Universitatea Tehnică a Moldovei

IDENTIFICAREA FRAUDELOR TRANZACȚIILOR BANCARE

IDENTIFICATION OF BANKING TRANSACTIONS FRAUD

Student: **gr. TI-194,**
Gumaniuc Alexandru

Coordonator: **Răducanu Octavian**
asistent universitar

Chişinău, 2023

MINISTERUL EDUCAŢIEI ŞI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Ingineria Software și Automatică

Admis la susţinere
Şef departament:
FIODOROV Ion dr., conf.univ.

„___" _____ 2023

IDENTIFICAREA FRAUDELOR TRANZACȚIILOR BANCARE

Proiect de licenţă

Student: _____ **Gumaniuc Alexandru, TI-194**

Coordonator: _____ **Răducanu Octavian, lect. univ.**

Consultant: _____ **Cojocaru Svetlana, asist.univ.**

Chişinău, 2023

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

Programul de studii Tehnologia informaţiei

Aprob
Șef departament:
Fiodorov Ion, dr., conf.univ.

_____

„ *06” octombrie 2022*

CAIET DE SARCINI
pentru proiectul de licenţă al studentului

Gumaniuc Alexandru
(numele şi prenumele studentului)

1. Tema proiectului de licenţă *Identificarea fraudelor tranzacţiilor bancare*

confirmată prin hotărârea Consiliului facultăţii nr. *2* **din** „ *06*” *octombrie 2022*

2. Termenul limită de prezentare a proiectului de licență *20.05.2023*

3. Date iniţiale pentru elaborarea proiectului de licență *Sarcina pentru elaborarea proiectului de diplomă.*

4. Conţinutul memoriului explicativ

Introducere
1 Analiza domeniului de studiu
2 Modelarea şi proiectarea sistemului
3 Realizarea sistemului
4 Documentarea produsului realizat
5 Estimarea costurilor proiectului
Concluzii

5. Conţinutul părţii grafice a proiectului de licență

Figura 2.3 – Diagrama use-case a detectarii unei fraude

6. Lista consultanţilor:

Consultant	Capitol	Confirmarea realizării activităţii	
		Semnătura consultantului (data)	Semnătura studentului
Cojocaru Svetlana	*Standarde tehnologice, Controlul calităţii, Estimarea costului proiectului*		

7. Data înmânării caietului de sarcini *02.09.2022*

Coordonator *Raducanu Octavian* _____

semnătura

Sarcina a fost luată pentru a fi executată de către studentul *Gumaniuc Alexandru*

02.09.2022

semnătura, data

PLAN CALENDARISTIC

Nr. crt.	Denumirea etapelor de proiectare	Termenul de realizare a etapelor	Nota
1	*Elaborarea sarcinii, primirea datelor pentru sarcină*	*02.09.22– 30.09.22*	*10%*
2	*Analiza domeniului de studiu*	*06.10.21– 30.11.21*	*15%*
3	*Proiectarea sistemului*	*01.12.21 – 25.12.21*	*15%*
4	*Realizarea sistemului*	*10.01.22 – 05.03.22*	*35%*
5	*Descrierea sistemului*	*06.03.22– 01.04.22*	*10%*
6	*Estimarea costurilor sistemului*	*02.04.22– 15.04.22*	*15%*
7	*Finisarea proiectului*	*16.04.22– 14.05.22*	*5%*

Student *Gumaniuc Alexandru ( )*

Coordonator de proiect de licență *Raducanu Octavian ( )*

REZUMAT

Lucrarea este compusa din cinci capitole care descriu anumite particularitati ale proiectului.

Sistemele similare cu proiectul realizat redau alte proiecte asemanatoare cu avantajele și dezavantajele fiecaruia. Scopul ,obiectivele și cerințele sistemului reprezinta ceea ce ne dorim prin realizarea unei aplicații sau al unui sistem. În ansamblu, scopul, obiectivele și cerințele sistemului reprezintă trei concepte-cheie pentru a asigura că dezvoltarea și implementarea unui sistem sunt bine direcționate, bine definite și îndeplinesc nevoile și așteptările utilizatorilor și a celor implicați în proiect.

Modelarea și proiectarea sistemului informatic sunt două activități strâns legate care sunt critice în dezvoltarea unui sistem informatic eficient și fiabil. Modelarea sistemului informatic implică crearea unor reprezentări abstracte ale sistemului informatic, în care se definesc elementele sale componente, relațiile dintre acestea și modul în care acestea interacționează. Scopul modelării este de a înțelege mai bine sistemul și de a ilustra modul în care funcționează acesta. Aceste modele pot fi utilizate pentru a identifica probleme potențiale, pentru a evalua impactul schimbărilor asupra sistemului și pentru a comunica cu ceilalți membri ai echipei de dezvoltare.

Descrierea comportamentală a sistemului se referă la o reprezentare a modului în care sistemul interacționează cu mediul său, utilizatorii și alte sisteme. Imaginea generală asupra sistemului se referă la o vedere de ansamblu a componentelor, funcționalității și interacțiunilor unui sistem. Modelarea vizuală a fluxurilor este o tehnică utilizată pentru a reprezenta grafic fluxurile de date și acțiuni în cadrul unui sistem sau proces. Stările de tranzacție a sistemului se referă la diferitele stări prin care trece un sistem în timpul procesării unei tranzacții. Fluxurile de mesaje și legăturile dintre componentele sistemului sunt esențiale în proiectarea și implementarea sistemelor informatice. Acestea permit comunicarea între componentele sistemului și asigură transferul de informații între acestea.

Descrierea structurală a sistemului se referă la modul în care componentele sistemului sunt organizate și conectate între ele. Aceasta include reprezentarea arhitecturii sistemului, precum și descrierea componentelor individuale și a relațiilor dintre acestea.

În capitolul de realizare a sistemului, se descriu pașii concreți necesari pentru construirea și implementarea sistemului informatice. Aceste pași pot include proiectarea detaliată a componentelor sistemului, dezvoltarea software-ului sau hardware-ului, testarea și implementarea sistemului. În faza de dezvoltare, se va construi sistemul, utilizând tehnologiile și instrumentele specificate în faza de proiectare. Dezvoltatorii vor scrie codul și vor integra componentele pentru a crea sistemul final. În general, capitolul de realizare a sistemului va conține detalii despre fiecare fază a dezvoltării sistemului și va include informații despre tehnologiile și metodele utilizate. Acest capitol este important pentru a asigura o implementare corectă și eficientă a sistemului informatic.

ABSTRACT

The work is composed of five chapters that describe certain particularities of the project. Similar systems to the project depict other similar projects with their advantages and disadvantages.

The purpose, objectives, and requirements of the system represent what we aim to achieve through the development of an application or system. Overall, the purpose, objectives, and requirements of the system are three key concepts to ensure that the development and implementation of a system are well-directed, well-defined, and meet the needs and expectations of users and those involved in the project.

Modeling and designing the information system are two closely related activities that are critical in developing an efficient and reliable information system. Modeling the information system involves creating abstract representations of the information system, defining its component elements, the relationships between them, and how they interact.

The purpose of modeling is to better understand the system and illustrate how it works. These models can be used to identify potential problems, evaluate the impact of changes on the system, and communicate with other members of the development team. The behavioral description of the system refers to a representation of how the system interacts with its environment, users, and other systems. The general picture of the system refers to an overview of the components, functionality, and interactions of a system.

Visual modeling of flows is a technique used to graphically represent data and action flows within a system or process. The transaction states of the system refer to the different states that a system goes through during the processing of a transaction.

Message flows and links between system components are essential in the design and implementation of information systems. They allow communication between system components and ensure the transfer of information between them.

The structural description of the system refers to how the system components are organized and connected to each other. This includes the representation of the system architecture, as well as the description of individual components and the relationships between them.

In the chapter on system implementation, concrete steps necessary for building and implementing the information system are described. These steps may include detailed design of system components, software or hardware development, testing, and system implementation. In the development phase, the system will be built using the technologies and tools specified in the design phase. Developers will write code and integrate components to create the final system. In general, the chapter on system implementation will contain details about each phase of system development and include information about the technologies and methods used. This chapter is important to ensure a correct and efficient implementation of the information system.

CUPRINS

INTRODUCERE

Detectarea fraudei în tranzacții este un subiect important astăzi din cauza creșterii semnificative a infracțiunilor financiare, mai ales în era digitală. Iată câteva argumente care susțin această afirmație:

- Costul fraudei: Tranzacțiile frauduloase pot duce la pierderi financiare semnificative pentru persoanele fizice, afaceri și instituții financiare. Potrivit unui raport al Asociației Examinatorilor de Fraudă Certificați, pierderea mediană din cauza fraudei ocupaționale a fost de 150.000 de dolari în 2020. Pentru companii, costul fraudei poate fi chiar mai mare, cu unele estimări sugestive că afacerile pierd până la 5% din venitul lor din cauza fraudei în fiecare an. Sistemele de detectare a fraudelor pot ajuta la prevenirea unor astfel de pierderi identificând și oprind tranzacțiile frauduloase înainte de a cauza pagube;

- Creșterea utilizării tranzacțiilor digitale: Pe măsură ce tot mai multe tranzacții financiare sunt efectuate online, riscul de fraudă a crescut;

- Încrederea clienților: Tranzacțiile frauduloase pot eroda încrederea clienților în instituțiile financiare, mai ales dacă instituția nu poate detecta și preveni frauda. Implementarea unor sisteme eficiente de detectare a fraudelor poate ajuta la asigurarea faptului că clienții se simt în siguranță atunci când efectuează tranzacții cu o instituție financiară, ceea ce poate ajuta la construirea încrederii și consolidarea relațiilor cu clienții.

Scopul cercetării detectării fraudelor în tranzacții este de a îmbunătăți eficiența și precizia sistemelor existente de detectare a fraudelor, astfel încât să se poată identifica mai rapid și mai precis tranzacțiile frauduloase. În plus, cercetarea se concentrează pe dezvoltarea de noi tehnologii și metode de detectare a fraudelor care să permită o mai mare acuratețe și eficiență în detectarea fraudei în timp real.

Obiectivele specifice ale cercetării detectării fraudelor în tranzacții includ:

- Identificarea celor mai eficiente tehnologii și metode de detectare a fraudelor utilizate în prezent și dezvoltarea de noi metode și tehnologii care să poată îmbunătăți detectarea fraudelor;

- Studiul comportamentului fraudulos și a tiparelor de tranzacții frauduloase, pentru a putea dezvolta algoritmi și modele de detectare a fraudelor care să fie mai precise și mai eficiente.

- Dezvoltarea unor sisteme de detectare a fraudelor care să poată fi personalizate și adaptate la nevoile specifice ale diferitelor instituții financiare și de plăți.

Obiectul cercetării detectării fraudelor în tranzacții este fraudarea tranzacțiilor financiare și de plată, precum și metodele și tehnologiile utilizate pentru a detecta aceste fraude. Această cercetare se concentrează pe identificarea și analiza diferitelor tipuri de fraude în tranzacții, precum și pe dezvoltarea și îmbunătățirea sistemelor de detectare a fraudelor utilizate de instituțiile financiare și de plăți.

1 ANALIZA DOMENIULUI DE STUDIU

Proiectul dat va fi o aplicatie care va detecta fraudele pe baza unor tranzactii care vor veni dintr-o sursa. Există o lipsă de seturi de date publice disponibile privind serviciile financiare și în special în domeniul emergent al tranzacțiilor cu bani mobil. Seturile de date financiare sunt importante pentru mulți cercetători și în special pentru noi care efectuăm cercetări în domeniul detectării fraudelor. O parte a problemei este natura intrinsec privată a tranzacțiilor financiare, care duce la lipsa unor seturi de date disponibile public.

Tranzactiile vor fi simulate de PaySim care pe baza unui eșantion de tranzacții reale extrase dintr-o lună de jurnalele financiare dintr-un serviciu de bani mobil implementat într-o țară africană.[1]

Abordarea învățării automate (ML) a detectării fraudei a primit multă publicitate în ultimii ani și a mutat interesul industriei de la sistemele de detectare a fraudei bazate pe reguli la soluții bazate pe ML.

Un model de învățare automată este un program care poate găsi modele sau poate lua decizii dintr-un set de date nevăzut anterior. Procesul de rulare a unui algoritm de învățare automată pe un set de date (numitdate de antrenament) și de optimizare a algoritmului pentru a găsi anumite modele sau rezultate se numește antrenament de model. Detectarea fraudelor se va produce folosind machine learning care va implementea un model de învățare automată (ML) și un exemplu de set de date de tranzacții cu cardul de credit pentru a instrui modelul să recunoască modelele de fraudă.

Activitățile frauduloase din domeniul financiar pot fi detectate analizând semnale evidente și la suprafață. În mod neobișnuit, tranzacțiile mari sau cele care au loc în locații atipice merită evident o verificare suplimentară. Sistemele pur bazate pe reguli presupun folosirea de algoritmi care realizează mai multe scenarii de detectare a fraudei, scrise manual de analiștii de fraudă. Astăzi, sistemele vechi aplică aproximativ 300 de reguli diferite în medie pentru a aproba o tranzacție. De aceea sistemele bazate pe reguli rămân prea simple. Acestea necesită adăugarea/ajustarea manuală a scenariilor și greu pot detecta corelații implicite. În plus, sistemele bazate pe reguli folosesc adesea software moștenitcare cu greu poate procesa fluxurile de date în timp real care sunt critice pentru spațiul digital.

Cu toate acestea, există, de asemenea, evenimente subtile și ascunse în comportamentul utilizatorului care pot să nu fie evidente, dar semnalează totuși o posibilă fraudă. Învățarea automată permite creareade algoritmi care procesează seturi mari de date cu multe variabile și ajută la găsirea acestor corelații ascunse între comportamentul utilizatorului și probabilitatea acțiunilor frauduloase. Un alt punct forte al sistemelor de învățare automată în comparație cu cele bazate pe reguli este procesarea mai rapidă a datelor și mai puțină muncă manuală. De exemplu, algoritmii inteligenți se potrivesc bine cu analiza comportamentului pentru a ajuta la reducerea numărului de pași de verificare.[2]

1.1 IMPORTANȚA TEMEI

Frauda costă afacerile în profit și reputația mărcii. În unele cazuri, pierderile sunt ireparabile sau durează ani să se recupereze. Prin urmare, învățarea modului de detectare a detectării fraudei în conturile de plătit este crucială pentru a continua producția la timp.

În mod tradițional, companiile se bazau numai pe reguli pentru a bloca plățile frauduloase. Astăzi, regulile sunt încă o parte importantă a setului de instrumente antifraudă, dar în trecut, folosirea lor pe cont propriu a cauzat și unele probleme. Folosirea unei abordări numai cu reguli înseamnă că biblioteca dvs. trebuie să se extindă în continuare pe măsură ce frauda evoluează. Acest lucru face ca sistemul să fie mai lent și implică o sarcină grea de întreținere asupra echipei dvs. de analiști în fraude, solicitând un număr tot mai mare de revizuiri manuale . Escrocii lucrează mereu la modalități mai inteligente, mai rapide și mai ascunse de a comite fraude online. Astăzi, infractorii folosesc metode sofisticate pentru a fura date îmbunătățite ale clienților și a uzurpa identitatea clienților autentici, ceea ce face și mai dificil ca regulile bazate pe conturile tipice de fraudă să detecteze acest tip de comportament.

Industria serviciilor financiare și industriile care implică tranzacții financiare suferă de pierderi și daune legate de fraudă. 2016 a fost un an banner pentru escrocii financiari. Numai în SUA, numărul clienților care s-au confruntat cu fraude a atins un record de 15,4 milioane de persoane, ceea ce este cu 16% mai mare decât în 2015. Escrocii au furat aproximativ 6 miliarde de dolari de la bănci în 2016. O trecere către spațiul digital deschide noi canale pentru distribuția serviciilor financiare. . De asemenea, a creat un mediu bogat pentru fraudatori. Dacă infractorii anteriori au fost nevoiți să falsifice ID-uri de client, acum obținerea parolei contului unei persoane poate fi tot ceea ce este necesar pentru a fura bani. Fidelizarea clienților și conversiile sunt afectate în ambele medii, digital și fizic. Potrivit Javelin Strategy & Research, este nevoie de peste 40 de zile pentru a detecta frauda pentru instituțiile financiare fizice. Frauda afectează și băncile care oferă servicii de plăți online. De exemplu, 20% dintre clienți își schimbă băncile după ce s-au confruntat cu escrocherii. Deci, provocarea pentru jucătorii din industrie este să implementeze evaluareaîn timp real a reclamațiilor și să îmbunătățească acuratețea detectării fraudelor.

De exemplu, MasterCard a integrat învățarea automată și AI pentru a urmări și procesa variabile precum dimensiunea tranzacției, locația, ora, dispozitivul și datele de achiziție. Sistemul evaluează comportamentul contului în fiecare operațiune și oferă o judecată în timp real asupra faptului că o tranzacție este frauduloasă. Proiectul urmărește reducerea numărului de scăderi false ale plăților la comerciant. Studiul recent arată că scăderile false îi fac pe comercianți să piardă aproximativ 118 miliarde de dolari pe an, în timp ce pierderea clienților este de aproximativ 9 miliarde de dolari pe an. Este cel mai mare domeniu de fraudă în serviciile financiare. Prin urmare, prevenirea fraudei este un obiectiv strategic pentru industria bancară și de plăți.[2]

1.2 SISTEME SIMILARE CU PROIECTUL REALIZAT

Fiecare sistem are plusurile si minusurile sale , iar in urma cercetarii domeniului am descoperit aceste 3 sisteme de detectare a fraudelor bancare:

- Kount – Fraude la nivel de întreprindere;
- Ekata – parte a grupului Mastercard;
- Signifyd – Combaterea automată a fraudei pentru comercianți.

Ekata – parte a grupului Mastercard care est un alt software matur și bine stabilit de detectare a fraudelor. Fondată în 1997 (inițial sub numele Whitepages), Ekata oferă verificarea identității globale și prevenirea fraudei prin intermediul API-urilor, inclusiv un API de risc de tranzacție, un API Address Risk și un API Phone Intelligence. [3]

Signifyd – protejează acum 10.000 de magazine online la nivel global, ajutându-le să prevină rambursările prin trei produse cheie: Protecția veniturilor, Protecția împotriva abuzurilor și Optimizarea plăților. Produsele lor sunt orientate în mod special către volume ridicate de tranzacții și chiar automatizează protecția contra costurilor printr-un model de garanție de rambursare, în care își asumă plata taxelor de administrare a rambursării. [3]

Kount are capacități limitate de machine learning, în ciuda capacităților de creare a regulilor personalizate, funcțiile de învățare automată nu sunt la egalitate cu unele soluții mai noi de pe piață, potrivit recenziilor, la fel este prezenta o lipsa de îmbogățire a datelor. Acuratețea de prevenire a fraudei este la fel de bună ca și datele la care aveți acces. Din păcate, cu puncte de date limitate, combaterea fraudei devine mai grea.[3]

Avantajele lui Kount sunt:

- Acoperă multe industrii: varietatea de produse disponibile în ofertă înseamnă că puteți utiliza Kountpentru un gateway de plată de mare volum sau un brand de comerț electronic de nivel 1;
- Reguli automate și personalizabile: personalizați regulile pentru a automatiza prevenirea rambursăriisau pentru a reduce timpul de examinare manuală, printre altele.[3]

Signifyd este o alegere daca avem de-a face cu volume mari de tranzacții. La fel are urmatoareleavantaje:

- Prevenirea automată a rambursării: nu este chiar plug-and-play, dar Signifyd este proiectat să rulezepe pilot automat cu intrare minimă a utilizatorului.
- Acoperă alte provocări legate de frauda comerțului electronic: abuz de returnare și fraudă prietenoasă,printre altele;
- Conform PSD2 : modulul de optimizare a plăților funcționează cu Strong Customer Authentication(SCA), care ajută la conformitatea plăților din UE.

Dezavantajele lui Signifyd:

- Fără date în timp real: bazarea pe baze de date poate funcționa. Dar îți lipsește o piesă a puzzle-ului,pentru că escrocii se mișcă repede;
- Conflict de interese: un model de garanție de rambursare poate fi excelent pentru comercianții mici. Dar ai putea pierde afaceri din cauza unui instrument de prevenire exagerat.

Avantajele lui Ekata:

- Instrument dedicat de revizuire manuală: Ekata oferă o soluție completă pentru revizuirea manuală atranzacțiilor și a verificărilor KYC, inclusiv algoritmi de învățare automată (Ekata Identity Engine);
- Vizualizare grafică: vândut ca produs separat Ekata Identity Grap;
- Recunoașterea mărcii: dacă este suficient de bun pentru Microsoft, ar trebui să aveți încredere în el.

Dezavantajele lui Ekataș

- Fără analiză a datelor în timp real: așa cum este adesea cazul cu software-ul antifraudă vechi, aceștia se bazează pe bazele lor uriașe construite de-a lungul anilor. Nu face ca datele să fie bune;
- Contracte scumpe, de lungă durată : nu avem o cifră clară, dar unele recenzii online menționează necesitatea unui buget mare pentru a integra Ekata.[3].

1.3 SCOPUL, OBIECTIVELE ȘI CERINȚELE SISTEMULUI

- Obiectivele acestui sistem informational sunt:
- Cercetarea domeniului „Identificarea fraudelor tranzactiilor bancare”;
- Analiza soluțiilor existente de „ Identificarea fraudelor tranzactiilor bancare”;
- Cercetarea si selectarea instrumentelor pentru dezvoltarea platformei;
- Modelarea funcționala a sistemului;
- Elaborarea unei aplicatie SPARK pentru analiza tranzactiilor si detectarea fraudelor pe baza analizei;
- Detectarea celui mai bun ML algoritm pentru sistemul dat;
- Alegerera unei tehnici de analiza a datelor : supraeșantionarea și subeșantionarea;
- Prelucrearea datelor tranzactiilor care se vor folosi pentru antrenarea ML
- Proiectarea structurii de baza a proiectului;
- Crearea aplicatiei de streaming a tranzactiilor de tip Kafka ;
- Crearea aplicatiei de tip Spring pentru salvarea tranzactiilor in db
- Concluzii și recomandări;
- Sistemul trebuie să identifice tranzacțiile frauduloase în timp real și să prevină tranzacțiile frauduloase ulterioare;
- Sistemul trebuie să protejeze informațiile confidențiale ale clienților, cum ar fi informațiile de cont

bancar și datele personale;

- Documentarea sistemului.

