraka. Centralni granični teorem kaže da je potrebno minimalno 30 uzoraka odabranih po željenoj frekvenciji trgovanja. Ipak, uzorci cijena dionica unutar jednog dana imaju veliku sezonalnost pa se stoga preporuča uzimati dnevne uzorke. Za robusno statističko zaključivanje poželjno je za T odabrati 500 dnevnih uzorka (približno dvije godine).

3. Za svaki par dionica, odabrati onaj sa najstabilnijim međusobnim odnosom – par dionica čije se cijene „kreću“ zajedno. Kako bi odredili takve parove dionica Gatev, Goetzmann i Rouwenhorst rade jednostavnu minimizaciju povijesnih razlika u povratima između svake dvije likvidne dionice:

$$\min_{i,j} \sum_{t=1}^{T} \left(\Delta S_{i,j,t} \right)^2$$

   Stabilnost veze se također može odrediti koristeći kointegraciju i druge statističke tehnike. Za svaku dionicu i, odabiremo dionicu j pomoću minimizacije sume kvadrata prikazane u prethodnoj jednadžbi.

4. Odrediti osnovna distribucijska svojstva niza razlika.

   Srednja vrijednost niza razlika:

$$E\left[\Delta S_t\right] = \frac{1}{T}\sum_{t=1}^{T}\Delta S_t$$

   Standardna devijacija:

$$\sigma\left[\Delta S_t\right] = \frac{1}{T-1}\sum_{t=1}^{T}\left(\Delta S_t - E\left[\Delta S_t\right]\right)^2$$

5. Promatrati nizove razlika cijena dionica te reagirati u određenom trenutku τ ako se dogodi jedna od slijedeće dvije situacije.

   a) Prodaj dionicu i i kupi dionicu j ako vrijedi:

$$\Delta S_\tau = S_{i,\tau} - S_{j,\tau} > E\left[\Delta S_\tau\right] + 2\sigma\left[\Delta S_\tau\right]$$

   b) Kupi dionicu i i prodaj dionicu j :

$$\Delta S_\tau = S_{i,\tau} - S_{j,\tau} < E\left[\Delta S_\tau\right] - 2\sigma\left[\Delta S_\tau\right]$$

6. Jednom kad se uočena nepravilnost u kretanju cijena dionica ispravi, zatvorimo pozicije (prodamo dionice i i j) kako bi ostvarili dobit. Inače, ako se cijene dionica kreću u smjeru suprotnom od predviđenog, zadajemo nalog za zaustavljanje (engl. *Stop-loss order*) kako bi ograničili gubitak.

Algoritam statističke arbitraže se može prilagoditi tako da se dinamički prilagođava uvjetima na tržištu. Srednja vrijednost varijable koja se promatra i prema kojoj bi uočeni statistički odnos trebao težiti, može se izračunati kao pomični težinski prosjek, (engl. *Weighted moving average*) pri čemu je veća težina dana svježijim uzorcima. Kako bi standardna devijacija što bolje odražavala najnovije ekonomsko okruženje, možemo ju izračunati koristeći ograničen broj najsvježijih uzoraka.

Međutim, metoda statističke arbitraže ima i ozbiljne nedostatke. Često su uočeni statistički odnosi nasumični ili varljivi pa imaju malu ili nikakvu prediktivnu moć. Ali zato statističke relacije koje se potvrđene kroz akademska istraživanja na području ekonomije i financija konzistentno proizvode pozitivne rezultate za mnoge ulagače. Stoga je potrebno temeljito poznavanje

ekonomske teorije kako bi mogli razlikovati nasumične relacije od onih čvrstih i pouzdanih koje nas mogu dovesti do željenog profita.

Treba istaknuti da je strategija statističke arbitraže izložena brojnim nepovoljnim situacijama zbog prirode samih tržišta kapitala:

- Neočekivanim promjenama u regulaciji, ili naglim padovima uzrokovanim nepredvidivim događajima poput terorističkih napada ili kataklizmi.
- Troškovi trgovanja mogu izbrisati svu dobit ostvarenu strategijom statističke arbitraže što je najizraženije kod malih investitora ili investitora s ograničenim kapitalom.
- Razlika između ponuđene i tražene cijene (engl. *Bid-ask spread*) može biti toliko velika da poništi svaku dobit stečenu ovakvim načinom trgovanja.
- Konačno, rad algoritma ne mora biti narušen samo neočekivanim događajima, već i onim očekivanim kao što je skok cijene dionice zbog objave rezultata poslovanja kompanije.

Unatoč navedenim slabostima strategije, rezultati znanstvenih istraživanja pokazuju da se pravilnim odabirom statističkih relacija i upravljanjem rizikom mogu postići značajni prinosi već na dnevnim uzorcima. Na uzorcima veće frekvencije je moguće postići još veće prinose.

