UNIVERSITATEA TEHNICĂ„GHEORGHE ASACHI” DIN IAȘI
FACULTATEA DE AUTOMATICĂȘI CALCULATOARE
Domeniul: Calculatoare și Tehnologia Informației
Programul de studii: Tehnologia Informației

PROIECT DE DIPLOMĂ

Coordonator științific:
prof.dr.ing. Florin LEON

Absolvent:
Andrei POPA

Iași, 2025

UNIVERSITATEA TEHNICĂ„GHEORGHE ASACHI” DIN IAȘI
FACULTATEA DE AUTOMATICĂȘI CALCULATOARE
Domeniul: Calculatoare și Tehnologia Informației
Programul de studii: Tehnologia Informației

TradeMind - Platformă pentru optimizarea deciziilor în tranzacționare

PROIECT DE DIPLOMĂ

Coordonator științific:
Prof.dr.ing. Florin LEON

Absolvent:
Andrei POPA

Iași, 2025

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII PROIECTULUI DE DIPLOMĂ

Subsemnatul _____,
legitimat cu _seria _____nr. _____, CNP_____
_____ autorul lucrării _____
_____

elaborată în vederea susţinerii examenului de finalizare a studiilor de licență, programul de studii _____organizat de către Facultatea de Automatică şi Calculatoare din cadrul Universităţii Tehnice „Gheorghe Asachi" din Iaşi, sesiunea _____ a anului universitar _____, luând în considerare conţinutul Art. 34 din Codul de etică universitară al Universităţii Tehnice
„Gheorghe Asachi" din Iaşi (Manualul Procedurilor, UTI.POM.02 - Funcţionarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române (legea 8/1996) şi a convenţiilor internaţionale privind drepturile de autor.

Data Semnătura

Cuprins

TradeMind - Platformă pentru optimizarea deciziilor în tranzacționare

Andrei POPA

Rezumat

Lucrarea de față abordează tema testării și analizei strategiilor de tranzacționare prin dezvoltarea unei platforme web, numită TradeMind. Motivația alegerii subiectului pornește de la nevoia reală a traderilor individuali de a dispune de un instrument accesibil, care să combine într-un singur mediu funcționalități de backtesting, analiză statistică și suport decizional. Majoritatea soluțiilor existente sunt fie fragmentate, fie greu de utilizat pentru persoanele fără experiență tehnică, ceea ce justifică propunerea unei alternative unitare și intuitive.

Obiectivul principal al proiectului a fost construirea unei platforme care să permită rularea strategiilor de tranzacționare pe date istorice, generarea automată de tranzacții simulate și calculul indicatorilor de performanță relevanți. Metodologia a inclus o arhitectură modulară, bazată pe separarea clară a componentelor frontend, backend și bază de date, precum și testări iterative pentru validarea stabilității și a corectitudinii rezultatelor.

Soluția propusă integrează un motor de backtesting, algoritmi de analiză statistică, rapoarte vizuale și o interfață prietenoasă. În plus, aplicația oferă posibilitatea de comparare a strategiilor, de personalizare a parametrilor și de obținere a rapoartelor grafice care evidențiază evoluția capitalului și riscul asumat.

Rezultatele experimentale arată că aplicația este fiabilă și scalabilă, capabilă să evidențieze diferențele de performanță între strategii și să ofere utilizatorului o imagine clară asupra eficienței acestora. Concluzia generală este că TradeMind reprezintă o bază solidă pentru extinderea ulterioară a cercetării și poate constitui un sprijin real pentru procesul decizional al traderilor.

Keywords: trading, backtesting, trader, raport risc/recompensă, drawdown, financiar.

Capitolul 1. Introducere

1.1. Motivația alegerii temei

În cadrul digitalizării accelerate, activitatea de tranzacționare pe piețele financiare a devenit un proces complex, în care deciziile bine fundamentate și o analiză de calitate facută în prealabil pot face diferența dintre un trader profitabil și unul neprofitabil. Traderii se confruntă zilnic cu o abundență de date, indicatori, semnale contradictorii și riscuri imprevizibile care sunt complet contrare raționamentului uman.

Tema aleasă pentru lucrarea de față abordează această nevoie concretă prin propunerea unei soluții software, numită TradeMind, care are ca scop optimizarea procesului de analiză a performanței financiare și de testare a strategiilor de tranzacționare. Proiectul se adresează în special traderilor individuali, amatori sau semi-profesioniști, care nu dispun de infrastructuri sofisticate, dar doresc să ia decizii informate bazate pe date proprii și să își dezvolte un fundament solid în vederea obținerii de rezultate pozitive.

Alegerea acestei teme a fost influențată de două direcții principale: interesul personal pentru domeniul tranzacționării algoritmice și observarea unei nevoi reale în rândul comunității de traderi. În pofida numeroaselor soluții comerciale existente pe piață, majoritatea sunt fie prea costisitoare, fie dificil de personalizat. În plus, multe dintre ele separă funcționalități esențiale precum backtesting-ul, analiza de performanță și gestionarea documentelor, făcând procesul greoi și fragmentat.

Proiectul TradeMind își propune să răspundă acestor limitări printr-o abordare unificată, oferind traderului o platformă accesibilă, care să-i permită să își înțeleagă mai bine performanțele, să își testeze strategiile și să-și gestioneze mai eficient resursele și riscurile.

1.2. Relevanța temei și contextul general

Tranzacționarea pe piețele financiare presupune un proces intens și rapid, în care deciziile se bazează pe date istorice, statistici și reacții la dinamica pieței. Spre deosebire de investiții, care urmăresc obținerea unor randamente pe termen lung prin deținerea unor active și monitorizarea tendințelor generale, tradingul are un orizont scurt, axat pe exploatarea variațiilor de preț și pe evaluarea continuă a performanțelor. În acest context, suportul tehnologic devine esențial pentru a reduce subiectivitatea deciziilor și a crește șansele de profitabilitate. Platforma TradeMind se aliniază acestei nevoi și propune o soluție dezvoltată pentru a sprijini traderii individuali în gestionarea și analiza eficientă a activității lor.

Astfel, tema este ancorată în tendințele actuale ale piețelor financiare, unde analiza bazată pe date și simulări este indispensabilă, și are un potențial real de aplicabilitate practică. Totodată, platforma contribuie la dezvoltarea abilităților traderilor prin oferirea unui mediu controlat, unde aceștia pot învăța, experimenta și valida propriile strategii.

1.3. Obiectivele generale și specifice ale lucrării

Obiectivul principal al lucrării este realizarea unei soluții care să sprijine traderii în procesul decizional, prin:

- analiza performanței conturilor reale/demo de tranzacționare conectate la cea mai cunoscută platformă de tranzacționare - MetaTrader 5;
- testarea și optimizarea strategiilor de tranzacționare pe baza datelor istorice;
- organizarea și gestionarea tranzacțiilor personale, împreună cu un set de statistici relevante;

- centralizarea și accesul rapid la informații și rezultate vizuale.

Obiectivele specifice includ:

- dezvoltarea unui modul de backtesting care folosește date istorice și permite reglajul parametrilor strategiilor;
- construirea unui dashboard vizual și intuitiv care să sintetizeze performanța utilizatorului atât pentru conturile reale, cât și pentru cele simulate;
- implementarea unui sistem de management al trade-urilor, cu posibilitatea de adăugare, vizualizare și filtrare a acestora;
- explorarea integrării cu surse de date externe și API-uri financiare (precum Yahoo Finance sau Binance) pentru rularea backtesturilor și generarea statisticilor.

1.4. Metodologia de cercetare utilizată

Metodologia adoptată în cadrul lucrării implică o abordare practică, centrată pe dezvoltarea incrementală a unei aplicații web functionale. Stack-ul tehnologic utilizat este compus din:

- **Frontend:** React.js, împreună cu Axios pentru gestionarea apelurilor HTTP către backend, asigurând o interfață responsive, modulară și intuitivă;
- **Backend:** FastAPI (Python) - pentru dezvoltarea rapidă și securizată a serviciilor REST, incluzând gestionarea autentificării prin JWT și a fluxurilor de utilizator;
- **Baza de date:** PostgreSQL - pentru stocarea structurată și eficientă a informațiilor despre utilizatori, conturi și tranzacții;
- **Containerizare:** Docker - pentru asigurarea unei distribuții scalabile, reproductibile;
- **Surse de date externe:** API-uri financiare (Yahoo Finance, Binance) pentru obținerea datelor istorice necesare backtesting-ului;
- **Integrare MetaTrader 5 (MT5 API)** - pentru extragerea și gestionarea datelor conturilor reale de tranzacționare;
- **Instrumente de analiză**: algoritmi proprii de calcul ai performanței și logici de backtesting pentru evaluarea strategiilor;
- **Serviciu de email (SMTP)** - pentru confirmarea adreselor de email la înregistrare și resetarea parolei, oferind un mecanism suplimentar de securitate și validare a utilizatorilor.

Dezvoltarea a fost realizată iterativ, cu testări parțiale și validări continue în fiecare etapă.

1.5. Structura generală a lucrării

Lucrarea este structurată în cinci capitole principale, fiecare adresând o etapă distinctă a procesului de cercetare și dezvoltare:

- **Capitolul 2 - Fundamentarea teoretică și soluții similare**: prezintă concepte-cheie despre tranzacționare algoritmică, backtesting, analiză de performanță și analizează soluțiile existente, cu accent pe avantajele și limitările acestora.
- **Capitolul 3 - Soluția propusă**: descrie în detaliu arhitectura generală a sistemului, tehnologiile utilizate și modul în care sunt implementate componentele platformei TradeMind.
- **Capitolul 4 - Testarea soluției și rezultate experimentale**: oferă o analiză detaliată a performanței aplicației, stabilitate, acurateță a calculelor și gradul de utilitate în practică.
- **Capitolul 5 - Concluzii**: sintetizează rezultatele obținute, contribuția adusă în domeniu și direcții viitoare de dezvoltare.

Capitolul 2. Fundamentarea teoretică și documentarea bibliografică

2.1. Noțiuni teoretice și concepte cheie

Activitatea de tranzacționare pe piețele financiare a evoluat semnificativ în ultimele decenii, fiind tot mai influențată de tehnologii digitale și algoritmi automatizați. În această secțiune sunt prezentate conceptele esențiale care fundamentează dezvoltarea unei soluții software de asistență în tranzacționare.

2.1.1. Trading algoritmic

Tradingul algoritmic reprezintă utilizarea unor algoritmi matematici și a regulilor prestabilite pentru executarea automată a ordinelor pe piețele financiare. Acest tip de tranzacționare elimină intervenția umană în luarea deciziilor operaționale și are ca obiectiv creșterea vitezei de reacție și a preciziei [1]. Sistemele moderne de trading algoritmic pot integra semnale generate pe baza indicatorilor tehnici, a analizei fundamentale sau chiar a informațiilor provenite din procesarea limbajului natural (NLP).

2.1.2. Strategii de tranzacționare și evaluarea acestora

O strategie de tranzacționare este un set de reguli care determină când și în ce condiții un activ financiar este cumpărat sau vândut. Aceste reguli pot fi bazate pe indicatori tehnici (precum medii mobile, RSI, MACD), modele statistice sau chiar rețele neuronale [2]. Strategiile sunt în general clasificate în două mari categorii: bazate pe trend (trend-following) sau pe revenire la medie (mean-reversion).

Evaluarea unei strategii presupune determinarea eficienței acesteia în termeni de profitabilitate, risc și stabilitate. Astfel, se folosesc metrici precum profitul mediu, rata de câștig, drawdown-ul maxim sau profit factor [3]. În cadrul platformei TradeMind, aceste valori vor fi calculate automat și afișate în cadrul dashboard-ului de performanță, facilitând compararea obiectivă a mai multor abordări de trading.

2.1.3. Backtesting - testarea strategiilor pe date istorice

Backtesting-ul este procesul prin care o strategie de tranzacționare este aplicată retroactiv pe date istorice pentru a-i evalua performanța în trecut. Această tehnică permite identificarea punctelor tari și slabe ale unei strategii fără a risca capital real [4]. Un sistem de backtesting robust trebuie să țină cont de aspecte precum alunecarea (slippage), comisioanele de tranzacționare și condițiile de lichiditate.

În lucrarea de față, backtesting-ul constituie una dintre componentele centrale ale aplicației, oferind traderilor posibilitatea de a testa rapid și vizual strategiile proprii, precum și de a ajusta parametrii în timp real pentru optimizare.

2.1.4. Analiza performanței financiare

Analiza performanței constă în evaluarea rezultatelor istorice ale unei strategii pentru a determina dacă aceasta este viabilă din punct de vedere economic. Printre cei mai utilizați indicatori se numără:

- **Profitul total** și **randamentul mediu pe tranzacție**;
- **Rata de câștig** (win-rate);
- **Drawdown-ul maxim** - măsură a celei mai mari pierderi continue;
- **Profit factor** - care exprimă raportul dintre profit total și pierdere totală.

Acești indicatori permit compararea obiectivă a strategiilor și sunt esențiali pentru luarea deciziilor în mod rațional. În TradeMind, acești KPIs (Key Performance Indicators) sunt prezentați grafic și tabular pentru a permite o interpretare rapidă și intuitivă.

2.1.5. Reprezentarea grafică prin lumânări japoneze (candlestick)

Una dintre cele mai populare metode de reprezentare a evoluției prețurilor pe piețele financiare este graficul cu lumânări japoneze. Fiecare lumânare (candlestick) corespunde unei perioade de timp (de exemplu 1 minut, 5 minute, 1 oră sau 1 zi) și oferă informații esențiale despre dinamica prețului. O reprezentare grafică a acestora se găsește în [Figura 1].

O lumânare este formată din două părți principale: corpul (body) - zona care reprezintă intervalul dintre prețul de deschidere și prețul de închidere - și fitilul sau umbra (wick / shadow) - liniile subțiri deasupra și sub corp, care arată prețul maxim respectiv minim atins în perioada respectivă.

Culoarea corpului unei lumânări indică direcția pieței: verde indică o închidere mai mare decât deschiderea ceea ce semnifică o creștere a prețului în acel interval (lumânare bullish) iar roșu indică o închidere mai mică decât deschiderea ceea ce semnifică o scădere a prețului (lumânare bearish).

Astfel, o simplă privire asupra graficului cu lumânări permite identificarea rapidă a tendințelor, zonelor de volatilitate sau posibile tipare.

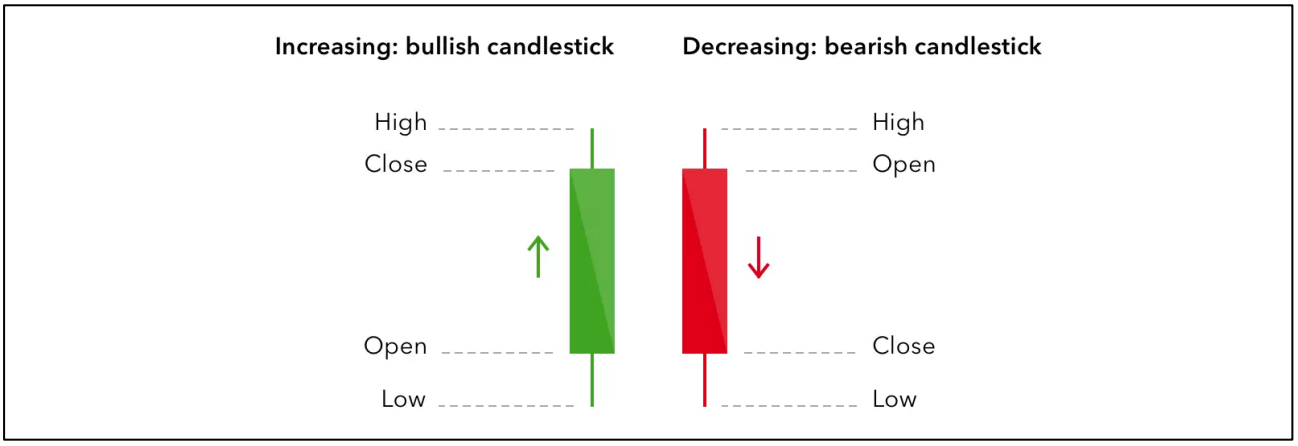


Figura 1. Structura unei lumânări japoneze

2.2. Strategii de tranzacționare analizate

2.2.1. Engulfing Pattern (Bullish/Bearish)

Este o formațiune de tip candlestick care apare atunci când o lumânare acoperă complet corpul lumânării precedente. Acest tipar semnalează, de obicei, o posibilă inversare a trendului. Există două variante ilustrate în [Figura 2]: Bearish Engulfing, ce apare după o creștere, sugerând un semnal de vânzare și Bullish Engulfing, ce apare după o scădere și marchează un potențial semnal de cumpărare,.

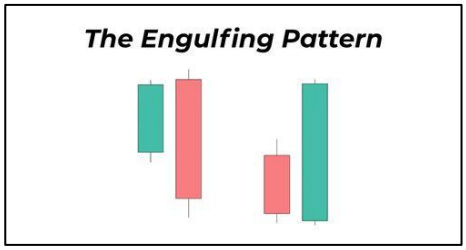


Figura 2. Exemplu Engulfing Pattern bearish & bullish

2.2.2. Hammer & Shooting Star

Lumânările de tip Hammer și Shooting Star ilustrate în [Figura 3] sunt ușor de identificat prin corpurile mici și umbrele foarte lungi. Hammer-ul apare la finalul unui trend descendent și indică posibilitatea unei mișcări ascendente, în timp ce Shooting Star se formează la capătul unui trend ascendent și poate anticipa o corecție descendentă.

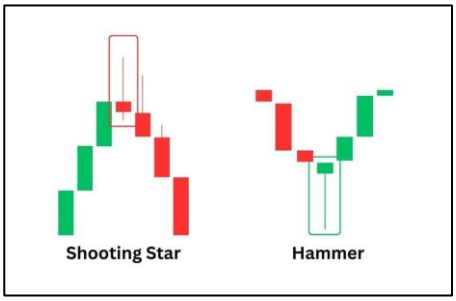


Figura 3. Exemplu Hammer & Shooting Star

2.2.3. Head and Shoulders

Modelul Head and Shoulders este unul dintre cele mai cunoscute tipare de inversare. Este alcătuit dintr-un vârf central mai înalt (cap) și două vârfuri mai mici (umeri).

Confirmarea schimbării de trend apare în momentul în care prețul străpunge linia „neckline” ilustrată cu albastru în [Figura 4].

Figura 4. Exemplu Head and Shoulders

2.2.4. Double Top & Double Bottom

Double Top și Double Bottom desenate în [Figura 5] sunt formațiuni grafice de tip „M” și „W”. Primul semnalează o posibilă scădere a prețului (pattern bearish), iar al doilea o posibilă creștere (pattern bullish). Ele arată, de regulă, incapacitatea pieței de a depăși un nivel cheie de rezistență sau suport.