Cerințele sistemului de detectare a fraudelor în tranzacțiile bancare includ:

- Accesul la date: Sistemul trebuie să aibă acces la datele relevante despre tranzacțiile financiare și de plată pentru a putea identifica tranzacțiile frauduloase.
- Analiza datelor: Sistemul trebuie să fie capabil să analizeze datele financiare și de plată pentru a identifica modele și tendințe care pot indica o tranzacție frauduloasă.
- Detecția în timp real: Sistemul trebuie să fie capabil să detecteze tranzacțiile frauduloase în timp real, astfel încât să poată fi prevenite sau anulate.
- Acuratețea și eficiența: Sistemul trebuie să fie precis și eficient în identificarea tranzacțiilor frauduloase, astfel încât să reducă numărul de alerte false și să minimizeze impactul asupra clienților legitimi. Scalabilitatea: Sistemul trebuie să fie capabil să gestioneze un număr mare de tranzacții și să se adapteze la creșterea volumului de date și de tranzacții în timp.
- Scalabilitatea: Sistemul trebuie să fie capabil să gestioneze un număr mare de tranzacții și să se adapteze la creșterea volumului de date și de tranzacții în timp.

Aceste obiective și cerințe sunt esențiale pentru un sistem eficient de detectare a fraudelor în tranzacțiile bancare.

2 MODELAREA ȘI PROIECTAREA SISTEMUL INFORMATIC

Proiectarea sistemului este procesul de proiectare a arhitecturii, componentelor și interfețelor pentru un sistem astfel încât să îndeplinească cerințele utilizatorului final.

Este un domeniu larg de studiu în Inginerie și include diverse concepte și principii care vă vor ajuta în proiectarea sistemelor scalabile. Aceste roluri de conducere necesită o mai bună înțelegere a modului în care rezolvați o anumită problemă de proiectare, a modului în care răspundeți atunci când există mai mult trafic decât se aștepta pe sistemul dvs., a modului în care proiectați baza de date a sistemului și multe altele. Toate aceste decizii trebuie luate cu atenție ținând cont de scalabilitate, fiabilitate, disponibilitate și întreținere.

Un sistem este fiabil atunci când poate îndeplini cerințele utilizatorului final. Când proiectați un sistem, ar fi trebuit să planificați implementarea unui set de caracteristici și servicii în sistemul dvs. Dacă sistemul dumneavoastră poate servi toate aceste funcții fără a se uza, atunci sistemul dumneavoastră poate fi considerat a fi de încredere .[4]

Sistemul trebuie sa fie proiectat in asa fel ca să funcționeze fiabil chiar și în prezența defecțiunilor. Defecțiunile sunt erorile care apar într-o anumită componentă a sistemului. Apariția unei defecțiuni nu garantează defecțiunea sistemului.

Limbajul de proiectare ales este UML care este perfect in domeniul ingineriei software, care are scopul de a oferi o modalitate standard de vizualizare a designului unui sistem.

Din punct de vedere istoric, UML a fost folosit pentru a modela sisteme software, dar nu se limitează la dezvoltatorii de software. Astăzi, diagramele UML sunt folosite și pentru a gestiona procese și proiecte. În acest scenariu, diagramele UML conturează întregi fluxuri de lucru și procese de afaceri.

Cel mai frecvent, o diagramă UML este utilizată pentru a analiza software-ul existent, a modela software-ul nou și pentru a planifica dezvoltarea și prioritizarea software-ului. Mai simplu spus, dacă aveți nevoie de o modalitate de a vizualiza și planifica procesul de dezvoltare a software-ului, o diagramă UML este incredibil de utilă.

Pentru dezvoltatorii de software, menținerea ierarhiilor și relațiilor din cadrul unui sistem poate fi dificilă - mai ales atunci când se concentrează pe zone complexe și detaliate ale codului. Dar utilizarea unui model UML împarte aceste sisteme în componente și subcomponente. Acest lucru facilitează vizualizarea, planificarea și executarea proiectului lor.

Mediul de dezvoltare ales pentru diagramele UML este Enterprise Architect. Acest instrument ne ajuta sa modelam și proiectam sistemele software.

2.1 DESCRIEREA COMPORTAMENTALĂ A SISTEMULUI

Descrierea comportamentala ilustreaza interactiunile generale care se produc in sistem pentru a produce un rezultat.

O descriere comportamentală a unui sistem este o reprezentare a modului în care sistemul se comportă în timpul execuției sale. Aceasta poate fi realizată prin utilizarea unor modele de comportament, cum ar fi diagramele de activitate, diagrama de stat, diagrama de colaborare, diagrama de secvență și altele.

De exemplu, o diagramă de activitate poate fi utilizată pentru a modela procesele de afaceri și activitățile pe care le desfășoară un sistem în timpul execuției sale. Aceasta poate fi utilizată pentru a ilustra secvența de activități care trebuie realizate pentru a finaliza o sarcină sau pentru a atinge un obiectiv.

Pe de altă parte, o diagramă de stare poate fi utilizată pentru a reprezenta diferitele stări ale unui obiect sau sistem, precum și tranzițiile între aceste stări. Aceasta poate fi utilizată pentru a ilustra modul în care un obiect sau sistem răspunde la diferite evenimente și cum se schimbă starea acestuia în funcție de aceste evenimente.

Diagrama de colaborare ilustrează interacțiunile între obiecte sau clase dintr-un sistem, iar diagrama de stare modelează stările și tranzițiile pe care le poate avea un obiect sau sistem.

În general, descrierea comportamentală a sistemului este importantă pentru a înțelege modul în care un sistem funcționează și pentru a identifica problemele din sistem. Aceasta poate fi utilizată în dezvoltarea software, analiza și proiectarea sistemelor și în alte domenii în care este important să se înțeleagă interacțiunile și funcționalitățile unui sistem.

Dupa figura de mai jos noi vedem ca toate tranzactiile vor veni din Transaction Gateway Simulator spre SparkML care va detecta tranzactiile fraudate , mai apoi tranzactiile vor fi salvate in baza de date si vor fi afisate intr-o pagina front end pentru a fi analizate manual.

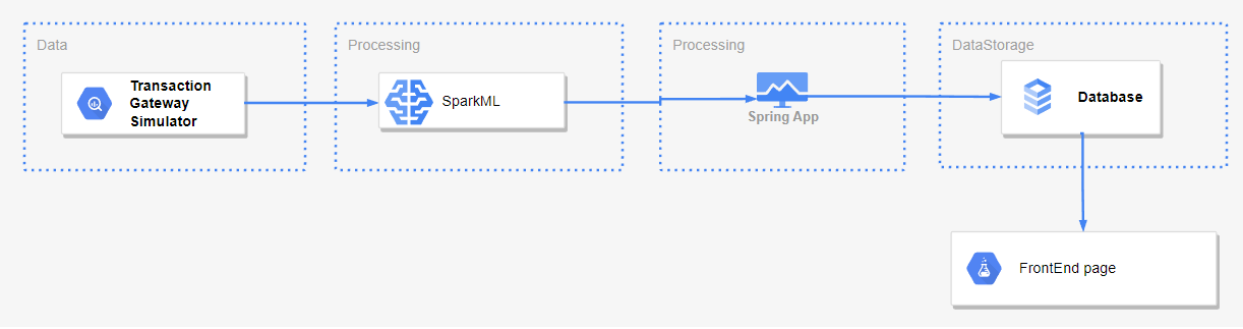


Figura 2.1– Fluxul initial al aplicatiei

2.1.1 IMAGINEA GENERALĂ ASUPRA SISTEMULUI

În aceasta diagrama avem reprezentat procesul prin care se creaza o tranzactie, iar prin relatia include ne dam seama ca este nevoie si de a detecta frauda si a salva tranzactia. Actori sunt trei iar ei sunt Utilizatorul , Fraud Detection Core si Spring Microservice fiecare raspunand de functionalul sau.

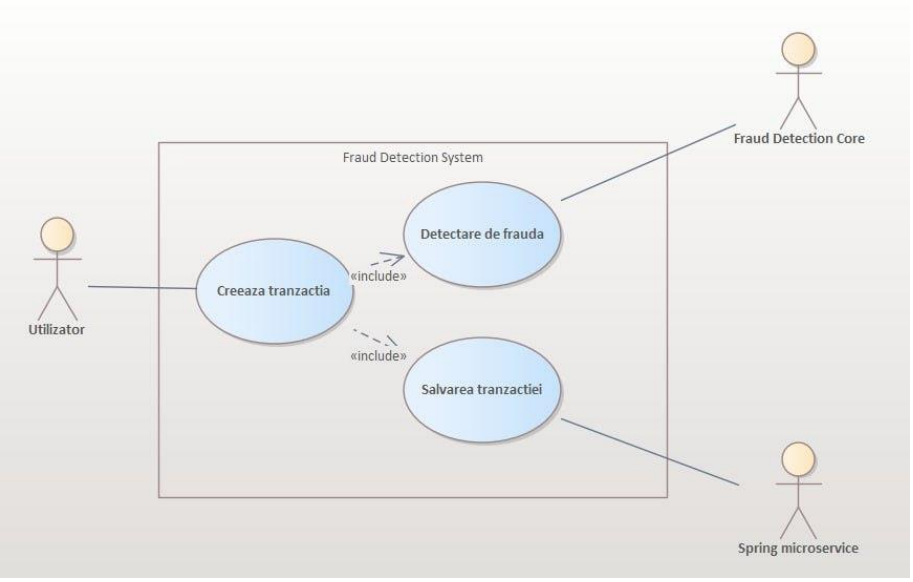


Figura 2.2 – Diagrama use-case a creari unei tranzactii

In diagrama de mai sus este ilustrata detectarea de frauda a unei tranzactii care include crearea modelui padurii aleatorii , preprocesarea datelor tranzactiei , aplicarea tranzactiei asupra padurii alaeatorii iar la final este optional afisarea tranzactiilor fraudate in console.

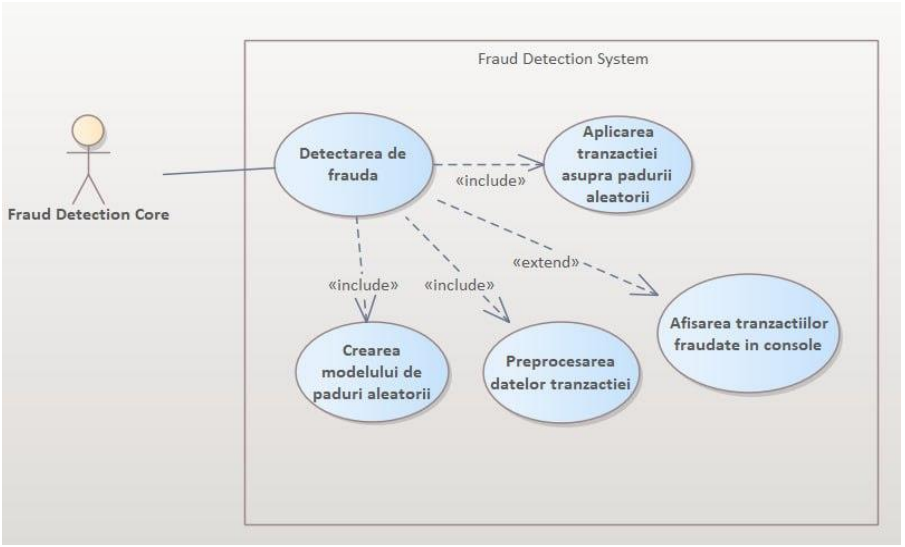


Figura 2.3 – Diagrama use-case a detectarii unei fraude

2.1.2 MODELAREA VIZUALĂ A FLUXURILOR

Diagrama de activitate este o reprezentare grafică a unui proces sau flux de lucru care descrie activitățile și acțiunile care se întâmplă în cadrul acestuia. Este utilizată pentru a vizualiza și a analiza modul în care o sarcină sau o activitate este realizată, evidențiind secvența de acțiuni și decizii luate de către actori implicați în proces.

Diagrama de activitate a microserviciului SparkML ne arata cele 2 tipui de scenarii care se pot întimpla dupa analiza datelor de SparkML, ambele tranzacții vor fi salvate adaugând un field flag deja în timpul salvarii.

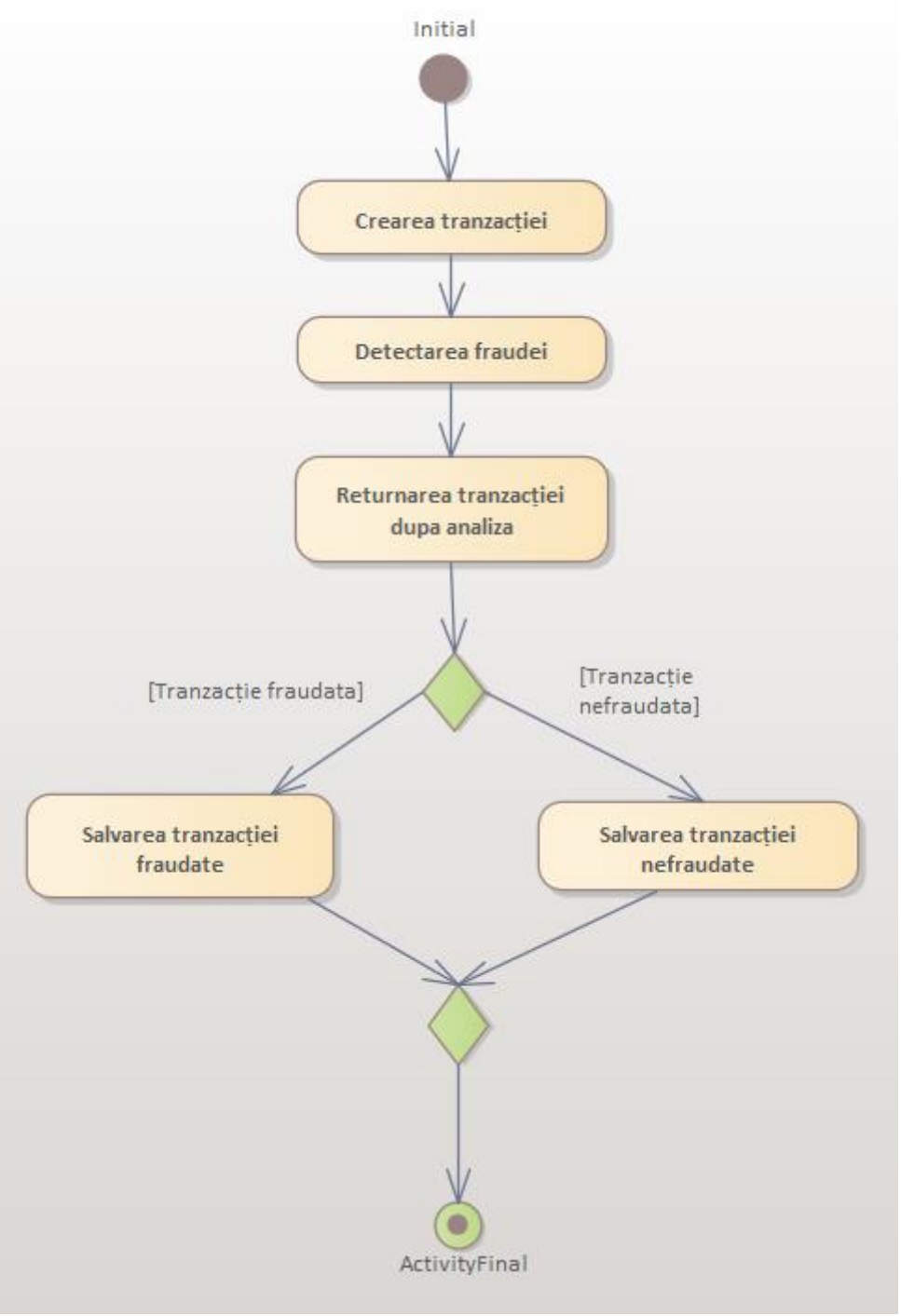


Figura 2.4 - Diagrama de activitate a microserviciului SparkML

2.1.3 STĂRILE DE TRANZACȚIE A SISTEMULUI

Diagrama de stare este o diagramă UML (Unified Modeling Language) utilizată pentru a modela comportamentul unui sistem sau obiect în termeni de stări, tranziții între stări și evenimente care declanșează tranzițiile de stare.

Diagrama de stare este utilă în special pentru modelarea sistemelor sau obiectelor complexe care au multe stări și tranziții posibile între ele. Aceasta poate fi utilizată în ingineria software, analiza și proiectarea sistemelor și în alte domenii în care sistemele complexe trebuie analizate și modelate.

Diagrama de stare pentru efectuarea operatiunilor CRUD ne arata ca preluarea datelor cit și setarea unui DTO trebuie sa fie sincronizate in așa fel ca sa se respecte principiul atomic.

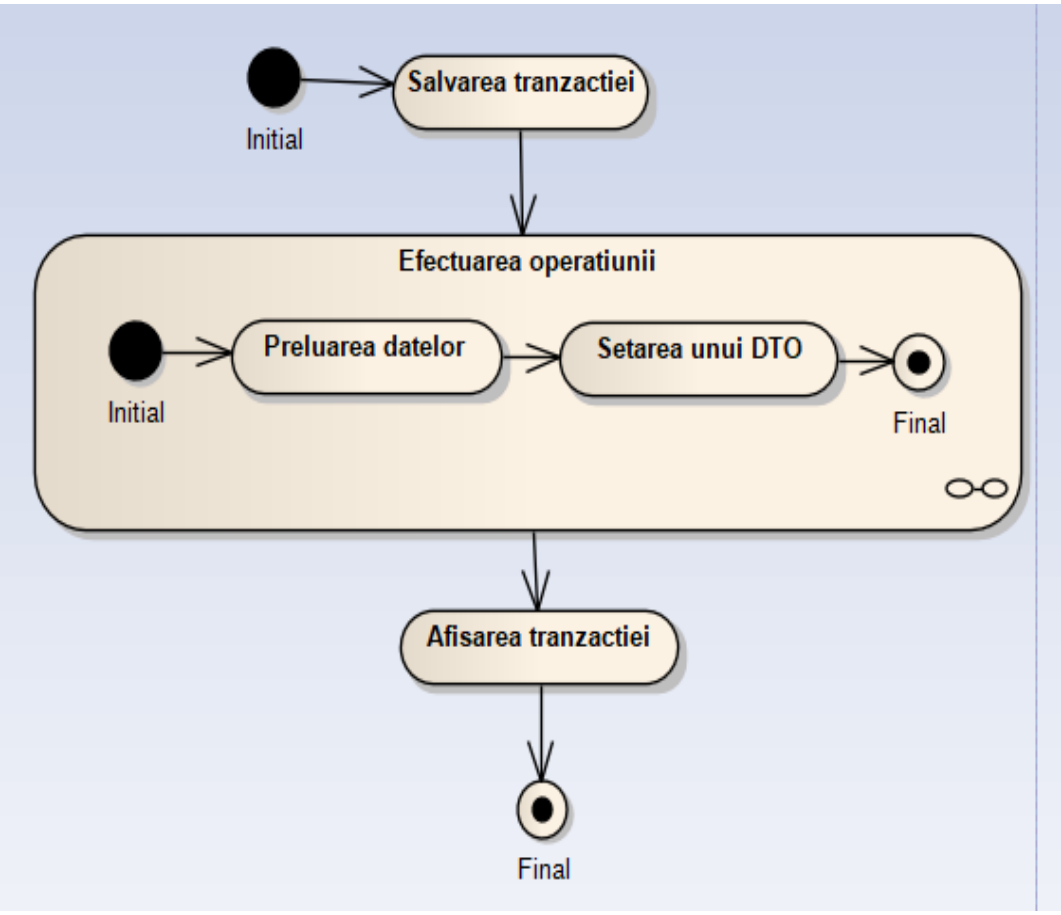


Figura 2.5 – Diagrama de stare pentru efectuarea operatiunilor CRUD care se pot intimpla in microserviciul de tip Spring

Aceste diagrame sunt folosite pentru a descrie diferitele stări în care un obiect sau sistem poate fi și evenimentele care pot declanșa tranziția sa de la o stare la alta.

2.1.4 DESCRIEREA SCENARIILOR DE UTILIZARE A APLICAŢIEI

In aceasta figura este ilustrat comportamentul actorilor in cazul unei tranzactii fraudate. Deci initial dupa design utilizatorul creaza o tranzactie si se duce spre transaction gateway care are rolul de load balancer,validare dar la general el o sa transmita tranzactia mai departe spre instanta care detecteaza frauda , iar in cazul ca tranzactia este fraudata este afisata pe o pagina web.

În aceasta figura este ilustrat comportamentul actorilor in cazul unei tranzactii nefraudate. Comportamentul este mai simplu deoarece dupa design nu este nevoie de afisat o tranzactie nefraudata doar de salvat.

