

Documentatie proiect individual:

Aplicatie Bancara

Ciprian Teletin
Departamentul de Informatică
Facultatea de Matematică și Informatică
Universitatea de Vest
Timișoara, România
Email: ciprian.teletin99@e-uvt.ro

9 ianuarie 2020

Cuprins

1	Introducere	4
1.1	Motivație	4
1.2	Prezentare generala a aplicatiei	5
2	Prezentare formală a interfetelor grafice si a butoanelor implementate	5
3	Descrierea procesului de implementare	12
3.1	LogIn	12
3.2	SignUp	14
3.3	ChooseAccountInterface	15
3.4	Application	16
3.5	SecondApplication	18
3.6	Settings	20
3.7	Bug-uri intalnite	22
4	Diagrama UML	23
5	Manual de utilizare al aplicatiei	26
5.1	Cerinte de utilizare	26
5.2	Creeare cont	26
5.3	Completarea datelor de card	27
5.4	Utilizarea aplicatiei	27
6	Bibliografie	27

1 Introducere

Java este un limbaj de programare orientat-obiect, puternic tipizat, conceput de catre James Gosling la Sun Microsystems(acum filiala Oracle) la inceputul anilor 90', fiind lansat in 1995. Cele mai multe aplicatii distribuite sunt scrise in Java, iar noile evolutii tehnologice permit utilizarea sa si pe dispozitive mobile gen telefon, agenda electronica etc. In felul acesta se creeaza o platforma unica, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Aceasta este utilizat in prezent cu succes si pentru programarea aplicatiilor destinate intranet-urilor.

Limbajul imprumuta o mare parte din sintaxa de C si C++, dar are un model al obiectelor mai simplu si prezinta mai putine facilitati de nivel jos. Un program Java compilat, corect scris, poate fi rulat fara modificarile pe orice platforma ce are instalata o masina virtuala Java(Java Virtual Machine-JVM). Acest nivel de portabilitate(inexistent pentru limbaje mai vechi precum C) este posibil deoarece sursele Java sunt compilate intr-un format standard numit cod de octeti(byte-code) care este intermediar intre codul masina(dependent de tipul calculatorului) si codul sursa.

Masina virtuala Java este mediul in care se executa programele Java. In prezent, exista mai multi furnizori de JVM, printre care Oracle, IBM, BEA, FSF. In 2006, Sun a anuntat ca face disponibila varianta sa de JVM ca open-source.