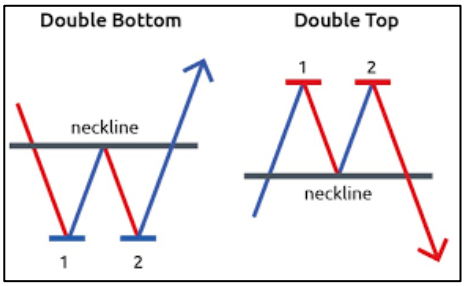


Figura 5. Exemplu Double Top & Double Bottom

2.2.5. Inside Bar Strategy

Inside Bar este o formațiune de consolidare, în care o lumânare are corpul și umbrele complet încadrate în intervalul lumânării anterioare [vezi exemplul din Figura 6]. Acest tipar indică, de obicei, o perioadă de indecizie, urmată adesea de un breakout (o spargere) în direcția trendului major.

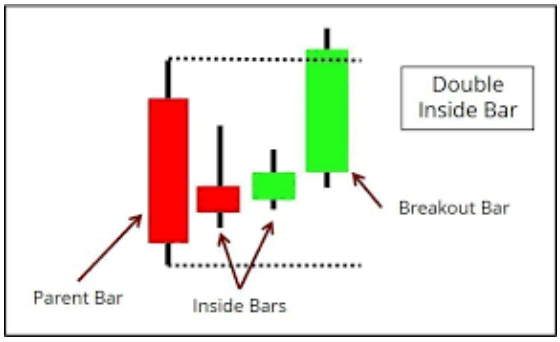


Figura 6. Exemplu Inside Bar Strategy

2.2.6. SMA Crossover

Strategia SMA Crossover [vezi Figura 7] utilizează două medii mobile simple pentru a genera semnale. Atunci când media mobilă scurtă (ex. 50 perioade) traversează în sus media mobilă lungă (ex. 200 perioade), se generează un semnal de cumpărare. În schimb, când linia scurtă intersectează în jos linia lungă, semnalul este de vânzare.

Figura 7. Exemplu SMA Crossover

2.2.7. EMA Crossover

Similară cu strategia SMA, varianta EMA Crossover folosește medii mobile exponențiale, care reacționează mai rapid la schimbările de preț. În acest fel, semnalele pot apărea mai devreme, dar și cu riscul unor alarme false mai frecvente.

2.2.8. RSI Overbought/Oversold

RSI (Relative Strength Index) este un indicator oscilator care variază între 0 și 100 și măsoară viteza mișcărilor de preț [vezi Figura 8]. Valorile peste 70 indică o piață „overbought" (supra-cumpărată), sugerând un potențial semnal de vânzare, iar valorile sub 30 indică o piață „oversold" (supra-vândută), semnalând o posibilă oportunitate de cumpărare.

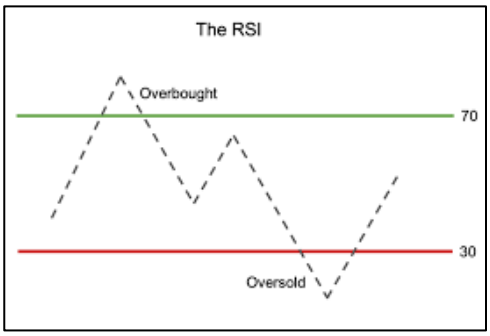


Figura 8. Exemplu RSI Overbought/Oversold

2.2.9. Bollinger Bands

Benzile Bollinger [vezi Figura 9] sunt formate dintr-o medie mobilă centrală și două benzi plasate la o distanță de două deviații standard deasupra și dedesubt. Atunci când prețul atinge banda superioară, se poate interpreta ca un semnal de vânzare, iar atingerea benzii inferioare poate indica un moment oportun pentru cumpărare.

Figura 9. Exemplu Bollinger Bands

2.2.10. Breakout Strategy

Strategia de tip Breakout se bazează pe tranzacționarea străpungerilor de niveluri cheie de suport sau rezistență; cele două niveluri fiind folosite ca bariere imaginare în care prețul se oprește în rânduri repetate. Suportul este bariera inferioară în timp ce rezistența este bariera superioră de preț. După o perioadă de consolidare, depășirea acestor niveluri poate genera mișcări puternice în direcția ruperii [vezi Figura 10].

Figura 10. Exemplu Breakout Strategy

2.2.11. Donchian Channel

Indicatorul Donchian Channel trasează trei linii: maximul și minimul ultimelor N perioade, precum și media dintre acestea. Atunci când prețul depășește maximul canalului, se generează un

semnal de cumpărare, iar când sparge minimul, un semnal de vânzare. Acest indicator este utilizat frecvent în strategiile de tip trend-following pentru a lua poziții în direcția trendului [vezi Figura 11].

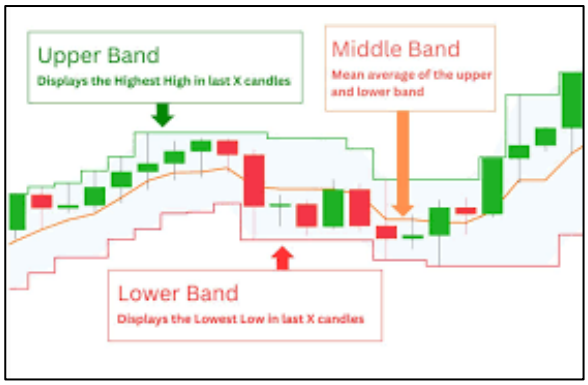


Figura 11. Exemplu Donchian Channel

2.3. Stadiul actual în literatura de specialitate

Literatura de specialitate din domeniul tradingului algoritmic și al sistemelor de backtesting evidențiază o evoluție constantă a soluțiilor software, în paralel cu complexificarea piețelor financiare. Mai multe lucrări recente au contribuit la dezvoltarea unor cadre teoretice și practice pentru testarea strategiilor și analizarea performanței, însă majoritatea acestor lucrări sunt fie specializate pe un subset limitat de funcționalități, fie dificil de aplicat pentru utilizatorii individuali.

În [1], Kostiainen propune un mediu modular pentru backtesting, bazat pe date istorice extrase prin aplicații existente și gestionate local. Lucrarea evidențiază faptul că investitorii pot dezvolta proprii algoritmi fără a depinde de platforme comerciale, dar sistemul este limitat la o utilizare statică, fără integrare cu surse de date dinamice sau interfață vizuală modernă. Prin comparație, proiectul TradeMind își propune să extindă această abordare prin integrarea în timp real cu API-uri externe și printr-o experiență vizuală intuitivă.

O abordare mai accesibilă este propusă de Sarasa-Cabezuelo în [2], care dezvoltă o aplicație web pentru backtesting și gestionarea strategiilor de investiții. Soluția este orientată către simplitate și accesibilitate, permițând utilizatorilor să-și definească și partajeze portofolii. Cu toate acestea, aplicația nu include un sistem robust de analiză a performanței, iar integrarea cu surse de date externe este limitată. Lucrarea TradeMind urmărește să compenseze aceste lipsuri, oferind atât posibilități de personalizare a strategiilor, cât și analize detaliate ale rezultatelor.

În [3], Nagaria și colaboratorii săi oferă o analiză cantitativă a trei strategii de tranzacționare - Moving Average, Bollinger Bands și Ichimoku Cloud - folosind un terminal propriu de backtesting. Lucrarea subliniază importanța unor indicatori-cheie precum drawdown-ul maxim și raportul risc-recompensă. Concluziile evidențiază diferențele de comportament între strategii și relevanța ajustării parametrilor. Această perspectivă este integrată și în TradeMind, printr-un modul care permite evaluarea vizuală și numerică a performanței, precum și ajustarea dinamică a parametrilor.

Vezeris et al., în [4], propun o metodologie inovatoare de backtesting dinamic, bazată pe selecția optimă a perioadelor istorice pentru parametrii tehnici. Lucrarea demonstrează că adaptarea continuă la condițiile pieței poate crește semnificativ profitabilitatea. Acest principiu este valoros și pentru proiectul TradeMind, unde se va urmări dezvoltarea unui algoritm simplificat de ajustare a parametrilor în funcție de evoluția rezultatelor obținute în timp real.

În [5], Pereira explorează automatizarea și testarea Perfect Order Strategy, o abordare algoritmică ce se bazează pe sincronizarea mai multor medii mobile pentru identificarea punctelor

optime de intrare și ieșire. Lucrarea evidențiază avantajele utilizării backtesting-ului pentru validarea robustă a strategiei și subliniază importanța disciplinei în ajustarea parametrilor. Această perspectivă întărește direcția proiectului TradeMind, care integrează posibilitatea de definire și testare flexibilă a strategiilor de tranzacționare în raport cu date istorice.

În concluzie, literatura de specialitate oferă numeroase perspective complementare asupra tradingului automatizat și al evaluării strategiilor. Cu toate acestea, nicio lucrare analizată nu propune o soluție integrată, scalabilă și prietenoasă cu utilizatorii neexperimentați, care să combine backtesting, analiză de performanță, gestionarea tranzacțiilor și generarea de statistici. Proiectul TradeMind își propune tocmai să umple acest gol, oferind un sistem unificat care să sprijine traderii individuali în optimizarea deciziilor lor financiare.

2.4. Analiza soluțiilor existente

Pe lângă cercetarea academică, dezvoltarea aplicațiilor de tip backtesting și analiză financiară s-a concretizat și în numeroase soluții comerciale sau open-source. Aceste platforme sunt utilizate frecvent de traderi, analiști și dezvoltatori pentru a testa și implementa strategii de tranzacționare. În cele ce urmează, sunt analizate câteva dintre cele mai cunoscute și relevante soluții din această categorie și de asemenea este prezentată o analiză comparativă între acestea în [Tabelul 1].

2.4.1. MetaTrader 4 și 5

MetaTrader (cu versiunile MT4 și MT5) este una dintre cele mai utilizate platforme în tradingul retail. Oferă suport pentru automatizarea strategiilor prin Expert Advisors (EAs), instrumente de analiză tehnică și un sistem de backtesting integrat. Cu toate acestea, backtesting-ul este limitat de viteza de execuție, complexitatea limbajului MQL și de lipsa unor metode vizuale avansate pentru comparația între strategii. În plus, nu oferă un mod intuitiv de gestionare a documentelor asociate fiecărei strategii.

2.4.2. TradingView

TradingView este una dintre cele mai populare platforme pentru analiza grafică și testarea strategiilor, folosită pe scară largă de traderi la nivel global. Este apreciată pentru interfața sa modernă și ușor de utilizat, dar și pentru limbajul de scripting propriu, Pine Script, care permite dezvoltarea de indicatori și strategii personalizate. Un alt avantaj important este accesibilitatea - funcționează direct în browser și nu necesită instalarea unor aplicații suplimentare. Totuși, platforma are anumite limite în ceea ce privește costurile foarte ridicate pentru accesul la o cantitate mai mare de date istorice sau personalizarea avansată a fluxurilor informaționale.

2.4.3. QuantConnect

QuantConnect este o platformă open-source construită peste motorul de backtesting LEAN. Oferă funcționalități avansate pentru utilizatorii tehnici, incluzând suport pentru limbaje precum Python și C#. Cu toate acestea, bariera de intrare este ridicată, interfața este adresată în special programatorilor, iar funcționalitatea educațională și de document management este aproape inexistentă. TradeMind, prin contrast, vizează o experiență de utilizare intuitivă, cu accent pe accesibilitate și vizualizare clară.

2.4.4. Amibroker

Amibroker este o platformă avansată de analiză tehnică și testare de strategii, folosită în special de utilizatorii avansați. Suportă backtesting rapid, optimizare și scripting propriu (AFL -

Amibroker Formula Language). Totuși, interfața sa este percepută ca fiind învechită și mai greu de utilizat pentru începători. TradeMind urmărește să acopere această zonă printr-o interfață modernă, construită în React, care oferă și funcții educaționale și asistență la configurare.

2.4.5. Alte soluții populare

Platforme precum NinjaTrader, MultiCharts sau Backtrader (în Python) oferă diverse niveluri de flexibilitate, dar suferă de probleme similare: fie sunt greu de utilizat pentru utilizatorii non tehnici, fie sunt dependente de date și infrastructuri comerciale costisitoare.

Funcționalitate	MetaTrader	TradingView	QuantConnect	Amibroker	TradeMind
Tip platformă	Comercială (desktop)	Comercială (web+desktop)	Open-source	Comercială (desktop)	Aplicație web, containerizată
Backtesting integrat	Da, dar limitat	Nu	Da, foarte avansat	Da, rapid și optimizat	Da, cu date istorice din API-uri
Ușurință de utilizare	Medie	Ridicată	Redusă (adresată programatorilor)	Medie–Redusă (pentru avansați)	Ridicată (dashboard intuitiv)
Viteză backtesting	Scăzută–Medie	-	Mare	Mare	Medie–Mare (optimizabil prin Docker)
Integrare date externe	Limitată	Limitată	Foarte bună	Limitată	Yahoo Finance, Binance, MT5 API
Interfață grafică modernă	Nu	Da	Nu	Nu	Da
Orientare utilizatori	Retail traders	Începători și intermediari	Programatori și utilizatori tehnici	Utilizatori avansați	Traderi individuali, amatori/semi-pro

Tabelul 1. Analiză comparativă între mai multe soluții existente și soluția propusă TradeMind

2.5. Limitări ale aplicațiilor existente

Deși piața aplicațiilor pentru trading și testarea strategiilor este variată și în continuă expansiune, majoritatea soluțiilor analizate atât în literatura de specialitate, cât și în practica utilizatorilor finali prezintă o serie de limitări semnificative. Acestea devin cu atât mai evidente atunci când sunt utilizate de traderi individuali, cu un nivel mediu de experiență sau cu resurse tehnice limitate, care nu dispun de infrastructură avansată sau de cunoștințe solide de programare.

2.5.1. Lipsa unei soluții integrate all-in-one

Cele mai multe aplicații existente sunt specializate pe un subset restrâns de funcționalități, fără a oferi un sistem complet. De exemplu, unele se concentrează exclusiv pe backtesting (precum Backtrader), altele pe execuție (precum MetaTrader), doar foarte puține îmbină testarea, monitorizarea performanței și gestionarea riscului într-un cadru unitar. Această fragmentare obligă traderul să alterneze între mai multe platforme, să exporte/importe date manual și să gestioneze rezultate dispersate. În lipsa unei integrări coerente, fluxul de lucru devine mai greoi, cu potențiale erori și întârzieri care afectează deciziile de tranzacționare.

2.5.2. Accesibilitate redusă pentru utilizatorii non-tehnici

Platforme precum QuantConnect sau Amibroker pun la dispoziție un set foarte avansat de instrumente, însă presupun un nivel ridicat de cunoștințe tehnice, în special de programare (Python, C#, MQL) sau matematică financiară. Din această cauză, ele devin inaccesibile pentru o categorie largă de traderi retail, care deși au o înțelegere bună a piețelor și a conceptelor de bază, nu pot implementa sau testa strategii fără asistență suplimentară. Lipsa unor interfețe intuitive și a unui ghidaj pas cu pas determină o curbă de învățare abruptă, ceea ce descurajează utilizarea lor pe termen lung.

2.5.3. Limitări în personalizarea strategiilor și a testării

Chiar și platformele globale, precum MetaTrader, impun limitări semnificative în definirea strategiilor personalizate, fiind necesară utilizarea unor limbaje dedicate (ex. MQL4/MQL5). În același timp, soluțiile online, precum TradingView, restricționează simularea la anumite tipuri de date (de exemplu doar timeframe-uri zilnice pentru perioade îndepărtate). În plus, puține dintre aplicații permit modificarea logicilor de intrare/ieșire în timp real sau compararea vizuală a mai multor strategii pe aceleași date. Acest lucru reduce potențialul de optimizare și limitează adaptabilitatea traderului la condiții de piață dinamice.

2.5.4. Lipsa funcționalităților educaționale și de suport decizional

O parte semnificativă a utilizatorilor sunt traderi aflați la început de drum, pentru care aspectele educaționale sunt esențiale. Cu toate acestea, foarte puține aplicații integrează mecanisme de învățare asistată: explicații detaliate ale semnalelor, vizualizări grafice intuitive sau feedback vizual pentru deciziile luate. Lipsa acestor elemente face ca multe platforme să fie percepute ca greu de folosit. În contrast, TradeMind adresează această problemă prin introducerea de rapoarte explicative, grafice clare și posibilitatea de a adăuga fotografii fiecărei tranzacții simulate, transformând aplicația într-un instrument de învățare practică, nu doar de testare.

2.5.5. Costuri ridicate și lipsă de transparență

Un alt obstacol important îl reprezintă costurile. Multe platforme comerciale condiționează accesul la funcționalitățile avansate de abonamente premium, care pot depăși cu mult bugetul unui trader individual. În plus, unele aplicații nu sunt transparente în privința algoritmilor de testare utilizați sau a datelor istorice disponibile, ceea ce ridică întrebări legate de încredere și replicabilitate. În absența acestor informații, utilizatorul nu poate ști dacă rezultatele backtesting-ului reflectă realitatea pieței sau sunt influențate de limitări interne ale platformei.

Aplicațiile existente pentru trading și backtesting, deși mature și răspândite în industrie, prezintă în continuare limitări importante legate de fragmentarea funcționalităților, accesibilitatea pentru utilizatorii non-tehnici, gradul de personalizare al strategiilor, lipsa suportului educațional și costurile ridicate. Toate aceste aspecte creează un context în care traderii individuali se confruntă cu bariere semnificative, fie în ceea ce privește utilizarea zilnică, fie în procesul de învățare și optimizare a strategiilor. Aceste limitări justifică dezvoltarea unor soluții noi, capabile să îmbine testarea strategiilor cu analiza statistică, vizualizări grafice și mecanisme de suport decizional. În acest cadru se încadrează și platforma TradeMind, care își propune să ofere un sistem unitar și accesibil pentru testarea, analiza și perfecționarea strategiilor de tranzacționare.

Capitolul 3. Soluția propusă

3.1. Problema abordată și strategia generală de rezolvare

În contextul actual al piețelor financiare, activitatea de tranzacționare se confruntă cu mai multe dificultăți:

- volumul mare de date și informații ce trebuie analizate și filtrate,
- lipsa unor instrumente unificate pentru gestionarea tranzacțiilor și strategiilor,
- dificultatea de a testa rapid ipoteze și strategii noi,
- accesul limitat la soluții profesioniste, multe dintre ele fiind costisitoare sau greu de personalizat,
- fragmentarea procesului între mai multe aplicații (platforme de trading, fișiere Excel, etc).

Aceste limitări fac ca traderii individuali să aibă o experiență dificilă, bazată adesea pe decizii incomplete sau pe resurse terțe, care nu se potrivesc neapărat nevoilor lor.

În lipsa unei soluții accesibile tuturor amatorilor aceștia ajung fie să își investească banii fără nici un fel de strategie în prealabil, fie ajung să plătească sume uriașe așa numiților guru de pe rețelele sociale care promit rezultate peste noapte - așteptări nerealiste, fie se avântă singuri pe acest drum necunoscut.

3.1.1. Scopul soluției propuse

Scopul platformei TradeMind este de a dezvolta o aplicație web accesibilă și ușor de utilizat, care să sprijine traderii în organizarea și analiza activității lor. Platforma are rolul de a centraliza datele, de a oferi instrumente vizuale de interpretare și de a facilita testarea strategiilor de tranzacționare pe baza datelor istorice.

Astfel, utilizatorii pot:

- gestiona și vizualiza datele conturilor de tranzacționare (prin conexiunea cu MT5),
- înregistra și analiza tranzacțiile personale,
- crea strategii pornind de la modele predefinite,
- rula backtesting pe date provenite din API-uri externe (Yahoo Finance, Binance),
- vizualiza rezultatele sub formă de statistici și rapoarte clare.

