



UNIVERSITATEA „DUNĂREA DE JOS” DIN GALAȚI  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE,  
INGINERIE ELECTRICĂ ȘI ELECTRONICĂ



# **LUCRARE DE DISERTAȚIE**

Coordonator științific,  
Prof. Dr. Ing. Luminița DUMITRIU

Absolvent,  
Tudorel OPRIȘAN

Galați  
2018



SPECIALIZAREA: TEHNOLOGII INFORMATICE AVANSATE

# **SISTEM DE TRANZACȚIONARE AUTOMAT**

Coordonator științific,  
Prof. Dr. Ing. Luminița DUMITRIU

Absolvent,  
Tudorel OPRIȘAN

Galați  
2018

## CUPRINS

Introducere.....	6
Capitolul 1. Cerințe și specificații.....	8
Capitolul 2. Analiza sistemelor existente.....	9
2.1 Expert Advisors .....	9
2.2 Programe de sine stătătoare .....	10
Capitolul 3. Proiectarea aplicației.....	13
3.1 Elementele de bază ale tranzacționării algoritmice: Concepte și exemple.....	13
3.1.1 Beneficiile tranzacționării algoritmice.....	16
3.1.2 Strategii de tranzacționare algoritmică .....	17
3.1.3 Cerințe tehnice pentru tranzacționarea algoritmică.....	19
3.1.4 Tipuri de strategii de arbitraj și aplicarea lor .....	19
3.1.5 Concluzii .....	20
3.2 Analize si backtesting in Python.....	21
3.3 Diagrame UML.....	24
3.3.1 Pachetul tradingAPI .....	24
3.4 Mod funcționare .....	36
Capitolul 4. mijloace de implementare a designului .....	39
4.1 Transferul de stat reprezentativ (REST).....	39
4.1.1 Derivarea REST.....	39
4.1.1.1 Începând cu stilul Null .....	39
4.1.1.2 Client-Server .....	40
4.1.1.3 Stateless (fără stare).....	40
4.1.1.4 Cache .....	41
4.2 NumPy.....	41
Capitolul 5. testarea aplicației.....	44
5.1 Tipuri de erori ale aplicației.....	44
5.1.1 Modificări în codul sursă al aplicației .....	44
5.1.2 Erori cauzate de programator .....	44
5.1.3 Erori cauzate de utilizator .....	44
5.1.4 Erori care țin de factori externi.....	44
5.2 Moduri de testare ale aplicației .....	45
5.2.1 Testarea unitară .....	45

5.2.2 Testarea integrării.....	45
5.3 Testarea aplicației .....	46
Concluzii .....	50
Bibliografie .....	51
Anexa 1.....	52
Diagrame UML(Continuare) .....	52

## LISTA FIGURILOR

Figura 1 Evoluția în timp a tranzacțiilor automate ca procent din piață .....	6
Figura 2 Raport al strategiei de testare .....	9
Figura 3 Rezultat grafic al unui EA bazat pe două Moving averages .....	10
Figura 4 AlgoTerminal .....	11
Figura 5 Sumar analiza .....	11
Figura 6 Câștiguri.....	12
Figura 7 Pierderi .....	12
Figura 8 Exemplu de sistem de trading automat.....	15
Figura 9 Script python.....	21
Figura 10 Script python(cont.) .....	21
Figura 11 Corelație parțială .....	22
Figura 12 indicații trading calculate automat.....	22
Figura 13 Strategia „Fade the move” .....	23
Figura 14 Diagrama UML pachetul account.....	25
Figura 15 Diagrama UML pachetul account în amănunt .....	26
Figura 16 Pachetul event .....	27
Figura 17 Diagrama UML pachetele event și heartbeat in amanunt.....	27
Figura 18 Diagrama UML pachetul market.....	28
Figura 19 Diagrama UML pachetul market în amănunt.....	29
Figura 20 Diagrama UML pachetul market in amanunt(cont.) .....	30
Figura 21 Diagrama UML pachetul order .....	31
Figura 22 Diagrama UML pachetul trade.....	31
Figura 23 Diagrama UML pachetul order în amănunt.....	32
Figura 24 Diagrama UML pachetul trade în amănunt .....	33
Figura 25 Diagrama UML pachetul twitter .....	34
Figura 26 Diagrama UML pachetul strategy.....	35
Figura 27 Stilul Null .....	39
Figura 28 Client-Server.....	40
Figura 29 Client – Stateless - Server .....	40

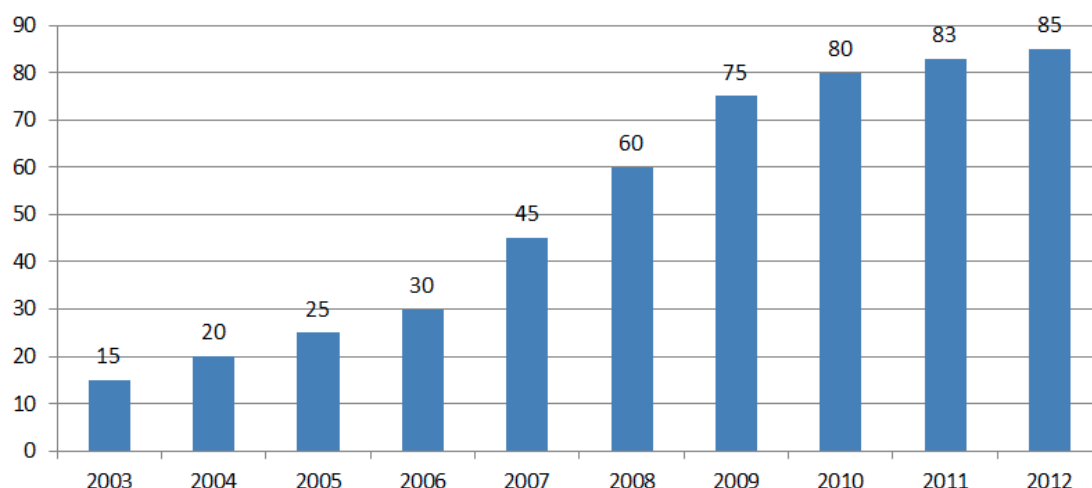
Figura 30 Client + Cache – Stateless - Server .....	41
Figura 31 Test cont utilizator si permisiuni .....	46
Figura 32est trimitere email.....	47
Figura 33Email primit.....	47
Figura 34 No such Endpoint .....	48
Figura 35 Test strategie.....	48
Figura 36 Test care nu a trecut.....	48
Figura 37 Ordinele nu se vor plasa datorită discrepanței de valori față de piață.....	48
Figura 38 Diagrama UML Pachetul de teste .....	49
Figura 39 Diagrama UML pachetul email .....	52
Figura 40 Diagrama UML pachetul broker/account .....	52
Figura 41 Diagrama UML pachetul broker/account în amănunt .....	53
Figura 42 Diagrama UML pachetul broker/events în detaliu.....	54
Figura 43 Diagrama UML pachetul broker în detaliu .....	55
Figura 44 Diagrama UML pachetul events în detaliu .....	56
Figura 45 Diagrama UML pachetele instruments/marketData/order în detaliu .....	57

## INTRODUCERE

Progresele tehnologice în domeniul telecomunicațiilor din ultimul deceniu au creat piețe financiare din ce în ce mai complexe, dinamice și extinse, care la rândul lor au stimulat tranzacționarea efectuată de programe software. Aceste sisteme automatizează unul sau mai multe etape din procesul de tranzacționare, plasând ordine care necesită o intervenție umană minimă (sau absentă). Printre funcțiile posibile se mai numără:

- detectarea anomaliilor temporare legate de preț;
- detectarea de modele statistice profitabile în cadrul piețelor financiare
- execuția optimă a ordinelor
- detectarea și exploatarea strategiilor rivale

Toate aceste funcții au, în cele din urmă, ca scop maximizarea profitului, fie că e vorba de costuri de tranzacționare mai reduse, de comisioane percepute sau de tranzacții pe termen lung și foarte lung.



*Figura 1 Evoluția în timp a tranzacțiilor automate ca procent din piață*

Tranzacționarea automată a luat amploare la începutul anilor 2000, sprijinită de explozia numărului de companii din industria IT (dot com bubble), iar în 2005 a ajuns să reprezinte deja un sfert din volumul total, ajungând ulterior la 75% în decurs de numai 4 ani. În prezent se estimează că 9 din 10 tranzacții sunt realizate în mod automat.

Lucrarea prezenta are ca scop studiul și implementarea modalităților curente de a tranzacționa algoritmic. Tranzacționarea financiară este un domeniu vast dar, de mare importanță fie că vorbim despre piața bursieră, FX, criptomonede, commodități sau alte instrumente financiare. Există o multitudine de tipuri de strategii, care diferă atât de la tipul de tranzacționare (bursa, forex, etc.) cât și la același tip de instrument. În ceea ce privește profitabilitatea acestora, se spune că un procentaj foarte mic din cei ce tranzacționează au profit, de unde și denumirea de “a juca la bursă”. Fapul că există multe strategii nu implică și profitabilitatea acestora.

În ultimii ani, odată cu evoluția sistemelor de calcul și accesibilitatea acestora față de publicul larg, a luat amploare domeniul machine learning. Deși este o certitudine că există algoritmi de tranzactionare profitabili, aceștia nu sunt disponibili publicului larg.

## CAPITOLUL 1. CERINȚE ȘI SPECIFICAȚII

Tema acestei lucrări este dezvoltarea unei aplicații, sau al unui ansamblu de aplicații cu ajutorul cărora să se identifice oportunități profitabile în piață și executarea ordinelor de tranzacționare automat, fără intervenție manuală. Identificarea oportunităților se va face prin crearea unui model bazat pe informații precedente rezultate din analiza pieței, sau prin accesarea de semnale provenite din alte surse.

În dezvoltarea aplicației am folosit următoarele tehnologii:

- Java – pentru sistemul de management al ordinelor din piață și pentru accesul la date provenite din exterior
- Python – pentru analiza seturilor de date istorice, fapt ce va identifica anumite oportunități. Am ales python pentru ca este un limbaj de programare cu ajutorul căruia calcule matematice se realizează cu ușurință.
- React.JS pentru realizarea interfaței cu utilizatorul

Aplicația Java comunică cu un sistem de analiză creat în Python, de la care primește anumite semnale. Intervenția umană este posibilă, pentru că, în final aplicația lucrează cu bani reali, și sunt necesare cât mai multe modalități de a opri tranzacțiile sau ordinele noi. De asemenea aplicația va comunica cu unul sau mai mulți brokeri, sistemul de conectare fiind configurabil în măsura în care aceștia o permit. Se pot selecta doar anumite perechi valutare tranzactionate, sau tipul de piață pe care se tranzacționează. Aplicația permite următoarele:

- crearea de ordine noi
- închiderea de poziții
- alterarea pozițiilor deja deschise
- anularea ordinelor ne-executate
- consultarea și limitarea volumelor tranzactionate
- vizualizarea evoluției algoritmilor folosiți
- conectarea la diferiți brokeri
- conectarea la diferite tipuri de piețe
- notificarea prin email atunci când se efectuează un ordin nou



## CAPITOLUL 2. ANALIZA SISTEMELOR EXISTENTE

Sisteme asemănătoare există cu siguranță, dar accesibilitatea lor este destul de mică. Cele care funcționează deja ori sunt foarte scumpe, ori sunt ascunse de ochii publicului, și pe bună dreptate. Cele care sunt accesibile, deși sunt promovate ca fiind de succes, se dovedesc a fi, de cele mai multe ori doar modalități de a extrage bani de la traderii neexperimentați.

### 2.1 Expert Advisors

Cele mai întâlnite sisteme de acest gen sunt cunoscute sub denumirea de Expert Advisors, pe scurt EA, și funcționează de regulă pe platforme de genul MT4/5 (MetaTrader 4/5). MetaTrader este o platformă de trading și analiză forex, care are suport pentru asemenea tip de roboți. Aceștia sunt disponibili gratuit sau contracost, prețurile variind între 20\$ - 5000\$.

Ca exemplu am accesat pagina <https://www.mql5.com>, care este un repository de astfel de EA, și am ales unul la întâmplare. Acesta se numește **MARTINGALE VI HYBRID - expert for MetaTrader 4**. Pe pagina de prezentare sunt afișate mai multe informații despre profitabilitatea acestuia:

Strategy Tester Report MARTINGALE VI HYBRID (Build 1090)			
Symbol	GBFUSD (British Pound vs US Dollar)		
Period	Daily (D1) 2013.01.02 00:00 - 2018.04.30 23:59 (2013.01.01 - 2018.05.01)		
Model	Open prices only (only for Expert Advisors that explicitly control bar opening)		
Parameters	TakeProfit=100; PipStep=20; Lots=0.1; Multiply=2; MaxTrade=11; FASTMA=2; SLOWMA=75; MagicNumber=8095;		
Bars in test	2061 Ticks modelled	3444 Modelling quality	n/a
Mismatched charts errors	0		
Initial deposit	10000.00	Spread	Current (20)
Total net profit	2750.89	Gross profit	7350.53
Profit factor	1.60	Expected payoff	4.98
Absolute drawdown	132.54	Maximal drawdown	526.47 (4.25%)
		Relative drawdown	4.57% (473.00)
Total trades	552	Short positions (won %)	94 (87.23%)
		Long positions (won %)	458 (87.34%)
		Profit trades (% of total)	482 (87.32%)
		Loss trades (% of total)	70 (12.68%)
		Largest profit trade	73.30
		loss trade	-196.15
		Average profit trade	15.25
		loss trade	-65.71
		Maximum consecutive wins (profit in money)	47 (687.74)
		consecutive losses (loss in money)	3 (-350.36)
		Maximal consecutive profit (count of wins)	687.74 (47)
		consecutive loss (count of losses)	-350.36 (3)
		Average consecutive win	14
		consecutive losses	2

Figura 2 Raport al strategiei de testare

Acest expert are la baza o strategie bazată pe două moving averages(MA). Un moving average reprezintă media cursului instrumentului respectiv pe o perioadă de timp selectată. Se poate observa din figura 2 că robotul funcționează, având un profit net de 2750.89USD. Tot din figură se mai observă faptul că acesta este corect în majoritatea cazurilor în care a fost testat. Ordinele consecutive pierdute sunt în număr mic, iar cele terminate pe profit sunt în număr mai ridicat. Acest EA este parametrizat iar parametrii cu care rulează au impact asupra rezultatului final.

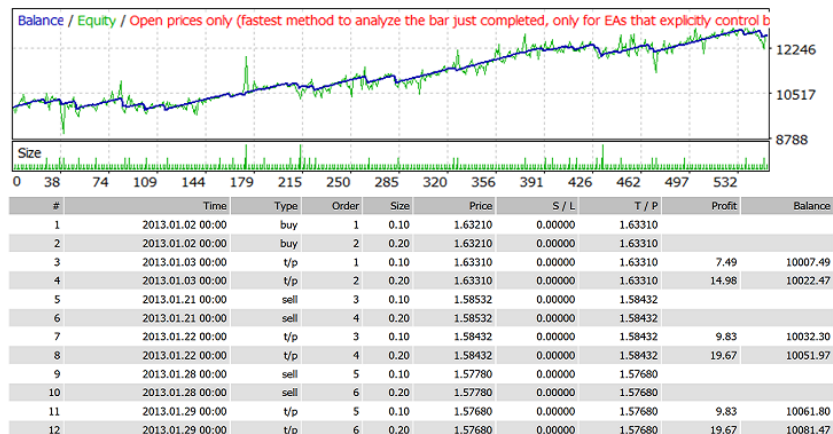


Figura 3 Rezultat grafic al unui EA bazat pe două Moving averages

Cuvântul important din titlul expertului este Martingale. Strategia Martingale, nu este o strategie nouă, ea fiind folosită des în gambling. Spre exemplu, la jocul de ruletă, poți paria mereu pe o culoare (roșu sau negru), cu condiția ca de fiecare dată când pariul nu are succes, suma pariată se va dubla. Probabilitatea ca aceiași culoare să apară de multe ori la rând scade cu fiecare pariu. Aplicat la cazul de față, dacă poziția deschisă merge în direcția opusă pe care o dorim, după un număr predefinit de pips, expertul va deschide încă o poziție în același sens ca prima poziție. Procesul se repeta până când avem profit.

Graficul din figura 3 arată că profitul este posibil, într-adevăr. Întrebarea cheie aici este: de ce ar vine cineva un astfel de robot (sau de ce l-ar distribui gratuit), care, cu siguranță îi aduce profit. Răspunsul este simplu, premisa că are profit este greșită din start. Toată lumea ar folosi acea strategie și ar avea profit. Dacă sună prea bine ca să fie adevărat, de cele mai multe ori doar sună bine. Piața nu evoluează în termenii simpli pe care îi avem față de analogia cu ruleta (în majoritatea cazinourilor strategia Martingale este interzisă). Nu este ușor de găsit o predictibilitate în evoluția prețului. În acest caz, Martingale nu este decât un gambling. Desigur că imaginile prezentate mai sus sunt reale, dar perioadele de timp pe care au fost testate și volumele sunt prea mici, backtestingul fiind inexistent. Din experiența personală a backtestingului mai multor tipuri de EA, am observat că aceștia înregistrează mai multe ordine cu profituri mici, dar ordinele terminate deși mai puține, au pierderi mult mai mari.

## 2.2 Programe de sine stătătoare

Există însă și alte tipuri de programe de pentru tranzacționare algoritmică, dar acestea nu sunt gratuite, funcționează ca un SaaS (Software as a Service) și de cele mai multe ori sunt destinate fondurilor de investiții (hedge fund). Prețurile pot varia de la câteva sute de dolari pe lună la câteva mii de dolari lunar. Un astfel de program este și AlgoTerminal. Subscripția lunară este de 299\$.