1.1 Motivatie

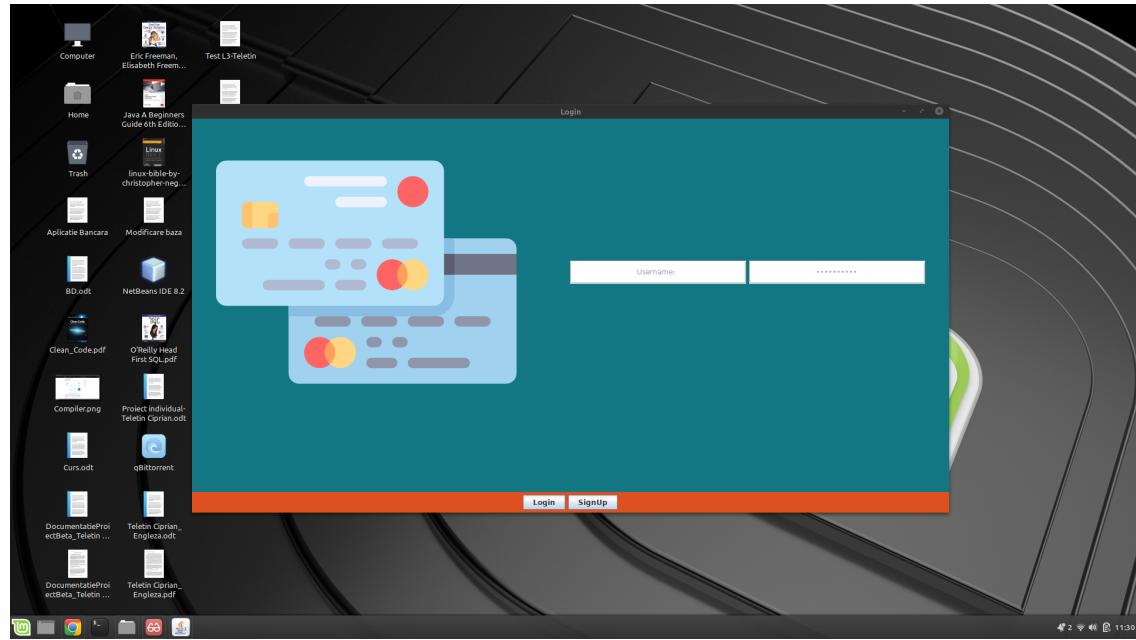
Tema aleasa pentru acest proiect individual are la baza aplicatiile bancare ce permit diferite facilitati clientilor, chiar de acasa(Home banking). Motivul principal al alegerii acestei teme consta in faptul ca tehnologia ne confera un factor de comoditate prin intermediul realizarii diferitelor procese fara interventie umana excesiva. Prin intermediul aplicatiilor de home banking, clientul poate realiza diverse plati(facturi, produse online), verificarea soldului curent, transferuri bancare si chiar si conversii valutare in diferite monede internationale, totul la cateva apasari de butoane, din orice locatie s-ar afla. Astfel, din dorinta de a imi aplica cunostintele in limbajul Java(cat si extinderea acestora prin folosirea unor API-uri, precum JDBC, Jsoup) si din dorinta de a crea ceva folositor omului de rand, am ales sa creez o aplicatie care simuleaza o aplicatie reala de Home Banking.

1.2 Prezentare generala a aplicatiei

Aplicatia bancara este alcatuita din 6 interfete grafice principale, de unde avem acces catre butoanele acestora ce confera functionalitate aplicatiei. Mare parte dintre butoane contin si ele la randul lor cate o interfata grafica ce are ca si scop oferirea unui mediu comod si simplu prin care utilizatorul sa poata realiza ceea ce isi doreste. Pentru utilizarea aplicatiei in sine, utilizatorul este nevoit sa isi creeze un cont in cadrul caruia va introduce date cu caracter personal(CNP, numar de telefon, adresa de e-mail etc), cat si date referitoare la cardul sau cardurile bancare pe care le detine(precum numar card, cvv si data expirarii a acestuia).Procesul de creare a aplicatiei, cat si modul de utilizare a aplicatiei vor fi descrise in capitolul *Manual de utilizare*, subcapitolele *Creare cont*, respectiv *Utilizarea aplicatiei*. Descrierea aplicatiei din punctul de vedere al unui programator se va face in cadrul capitolului *Descrierea procesului de implementare*.

2 Prezentare formală a interfetelor grafice si a butoanelor implementate

Cele 6 interfete principale(LogIn interface, SignUp interface, ChooseAccount interface, Application interface, SecondApplication interface si Settings interface, cat si celelalte interfete din cadrul butoanelor sunt realizate prin cadrul metodelor si a utilizarii obiectelor din pachetele javax.swing.* si java.awt.* . La baza acestora se afla cate un layout manager cat si diferite metode folosite pentru stilizarea design-ului aplicatiei (Documentatie aici).



(LogIn interface)

Prezentare functionalitati butoane:

LogIn: Buton ce are ca si scop verificarea corectitudinii datelor introduse cat si existenta acestora in cadrul bazei de date. Daca aceste criterii sunt verificate cu success, butonul va face trimitera catre Application interface.

SignUp: Buton ce va face trimitera catre interfata de inregistrare a contului utilizator, interfata prezentata mai jos.