3.1.2. Strategia generală de rezolvare

Strategia de rezolvare se bazează pe următoarele direcții:

- **Centralizarea datelor** - integrarea într-o singură platformă a tuturor informațiilor necesare desfășurării activității de tranzacționare;
- **Automatizarea analizei** - implementarea unui motor de backtesting configurabil, care să evalueze performanța strategiilor după indicatori profesionali (profit, drawdown, raport risc/recompensă, etc.);
- **Accesibilitate și vizualizare intuitivă** - interfață prietenoasă, care să ofere un dashboard central cu grafice și rapoarte;
- **Scalabilitate și siguranță** - infrastructură containerizată și mecanisme de autentificare securizată.

În acest mod, soluția propusă urmărește să reducă barierele tehnice întâmpinate de utilizatorii fără experiență și să creeze o platformă completă, dar ușor de utilizat.

3.1.3. Contribuții personale și elemente originale

Proiectul TradeMind aduce mai multe elemente de noutate și contribuții proprii, care îl diferențiază atât de soluțiile comerciale existente, cât și de platformele open-source orientate exclusiv către programatori.

- Un prim element de originalitate îl constituie integrarea într-un singur sistem a trei direcții fundamentale ale activității de trading:
  - gestionarea conturilor de tranzacționare și a tranzacțiilor personale,
  - crearea și testarea strategiilor pe date istorice,
  - vizualizarea și interpretarea performanței printr-un sistem de statistici intuitive.

- Un al doilea element este flexibilitatea în configurarea strategiilor de tranzacționare. Aplicația nu obligă utilizatorul să scrie cod sau să înțeleagă limbaje de programare specifice (precum MQL sau Pine Script), ci îi permite să pornească de la un set de modele predefinite și să ajusteze parametrii printr-o interfață intuitivă. Astfel, un trader amator poate experimenta diferite scenarii și poate înțelege impactul modificărilor asupra performanței strategiei, fără să depindă de cunoștințe tehnice avansate.

- Un alt aspect important este componenta educațională implicită a aplicației. Prin modul în care sunt afișate rapoartele și indicatorii cheie (profit, drawdown, raport risc/recompensă etc.), utilizatorii pot dobândi o înțelegere mai bună a conceptelor fundamentale din trading și pot învăța din propriile rezultate. Vizualizarea clară a evoluției contului și a performanței strategiilor contribuie la dezvoltarea unui proces decizional mai informat, bazat pe date, și nu pe intuiții sau recomandări externe.

Prin aceste elemente, proiectul TradeMind nu se limitează la reproducerea funcționalităților deja existente pe piață, ci propune o viziune unificată și educațională asupra tradingului, adaptată nevoilor reale ale utilizatorilor amatori.

3.2. Cerințe și așteptări ale utilizatorilor

3.2.1. Cerințe funcționale

- **Gestionarea conturilor de tranzacționare** - platforma trebuie să permită utilizatorilor conectarea și vizualizarea datelor provenite din conturile lor reale de tranzacționare (MT5);
- **Gestionarea trade-urilor personale** - utilizatorii trebuie să poată adăuga, vizualiza și organiza manual tranzacții personale, împreună cu un set de statistici relevante;
- **Crearea strategiilor personalizate** - aplicația trebuie să ofere posibilitatea de a defini strategii noi, pornind de la cele existente în baza de date și ajustând parametrii acestora;
- **Backtesting configurabil** - sistemul trebuie să permită rularea de backtesting pe baza datelor istorice obținute prin API-uri financiare (ex. Yahoo Finance, Binance), cu posibilitatea reglării parametrilor strategiilor;
- **Vizualizarea rezultatelor** - platforma trebuie să afișeze rapoarte vizuale și numerice (grafice, tabele, indicatori) care să includă elemente precum profitul, drawdown-ul, raportul risc-recompensă și alți indicatori de performanță;
- **Autentificarea și securitatea** - aplicația trebuie să includă mecanisme de autentificare cu parolă criptată și validare prin email (confirmare email, resetare parolă prin link securizat);

- **Interfață intuitivă** - sistemul trebuie să ofere o experiență prietenoasă, cu navigare ușoară și funcționalități clar structurate, accesibile și pentru utilizatori fără pregătire tehnică avansată.

3.2.2. Cerințe nefuncționale

- **Scalabilitate și modularitate** - arhitectura sistemului trebuie să permită adăugarea ușoară de noi module sau funcționalități;
- **Securitate și autentificare** - accesul la datele personale și strategiile utilizatorului trebuie să fie protejat printr-un sistem sigur de autentificare (JWS);
- **Compatibilitate multiplatformă** - sistemul va fi accesibil prin browser, indiferent de dispozitiv (desktop/tabletă);
- **Performanță și timp de răspuns redus** - rularea simulărilor și afișarea rezultatelor trebuie să se realizeze rapid, inclusiv pentru seturi de date mari;
- **Portabilitate și containerizare** - sistemul va fi livrabil printr-un setup Docker, facilitând instalarea și rularea în medii locale sau cloud.

3.2.3. Profilul utilizatorului final și așteptările acestuia

Utilizatorul final vizat de TradeMind este traderul individual aflat la nivel de amator sau semi-profesionist, pasionat de piețele financiare, dar care nu dispune de infrastructuri profesionale sau cunoștințe avansate de programare. Acesta are nevoie de o soluție intuitivă, accesibilă și sigură, care să centralizeze informațiile, să simplifice analiza performanței și să ofere posibilitatea de testare a strategiilor într-un mod rapid și vizual.

Pentru o imagine de ansamblu, în [Tabelul 2] sunt sintetizate principalele caracteristici ale utilizatorului și așteptările sale de la platforma TradeMind.

Caracteristici utilizator	**Așteptări de la aplicație**
Trader individual - amator sau semi-profesionist	O platformă accesibilă, ușor de utilizat fără cunoștințe tehnice avansate
Resurse limitate - nu are acces la infrastructuri financiare sofisticate	Cost redus și funcționalități integrate într-o singură aplicație
Nu cunoaște limbaje de programare	Posibilitatea de a crea și personaliza strategii prin interfață grafică, fără cod
Preferă vizualizarea datelor într-o formă clară	Dashboard intuitiv cu rapoarte, grafice și statistici relevante
Are nevoie de decizii rapide pe baza testelor	Rezultate imediate la backtesting și feedback vizual instant
Este preocupat de siguranța datelor	Autentificare securizată, validare prin email, protecția informațiilor financiare

Tabelul 2. Profilul utilizatorului final TradeMind

3.2.4. Actorii sistemului

În cazul platformei TradeMind, există un singur actor principal: utilizatorul final - traderul individual. Acesta este punctul central al sistemului, pentru că toate funcționalitățile sunt concepute pentru a răspunde nevoilor sale. Traderul interacționează direct cu platforma prin intermediul interfeței web și poate întreprinde toate activitățile disponibile pe platformă:

- gestionarea conturile de tranzacționare;
- introducerea și organizarea manuală a tranzacțiilor proprii;

- crearea strategiilor personalizate și testarea acestora în sesiuni de backtesting;
- vizualizarea rapoartelor și statisticilor de performanță;
- securizarea accesul la aplicație prin autentificare și validarea email-ului.

3.3. Arhitectura și modelarea sistemului

3.3.1. Arhitectura generală

Platforma TradeMind a fost concepută pe baza unei arhitecturi web împărțită în trei componente majore: Frontend, Backend și Baza de date. Arhitectura generală urmează modelul client–server, unde frontend-ul (clientul) comunică cu backend-ul (serverul aplicației) prin apeluri HTTP securizate. Backend-ul gestionează logica de business, interacțiunea cu baza de date și integrarea cu surse externe, în timp ce frontend-ul oferă utilizatorului o interfață vizuală prietenoasă și interactivă.

- La nivel de Backend, s-a ales o arhitectură monolitică pe trei straturi (Prezentare - Logică – Persistență), pentru a asigura o separare clară a responsabilităților și o mentenanță mai ușoară. Această abordare permite extinderea ulterioară a aplicației și integrarea de noi module fără a afecta structura deja implementată.
- Frontend-ul, dezvoltat în React cu TypeScript, respectă o arhitectură modulară pe funcționalități (feature-based). Fiecare modul acoperă o zonă distinctă a aplicației (de exemplu: gestionarea trade-urilor, rularea backtesturilor, vizualizarea statisticilor), ceea ce contribuie la scalabilitatea și lizibilitatea proiectului.
- Baza de date utilizată este PostgreSQL, aleasă pentru robustețea și stabilitatea ei în gestionarea datelor structurate. Ea stochează informațiile despre utilizatori, conturi de tranzacționare, tranzacții, strategii, statistici și rezultatele backtesturilor.

Acestea sunt completate de servicii externe (API-uri financiare, MT5 API, SMTP) care facilitează accesul la date de piață, conectarea la conturi reale de tranzacționare și trimiterea notificărilor prin email.

Arhitectura generală este ilustrată schematic în [Figura 12], unde sunt evidențiate fluxurile de date dintre componente și modul în care utilizatorul interacționează cu platforma.

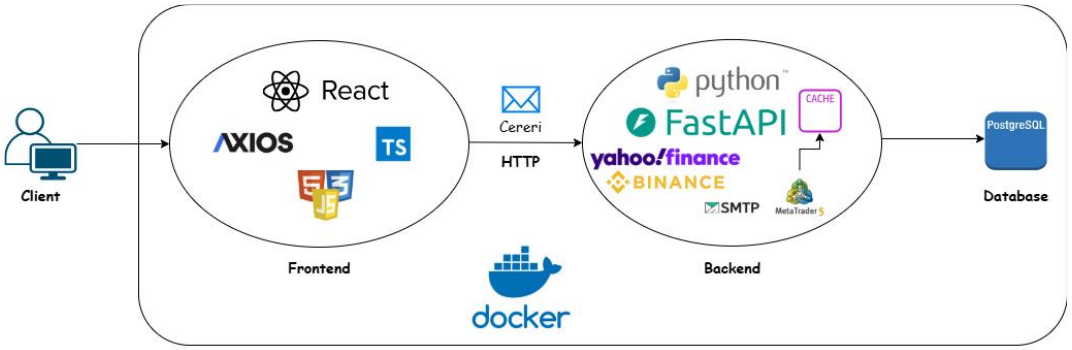


Figura 12. Arhitectura Top-Level a aplicației

Un aspect esențial este gestionarea conexiunii cu serverul MT5. Deoarece apelurile directe către server pot fi costisitoare și pot genera latențe, aplicația implementează un mecanism de cache. O parte din datele provenite din MT5 sunt stocate temporar în memoria backend-ului pentru a fi reutilizate rapid în interogările ulterioare. Astfel:

- se reduc apelurile repetitive către serverul MT5;

- se îmbunătățește timpul de răspuns pentru utilizator;
- se optimizează consumul de resurse al aplicației.

În plus, sistemul integrează și alte servicii externe esențiale:

- API-uri financiare (Yahoo Finance, Binance) - pentru preluarea datelor istorice și actuale necesare backtestării;
- SMTP - pentru trimiterea emailurilor automate (confirmare cont, resetare parolă).

3.3.2. Componentele principale și interacțiunea dintre ele

3.3.2.1. Frontend-ul

Frontend-ul este realizat folosind React împreună cu TypeScript, completat de tehnologii web standard (HTML, CSS, JavaScript). Rolul său principal este de a oferi utilizatorului o interfață intuitivă pentru gestionarea conturilor de tranzacționare, vizualizarea statisticilor și efectuarea testelor de backtesting pe strategii publice sau personalizate de către utilizator.

Pentru comunicarea cu serverul backend, aplicația utilizează biblioteca Axios, care trimite cereri HTTP către rutele expuse de FastAPI. Totodată, pentru vizualizarea graficelor financiare și a rezultatelor backtesting-ului este integrată biblioteca Lightweight Charts, ce permite redarea unor grafice interactive de tip candlestick și linii de performanță.

Proiectul este dezvoltat în mediul de lucru WebStorm (JetBrains), care oferă suport nativ pentru React și TypeScript, facilitând organizarea modulară a codului și debugging-ul.

Structura proiectului este modulară și organizată pe directoare tematice, ceea ce permite dezvoltarea și extinderea rapidă a aplicației. Astfel, cele mai importante module și directoare sunt:

- components/ - include componente reutilizabile:
  - footer/, nav_bar/, side_menu/ - elemente de navigare și schelet de pagină.
  - ThemeContext.tsx - comutarea temei și partajarea preferințelor vizuale la nivelul întregii aplicații;
- configuration/ - conține fișiere de configurare precum AxiosConfigurations.ts și utilități pentru autentificare (UseAuth.ts);
- pages/ – ecranări pe fluxuri majore, fiecare subdirector grupând pagini înrudite:
  - auth_pages/ - autentificare, înregistrare, recuperare parolă, verificare email la înregistrare;
  - backtesting_pages/ - inițiere/rulare backtest, listă de sesiuni salvate, detalii și statistici ale unei sesiuni;
  - economic_calendar_pages/ - afișare evenimente macro-economice și filtrare după relevanță/interval;
  - help_page/ - ghiduri, întrebări frecvente, instrucțiuni de utilizare;
  - home_page/ - pagina de pornire;
  - settings_page/ - configurări cont și parolă;
  - statistics/ - tablouri general de statistici, curbe de evoluție a capitalului, tabele de tranzacții, agregări pe instrument/zi/direcție în piață;
  - trading_accounts_pages/ - adăugare/gestionare conturi, panou per cont, tranzacții și istoric;
- styles/ - definește stilurile CSS globale și pe cele folosite pentru formulare și dashboard-uri;
- types/ - tipuri de date folosite în întregul frontend (entități precum conturi, trade-uri, sesiuni de backtest, payload-uri de cerere și răspuns etc.).

3.3.2.2. Backend-ul

Backend-ul aplicației reprezintă nucleul logicii de business și este responsabil de gestionarea datelor, implementarea algoritmilor de backtesting și integrarea cu sursele externe de date. Arhitectura backend-ului este de tip monolitic stratificat, organizată în trei niveluri: Prezentare, Logică și Persistență. Această separare pe straturi permite izolarea responsabilităților, facilitând întreținerea și extinderea ulterioară a aplicației.

1. Stratul de prezentare

Stratul de prezentare are rol de interfață între frontend și logica aplicației, primește cereri, validează la nivel de interfață și returnează răspunsuri coerente; nu conține logică de domeniu.

Structura acestuia este următoarea:

- controllers/ - expun punctele de intrare ale aplicației, mapând rutele către operații de business:
  - backtest_controller.py, strategy_controller.py, trade_controller.py, trading_account_controller.py, statistic_controller.py, user_controller.py, chart_data_controller.py;
- pao/ (Presentation Access Object) - adaptor între controllere și servicii; grupează interfaces/ și services/ pentru orchestrarea apelurilor către stratul de logică (compunere de request/response, paginare, filtre, sortări);
- pal/ (Presentation Access Layer) - validări la nivel de payload, conversii locale, normalizări simple, înainte de trimiterea către stratul de logică.

2. Stratul de logică

Acest strat este punctul central al aplicației și conține implementările efective pentru toate funcționalitățile TradeMind.

Structura acestuia este următoarea:

- bal/ (Business Access Layer) - expune logica efectivă a aplicației către stratul de prezentare;
- bao/ (Business Access Object):
  - interfaces/ - metode abstracte definite pentru logica aplicației;
  - services/ - implementări pentru interfețele definite, aici se implementează efectiv logica aplicației;
- bto/ (Business Transfer Objects) - modele de transfer între straturile de logică și prezentare;
- mappers/ - mapează obiectele DTO în BTO și invers;
- utils/ - nucleul algoritmic și utilitare de domeniu:
  - backtest/
    - backtest_executor.py - motorul de simulare: parcurge candlestick-urile, cere semnale de la strategie, aplică reguli de intrare/ieșire, marchează tranzacții și calculează rezultatele; gestionează consistența temporală, conflictele de semnale și ordinele consecutive;
    - sl_calculator.py - calcul automat al stop-loss-ului folosind:
      - ATR - Average True Range - volatilitate medie (calculează volatilitatea medie și poziționează stop-loss-ul la o distanță proporțională);
      - structuri de preț (minime/maxime locale pe ultimul N);
      - fallback de volatilitate (percentile ale intervalelor High–Low). Alegerea finală este valoarea maximă dintre metodele valide.

- strategies/
  - base_strategy.py - interfață comună (inițializare, generare semnale, parametrizare);
  - strategy_registry.py - registru pentru instanțiere controlată a strategiilor;
    Implementări:
    - bazate pe indicatori tehnici (SMA și EMA crossover, Bollinger Bands, RSI overbought/oversold, MACD crossover, Breakout și Donchian Channel);
    - bazate pe pattern-uri și price action (Engulfing, Hammer & Shooting Star, Head and Shoulders, Double Top / Double Bottom, Inside Bar Strategy);

  Fiecare strategie expune parametri, condiții de intrare/ieșire și reguli de management al poziției; structura comună permite comutarea ușoară între strategii în timpul backtest-ului.
- chart_data_loader.py - gestionează preluarea și normalizarea datelor istorice din surse externe precum Yahoo Finance sau Binance. Acest modul transformă datele brute (OHLCV) în format unitar și compatibil cu aplicația, asigurând consistență pentru backtesting și analize ulterioare;

3. Stratul de persistență

Stratul de persistență gestionează stocarea permanentă a datelor în baza de date și este stratul de legătura între date și logica aplicației.

Structura acestuia este următoarea:

- dal/ (Data Access Layer) - expune funcționalitatea efectivă din dao/ către stratul de logică;
- dao/ (Data Access Object):
  - interfaces/ - metode abstracte definite pentru a accesa datele;
  - repositories/ - implementări pentru interfețele definite, aici se implementează efectiv operațiile cu baza de date;
- dto/ (Data Transfer Object) - relație 1 la 1 cu entitățile, modele de transfer între straturile de persistență și logică;
- entities/ - definește structura efectivă a tabelelor;
- mappers/ - mapează obiectele DTO în entități și invers;
- utils/ – suport transversal:
  - data_validators.py - validări de integritate la granița cu persistența;
  - encryption.py - utilitare pentru stocare sigură a datelor sensibile;

4. Module ajutătoare și integrare

- configurations/ - configurări la nivel de securizare prin JWS (JSON Web Signature);
- email_service/ - utilizat la trimiterea de notificări:
  - email_sender.py (expediere);
  - email_templates.py (șabloane),
  - verification_token.py (generare/verificare tokenuri pentru confirmări);
- mt5_service/ - integrare pentru import tranzacții și date de cont:
  - mt5_client.py (clientul);
  - mt5_main.py (scenarii de import).

3.3.2.3. Baza de date

Baza de date reprezintă componenta de stocare a platformei, fiind responsabilă de păstrarea în format persistent a tuturor informațiilor critice necesare funcționării aplicației. Ea gestionează entități precum utilizatorii, conturile de tranzacționare, tranzacțiile individuale, sesiunile de backtesting, strategiile definite și statisticile rezultate.