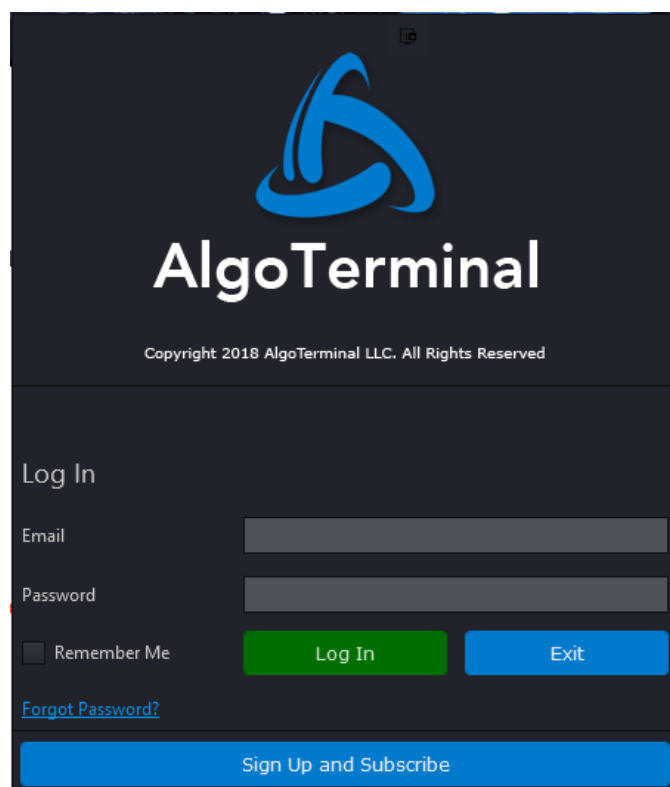


Figura 4 AlgoTerminal

Varianta trial a programului are două strategii implementate, ambele având backtesting cu un ROI pozitiv. Din păcate nu am exista set de date istoric pentru instrumentele forex, ci doar pentru bursă. Analiza a fost efectuată pe un portofiu mare, pe o perioadă începând cu Ianuarie 2017 până în prezent. Mai jos sunt rezultatele algoritmului

Heading	Value
Starting Date	01/01/2017
Ending Date	08/23/2018
Starting Equity	\$101,000
Ending Equity	\$182,068.01
Net Profit	\$81,068.01
Net Profit %	80.27%
Gross Profit	\$246,792.03
Gross Loss	\$-165,724.02
CAGR	43.20%
Max. DD	\$-77,376
Max DD %	-59.52%

Figura 5 Sumar analiza

Heading	Value	Long	Short
Total Wins	394	339	55
Winning %	71.38%	72.28%	66.27%
Total Win PnL	\$246,792.03	\$215,641.54	\$31,150.49
Max. Win PnL	\$4,725.7	\$4,725.7	\$1,889.04
Max. Win PnL %	15.73%	15.73%	6.30%
Avg. Win PnL	\$626.38	\$636.11	\$566.37
Avg. Win PnL %	2.09%	2.12%	1.89%
Avg. Win Bars	9.09	9.36	7.44
Avg. Win Bar PnL	\$113.42	\$115.15	\$102.74
Avg. Win Time	11.8 / d	12.18 / d	9.42 / d
Total Efficiency (Win)	51.51%	51.82%	49.63%
Entry Efficiency (Win)	66.01%	66.05%	65.73%
Exit Efficiency (Win)	85.51%	85.77%	83.90%
Max. Consecutive Wins	19	21	13

Figura 6 Câștiguri

Heading	Value	Long	Short
Total Losses	158	130	28
Losing %	28.62%	27.72%	33.73%
Total Loss PnL	\$-165,724.02	\$-129,447.96	\$-36,276.06
Max. Loss PnL	\$-8,360.27	\$-7,134	\$-8,360.27
Max. Loss PnL %	-27.96%	-24.07%	-27.96%
Avg. Loss PnL	\$-1,048.89	\$-995.75	\$-1,295.57
Avg. Loss PnL %	-3.50%	-3.33%	-4.30%
Avg. Loss Bars	25.37	25.6	24.29
Avg. Loss Bar PnL	\$-44.77	\$-44.09	\$-47.94
Avg. Loss Time	35.47 / d	35.87 / d	33.58 / d
Total Efficiency (Loss)	-33.68%	-33.47%	-34.73%
Entry Efficiency (Loss)	16.54%	16.51%	16.70%
Exit Efficiency (Loss)	49.78%	50.02%	48.56%
Max. Consecutive Loss...	5	5	3

Figura 7 Pierderi

După cum se poate observa algoritmul pare a fi profitabil în proporție de ~70% din timp. Însă pentru a tranzacționa la bursă capitalul inițial necesar este mult mai mare față de cel din forex. Majoritatea programelor de acest fel, nu au prețul direct afișat pe site, ceea ce înseamnă, de cele mai multe ori, că nu este accesibil publicului larg.

### **CAPITOLUL 3. PROIECTAREA APLICAȚIEI**

#### **3.1 Elementele de bază ale tranzacționării algoritmice: Concepte și exemple**

Primele modele de sisteme automate de trading erau folosite de manageri financiari, brokeri sau alte entități asemănătoare pentru a administra automat portofoliile clienților. Ulterior sistemele au fost îmbunătățite pentru a genera semnale de tranzacție, folosind limbajul de programare MQL, iar mai târziu cu acces direct la piață. Algoritmii procesează input-ul programului, execuția lor fiind condusă în totalitate de datele de intrare venite din piețe, cum ar fi ratele de schimb, tranzacțiile în sine, informații legate de ordinele plasate (anulate, onorate, neonorate) etc.

Cel mai important avantaj al sistemelor automate față de metodele discreționare îl reprezintă evaluarea performanței algoritmilor prin aplicarea lor pe date istorice, proces cunoscut sub numele de backtesting. Acesta permite determinarea proprietăților statistice ale unei strategii și indică șansele ei de profitabilitate.

Un sistem automat de trading este, de asemenea, mai eficient, prin reducerea numărului de oameni care monitorizează piețele pentru creșteri / scăderi de prețuri, știri sau alte vești din piață. Astfel, se economisește timp prețios care ar putea fi folosit în scopuri mult mai productive, cum ar fi perfecționarea strategiei sau chiar crearea mai multor strategii profitabile pentru același portofoliu de investiții. Mai mult, se automatizează managementul de risc și procesul de stabilire a valorii tranzacției, răspunzând direct la condițiile pieței, în timp real, spre deosebire de tranzacționarea clasică, unde este imposibilă calcularea continuă a riscului și monitorizarea continuă a pieței.

Un alt avantaj important îl reprezintă statisticile generate pe baza strategiilor implicate în sisteme, care produc comparații mult mai amănunțite între date (ex: creșterile valorii capitalului, risc (în diversele lui forme), frecvența de tranzacționare) și permit alocarea optimă de resurse financiare pentru o anumită poziție. Aceste cifre au un impact semnificativ mai mare asupra profitabilității, deoarece în cazul tranzacțiilor clasice doar informațiile despre profituri și pierderi sunt contorizate, putând masca potențialele diminuări sau reveniri rapide.

Deși avantajele sistemului de trading sunt numeroase, există și câteva dezavantaje, printre cele mai importante numărându-se cerințele mult mari de capital decât în cazul tranzacționării clasice. Motivul se datorează numărului mic de brokeri din spațiul comerțului automat care lucrează cu conturi sub 10.000 de dolari balanța minimă. Pe lângă acestea se adaugă costurile de acces la fluxurile de date financiare (cele mai ieftine situându-se între 300 și 500 de dolari pe lună), costurile achiziției de echipamente performante, precum și costurile aferente fiecărei piețe în parte - de exemplu, în Statele Unite, Comisia pentru Valori Mobiliare și de Schimb solicită un minim de 25.000 de dolari menținut în capitalul propriu în orice moment, în anumite situații din piață.

Cu toate acestea, tranzacționarea bazată pe algoritmi se diferențiază de alte tipuri de investiții prin oferirea unor așteptări solide despre performanțele din viitor, consecință a disponibilității ridicate de date folosite în backtesting. Pe scurt, acest proces presupune expunerea algoritmului creat pe baza unei strategii la un flux de date financiare, cu scopul de a crea semnale sau alerte de tranzacționare. În urma acestui semnal se plasează o tranzacție, care va înregistra profit sau va pierde; suma tuturor acumulărilor de profit / pierderi pe durata procesului de backtesting va conduce la analiza profitabilității strategiei testate. Astfel se elimină strategiile care nu respectă criteriile stabilite de performanță din lista strategiilor de tranzacționare. Pe lângă filtrarea strategiilor, backtesting-ul mai permite vizualizarea a noi modele de fenomene întâlnite în piețele financiare, cum ar fi costurile de tranzacționare, rutarea ordinelor, latența, lichiditatea etc.

Backtesting-ul oferă numeroase avantaje sistemelor automate de trading, însă nu mereu există posibilitatea testării corespunzătoare a unei strategii, datorită a diverse tipuri de influențe sau interferențe:

- de optimizare – presupune ajustarea sau introducerea de parametri de tranzacție în plus până când performanța strategiei testate este foarte atractivă. Se poate încerca minimizarea ei prin reducerea numărului de parametri și mărirea cantității de date din setul propus.
- de anticipare – această interferență apare când date viitoare sunt incluse accidental într-un punct din simulare unde nu ar fi fost disponibile.
- de supraviețuire – apare când strategiile sunt testate pe seturi de date care nu includ tot ce s-a întâmplat în cazul unor bunuri care ar fi putut fi alese la un moment dat în timp, ci iau în considerare doar cele care au „supraviețuit” până în prezent. De exemplu, datele Yahoo Finance (gratuit, cel mai utilizat de traderi) nu sunt foarte ferite de acest tip de interferență; în cazul datelor de capital este posibilă achiziționarea de seturi de date care conțin entități delistate de pe piață, dar sunt scumpe și tind să fie utilizate doar de instituții.
- cognitivă – afectează cu precădere procesul de backtesting. Orice intervenție umană care are la bază motive psihologice, nu financiare poate avea o influență gravă asupra profitabilității unei strategii, de aceea este cunoscut că în realitate rezultatele nu sunt aproape niciodată la fel de bune precum cele obținute în backtesting.

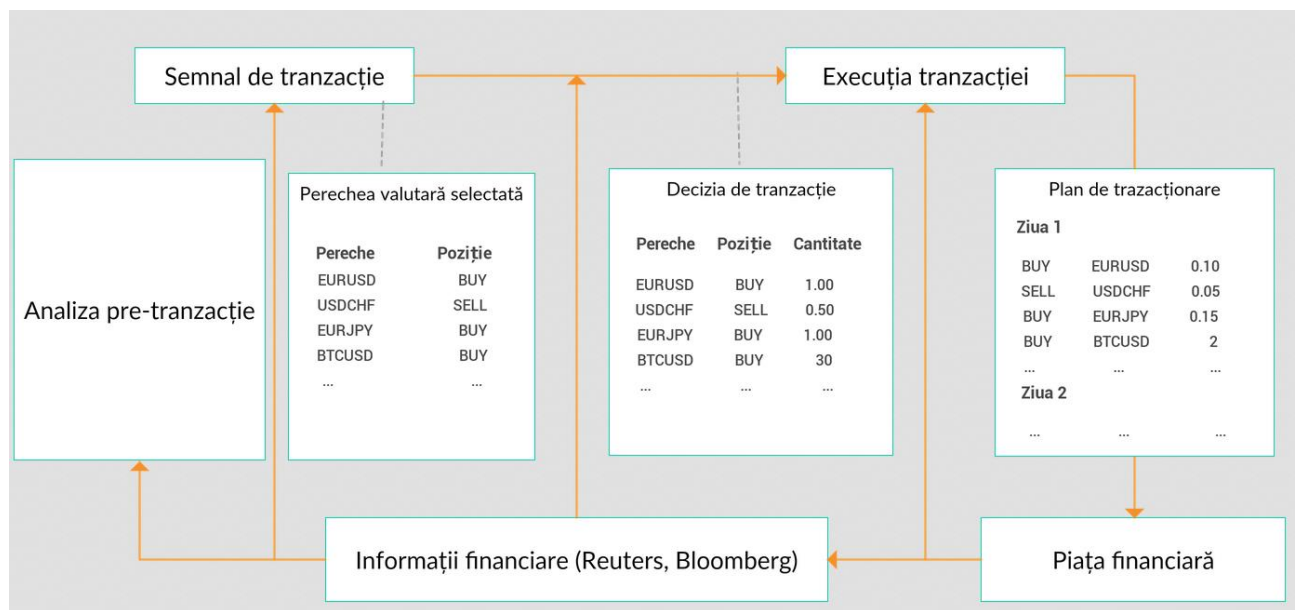


Figura 8 Exemplu de sistem de trading automat

Analiza pre-tranzacție alege un grup de bunuri valutare și trimite informația către componenta de semnale de trading, care determină cantitatea fiecărui bun cu care se face tranzacția respectivă. La final, execuția tranzacției determină planul de tranzacționare prin alegerea caselor de schimb și a cantităților stabilite. Analiza pre-tranzacție este cea mai întâlnită utilizare a algoritmilor într-un mediu comercial și cuprinde orice sistem care utilizează date financiare sau știri pentru a analiza anumite aspecte ale unui activ. Poate fi la fel de ușor folosită ca o metodă de a evalua o companie sau poate implica algoritmi de ultimă generație care utilizează tehnici de inteligență artificială pentru a scana știri sau fluxuri de pe Twitter pentru a anticipa volatilitatea prețurilor activelor.

Următorul pas în automatizarea procesului de tranzacționare este generarea semnalului de tranzacționare folosit adesea de managerii de active și instituțiile de tranzacționare. Acest nivel de sistematizare este în general aplicabil întregii ramuri de tranzacționare la frecvență înaltă (HFT), unde automatizarea completă este o cerință de bază.

Al treilea pas este executarea ordinelor comerciale, tranzacționarea algoritmică putând executa tranzacții și plasa comenzi într-unul sau mai multe schimburi. Un trader poate interveni și lua decizia de tranzacționare manual, caz în care algoritmul optimizează doar execuția (de regulă asociat cu tranzacționarea prin agenție).

Trasarea algoritmică (tranzacționarea automată, tranzacționarea black-box sau pur și simplu algo-tranzacționarea) este procesul de utilizare a calculatoarelor programate să urmeze un set definit de instrucțiuni (un algoritm) pentru plasarea unei tranzacționări pentru a genera profituri la o viteză și o frecvență imposibile pentru un comerciant uman. Seturile de reguli definite se bazează pe momentul, prețul, cantitatea

sau orice model matematic. În afară de oportunitățile de profit pentru comerciant, algo-trading-ul face piețele mai lichide și face comerțul mai sistematic prin excluderea impactului emoțiilor umane asupra activităților de tranzacționare.

Să presupunem că un comerciant respectă aceste criterii comerciale simple:

- Cumpărăm 50 de puncte dintr-o pereche de tranzacționare atunci când media mobilă de 50 de zile depășește media mobilă de 200 de zile. (O medie mobilă este o medie a punctelor de date anterioare care elimină fluctuațiile de preț de la o zi la alta și identifică astfel tendințele.)
- Vindem puncte dintr-un stoc atunci când media mobilă de 50 de zile depășește media mobilă de 200 de zile.

Utilizând acest set de două instrucțiuni simple, este ușor să scriem un program de calculator care să monitorizeze automat prețul punctelor (și indicatorii mediei mobile) și să plaseze ordinele de cumpărare și vânzare atunci când sunt îndeplinite condițiile definite. Comerciantul nu mai trebuie să supravegheze prețurile și graficele live sau să plaseze ordinele manual. Sistemul de tranzacționare algoritmică o face automat pentru el, identificând în mod corect oportunitatea de tranzacționare.

### 3.1.1 Beneficiile tranzacționării algoritmice

Tranzacționarea algoritmică oferă următoarele beneficii:

- Tranzacții executate la cele mai bune prețuri posibile;
- Plasarea rapidă și exactă a ordinelor comerciale (prin urmare, șanse ridicate de execuție la nivelurile dorite);
- Tranzacțiile au fost plasate corect și instantaneu, pentru a evita schimbările semnificative ale prețurilor;
- Reducerea costurilor de tranzacție (a se vedea exemplul de deficit de implementare de mai jos);
- Verificări automate simultane ale mai multor condiții de piață;
- Reducerea riscului de erori manuale în plasarea tranzacțiilor;
- Poate fi backtested, pe baza datelor disponibile și în timp real, pentru a vedea dacă este o strategie comercială viabilă;
- Posibilitatea redusă de greșeli ale comercianților umani pe baza factorilor emoționali și psihologici.

Cea mai mare parte a comerțului cu algoritmi de astăzi este tranzacționarea de înaltă frecvență (HFT), care încearcă să valorifice plasarea unui număr mare de comenzi la viteze foarte rapide pe mai multe piețe și parametrii decizionali multipli, pe baza instrucțiunilor preprogramate.

Tranzacționarea algoritmică este utilizată în multe forme de tranzacționare și activități de investiții, printre care:



- Investitorii pe termen mediu sau lung sau fonduri de investiții (fondurile de pensii, fondurile mutuale, companiile de asigurări) o utilizează pentru a tranzacționa în cantități mari atunci când nu doresc să influențeze prețurile punctelor cu investiții discrete și de volum mare;
- Operatorii pe termen scurt și participanții la vânzare (factorii de decizie, cum ar fi casele de brokeraj, speculatorii și arbitrii) beneficiază de execuția automată a tranzacțiilor; în plus, ajutoarele pentru comercializarea algo-urilor în crearea unei lichidități suficiente pentru vânzătorii de pe piață;
- Traderi sistematici - adepți ai tendințelor, fonduri speculative sau perechi de comercianți (o strategie de tranzacționare neutră pe piață care se potrivește unei poziții lungi cu o poziție scurtă într-o pereche de instrumente foarte corelate, cum ar fi două puncte, fonduri tranzacționate la bursă (ETF) sau valute) etc - este mult mai eficient să programăm regulile de tranzacționare și să lăsăm programul să tranzacționeze automat.

Tranzacționarea algoritmică oferă o abordare mai sistematică tranzacției active decât metodele bazate pe intuiția sau instinctul unui trader uman.

### 3.1.2 Strategii de tranzacționare algoritmică

Orice strategie de tranzacționare algoritmică necesită o oportunitate identificată care să fie profitabilă în ceea ce privește câștigurile îmbunătățite sau reducerea costurilor. Următoarele sunt strategiile comune de tranzacționare folosite în tranzacționarea algo-urilor:

- Strategii de urmărire a tendințelor

Cele mai obișnuite strategii de tranzacționare algoritmice urmăresc tendințele în mediile mobile, creșterile canalelor, mișcările nivelului prețurilor și indicatorii tehnici aferenți. Acestea sunt cele mai ușoare și simple strategii de implementare prin intermediul tranzacțiilor algoritmice, deoarece aceste strategii nu implică realizarea de previziuni sau prognoze de prețuri. Tranzacțiile sunt inițiate pe baza apariției unor tendințe dorite, care sunt ușor și direct de implementat prin algoritmi, fără a intra în complexitatea analizei predictive. Exemplul menționat de mai sus, de a utiliza mediile mobile de 50 și 200 de zile, este o strategie populară care urmărește tendințele.

- Oportunități de arbitraj

Cumpărarea unui instrument de tranzacționare dual-listat la un preț mai mic într-o piață și vânzarea simultană a acestuia la un preț mai mare pe o altă piață oferă diferența de preț ca profit fără risc sau arbitraj. Aceeași operațiune poate fi reprodusă pentru puncte față de instrumentele futures, deoarece diferențele de preț există din când în când. Implementarea unui algoritm pentru identificarea unor astfel de diferențe de preț și plasarea comenzilor permite oportunități profitabile într-o manieră eficientă.

- Rebalansarea fondurilor index

Fondurile indexului au definite perioade de reechilibrare pentru a-și aduce participațiile la paritate cu indicele de referință respectiv. Acest lucru creează oportunități profitabile pentru trader-ri algoritmi care valorifică tranzacțiile așteptate

care oferă profituri de la 20 la 80 de puncte în funcție de numărul de puncte din fondul de bază, chiar înainte de reechilibrarea fondului de bază. Astfel de tranzacții sunt inițiate prin intermediul sistemelor de tranzacționare algoritmică pentru execuția oportună și cele mai bune prețuri.

- Strategii bazate pe modele matematice

Modelele matematice dovedite, cum ar fi strategia de tranzacționare neutră delta, permit tranzacționarea pe o combinație de opțiuni și securitatea ei fundamentală. (Delta neutral este o strategie de portofoliu compusă din mai multe poziții cu compensare delta pozitivă și negativă - un raport care compară variația prețului unui activ, de obicei o garanție comercializabilă, cu modificarea corepunzătoare a prețului derivatului său - astfel încât totalul delta activelor în cauză este zero.)

- Interval de tranzacționare (întoarcerea la valoarea medie)

Strategia de întoarcerea la valoarea medie se bazează pe ideea că prețurile ridicate și scăzute ale unui activ reprezintă un fenomen temporar care revine periodic la valoarea medie. Identificarea și definirea unui interval de preț și implementarea unui algoritm bazat pe acesta permite ca tranzacțiile să fie plasate automat când prețul activului intră în și iese din intervalul definit.

- Preț mediu ponderat în funcție de volum (VWAP)

Strategia preț mediu ponderat în funcție de volum întrerupe o comandă mare și oferă bucăți mai mici, ordonate din punct de vedere al pieței, utilizând profiluri de volum istorice specifice stocurilor. Scopul este de a executa comanda aproape de preț mediu ponderat la volum (VWAP).

- Preț mediu ponderat în timp (TWAP)

Strategia medie ponderată pe timp sparge o comandă mare și eliberează în mod dinamic bucăți mai mici ale ordinului către piață, utilizând sloturi de timp egal împărțite între un început și un moment de sfârșit. Scopul este de a executa ordinul aproape de preț mediu între orele de început și de sfârșit, minimizând astfel impactul pe piață.