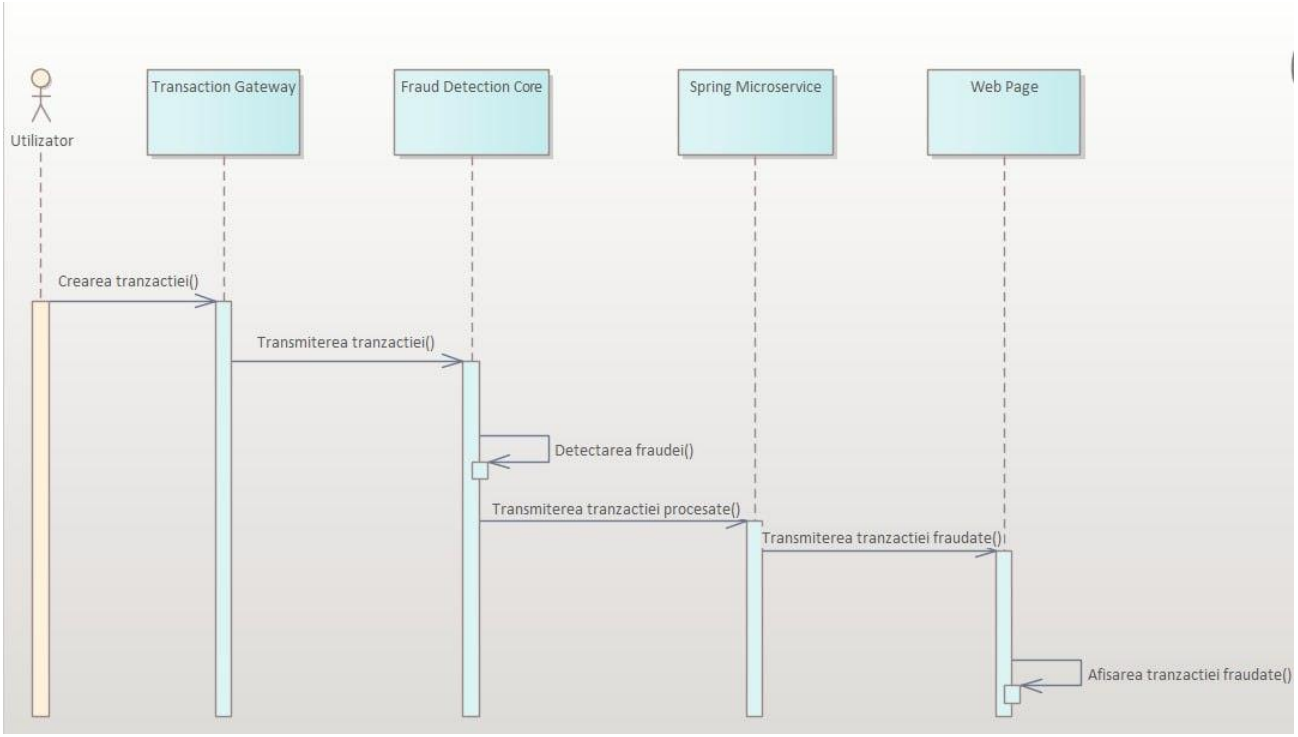


Figura 2.6 – Diagrama de secvența a sistemului cind tranzactia este fraudata

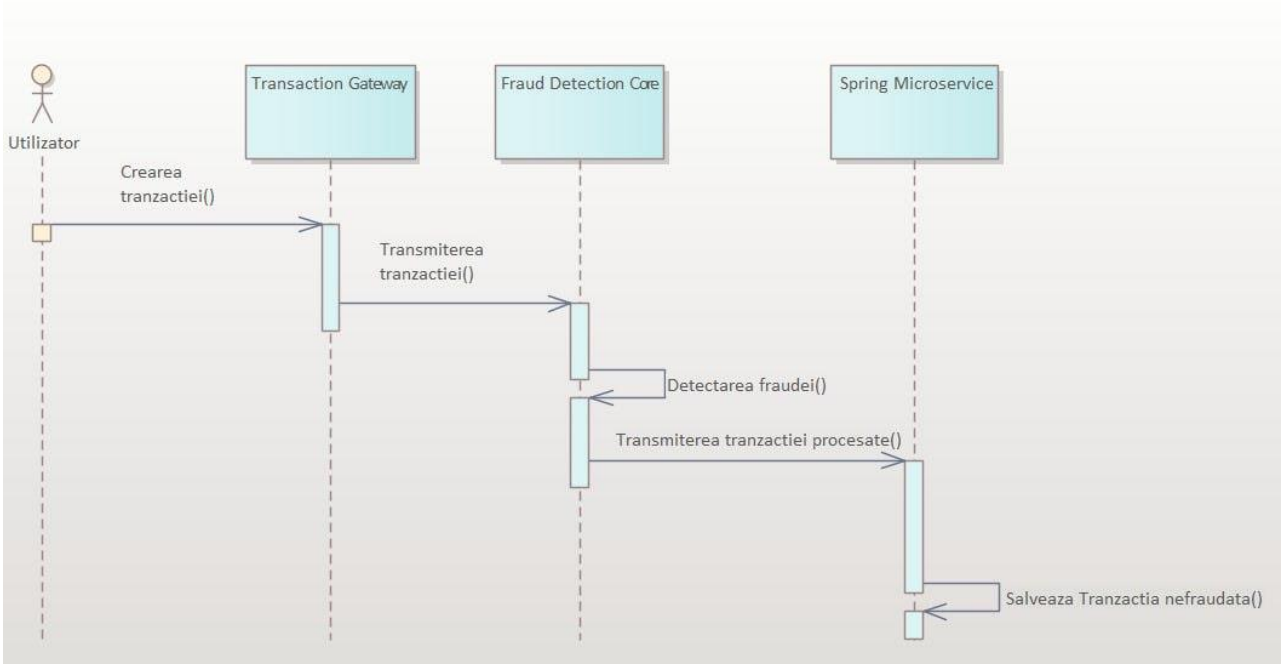


Figura 2.7 – Diagrama de secvența a sistemului cind tranzactia este nefraudata

În aceasta figura este ilustrat comportamentul actorilor in cazul unei tranzactii nevalide. De exemplu utilizatorul creeaza o tranzactie nevalida atunci transaction gateway va intoarce un mesaj de eroare.

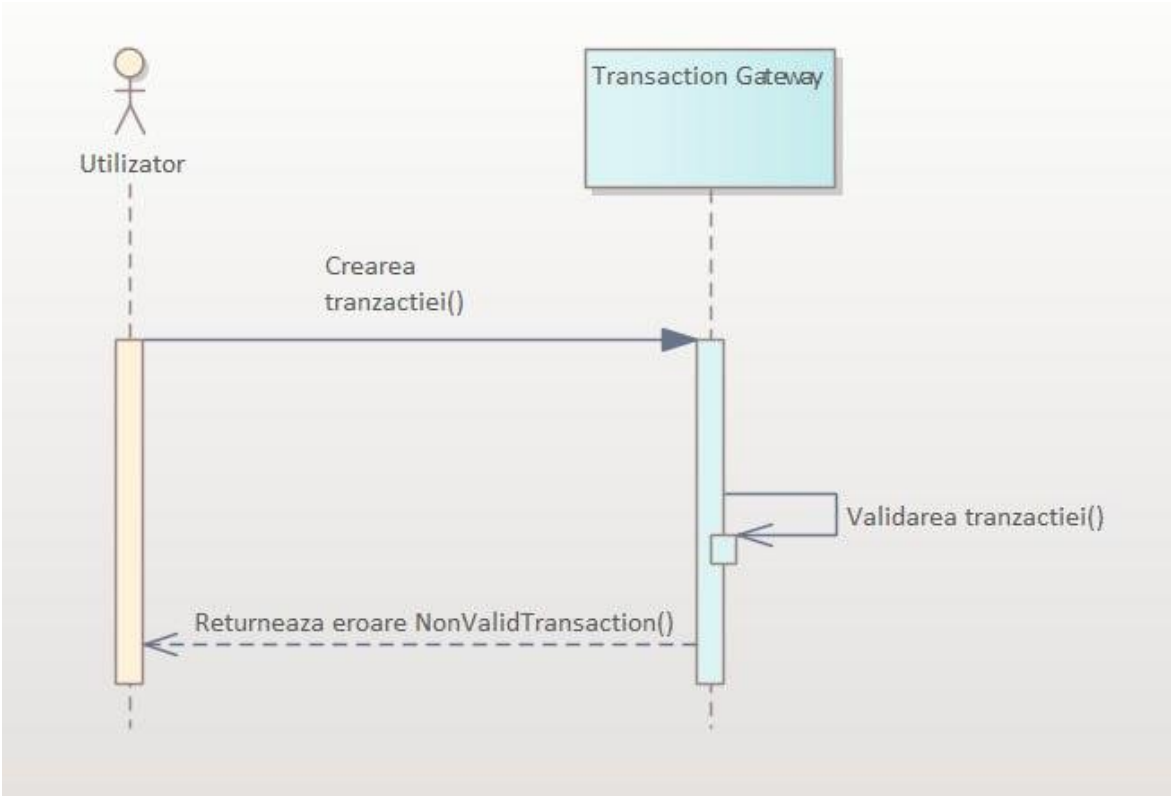


Figura 2.8 – Diagrama de secventa a sistemului cind tranzactia este nevalida.

2.1.5 FLUXURILE DE MESAJE ŞI LEGĂTURILE DINTRE COMPONENTELE SISTEMULUI

Diagrama de colaborare este folosită pentru a ilustra modul în care obiectele sau clasele unui sistem interacționează pentru a realiza o funcționalitate. Aceasta evidențiază mesajele care sunt trimise între obiecte și arată relațiile dintre acestea.

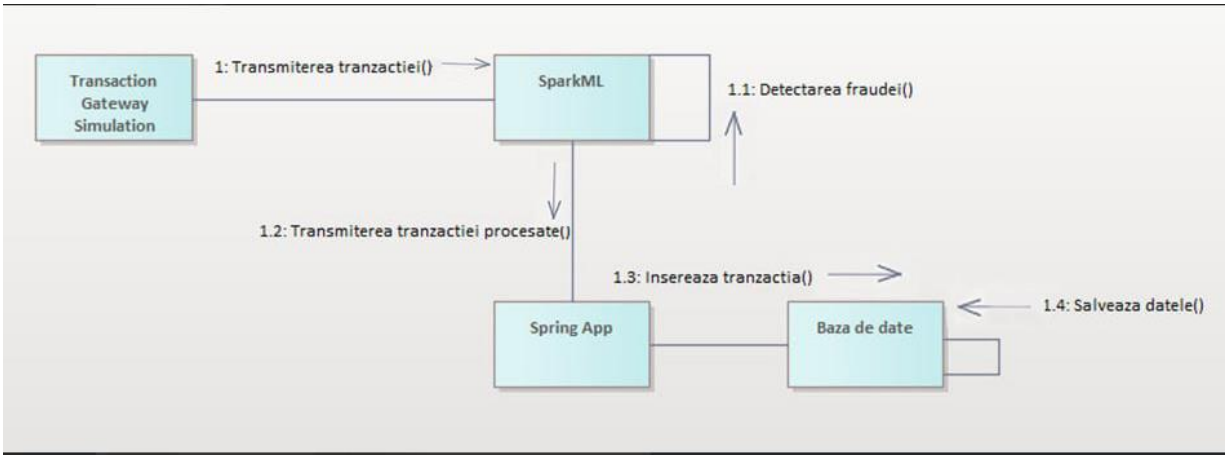


Figura 2.9 – Diagrama de colaborare a sistemului in cazul tranzactiei nefraudate.

In aceasta figura este ilustrat colaborarea actorilor in cazul unei tranzactii nefraudate. Flowul este asemanator cu diagrama de secventa in cazul unei tranzactii nefraudate.

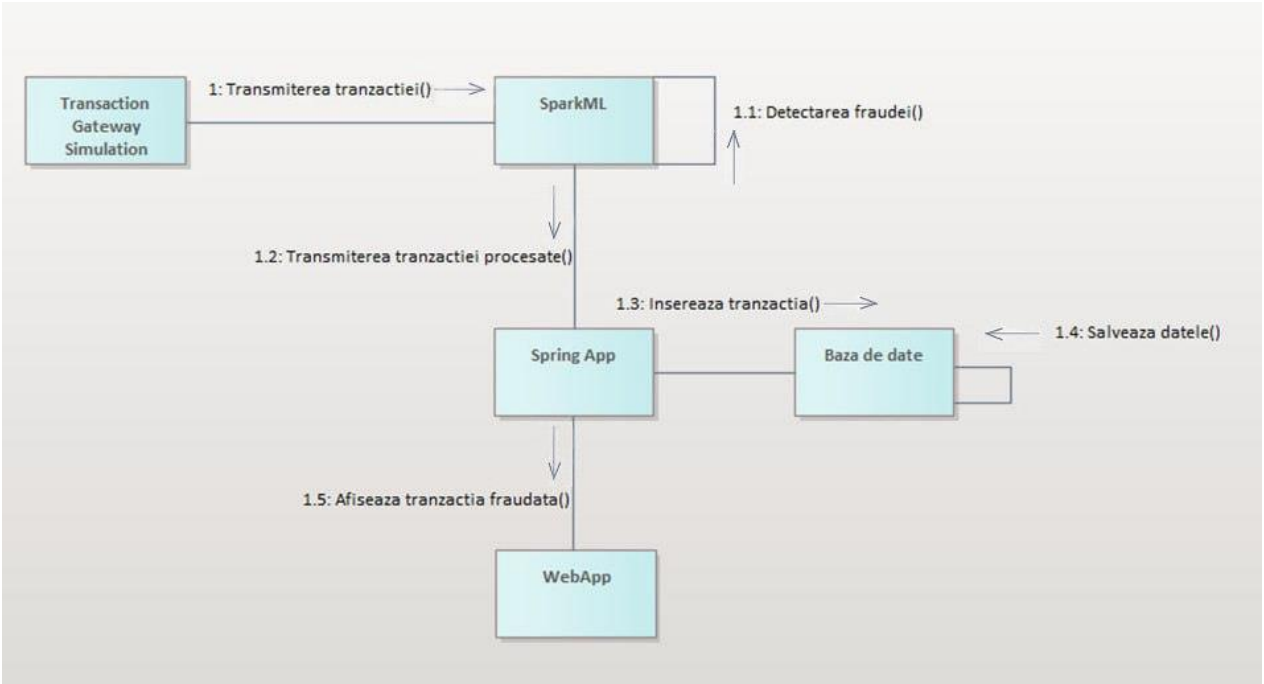


Figura 2.10 – Diagrama de colaborare a sistemului in cazul tranzactiei fraudate

2.2 DESCRIEREA STRUCTURALĂ A SISTEMULUI

Structura statică a unui sistem software poate fi ilustrată prin intermediul diagramelor UML (Unified Modeling Language), care sunt utilizate pentru a reprezenta structura și comportamentul sistemelor software. De exemplu, o diagramă de clasă poate fi utilizată pentru a arăta relațiile dintre diferitele clase și atributele și metodele pe care le conțin. O diagramă de pachet poate fi utilizată pentru a arăta modul în care modulele și clasele sunt organizate într-o structură ierarhică. În general, o diagramă UML poate oferi o vedere clară și precisă asupra structurii statice a unui sistem software.

Relațiile de dependență între componente sunt importante în descrierea structurală a unui sistem, deoarece acestea arată modul în care fiecare componentă depinde de celelalte pentru a funcționa corect și pentru a atinge obiectivele sistemului.

Aceste relații pot fi de mai multe tipuri, cum ar fi:

- Relații de ierarhie - acestea arată modul în care componentele sunt organizate într-o structură ierarhică, cu unele componente având o poziție superioară sau inferioară în ierarhie. De exemplu, într-un sistem informatic, un sistem de operare poate fi componenta superioară a ierarhiei, cu alte componente precum aplicații, drivere și hardware care depind de sistemul de operare pentru a funcționa.

- Relații de asociere - acestea arată modul în care componente diferite lucrează împreună pentru a realiza o anumită funcție sau sarcină. De exemplu, într-un sistem de transport, un motor și un transmisie pot fi componente asociate care trebuie să lucreze împreună pentru a propulsa vehiculul.

- Relații de dependență - acestea arată modul în care o componentă depinde de o altă componentă pentru a funcționa. De exemplu, într-un sistem de securitate, un senzor de mișcare poate depinde de o sursă de alimentare pentru a fi alimentat și a funcționa corect

- Relații de interfață - acestea arată modul în care componente diferite comunică sau interacționează între ele. De exemplu, într-un sistem informatic, un mouse și un ecran pot interacționa prin intermediul unei interfețe grafice cu utilizatorul.

Aceste relații de dependență pot fi reprezentate într-un mod grafic, prin intermediul unui diagramă de relații sau a unei diagrame de blocuri, care ilustrează modul în care componentele sunt interconectate și depind unele de altele. Aceste diagrame pot fi utile pentru înțelegerea structurii sistemului și pentru identificarea problemelor sau defecțiunilor care pot apărea în timpul funcționării acestuia.

2.2.1 DESCRIEREA STRUCTURII STATICE A SISTEMULUI

Diagrama de clase a Fraud-Detection-Core ne ilustreaza interfetele de la care clasele implementeaza functionalitatile necesare , iar Core este locul principal in microserviciu unde si se detecteaza frauda. In CoreProcessor va fi pre procesarea tranzactiilor ce vin.

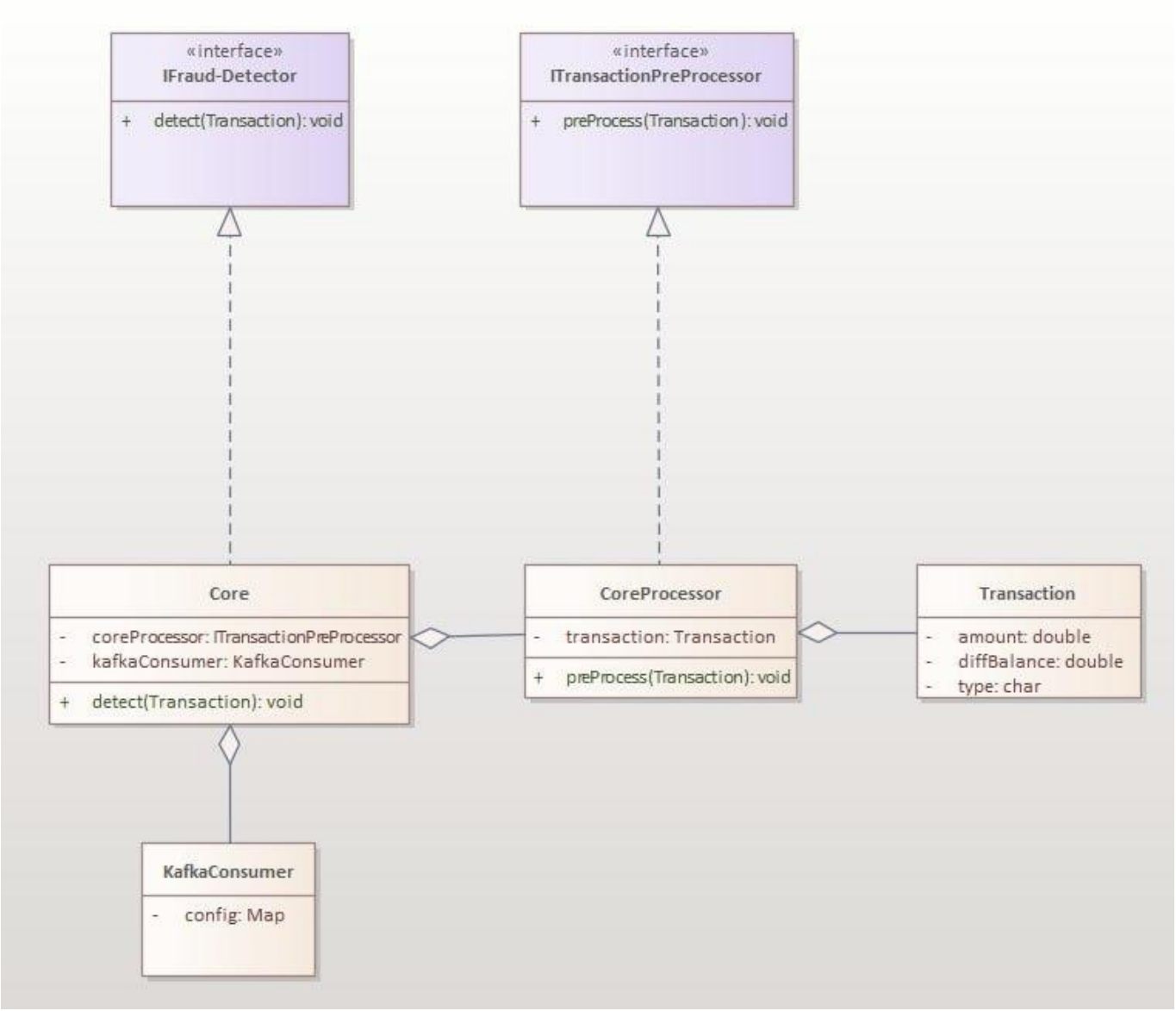


Figura 2.11 – Diagrama de clase a Fraud-Detection-Core

Diagrama de clase a microserviciului de salvare a tranzactiilor care are ca scop doar salvarea tranzactiilor intr-o baza de date , deci avem nevoie de un kafka consumer care va citi datele dintr-un topic , un repositoriu care are implementeaza operatiuniile CRUD.