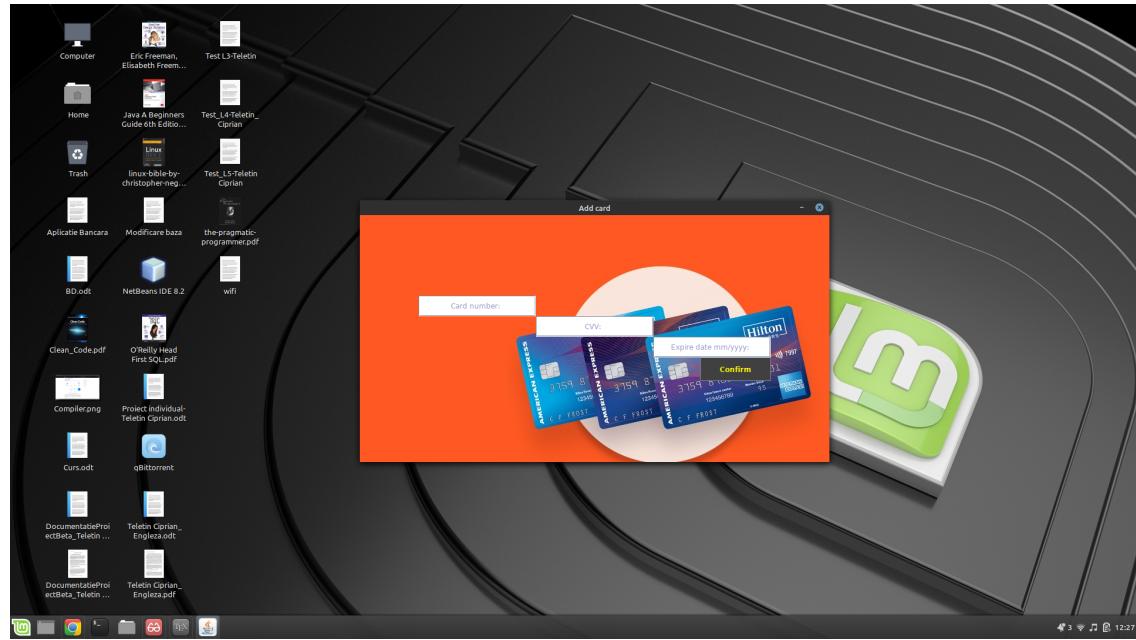


(SignUp interface)

Prezentare functionalitati butoane:

Confirm: Buton ce are ca si scop verificarea corectitudinii datelor introduse in campurile de mai sus(verificarea formatului fiecarui camp, inexistentă datelor unice personale in baza de date, verificarea daca campurile au fost sau nu completate), salvarea acestora in baza de date cat si trimiterea catre urmatoarea interfata de inregistrare, ChooseAccount.

Reset: Buton ce are ca si rol resetarea la starea default a fiecarei zone de text din cadrul interfetei. Resetarea se va face in cazul in care utilizatorul confirma dorinta acestui lucru.



(ChooseAccount interface)

Prezentarea functionalitate buton:

Confirm: Butonul are ca si scop verificarea corectitudinii datelor introduse(daca respecta formatul de date introdus unui card utilizator), salvarea acestora in baza de date, asociate contului creat anterior(la sign up interface), cat si trimiterea inapoi catre interfata de logare.



(Application interface)

Prezentare facilitati butoane:

PlataFactura: Buton ce are ca si scop permiterea utilizatorului sa plateasca facturi spre diversi furnizori. In momentul apasarii butonului, se va deschide o interfata grafica unde vom introduce numele furnizorului si pretul facturii, dupa care se apasa pay; In cazul in care suma si limita contului sunt suficiente pentru efectuarea platii, aceasta se va realiza cu succes.