- Procentul volumului (POV)

Până la completarea comenzii comerciale, acest algoritm continuă să trimită comenzi parțiale, în funcție de rata de participare definită și în funcție de volumul tranzacționat pe piațe. Strategia "etapelor" aferente trimite comenzi la un procent definit de utilizator din volumele pieței și crește sau scade această rată de participare atunci când prețul punctelor atinge niveluri definite de utilizator.

- Implementarea deficitului

Strategia privind deficitul de implementare vizează minimizarea costului de execuție a unei comenzi prin tranzacționarea pe piața în timp real, economisind astfel costul ordinului și beneficiind de costul de oportunitate al executării întârziate. Strategia va crește rata de participare țintă atunci când prețul punctelor se va mișca favorabil și va scădea atunci când prețul punctelor se va mișca negativ.

- Dincolo de algoritmii obișnuiți de tranzacționare

Există câteva clase speciale de algoritmi care încearcă să identifice "întâmplările" de cealaltă parte. Acești "algoritmi de sniffing" - utilizați, de exemplu, de un producător

de piață de vânzare - au inteligența încorporată pentru a identifica existența unor algoritmi pe partea de cumpărare a unei comenzi mari. O astfel de detectare prin algoritmi va ajuta producătorul de piață să identifice oportunități de comandă mari și să le permită să beneficieze prin completarea comenzilor la un preț mai mare. Acest lucru este uneori identificat ca o tehnologie de nivel înalt.

### 3.1.3 Cerințe tehnice pentru tranzacționarea algoritmică

Implementarea algoritmului folosind un program de calculator este ultima parte, însoțită de backtesting (încercarea algoritmului pe perioadele istorice de performanță a pieței bursiere din trecut pentru a vedea dacă utilizarea acestuia ar fi fost profitabilă). Provocarea este transformarea strategiei identificate într-un proces informatizat integrat care are acces la un cont de tranzacționare pentru plasarea comenzilor. Sunt necesare următoarele:

- Cunoștințe de programare pe calculator pentru a programa strategia de tranzacționare necesară, programatorii angajați sau software-ul de tranzacționare prestabilit;
- Conectivitatea la rețea și accesul la platformele de tranzacționare pentru plasarea comenzilor;
- Accesul la fluxurile de date de pe piață, care vor fi monitorizate de algoritmul de plasare a comenzilor;
- Abilitatea și infrastructura de a proteja sistemul odată ce acesta este construit - înainte de a intra pe piețe reale;
- Datele istorice disponibile pentru backtesting, în funcție de complexitatea regulilor implementate în algoritm.

Cu toate acestea, practica tranzacționării algoritmice nu este atât de simplă să fie menținută și executată. Amintiți-vă, dacă puteți plasa o comandă generată de algoritm, la fel pot face și ceilalți participanți la piață. În consecință, prețurile fluctuează în milli și chiar microsecunde. În exemplul de mai sus, ce se întâmplă dacă tranzacția dvs. de cumpărare este executată, dar vânzarea nu se datorează faptului că prețurile de vânzare se modifică până la momentul în care comanda dvs. atinge piața? Veți ajunge cu o poziție deschisă, ceea ce face ca strategia de arbitraj să fie fără valoare.

Există riscuri și provocări suplimentare: De exemplu, riscurile de eșec al sistemului, erorile de conectivitate în rețea, timpii de întârziere între ordinele de comerț și execuție și, cel mai important, algoritmi imperfecti. Cu cât este mai complex un algoritm, este nevoie de o testare mai stringentă înainte de a fi pusă în practică.

### 3.1.4 Tipuri de strategii de arbitraj și aplicarea lor

Este important să discutăm despre strategia de arbitraj triunghiular, ce este o practică uzuală executată pe piața de schimb valutar. Un aspect important este că procesul acestui arbitraj este organizat gradual și chiar și cea mai nesemnificativă mișcare îl poate forța să depășească limitele arbitrajului triunghiular. Acest lucru se petrece întrucât poate fi privit ca un fenomen sensibil sau substanțial. Ideea este că diferențele de preț care se

produc ocazional te pot ajuta să obții o sumă mai mare decât cea cu care alegi să intri în acest joc. Aceste diferențe menționate anterior există zilnic în diverse piețe dar, cum am specificat încă de la început, ele sunt disponibile doar pentru o perioadă scurtă de timp. Poate te întrebi care sunt pașii pe care trebuie să îi parcurgi la nivelul acestui tip de strategie. Iată etapele unei strategii corecte de arbitraj triunghiular:

- Oferirea valutei inițiale în schimbul celei de-a doua;
- A doua monedă este schimbată cu cea de-a treia;
- A treia monedă este oferită pentru cea inițială.

Pentru a vă oferi un exemplu concludent, vom utiliza trei monede reale: USD, EUR, GBP. Astfel, puteți urmări în rândurile de mai jos pașii arbitrajului triunghiular:

- Se oferă USD în schimbul EUR;
- Se oferă EUR pentru a obține GBP;
- GBP se oferă, de această dată, în schimbul USD.

Există o serie de avantaje pe care traderii sau investitorii le pot obține utilizând strategia de arbitraj triunghiular:

- Pot fi obținute profituri pe investiții foarte mari;
- Riscul este redus considerabil, dar totuși tranzacțiile de investiții îl prezintă la cel mai mic nivel;
- Scopul investiției constă în obținerea profitului și nu a problemelor.

Strategia poate fi adoptată în situații incerte de piață pentru a obține profit. În astfel de situații, este recomandat să dezvoltați investiții pe termen scurt, fiindcă strategia implică faptul că investitorii sunt totuși capabili să obțină profituri.

În plus, nu este nevoie să deții fonduri în valoare de un million de dolari sau conturi la câțiva brokeri FOREX. Poți câștiga suficienți bani prin utilizarea strategiei de arbitraj statistic, tranzacționând micro sau mini loturi și utilizând un singur cont FOREX. Strategia de arbitraj sau tranzacționarea în perechi (numită și tranzacționare de convergență) se bazează pe statistici și inversări. Ceea ce trebuie să faci este să identifici, utilizând un calculator de corelație, două perechi FOREX corelate (privind din perspectivă istorică). În momentul în care corelația dintre cele 2 perechi diferă la nivelul unei anumite valori sau devine slabă, trebuie să cumperi cea mai slabă pereche și să o vinzi pe cea mai puternică. În acest caz, când inversarea este prezentă, numărul de pipși rezultat în urma celor două tranzacții va fi fără îndoială pozitiv. Arbitrajul statistic FOREX necesită o bună percepție a levierului și controlul riscului. De asemenea necesită abilitatea de a analiza instrumentele înalt corelate între ele prin diferite clase de active și o înțelegere completă a modului în care spreadurile trebuie interpretate.

### 3.1.5 Concluzii

Este interesant să mergem pentru automatizarea ajutată de calculatoare, cu scopul de a face bani fără efort. Dar trebuie să vă asigurați că sistemul este testat temeinic și că sunt setate limitele necesare. Traderii analitici ar trebui să ia în considerare învățarea programării și a sistemelor de construcție pe cont propriu, să aibă încredere în

implementarea strategiilor corecte într-o manieră rezonabilă. Utilizarea cu precauție și testarea amănunțită a tranzacțiilor cu algoritmi pot crea oportunități profitabile.

### 3.2 Analize si backtesting in Python

Pentru exemplificare am ales sa fac backtesting pe o perioada de 3 săptămâni având ca instrument de tranzacționare perechea EURUSD

```
In [36]: import pandas as pd
import numpy as np
import seaborn as sbn
import matplotlib.pyplot as plt
from scipy.stats import norm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
%matplotlib nbagg

In [37]: data1 = pd.read_csv("AUD_CAD_Week1.csv")
data2 = pd.read_csv("AUD_CAD_Week2.csv")
data3 = pd.read_csv("AUD_CAD_Week3.csv")

In [38]: data1 = pd.read_csv("EUR_USD_Week1.csv")
data2 = pd.read_csv("EUR_USD_Week2.csv")
data3 = pd.read_csv("EUR_USD_Week3.csv")

In [39]: data = pd.concat([data1,data2,data3])

In [40]: columns = data.columns
print(columns)

Index(['LTid', 'cDealable', 'CurrencyPair', 'RateDateTime', 'RateBid',
      'RateAsk'],
      dtype='object')
```

Figura 9 Script python

```
In [48]: def getProbabilityForPACFEqual(data,time):
l = data.size-time-1
count = 0
for index in range(1,l):
    if (data[index+time]-data[index] == 0):
        count += 1
return (count/l)

In [49]: def getReturns(data):
ret = pd.DataFrame(data=np.array(getPandasChange(columnName=columns[-2],data=data)),columns=["retBid"])
ret2 = (pd.DataFrame(data=np.array(getPandasChange(columnName=columns[-2],data=data)),columns=["retAsk"]))
return pd.concat([ret,ret2],axis=1)

In [50]: ret = getReturns(data)

In [51]: getProbabilityForPACF(ret.iloc[:100000,1],168)
Out[51]: 0.6657451092346065

In [52]: getProbabilityForPACFEqual(ret.iloc[:100000,1],168)
Out[52]: 1.00169286093498e-05

In [53]: getProbabilityForPACF(ret.iloc[:1000,1],42)
Out[53]: 0.5653082549634274

In [54]: getProbabilityForPACF(ret.iloc[:1000,1],190)
Out[54]: 0.4437577255871446
```

Figura 10 Script python(cont.)

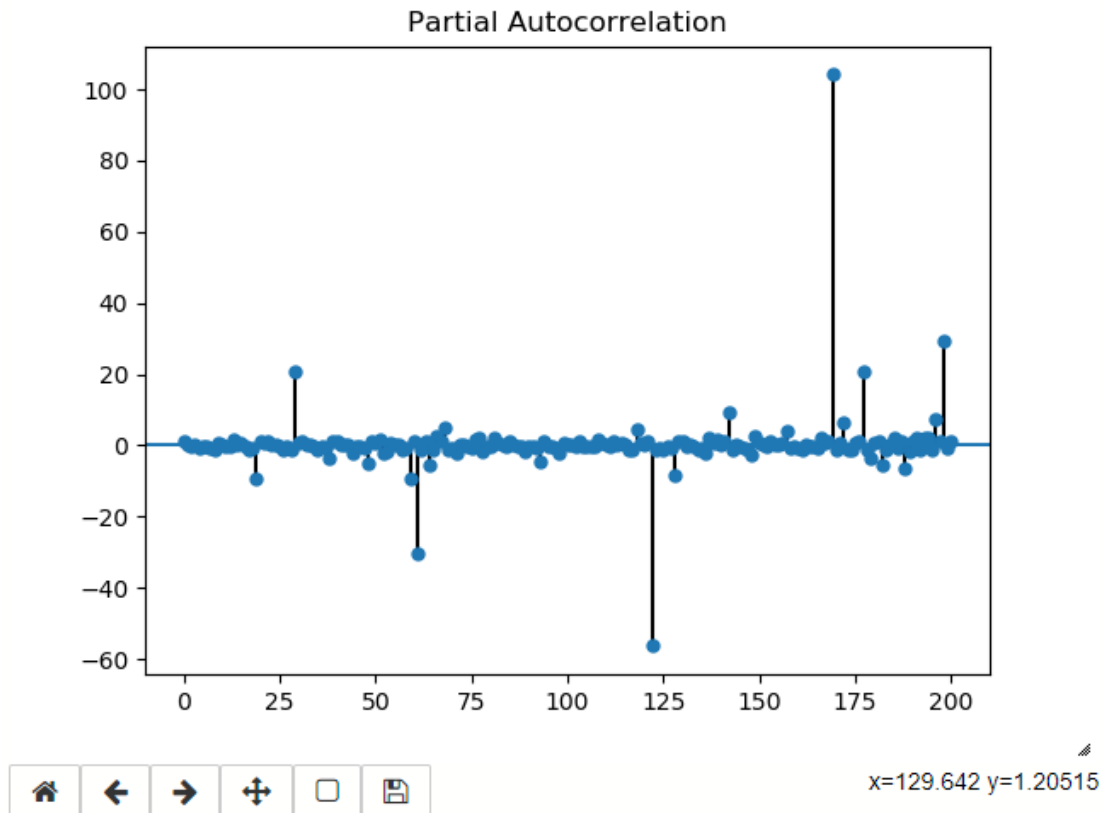


Figura 11 Corelație parțială

În mod normal funcțiile PAC se aplică pe două perechi de tranzacționare diferite. Spre exemplu putem calcula corelația dintre EURJPY și GBPJPY. Calculând cele două corelații putem determina prin backtesting dacă există o corelație între EURGBP. Altfel spus, încercăm să determinăm evoluția cursului în funcție de alte două perechi legate între ele.

În exemplul de mai sus, am aplicat corelația pe același instrument, însă la intervale de timp diferite. Există motive să cred că acest algoritm ajustat ar putea să aibă rezultate incurajatoare. Exemplul de față nu a dat rezultatele scontate dar este un început.

```
[ 'sell ', 0, ' at ', 0.9954519516578233, 'suma ', 100.0]
[ 'sell ', 0, ' at ', 1.0, 'suma ', 100.0]
[ 'buy ', 0.3500000000000001, ' at ', 1.0, 1.0, 'suma ', 65.0]
[ 'sell ', 35.00000000000001, ' at ', 1.0055410439772192, 'suma ', 100.0]
[ 'buy ', 0.35179091574966787, ' at ', 1.0055410439772192, 1.0055410439772192, 'suma ', 64.82090842503322]
[ 'sell ', 35.17909157496678, ' at ', 1.0, 'suma ', 100.19492889349553]
[ 'sell ', 0, ' at ', 1.0045172992026283, 'suma ', 100.19492889349553]
[ 'buy ', 0.35146152011719434, ' at ', 1.0045172992026283, 1.0045172992026283, 'suma ', 64.9802668765534]
[ 'sell ', 35.214662016942135, ' at ', 1.0010191410097202, 'suma ', 100.35400405814549]
[ 'sell ', 0, ' at ', 0.9955030150240178, 'suma ', 100.35400405814549]
[ 'sell ', 0, ' at ', 1.0, 'suma ', 100.35400405814549]
[ 'buy ', 0.3500000000000001, ' at ', 1.0, 1.0, 'suma ', 65.23010263779456]
[ 'sell ', 35.12390142035093, ' at ', 1.0, 'suma ', 100.35400405814549]
[ 'buy ', 0.3500000000000001, ' at ', 1.0, 1.0, 'suma ', 65.23010263779456]
[ 'sell ', 35.12390142035093, ' at ', 0.9943273284783442, 'suma ', 100.35400405814549]
[ 'buy ', 0.3481458638501273, ' at ', 0.9943273284783442, 0.9943273284783442, 'suma ', 65.41617262450325]
[ 'sell ', 34.937831433642245, ' at ', 0.9986398398603912, 'suma ', 100.15581321674345]
[ 'buy ', 0.34955734587412957, ' at ', 0.9986398398603912, 0.9986398398603912, 'suma ', 65.14561297483354]
[ 'sell ', 35.01020024190991, ' at ', 1.005705034307301, 'suma ', 100.10819373789468]
```

Figura 12 indicații trading calculate automat

O alta strategie cunoscută este așa numita “Fade the move”. Această strategie presupune identificarea unei posibilități de a intra într-o poziție inversă trendului general al pieței, chiar dacă graficele nu indică acest lucru

2018-08-22 05:10:10,552 INFO [main] - Executing request : GET /pricing?instruments=EUR\_USD%2CAUD\_JPY HTTP/1.1

2018-08-22 05:10:11,118 INFO [main] - ASK Object: {"price":"1.15783","liquidity":10000000}

2018-08-22 05:10:11,118 INFO [main] - ASK Object: {"price":"81.086","liquidity":10000000}

2018-08-22 05:10:11,136 INFO [main] - Executing request : GET /pricing?instruments=AUD\_CHF HTTP/1.1

2018-08-22 05:10:11,672 INFO [main] - ASK Object: {"price":"0.72403","liquidity":10000000}

The strategy signalled to go LONG on instrument AUD\_JPY at limit price 83.50300 and take profit 83.60300

The strategy signalled to go SHORT on instrument AUD\_CHF at limit price 0.71420 and take profit 0.71320



Figura 13 Strategia „Fade the move”

### 3.3 Diagrame UML

Deși conceptul este simplu implementarea aplicației s-a dovedit mai complexă decât părea la prima vedere. Diagramele prezentate în continuare nu cuprind toate elementele claselor pentru că ar fi devenit mult prea mari, vizualizarea lor devenind greoaie.

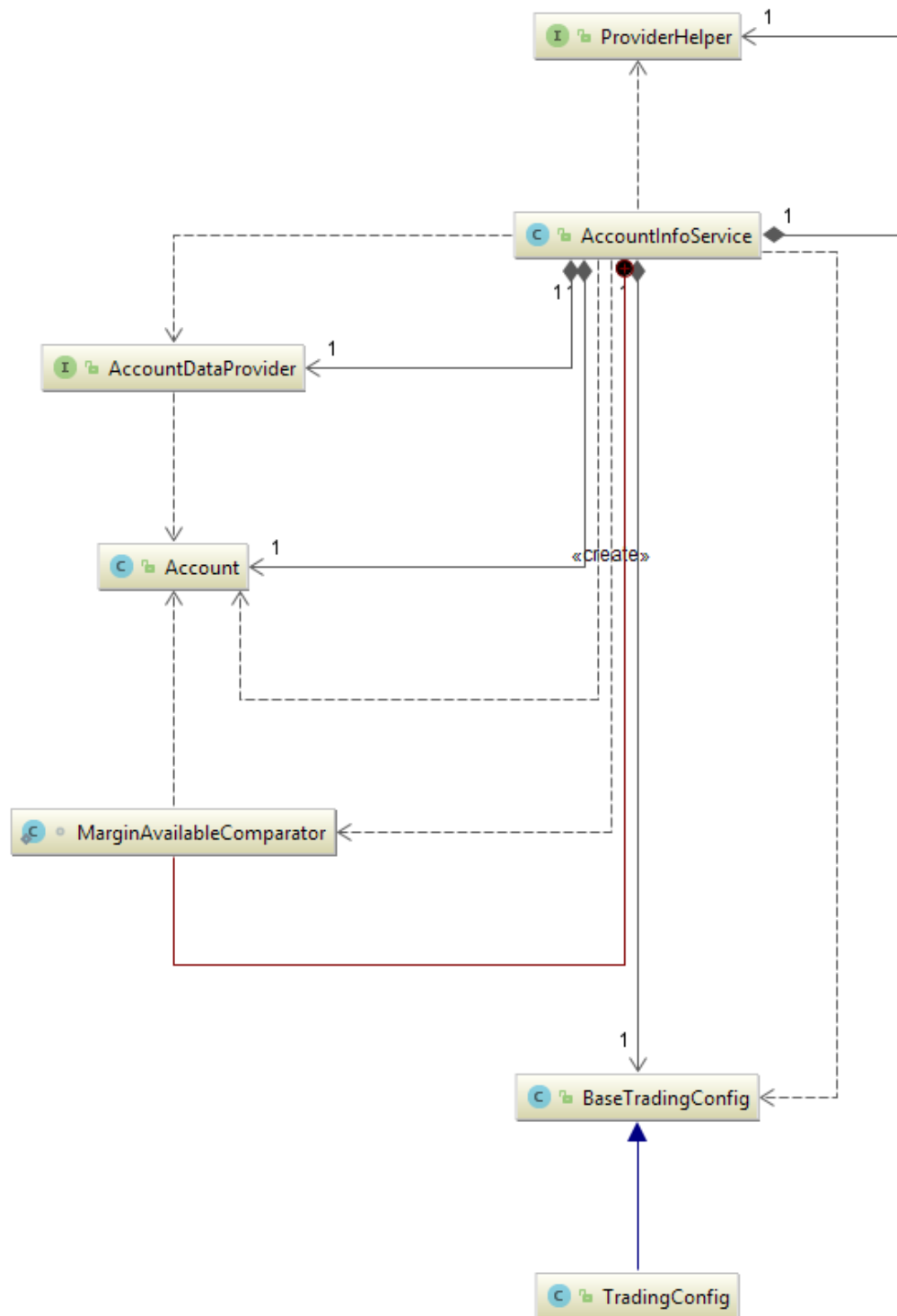
Pentru analiza datelor este necesară conectarea cu un provider de informații, acesta fiind brokerul. Deși acțiunile ce se pot efectua pe platformele acestora sunt de multe ori asemănătoare implementarea API-ului diferă, astfel fiind necesară o configurare sau dezvoltare de componente în plus pentru fiecare broker.