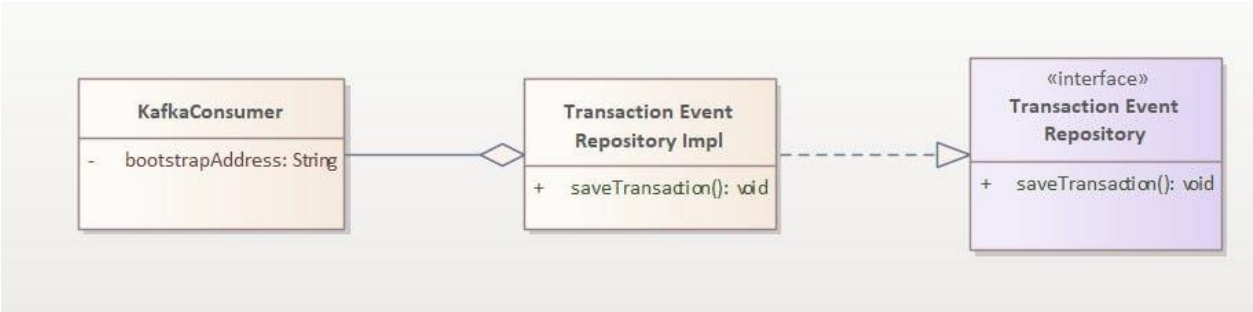


Figura 2.12 – Diagrama de clase a microserviciului de salvare a tranzactiilor

Diagrama de clase a Transaction-Gateway-Simulation ilustreaza un kafka producer , serviciul care are ca scopul principal transmiterea tranzactiilor spre Fraud-Detection-Core

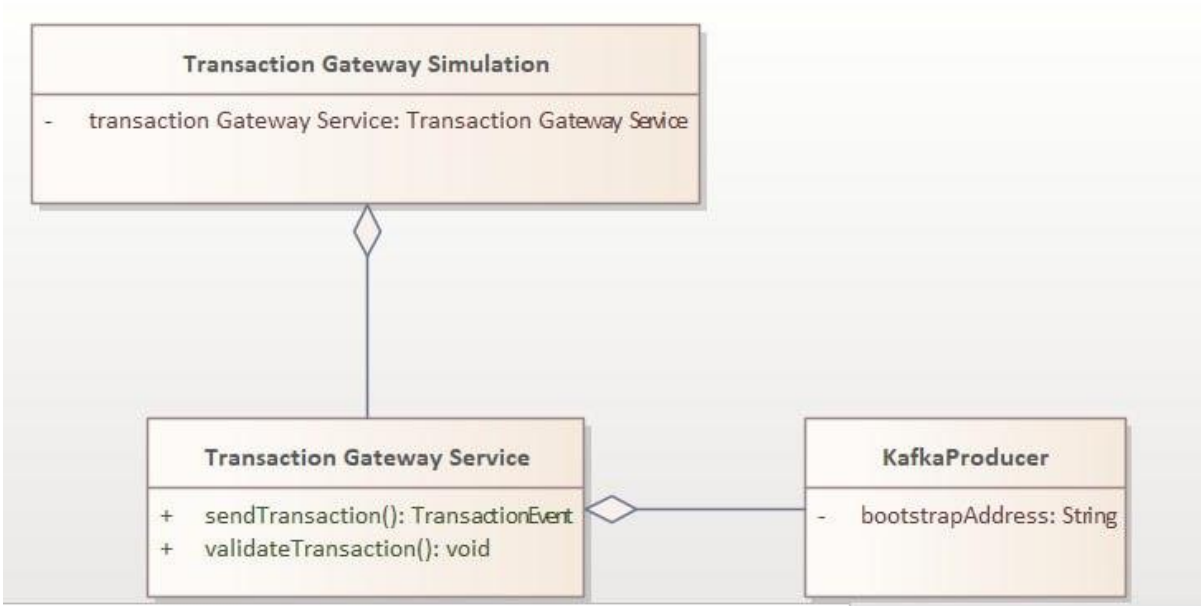


Figura 2.13 – Diagrama de clase a Transaction-Gateway-Simulation

.

2.2.2 RELATIILE DE DEPENDENȚĂ ÎNTRE COMPONENTELE SISTEMULUI

O diagramă de componentă este un tip de diagramă UML (Unified Modeling Language) care este utilizată pentru a reprezenta componentele unui sistem software și interacțiunile dintre acestea. Aceasta este utilizată în special pentru a arăta arhitectura sistemului și structura sa fizică.

Într-o diagramă de componentă, componentele sunt reprezentate sub forma de pătrate sau dreptunghiuri, iar relațiile dintre acestea sunt reprezentate sub formă de săgeți.

În diagrama sunt patru componente care participa in sistemul de detectare a fraudei. Componentele fiind Transaction-Gateway , Fraud-Detection-Core, Spring CRUD App și Baza de date.

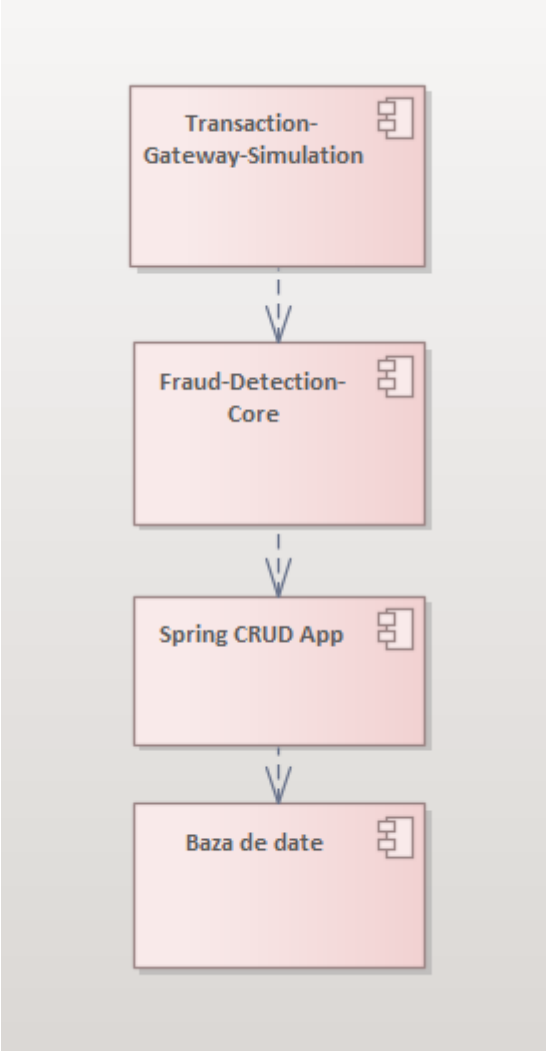


Figura 2.14 – Diagrama de componente a sistemului întreg

2.2.3 MODELAREA ECHIPAMENTELOR MEDIULUI DE IMPLEMENTARE

Diagrama de deployment este o diagramă din domeniul ingineriei software care descrie modul în care componentele fizice sau logice ale unui sistem sunt interconectate și sunt distribuite pe diferite noduri hardware sau software.

Prin utilizarea unei diagrame de deployment, se poate înțelege mai bine arhitectura și distribuția sistemului software și hardware, ceea ce poate fi util pentru dezvoltarea, testarea și implementarea sistemului. De asemenea, poate ajuta la identificarea problemelor de performanță sau a altor probleme care ar putea apărea în timpul utilizării sistemului și la găsirea soluțiilor corespunzătoare.

Diagrama de deployment a sistemului reprezinta artefactele care sunt nevoie de desfașurat pe un server. Ele sunt trei la numar: Transaction-Gateway , Fraud-Detection-Core și Spring Microservice, iar cu ajutorul lui Docker putem sa le desfașuraram în niște containere pe un server. Serverul va rula pe un sistem de operare pe Windows 11.

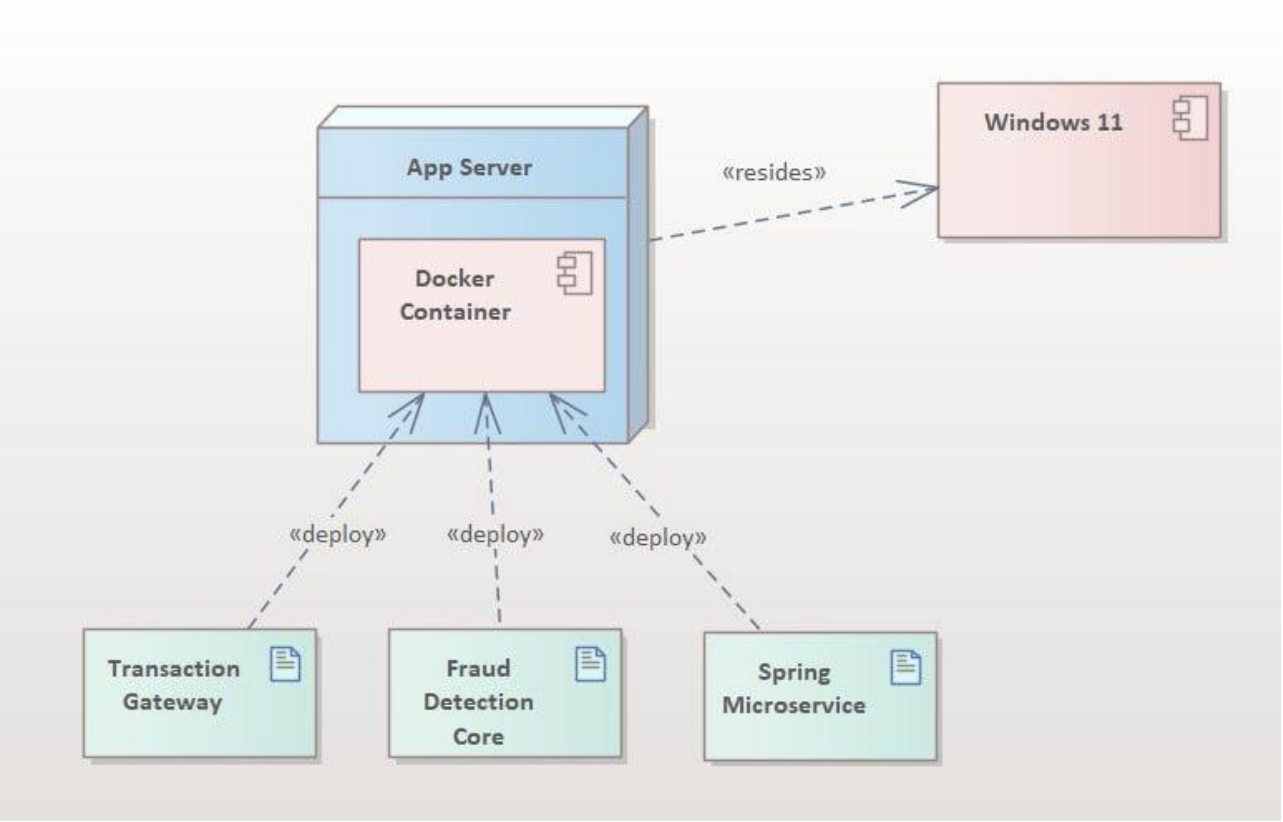


Figura 2.15 – Diagrama de deployment a sistemului

3 REALIZAREA SISTEMULUI

Sistemul va avea nevoie sa proceseze sute de tranzactii pe secunda, la fel si sa foloseasca un set mare de date pentru antrenarea modelului folosit la învățare automata astfel vom avea nevoie de un message broker care va oferi comunicare intre componente. Brokerul care il vom folosi va fi Kafka. Pentru a procesa datele si a detecta fraudele vom folosi Spark ML. Limbajul de baza al sistemului va fi Java.

Java este un limbaj de programare la nivel înalt, bazat pe clasă, orientat pe obiecte, care este proiectat să aibă cât mai puține dependențe de implementare.

Java este al treilea cel mai popular limbaj de programare din lume , după Python și C, care evaluează popularitatea limbajului de programare. Putem atribui utilizarea pe scară largă a limbajului mai multor caracteristici notabile ale Java:

- Versatilitate - Java a fost mult timp limbajul de programare de facto pentru crearea de aplicații web, aplicații Android și instrumente de dezvoltare software , cum ar fi Eclipse, IntelliJ IDEA și NetBeans IDE;

- Instrumente de dezvoltare - Mediul de dezvoltare integrat (IDE) este una dintre cele mai interesante caracteristici ale Java. Java IDE este o colecție de instrumente de automatizare, editori și depanatoare. Ușurință de utilizare . Java are o gramatică asemănătoare englezei, ceea ce îl face ideal pentru începători. Puteți învăța Java în două etape: mai întâi Java de bază, apoi Java avansat;

- Documentare buna - Deoarece Java este un limbaj de programare open-source, este complet gratuit. Java este bine documentat, ceea ce este o caracteristică importantă a limbajului. Include un ghid complet care va explica orice probleme pe care le puteți întâlni la codificare în Java;

- Un API robust - Deși Java are doar aproximativ cincizeci de cuvinte cheie, interfața sa de programare a aplicațiilor (API) este largă și cuprinzătoare, cu diverse metode care pot fi utilizate direct în orice cod;

- Comunitate mare - Suportul comunității pentru Java este unul dintre motivele din spatele popularității sale. Are distincția de a fi a doua cea mai mare comunitate Stack Overflow.[5]

Kafka este folosit în principal pentru a construi conducte de date în flux în timp real și aplicații care se adaptează la fluxurile de date. Combină mesageria, stocarea și procesarea fluxului pentru a permite stocarea și analiza datelor istorice și în timp real.

MLlib este biblioteca de învățare automată (ML) a Spark. Scopul său este de a face învățarea automată practică scalabilă și ușoară. La un nivel înalt, oferă instrumente precum:

- Algoritmi ML: algoritmi comuni de învățare, cum ar fi clasificarea, regresia, gruparea și filtrarea

colaborativă

- Caracterizare: extragerea caracteristicilor, transformarea, reducerea dimensionalității și selecția
- Conducte: instrumente pentru construirea, evaluarea și reglarea conductelor ML
- Persistență: salvarea și încărcarea algoritmilor, modelelor și conductelor
- Utilități: algebră liniară, statistică, manipulare a datelor etc. [6]

3.1 DESCRIEREA LA NIVEL DE COD PE MODULE

Dupa mai multe incercari s-a descoperit ca cel mai bun algoritm este „padurea aleatorie”, cu o precizie ridicata.

Pădurea aleatorie este un algoritm de învățare automat utilizat în mod obișnuit, marcat de Leo Breiman și Adele Cutler, care combină rezultatul mai multor arbori de decizie pentru a ajunge la un singur rezultat .

Ca model de invatare voi folosi padurile aleatorii cu o adincime de trei a algoritmului. Parametrii pe care se va baza algoritmul sunt diferenta de balans si daca este comerciant sau nu. Modelul este salvat ca sa nu trebuiasca la fiecare deschidere a aplicatiei de rulat antrenamentul pe baza de date deoarece se ia mult timp.

```
VectorAssembler vectorAssembler = new VectorAssembler();
vectorAssembler.setInputCols(new String[]{"dest_diff",
        "orig_diff", "surge", "merchantTrack"});

vectorAssembler.setOutputCol("features");
Dataset<Row> inputData = vectorAssembler.transform(csvData).select("label", "features");
Dataset<Row>[] trainingAndHoldoutData = inputData.randomSplit(new double[] {0.9,0.1});
Dataset<Row> trainingData = trainingAndHoldoutData[0];
Dataset<Row> holdoutData = trainingAndHoldoutData[1];

MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator();
evaluator.setMetricName("accuracy");

RandomForestClassifier rfClassifier = new RandomForestClassifier();
rfClassifier.setMaxDepth(3);
RandomForestClassificationModel rfModel = rfClassifier.fit(trainingData);

rfModel.save("src/main/resources/model");
```

Entitatea contine niste fielduri care se folosesc pentru determinarea unicitati a unei tranzacti , la fel se folosesc pentru determinarea fraudei pe baza valorilor ce contin.

```
@Data
public class TransactionEvent implements Serializable {
    private String type;
```

```
    private Double amount;
    private Double oldbalanceOrg;
    private Double newbalanceOrig;
    private Double oldbalanceDest;
    private Double newbalanceDest;
    private Integer isFlaggedFraud;
    private String nameDest;
}
```

Furnizorul de evente Kafka o folosesc pentru a transmite evente spre un Kafka topic pe nume “transaction-gateway”. Clasa contine parametrii necesari pentru conexiune. Avem nevoie de a serializa datele ce transmitem astfel vom folosi StringSerializer pentru cheie si JsonSerializer pentru entitate.

```
@EnableKafka
@Configuration
public class KafkaProducer {

    @Value(value = "${kafka.bootstrapAddress}")
    private String bootstrapAddress;

    @Bean
    public ProducerFactory<String, TransactionEvent> producerFactory() {
        Map<String, Object> configProps = new HashMap<>();
        configProps.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapAddress);
        configProps.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        configProps.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);
        return new DefaultKafkaProducerFactory<>(configProps);
    }

    @Bean
    public NewTopic orders() {
        return TopicBuilder.name("transaction-gateway3")
                .partitions(1)
                .build();
    }

    @Bean
    public KafkaTemplate<String, TransactionEvent> kafkaTemplate() {
        return new KafkaTemplate<>(producerFactory());
    }

}
```

Toate tranzactiile vor fi produse de un KafkaProducer spre un topic care l-a rindul sau va fi consumat de un Kafka Consumer de tip stream.

```
Dataset<TransactionEvent> csvDataKafka = spark.readStream()
        .format("kafka")
```

```
        .option("kafka.bootstrap.servers", "localhost:9092")
        .option("subscribe", "transaction-gateway3")
        .load()
        .selectExpr("CAST(value AS STRING) as message")
        .select(functions.from_json(functions.col("message"), schema).as("json"))
        .select("json.*")
        .as(Encoders.bean(TransactionEvent.class));
```

Deci ne conectam la un topic(transaction-gateway3) cu ajutorul optiunii subscribe pe un anumit server si deserializam toate tranzactiile care vin. Deserializarea se produce prin faptul ca producem un obiect cu ajutorul lui Encoders.bean. La final noi obținem un dataset de tipul TransactionEvent , anume asupra lui se vor produce toate schimbarile viitoare.

```
Dataset<Row> csvDataUnderBallanced = csvData
        .where("label=1");
Dataset<Row> csvDataOverballanced = csvData
        .where("label=0").sample(false, 0.001, 0);
```

De obicei datele sunt prea nebalansate ca sa ne ofere rezultatul dorit , de exemplu datele ce le-am folosit pentru antrenamentul ML aveau un procent de 0.2 tranzactii fraudate ceea ce este tare putin ca sa antreneze ML. In asa situatii se folosesc procese de balansare numite supraeșantionare sau subeșantionare. In proiect am ales subeșantionare care se bazeaza ca datele nefraudate sa fie sterse si sa ramana de exemplu 50/50 date nefraudate si fraudate.

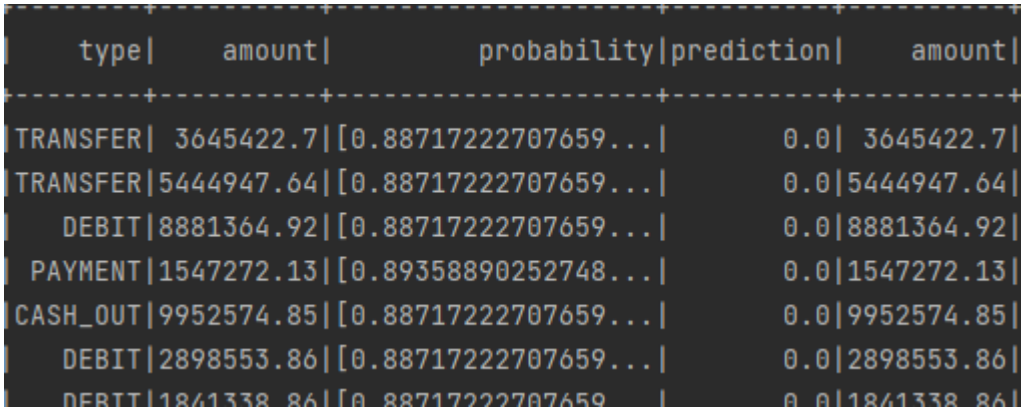
```
@Service
public class TransactionSimulationFactory {

    public TransactionEvent createTransaction() {

        TransactionEvent transactionEvent = new TransactionEvent();

        double amount = BigDecimal.valueOf(nextDoubleRange(10, 10_000_000)).setScale(2,
RoundingMode.HALF_UP).doubleValue();
        double oldbalanceOrg = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale(2,
RoundingMode.HALF_UP).doubleValue();
        double newbalanceOrig = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale(2,
RoundingMode.HALF_UP).doubleValue();
        double oldbalanceDest = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale(2,
RoundingMode.HALF_UP).doubleValue();
        double newbalanceDest = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale(2,
RoundingMode.HALF_UP).doubleValue();

        transactionEvent.setAmount(amount);
        transactionEvent.setOldbalanceDest(oldbalanceDest);
        transactionEvent.setNewbalanceDest(newbalanceDest);
        transactionEvent.setOldbalanceOrg(oldbalanceOrg);
        transactionEvent.setNewbalanceOrig(newbalanceOrig);
```

```
        int randomNumberUsingNextInt = getRandomNumberUsingNextInt(1, 5);
        switch (randomNumberUsingNextInt) {
            case 1 -> transactionEvent.setType("TRANSFER");
            case 2 -> transactionEvent.setType("DEBIT");
            case 3 -> transactionEvent.setType("PAYMENT");
            case 4 -> transactionEvent.setType("CASH_OUT");
        }
        int transfer = (transactionEvent.getType().equals("TRANSFER") &&
transactionEvent.getAmount() > 200_000) ? 1 : 0;
        transactionEvent.setIsFlaggedFraud(transfer);

        String nameDest = getRandomNumberUsingNextInt(1, 3) == 1 ? "M" + new
Random().nextInt(9999999) : "C" + new Random().nextInt(9999999);

        transactionEvent.setNameDest(nameDest);

        return transactionEvent;
    }

    private static double nextDoubleRange(int min, int max) {
        return (new Random().nextDouble() * (max - min)) + min;
    }

    private static int getRandomNumberUsingNextInt(int min, int max) {
        Random random = new Random();
        return random.nextInt(max - min) + min;
    }
}
```

Aceasta secvența de cod reprezinta implementarea prin care noi simulam tranzactiile ce vor fi analizate de Fraud-Detection-Core, din cauza faptului ca la etapa de dezvoltare si testare initiala noi nu avem tranzactii adevarate ne vom folosi de aceasta metoda.

```
csvData = csvData.withColumn("surge", when(col("amount")
        .$greater(450000), 1).otherwise(0));

csvData = csvData.withColumn("dest_change", col("newbalanceDest").$minus(
        col("oldbalanceDest")));

csvData = csvData.withColumn("dest_txn_diff", when(col("dest_change")
        .$less(0), round(col("amount").plus(col("dest_change")), 2))
        .otherwise(round(col("amount").minus(col("dest_change")), 2)));

csvData = csvData.withColumn("dest_diff", when(col("dest_txn_diff")
        .notEqual(0), 1).otherwise(0));

csvData = csvData.withColumn("merchantTrack", when(col("nameDest")
        .startsWith("M"), 1).otherwise(0));


csvData = csvData.withColumn("orig_change", col("newbalanceOrig").$minus(
        col("oldbalanceOrg")));

csvData = csvData.withColumn("orig_txn_diff", when(col("orig_change")
        .$less(0), round(col("amount").plus(col("orig_change")), 2))
        .otherwise(round(col("amount").minus(col("orig_change")), 2)));
```

```
csvData = csvData.withColumn("orig_diff", when(col("orig_txn_diff")
        .notEqual(0), 1).otherwise(0));

csvData = csvData.drop("count", "type", "nameOrig", "nameDest", "step", "destName");
```



type	amount	probability	prediction	amount
TRANSFER	3645422.7	[0.88717222707659...	0.0	3645422.7
TRANSFER	5444947.64	[0.88717222707659...	0.0	5444947.64
DEBIT	8881364.92	[0.88717222707659...	0.0	8881364.92
PAYMENT	1547272.13	[0.89358890252748...	0.0	1547272.13
CASH_OUT	9952574.85	[0.88717222707659...	0.0	9952574.85
DEBIT	2898553.86	[0.88717222707659...	0.0	2898553.86
DEBIT	1841338.86	[0.88717222707659...	0.0	1841338.86

Figura 3.1 – Tranzactii analizate

Deobicei toate datele ce trebuiesc procesate si analizate de un ML sunt neprelucrate si nu vor oferi rezultatul maxim pe care ne dorim. Astfel se fac diferite manipulari asupra datelor ca sa obtinem rezultatul maxim. De exemplu daca suma tranzactiei depaseste 450000 se adauga un parametru nou.