Această componentă nu este accesată direct de către interfața utilizator sau de către logica de business, ci prin intermediul stratului de persistență din backend (dao/repositories/). Astfel se realizează o izolare completă între nivelul de stocare și nivelul de procesare: logica aplicației poate evolua sau se poate extinde fără a depinde de detaliile tehnice de implementare din baza de date.

3.3.3. Modelarea bazei de date

Pentru stocarea datelor platformei TradeMind s-a utilizat un sistem de gestiune a bazelor de date relațional - PostgreSQL.

Datele sunt organizate în 6 tabele principale [vezi Figura 22], fiecare reflectând un aspect central al aplicației: utilizatori, conturi de tranzacționare, tranzacții (trades), strategii, statistici și backtest-uri. Aceste tabele sunt interconectate prin chei primare și chei străine, asigurând consistența datelor și permițând interogări complexe.

În continuare sunt prezentate detaliile fiecărei entități din baza de date.

3.3.3.1. Tabela users

Această tabelă gestionează informațiile despre utilizatorii platformei. Fiecare utilizator reprezintă un trader care își poate crea cont, adăuga strategii, salva statistici sau efectua backtest-uri. Rolul său principal este de a centraliza datele de identificare și autentificare ale utilizatorilor, asigurând mecanismele de securitate prin parolă criptată și verificarea contului prin email.

users		
Coloana	**Tip de date**	**Descriere**
id	integer (PK)	Identificator unic pentru fiecare utilizator (generat automat).
first_name	varchar	Numele utilizatorului.
last_name	varchar	Prenumele utilizatorului.
email	varchar (unic)	Adresa de email folosită pentru autentificare și comunicare.
password	varchar	Parola utilizatorului, stocată sub formă de hash criptat (SHA-256).
phone	varchar	Numărul de telefon al utilizatorului.
gender	varchar	Genul utilizatorului (masculin/feminin/altul).
country	varchar	Țara de proveniență a utilizatorului.
is_verified	boolean	Flag care indică dacă adresa de email a fost confirmată.

Tabelul 3. Structura tabelei *users*

Observații:

- Coloana email este unică pentru a preveni crearea mai multor conturi cu aceeași adresă.
- Coloana is_verified permite diferențierea între conturile activate și cele care încă nu au confirmat emailul.

3.3.3.2. Tabela trading_accounts

Această tabelă reține conturile de tranzacționare gestionate de utilizatori (demo sau reale). Fiecare cont este asociat unui utilizator și stochează datele minime necesare pentru a-l identifica în platformă (nume broker, id cont, parolă, server). Tabela permite centralizarea conturilor pentru care

se calculează statistici și se gestionează tranzacții.

trading_accounts		
Coloană	**Tip de date**	**Descriere**
id	integer	Identificator unic
user_id	integer	Referință către utilizatorul asociat
broker_name	varchar	Numele brokerului
account_id	bigint	ID-ul contului de tranzacționare
server	varchar	Serverul brokerului
password	varchar	Parola asociată contului

Tabelul 4. Structura tabelei *trading_accounts*

3.3.3.3. Tabela trades

Stochează tranzacțiile individuale. În practică, aceleași câmpuri acoperă atât tranzacțiile manuale, cât și pe cele simulate în urma unui backtest. Fiecare tranzacție aparține unui utilizator și, dacă provine din backtest, este legată de sesiunea respectivă. Structura actuală permite crearea de statistici complexe pe baza câmpurilor asociate fiecărui trade, impunându-se totodată și o serie de validări în cadrul acestora (validări temporale pentru data și ora de deschidere respectiv închidere a unui trade, validări de preț în funcție de tipul tranzacției și prețul de deschidere, etc.).

trades		
Coloană	**Tip de date**	**Descriere**
id	integer	ID unic al tranzacției
user_id	integer	Referință către utilizatorul care a realizat tranzacția
market	varchar(20)	Simbolul pieței (ex: EURUSD, DE30)
volume	numeric(10,2)	Volumul tranzacționat
type	tradetype	Tipul tranzacției (Buy, Sell, Unknown)
open_date	date	Data deschiderii tranzacției
open_time	time	Ora deschiderii tranzacției
close_date	date	Data închiderii tranzacției
close_time	time	Ora închiderii tranzacției
session	sessiontype	Sesiunea de tranzacționare (London, New York, Asia, Unknown)
open_price	numeric(14,5)	Prețul de deschidere
close_price	numeric(14,5)	Prețul de închidere
sl_price	numeric(14,5)	Prețul nivelului Stop Loss
tp_price	numeric(14,5)	Prețul nivelului Take Profit
swap	numeric(12,2)	Valoarea swap-ului
commission	numeric(12,2)	Comision aplicat tranzacției
profit	numeric(12,2)	Profitul/pierderea obținută
pips	numeric(12,2)	Numărul de pips (puncte) obținuți/pierduți
link_photo	varchar(150)	Link către o captură de ecran asociată tranzacției
source_type	sourcetype	Sursa tranzacției (User, Backtest, Unknown)
backtest_id	integer	Referință către sesiunea de backtest asociată

Tabelul 5. Structura tabelei *trades*

3.3.3.1. Tabela strategies

Această tabelă reprezintă catalogul de strategii disponibile în platformă. O strategie are un tip (enum) și un set de parametri care îi controlează comportamentul în backtesting. Strategiile pot fi

publice sau private (personalizate de un anumit utilizator).

strategies		
Coloană	**Tip de date**	**Descriere**
id	integer	ID unic al strategiei
name	varchar(50)	Numele strategiei
description	varchar(255)	Descrierea strategiei
type	strategytype	Tipul strategiei (SMA, EMA, RSI etc.)
parameters	json	Parametrii strategiei în format JSON
created_by	integer	ID-ul utilizatorului care a creat strategia
is_public	boolean	Indicator dacă strategia este publică sau privată
created_at	timestamp	Data și ora creării strategiei

Tabelul 6. Structura tabelei *strategies*

3.3.3.2. Tabela backtests

Tabela backtests descrie sesiunile de backtesting rulate de utilizatori. Conține referința la strategie, simbolul și intervalul testat, parametrii financiari inițiali și indicatorii sintetici calculați la final (profit total, drawdown, profit factor etc.). Curba de capital poate fi salvată ca serie pentru vizualizare ulterioară.

backtests		
Coloană	**Tip de date**	**Descriere**
id	integer	ID unic al backtestului
user_id	integer	Referință către utilizator
strategy_id	integer	Referință către strategia utilizată
symbol	varchar(20)	Simbolul instrumentului analizat
time_frame	varchar(20)	Intervalul de timp (de creare a unei candele) utilizat
start_date	timestamp	Data de început a backtestului
end_date	timestamp	Data de sfârșit a backtestului
initial_balance	double precision	Capitalul inițial al backtestului
risk_per_trade	double precision	Risc per tranzacție (%)
total_profit	double precision	Profitul total obținut
drawdown_max	double precision	Drawdown-ul maxim înregistrat
winrate	double precision	Procentajul de tranzacții câștigătoare
nr_trades	integer	Numărul total de tranzacții executate
profit_factor	double precision	Raport profit/pierdere (Profit Factor)
expectancy	double precision	Indicator de așteptare matematică a strategiei
created_at	timestamp	Data creării backtestului
name	varchar(150)	Numele atribuit backtestului
balance_curve	jsonb	Evoluția balanței contului în format JSON

Tabelul 7. Structura tabelei *backtests*

3.3.3.3. Tabela statistics

Tabela statistics păstrează seturi de indicatori/parametri salvați de utilizator pentru raportare sau reutilizare (preset-uri de filtre, layout-uri de dashboard, rezultatele periodice). Este gândită ca o zonă flexibilă, dependentă de utilizator astfel încât acesta să poată personaliza (în anumite limite) parametrii statisticii.

Coloană	Tip de date	Descriere
id	integer	ID unic al statisticii
user_id	integer	Referință către utilizator
name	varchar	Numele statisticii
params	json	Parametrii statistici în format JSON
is_active	boolean	Indicator dacă statistica este activă
created_at	timestamp	Data creării statisticii
updated_at	timestamp	Data ultimei actualizări

Tabelul 8. Structura tabelei statistics

3.3.4. Modelarea logică - Diagrame UML

3.3.4.1. Diagrama de cazuri de utilizare (Use Cases)

Diagrama de cazuri de utilizare evidențiază principalele interacțiuni dintre utilizator și site-ul web, de la autentificare până la rularea unui backtest sau analizarea statisticilor contului. Fluxul este ramificat, acoperind atât scenariile în care utilizatorul este nou (înregistrare și validare prin e-mail), cât și pe cele în care există deja un cont valid și se accesează funcționalitățile interne.

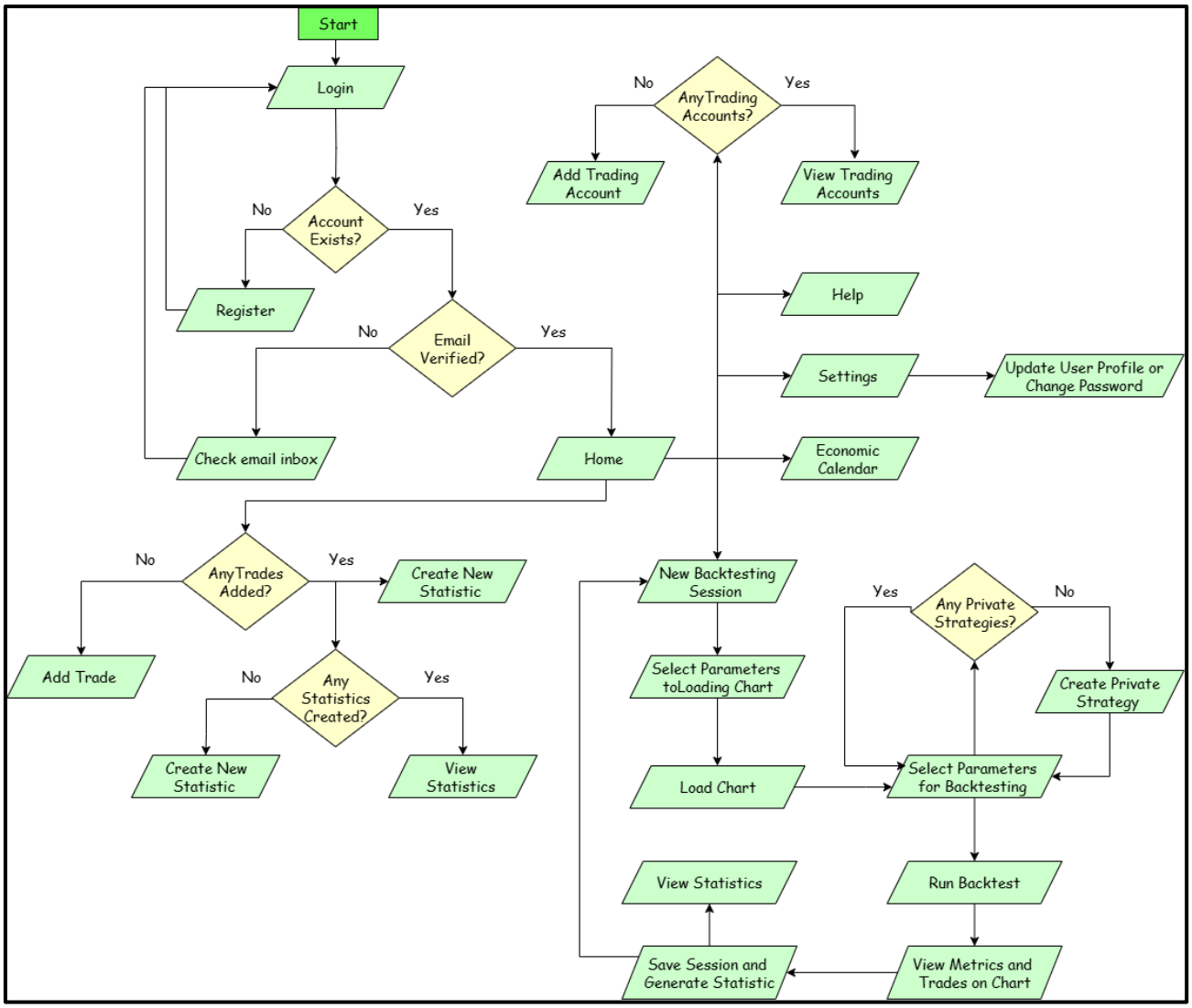


Figura 13. Diagrama cazurilor de utilizare

Cazuri de utilizare:

- Autentificare și înregistrare - verificarea contului, validarea e-mailului și accesul la pagina principală;
- Gestionare conturi și tranzacții - adăugarea conturilor de tranzacționare, introducerea sau importul tranzacțiilor și consultarea istoricului;
- Gestionare strategii și rulare backtesting - vizualizarea sau definirea strategiilor, configurarea unei sesiuni și obținerea rezultatelor pe baza datelor istorice;
- Creare statistici și rapoarte - analiza performanței prin indicatori, salvarea statisticilor și generarea de rapoarte.

Aceste interacțiuni sunt ilustrate schematic în [Figura 13] care prezintă succesiunea de pași și ramificațiile logice în funcție de acțiunile utilizatorului și de validările efectuate de sistem.

3.3.4.2. Diagrama de componente arhitecturale – Backend

Backend-ul aplicației este organizat pe trei straturi logice: Prezentare, Logică și Persistență. În stratul de prezentare, controllerele primesc cererile din partea clientului și, prin intermediul PAL și PAO, le validează și le transformă în apeluri către servicii. Stratul de logică conține logica principală a aplicației: servicii (BAO) care implementează interfețele definite, obiecte de transfer (BTO) și componentele utilitare care rulează backtest-uri și calculează indicatorii statistici. Stratul de persistență asigură interacțiunea cu baza de date prin DAO, DTO, entități și mappere, izolând logica de business de detaliile stocării.

Direcția fluxului este strict descendentă: Prezentare - Logică - Persistență. Stratul de prezentare nu are acces la entități sau DTO, iar stratul de logică interacționează cu stratul de persistență doar prin DAL. Stratul de persistență nu conține logică de domeniu, ci doar operații pe date și transformări. Această separare reduce complexitatea și crește flexibilitatea sistemului. Această schemă permite extinderea ușoară a funcționalităților, menținând în același timp coerența arhitecturală [Figura 14].

3.3.4.3. Diagrama modulară - Frontend

Frontend-ul aplicației este organizat pe module distincte, fiecare reflectând o funcționalitate majoră a platformei. Această împărțire asigură o structură clară și o experiență de utilizare coerentă, în care fiecare secțiune a interfeței se ocupă de un set specific de operații. Modulele includ atât zone dedicate autentificării și securității accesului, cât și secțiuni pentru conturi de tranzacționare, jurnale de tranzacții, dashboard-uri statistice și administrarea strategiilor sau sesiunilor de backtesting.

Diagrama modulară evidențiază faptul că aceste componente sunt dezvoltate independent, dar interconectate printr-un flux comun de date și interfețe. De exemplu, modulul Statistics and Dashboard folosește date colectate din modulele Trades (Journal) și Backtesting, pentru a genera rapoarte și grafice centralizate. În același timp, module precum Settings and Profile sau Help au rol de suport, asigurând personalizarea experienței și furnizarea de informații suplimentare utilizatorului.

Prin această organizare, aplicația devine ușor extensibilă: noi module pot fi adăugate fără a afecta structura existentă, iar fiecare componentă poate fi dezvoltată, testată și întreținută separat. Această structură este ilustrată în [Figura 15].

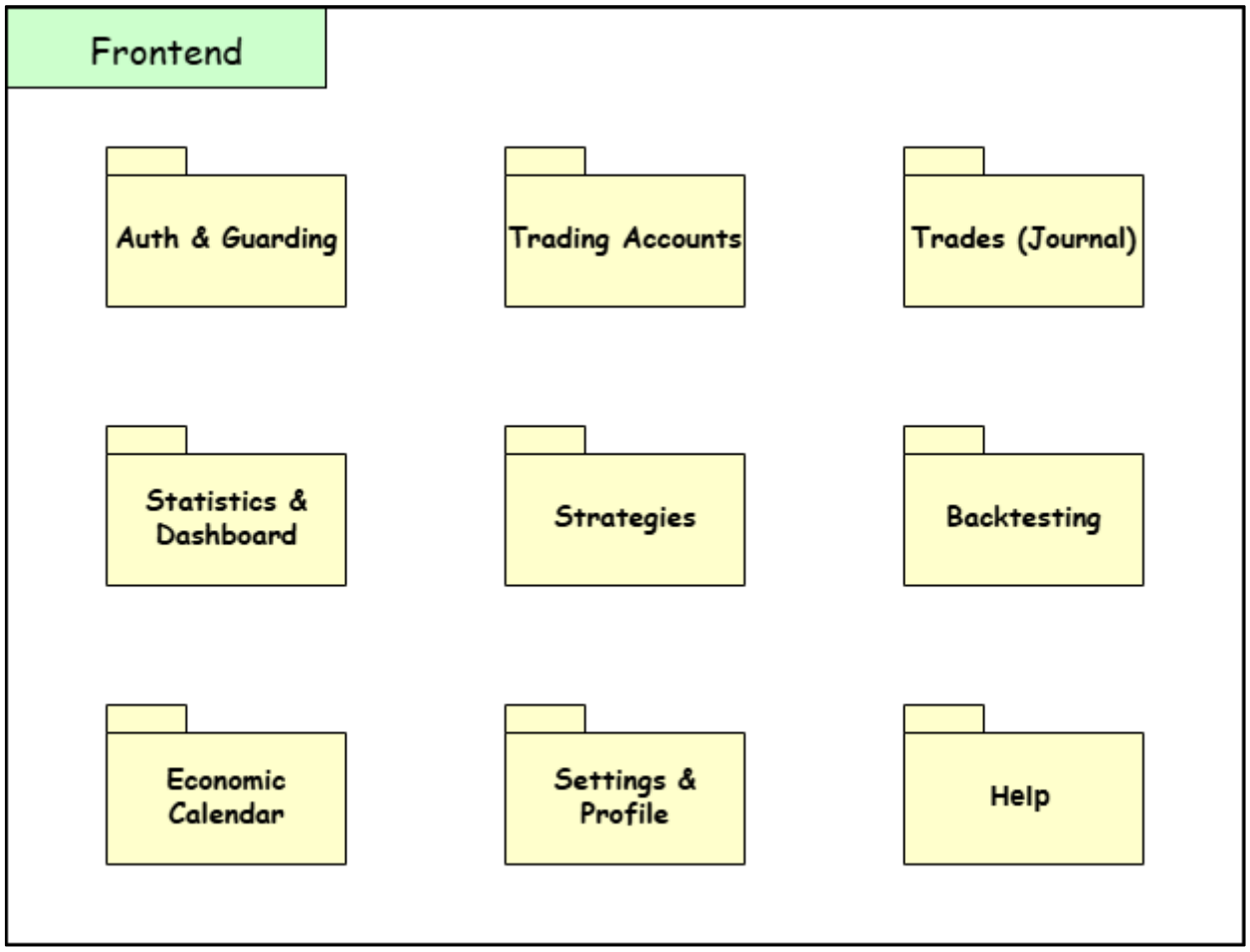


Figura 15. Diagrama modulară a frontend-ului

3.4. Tehnologii și justificări

Pentru dezvoltarea aplicației au fost utilizate tehnologii actuale, selectate astfel încât să asigure un echilibru optim între performanță, scalabilitate, securitate și ușurința de dezvoltare. Alegerea acestor tehnologii a ținut cont de un set de cerințe atât tehnice cât și funcționale (interfață interactivă, procesare rapidă a datelor, execuție de backtesting și analiză statistică) și de integrarea cu servicii externe relevante pentru domeniul în cauză.