6.3.2 Izvedba strategije statističke arbitraže

Analizom podataka o izvršenim trgovanjima unutar dana za prvo polugodište 2011. godine, identificirali smo sedam najlikvidnijih dionica na Zagrebačkoj burzi:

- Hrvatski Telekom d.d.: HT-R-A
- Ingra d.d.: INGR-R-A
- Belje d.d.: BLJE-R-A
- Petrokemija d.d.: PTKM-R-A
- Ericsson Nikola Tesla d.d.: ERNT-R-A
- Zagrebačka banka d.d.: ZABA-R-A
- Privredna banka Zagreb: PBZ-R-A

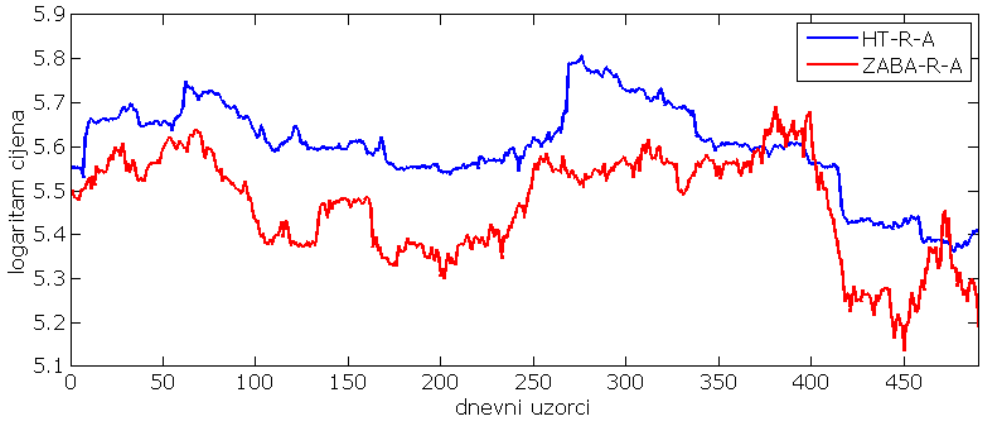
Za sve navedene dionice smo prikupili 500 zaključnih cijena koje odgovaraju vremenskom razdoblju: svibanj 2009. – svibanj 2011. Promatrat ćemo logaritmirane cijene (engl. *Price levels*). Prema 2. koraku izračunamo razlike između logaritamskih cijena dviju promatranih dionica. U 3. koraku koristimo minimizaciju sume kvadrata za svaki par promatranih dionica. Rezultati trećeg koraka su prikazani slijedećom tablicom. Osjenčana polja prikazuju par dionica koje imaju najmanju sumu kvadrata razlika. Odlučili smo se odabrati jedan par

takvih dionica te za njih implementirati algoritam statističke arbitraže koji bi se izvršavao u stvarnom vremenu na našem sustavu.

Tablica 6. Sume kvadrata razlika za odabrane dionice

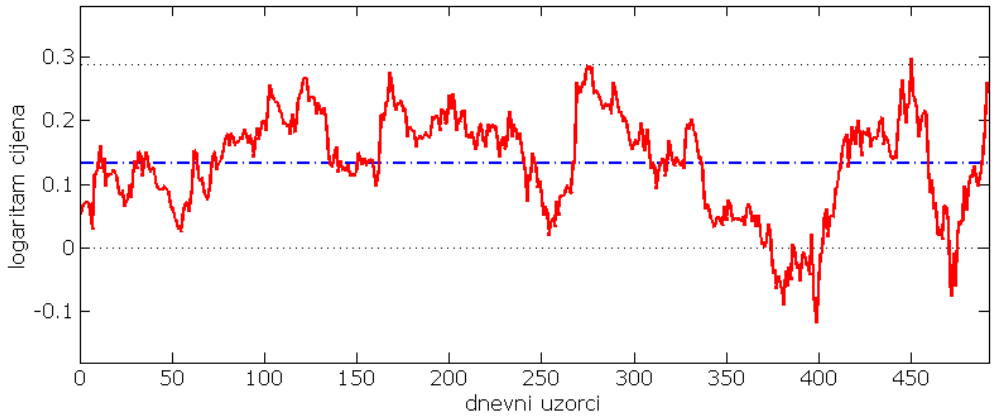
	BLJE-R-A	HT-R-A	INGR-R-A	PTKM-R-A	ERNT-R-A	ZABA-R-A	PBZ-R-A
BLJE-R-A		741,61	**624,34**	171,86	4080,15	582,51	1766,91
HT-R-A	741,61		2623,63	222,01	1395,97	**11,61**	**247,49**
INGR-R-A	624,34	2623,63		1378,65	7706,03	2305,21	4294,71
PTKM-R-A	**171,86**	222,01	1378,65		2676,54	138,75	893,86
ERNT-R-A	4080,15	1395,97	7706,03	2676,54		1572,09	393,57
ZABA-R-A	582,51	**11,61**	2305,21	**138,75**	1572,09		345,26
PBZ-R-A	1766,91	247,49	4294,71	893,86	**393,57**	345,26	

Za daljnju razradu smo izabrali dionice Hrvatskog telekoma i Zagrebačke banke. Ustanovili smo da između ovog para postoji najmanje odstupanje u cijenama. Slika 18. prikazuje 500 povijesnih cijena za ove dvije dionice te nam potvrđuje postojanost povezanosti cijena odabranih dionica.