#### 3.3.1 Pachetul tradingAPI

Pachetul tradingAPI conține următoarele pachete:

- Account – clase necesare conectării
- Events – clase pentru evenimente
- Heartbeat – clase pentru serviciul heartbeat
- Instruments – clase necesare pentru instrumente
- Market – clase necesare pentru obținerea prețurilor
- marketData – clase necesare pentru informații din piață(candele, MA, pipJump)
- order – clase necesare sistemului de ordine
- streaming – clase necesare serviciului de streaming(heartbeat/marketDat)
- trade – clase necesare tranzacțiilor(semnale, decizii, direcții)
- util – constante și clase ajutătoare





Powered by yFiles

Figura 14 Diagrama UML pachetul account

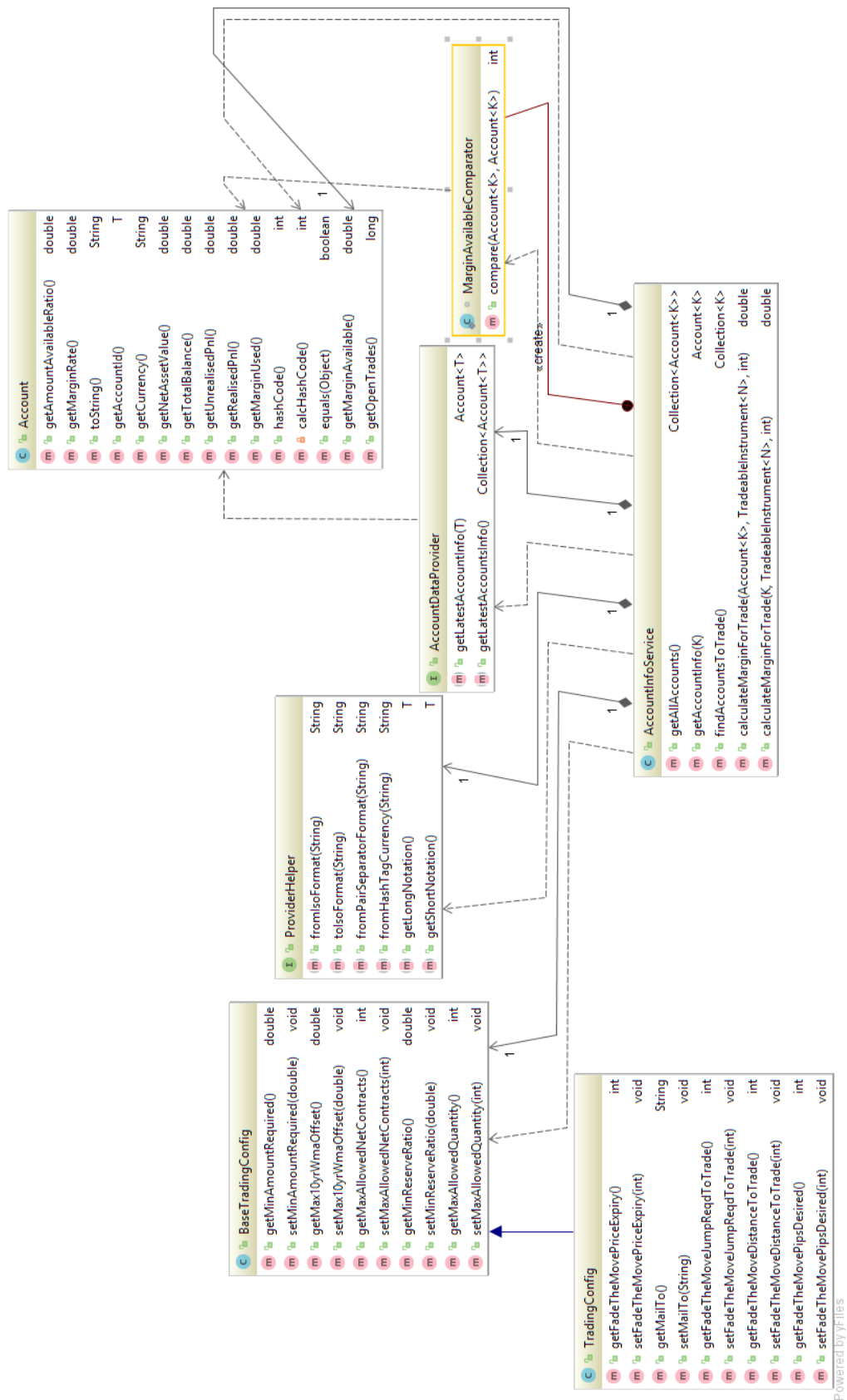


Figura 15 Diagrama UML pachetul account în amănunt

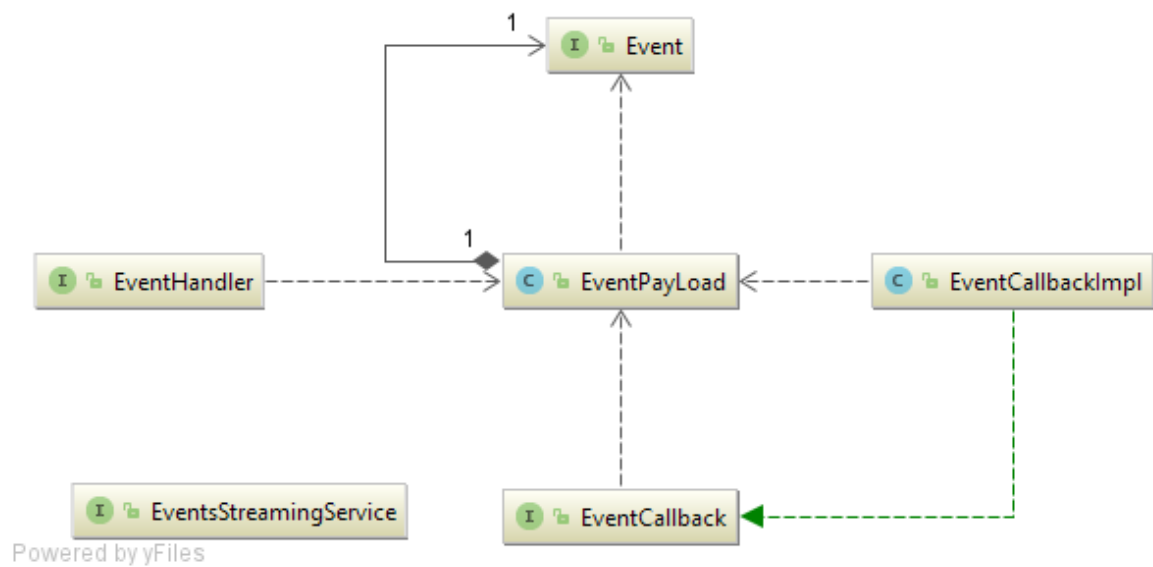


Figura 16 Pachetul event

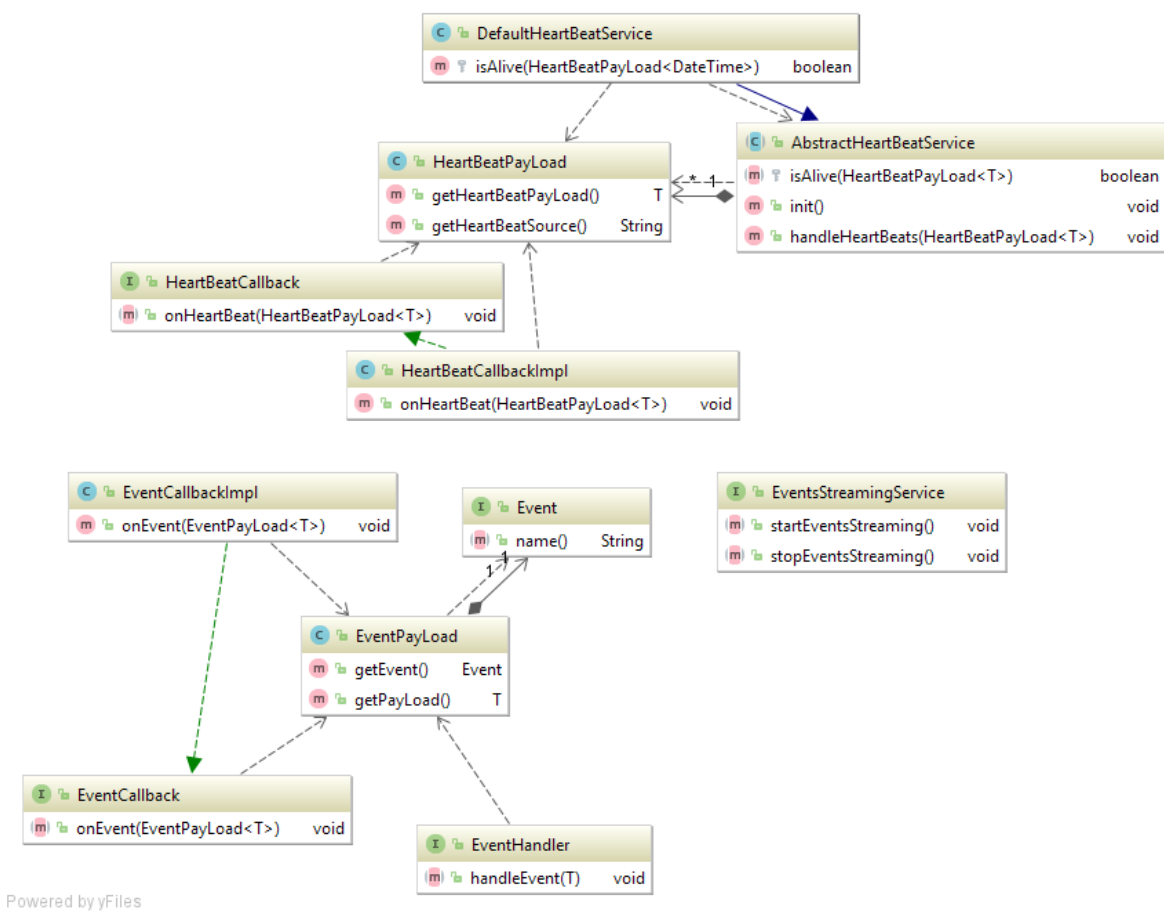


Figura 17 Diagrama UML pachetele event și heartbeat in amanunt

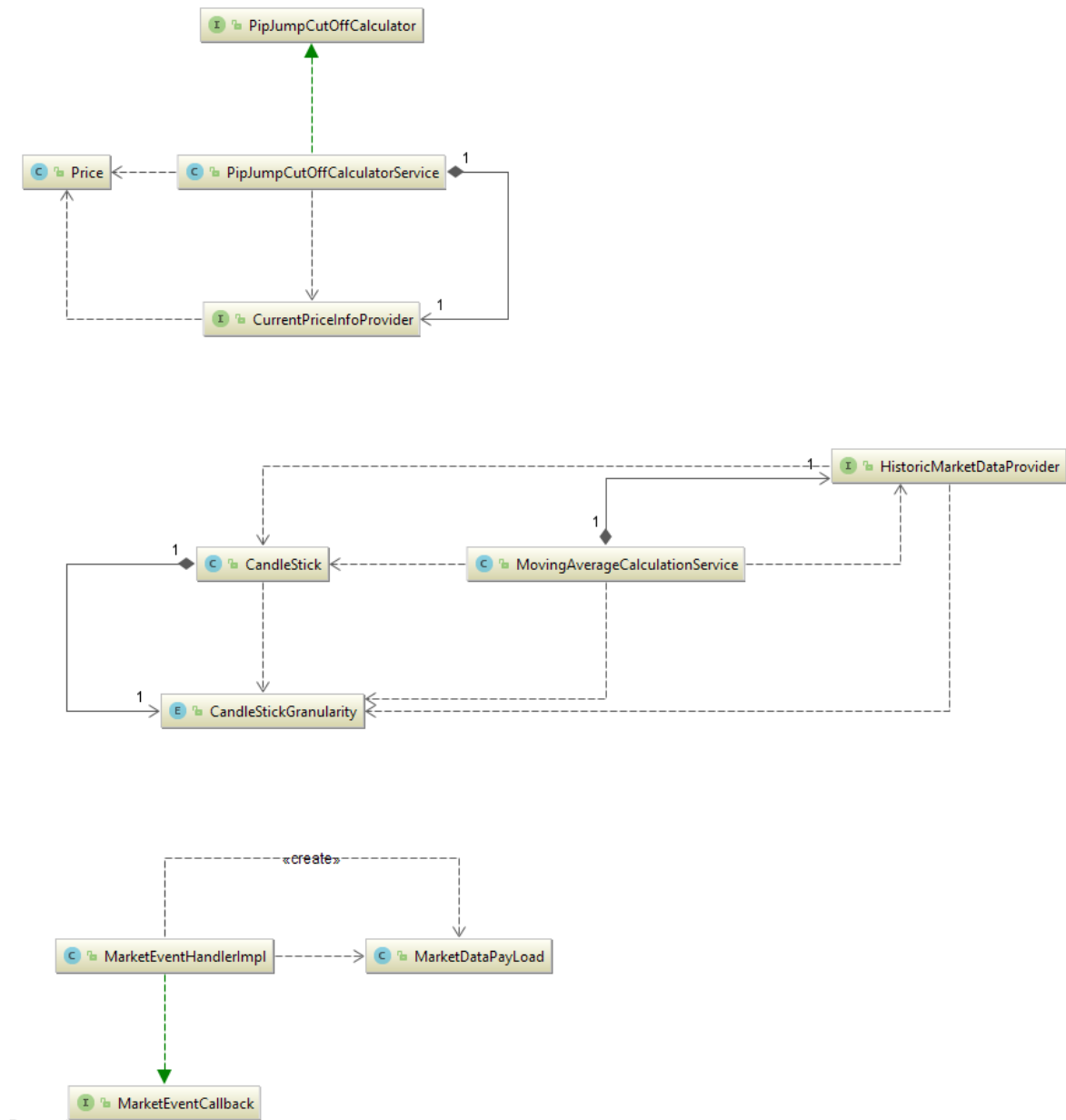


Figura 18 Diagrama UML pachetul market

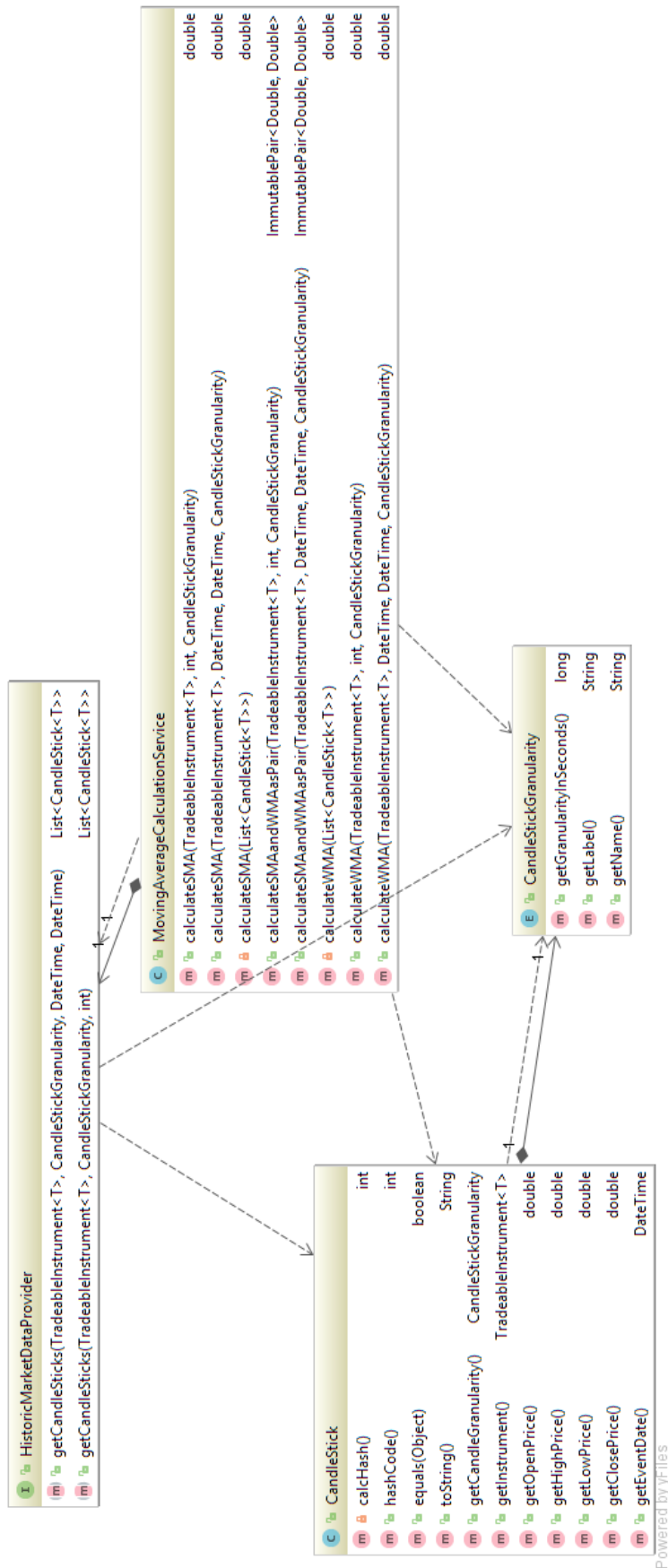


Figura 19 Diagrama UML pachetul market în amănunt

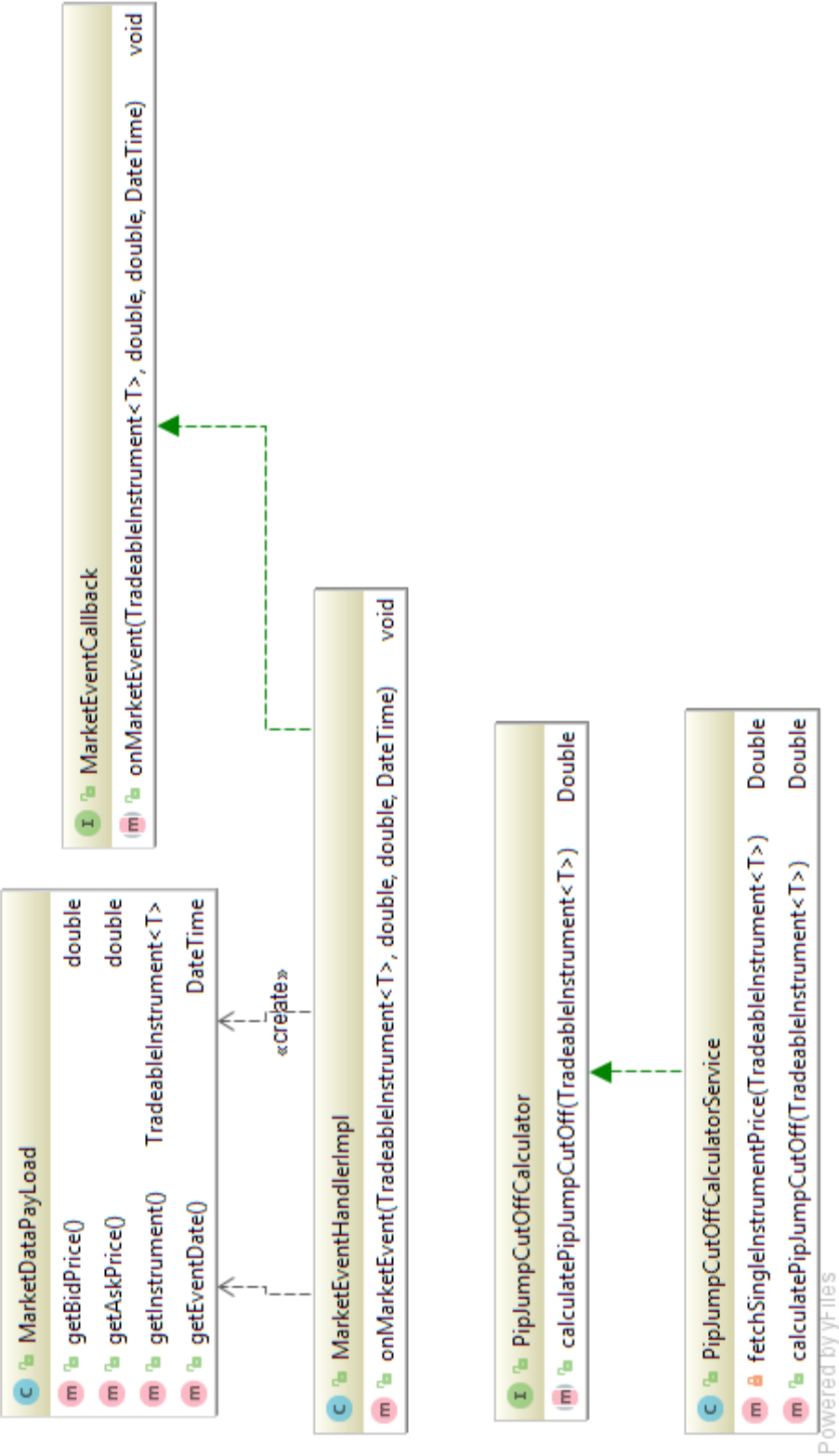


Figura 20 Diagrama UML pachetul market in amanunt(cont.)