Frontend

Partea de interfață cu utilizatorul este construită pe baza React, un framework pentru dezvoltarea aplicațiilor web de tip single-page application (SPA). Utilizarea sa asigură o experiență fluidă, fără reîncărcări complete ale paginilor și cu posibilitatea de a compune interfața din componente reutilizabile.

Pentru asigurarea tipizării și prevenirea erorilor, a fost utilizat TypeScript, care extinde JavaScript cu un sistem static de tipuri. Acest lucru permite o dezvoltare mai sigură și o mentenanță mai ușoară pe termen lung.

Comunicarea dintre interfață și serverul de backend este realizată prin Axios, o bibliotecă HTTP care facilitează gestionarea cererilor asincrone, interceptarea și tratarea erorilor, precum și atașarea automată a token-urilor de autentificare.

Alături de acestea, tehnologiile web de bază (HTML5 și CSS3) sunt folosite pentru structurarea și stilizarea aplicației. Ele asigură respectarea standardelor web și oferă suport pentru design responsiv, necesar accesului de pe diferite tipuri de dispozitive.

Backend

Logica principală a aplicației este implementată în Python, datorită ecosistemului său bogat în biblioteci și a versatilității în procesarea datelor. Pentru expunerea funcționalităților către frontend a fost utilizat FastAPI, un framework pentru dezvoltarea de API-uri REST. Acesta oferă timpi de răspuns foarte buni, suport nativ pentru validarea datelor și documentație automată a rutelor expuse, facilitând integrarea rapidă cu clientul.

Aplicația integrează servicii externe prin API-uri, printre care:

- Yahoo Finance și Binance: utilizate pentru preluarea datelor istorice necesare backtesting-ului;
- MetaTrader 5: pentru importul datelor din conturi de tranzacționare reale sau demo;
- SMTP: pentru trimiterea de e-mailuri (confirmări, resetări de parolă, notificări către utilizatori).

Pentru optimizarea timpilor de răspuns și reducerea numărului de interogări la sursele externe, este folosit un mecanism de cache. Acesta reține temporar datele cele mai frecvent accesate, îmbunătățind performanța generală a aplicației.

Bază de date

Persistența datelor este asigurată prin PostgreSQL, un sistem de gestiune a bazelor de date relaționale robust. Alegerea sa se justifică prin suportul extins pentru tipuri de date complexe, integritatea tranzacțională și capacitatea de a rula interogări complexe necesare pentru analize statistice. În plus, PostgreSQL oferă un bun echilibru între performanță și siguranța datelor, fiind potrivit pentru aplicații de orice nivel.

Infrastructură și orchestrare

Pentru rularea aplicației într-un mediu controlat și reproductibil, a fost utilizat Docker. Prin containerizare, fiecare componentă a sistemului (frontend, backend, bază de date) este izolată și configurată independent, ceea ce permite rularea aplicației pe orice mașină fără diferențe de mediu. De asemenea, Docker simplifică procesul de instalare, testare și scalare a aplicației.

Observație: Integrarea MetaTrader în mediul de containerizare s-a dovedit impracticabilă, întrucât aplicația necesită instalarea și rularea locală a terminalului MetaTrader, care nu dispune de o variantă nativă compatibilă cu Docker. Astfel, pentru a putea utiliza serviciile MetaTrader în cadrul proiectului, serverul aferent a fost menținut să ruleze direct pe mașina gazdă, unde aplicația este deja instalată, iar restul componentelor au rămas containerizate în Docker.

3.5. Implementare și dezvoltare

3.5.1. Etapele implementării

Procesul de dezvoltare al aplicației a fost realizat local, pentru a permite validarea rapidă a funcționalităților și adaptarea la cerințe. Etapele principale au fost:

- **Analiza și proiectarea arhitecturii** - definirea componentelor frontend, backend și baza de date, precum și stabilirea relațiilor dintre acestea;
- **Configurarea mediului de lucru** - setarea infrastructurii cu Docker, inițializarea proiectelor pentru frontend și backend, configurarea bazei de date PostgreSQL;
- **Dezvoltarea backend-ului** - implementarea structurii pe 3 straturi (Prezentare, Logică, Persistență), a modulelor de backtesting, strategii și statistici, precum și a serviciilor auxiliare (autentificare JWS, trimitere email, integrare cu surse externe);
- **Dezvoltarea frontend-ului** - realizarea interfeței web pe module (autentificare, conturi,

tranzacții, strategii, statistici, backtesting), integrarea cu API-urile backend și implementarea graficelor interactive;

- **Integrarea și testarea componentelor** - validarea fluxurilor end-to-end, verificarea consistenței datelor între straturi și ajustarea performanțelor (ex: cache pentru o serie de date externe);
- **Containerizarea și rularea** - orchestrarea aplicației prin Docker Compose, asigurând portabilitatea și posibilitatea de rulare pe orice mediu;
- **Optimizarea finală** - rafinarea interfeței, completarea rapoartelor statistice și validarea securității (hash parole, verificare email, protecție rute).

3.5.2. Funcții cheie

- **Funcția _calculate_metrics**

Această funcție [vezi _calculate_metrics] are rolul de a genera statistici detaliate pe baza unei liste de tranzacții efectuate de utilizator sau rezultate dintr-un backtest. Ea primește ca parametri lista de tranzacții (trades), un nume pentru statistica analizată (name) și eventuale filtre aplicate (filters). Rezultatul este o structură de tip dicționar care conține indicatorii calculați, împreună cu detalii despre distribuția tranzacțiilor.

Principalele valori returnate includ:

- **Indicatori generali**: număr total de tranzacții, winrate, profit și pierdere medie, profit total, profit/pierdere maximă;
- **Evoluția soldului**: prin construcția unei curbe cumulative a profiturilor (balance curve), care arată evoluția capitalului după fiecare tranzacție;
- **Statistici pe tipuri de tranzacții**: comparația între pozițiile long și short, fiecare cu număr, profit și winrate;
- **Distribuții detaliate**: profitul agregat după instrument tranzacționat (ex. DE30EUR, EURUSD), după ziua săptămânii și după durata tranzacției (intervale predefinite între 0–1h și peste 24h).

Prin această funcție, platforma oferă utilizatorului o analiză completă a performanțelor, nu doar prin indicatori globali, ci și prin perspective granulare asupra modului în care strategiile sau obiceiurile de tranzacționare se reflectă în rezultate. Această componentă constituie fundamentul modulului de Statistică și Dashboard, fiind utilizată pentru afișarea rapoartelor.

- **Funcția simulate_trades**

Această funcție [vezi simulate_trades] reprezintă nucleul procesului de backtesting. Ea primește ca parametri o listă de semnale generate de o strategie (signals), datele istorice sub formă de lumânări (candles), intervalul de timp analizat (timeframe) și simbolul instrumentului (symbol). Rezultatul final este un obiect care conține atât lista tranzacțiilor executate, cât și un set de metrici agregate. Logica de simulare urmează pașii esențiali ai tranzacționării reale:

- **Intrarea în poziție** - atunci când apare un semnal valid, se deschide o poziție BUY sau SELL la prețul specificat. Funcția reține detaliile poziției: prețul de intrare, nivelurile de stop-loss și take-profit;
- **Monitorizarea poziției** - pe parcursul derulării candle-urilor, se verifică în permanență dacă prețul atinge nivelul de SL sau TP. În caz afirmativ, poziția este închisă, profitul este calculat, iar tranzacția este înregistrată în jurnal;
- **Schimbarea semnalului** - dacă apare un semnal contrar poziției curente, aceasta este închisă la prețul curent, se calculează profitul/pierderea, iar apoi se deschide o nouă poziție conform

noului semnal;

- **Închiderea forțată la final** - dacă la sfârșitul datelor există încă o poziție deschisă, aceasta este închisă automat la ultima lumânare disponibilă, pentru a încheia corect simularea.

Funcția construiește astfel lista completă a tranzacțiilor generate de strategie, fiecare conținând informații despre punctele de intrare și ieșire, direcție, prețuri și profit. În plus, sunt actualizate metricile intermediare și finale, care vor fi utilizate ulterior pentru analiza performanței.

Această funcție este esențială pentru evaluarea obiectivă a strategiilor: ea transpune regulile abstracte definite în strategie într-un set concret de tranzacții, permițând utilizatorului să observe cum ar fi performat regulile respective pe date istorice.

- **Funcția get_dynamic_sl**

Funcția [vezi get_dynamic_sl] calculează un nivel dinamic de stop-loss (SL) la deschiderea unei poziții, adaptat la volatilitatea curentă și la structura recentă a prețului. Intrările sunt: prețul de intrare (entry_price), seria de lumânări (candles), indexul lumânării la care se deschide poziția (entry_index), simbolul instrumentului (symbol) și direcția tranzacției (action: BUY/SELL). Ieșirea este o distanță pozitivă (în unități de preț) care va fi aplicată sub preț (la BUY) sau peste preț (la SELL).

Pașii principali și raționamentul:

- Validare & fallback inițial: Dacă setul de date lipsește sau indexul este invalid, funcția returnează un SL implicit. Această valoare garantează existența unui prag de protecție chiar și în condiții neprevăzute;
- Se calculează Average True Range pe 14 perioade, apoi se folosește un multiplu (1.5) pentru a obține o distanță proporțională cu volatilitatea curentă. Această metodă reacționează la schimbările neașteptate ale pieței, plasând SL mai departe în perioade agitate și mai aproape în perioade calme;
- În cazul structurii de preț pe o fereastră de N candele, SL se aliniază suporturilor/rezistențelor locale, evitând scoaterea din piață de zgomotul minor sub/peste extremele recente;
- Fallback de volatilitate pură (percentila 75). Ca alternativă robustă la ATR, se estimează o distanță pe baza percentilei 75 a intervalului High-Low în fereastra de 30 de perioade. Aceasta surprinde coada superioară a distribuției mișcărilor, fără a fi prea sensibilă la outliere izolate;
- Selecția finală: Din candidații validați (ATR, structură, percentila 75) funcția alege valoarea maximă. Această regulă „max-distance wins” privilegiază conservarea capitalului, preferând un SL mai mare atunci când cel puțin una dintre metode sugerează risc ridicat.

Impact în backtesting/execuție:

- Unifică trei surse de semnal pentru SL (volatilitate medie, structură, volatilitate pură), reducând riscul de underfitting la o singură metodă;
- Asigură consistență: pentru fiecare intrare există mereu un SL (fallback determinist), ceea ce simplifică testarea strategiilor și comparabilitatea rezultatelor.

3.6. Ilustrarea funcționalității

Pentru a evidenția modul de utilizare concret al platformei, au fost realizate mai multe capturi de ecran care surprind principalele interfețe și fluxuri logice.

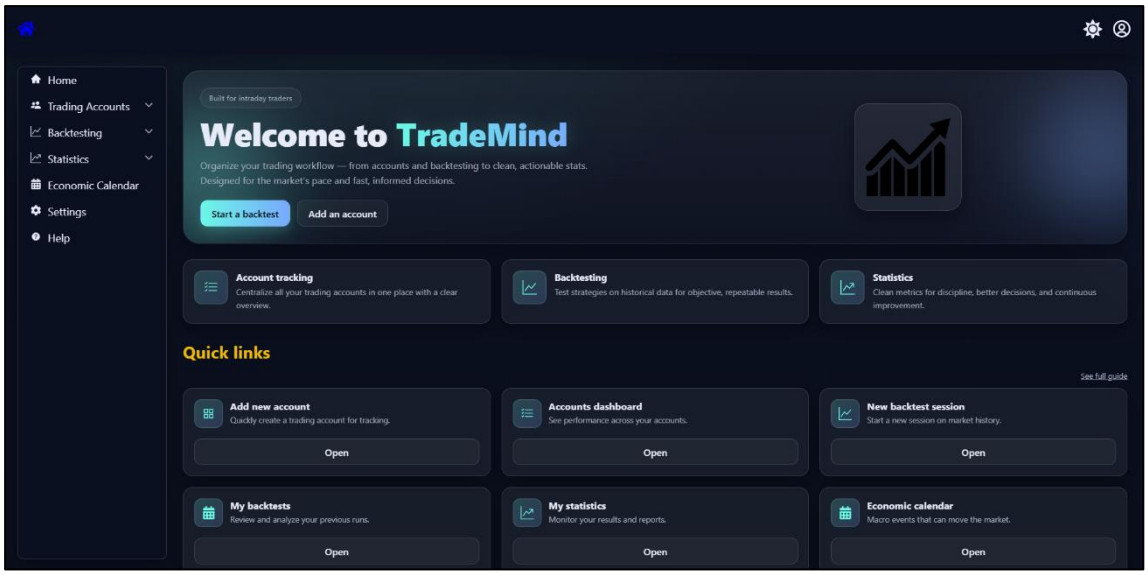


Figura 16. Pagina Home, meniul principal al platformei

Această interfață [Figura 16] reprezintă punctul de pornire al aplicației, centralizând funcționalitățile esențiale: gestionarea conturilor de tranzacționare, inițierea de sesiuni de backtesting și consultarea statisticilor derivate. Utilizatorul are acces rapid la adăugarea de conturi noi, la vizualizarea rezultatelor anterioare, la lansarea unor sesiuni noi de testare, prin intermediul secțiunii „Quick links”.

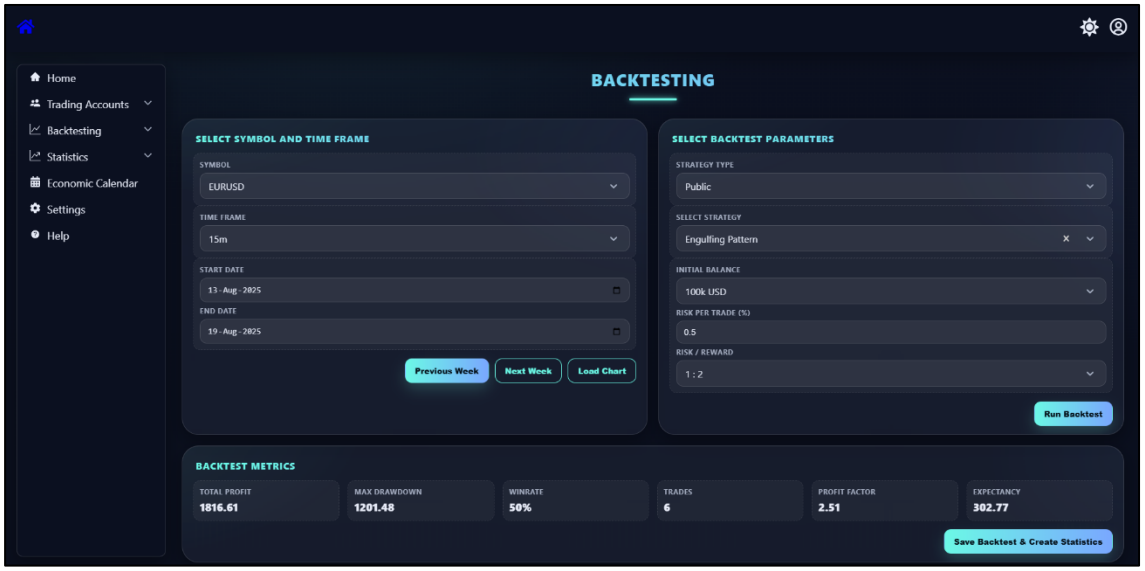


Figura 17. Pagina Backtesting cu parametrii setați pentru o sesiune

Această secțiune [Figura 17] permite selectarea și configurarea graficului (symbol, time_frame, interval de timp) și configurarea parametrilor strategiei (tip strategie, balanță inițială, risc per tranzacție, raport risc/recompensă). După rularea backtest-ului, aplicația afișează indicatori relevanți, oferind utilizatorului o imagine obiectivă asupra performanței strategiei testate.

Rezultatele backtest-ului [Figura 18] sunt redate vizual pe un grafic, pe care sunt marcate punctele de intrare și ieșire din tranzacții. Acest mod de reprezentare facilitează înțelegerea comportamentului strategiei pe intervalul analizat, evidențiind atât tranzacțiile profitabile, cât și pe cele pierzătoare, precum și contextul de piață în care au apărut.

Figura 18. Graficul, în format full screen, pentru sesiunea prezentată anterior

Capitolul 4. Testarea aplicației și rezultate experimentale

4.1. Punerea în funcțiune

Aplicația a fost pusă în funcțiune într-un mediu de dezvoltare local, utilizând containere Docker pentru fiecare componentă: frontend, backend și baza de date. Astfel, s-a realizat o izolare completă a serviciilor și s-a asigurat portabilitatea între diferite sisteme de operare. Configurația serviciilor s-a realizat cu ajutorul fișierului docker-compose.yml, care definește rețeaua internă de comunicare și stabilește dependențele dintre containere. Variabilele esențiale, precum chei secrete, date de conectare la baza de date PostgreSQL și alte date sensibile, au fost stocate în fișierul .env.

Procesul de lansare al aplicației a fost configurat printr-o singură comandă docker-compose up, serviciile au fost pornite în paralel și interconectate automat. Backend-ul (serviciul principal de API) a fost disponibil pe un port dedicat, frontend-ul a putut fi accesat direct din browser prin localhost, iar baza de date a putut fi inspectată și administrată direct din mediul de dezvoltare al backendului, PyCharm. În urma acestei configurări, aplicația a fost complet funcțională și accesibilă în browser.

4.2. Testarea funcțională

Testarea funcțională a vizat verificarea fiecărui flux major al aplicației, pentru a confirma că toate componentele răspund conform specificațiilor. Procesul a fost realizat incremental: inițial, fiecare rută din backend a fost verificată individual, folosind Swagger UI și Postman. Ulterior, după integrarea frontend-ului, s-au validat fluxurile complete end-to-end.

Swagger UI a fost utilizat pentru a examina structura cererilor și răspunsurilor, precum și pentru a valida tipurile de date transmise. Postman a permis simularea unor scenarii complexe, precum: crearea unui cont nou, autentificarea și primirea tokenului JWS, introducerea unei tranzacții manuale, lansarea unei sesiuni de backtest și obținerea rezultatelor statistice. În toate cazurile, răspunsurile serverului au respectat cerințele definite, cu coduri de status corecte.

Pentru a verifica funcționalitatea în condiții variate, au fost definite mai multe seturi de date fictive: conturi demo cu balanțe diferite, tranzacții manuale plasate pe indici și perechi valutare, precum și sesiuni de backtest pe intervale temporale scurte și lungi. Acest lucru a permis observarea

comportamentului aplicației atât în scenarii simple (câteva tranzacții), cât și în scenarii mai complexe (zeci de tranzacții și strategii rulate concomitent).