3.2 TESTAREA SISTEMULUI

Testarea unui sistem software este procesul de verificare a funcționalității, performanței și securității sistemului înainte de livrarea acestuia către utilizatori. Scopul testării este de a identifica și de a corecta erorile, problemele și defectele din sistemul software, astfel încât acesta să poată funcționa conform așteptărilor utilizatorilor. Testarea poate fi realizată manual sau automat, iar procedurile și tehnologiile utilizate variază în funcție de tipul și complexitatea sistemului software.

Fiind un sistem care include lucrul între mai multe microservicii avem nevoie de a testa integrarea lor. În primul rând este nevoie de testat si cum ele lucreaza individual, de exemplu Fraud-Detection-Core.

Din cauza faptului ca sistemul este unu de antifrauda chiar daca o tranzactie este nefraudata dar sistemul o analizeaza ca fraudata nu este critic , este mult mai important sa detectam tranzactiile fraude. De exemplu in screenul de mai jos este actualul model. Acuratetea este de 0.88 procente, 1 fiind rezultatul ideal.Label este categorizarea 0 fiind nefraudata si 1 fiind fraudata. Deci doar 7 tranzactii au fost scapate de sistem fiind fraudate , un rezultat bun la inceputul crearii si imbunatatirii modelului.

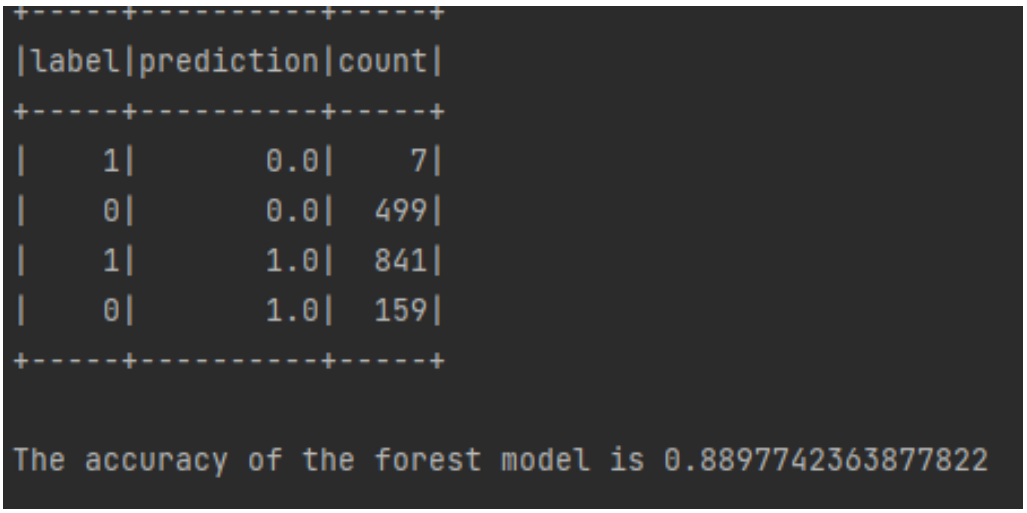


Figura 3.2 – Acuratetea modelului

Cea mai mare problema este și faptul ca la etapa de antrenare noi putem folosi 80 de procente iar pentru testarea modelului celelalte 20 procente. Totuși la integrare este nevoie de a crea sau simula tranzacții ce vor veni din Transaction-Gateway spre Fraud-Detection-Core deci sunt doua opțiunisa folosim celelalte 20 de procente ca sursa de tranzacți ce le vom transmite spre core sau singuri simulam crearea de tranzacții. Astfel ca un avantaj avem ca acele 20 de procente le putem folosi pentru antrenament avind un model mai puternic.

BIBLIOGRAFIE

1. EDGAR LOPEZ-ROJAS; Synthetic Financial Datasets For Fraud Detection.Citat [19.01.23]. Regim de Acess - https://www.kaggle.com/datasets/ealaxi/paysim1
2. Fraud Detection: How Machine Learning Systems Help Reveal Scams in Fintech, Healthcare, and eCommerce.Citat [20.01.23]. Regim de Acess : https://www.altexsoft.com/whitepapers/fraud-detection-how-machine-learning-systems-help-reveal-scams-in-fintech-healthcare-and-ecommerce/
3. Bence Jendruszak; 12 Best Fraud Detection Software and Tools in 2023. Citat [21.01.23]. Regim de Acess : https://seon.io/resources/the-best-fraud-detection-and-prevention-software-tools/#h-what-is-fraud-%20detection-software
4. Saurav Prateek; Getting Started with System Design. Citat [21.01.23]. Regim de Acess : https://www.geeksforgeeks.org/getting-started-with-system-design/
5. What is Java used for?. Citat [22.01.23]. Regim de Acess : https://www.futurelearn.com/info/blog/what-is-java-used-for
6. Machine Learning Library (MLlib) Guide. Citat [23.01.23]. Regim de Acess : https://spark.apache.org/docs/latest/ml-guide

ANEXA A

FIŞA DE ACTIVITATE
Săptâmâna I

Sarcini planificate:

- Descrierea domeniului aplicatiei, tipul de aplicatie si domeniul in care este utilizat
- Argumentarea importantei unui astfel de sistem
- Studierea sistemelor existente asemanatoare cu sistemul planificat.
- Identificarea scopului si obiectivelor, functionalului unui asemenea sistem

Activităţi desfăşurate, observaţii personale

Descrierea domeniului aplicatiei, tipul de aplicatie si domeniul in care este utilizat

Proiectul dat va fi o aplicatie care va detecta fraudele pe baza unor tranzactii care vor veni dintr-o sursa. Există o lipsă de seturi de date publice disponibile privind serviciile financiare și în special în domeniul emergent al tranzacțiilor cu bani mobil. Seturile de date financiare sunt importante pentru mulți cercetători și în special pentru noi care efectuăm cercetări în domeniul detectării fraudelor. O parte a problemei este natura privată a tranzacțiilor financiare, care duce la lipsa unor seturi de date disponibile public.

Tranzactiile vor fi simulate de PaySim care pe baza unui eșantion de tranzacții reale extrase dintr-o lună de jurnalele financiare dintr-un serviciu de bani mobil implementat într-o țară africană.

Abordarea învățării automate (ML) a detectării fraudei a primit multă publicitate în ultimii ani și a mutatinteresul industriei de la sistemele de detectare a fraudei bazate pe reguli la soluții bazate pe ML.Deși, în timp ce sistemele bazate pe reguli sunt inferioare celor bazate pe ML, ele domină totuși piața.

Activitățile frauduloase din domeniul financiar pot fi detectate analizând semnale evidente și la suprafață. În mod neobișnuit, tranzacțiile mari sau cele care au loc în locații atipice merită evident o verificare suplimentară. Sistemele pur bazate pe reguli presupun folosirea de algoritmi care realizează mai multe scenarii de detectare a fraudei, scrise manual de analiștii de fraudă. Astăzi, sistemele vechi aplică aproximativ 300 de reguli diferite în medie pentru a aproba o tranzacție. De aceea sistemele bazate pe reguli rămân prea simple. Acestea necesită adăugarea/ajustarea manuală a scenariilor și cu greu pot detecta corelații implicite. În plus, sistemele bazate pe reguli folosesc adesea software

moștenit care cu greu poate procesa fluxurile de date în timp real care sunt critice pentru spațiul digital.

Cu toate acestea, există, de asemenea, evenimente subtile și ascunse în comportamentul utilizatorului care pot să nu fie evidente, dar semnalează totuși o posibilă fraudă. Învățarea automată permite crearea de algoritmi care procesează seturi mari de date cu multe variabile și ajută la găsirea acestor corelații ascunse între comportamentul utilizatorului și probabilitatea acțiunilor frauduloase. Un alt punct forte al sistemelor de învățare automată în comparație cu cele bazate pe reguli este procesarea mai rapidă a datelor și mai puțină muncă manuală. De exemplu, algoritmii inteligenți se potrivesc bine cu analiza comportamentului pentru a ajuta la reducerea numărului de pași de verificare.[1]

In acest proiect am ca scop si filtrarea datelor si manipularea lor pentru a avea un rezultat final cît mai bun. ML depinde foarte mult de date. Este cel mai crucial aspect care face posibilă antrenamentul cu algoritm și explică de ce învățarea automată a devenit atât de populară în ultimii ani. Dar, indiferent de terabyții dvs. reali de informație și expertiză în știința datelor, dacă nu puteți înțelege înregistrările de date,o mașină va fi aproape inutilă sau poate chiar dăunătoare. Chestia este că toate seturile de date sunt defecte.De aceea, pregătirea datelor este un pas atât de important în procesul de învățare automată. Pe scurt, pregătirea datelor este un set de proceduri care vă ajută să faceți setul de date mai potrivit pentru învățarea automată. În termeni mai largi, pregătirea datelor include și stabilirea mecanismului corect de colectare a datelor. Și aceste proceduri consumă cea mai mare parte a timpului petrecut cu învățarea automată.

Argumentarea importantei unui astfel de sistem

Frauda costă afacerile în profit și reputația mărcii. În unele cazuri, pierderile sunt ireparabile sau durează ani să se recupereze. Prin urmare, învățarea modului de detectare a detectării fraudei în conturile de plătit este crucială pentru a continua producția la timp.

In mod tradițional, companiile se bazau numai pe reguli pentru a bloca plățile frauduloase. Astăzi, regulile sunt încă o parte importantă a setului de instrumente antifraudă, dar în trecut, folosirea lor pe cont propriu a cauzat și unele probleme. Folosirea unei abordări numai cu reguli înseamnă că biblioteca dvs. trebuie să se extindă în continuare pe măsură ce frauda

evoluează. Acest lucru face ca sistemul să fie mai lent și implică o sarcină grea de întreținere asupra echipei dvs. de analişti în fraude, solicitând un număr tot mai mare de revizuiri manuale.

Escrocii lucrează mereu la modalități mai inteligente, mai rapide și mai ascunse de a comite fraude online. Astăzi, infractorii folosesc metode sofisticate pentru a fura date îmbunătățite ale clienților și a uzurpa identitatea clienților autentici, ceea ce face și mai dificil ca regulile bazate pe conturile tipice de fraudă să detecteze acest tip de comportament.

Deși, în timp ce sistemele bazate pe reguli sunt inferioare celor bazate pe ML, ele domină totuși piața.Cu toate acestea, instituțiile financiare de vârf folosesc deja tehnologia ML pentru a combate fraudatorii. De exemplu, MasterCard a integrat învățarea automată și AI pentru a urmări și procesa variabile precum dimensiunea tranzacției, locația, ora, dispozitivul și datele de achiziție.

Astfel din cauza ca metodele frauduloase se dezvolta in continuu este nevoie de realizat un sistem care va detecta in realtime tranzactiile care sunt cu frauda fara a impacta experienta utilizatorului asteptind mult timp cum se intimpla prin metoda bazata pe reguli astfel avem nevoie de un sistem care sta la baza unui component de machine learning ce va detecta acele fraude si va actualiza regular modelul pe baza caruia se bazeaza.

Studierea sistemelor existente asemanatoare cu sistemul planificat.

Software-ul de detectare a fraudei ar trebui să acopere o mulțime de baze, de la prevenirea rambursării până la verificarea identității. Iată exemple de caracteristici obligatorii:

- Reguli de risc: piatra de temelie a detectării fraudelor. Regulile de risc vă permit să filtrați acțiunile utilizatorului în funcție de datele pe care le găsiți. O regulă de risc de bază ar fi: dacăadresa IP aparține unui VPN, blocați încercarea de autentificare;
- Scorul de risc: Unele reguli pot fi statice. cu opțiunea simplă de a permite sau de a respinge acțiunile utilizatorului. Sistemele mai sofisticate vă permit să vă jucați cu scoruri și praguri, care vă oferă o evaluare a cât de riscantă este o acțiune. De exemplu: dacă un client nu are sediul în aceeași țară cu cardul său de plată, adăugați 5 puncte la scorul său de risc;
- Monitorizare în timp real: luați în considerare nevoile dvs. Când vine vorba de prevenirea rambursărilor sau de a respecta reglementările împotriva spălării banilor , software-ul dvs. de detectare trebuie să poată monitoriza

plățile pe site-ul dvs. Pentru verificarea actului de identitate și protecția contului, trebuie, de asemenea, să puteți bloca imediat o acțiune suspectă înainte de a fi prea târziu;

- Motor de învățare automată: sunt șanse să examinați cu atenție volume uriașe de date. Cel maibun software de prevenire a fraudei va include un algoritm ML sau de învățare automată, care vă poate ajuta să sugerați reguli de risc pe baza datelor istorice ale afacerii dvs. Puncte bonus dacă este un sistem de cutie albă care vă permite să înțelegeți raționamentul din spatele regulilorde risc.

Fiecare sistem are plusurile si minusurile sale , iar in urma cercetarii domeniului am descoperit aceste3 sisteme de detectare a fraudelor bancare:

- Kount – Fraude la nivel de întreprindere;
- Ekata – parte a grupului Mastercard;
- Signifyd – Combaterea automată a fraudei pentru comercianți.

Kount se concentrează asupra tehnologiei inovatoare de risc (deține 30 de brevete de tehnologie) și clienților întreprinderi, precum Barclays, PetSmart, Staples și Chase, printre altele. Ele oferă tot ce aveți nevoie pentru monitorizarea și protecția CNP (cardul nu este prezent) prin Kount Command.

Ekata – parte a grupului Mastercard Un alt software matur și bine stabilit de detectare a fraudelor. Fondată în 1997 (inițial sub numele Whitepages), Ekata oferă verificarea identității globale și prevenirea fraudei prin intermediul API-urilor, inclusiv un API de risc de tranzacție, un API Address Risk și un API Phone Intelligence.

- Contracte scumpe, de lungă durată : nu avem o cifră clară, dar unele recenzii online menționează necesitatea unui buget mare pentru a integra Ekata.[4]

Signifyd – protejează acum 10.000 de magazine online la nivel global, ajutându-le să prevină rambursările prin trei produse cheie: Protecția veniturilor, Protecția împotriva abuzurilor și Optimizarea plăților. Produsele lor sunt orientate în mod special către volume ridicate de tranzacții și chiar automatizeazăprotecția contra costurilor printr-un model de garanție de rambursare, în care își asumă plata taxelor de administrare a rambursării. [4]

Kount are capacități limitate de machine learning, în ciuda capacităților de creare a regulilorpersonalizate, funcțiile de învățare automată nu sunt la egalitate cu unele soluții mai noi de pe piață, potrivitrecenziilor, la fel este prezenta o lipsa de îmbogățire a datelor. Acuratețea de prevenire a fraudei este la felde bună ca și datele la care aveți acces. Din păcate, cu puncte de date limitate, combaterea fraudei devine mai grea.

Avantajele lui Kount sunt:

- Acoperă multe industrii: varietatea de produse disponibile în ofertă înseamnă că puteți utiliza Kountpentru un gateway de plată de mare volum sau un brand de comerț electronic de nivel 1.
- Reguli automate și personalizabile: personalizați regulile pentru a automatiza prevenirea rambursăriisau pentru a reduce timpul de examinare manuală, printre altele.

Signifyd este o alegere daca avem de-a face cu volume mari de tranzacții. La fel are urmatoareleavantaje:

- Prevenirea automată a rambursării: nu este chiar plug-and-play, dar Signifyd este proiectat să rulezepe pilot automat cu intrare minimă a utilizatorului.
- Acoperă alte provocări legate de frauda comerțului electronic: abuz de returnare și fraudă prietenoasă,printre altele;
- Conform PSD2 : modulul de optimizare a plăților funcționează cu Strong Customer Authentication(SCA), care ajută la conformitatea plăților din UE.

Dezavantajele lui Signifyd:

- Fără date în timp real: bazarea pe baze de date poate funcționa. Dar îți lipsește o piesă a puzzle-ului,pentru că escrocii se mișcă repede.
- Conflict de interese: un model de garanție de rambursare poate fi excelent pentru comercianții mici.

Dar ai putea pierde afaceri din cauza unui instrument de prevenire exagerat.

Avantajele lui Ekata:

- Instrument dedicat de revizuire manuală: Ekata oferă o soluție completă pentru revizuirea manuală atranzacțiilor și a verificărilor KYC, inclusiv algoritmi de

învățare automată (Ekata Identity Engine);

- Vizualizare grafică: vândut ca produs separat Ekata Identity Graph.
- Recunoașterea mărcii: dacă este suficient de bun pentru Microsoft, ar trebui să aveți încredere în el.Dezavantajele lui Ekata:
- Fără analiză a datelor în timp real: așa cum este adesea cazul cu software-ul antifraudă vechi, aceștiase bazează pe bazele lor uriașe construite de-a lungul anilor. Nu face ca datele să fie bune;
- Contracte scumpe, de lungă durată : nu avem o cifră clară, dar unele recenzii online menționeazănecesitatea unui buget mare pentru a integra Ekata.[4]

Identificarea scopului si obiectivelor, functionalului unui asemenea sistem

Obiectivele acestui sistem informational sunt:

- Cercetarea domeniului „Identificarea fraudelor tranzactiilor bancare”;
- Analiza soluțiilor existente de „ Identificarea fraudelor tranzactiilor bancare”;
- Cercetarea si selectarea instrumentelor pentru dezvoltarea platformei;
- Modelarea funcționala a platformei;
- Proiectarea unei bazei de date NoSQL;
- Elaborarea unei aplicatie SPARK pentru analiza tranzactiilor si detectarea fraudelor pe baza analizei;
- Detectarea celui mai bun ML algoritm pentru sistemul dat;
- Alegerea unei tehnici de analiza a datelor : supraeșantionarea și subeșantionarea;
- Prelucrearea datelor tranzactiilor care se vor folosi pentru antrenarea ML
- Proiectarea structurii de baza a proiectului;
- Crearea aplicatiei de streaming a tranzactiilor de tip Kafka ;
- Crearea aplicatiei de tip Spring pentru salvarea tranzactiilor in db
- Testarea sistemului;
- Concluzii și recomandări;

1. https://www.altexsoft.com/whitepapers/fraud-detection-how-machine-learning-systems-help-reveal-scams-in-fintech-healthcare-and-ecommerce/
2. https://seon.io/resources/the-best-fraud-detection-and-prevention-software-tools/#h-what-is-fraud-%20detection-software

FIŞA DE ACTIVITATE
Săptâmâna II

Sarcini planificate:

- Descrierea comportamentală a sistemului (imaginea generală asupra sistemului; modelarea vizuală a fluxurilor; stările de tranzacție a sistemului (Statechart Diagram); descrierea scenariilor de utilizare a aplicaţiei (Sequence Diagram); fluxurile de mesaje şi legăturile dintre componentele sistemului (Collaboration Diagram).
- Descrierea structurală a sistemului (descrierea structurii statice a sistemului; relatiile de dependență între componentele)
- Se face o introducere în ce s-a folosit pentru realizarea sarcinilor. Se descrie pe scurt tehnologiile folosite, limbaje de programare utilizate, instrumente, algoritmi, șabloane...etc , totul ce s-a folosit la realizarea proiectului. · Se descrie la nivel de cod realizarea funcționalităților
- Se descrie testarea sistemului

Activităţi desfăşurate, observaţii personale

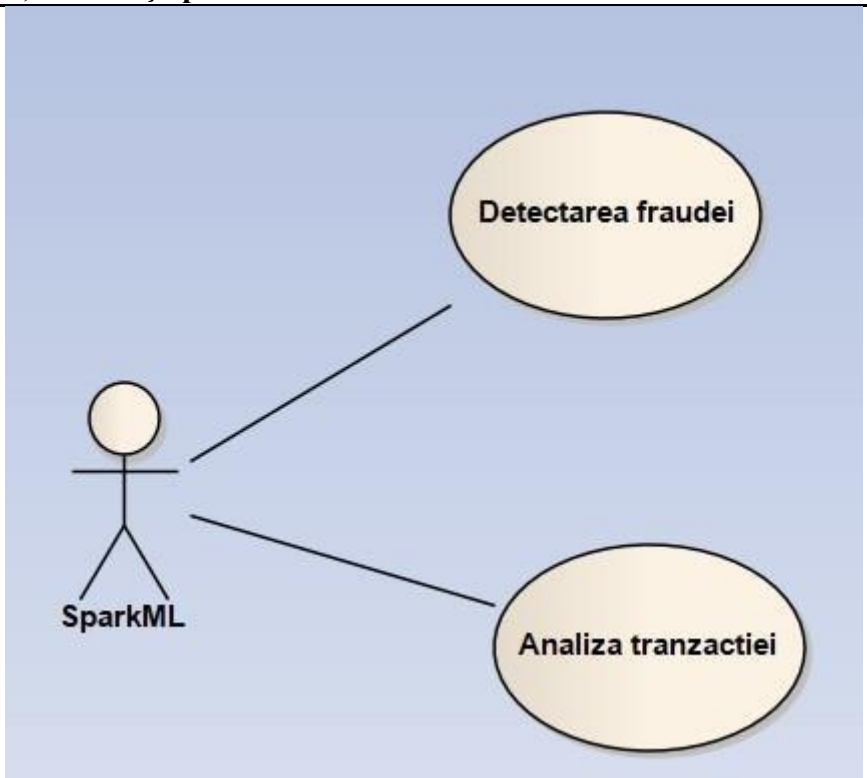


Figura 1 – Diagrama use-case a microserviciului pe SparkML

Diagrama use-case a microserviciului pe SparkML reprezinta functionalul de baza a componentei.