Slika 19. Prikaz 500 povijesnih uzoraka cijena dionica HT-a i ZABA-e

Sada, prema 5. koraku algoritma izračunamo srednju vrijednost i standardnu devijaciju razlike cijena dviju dionica. Za srednju vrijednost dobivamo $E[\Delta S_t] = 0.1322$, a za standardnu devijaciju dobivamo $\sigma[\Delta S_t] = 0.0782$. Nakon što smo obavili ove pripremne korake, preostaje nam još samo završni, 6. korak u kojem promatramo cijene dionica i reagiramo u skladu s relacijama navedenim u koraku 6. Osnovna ideja promatranja tržišta je ilustrirana slikom 19. Isprekidana plava linija označuje srednju vrijednost razlike cijena, dok dvije sive isprekidane linije prikazuju odmak od srednje vrijednosti za dvije standardne devijacije (*interval pouzdanosti od 95%*). Crvena linija predstavlja razliku logaritmiranih cijena, a algoritam bi trebao napraviti odgovarajući potez kada ta razlika izađe van intervala pouzdanosti.



Slika 20. Srednja vrijednost i intervali pouzdanosti

U nastavku prilažemo izvorni kod algoritma statističke arbitraže koji je prilagođen za rad s našim programom. Testirali smo algoritam statističke arbitraže na povijesnim visokofrekventnim podacima iz razdoblja od 2. svibnja do 13. svibnja. Međutim, u navedenom razdoblju, algoritam nije zadao niti jedan nalog što možemo objasniti slabom likvidnošću dionica na Zagrebačkoj burzi. Također ako pogledamo Sliku 20, možemo vidjeti da se odstupanje od dvije standardne devijacije dogodilo samo tri puta u dvije godine pa je ovakav ishod testiranja algoritma bio očekivan. Stoga, zaključujemo da na nelikvidnim tržištima kapitala moramo koristiti više algoritama koji bi konstantno promatrali tržište i iskoristili priliku kada se ona ukaže.

```
% Primjer algoritma statisticke arbitraze
mean = 0.1322;
std = 0.0782;

first = stocks{tickerID('HT-R-A', tickers)};
second = stocks{tickerID('ZABA-R-A', tickers)};

diff = log(first.LastPrice) - log(second.LastPrice);

% Prodaj prvu, kupi drugu
if (diff > (mean + 2*std))
    myFirst = myPotfolio.FindStock(first.Ticker);

    if (myFirst ~= 0)
       putOrder('Sell', first.Ticker, first.BestBid, myFirst.Volume);

       % Sav novac od prodaje ulozi u kupnju
       cash = first.BestBid*myFirst.Volume;

       putOrder('Buy', second.Ticker, second.BestAsk,
       floor(cash/second.BestAsk));
    end

% Kupi prvu, prodaj drugu
elseif (diff < (mean - 2*std))
    mySecond = myPortfolio.FindStock(second.Ticker);

    if (mySecond ~= 0)
        putOrder('Sell', second.Ticker, second.BestBid,
        mySecond.Volume);

        % Sav novac od prodaje ulozi u kupnju
        cash = second.BestBid*mySecond.Volume;

        putOrder('Buy', first.Ticker, first.BestAsk,
        floor(cash/first.BestAsk));
    end
end
```

7 Zaključak

U ovom radu smo predstavili visokofrekventno trgovanje kao nov način trgovanja koji se uvelike razlikuje od svojih tradicionalnih prethodnika. Visokofrekventno trgovanje donosi brojne pogodnosti i zasigurno predstavlja budućnost trgovanja na svjetskim tržištima kapitala.

Budući da je za ovaj način trgovanja potrebna složena tehnička infrastruktura te pristup podacima u stvarnom vremenu, istraživači, mali investitori i entuzijasti nemaju priliku provesti svoje ideje u djela. Sustav za simulaciju visokofrekventnog trgovanja na Zagrebačkoj burzi je namijenjen upravo njima.