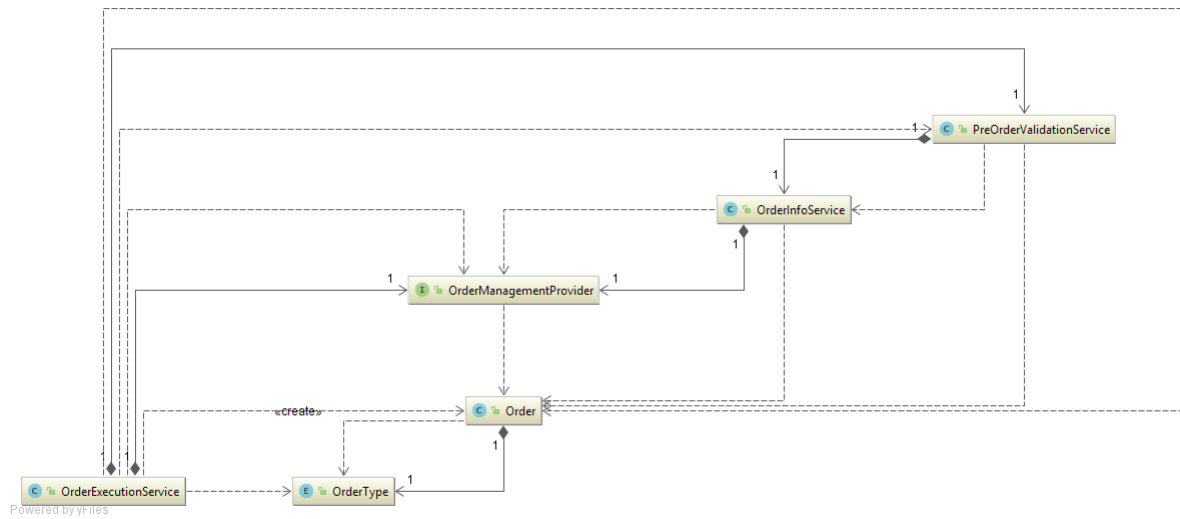


Figura 21 Diagrama UML pachetul order

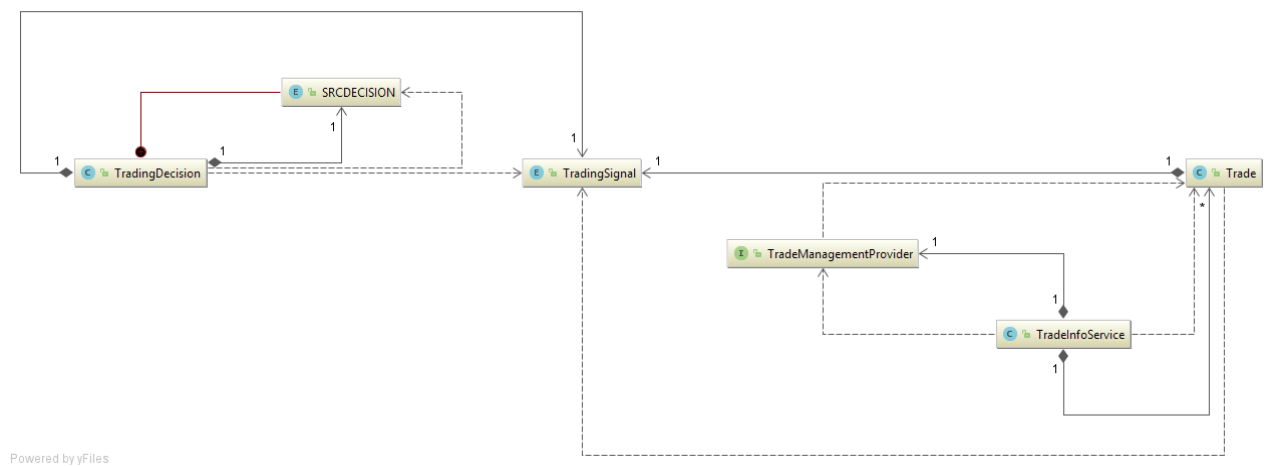


Figura 22 Diagrama UML pachetul trade

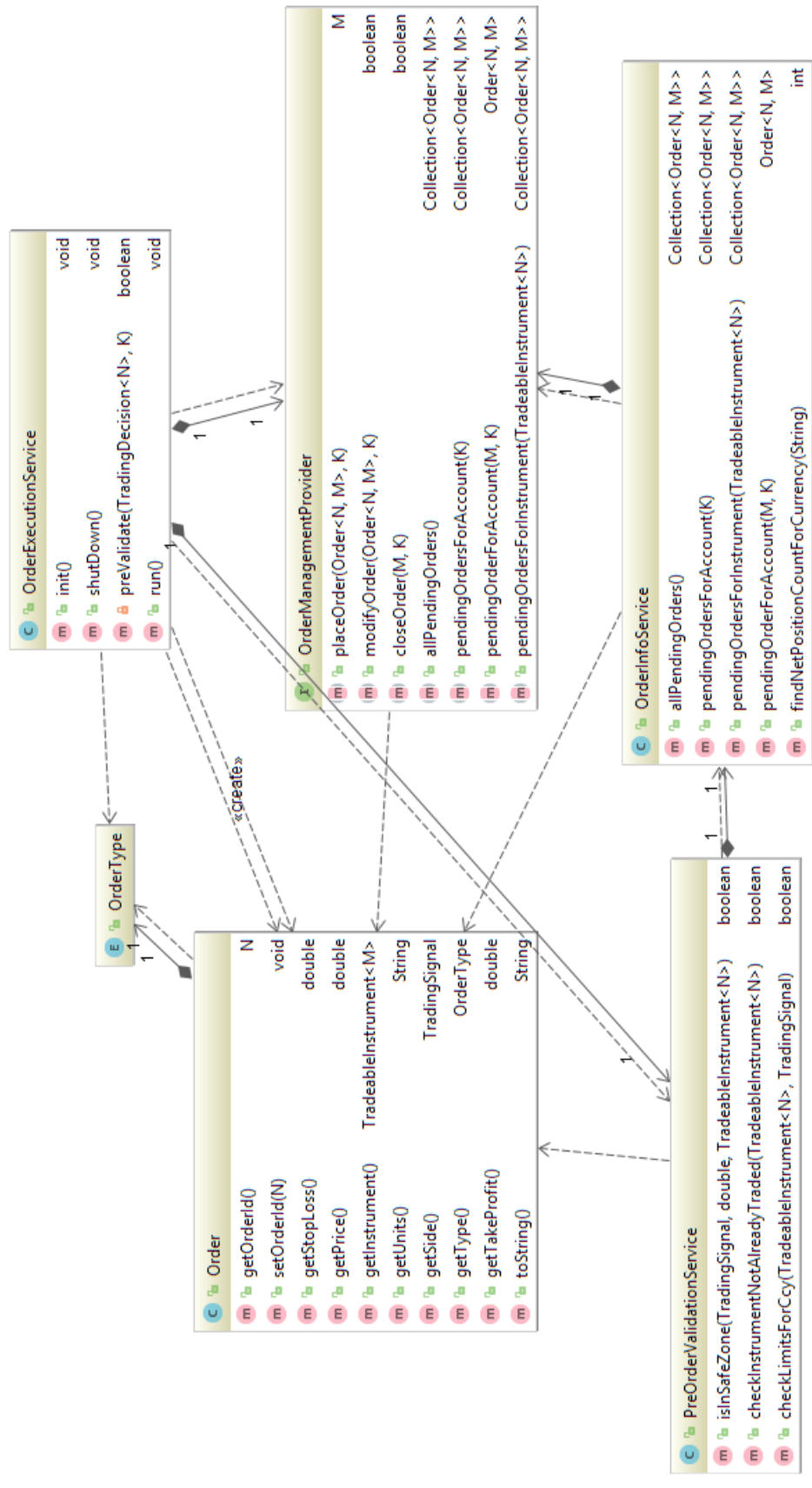


Figura 23 Diagrama UML pachetul order în amănunt



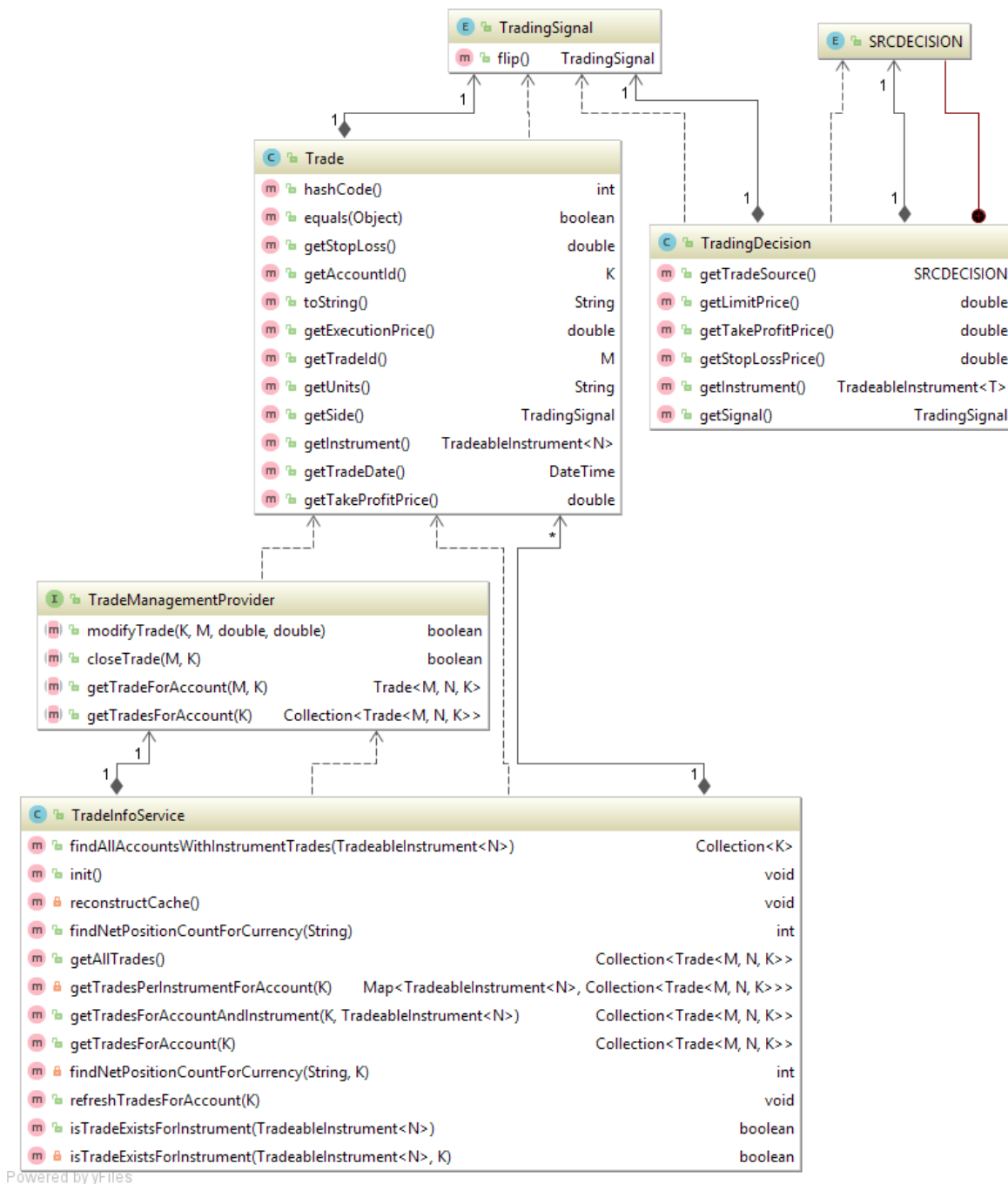
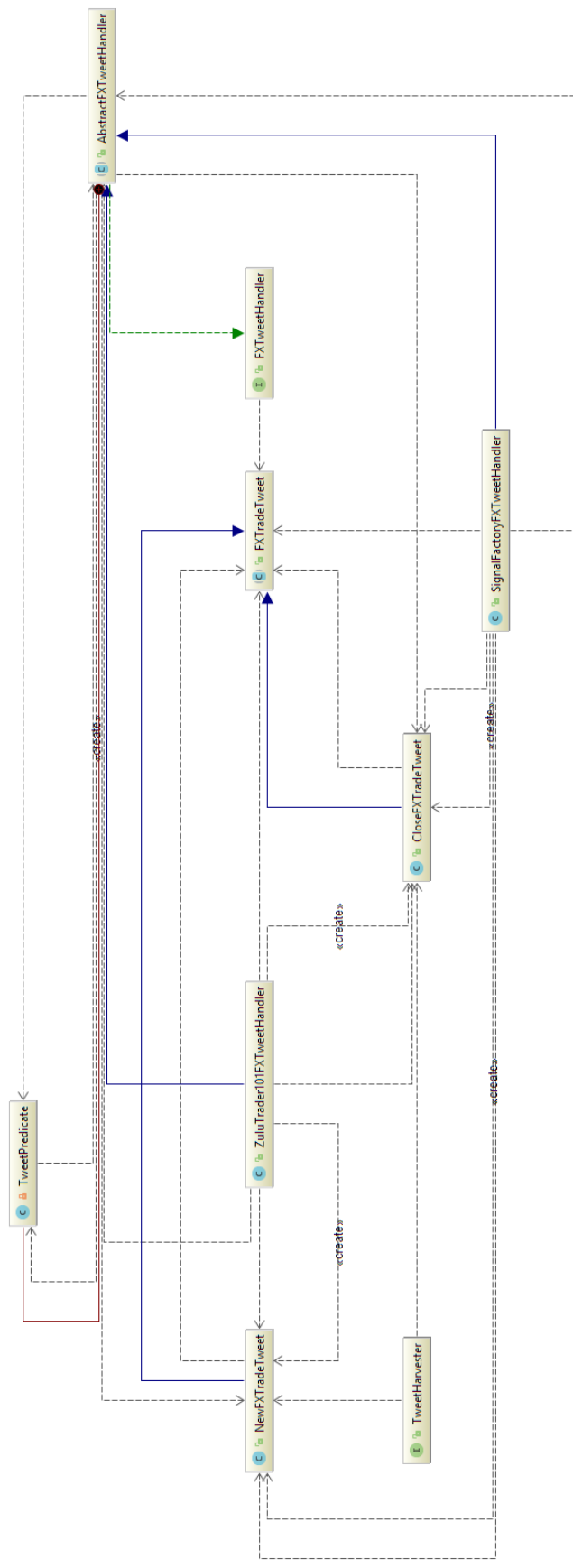


Figura 24 Diagrama UML pachetul trade în amănunt



Powered by yFiles

Figura 25 Diagrama UML pachetul twitter



Figura 26 Diagrama UML pachetul strategy

### 3.4 MOD FUNCȚIONARE

Aplicația se conectează la brokerul ales și extrage informațiile necesare analizei în timp real. Se face o scanare a tuturor instrumentelor de tranzacționare disponibile:

- Executing request : GET /instruments HTTP/1.1

```
-      ++++++ Dumping      Instrument      Info
+++++
```

- TradeableInstrument [instrument=GBP\_NZD, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=EUR\_GBP, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=GBP\_HKD, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=GBP\_CHF, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=UK10YB\_GBP, description=null, instrumentId=null, pip=0.01]

- TradeableInstrument [instrument=GBP\_JPY, description=null, instrumentId=null, pip=0.01]

- TradeableInstrument [instrument=GBP\_SGD, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=UK100\_GBP, description=null, instrumentId=null, pip=1.0]

- TradeableInstrument [instrument=GBP\_CAD, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=GBP\_PLN, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=XAG\_GBP, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=GBP\_AUD, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=GBP\_USD, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=GBP\_ZAR, description=null, instrumentId=null, pip=1.0E-4]

- TradeableInstrument [instrument=XAU\_GBP, description=null, instrumentId=null, pip=0.01]

```
-      ++++++Finished      Dumping      Instrument      Info
+++++
```

- Pip for instrument EUR\_AUD is 0.00010

- Pip for instrument USD\_JPY is 0.01000

- Pip for instrument USD\_ZAR is 0.00010

Scripturile Python analizează această informație și emit semnale oricând identifică posibilități de profit. Pe baza acestora aplicația creează automat ordine de intrare în piață.

Ordinele create au ca parametri instrumentul de tranzacționare, cursul la care se deschide poziția, momentul de oprire al poziției fie el pozitiv sau negativ. În cazul unui trade de succes, care evoluează conform așteptărilor algoritmului ce l-a determinat, aplicația va altera targetul ordinului cât și cel de stop loss și, astfel obținându-se un profit cât mai mare până în momentul în care direcția cursul se inversează. Acest mecanism mai este cunoscut și sub denumirea de trailing stop loss. Mecanismul de trailing stop loss, nu este oferit de toți brokeri, sau nu este oferit mereu pentru tranzacționarea automată. Uneori acesta este oferit pentru un alt tip de cont, pentru care se plătește o sumă în plus.