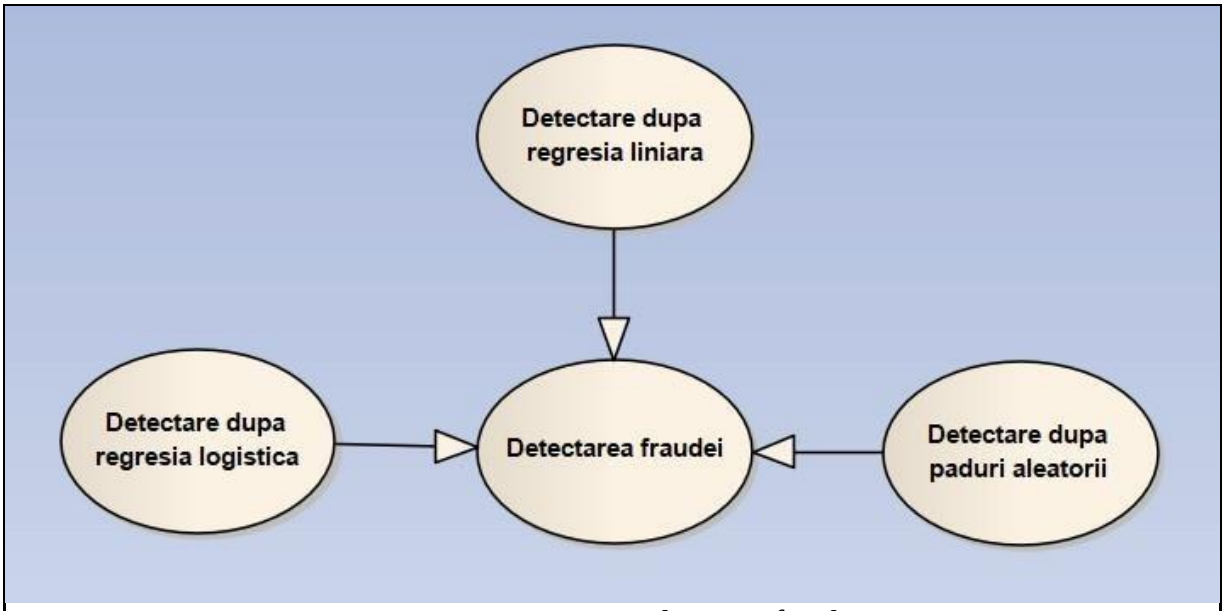


Figura 2 - Diagrama use-case a detectarii fraudei

Diagrama use-case a detectarii fraudei este aratata prin 3 use-caseuri care la rindul lor suntmodelele de invatare automata : regresia logistica , paduri aleatorii , regresia liniara .

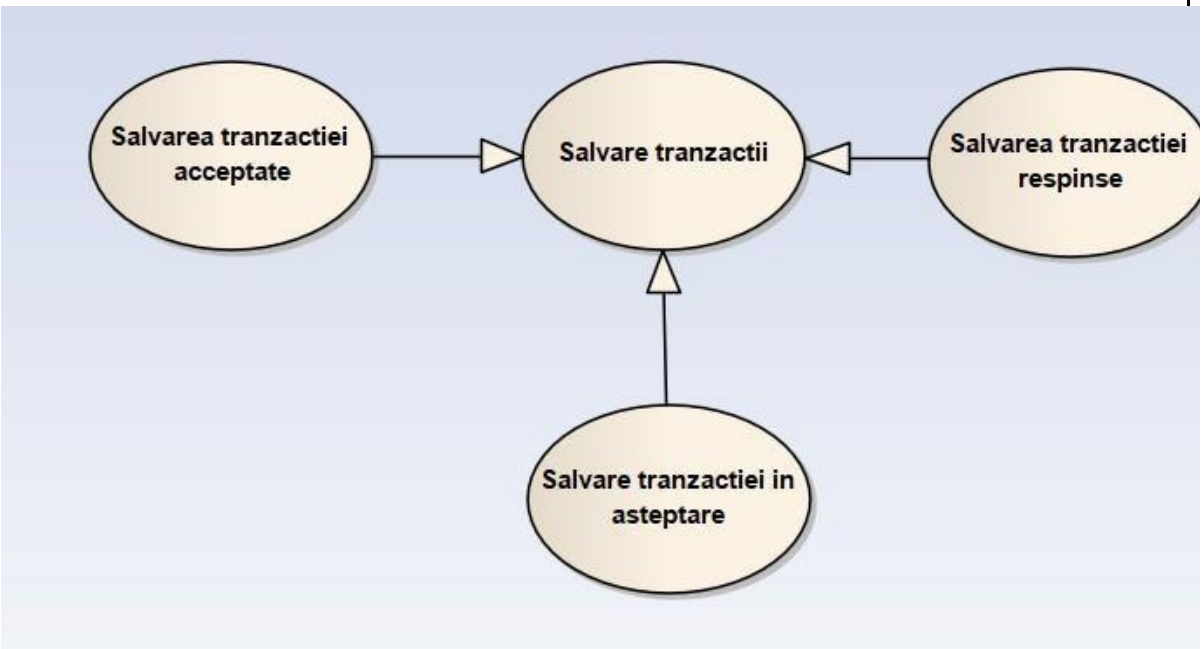


Figura 3 – Diagrama use-case a salvarii tranzactiei

Diagrama use-case a salvarii tranzactiei este ilustrata prin 3 cazuri in dependenta de tipultranzactiei.

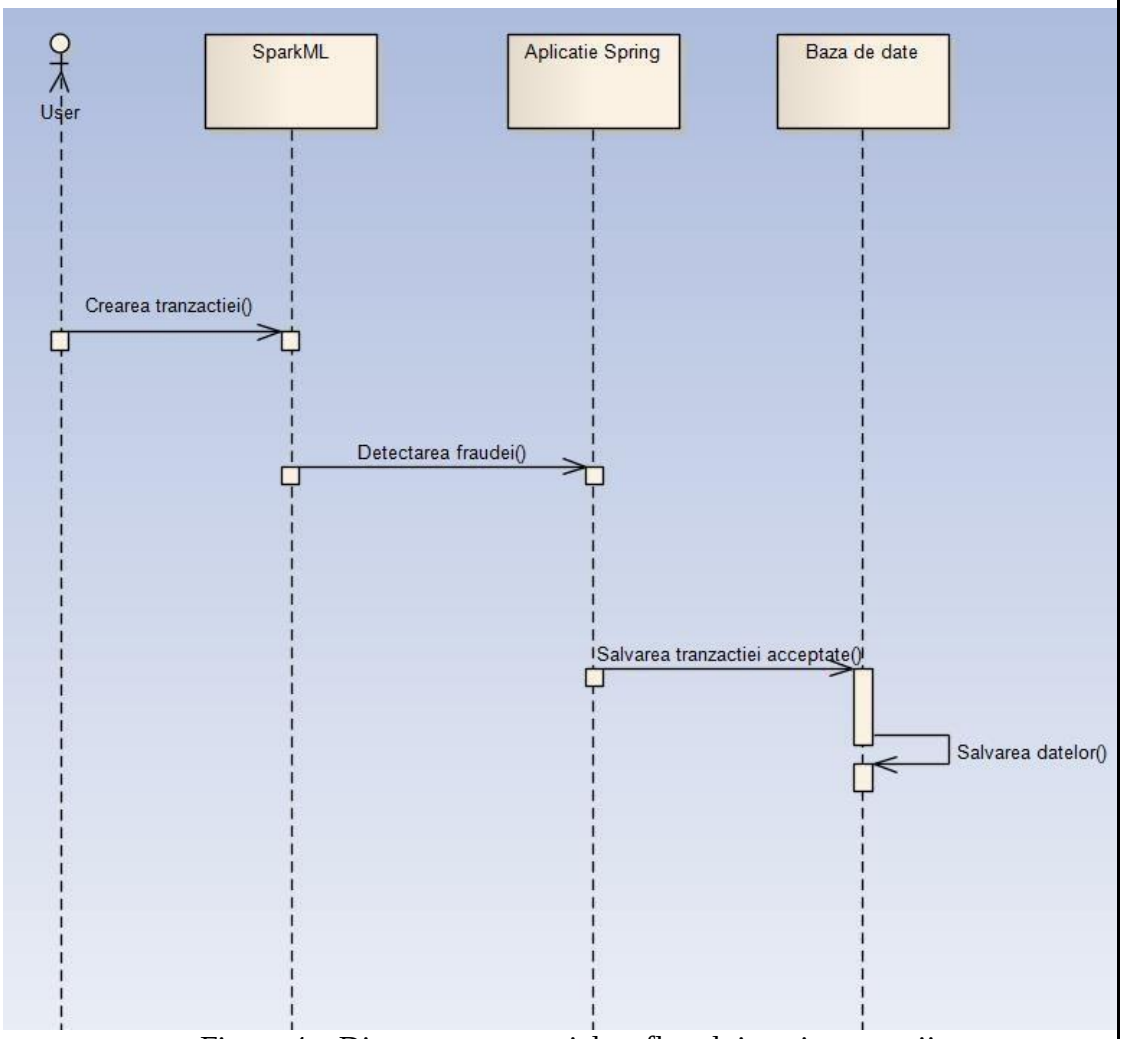


Figura 4 – Diagrama secventiala a fluxului unei tranzactii acceptate

Diagrama secventiala a fluxului unei tranzactii acceptate arata cursul unei tranzactii care are unprocent mare de estimare ca nu este cu frauda in urma analizei de SparkML.

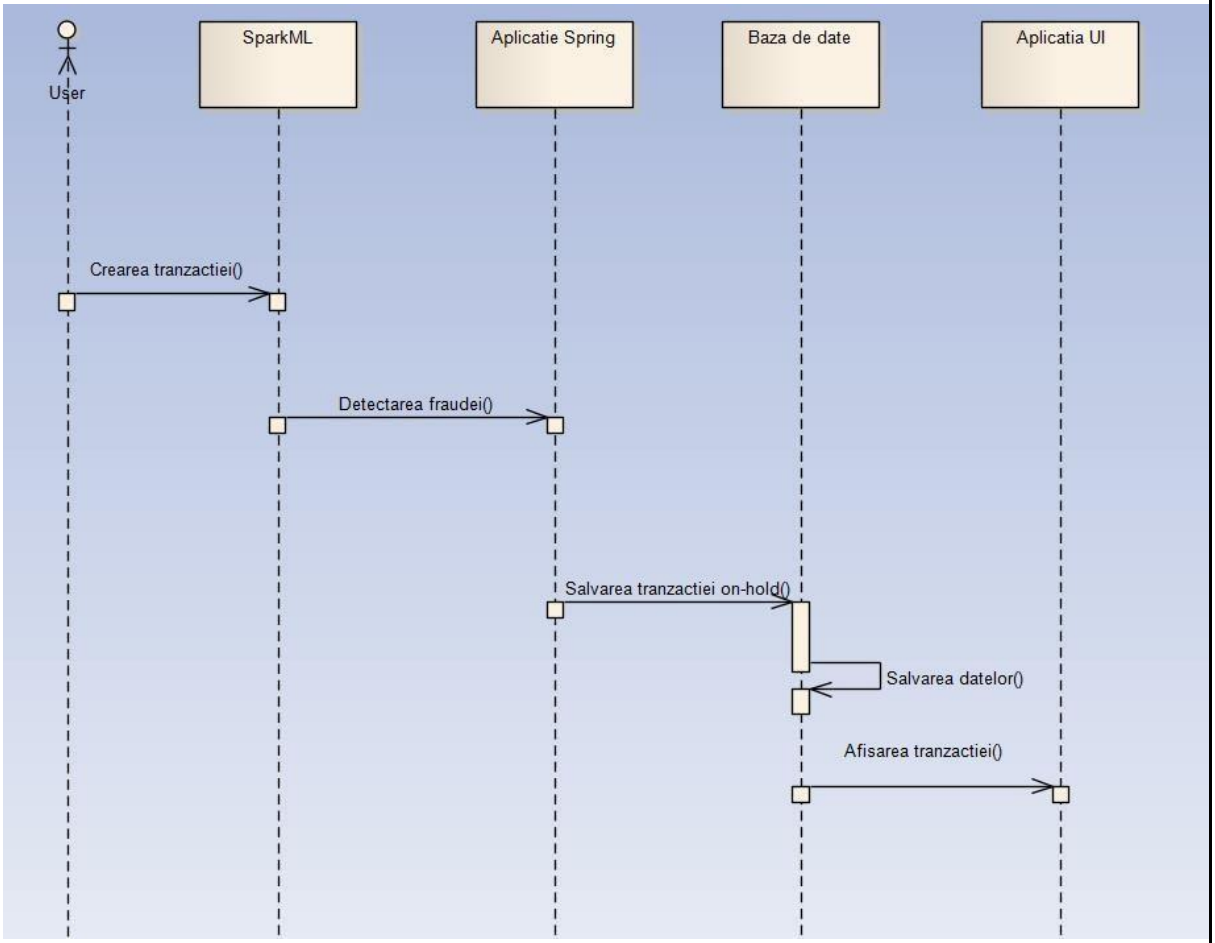


Figura 5 – Diagrama secventiala a fluxului unei tranzactii in asteptare

Diagrama secventiala a fluxului unei tranzactii in asteptare arata cursul unei tranzactii care are un procentmediu de estimare ca nu este cu frauda in urma analizei de SparkML.

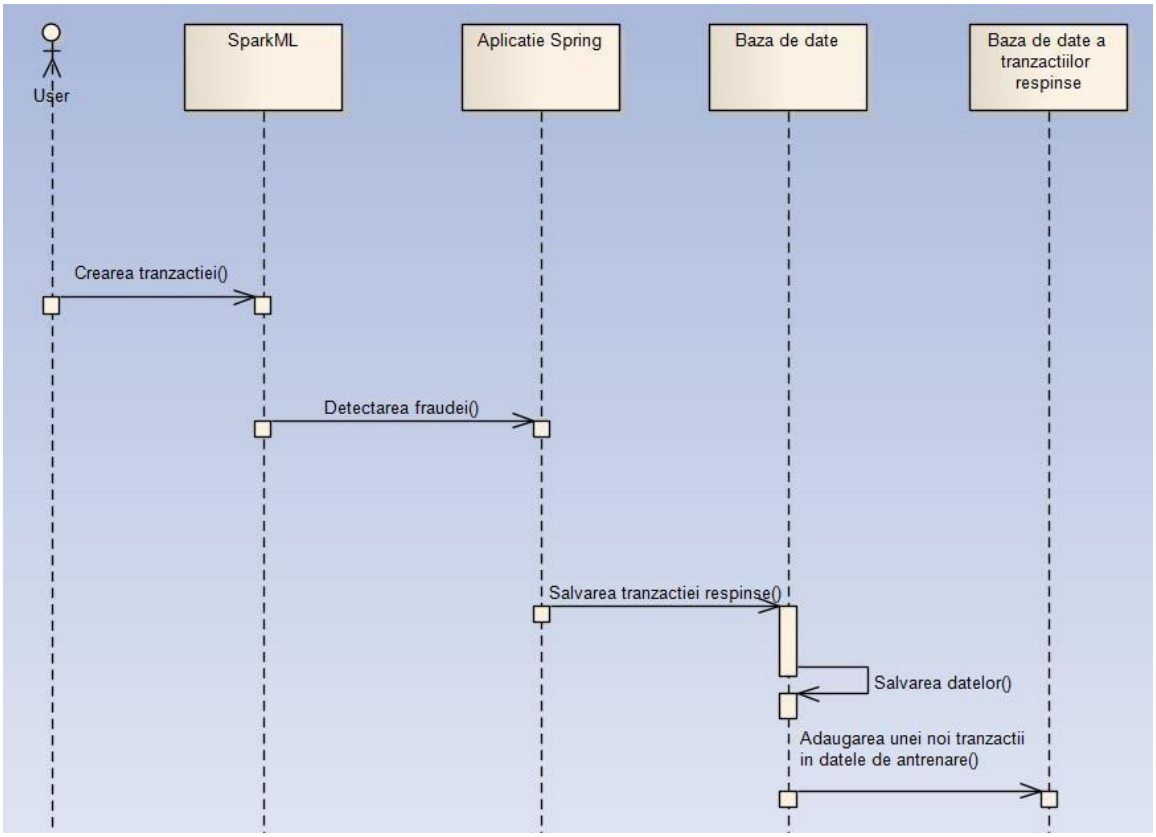


Figura 6 – Diagrama secventiala a fluxului unei tranzactii respinsa

Diagrama secventiala a fluxului unei tranzactii respinsa arata cursul unei tranzactii care are unprocent mic de estimare ca nu este frauda in urma analizei de SparkML.

Figura 7 – Diagrama de colaborare a fluxului unei tranzactii acceptate

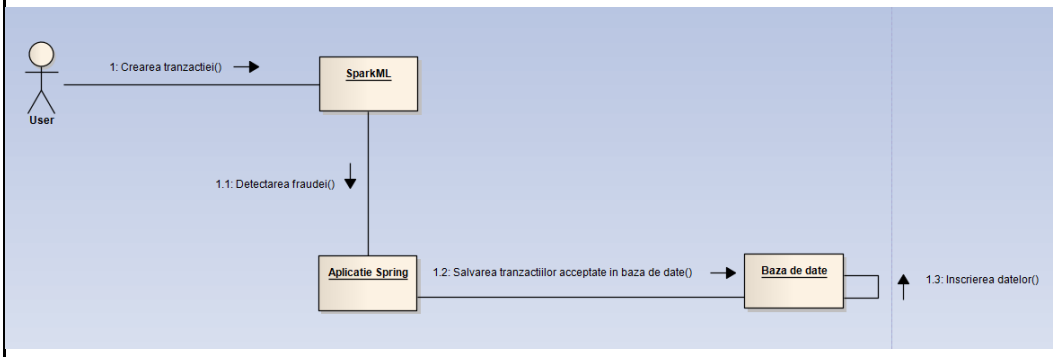


Diagrama de colaborare a fluxului unei tranzactii acceptate ilustreaza componentele si tipul de interactiuni intre ele. Mesajele sunt asinhrone deci nu asteapta raspuns.

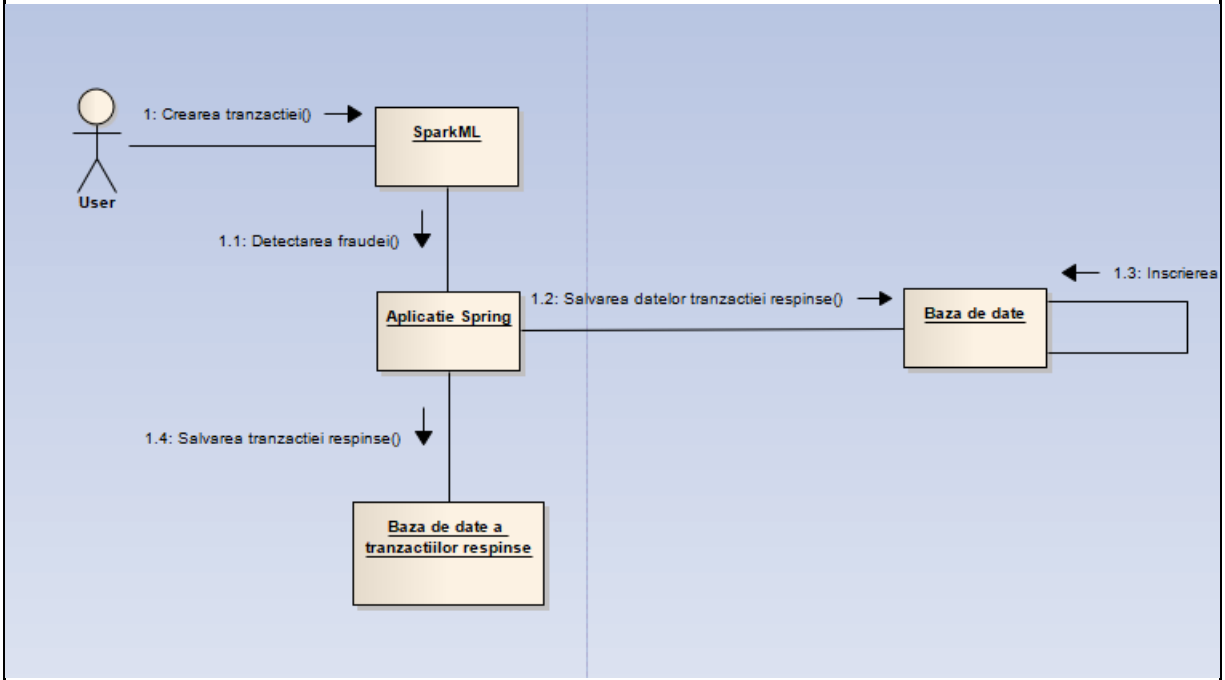


Figura 8 – Diagrama de colaborare a fluxului unei tranzactii respinsa

Diagrama de colaborare a fluxului unei tranzactii respinsa ilustreaza componentele si tipul de interactiuni intre ele la fel ca diagrama de colaborare a fluxului unei tranzactii acceptate ,totusi apare unobiect nou “baza de date a tranzactiilor response”.