Razvijeni sustav se sastoji od glavnog programa i četiri pomoćna modula koji se oslanjaju na programski model domene. Modul za prikupljanje podataka u stvarnom vremenu koristi JSON web servise te osigurava da svaki modul sustava uvijek ima najnovije podatke o stanju na tržištu. Modul za trajnu pohranu podataka sprema podatke o tržištu na kompaktan način što ovom sustavu daje dodatnu vrijednost jer dobivamo vrijedne visokofrekventne serije povijesnih podataka. Štoviše, ovaj modul može vizualizirati pohranjene serije podataka. Modul za simuliranje strategija visokofrekventnog trgovanja se brine o organizaciji i uređenju korisničkih algoritama. Također ovaj modul automatski pokreće simulacije i predstavlja njihove rezultate. Prije nego se ostvareni algoritmi aktiviraju, moguće ih je testirati. Za testiranje je odgovoran poseban modul koji provodi testiranje na povijesnim podacima koje je program dotada prikupio.

Kroz cijeli proces izrade sustava, od modeliranja arhitekture do konkretne implementacije, vodilo se računa o modularnosti te jednostavnosti uvođenja budućih poboljšanja i nadogradnji. Sustav je potpuno funkcionalan, a daljnja poboljšanja su moguća u vidu paralelizacije izvršavanja simulacija te dodavanja dodatnih mogućnosti za manipulaciju i praćenje rada simulacije. No, najveći izazovi kriju se u osmišljanju novih strategija trgovanja, stvaranju inovativnih modela kojima možemo pobijediti tržište.

Željan Juretić

Literatura

Aldridge, I. (2010). *High-Frequency Trading: a practical guide to algorithmic strategies and trading system.* New Jersey: John Wiley & Sons.

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2010). *Web Services: Concepts, Architectures and Applications (Data-Centric Systems and Applications).* Palo Alto: Springer.

Crockford, D. (n.d.). *JSON*. Preuzeto 22. svibanj 2011 iz http://www.json.org

Durbin, M. (2010). *All About High-Frequency Trading.* New York: McGraw-Hill.

Evans, E. (2004). *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Boston: Pearson Education.

Freeman, E., Freeman, E., Bates, B., & Sierra, K. (2004). *Head First Design Patterns.* Sebastopol: O'Reilly.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (2000). *Design Patterns: Elements of Reusable Object-Oriented Software.* New York: Addison-Wesley Professional.

Gatev, G. R. (2006). Pairs Tradeing: Performance of Relative-Value Arbitrage Rule. *Review of Financial Studies*, 797-827.

Hull C., J. (2002). *Options, Futures, and Other Derivatives (5th Edition).* New Jersey: Prentice Hall.

Pratap, R. (2009). *Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers.* New York: Oxford University Press .

Troelsen, A. (2010). *Pro C# 2010 and the .NET 4 Platform.* New York: APress.

PLATFORMA ZA SIMULIRANJE VISOKOFREKVENTNOG TRGOVANJA DIONICAMA

Sažetak:

Tehnološkim razvojem i modernizacijom burza pojavilo se visokofrekventno trgovanje. Ovaj način trgovanja se temelji na velikom broju transakcija s relativno malim prinosima, pri čemu se dionice drže od samo nekoliko minuta do najviše nekoliko dana. Razvili smo programski sustav za simuliranje korisničkih strategija za visokofrekventno trgovanje na Zagrebačkoj burzi. Sustav može simulirati više strategija u stvarnom vremenu i istodobno prikazivati stanje korisničkog portfelja te zadane naloge i izvršene transakcije. Dodatna vrijednost programa je njegova mogućnost trajne pohrane visokofrekventnih serija cijena koje su inače teško dostupne. Opisali smo neke napredne vrste naloga te ponudili njihovo ostvarenje prilagođeno za rad sa simulatorom. Kao primjer algoritma za visokofrekventno trgovanje smo istaknuli strategiju statističke arbitraže te smo ponudili njenu implementaciju.

Ključne riječi:

Visokofrekventno trgovanje, algoritamsko trgovanje, dionice, Zagrebačka burza, simulacija trgovanja

SOFTWARE PLATFORM FOR HIGH-FREQUENCY TRADING

Abstract:

High-frequency trading has arisen along with the technological advancement and stock exchange modernization. This way of trading is based on large number of transactions with relatively small returns and it is characterized by brief position-holding periods, ranging from several minutes to several days. We have developed software system for simulating user high-frequency strategies on Zagreb Stock Exchange. System is capable to simulate multiple strategies in real time and provide information on user portfolio state, pending orders and executed transactions. Furthermore, the system is able to permanently store high-frequency price series, which are hard to obtain otherwise. We have described some advanced orders and offered their implementation adapted for our system. Moreover, we discussed stat-arbitrage as an example of high-frequency trading algorithm and gave its implementation.

Keywords:

High-frequency trading, algorithmic trading, securities, Zagreb Stock Exchange, High-frequency trading