Ordinele sunt plasate foarte repede în piață. Se pot crea până la 4 ordine pe secundă. În piața forex cursul reacționează imediat, și se modifică în fiecare secundă, de aceea o conexiune rapidă la internet este de preferat. Totuși, în cazul în care se tranzacționează pe intervale mai mari de timp, acest lucru devine mai puțin important. Spre exemplu, dacă se tranzacționează la interval de 1 minut sau 5 minute latența este critică pentru a asigura pierderi cât mai mici. Dacă se tranzacționează în schimb pe perioade mari, de 4 ore sau mai mari importanța latenței scade.

```
Executing request : GET
/instruments/USD_CHF/candles?granularity=D&alignmentTimezone=GMT&dailyAlignment=0
&price=M&count=15 HTTP/1.1
- ++++++ Last 15 Candle Sticks with Daily Granularity for USD_CHF ++++++
- Open=0.99466, high=0.99852, low=0.99422,close=0.99640,date=2018-07-06T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99648, high=0.99682, low=0.99307,close=0.99526,date=2018-07-07T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99526, high=0.99708, low=0.99198,close=0.99332,date=2018-07-08T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99331, high=0.99513, low=0.98946,close=0.99402,date=2018-07-09T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99400, high=0.99754, low=0.99284,close=0.99512,date=2018-07-10T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99446, high=0.99530, low=0.99260,close=0.99442,date=2018-07-12T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99443, high=0.99543, low=0.99216,close=0.99342,date=2018-07-13T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99341, high=0.99509, low=0.99012,close=0.99426,date=2018-07-14T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99426, high=0.99829, low=0.99264,close=0.99344,date=2018-07-15T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99344, high=0.99750, low=0.99161,close=0.99736,date=2018-07-16T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
```

---



---

```

- Open=0.99736, high=0.99776, low=0.99326,close=0.99580,date=2018-07-17T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99572, high=0.99574, low=0.99494,close=0.99534,date=2018-07-19T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99535, high=0.99686, low=0.99042,close=0.99044,date=2018-07-20T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.99050, high=0.99069, low=0.98427,close=0.98437,date=2018-07-21T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Open=0.98436, high=0.98440, low=0.98368,close=0.98403,date=2018-07-18T00:00:00.000Z,
instrument=TradeableInstrument [instrument=USD_CHF, description=null, instrumentId=null,
pip=0.0], granularity=D
- Executing request : GET
/instruments/GBP_AUD/candles?granularity=M&alignmentTimezone=GMT&from=2018-07-
22T01:22:02.879Z&dailyAlignment=0&to=2018-07-18T01:22:02.879Z&price=M HTTP/1.1
- ++++++Candle Sticks From 2018-07-22T01:22:02.879Z To 2018-07-18T01:22:02.879Z
with Monthly Granularity for GBP_AUD ++++++
- Open=1.78055, high=1.79584, low=1.75683,close=1.77324,date=2018-06-30T00:00:00.000Z,
instrument=TradeableInstrument [instrument=GBP_AUD, description=null, instrumentId=null,
pip=0.0], granularity=M
- Open=1.77324, high=1.78253, low=1.72832,close=1.75394,date=2018-07-31T00:00:00.000Z,
instrument=TradeableInstrument [instrument=GBP_AUD, description=null, instrumentId=null,
pip=0.0], granularity=M

```

Configurarea se face completând fișierele de proprietăți existente în aplicație, completând datele de conectare către API-ul brokerului, cât și endpointurile acestuia. În caz de erori, crearea de ordine noi se sistează, pentru a nu risca să pierdem capital.

## CAPITOLUL 4. MIJLOACE DE IMPLEMENTARE A DESIGNULUI

### 4.1 Transferul de stat reprezentativ (REST)

REST este un stil hibrid derivat din mai multe stiluri arhitecturale bazate pe rețea descrise și combinat cu constrângeri suplimentare care definesc o interfață uniformă a conectorului. Cadrul de arhitectură software este folosit pentru a defini elementele arhitecturale ale REST și pentru a examina procesul de eșantionare, conectorul și vizualizările de date ale arhitecturilor prototipic

#### 4.1.1 Derivarea REST

Rațiunea de proiectare din spatele arhitecturii Web poate fi descrisă printr-un stil arhitectural format din setul de constrângeri aplicate elementelor din arhitectură. Examinând impactul fiecărei constrângeri pe măsură ce este adăugat stilului în evoluție, putem identifica proprietățile induse de constrângerile Web. Restricțiile suplimentare pot fi aplicate pentru a forma un nou stil arhitectural care să reflecte mai bine proprietățile dorite ale unei arhitecturi Web moderne. Această secțiune oferă o imagine de ansamblu generală a REST prin umblarea prin procesul de derivare a acesteia ca stil arhitectural.

##### 4.1.1.1 Începând cu stilul Null

Există două perspective comune asupra procesului de proiectare arhitecturală, fie că este vorba despre clădiri sau despre software. Primul este că un designer începe cu nimic - o idee, o tablă albă sau o placă de desen - și construiește o arhitectură de la componentele familiare până când satisface nevoile sistemului dorit. Al doilea este că un designer începe cu nevoile sistemului în ansamblul său, fără constrângeri, și apoi identifică treptat și aplică constrângeri la elementele sistemului pentru a diferenția spațiul de proiectare și pentru a permite forțelor care influențează comportamentul sistemului să curgă în mod natural, în armonie cu sistemul. În cazul în care prima subliniază creativitatea și viziunea nelimitată, al doilea accentuează restrângerea și înțelegerea contextului sistemului. REST a fost dezvoltat folosind acest ultim proces.

Stilul Null (Figura 1) este pur și simplu un set gol de constrângeri. Din perspectivă arhitecturală, stilul nul descrie un sistem în care nu există limite distincte între componente. Acesta este punctul de plecare pentru descrierea REST.



Figura 27 Stilul Null

#### 4.1.1.2 Client-Server

Primele constrângeri aduse stilului nostru hibrid sunt cele ale stilului arhitectural client-server (Figura 2). Separarea preocupărilor este principiul din spatele constrângerilor client-server. Prin separarea preocupărilor legate de interfața cu utilizatorul de problemele de stocare a datelor, îmbunătățim portabilitatea interfeței cu utilizatorul pe mai multe platforme și îmbunătățim scalabilitatea prin simplificarea componentelor serverului. Poate că cel mai semnificativ pentru Web este totuși că separarea permite componentelor să evolueze independent, susținând astfel cerința mai multor domenii organizaționale de scalare a Internet.

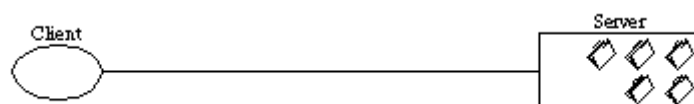


Figura 28 Client-Server

#### 4.1.1.3 Stateless (fără stare)

Apoi adăugăm o constrângere interacțiunii client-server: comunicarea trebuie să aibă un caracter lipsit de stare, ca în stilul client-stateless-server (CSS) (Figura 3), astfel încât fiecare solicitare de la client la server trebuie să conțină toate informațiile necesare pentru a înțelege solicitarea și nu poate profita de niciun context stocat pe server. Starea sesiunii este, prin urmare, păstrată în întregime la client.

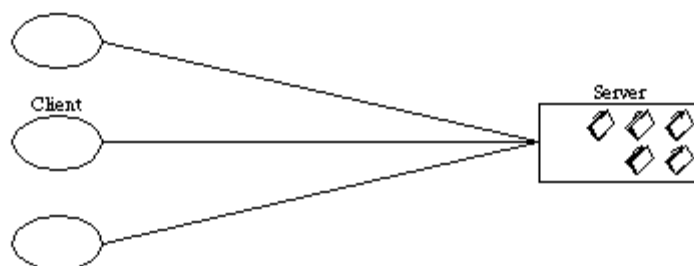


Figura 29 Client - Stateless - Server

Această constrângere induce proprietățile vizibilității, fiabilității și scalabilității. Vizibilitatea este îmbunătățită deoarece sistemul de monitorizare nu trebuie să privească mai mult decât un singur punct de solicitare pentru a determina natura completă a cererii. Fiabilitatea este îmbunătățită deoarece facilitează recuperarea de la eșecuri parțiale. Scalabilitatea este îmbunătățită deoarece nu trebuie stocată starea între cereri, permite componentei serverului să elibereze rapid resurse și simplifică implementarea, deoarece serverul nu trebuie să gestioneze utilizarea resurselor între solicitări.

Ca majoritatea alegerilor arhitecturale, constrângerea stateless reflectă un compromis de proiectare. Dezavantajul este că poate scădea performanța rețelei prin



creșterea datelor repetitive trimise într-o serie de solicitări, deoarece aceste date nu pot fi lăsate pe server într-un context comun. În plus, plasarea stării aplicației pe partea clientului reduce controlul serverului asupra comportamentului consistent al aplicației, deoarece aplicația devine dependentă de implementarea corectă a semanticii în mai multe versiuni client.

#### 4.1.1.4 Cache

Pentru a îmbunătăți eficiența rețelei, adăugăm constrângeri de cache pentru a forma stilul client-cache-stateless-server (Figura 4). Restricțiile de tip cache necesită ca datele dintr-un răspuns la o solicitare să fie etichetate implicit sau explicit ca date care pot fi păstrate în cache sau care nu pot fi păstrate în cache. Dacă un răspuns este disponibil în memorie, atunci cache-ul unui client are dreptul de a reutiliza datele de răspuns pentru solicitările ulterioare, echivalente.

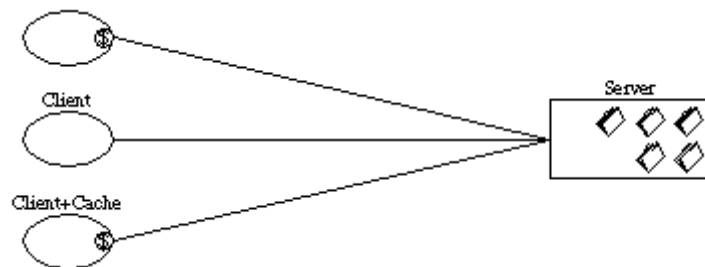


Figura 30 Client + Cache – Stateless - Server

Avantajul adăugării constrângerilor de cache este că acestea au potențialul de a elimina parțial sau complet unele interacțiuni, îmbunătățind eficiența, scalabilitatea și performanța percepută de utilizator prin reducerea latenței medii a unei serii de interacțiuni. Compromisul este, totuși, că o memorie cache poate scădea fiabilitatea dacă datele rămase în cache diferă semnificativ de datele pe care le-ar fi obținut dacă cererea ar fi fost trimisă direct pe server.

Arhitectura Web timpurie, a fost definită de setul de constrângeri client-cache-stateless-server. Adică, rațiunea de proiectare prezentată pentru arhitectura Web înainte de 1994 sa axat pe interacțiunea client-server fără stare pentru schimbul de documente statice pe Internet. Protocoalele pentru interacțiunile de comunicare au avut suport rudimentar pentru cache-urile nerepartizate, dar nu au constrâns interfața cu un set consistent de semantică pentru toate resursele. În schimb, Web-ul sa bazat pe utilizarea unei biblioteci comune de implementare a unui client-server (CERN libwww) pentru a menține coerența între aplicațiile Web.

## 4.2 NumPy

NumPy este o bibliotecă Python care oferă un obiect matrice, diverse obiecte derivate și un sortiment de rutine pentru operații rapide pe matrice, inclusiv manipulare matematică, logică, formă, sortare, selectare, I / O , transformări discrete Fourier, algebră liniară de bază, operații statistice de bază, simulare aleatorie și multe altele.

În nucleul pachetului NumPy, este obiectul `ndarray`. Aceasta cuprinde  $n$ -matrici de tipuri de date omogene, multe operații fiind efectuate în cod compilat pentru performanță. Există mai multe diferențe importante între matricele NumPy și secvențele Python standard:

- Matricile NumPy au o dimensiune fixă la creare, spre deosebire de listele Python (care pot crește dinamic). Modificarea dimensiunii unui `ndarray` va crea un nou tablou și va șterge originalul.
- Elementele dintr-o matrice NumPy trebuie să fie de același tip de date și astfel vor avea aceeași dimensiune în memorie. Excepția: se pot folosi relee de obiecte (Python, inclusiv NumPy), permițând astfel o serie de elemente de dimensiuni diferite.
- Matricile NumPy facilitează procese matematice avansate și alte tipuri de operații pe un număr mare de date. În mod obișnuit, astfel de operațiuni sunt executate mai eficient și cu un cod mai mic decât este posibil utilizând secvențele Python încorporate.
- O multitudine crescândă de pachete științifice și matematice bazate pe Python utilizează matrice NumPy; deși acestea suportă în mod tipic intrarea în secvența Python, ele convertesc astfel de intrări în matricea NumPy înainte de procesare și deseori produc matrice NumPy.

Punctele despre dimensiunea și viteza secvenței sunt deosebit de importante în calculul științific. Ca un exemplu simplu, se ia în considerare cazul înmulțirii fiecărui element într-o secvență 1-D cu elementul corespunzător într-o altă secvență de aceeași lungime. Dacă datele sunt stocate în două liste Python `a` și `b` am putea itera peste fiecare element:

```
c = []
for i in range(len(a)):
    c.append(a[i]*b[i])
```

Acest lucru produce răspunsul corect, dar dacă fiecare dintre `a` și `b` conține milioane de numere, vom plăti prețul pentru ineficiența execuției în buclă din Python. Am putea realiza aceeași sarcină mult mai rapid în C scriind:

```
for (i = 0; i < rows; i++) {
    c[i] = a[i]*b[i];
}
```

Aceasta salvează toate probleme implicate în interpretarea codului Python și manipularea obiectelor Python, dar în detrimentul beneficiilor obținute din codarea în Python. Mai mult, munca de codificare necesară crește cu dimensiunile datelor noastre. În cazul unei matrice 2-D, de exemplu, codul C (prescurtat ca înainte) se extinde la

```

for (i = 0; i < rows; i++): {
    for (j = 0; j < columns; j++): {
        c[i][j] = a[i][j]*b[i][j];
    }
}

```

NumPy ne oferă cele mai bune din ambele lumi: operațiile element-cu-element sunt "modul implicit" atunci când este implicat un ndarray, dar operația element-cu-element este executată rapid prin codul C precompilat. În NumPy

```
c = a * b
```

face ceea ce fac exemplele anterioare, la viteze apropiate de C, dar cu simplitatea codului pe care ne așteptăm de la ceva bazat pe Python. Într-adevăr, idiomul NumPy este chiar mai simplu! Acest ultim exemplu ilustrează două dintre trăsăturile lui NumPy care stau la baza unei mari părți a puterii sale: vectorizare și difuzare.

Vectorizarea descrie absența oricăror bucle, indexare etc. în cod - aceste lucruri au loc, bineînțeles, doar "în spatele scenei" în codul C optimizat, precompilat. Codul vectorizat are multe avantaje, printre care:

- codul vectorizat este mai concis și mai ușor de citit;
- mai puține linii de cod înseamnă în general mai puține erori;
- codul se aseamănă mai mult cu notația matematică standard (făcând mai ușor, de obicei, codarea corectă a construcțiilor matematice);
- vectorizarea rezultă mai mult în codul "Pythonic". Fără vectorizare, codul nostru ar fi alunecat cu bucle for ineficiente și greu de citit.

Difuzarea este termenul utilizat pentru a descrie comportamentul implicit al operațiilor element cu element; în general, în NumPy toate operațiunile, nu doar operațiile aritmetice, ci logice, pe biti, funcționale, etc., se comportă în acest mod implicit element-cu-element, adică, difuzate. Mai mult decât atât, în exemplul de mai sus a și b ar putea fi matrici multidimensionale de aceeași formă, sau un scalar și o matrice sau chiar două matrici cu forme diferite, cu condiția ca matricea mai mică să fie "expandabilă", o modalitate prin care difuzarea rezultată este lipsită de ambiguitate.

## CAPITOLUL 5. TESTAREA APLICAȚIEI

O aplicație netestată va duce mereu către un comportament neașteptat. În funcție de tipul de aplicație consecințele pot varia de la neplăceri minore în cazul aplicațiilor simple la pierderi însemnate de bani sau chiar vieți omenești. De aceea testarea unei aplicații este imperativă!

### 5.1 Tipuri de erori ale aplicației

În rândurile următoare voi trece în revistă câteva dintre tipurile de erori ce pot apărea pe parcursul dezvoltării aplicației cât și pe perioada în care aceasta se află în mentenanță

#### 5.1.1 Modificări în codul sursă al aplicației

De multe ori se întâmplă ca aplicația să capete comportamente noi, datorită unor factori externi cum ar fi dorința clientului sau schimbări de design dorite de dezvoltatori. În cazul aplicațiilor mai mari, este foarte greu de urmărit impactul general al acestor modificări asupra aplicației. Dacă pentru dezvoltarea aplicației lucrează o echipă de programatori și nu unul singur este aproape imposibil de urmărit. Prin schimbarea codului sursă se introduc de obicei buguri de logică sau care afectează aplicația în alte părți ale ei. De exemplu, schimbarea unei metode de calcul care este folosită în alt modul al aplicației sau în altă aplicație.

#### 5.1.2 Erori cauzate de programator

Programatorii pot, și vor introduce buguri în aplicație. Acestea pot fi erori logice sau erori de sintaxă. În cadrul mediilor de dezvoltare recente, erorile de sintaxă sunt evidențiate imediat de către IDE. Erorile logice, sau semantice sunt însă mai greu de depistat. În cazul unor limbaje în care identarea nu contează doar pentru confortul programatorului acestea pot compromite funcționalitatea programului. Un exemplu este Python, unde nu există acoladele clasice, și toate blocurile de instrucțiuni sunt interpretate prin indentare. Alte erori logice sunt introducerea de referințe ce pot fi nule, omiterea eliberării memoriei dacă limbajul de programare utilizat nu conține un mecanism de GC.

#### 5.1.3 Erori cauzate de utilizator

Utilizatorii vor folosi aplicația într-un mod neașteptat. Este un lucru normal, pentru că dezvoltatorul știe exact ce are de implementat și nu este obișnuit să folosească aplicația într-un alt mod față de cel propus. Aceste erori nu apar din vina utilizatorilor, dezvoltatorii ar trebui să fie conștienți de aceste riscuri și ar trebui să acopere cazurile de folosire eronată a aplicației.

#### 5.1.4 Erori care țin de factori externi

O mare parte din aplicațiile existente comunică cu alte sisteme prin rețele LAN sau prin internet. Și acestea trebuie tratate corect în codul sursă. Un alt factor extern

care poate influența funcționarea corectă a aplicației este modificarea resurselor folosite de aplicația noastră, a SDK-ului, sau a API-ului folosit. Un exemplu clasic în acest sens este librăria de testare unitară JUnit. În versiunile mai vechi, JUnit nu avea prea multe mecanisme expuse către dezvoltatori. Deși era o librărie robustă, în lipsa unui API care să acopere cât mai multe nevoi ale programatorilor s-a folosit reflexia pentru a obține efectele dorite. Consecința a constat în faptul că echipa care dezvoltă librăria nu mai putea face livrări fără să afecteze o mare parte din programatori. Prin reflexie, aceștia aveau acces la metode și date membre la care nu ar fi avut acces în mod obișnuit. Modificând structura internă a librăriei, aplicațiile celor ce o foloseau nu mai puteau să funcționeze în parametri normali. Echipa care a dezvoltat JUnit a învățat din acest lucru iar la următoarea versiune a librăriei, programatorii au avut parte de un API mai consistent.

Un alt exemplu în acest sens, deși la o scară mult mai mică este implementarea internă a interfeței Map și a claselor ce derivă din aceasta din JDK. Între Java 7 și Java 8, Oracle a modificat modul de funcționare al interfeței. Am căutat documentație care să ateste acest lucru, dar nu am găsit. Modificarea constă în faptul că, la iterare prin mapă ordinea se schimbă de la JDK 7 la JDK 8. Acest lucru a condus la apariția unui bug în aplicație, un bug foarte greu de găsit. Pentru că același cod sursă avea comportamente diferite în cele două JDK.

## 5.2 Moduri de testare ale aplicației

Tipurile de testare prezentate mai jos nu se exclud unele pe altele, ci sunt complementare. Unele tipuri privesc aplicația din punctul de vedere al utilizatorului, altele sunt orientate către funcționalitate și client.