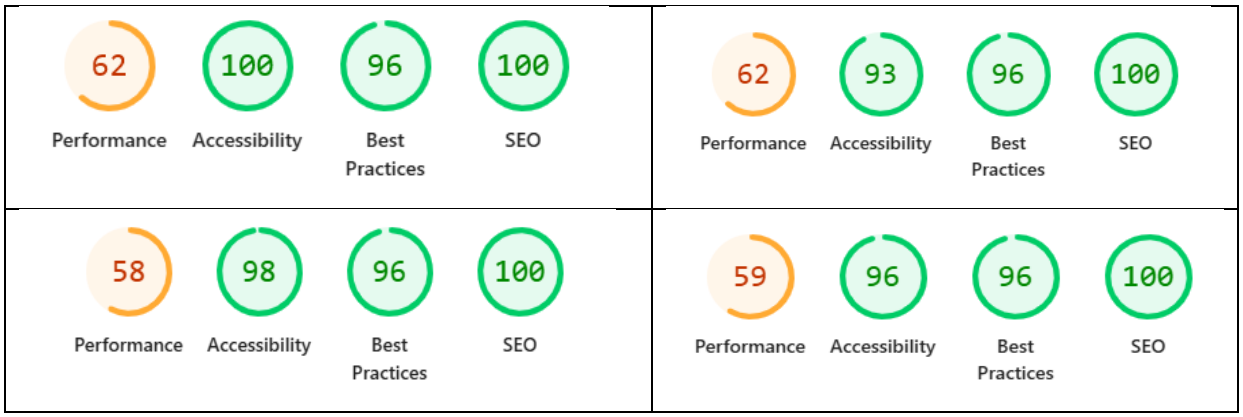
După integrarea frontend-ului, s-a testat întregul parcurs al utilizatorului: autentificare - adăugare cont - introducere tranzacții - lansare backtest - vizualizare statistici. Rezultatele afișate în interfață au fost comparate cu valorile returnate direct de API, confirmând că datele sunt prelucrate și prezentate corect.

4.3. Testarea performanței aplicației

Pentru evaluarea performanței aplicației s-a utilizat Lighthouse, un instrument integrat în Chrome DevTools, care măsoară calitatea aplicațiilor web pe patru dimensiuni: Performance, Accessibility, Best Practices și SEO. Testele au fost rulate de mai multe ori, pe același mediu local, iar rezultatele au fost consistente [vezi Tabelul 9].

- Performance – acest indicator măsoară timpul de încărcare al aplicației, rapiditatea execuției scripturilor și viteza cu care utilizatorul poate interacționa cu interfața. Scorul mai redus (58-62) se explică prin faptul că aplicația folosește componente complexe, precum grafice interactive de tip candlestick și dashboard-uri cu calcule în timp real, care au nevoie de mai mult timp pentru randare. De asemenea, încărcarea simultană a mai multor resurse (grafice, statistici și interogări API) afectează inițial timpul de răspuns;
- Accessibility - scorul ridicat (93-100) arată că interfața respectă standardele de accesibilitate: contrast bun al textului, etichete pentru elementele interactive și posibilitatea de navigare cu tastatura;
- Best Practices - valoarea constantă de 96-100 indică respectarea celor mai bune practici în dezvoltarea web: utilizarea resurselor securizate, evitarea vulnerabilităților, respectarea standardelor JavaScript și optimizarea imaginilor;
- SEO (Search Engine Optimization) - scorul maxim (100) confirmă faptul că structura aplicației permite o indexare bună de către motoarele de căutare. Pagina conține metadate corecte, titluri unice și respectă structura semantică HTML, ceea ce asigură vizibilitate ridicată în cazul publicării online.

Aplicația obține rezultate excelente la capitolele de calitate a codului, accesibilitate și optimizare SEO, confirmând maturitatea designului și conformitatea cu standardele web. Singurul aspect identificat pentru îmbunătățire este scorul de Performance, care ar putea fi optimizat prin: încărcarea lazy a componentelor de grafic, reducerea dimensiunii bundle-urilor JavaScript și utilizarea unor mecanisme de cache pe partea de frontend.



Tabelul 9. Performanțe înregistrate – Lighthouse

4.4. Validarea rezultatelor sesiunilor de backtesting

Pentru a valida corectitudinea și relevanța implementării motorului de backtesting, au fost definite două tipuri de experimente. Primul test a constat în compararea mai multor strategii rulate pe aceeași perioadă de timp, pe același instrument financiar și cu aceiași parametri de capital și risc, pentru a evidenția diferențele de performanță generate exclusiv de regulile fiecărei strategii. Al doilea test, urmărește evaluarea unei singure strategii prin modificarea variabilelor de intrare, precum perioada de timp analizată și/sau intervalul de time frame utilizat. În acest mod se pot observa diferențele de comportament și stabilitatea strategiei în condiții de piață variate.

4.4.1. Evaluarea mai multor strategii în aceleași condiții de piață

Parametri de testare

În cadrul experimentului s-a ales indicele bursier GER40 (DAX – indicele Germaniei), pe un interval de timp de 5 minute, cu perioada de testare 4-10 august 2025. Capitalul inițial setat a fost de 100.000 USD, iar riscul per tranzacție a fost configurat la 1%, cu un raport risc/recompensă de 1:2. Acești parametri au fost menținuți constanți pentru toate strategiile testate, pentru a putea compara rezultatele în mod obiectiv.

Indicatorii analizați

Rezultatele fiecărei strategii au fost sintetizate în [Tabelul 10], care conține următorii parametri:

- Profit total - suma netă a rezultatelor tranzacțiilor (pozitivă sau negativă);
- Max Drawdown - cea mai mare pierdere înregistrată raportată la un vârf anterior de capital; reflectă riscul real pe perioada testului;
- Winrate - procentul tranzacțiilor câștigătoare din total;
- Trades - numărul total de tranzacții efectuate;
- Profit Factor - raportul dintre profitul total și pierderile totale; valori >1 indică o strategie profitabilă;
- Expectancy - profitul mediu estimat pe tranzacție, exprimat în unități monetare.

Rezultatele obținute

Se poate observa că strategiile testate au oferit rezultate extrem de variate:

- Strategii profitabile precum Bollinger Bands (+5892 USD, PF=1.24) sau Inside Bar (+2737 USD, PF=1.17) au generat câștiguri pe perioada testului, având o expectanță pozitivă;
- Alte strategii, cum ar fi RSI Overbought/Oversold, a înregistrat rezultate aproape neutre, cu profituri foarte mici raportate la capitalul inițial;
- În schimb, strategii precum Hammer & Shooting Star sau Head & Shoulders au produs pierderi considerabile (până la -7208 USD), cu un profit factor subunitar și o expectanță negativă.

Aceste diferențe confirmă faptul că, deși toate strategiile au fost rulate pe același interval temporal și pe aceleași date de piață, regulile de generare a semnalelor pot conduce la performanțe diametral opuse. De exemplu, Inside Bar Strategy a avut un winrate ridicat de 52,5%, în timp ce Head & Shoulders a avut doar 21,7%. Totodată, RSI Overbought/Oversold a generat un drawdown maxim de peste 10.000 USD, ceea ce ar fi periculos într-un cont real, chiar dacă profitul final a fost ușor pozitiv. O parte din rezultatele experimentale obținute au fost ilustrate în [Figura 23], [Figura 24], [Figura 25] și [Figura 26].

Interpretarea rezultatelor

Această variație demonstrează două aspecte fundamentale:

- Backtesting-ul este indispensabil pentru validarea strategiilor, deoarece reguli aparent promițătoare pot genera pierderi semnificative în practică.
- Parametrii de test trebuie controlați pentru a permite comparații corecte între strategii. Menținerea capitalului inițial, riscului per tranzacție și a raportului risc/recompensă constante a permis ca diferențele să reflecte exclusiv calitatea regulilor de intrare și ieșire.

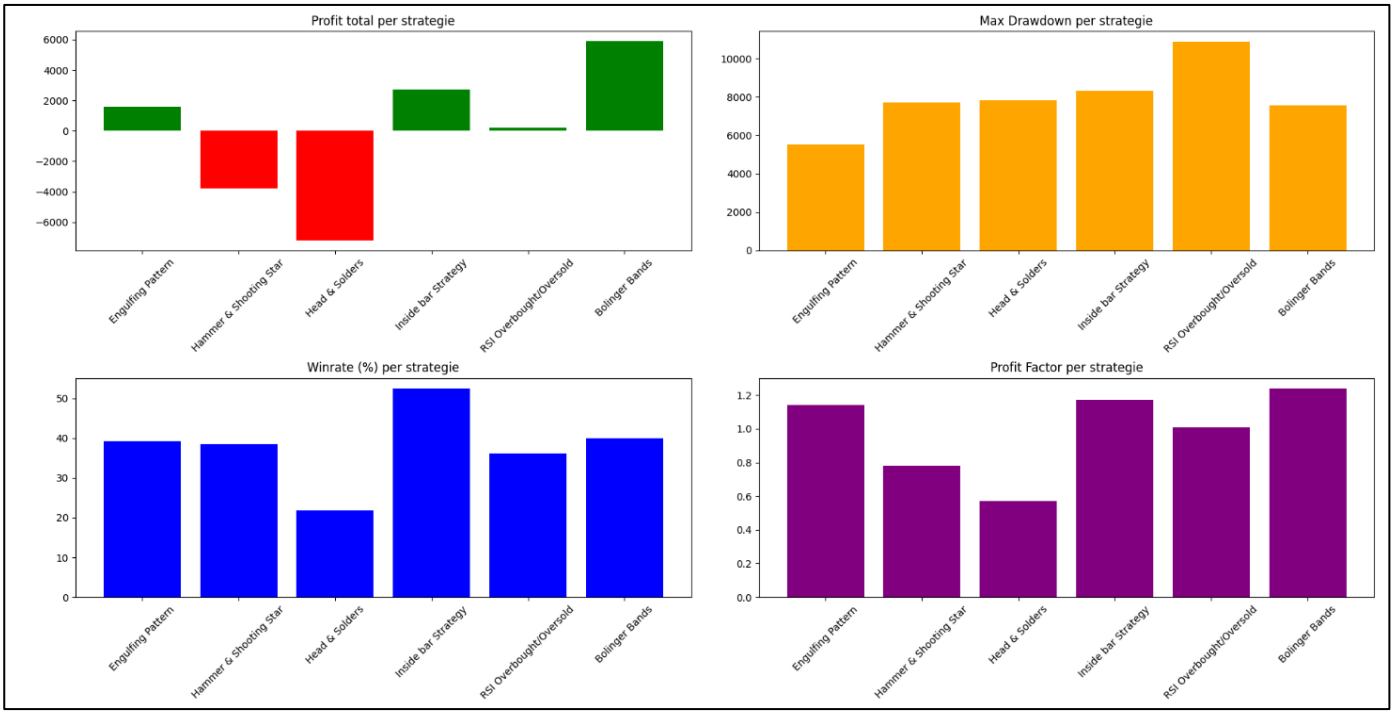


Figura 19. Analiză comparativă a performanțelor a 6 strategii publice

Figura 23. Evoluția prețului 6-7 august 2025

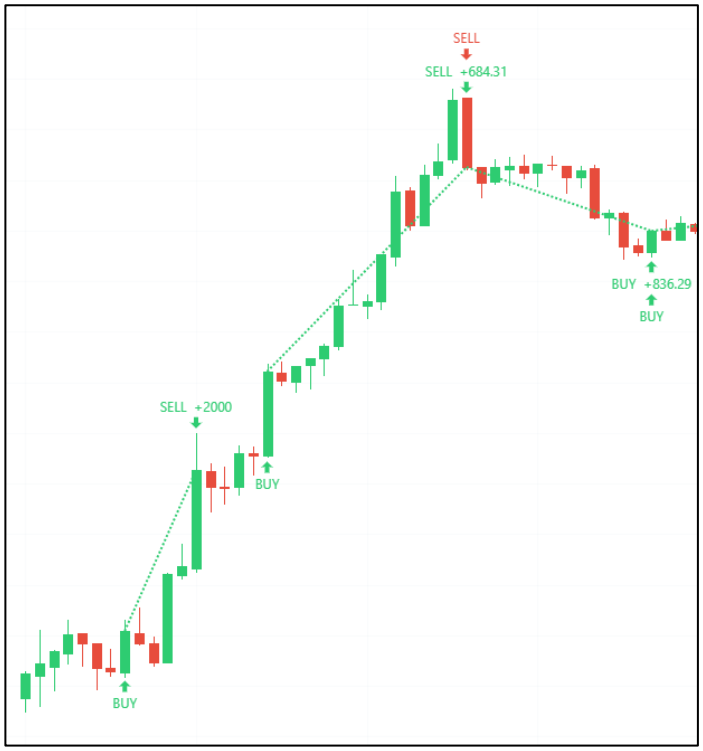


Figura 24. O parte din tranzacțiile înregistrate folosind Engulfing Pattern

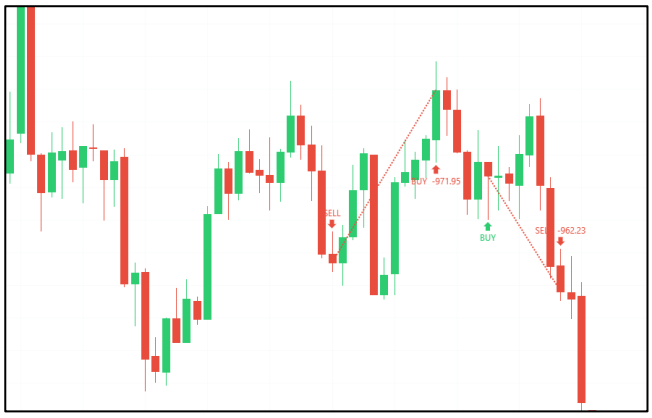


Figura 25. O parte din tranzacțiile înregistrate folosind Hammer & Shooting Star

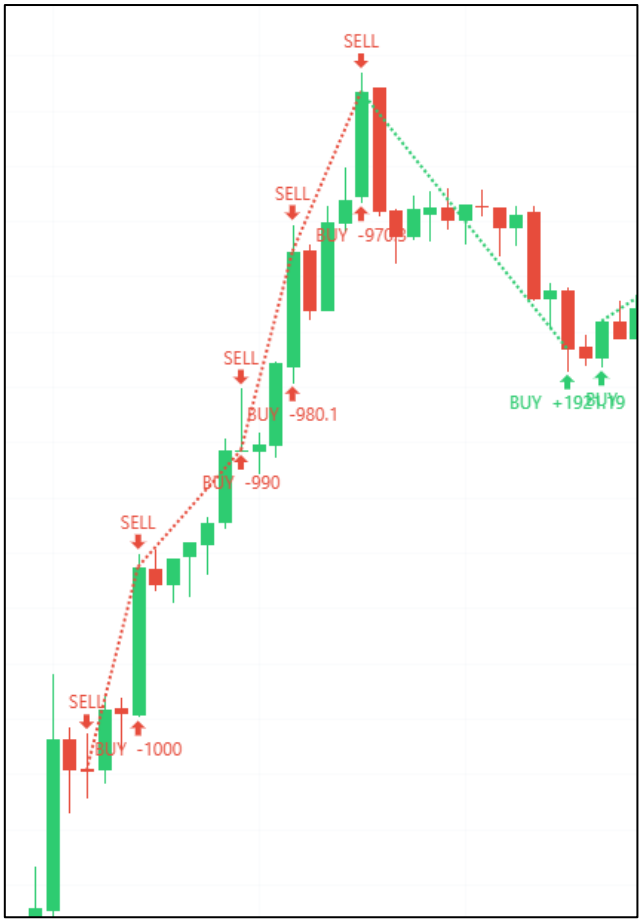


Figura 26. O parte din tranzacțiile înregistrate folosind RSI Overbought/Oversold

Strategy	Profit total	Max Drawdown	Winrate	Trades	Profit Factor	Expectancy
Engulfing Pattern	$1601.27	$5540.58	39.29%	28	1.14	57.19
Hammer & Shooting Star	-$3807.33	$7695.69	38.46%	39	0.78	-97.62
Head & Solders	-$7208.32	$7835.47	21.74%	23	0.57	-313.41
Inside bar Strategy	$2737.93	$8313.05	52.50%	40	1.17	68.45
RSI Overbought/Oversold	$194.23	$10879.51	36.11%	36	1.01	5.4
Bolinger Bands	$5892.26	$7547.92	40.00%	40	1.24	147.31

Tabelul 10. Analiză comparativă a performanțelor a 6 strategii publice

4.4.2. Evaluarea unei strategii în funcție de timeframe și perioadă

Testul A

Parametri de testare

Pentru validarea consistenței strategiei Engulfing Pattern s-au efectuat cinci sesiuni de backtesting pe perechea valutară EURUSD, timeframe 5 minute, păstrând constanți parametrii: capital inițial 100.000 USD, risc per tranzacție 1% și raport risc/recompensă 1:2. Rezultatele sintetizate în [Tabelul 11] arată o variație semnificativă a performanței de la o săptămână la alta, ceea ce confirmă caracterul dependent de context al acestei strategii.

Rezultatele obținute

În prima săptămână analizată (30 iunie - 6 iulie 2025), strategia a generat un rezultat negativ

(-1830 USD) cu un winrate modest de 25%. Numărul redus de tranzacții (8) și profit factorul subunitar (0.64) sugerează că semnalele furnizate au fost mai degrabă aleatorii, fără să se plieze pe direcția predominantă a pieței.

Situația se schimbă radical în săptămânile 7 - 13 iulie și 14 - 20 iulie 2025, când strategia a înregistrat profituri consistente (+6057 USD respectiv +3206 USD) și un winrate de peste 55%. Factorul de profit a depășit valoarea 2, ceea ce arată o strategie eficientă într-un context de piață favorabil (probabil perioade cu mișcări direcționale clare, unde pattern-urile Engulfing au avut relevanță).

În schimb, în săptămânile 21 - 27 iulie și 28 iulie - 3 august 2025, performanța s-a deteriorat semnificativ. Strategia a generat pierderi mari (-9241 USD și -7423 USD), cu drawdown-uri maxime de peste 9200 USD și profit factor între 0.18 și 0.47. Winrate-ul a scăzut dramatic (14 - 25%), ceea ce indică faptul că pattern-urile identificate au eșuat să reflecte mișcările reale ale pieței, probabil din cauza unei volatilități ridicate sau a unor condiții laterale (range-bound).

Interpretarea rezultatelor

Rezultatele obținute confirmă că strategia Engulfing Pattern pe timeframe de 5 minute are o performanță inconstantă. În anumite săptămâni poate genera profituri semnificative (profit factor >2, expectancy pozitiv), dar în altele conduce la pierderi considerabile și drawdown-uri mari. Acest lucru demonstrează că:

- strategia este sensibilă la contextul pieței, funcționând mai bine în condiții de trend clar;
- pentru utilizare reală, este necesară filtrarea semnalelor cu indicatori suplimentari (ex. volum, RSI, context de time frame mai mare);
- timeframe-ul mic amplifică atât oportunitățile, cât și riscurile, generând rezultate volatile.

Astfel, testul A arată limitele utilizării brute ale strategiei Engulfing Pattern și confirmă importanța validării multi-perioadă și multi-timeframe pentru a obține concluzii robuste. O parte din rezultatele experimentale obținute au fost ilustrate în [Figura 27] și [Figura 28].