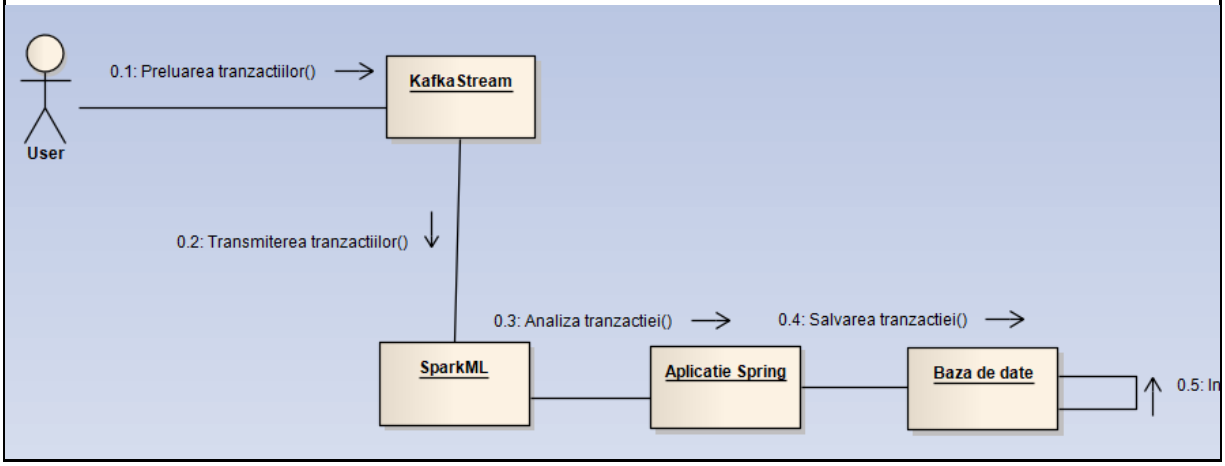


Figura 9 – Diagrama de colaborare a fluxului unei tranzactii

Diagrama de colaborare a fluxului unei tranzactii apare un obiect nou “Kafka Stream” care va face un stream de tranzactii dintr-un Kafka topic.

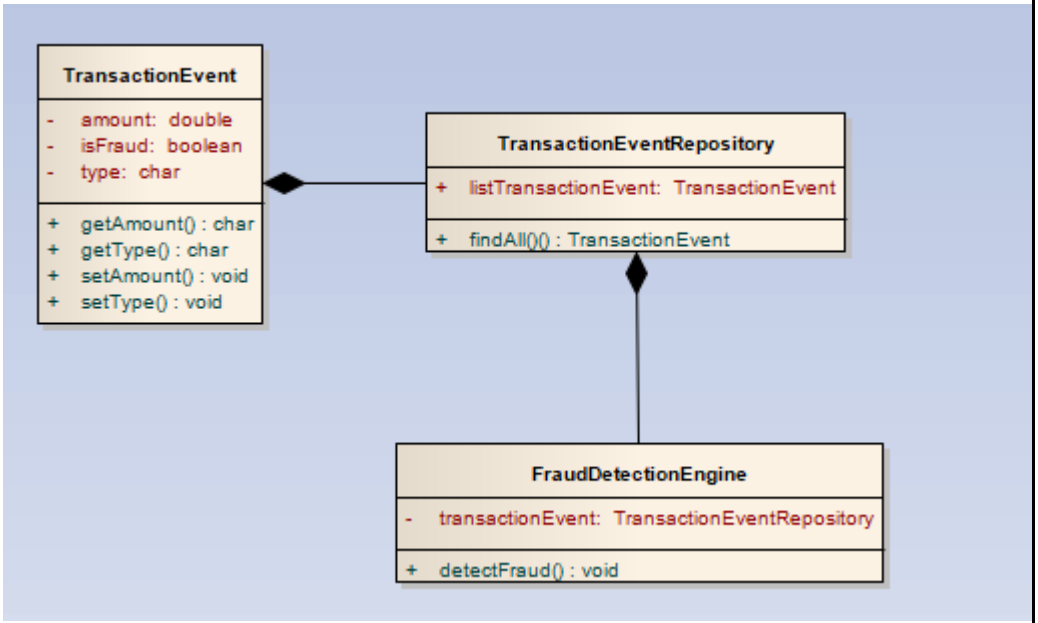


Figura 10 – Diagrama de clase a
aplicatiei de tipSparkML

Diagrama de clase a aplicatiei de tip SparkML ne arata relatiile intre cele 3 clase care sunt cele maiimportante in aplicatia ce detecteaza si analizeaza frauda. TransactionEvent este o clasa de tip POJO careeste serializata dintr-un stream pentru a fi data spre FraudDetectionEngine care este baza aplicatiei , anume aici se identifica estimarea daca o tranzactie este sau nu cu frauda.

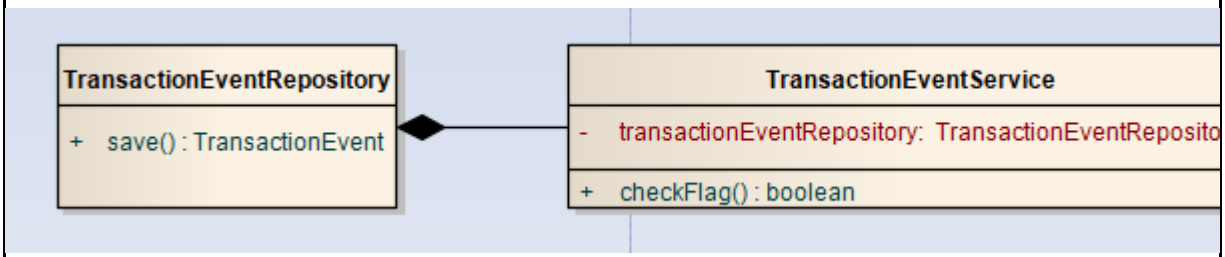


Figura 11 - Diagrama de clase a aplicatiei unde se salveaza datele

Diagrama de clase a aplicatiei unde se salveaza datele reprezinta un microserviciu simplu care areca scop indeplinirea functiilor CRUD asupra tranzactiilor procesate.

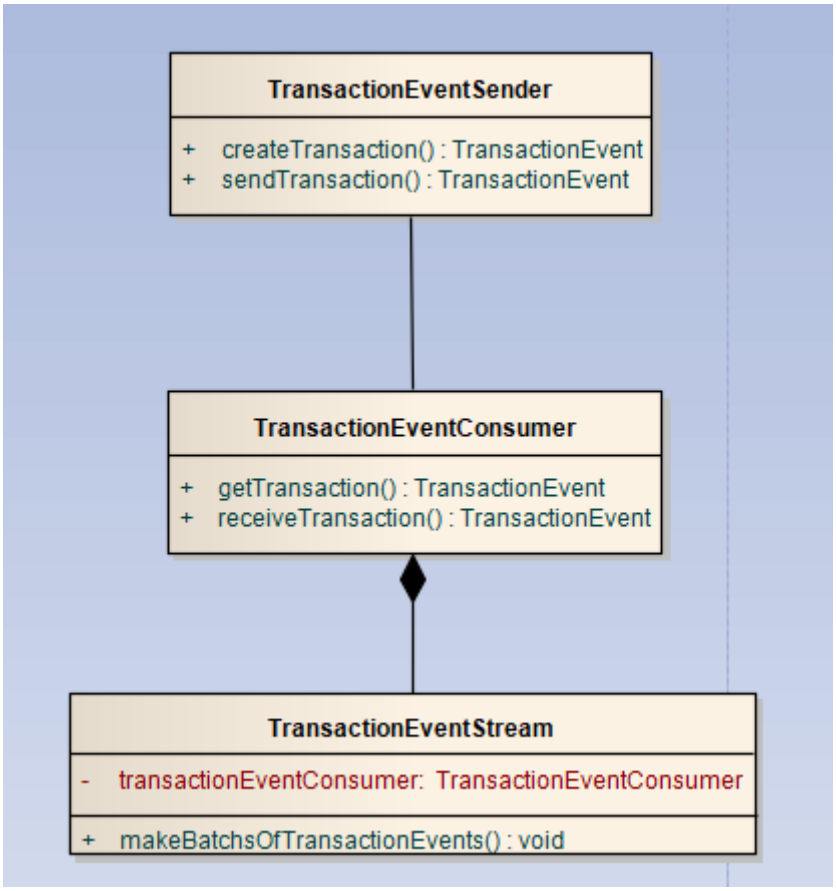


Figura 12 - Diagrama de clase a aplicatiei de streaming

Diagrama de clase a aplicatiei de streaming ne arata anume clasele ce interactioneaza pentru aproduce un flux de tranzactii care la urma lor prin forma de batchuri sunt procesate de SparkML.

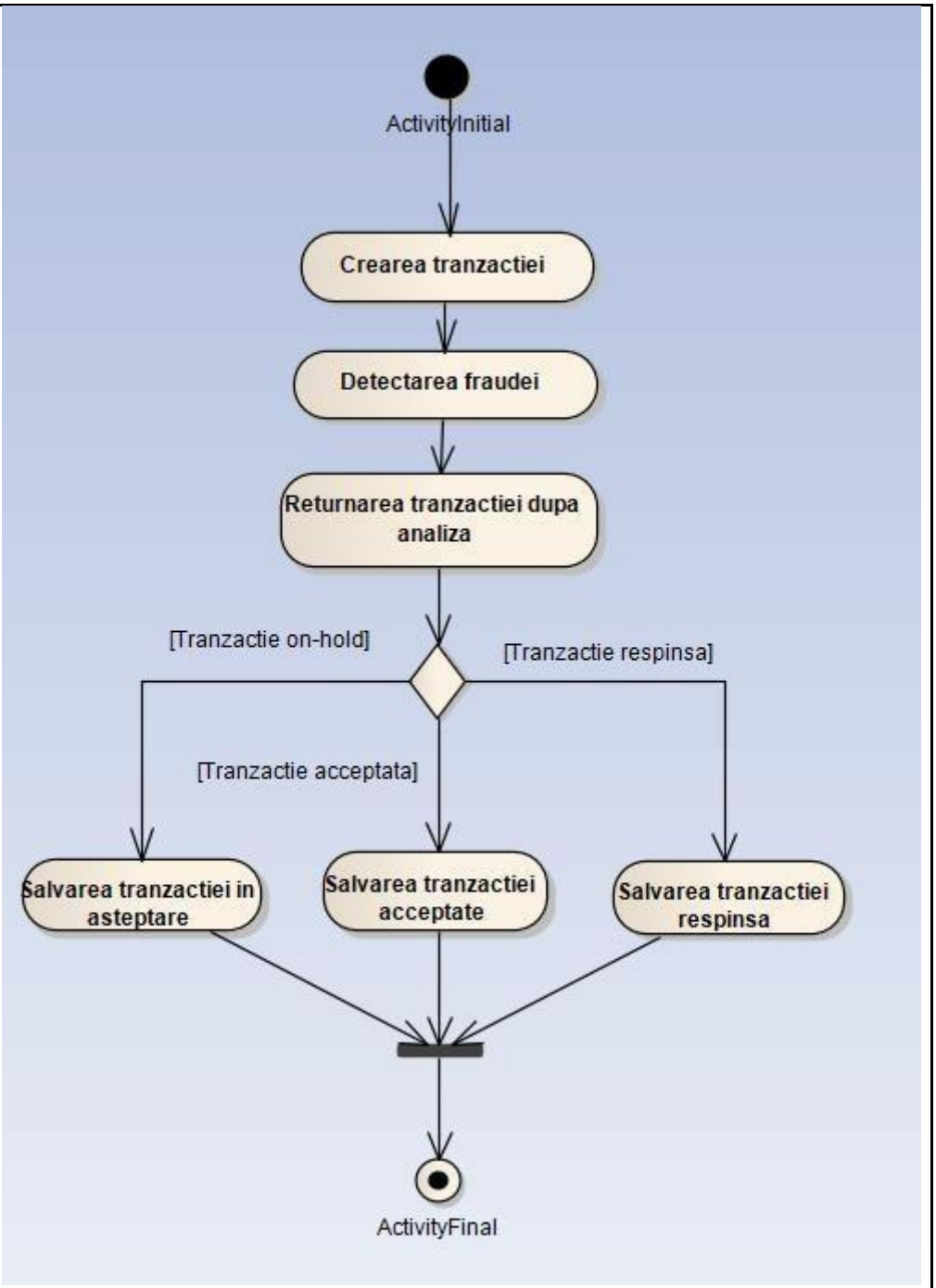


Figura 13 - Diagrama de activitate a microserviciului SparkML

Diagrama de activitate a microserviciului SparkML ne arata cele 3 tipui de scenario care se potintimpla dupa analiza datelor de SparkML.

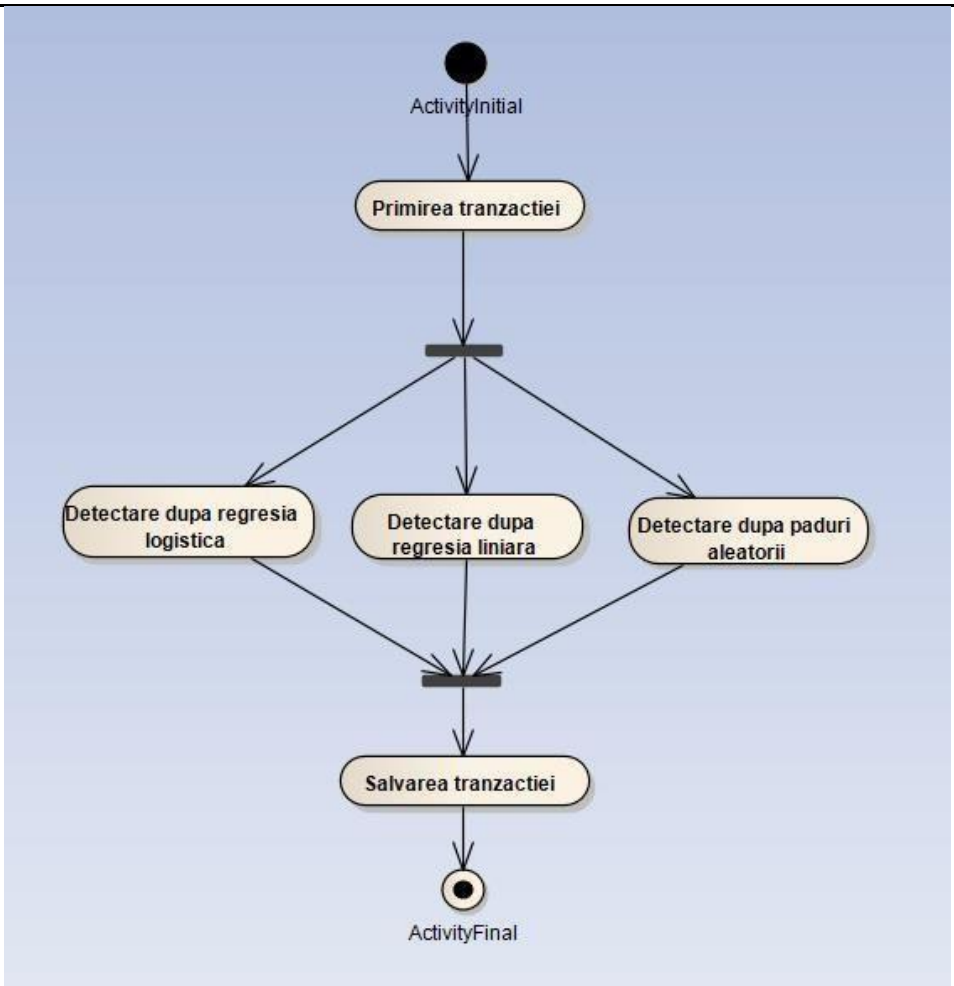


Figura 14 - Diagrama de activitate a modelelor de invatare automata

Diagrama de activitate a modelelor de invatare automata ne ilustreaza cele 3 tipuri de invatare automatacare se vor folosi in proiect. Acel tip de model care va avea o precizie mai mare se va lua ca baza la decizia finala de a respinge sau nu o tranzactie.

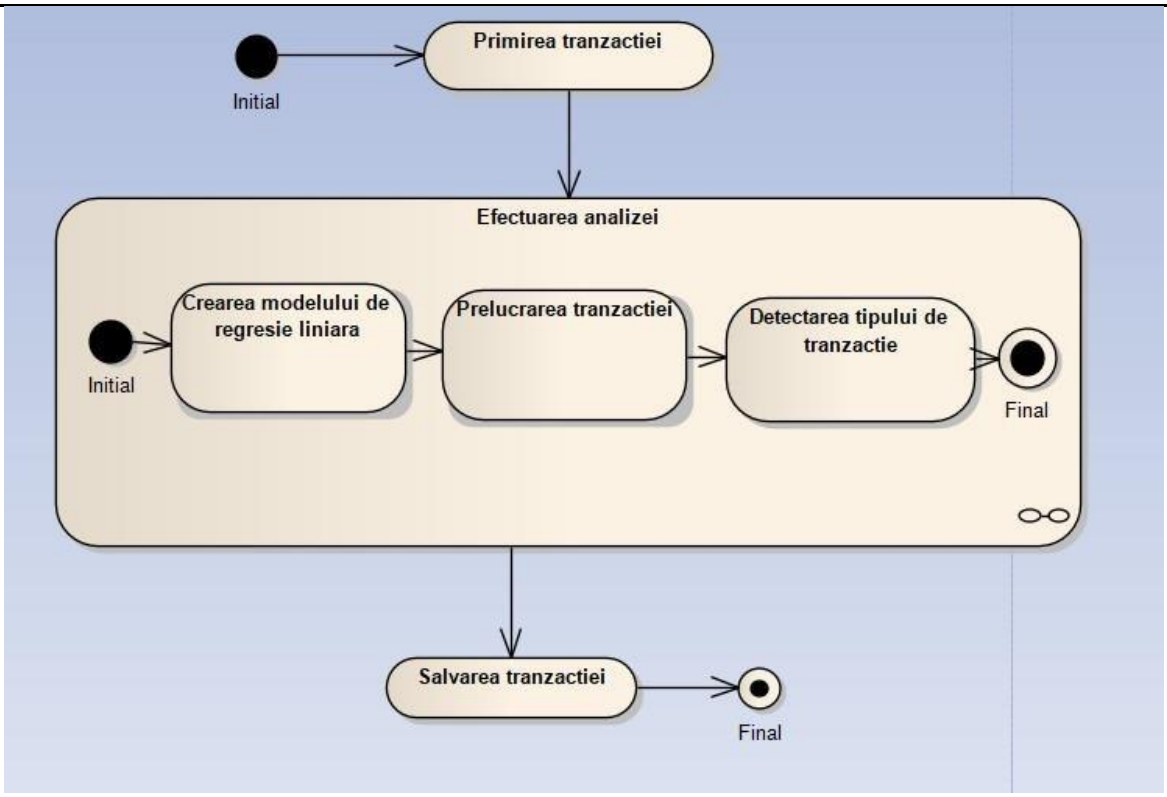


Figura 15 – Diagrama de activitate pentru efectuarea analizei

Diagrama de activitate pentru efectuarea analizei reprezinta operatiunile necesare in cadrul unui pipeline folosit pentru analiza datelor ce vin in SparkML. De exemplu o tranzactie va fi procesata dupamodelul de regresie liniara ca la final sa fie detectata sau nu o frauda.

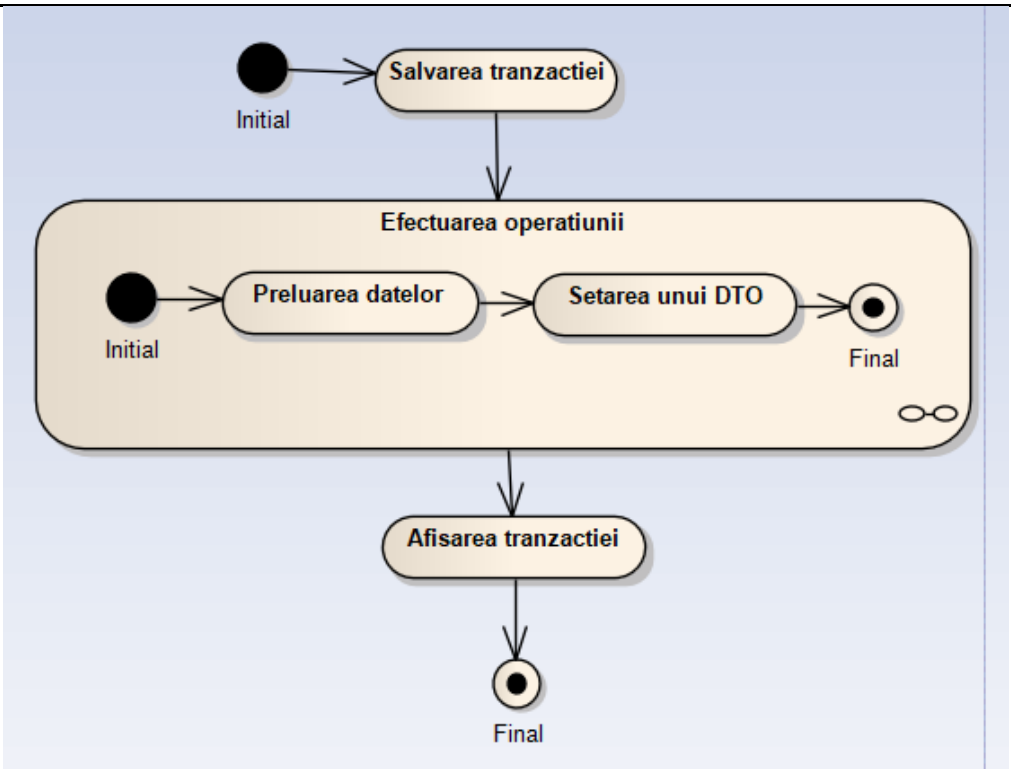


Figura 16 – Diagrama de activitate pentru efectuarea operatiunilor CRUD care se pot intimpla inmicroserviciul de tip Spring

Diagrama de activitate pentru efectuarea operatiunilor CRUD ne arata ca preluarea datelor cit sisetarea unui DTO trebuie sa fie sinhronizate in asa fel ca sa respecte principiul atomic.