### 5.2.1 Testarea unitară

Testarea unitară ar trebui făcută de fiecare programator în parte pentru codul pe care l-a scris. Aceasta presupune scrierea de metode care testează codul scris anterior. Este atât de importantă, încât o metodologie de programare îi poartă numele: Test Driven Development, sau mai pe scurt TDD. Aceasta presupune ca dezvoltatorul să scrie testele unitare ce vor testa funcționalitatea ce urmează a fi implementată. Testul nu va trece în starea inițială. Acest lucru asigură faptul că testul funcționează și poate detecta o eroare. Odată apărută eroarea, funcționalitatea poate fi implementată. TDD repeta constant acești pași în care se adaugă teste care nu trec, implementare până acestea trec cu succes și la final refactorizarea codului. Deși pare “pe dos” această metodologie îmbunătățește modelul mental al codului pe care îl are dezvoltatorul și crește productivitatea.

### 5.2.2 Testarea integrării

Acest tip de testare este efectuată pentru sistemele distribuite și pentru sistemele care comunică cu alte module. Aceasta constă în testarea tuturor modulelor ce alcătuiesc un sistem. Spre exemplu în cazul unei aplicații web care conține atât back-end cât și front-

end. În cadrul acestui tip de testare se verifică funcționalitatea aplicației într-un scenariu real. Acestea trebuie să acopere toate use-case-urile aplicației.

### 5.3 Testarea aplicației

Pentru a testa aplicația am folosit librăriile ajutoare JUnit si Mockito. JUnit este o librărie pentru testarea unitară, iar Mockito este o librărie care se folosește pentru a testa componente la care accesul nu se poate face în mod normal din cod. Spre exemplu comunicarea cu un modul care nu există sau verificarea înregistrărilor într-o bază de date. În loc să verificăm manual dacă înregistrările s-au făcut putem verifica de câte ori, și cu ce parametri s-a apelat metoda care efectuează operațiunea pe baza de date.

Pentru testarea aplicației dezvoltate am scris un număr de teste unitare, cât și teste de conectivitate sau teste end to end. În urma rulării testelor am descoperit erori de tipurile menționate mai sus.

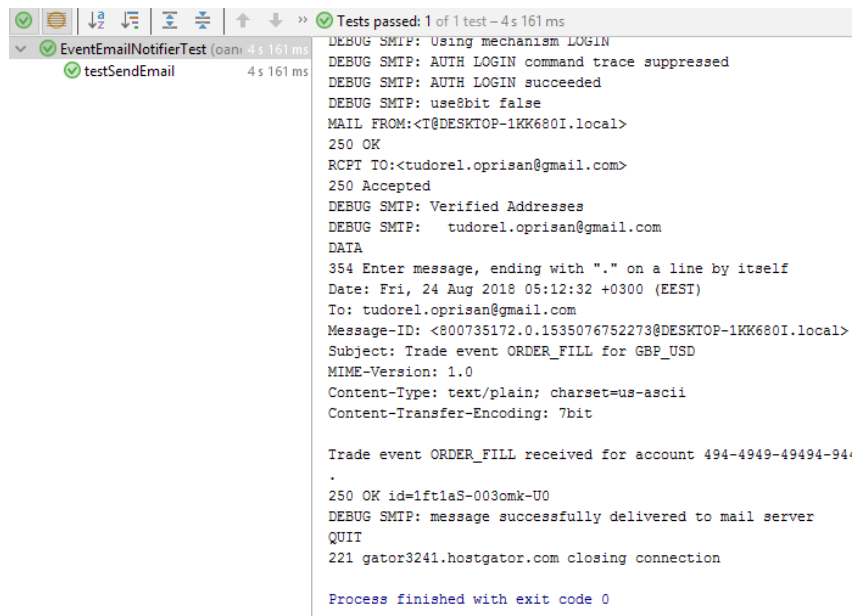
```
"C:\Program Files\Java\jdk1.8.0_151\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2018.1.4\lib\idea_rt.jar=5'
2018-08-24 04:58:59,562 INFO [main] - Entering method getLatestAccountsInfo in OADPS
0 [main] INFO brokerAPI.account.BrokerAccountDataProviderService - Entering method getLatestAccountsInfo in OADPS
2018-08-24 04:58:59,995 INFO [main] - Executing request : GET https://api-fxpractice.forex.com/v3/accounts HTTP/1.1
433 [main] INFO brokerAPI.account.BrokerAccountDataProviderService - Executing request : GET https://api-fxpractice.forex.com/v3/
2018-08-24 04:59:00,711 INFO [main] - {"accounts":[{"id":"101-004-9126938-001","tags":[]}]}
1149 [main] INFO brokerAPI.account.BrokerAccountDataProviderService - {"accounts":[{"id":"101-004-9126938-001","tags":[]}]}
2018-08-24 04:59:00,716 INFO [main] - Got ID 101-004-9126938-001
1154 [main] INFO brokerAPI.account.BrokerAccountDataProviderService - Got ID 101-004-9126938-001
2018-08-24 04:59:00,717 INFO [main] - Executing request : GET https://api-fxpractice.forex.com/v3/accounts/101-004-9126938-001 H:
1155 [main] INFO brokerAPI.account.BrokerAccountDataProviderService - Executing request : GET https://api-fxpractice.forex.com/v/
{"account":{"guaranteedStopLossOrderMode":"DISABLED","id":"101-004-9126938-001","createTime":"2018-08-21T12:29:28.248646359Z","c
{"createdByUserID":"9126938","NAV":"99999.9985","marginCloseoutUnrealizedPL":"0.0000","marginCallMarginUsed":"0.0000","openPosition(
2018-08-24 04:59:00,865 INFO [main] - ID: 101-004-9126938-001 Currency: EUR
1303 [main] INFO brokerAPI.account.BrokerAccountDataProviderService - ID: 101-004-9126938-001 Currency: EUR
2018-08-24 04:59:00,866 INFO [main] - Found 1 accounts to trade for user toprisan
1304 [main] INFO test.AccountTest - Found 1 accounts to trade for user toprisan
2018-08-24 04:59:00,866 INFO [main] - ++++++ Dumping Account Info ++++++
1304 [main] INFO test.AccountTest - ++++++ Dumping Account Info ++++++
2018-08-24 04:59:00,866 INFO [main] - Currency=EUR,NAV=100000.00,Total Balance=100000.00, UnrealisedPnl= 0.00, RealisedPnl= 0.00,
1304 [main] INFO test.AccountTest - Currency=EUR,NAV=100000.00,Total Balance=100000.00, UnrealisedPnl= 0.00, RealisedPnl= 0.00,
2018-08-24 04:59:00,867 INFO [main] - ++++++ Finished Dumping Account Info ++++++
1305 [main] INFO test.AccountTest - ++++++ Finished Dumping Account Info ++++++
Base currency GBP.....
GBP_EUR
2018-08-24 04:59:00,878 INFO [main] - Executing request : GET https://api-fxpractice.forex.com/v3/accounts/101-004-9126938-001/p:
1316 [main] INFO brokerAPI.market.BrokerCurrentPriceInfoProvider - Executing request : GET https://api-fxpractice.forex.com/v3/a
{"errorMessage":"Invalid Instrument GBP_EUR"}
2018-08-24 04:59:01,515 INFO [main] - Executing request : GET https://api-fxpractice.forex.com/v3/accounts/101-004-9126938-001/p:
1953 [main] INFO brokerAPI.market.BrokerCurrentPriceInfoProvider - Executing request : GET https://api-fxpractice.forex.com/v3/a
2018-08-24 04:59:02,177 INFO [main] - ASK Object: {"price":"0.90172","liquidity":10000000}
2615 [main] INFO brokerAPI.market.BrokerCurrentPriceInfoProvider - ASK Object: {"price":"0.90172","liquidity":10000000}
After GBP_USD.....
Base currency EUR.....
After EUR_CHF.....
2018-08-24 04:59:02,178 INFO [main] - Margining requirement for trading pair 5000 units of GBP_USD is 110.91 EUR
2616 [main] INFO test.AccountTest - Margining requirement for trading pair 5000 units of GBP_USD is 110.91 EUR
2018-08-24 04:59:02,178 INFO [main] - Margining requirement for trading pair 5000 units of EUR_CHF is 100.00 EUR
2616 [main] INFO test.AccountTest - Margining requirement for trading pair 5000 units of EUR_CHF is 100.00 EUR
```

Figura 31 Test cont utilizator si permisiuni

În figură se poate observa și un mesaj de eroare venit de la server:

```
{"errorMessage":"Invalid Instrument GBP_EUR"}
```

Acesta apare pentru că instrumentele de trade nu sunt reversibile. Această eroare poate fi privită ca o eroare de logică.



```

Tests passed: 1 of 1 test - 4s 161 ms
EventEmailNotifierTest (oani 4s 161 ms)
  testSendEmail 4s 161 ms
    DEBUG SMTP: Using mechanism LOGIN
    DEBUG SMTP: AUTH LOGIN command trace suppressed
    DEBUG SMTP: AUTH LOGIN succeeded
    DEBUG SMTP: use8bit false
    MAIL FROM:<T@DESKTOP-1KK680I.local>
    250 OK
    RCPT TO:<tudorel.oprisan@gmail.com>
    250 Accepted
    DEBUG SMTP: Verified Addresses
    DEBUG SMTP: tudorel.oprisan@gmail.com
    DATA
    354 Enter message, ending with "." on a line by itself
    Date: Fri, 24 Aug 2018 05:12:32 +0300 (EEST)
    To: tudorel.oprisan@gmail.com
    Message-ID: <800735172.0.1535076752273@DESKTOP-1KK680I.local>
    Subject: Trade event ORDER_FILL for GBP_USD
    MIME-Version: 1.0
    Content-Type: text/plain; charset=us-ascii
    Content-Transfer-Encoding: 7bit

    Trade event ORDER_FILL received for account 494-4949-49494-944.
    .
    250 OK id=1ft1a5-003omk-U0
    DEBUG SMTP: message successfully delivered to mail server
    QUIT
    221 gator3241.hostgator.com closing connection

    Process finished with exit code 0
  
```

Figura 32 test trimitere email

Deși testul trece cu succes, acesta o face doar pentru că serverul smtp a returnat 200. În realitate mesajele email nu ajungeau, așa că am investigat problema. Din cauza unei măsuri de securitate a serverului SMTP pe care l-am folosit, acesta nu trimite e-mailuri dacă nu găsește o valoare în câmpul from. După efectuarea acestei schimbări mesajele se transmit

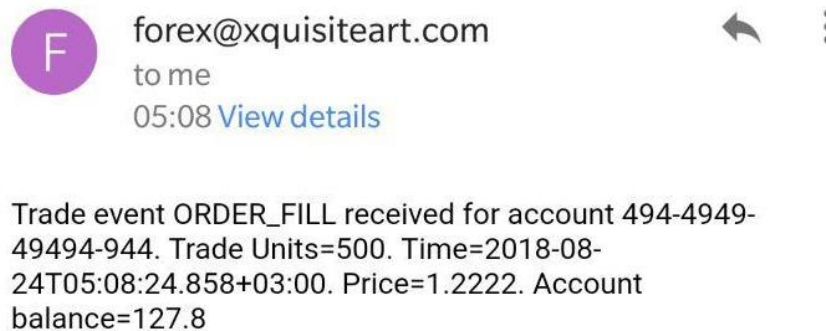


Figura 33 Email primit

Un alt test pe care l-am efectuat este cel de serviciu de stream. Aici am descoperit că API-ul oferit de broker s-a schimbat o parte din funcționalitățile aplicației pierzându-se. Pentru a înlocui facilitățile pierdute voi folosi alt tip de implementare.

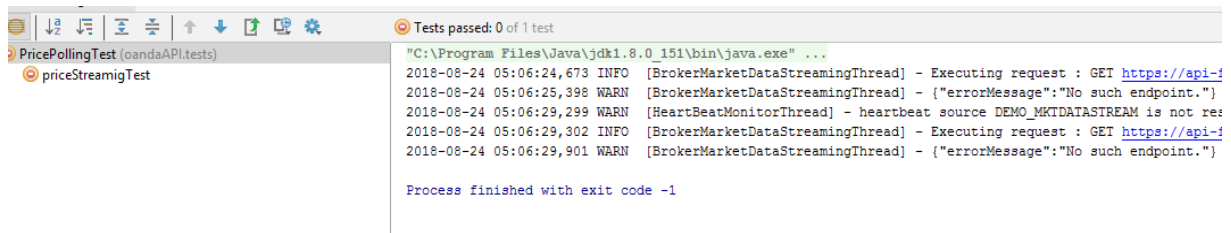


Figura 34 No such Endpoint

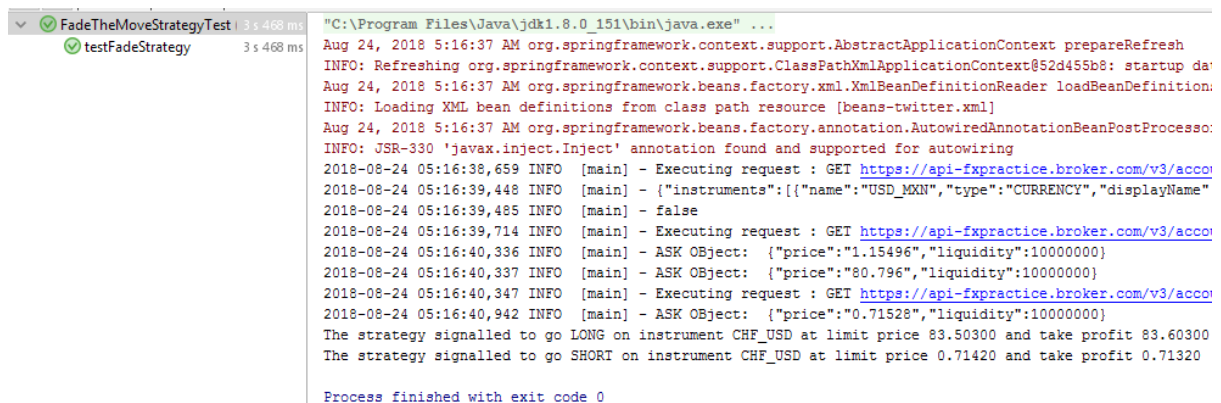


Figura 35 Test strategie

Deși algoritmi sunt scriși în Python, o parte din testare se efectuează și în Java, pentru a scade riscurile aferente tradingului. Dacă valorile nu algoritmului nu corespund cu măsurile de siguranță implementate în Java ordinul nu se va plasa.

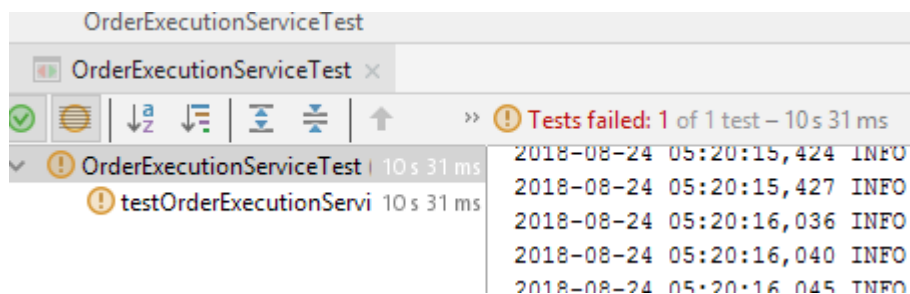
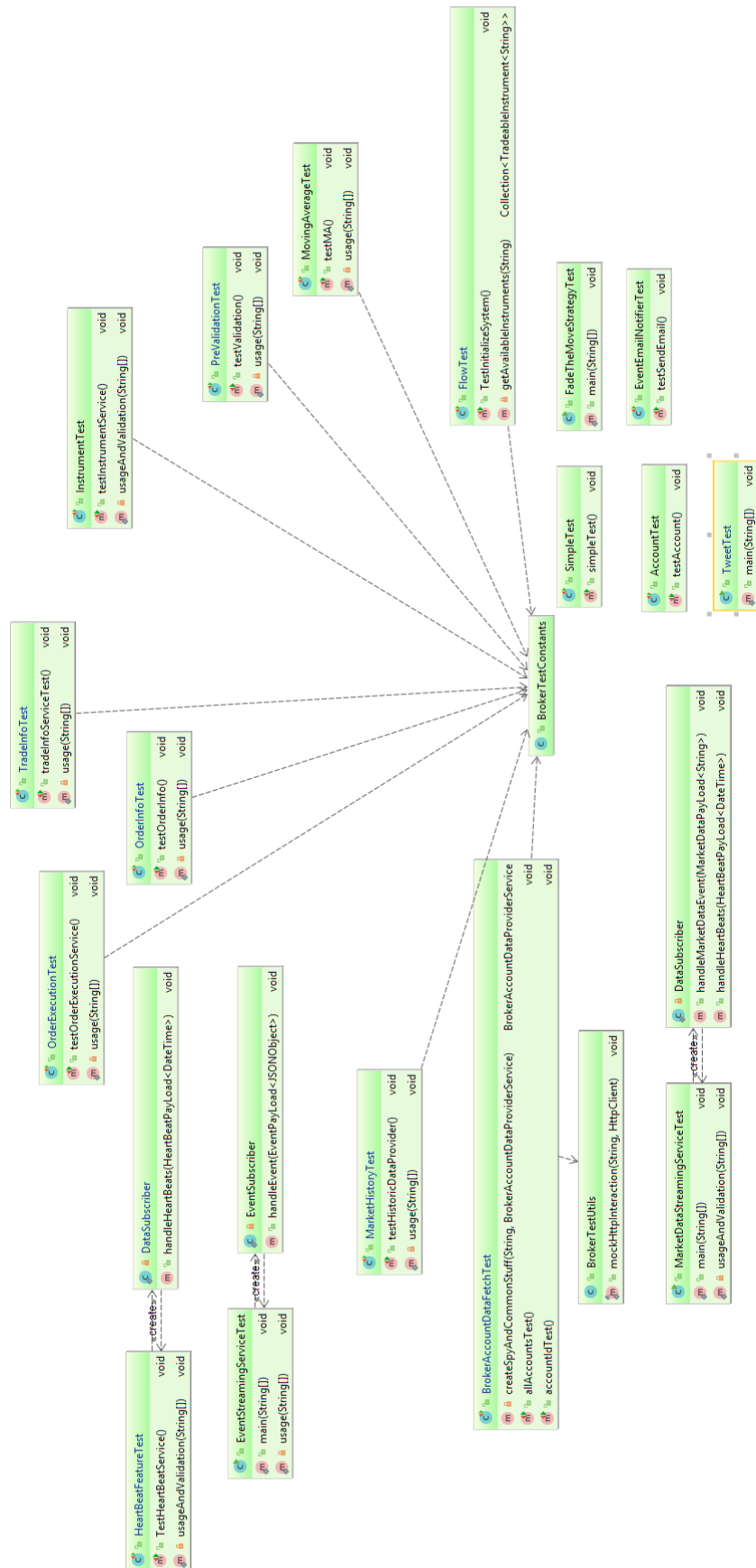


Figura 36 Test care nu a trecut

Parsing order... {"triggerCondition":"DEFAULT","stopLossOnFill":{"price":"1.20000","timeInForce":"GTC"},"takeProfitOnFill":{"price":"1.4  
Cannot place trade CHF\_USD because max limit exceeded. max allowed=0 and future net positions=-2 for currency USD if trade executed

Figura 37 Ordinele nu se vor plasa datorită discrepantei de valori față de piață





Powered by YFiles

Figura 38 Diagrama UML Pachetul de teste

## CONCLUZII

Vor cuprinde într-o formă cât mai concisă principale rezultatele obținute în tema tratată, subliniind contribuția adusă prin propriile cercetări.