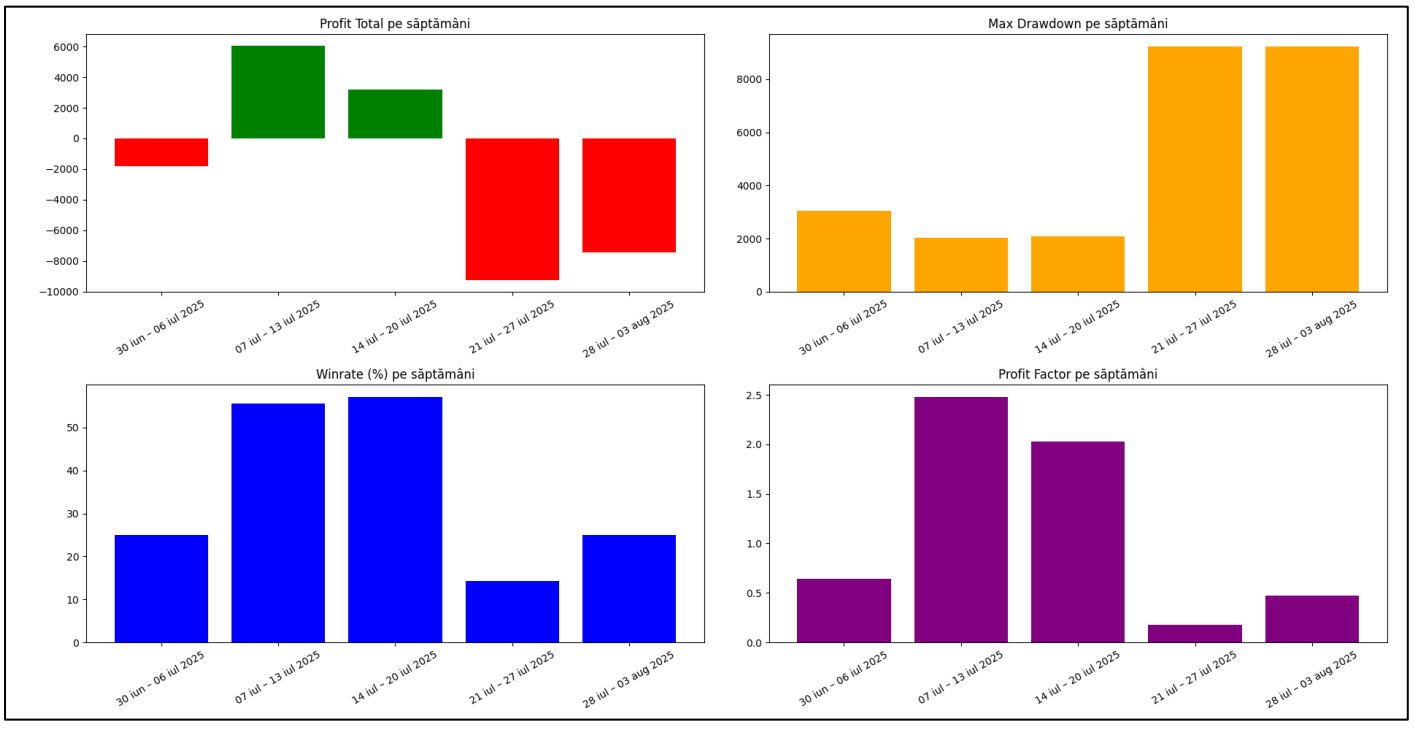


Figura 20. Analiză comparativă a performanțelor unei strategii timp de 5 săptămâni - Test A

Figura 27. Exemplu de tranzacție profitabilă din perioada 7-13 iulie 2025

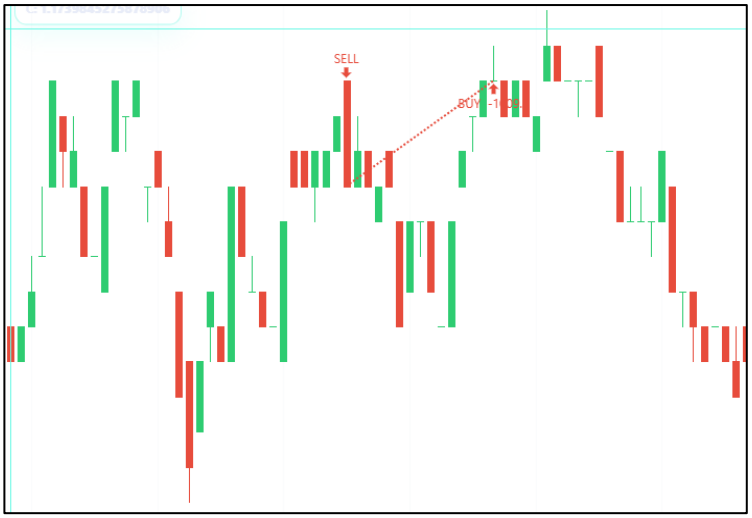


Figura 28. Exemplu de tranzacție neprofitabilă din perioada 7-13 iulie 2025

Perioada testată	Profit Total	Max Drawdown	Winrate	Trades	Profit Factor	Expectancy
30 iun – 06 iul 2025	-$1830.53	$3035.15	25.00%	8	0.64	-228.82
07 iul – 13 iul 2025	+$6057.55	$2029.80	55.56%	9	2.48	+673.06
14 iul – 20 iul 2025	+$3206.95	$2095.51	57.14%	7	2.03	+458.14
21 iul – 27 iul 2025	-$9241.83	$9241.83	14.29%	14	0.18	-660.13
28 iul – 03 aug 2025	-$7423.05	$9238.28	25.00%	20	0.47	-371.15

Tabelul 11. Analiză comparativă a performanțelor unei strategii timp de 5 săptămâni - Test A

Testul B

Parametri de testare

În cadrul celui de-al doilea set de experimente s-a testat strategia Breakout Strategy pe perechea cripto ETHUSDT, timeframe 15 minute. Parametrii au rămas constanți pentru fiecare săptămână: capital inițial 100.000 USD, risc per tranzacție 1%, raport risc/recompensă 1:2. Scopul testului a fost de a evalua stabilitatea și consistența acestei strategii atunci când este aplicată pe mai multe intervale succesive de o săptămână, în condiții variabile de piață. Rezultatele sunt sintetizate în [Tabelul 12].

Rezultatele obținute

În prima săptămână analizată (30 iunie - 6 iulie 2025), strategia a oferit rezultate excelente, cu un profit de +4873 USD, un winrate ridicat de 57,89% și un profit factor de 1.76, ceea ce indică o strategie eficientă într-un context de piață cu mișcări clare de continuare după zone de breakout.

În săptămânile următoare (7 - 13 iulie și 14 - 20 iulie 2025), rezultatele s-au menținut pozitive, dar la un nivel mai redus (+1445 USD și +387 USD). Deși winrate-ul a scăzut la 27,78% în a treia săptămână, profitul a rămas pozitiv, ceea ce arată că tranzacțiile câștigătoare au avut o dimensiune suficient de mare pentru a compensa pierderile, confirmând relevanța unui raport risc/recompensă favorabil. O parte din rezultatele experimentale obținute au fost ilustrate în [Figura 29].

În schimb, în ultimele două săptămâni (21 - 27 iulie și 28 iulie - 3 august 2025), performanța

strategiei s-a deteriorat. Rezultatele au fost negative (-154 USD și -3080 USD), winrate-ul a scăzut sub 35%, iar profit factorul a coborât spre valori aproape neutre sau chiar subunitare (0.48 în ultima săptămână). Max drawdown-ul de peste 4300 USD a arătat o vulnerabilitate crescută a strategiei în perioade cu mișcări false de breakout sau cu volatilitate ridicată, tipică pentru piața crypto.

Interpretarea rezultatelor

Testul B arată că Breakout Strategy pe ETHUSDT și timeframe de 15 minute are o performanță bună pe termen scurt, dar cu rezultate inconstante pe perioade mai lungi:

- Strategia are potențial ridicat în sesiuni cu trenduri puternice, oferind profituri semnificative și un profit factor consistent (>1.5).
- În schimb, în condiții de piață laterală sau cu mișcări bruște de respingere, semnalele devin mai puțin fiabile, ceea ce duce la scăderea winrate-ului și la creșterea drawdown-ului.
- Menținerea unui raport risc/recompensă de 1:2 permite obținerea unor rezultate pozitive chiar și cu un procentaj redus de tranzacții câștigătoare, dar numai dacă există câteva breakout-uri valide pe parcursul săptămânii.

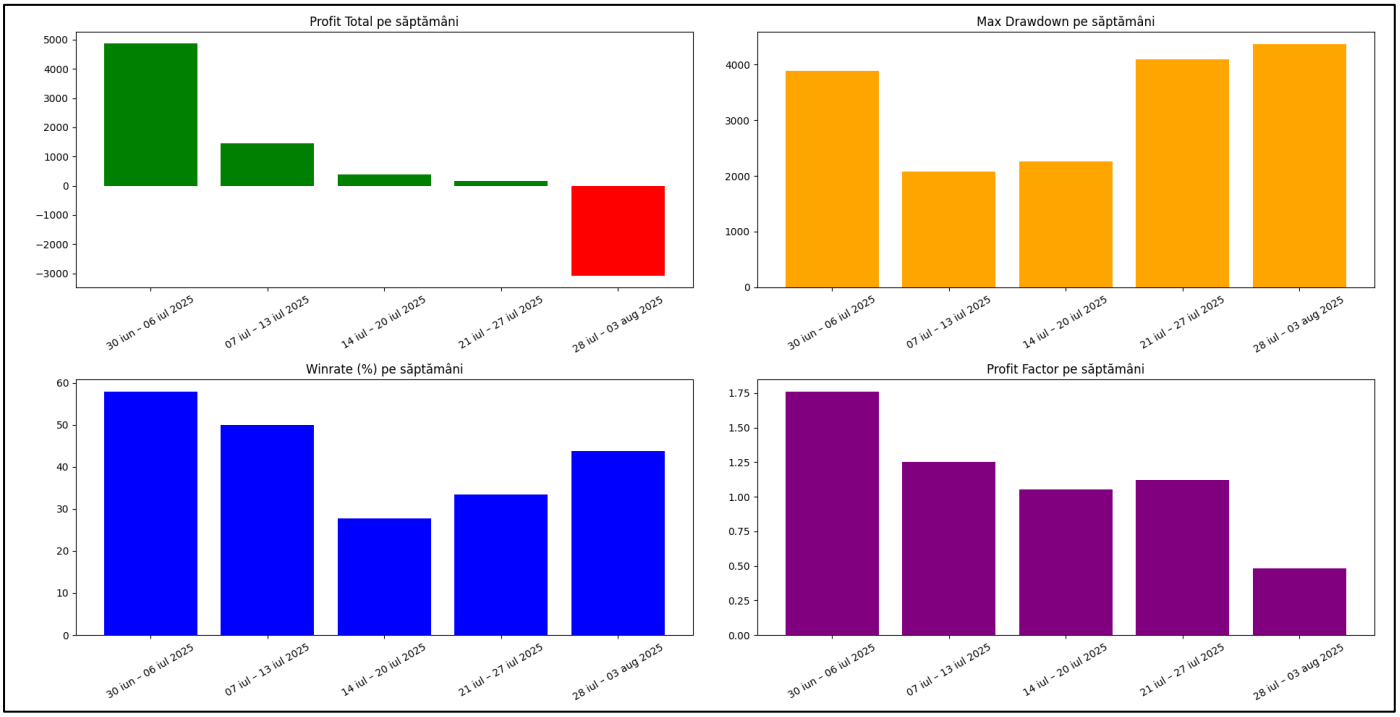


Figura 21. Analiză comparativă a performanțelor unei strategii timp de 5 săptămâni - Test B

Perioada testată	Profit Total	Max Drawdown	Winrate	Trades	Profit Factor	Expectancy
30 iun – 06 iul 2025	+$4873.95	$3884.87	57.89%	19	1.76	+256.52
07 iul – 13 iul 2025	+$1445.51	$2085.49	50.00%	14	1.25	+103.25
14 iul – 20 iul 2025	+$387.94	$2527.82	27.78%	18	1.05	+21.55
21 iul – 27 iul 2025	+$154.40	$4099.27	33.33%	21	1.12	+54.97
28 iul – 03 aug 2025	-$3080.45	$4374.17	43.75%	16	0.48	-192.53

Tabelul 12. Analiză comparativă a performanțelor unei strategii timp de 5 săptămâni - Test B

Figura 29. Exemplu de tranzacții din perioada 14-20 iulie 2025

4.5. Concluzii finale ale testării

Testele efectuate au confirmat funcționalitatea corectă a aplicației și relevanța indicatorilor calculați în procesul de backtesting. Toate componentele sistemului - de la rularea strategiilor pe date istorice, până la generarea rapoartelor statistice - au răspuns conform așteptărilor, validând atât arhitectura implementată, cât și acuratețea algoritmilor de simulare.

Analiza comparativă a celor două seturi de teste a evidențiat diferențe importante. Strategia Engulfing Pattern (EURUSD, timeframe M5) s-a dovedit extrem de volatilă, cu rezultate foarte bune în unele săptămâni, dar și cu pierderi considerabile în altele, ceea ce indică o sensibilitate ridicată la contextul pieței și o nevoie de filtre suplimentare pentru a crește consistența. În schimb, strategia Breakout (ETHUSDT, timeframe M15) a prezentat o stabilitate mai mare și o capacitate de a rămâne profitabilă pe mai multe intervale succesive, chiar dacă ultimele perioade au arătat vulnerabilitate la semnale false generate în condiții de volatilitate ridicată.

Pe partea tehnică, aplicația a demonstrat o bună performanță în ceea ce privește accesibilitatea, calitatea codului și securitatea, singura zonă unde sunt necesare optimizări suplimentare fiind performanța inițială de încărcare (scorurile Lighthouse la „Performance”). În testele de consum de resurse hardware, sistemul a funcționat fluent pentru volume moderate de date, însă scenariile cu backtest-uri multiple și seturi foarte mari de lumânări pot duce la creșterea consumului de CPU și RAM, sugerând că pentru utilizare intensivă ar fi necesară o infrastructură mai puternică

În concluzie, aplicația își atinge scopul propus: oferă utilizatorului posibilitatea de a testa și compara strategii într-un cadru controlat, cu vizibilitate asupra tuturor indicatorilor. Rezultatele obținute arată că soluția este funcțională și fiabilă, reprezentând o bază solidă pentru extinderi viitoare - atât prin integrarea de noi strategii, cât și prin optimizarea performanțelor la nivel de execuție și consum de resurse.

Capitolul 5. Concluzii

Lucrarea de față a avut ca obiectiv principal dezvoltarea unei platforme web, destinată sprijinirii activității de trading prin centralizarea conturilor, rularea de sesiuni de backtesting și analiza statistică a rezultatelor. Pornind de la obiectivele stabilite în introducere, se poate afirma că acestea au fost realizate, iar funcționalitățile implementate demonstrează viabilitatea unei soluții digitale care să sprijine procesul decizional al traderilor.

Unul dintre rezultatele semnificative îl reprezintă modulul de backtesting, care permite testarea strategiilor pe date istorice și oferă rapoarte detaliate, cu indicatori precum profitabilitate, drawdown, winrate sau expectancy. Acesta constituie o contribuție personală importantă, întrucât logica a fost implementată și optimizată pentru a reflecta cât mai fidel realitatea tranzacționării. În plus, integrarea secțiunilor de statistici și vizualizări grafice oferă o perspectivă clară și interactivă asupra performanței strategiilor, îmbunătățind experiența utilizatorului.

Comparativ cu alte soluții existente, precum platformele comerciale de backtesting sau cele integrate în terminale de tranzacționare, aplicația dezvoltată aduce avantajul unei arhitecturi modulare și a unei interfețe web moderne, ceea ce o face mai accesibilă și mai ușor de adaptat unor nevoi diverse. De asemenea, utilizarea unor tehnologii open-source (React, FastAPI, Docker) asigură portabilitate și scalabilitate. Totuși, trebuie subliniat că platformele utilizate la nivel global (ex. MetaTrader, TradingView) beneficiază de un sistem mult mai matur și de integrarea directă cu piețele financiare, ceea ce îi menține în continuare lideri de piață.

Limitările principale ale proiectului sunt legate de integrarea incompletă a unor servicii externe (precum MetaTrader, care necesită rulare locală), precum și de complexitatea optimizării strategiilor pentru scenarii reale de piață. Cu toate acestea, lucrarea demonstrează fezabilitatea unei soluții personalizate și oferă o bază solidă pentru dezvoltări ulterioare.

5.1. Direcții viitoare de dezvoltare

Direcțiile de dezvoltare ulterioară a aplicației pot include:

- Integrarea cu alte platforme de tranzacționare (ex. Interactive Brokers), pentru a transforma sistemul într-un instrument utilizabil și în alte platforme cunoscute la nivel global.
- Extinderea gamei de strategii, incluzând algoritmi mai avansați și metode bazate pe machine learning pentru o mai buna prezentare a rezultatelor.
- Adăugarea unui sistem de alerte și notificări în timp real, care să sprijine deciziile traderului direct din platformă.
- Crearea unui sistem colaborativ unde utilizatorii pot partaja strategii și rezultate, consolidând o comunitate în jurul aplicației.
- Implementarea unui model bazat pe machine learning care, pe baza time-frame-ului, a simbolului și a volatilității să sugereze un nivel de stop-loss

5.2. Lecții învățate pe parcursul dezvoltării proiectului de diplomă

Privind înapoi la tot procesul de dezvoltare a proiectului, realizez cât de multe lecții am acumulat, nu doar tehnice, ci și de organizare și de răbdare. Una dintre cele mai importante a fost legată testare. Recunosc că inițial am privit testarea ca pe o etapă de final, ceva ce trebuie bifat. Dar realitatea a fost alta: prin testări succesive, la fiecare etapă, am reușit să identific probleme mai devreme decât m-aș fi așteptat și să le corectez pe loc. Fie că era vorba de timpi de încărcare prea

mari, de un bug ascuns în logica de backtesting sau de un grafic care nu arăta cum trebuie, toate s-au rezolvat mai ușor atunci când am luat testarea ca pe o rutină și nu ca pe o obligație finală.

Poate cea mai neașteptată lecție a fost legată de adaptarea la limitări. De exemplu, integrarea cu MetaTrader a fost un exercițiu de răbdare și creativitate. A trebuit să învăț că uneori nu există o soluție perfectă și că e nevoie de compromisuri și abordări hibride - o parte rulată local, alta containerizată. La început am privit aceste obstacole ca pe niște bariere, dar acum le văd ca pe ocazii de a gândi mai flexibil și de a ieși din zona de confort.

În final, acest proiect nu a fost doar o demonstrație tehnică, ci și un proces prin care am înțeles mai bine cum se construiește, se testează și se adaptează un sistem complex în condiții reale. Iar lecția cea mai mare rămâne aceea că dezvoltarea unui proiect nu e niciodată liniară - e un drum plin de provocări, ajustări și satisfacții pe care doar practica ți le poate oferi.

Bibliografie

[1] Kari Kostiainen, “ Development of Trading Algorithm Backtest Environment”, Helsinki Metropolia University of Applied Sciences - Master’s Degree, 2016.

[2] Sarasa-Cabezuelo, “Development of a Backtesting Web Application for the Definition of Investment Strategies”, Department of Computer Science, Complutense University of Madrid (UCM), 28040 Madrid, Spain, 2023

[3] Vatsa Nagaria et al., “Backtesting and profitability analysis of algorithmic trading strategies,” IEEE ICAECC, 2023.