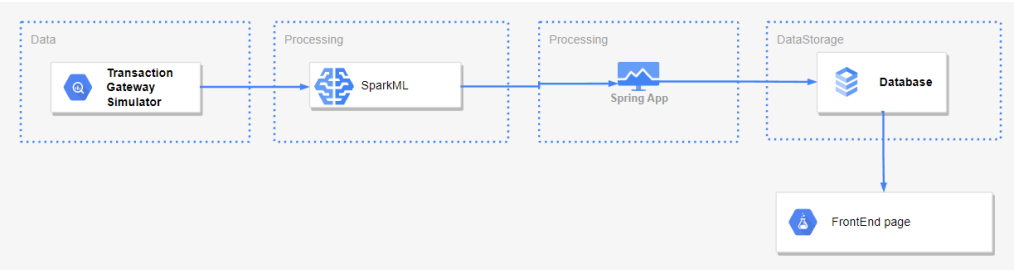


Figura 17 – Fluxul initial al aplicatiei

Dupa figura noi vedem ca toate tranzactiile vor veni din TransactionGatewaySimulator spre SparkML care va detecta tranzactiile fraudate , mai apoi tranzactiile vor fi salvate in baza de date si vor fi afisate intr-o pagina front end pentru a fi analizate manual.

Realizarea sistemului

Sistemul va avea nevoie sa proceseze sute de tranzactii pe secunda, la fel si sa foloseasca un set mare de date pentru antrenarea modelului folosit la învățare automata astfel vom avea nevoie de un message broker care va oferi comunicare intre componente. Brokerul care il vom folosi va fi Kafka. Pentru a procesa datele si a detecta fraudele vom folosi Spark ML. Limbajul de baza al sistemului va fi Java.

Java este un limbaj de programare la nivel înalt, bazat pe clasă, orientat pe obiecte, care este proiectat să aibă cât mai puține dependențe de implementare.

Kafka este folosit în principal pentru a construi conducte de date în flux în timp real și aplicații care se adaptează la fluxurile de date. Combină mesageria, stocarea și procesarea fluxului pentru a permite stocarea și analiza datelor istorice și în timp real.

MLlib este biblioteca de învățare automată (ML) a Spark. Scopul său este de a face învățarea automată practică scalabilă și ușoară. La un nivel înalt, oferă instrumente precum:

- Algoritmi ML: algoritmi comuni de învățare, cum ar fi clasificarea, regresia, gruparea și filtrarea colaborativă
- Caracterizare: extragerea caracteristicilor, transformarea, reducerea dimensionalității și selecția
- Conducte: instrumente pentru construirea, evaluarea și reglarea conductelor ML
- Persistență: salvarea și încărcarea algoritmilor, modelelor și conductelor
- Utilități: algebră liniară, statistică, manipulare a datelor etc.

```java
@Data
public class TransactionEvent implements Serializable {

    private String type;
    private Double amount;
    private Double oldbalanceOrg;
    private Double newbalanceOrig;
    private Double oldbalanceDest;
    private Double newbalanceDest;
    private Integer isFlaggedFraud;
    private String nameDest;

}
```

Figura 19 – Entitatea tranzactiei

Entitatea contine niste fielduri care se folosesc pentru determinarea unicitati a unei tranzacti , la fel se folosesc pentru determinarea fraudei pe baza valorilor ce contin.

```java
VectorAssembler vectorAssembler = new VectorAssembler();
vectorAssembler.setInputCols(new String[]{"dest_diff",
        "orig_diff", "surge", "merchantTrack"});

vectorAssembler.setOutputCol("features");
Dataset<Row> inputData = vectorAssembler.transform(csvData).select( col: "label",  ...cols: "features");

MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator();
evaluator.setMetricName("accuracy");

RandomForestClassifier rfClassifier = new RandomForestClassifier();
rfClassifier.setMaxDepth(3);
RandomForestClassificationModel rfModel = rfClassifier.fit(inputData);

rfModel.save( path: "src/main/resources/model");
```

Figura 20 – Crearea modelului invatarii automate

Ca model de invatare voi folosi padurile aleatorii cu o adincime de 3 a algoritmului. Parametrii pe care se va baza algoritmul sunt diferenta de balans si daca este comerciant sau nu. Modelul este salvat ca sa nu trebuiasca la fiecare deschidere a aplicatiei de rulat antrenamentul pe baza de date deoarece se ia mult timp.

Figura 21 – Furnizorul de evente Kafka

Aceasta clasa o folosesc pentru a furniza evente spre un Kafka topic pe nume “transaction-gateway”. Clasa contine parametrii necesari pentru conexiune.

Pentru testarea sistemului se va folosi unit teste si manual testing.

FIŞA DE ACTIVITATE
Săptâmâna III

Sarcini planificate:

- Se descrie la nivel de cod realizarea funcționalităților

Activităţi desfăşurate, observaţii personale

Toate tranzactiile vor fi produse de un KafkaProducer spre un topic care l-a rindul sau va fi consumat de un Kafka Consumer de tip stream.

```
Dataset<TransactionEvent> csvDataKafka = spark.readStream() DataStreamReader
        .format( source: "kafka")
        .option("kafka.bootstrap.servers", "localhost:9092")
        .option("subscribe", "transaction-gateway3")
        .load() Dataset<Row>
        .selectExpr( ...exprs: "CAST(value AS STRING) as message")
        .select(functions.from_json(functions.col( s: "message"), schema).as( alias: "json"))
        .select( col: "json.*")
        .as(Encoders.bean(TransactionEvent.class));
```

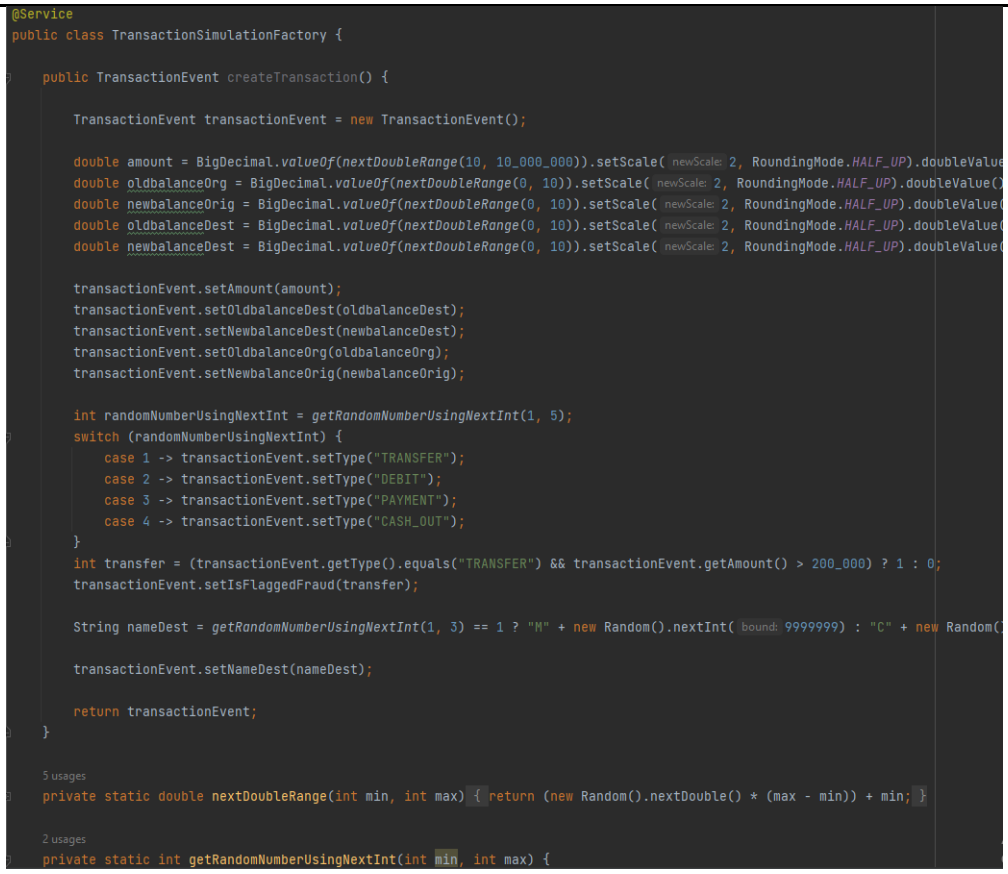
Figura 1 – Kafka Stream Consumer

Deci ne conectam la un topic(transaction-gateway3) cu ajutorul optiunii subscribe pe un anumit server si deserializam toate tranzactiile care vin. Deserializarea se produce prin faptul ca producem un obiect cu ajutorul lui Encoders.bean. La final noi obtinem un dataset de tipul TransactionEvent , anume asupra lui se vor produce toate schimbarile viitoare.

```
Dataset<Row> csvDataUnderBallanced = csvData
        .where( conditionExpr: "label=1");
Dataset<Row> csvDataOverballanced = csvData
        .where( conditionExpr: "label=0").sample( withReplacement: false, fraction: 0.001, seed: 0);
```

Figura 2 – Manipularea de date folosite pentru antrenamentul ML

De obicei datele sunt prea nebalansate ca sa ne ofere rezultatul dorit , de exemplu datele ce le-am folosit pentru antrenamentul ML aveau un procent de 0.2 tranzactii fraudate ceea ce este tare putin ca sa antreneze ML. In asa situatii se folosesc procese de balansare numite supraeșantionare sau subeșantionare. In proiect am ales subeșantionare care se bazeaza ca datele nefraudate sa fie sterse si sa ramana de exemplu 50/50 date nefraudate si fraudate.



```
@Service
public class TransactionSimulationFactory {

    public TransactionEvent createTransaction() {

        TransactionEvent transactionEvent = new TransactionEvent();

        double amount = BigDecimal.valueOf(nextDoubleRange(10, 10_000_000)).setScale( newScale: 2, RoundingMode.HALF_UP).doubleValue
        double oldbalanceOrg = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale( newScale: 2, RoundingMode.HALF_UP).doubleValue()
        double newbalanceOrig = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale( newScale: 2, RoundingMode.HALF_UP).doubleValue(
        double oldbalanceDest = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale( newScale: 2, RoundingMode.HALF_UP).doubleValue(
        double newbalanceDest = BigDecimal.valueOf(nextDoubleRange(0, 10)).setScale( newScale: 2, RoundingMode.HALF_UP).doubleValue(

        transactionEvent.setAmount(amount);
        transactionEvent.setOldbalanceDest(oldbalanceDest);
        transactionEvent.setNewbalanceDest(newbalanceDest);
        transactionEvent.setOldbalanceOrg(oldbalanceOrg);
        transactionEvent.setNewbalanceOrig(newbalanceOrig);

        int randomNumberUsingNextInt = getRandomNumberUsingNextInt(1, 5);
        switch (randomNumberUsingNextInt) {
            case 1 -> transactionEvent.setType("TRANSFER");
            case 2 -> transactionEvent.setType("DEBIT");
            case 3 -> transactionEvent.setType("PAYMENT");
            case 4 -> transactionEvent.setType("CASH_OUT");
        }
        int transfer = (transactionEvent.getType().equals("TRANSFER") && transactionEvent.getAmount() > 200_000) ? 1 : 0;
        transactionEvent.setIsFlaggedFraud(transfer);

        String nameDest = getRandomNumberUsingNextInt(1, 3) == 1 ? "M" + new Random().nextInt( bound: 9999999) : "C" + new Random(

        transactionEvent.setNameDest(nameDest);

        return transactionEvent;
    }

    5 usages
    private static double nextDoubleRange(int min, int max) { return (new Random().nextDouble() * (max - min)) + min; }

    2 usages
    private static int getRandomNumberUsingNextInt(int min, int max) {
```

Figura 3 – Crearea de tranzactii simulate

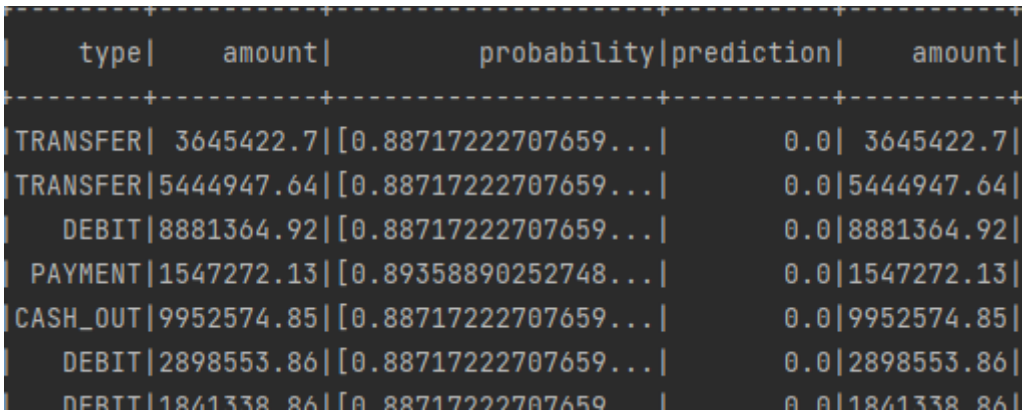
Aceasta figura reprezinta implementarea prin care noi simulam tranzactiile ce vor fi analizate de Fraud-Detection-Core, din cauza faptului ca la etapa de dezvoltare si testare initiala noi nu avem tranzactii adevarate ne vom folosi de aceasta metoda.

```
Dataset<Row> csvDataKafkaForManipulation = csvDataKafka.withColumn( colName: "surge", when(col( s: "amount")
        .$greater( other: 450000), o: 1).otherwise( value: 0));

csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "dest_change", col( s: "newbalanceDest").$minus(
        col( s: "oldbalanceDest")));

csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "dest_txn_diff", when(col( s: "dest_change")
        .$less( other: 0), round(col( s: "amount").plus(col( s: "dest_change")), i: 2))
        .otherwise(round(col( s: "amount").minus(col( s: "dest_change")), i: 2)));

csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "dest_diff", when(col( s: "dest_txn_diff")
        .notEqual( other: 0), o: 1).otherwise( value: 0));

csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "merchantTrack",when(col( s: "nameDest")
        .startsWith("M"), o: 1).otherwise( value: 0));


csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "orig_change", col( s: "newbalanceOrig").$minus(
        col( s: "oldbalanceOrg")));

csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "orig_txn_diff", when(col( s: "orig_change")
        .$less( other: 0), round(col( s: "amount").plus(col( s: "orig_change")), i: 2))
        .otherwise(round(col( s: "amount").minus(col( s: "orig_change")), i: 2)));

csvDataKafkaForManipulation = csvDataKafkaForManipulation.withColumn( colName: "orig_diff", when(col( s: "orig_txn_diff")
        .notEqual( other: 0), o: 1).otherwise( value: 0));
```

Figura 4– Crearea de tranzactii simulate

Deobicei toate datele ce trebuiesc procesate si analizate de un ML sunt neprelucrate si nu vor oferi rezultatul maxim pe care ne dorim. Astfel se fac diferite manipulari asupra datelor

ca sa obtinem rezultatul maxim. De exemplu daca suma tranzactiei depaseste 450000 se adauga un parametru nou.



type	amount	probability	prediction	amount
TRANSFER	3645422.7	[0.88717222707659...	0.0	3645422.7
TRANSFER	5444947.64	[0.88717222707659...	0.0	5444947.64
DEBIT	8881364.92	[0.88717222707659...	0.0	8881364.92
PAYMENT	1547272.13	[0.89358890252748...	0.0	1547272.13
CASH_OUT	9952574.85	[0.88717222707659...	0.0	9952574.85
DEBIT	2898553.86	[0.88717222707659...	0.0	2898553.86
DEBIT	1841338.86	[0.88717222707659...	0.0	1841338.86

Figura 5 – Tranzactii analizate

In figura putem observa ca niste tranzactii au fost analizate cu o probabilitate de 0.88 ca nu sunt fraudate ,0 fiind non-fraud iar 1 fraud. Evident modelul trebuie imbunatatit iar tranzactiile test la fel au nevoie de anumite ajustari.

FIŞA DE ACTIVITATE
Săptâmâna IV

IDENTIFICAREA FRAUDELOR TRANZACȚIILOR BANCARE

A elaborat: st.gr. TI-194
Gumaniuc Alexandru

Analiza domeniului

- Machine Learning
- Fraud Detection
- Natura intrinsec privată a tranzacțiilor

Scopul și obiectivele propuse

- Sistemul trebuie să identifice tranzacțiile frauduloase în timp real și să prevină tranzacțiile frauduloase ulterioare;
- Alegerera unei tehnici de analiza a datelor : supraeșantionarea și subeșantionarea;
- Crearea aplicatiei de streaming a tranzactiilor de tip Kafka ;

MINISTERUL EDUCAŢIEI ŞI CERCETĂRII AL REPUBLICII MOLDOVA

UNIVERSITATEA TEHNICĂ A MOLDOVEI

CAIETUL

STAGIULUI DE PRACTICĂ

PENTRU STUDENȚII STUDIILOR SUPERIOARE DE LICENȚĂ – CICLULUI I

Stagiul de practică *de licență*
tipul stagiului de practică

A studentului (ei) *Gumaniuc Alexandru*
numele, prenumele

Facultatea *Calculatoare, Informatică și Microelectronică*

Programul de studii *Tehnologia informației*

Ciclul *I* Anul de studii *IV* Grupa *TI-194*

Locul stagiului de practică Endava
denumirea unităţii economice

Conducătorul stagiului de practică
de la UTM *asist.univ. Cojocaru Svetlana*
funcţia, numele, prenumele

Conducătorul stagiului de practică
de la unitatea economică
funcţia, numele, prenumele

Chişinău, 2023

I. CAIETUL DE SARCINI

Nr. crt.	Tematica lucrărilor preconizate	Termene planificate	
		început/ sfârşit	**numărul de zile**
1	*Analiza domeniului de studii și Modelarea și proiectarea sistemul informatic*	*30.01.2023-03.02.2023*	*5 zile*
2	*Realizarea sistemului*	*06.02.2023-10.02.2023*	*5 zile*
3	*Realizarea sistemului*	*20.02.2023-24.02.2023*	*5 zile*
4	*Realizarea raportului și prezentării*	*27.02.2023-03.03.2023*	*5 zile*

II. SARCINA INDIVIDUALĂ
(SE INDICĂ TEMA PROIECTULUI DE LICENȚĂ)

Tema: Identificarea fraudelor tranzactiilor bancare

Nr. crt.	Conținutul concis al sarcinilor individuale	Menţiune cu privire la realizare
1	*Analiza domeniului de studii*	• *Se descrie domeniul din care face parte proiectul creat (tehnologii informaționale). Ce tip de aplicație (web, mobile...etc). În ce domeniu se va aplica proiectul (învățământ, sfera serviciilor.....etc).* • *Se va argumenta importanța creării unui astfel de proiect.* • *Se va face o descriere a cel puțin 3 sisteme deja existente (sau sisteme care au ceva comun din punct de vedere funcțional cu ce se va face în proiect). Se va face o comparare a sistemelor descrise.* • *În baza comparării se va scri scopul și obiectivele, cerințele sistemului. Determinarea cerințelor funcționale, nefuncționale. Detrminarea funcționalului sistemului și subsistemelor. Se va descrie cît mai detaliat caietul de sarcini a proiectului.*
2	*Modelarea și proiectarea sistemul informatic*	• *Descrierea comportamentală a sistemului (imaginea generală asupra sistemului; modelarea vizuală a fluxurilor; stările de tranzacție a sistemului (Statechart Diagram); descrierea scenariilor de utilizare a aplicaţiei (Sequence Diagram); fluxurile de mesaje şi legăturile dintre componentele sistemului (Collaboration Diagram).* • *Descrierea structurală a sistemului (descrierea structurii statice a sistemului; relatiile de dependență între componentele; modelarea echipamentelor mediului de implementare.*
3	*Realizarea sistemului*	• *Se face o introducere în ce s-a folosit pentru realizarea sarcinilor. Se descrie pe scurt tehnologiile folosite, limbaje de programare utilizate, instrumente, algoritmi, șabloane...etc , totul ce s-a folosit la realizarea proiectului.* • *Se descrie la nivel de cod realizarea funcționalităților* • *Se descrie testarea sistemului*
4	*Realizarea raportului*	• *De structurat după sarcini individuale* • *De redactat confor standardelor de redactare*
5	*Realizarea prezentării*	• *Slide cu denumirea temei* • *Slide cu Analiza domeniului* • *Slide cu Scopul și Obiectivele proiectului* • *Slide cu Imaginea generală a sistemului* • *Slide cu Descrierea structurii statice a sistemului* • *Slide cu Instrumente, tehnologii utilizate pentru realizarea sustemului* • *Slide cu Video- prezentarea aplicației*

Conducătorul _____ / *Cojocaru Svetlana* /
semnătura *numele şi prenumele*