Din testele și încercările efectuate de mine pot să afirm ca posibilitatea de a identifica un tipar cu precizie ridicată în piața forex nu este foarte mare, însă avantajul acestor metode permit evoluția constantă a algoritmilor și dezvoltarea unor algoritmi noi. Calculele sunt efectuate de calculator, iar în perioada în care ne aflăm puterea de calcul nu este foarte scumpă în comparație cu anii 2000. Algoritmii pot rula permanent și pot identifica oportunități care au un ROI pozitiv. Deși încercările inițiale nu au avut efectul dorit pe deplin, pot să spun că am implementat un algoritm destul de simplist care are o rată profit de profit de 10% pe an, efectuat pe perechile EURUSD și USDCAD. Deși 10% este un profit destul de mic, trebuie luat în calcul atât volumul de tranzacționare cât și perioada pe care se tranzacționează. Pe lângă acest lucru se mai poate lua în calcul și posibilitatea de a tranzacționa cu leverage, ceea ce înseamnă că profitul poate crește destul de mult. Acest lucru nu vine însă fără dezavantaje pentru că, în cazul în care cursul nu evoluează în direcția prezisă de algoritm și pierderile vor fi direct proporționale cu potențialul de câștig.

Ca o concluzie generală, acești algoritmi trebuie testați foarte bine, și pe o perioadă mai mare de timp, pentru a asigura o șansă de reușită cât mai mare. Piața de tranzacționare există de foarte mult timp, iar studiile efectuate asupra ei sunt pe măsură.

## BIBLIOGRAFIE

1. A. Chaboud et al, "Rise of the Machines: Algorithmic Trading in the Foreign Exchange Market,"Int'l Finance Discussion Papers, Board of Governors of the Federal Reserve System, Oct. 2009; [www.federalreserve.gov/pubs/ifdp/2009/980/ifdp980.pdf](http://www.federalreserve.gov/pubs/ifdp/2009/980/ifdp980.pdf)
2. Morton Glantz, Robert Kissell. *Multi-Asset Risk Modeling: Techniques for a Global Economy in an Electronic and Algorithmic Trading Era*. Academic Press, Dec 3, 2013, p. 258.
3. <https://www.onestepremoved.com/autocorrelation/>
4. Kyong Shik Eoma\*, Sang Buhm Hahn, Sangyong Jooc - Partial price adjustment and autocorrelation in foreign exchange markets; <https://pdfs.semanticscholar.org/f5b3/c24c7773d252e844d36884950a455c8d8316.pdf>
5. The Pennsylvania State University - Partial Autocorrelation Function (PACF)
6. <http://www.investopedia.com>
7. <https://www.wikipedia.org>
8. Derivative analysis with Python - Wileys

## ANEXA 1

### Diagrame UML(Continuare)

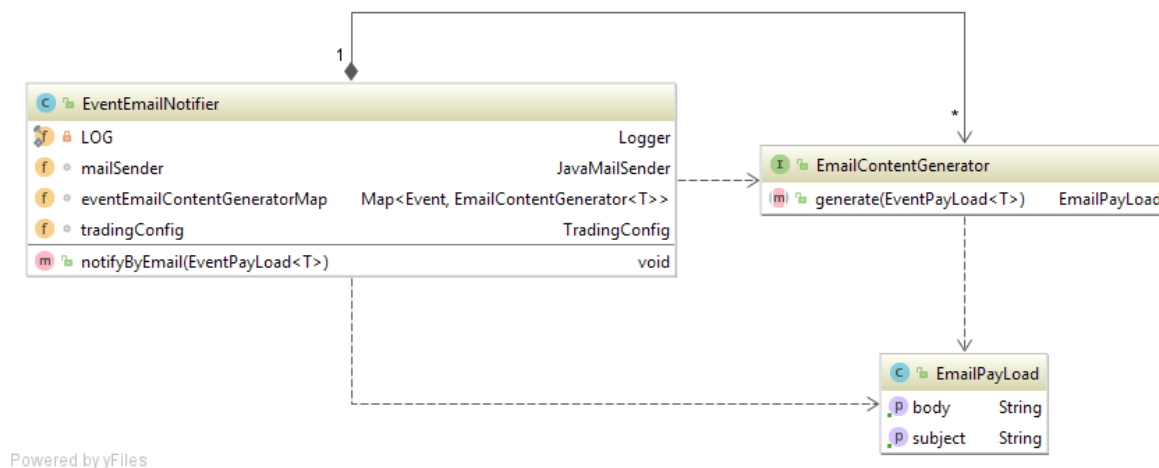


Figura 39 Diagrama UML pachetul email

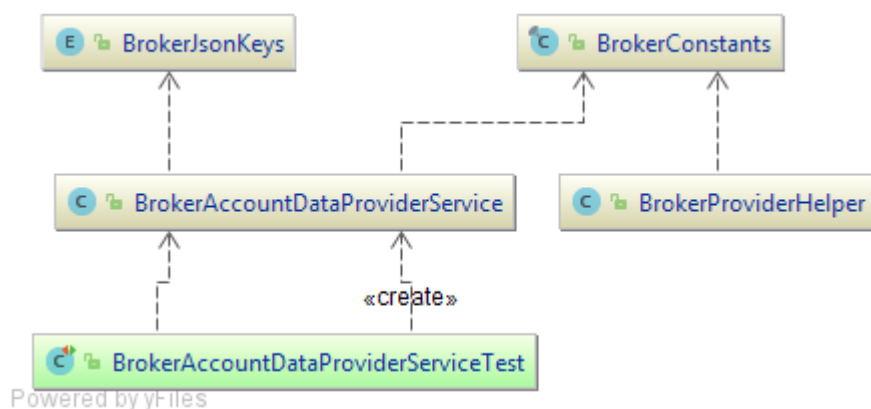


Figura 40 Diagrama UML pachetul broker/account



53

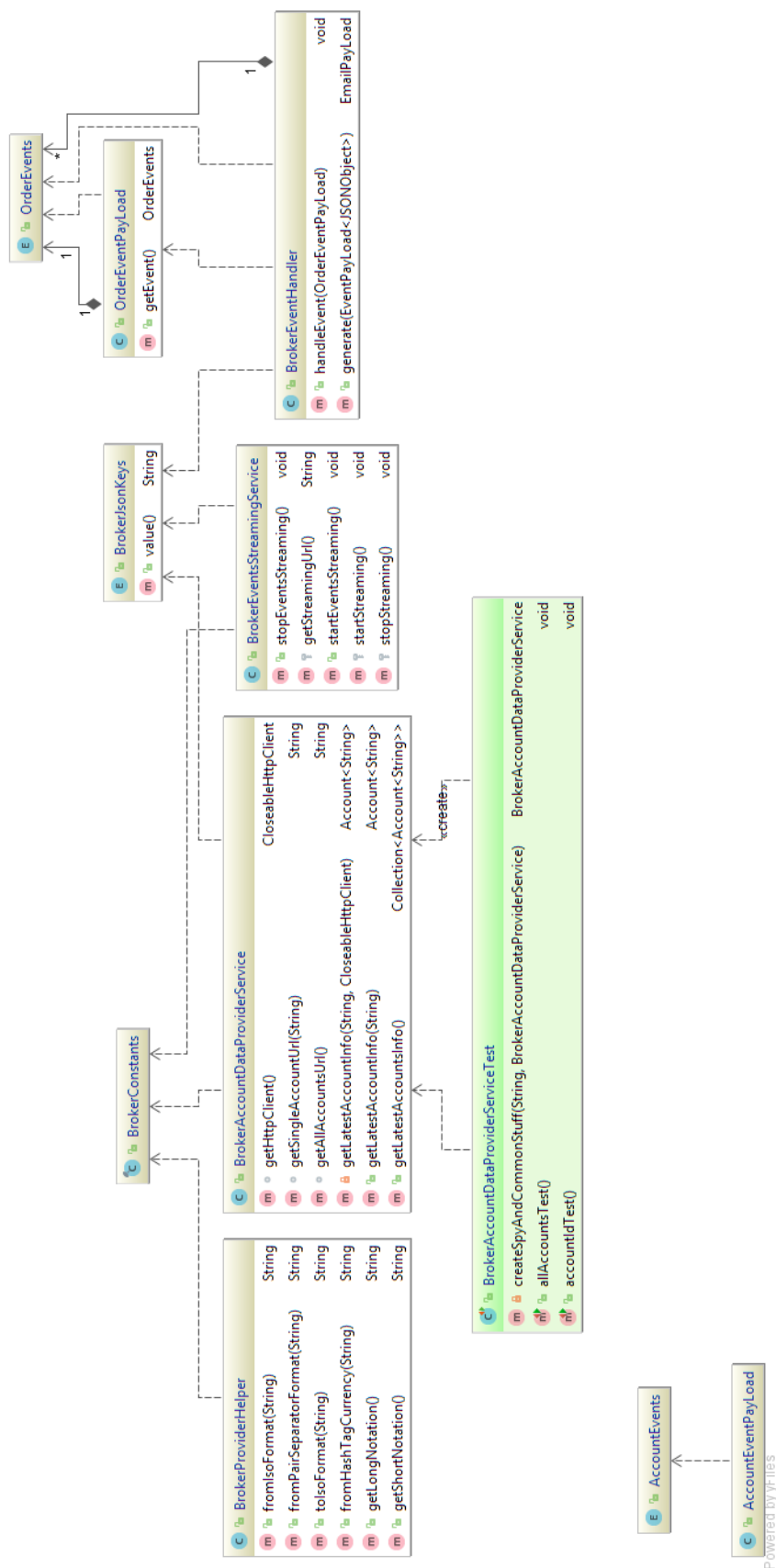


Figura 42 Diagrama UML pachetul broker/events în detaliu

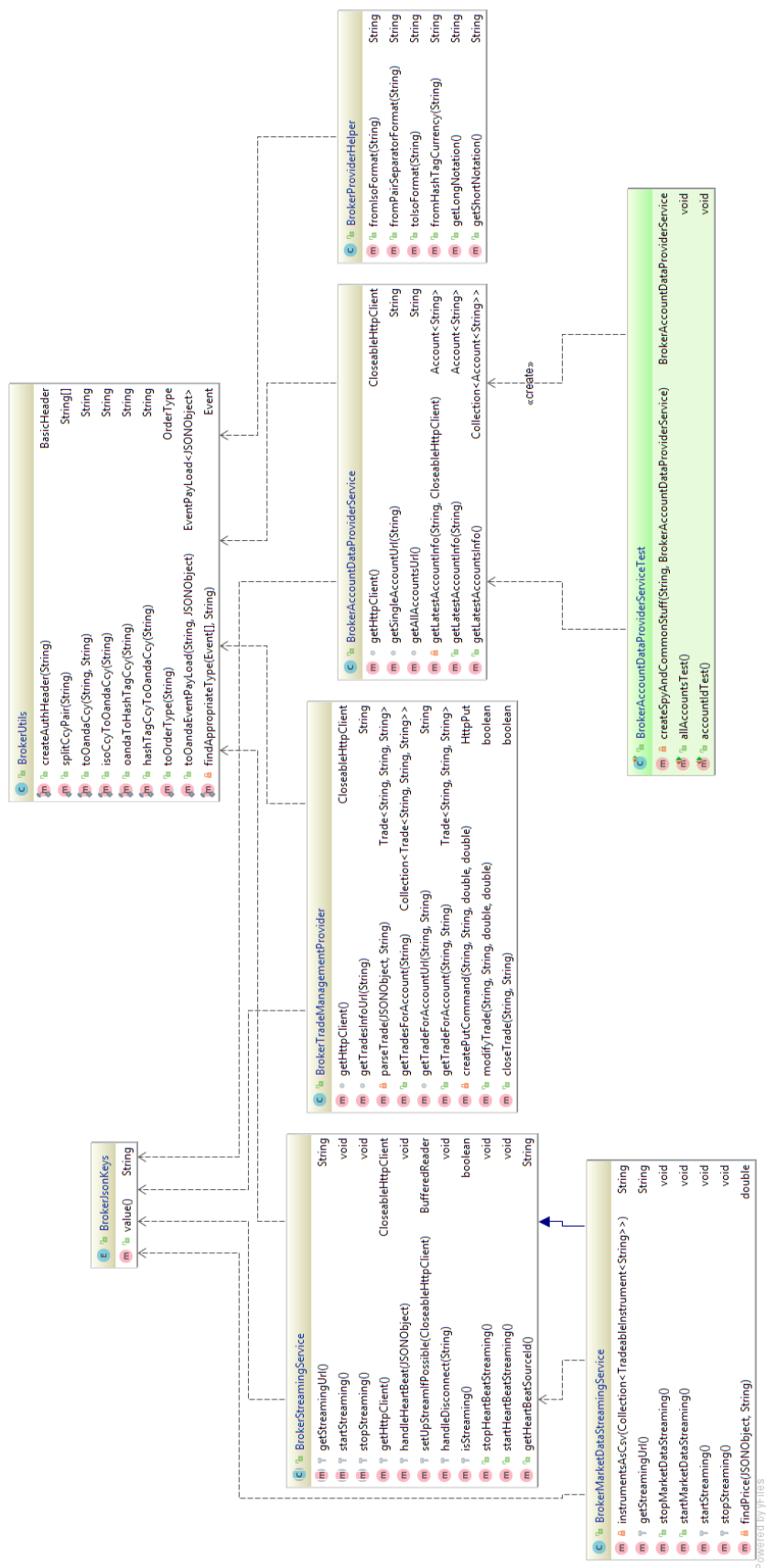


Figura 43 Diagrama UML pachetul broker în detaliu

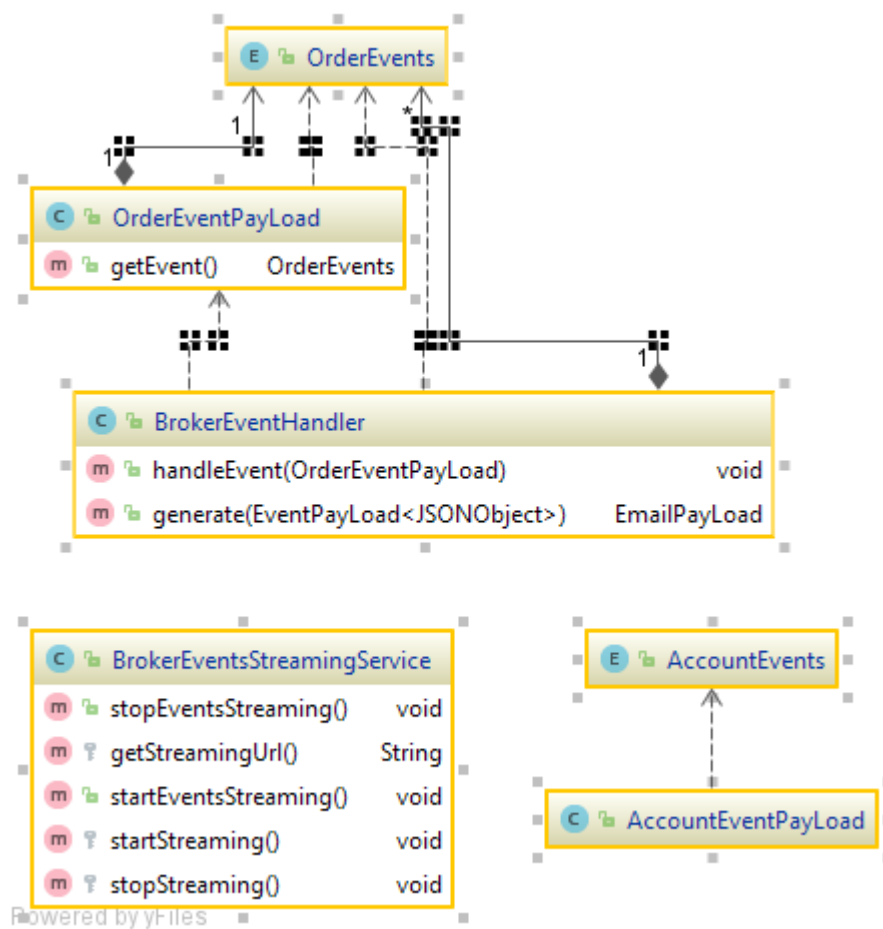


Figura 44 Diagrama UML pachetul events în detaliu



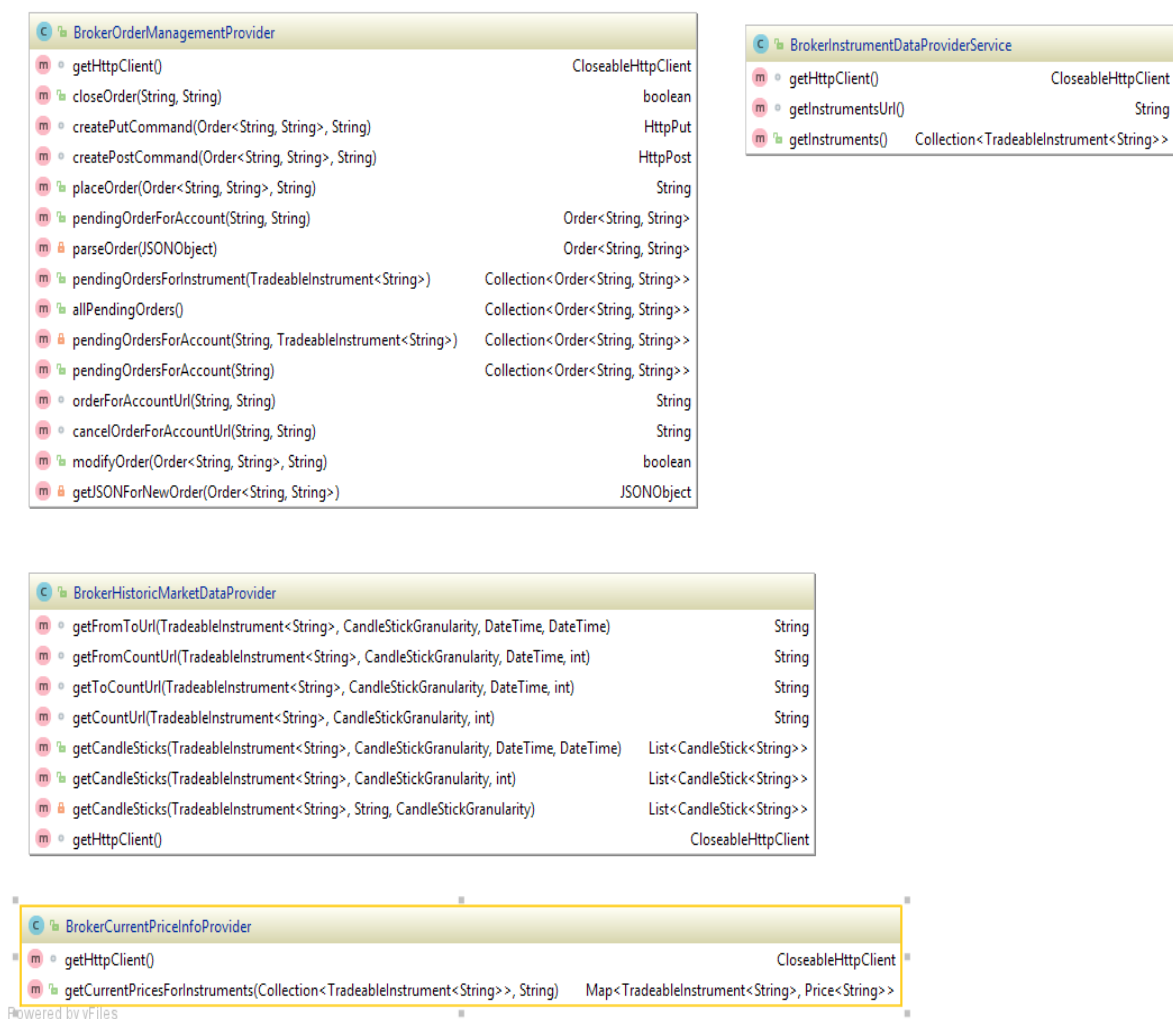


Figura 45 Diagrama UML pachetele instruments/marketData/order în detaliu