[4] Dimitrios Vezeris et al., “ Automated trading systems’ evaluation using d-Backtest PS method and WM ranking in financial markets”, Investment Management and Financial Innovations, 2020.

[5] Otavio Silva Pereira. Algorithmic trading strategies: Automating and back-testing the perfect order strategy. Master’s thesis, Universidade NOVA de Lisboa (Portugal), 2021

[6] V. Verma, K. Jangra, J. Chawla and H. Singh, "Design and Development of Algorithmic Trading in Stock Market Using Artificial Intelligence," 2024 4th International Conference on Technological Advancements in Computational Sciences *(*ICTACS*)*, Tashkent, Uzbekistan, 2024

[7] Marton, Bence and Cakir, Hasan, Usage of the Hurst Exponent for Short Term Trading Strategies (December 1, 2022)

[8] Pan, T., Yu, J., Zheng, L., Li, Y. (2024). Application and Modeling of LLM in Quantitative Trading Using Deep Learning Strategies. In: Samsonovich, A.V., Liu, T. (eds) Biologically Inspired Cognitive Architectures 2023.

[9] J. Wang, Q. Yang, Z. Jin, W. Chen, T. Pan and J. Shen, "Research on quantitative trading strategy based on LSTM," *2020* Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 2020

[10] https://ftmo.com/en/ftmo-academy/

[11] https://www.babypips.com/

[12] https://admiralmarkets.com/education#Beginner

Anexe

Anexa 1. Diagrama Arhitecturală – Backend

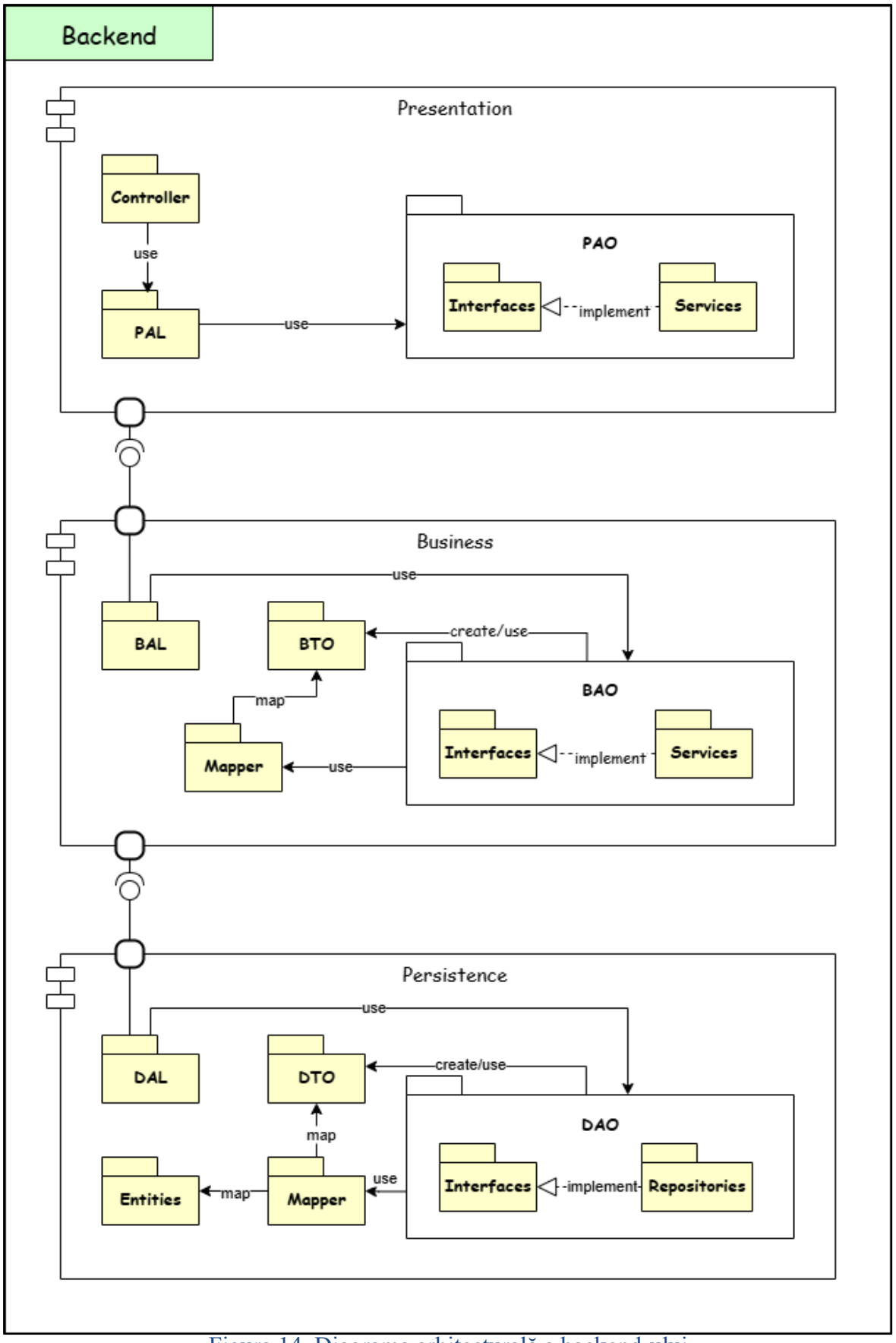


Figura 14. Diagrama arhitecturală a backend-ului

Anexa 2. Structura Bazei de Date

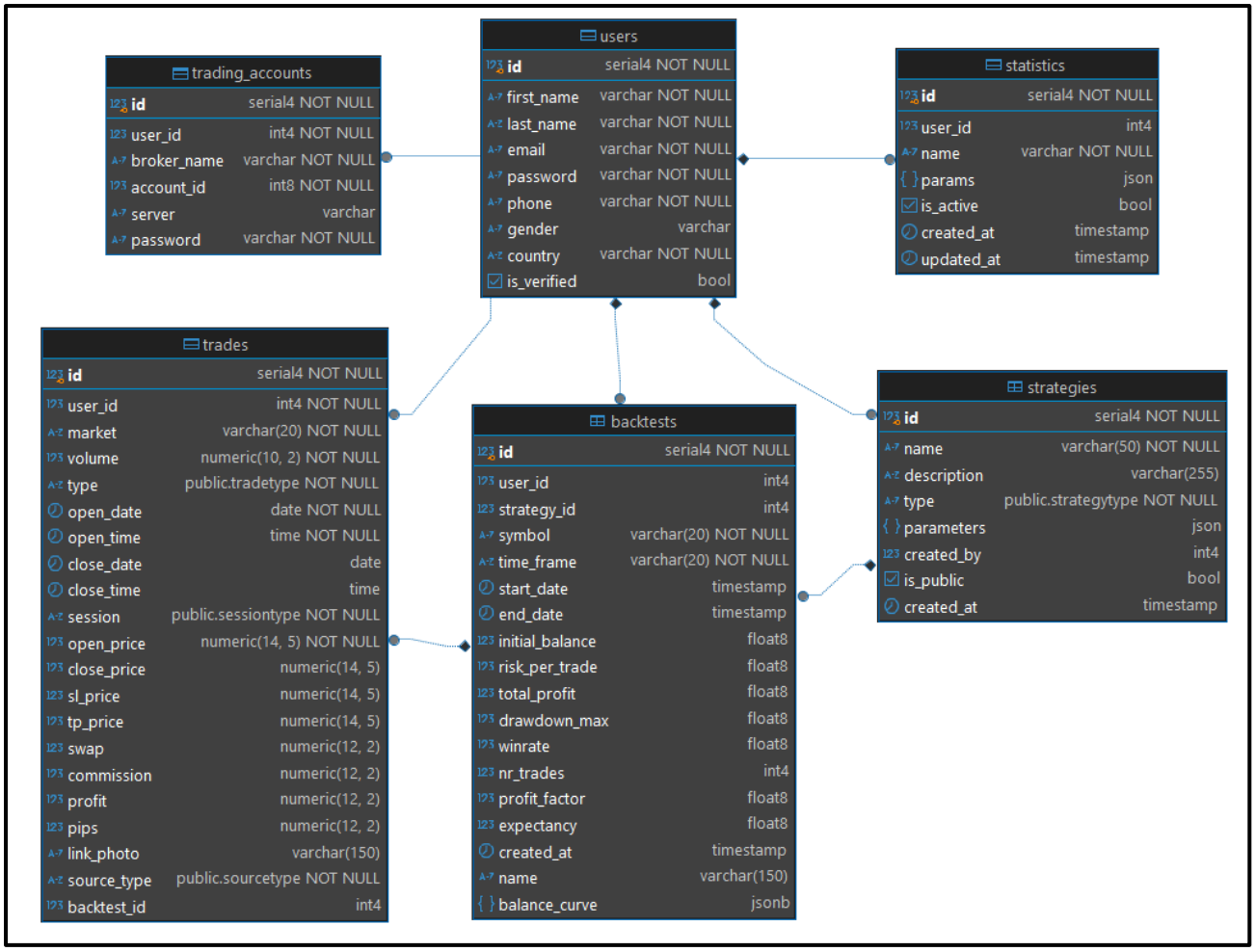


Figura 22. Structura bazei de date

Anexa 3. Codul funcțiilor prezentate

```
def _calculate_metrics(self, trades: List[TradeBTO], name: str, filters:
                                                              dict) -> dict:
    total = len(trades)
    if total == 0:
        return {
            "statistic_name": name,
            "filters_applied": filters,
            "metrics": {
                "total_trades": 0,          "winrate": 0,
                "lossrate": 0,              "break_even": 0,
                "avg_rr": None,             "avg_profit": None,
                "avg_loss": None,           "total_result": 0,
                "max_profit": None,         "max_loss": None,
                "balance_curve": [],        "long_stats": {},
                "short_stats": {},          "by_instrument": [],
                "by_day": [],               "by_duration": []
            }
        }

    wins = [t for t in trades if t.profit and t.profit > 0]
    losses = [t for t in trades if t.profit and t.profit < 0]
    be = [t for t in trades if t.profit == 0]

    rr_list = []
    for t in trades:
        if t.tp_price and t.sl_price and t.open_price:
            rr = abs(t.tp_price - t.open_price) /
                      abs(t.open_price - t.sl_price)
                      if t.sl_price != t.open_price  else None
            if rr:
                rr_list.append(rr)

    profits = [t.profit for t in trades if t.profit is not None]

    balance_curve = []
    cumulative_balance = 0
    for idx, t in enumerate(trades):
        profit = t.profit if t.profit is not None else 0
        cumulative_balance += profit
        balance_curve.append({
            "trade": idx + 1,
            "balance": round(cumulative_balance, 2)
        })

    long_trades = [t for t in trades if t.type == TradeType.BUY]
    short_trades = [t for t in trades if t.type == TradeType.SELL]

    def stat_info(trades_subset):
        if not trades_subset:
```

```
            return {"count": 0, "profit": 0.0, "winrate": 0}
        total_profit = sum(t.profit for t in trades_subset
                           if t.profit is not None)
        wins = [t for t in trades_subset if t.profit and t.profit > 0]
        return {
            "count": len(trades_subset),
            "profit": round(total_profit, 2),
            "winrate": round(len(wins) / len(trades_subset) * 100, 2)
        }

    instrument_map = defaultdict(list)
    for t in trades:
        instrument_map[t.market].append(t)

    by_instrument = []
    for market, market_trades in instrument_map.items():
        profit = sum(t.profit for t in market_trades
                     if t.profit is not None)
        by_instrument.append({
            "instrument": market,
            "trades": len(market_trades),
            "profit": round(profit, 2)
        })

    day_map = defaultdict(list)
    for t in trades:
        if t.open_date:
            dt = datetime.combine(t.open_date, datetime.min.time())
            day = dt.strftime('%A')
            day_map[day].append(t)

    by_day = []
    for day, day_trades in day_map.items():
        profit = sum(t.profit for t in day_trades
                     if t.profit is not None)
        by_day.append({
            "day": day,
            "trades": len(day_trades),
            "profit": round(profit, 2)
        })

    duration_bins = {'0-1h':[], '1-4h':[],
                     '4-12h':[], '12-24h':[],
                      '>24h':[]}

    for t in trades:
        if t.open_date and t.open_time and t.close_date and t.close_time:
            open_dt = datetime.combine(t.open_date, t.open_time)
            close_dt = datetime.combine(t.close_date, t.close_time)
            duration = (close_dt - open_dt).total_seconds() / 3600
```

```
            if duration <= 1:
                duration_bins['0-1h'].append(t)
            elif duration <= 4:
                duration_bins['1-4h'].append(t)
            elif duration <= 12:
                duration_bins['4-12h'].append(t)
            elif duration <= 24:
                duration_bins['12-24h'].append(t)
            else:
                duration_bins['>24h'].append(t)

    by_duration = []
    for label, group in duration_bins.items():
        profit = sum(t.profit for t in group if t.profit is not None)
        by_duration.append({
            "range": label,
            "trades": len(group),
            "profit": round(profit, 2)
        })

    return {
        "statistic_name": name,
        "filters_applied": filters,
        "metrics": {
            "total_trades": total,
            "winrate": round(len(wins) / total * 100, 2),
            "lossrate": round(len(losses) / total * 100, 2),
            "break_even": len(be),
            "avg_rr": round(sum(rr_list) / len(rr_list), 2)
                      if rr_list else None,
            "avg_profit": round(sum(t.profit for t in wins) /
                          len(wins), 2) if wins else None,
            "avg_loss": round(sum(t.profit for t in losses) /
                          len(losses), 2) if losses else None,
            "total_result": round(sum(profits), 2),
            "max_profit": max(profits, default=None),
            "max_loss": min(profits, default=None),
            "balance_curve": balance_curve,
            "long_stats": stat_info(long_trades),
            "short_stats": stat_info(short_trades),
            "by_instrument": by_instrument,
            "by_day": by_day,
            "by_duration": by_duration
        }
    }
```

Listing 1. Codul funcției *_calculate_metrics*

```
def simulate_trades(self, signals: List[Dict], candles: List[Dict],
                                    timeframe: str, symbol: str) -> Dict:
    executed_trades = []
    self.timeframe = timeframe

    position = None
    entry_price, entry_index = None, None
    sl_price, tp_price = None, None

    signal_pointer = 0
    current_signal = signals[signal_pointer] if signals else None

    for i in range(len(candles)):
        candle_time = candles[i]["time"]

        if position is not None:
            hit_tp, hit_sl = self._check_exit_conditions(candles,
                            entry_index, i, sl_price, tp_price, position)
            if hit_tp or hit_sl:
                exit_price = candles[i]["close"]
                exit_action = "SELL" if position == "BUY" else "BUY"
                profit = self._calculate_profit(exit_price, entry_price,
                         sl_price, tp_price, hit_tp, hit_sl, position)

                executed_trades.append(
                    self._record_trade(
                        candles, entry_index, i,
                        entry_action=position,
                        exit_action=exit_action,
                        entry_price=entry_price,
                        exit_price=exit_price,
                        sl_price=sl_price,
                        tp_price=tp_price,
                        profit=profit,
                    )
                )

                self._update_metrics(profit)
                position = entry_price = entry_index = sl_price =
                           tp_price = None

        if current_signal and candle_time >= current_signal["timestamp"]:
            action = current_signal["action"]
            price = current_signal["price"]

            if position is None:
                position, entry_price, entry_index, sl_price, tp_price =
                (self._enter_position(action, price, i, candles, symbol))

            elif position != action:
                exit_price = price
                hit_tp, hit_sl = self._check_exit_conditions(candles,
                            entry_index, i, sl_price, tp_price, position)
                profit = self._calculate_profit(exit_price, entry_price,
                            sl_price, tp_price, hit_tp, hit_sl, position)
                exit_action = "SELL" if position == "BUY" else "BUY"
```

```
                executed_trades.append(
                    self._record_trade(
                        candles, entry_index, i,
                        entry_action=position,
                        exit_action=exit_action,
                        entry_price=entry_price,
                        exit_price=exit_price,
                        sl_price=sl_price,
                        tp_price=tp_price,
                        profit=profit,
                    )
                )

                self._update_metrics(profit)

                position, entry_price, entry_index, sl_price, tp_price =
               (self._enter_position(action, price, i, candles, symbol))

            signal_pointer += 1
            current_signal = signals[signal_pointer]
                             if signal_pointer < len(signals) else None

    if position is not None:
        exit_index = len(candles) - 1
        exit_price = candles[exit_index]["close"]
        hit_tp, hit_sl = self._check_exit_conditions(candles,
                  entry_index, exit_index, sl_price, tp_price, position)
        profit = self._calculate_profit(exit_price, entry_price,
                 sl_price, tp_price, hit_tp, hit_sl, position)
        exit_action = "SELL" if position == "BUY" else "BUY"

        executed_trades.append(
            self._record_trade(
                candles, entry_index, exit_index,
                entry_action=position,
                exit_action=exit_action,
                entry_price=entry_price,
                exit_price=exit_price,
                sl_price=sl_price,
                tp_price=tp_price,
                profit=profit,
            )
        )
        self._update_metrics(profit)

    metrics = self._calculate_final_metrics()

    return {
        "metrics": metrics,
        "executed_trades": executed_trades,
    }
```

Listing 2. Codul funcției simulate_trades

```
def get_dynamic_sl(
    entry_price: float,
    candles: List[Dict],
    entry_index: int,
    symbol: str,
    action: str,
) -> float:

    if not candles or entry_index <= 0 or entry_index >= len(candles):
        return 20 * _pip_size(symbol)

    df = pd.DataFrame(candles, dtype=float)
    for col in ("high", "low", "close", "open"):
        if col not in df.columns:
            raise ValueError(f"Missing column: {col}")

    # 1) ATR(14) * 1.5
    dist_atr = None
    try:
        atr_series = AverageTrueRange(
            high=df["high"], low=df["low"], close=df["close"], window=14
        ).average_true_range()
        atr_val = float(atr_series.iloc[entry_index])
        if math.isfinite(atr_val) and atr_val > 0:
            dist_atr = atr_val * 1.5
    except Exception:
        pass

    # 2) Price structure (swing low/high for last N candles)
    lookback = 10
    left = max(0, entry_index - lookback)
    lows = df["low"].iloc[left:entry_index]
    highs = df["high"].iloc[left:entry_index]
    dist_structure = None
    if len(lows) and len(highs):
        is_buy = (action or "").upper() == "BUY"
        if is_buy:
            dist_structure = max(0.0, entry_price - float(lows.min()))
        else:
            dist_structure = max(0.0, float(highs.max()) - entry_price)

    # 3) Fallback: percentila 75 a True Range
    dist_vol = None
    try:
        tr = (df["high"] - df["low"]).astype(float)
        W = 30
        tr_win = tr.iloc[max(0, entry_index - W + 1): entry_index + 1]
        if len(tr_win):
            p75 = float(np.nanpercentile(tr_win, 75))
            if math.isfinite(p75) and p75 > 0:
```

```
            dist_vol = p75 * 1.2
    except Exception:
        pass

    candidates = [d for d in (dist_atr, dist_structure, dist_vol)
                  if d is not None and d > 0]
    if candidates:
        return float(max(candidates))

    return float(20 * _pip_size(symbol))
```

Listing 3. Codul funcției get_dynamic_sl