PlataOnline: Buton ce are ca si rol conferirea posibilitatii de cumparare a diferitelor produse online, din cadrul site-urilo Emag, PCGarage si Cel.ro; in momentul apasarii butonului, se va deschide o interfata grafica unde se va introduce link-ul produsului, iar pe baza acelui link, aplicatia va detecta numele si pretul produsului, moment in care se poate face plata acestuia.

Transfer: Buton ce permite utilizatorului sa transmita o suma de bani catre un alt utilizator al aplicatiei, pe baza iban-ului contului in care se doreste sa se faca transferul, cat si a numelui de utilizator. Transferul se va face doar daca exista suma respectiva pe card iar limita nu este depasita.

TransferPers: Buton ce are ca si rol transmiterea de bani intre carduri personale(in cazul existentei acestora) pe baza iban-ului contului. La fel ca in cazul butonului **Transfer**, banii vor fi transmisi doar daca suma curenta permite asta. Diferentele se regasesc doar in cazul limitei de transfer si a

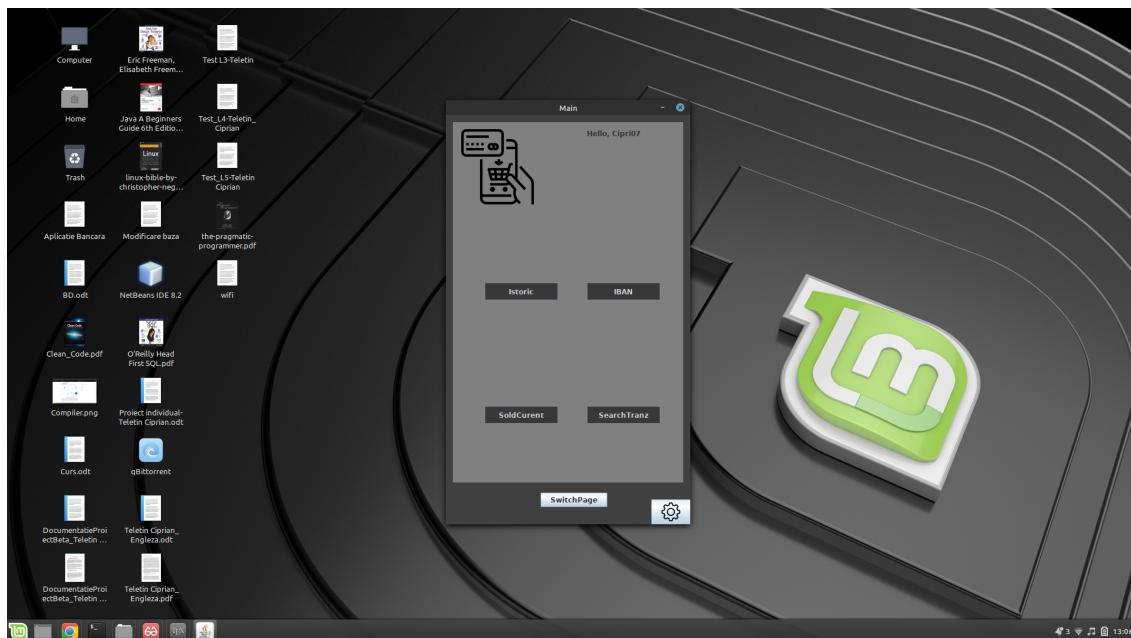
comisioanelor, care nu sunt aplicabile aici.

Conversie: Buton ce permite conversia sumei curente de pe card in alta moneda internationala(Euro,Dolar,Lira) pe baza taxelor de conversie din ziua respectiva.

CursValutar: Buton ce va deschide o interfata grafica ce ne va prezenta cursul valutar din ziua curenta, cursul valutar fiind preluat online(prin intermediul unui scrapper).

SwitchPage: Buton ce va face trecerea catre a doua interfata bazata pe functionalitatea utilizator, SecondApplication.

Rotita setari: Buton ce ne face trecerea catre interfata destinata setarilor, Settings.



(SecondApplication interface)

Prezentare facilitati butoane:

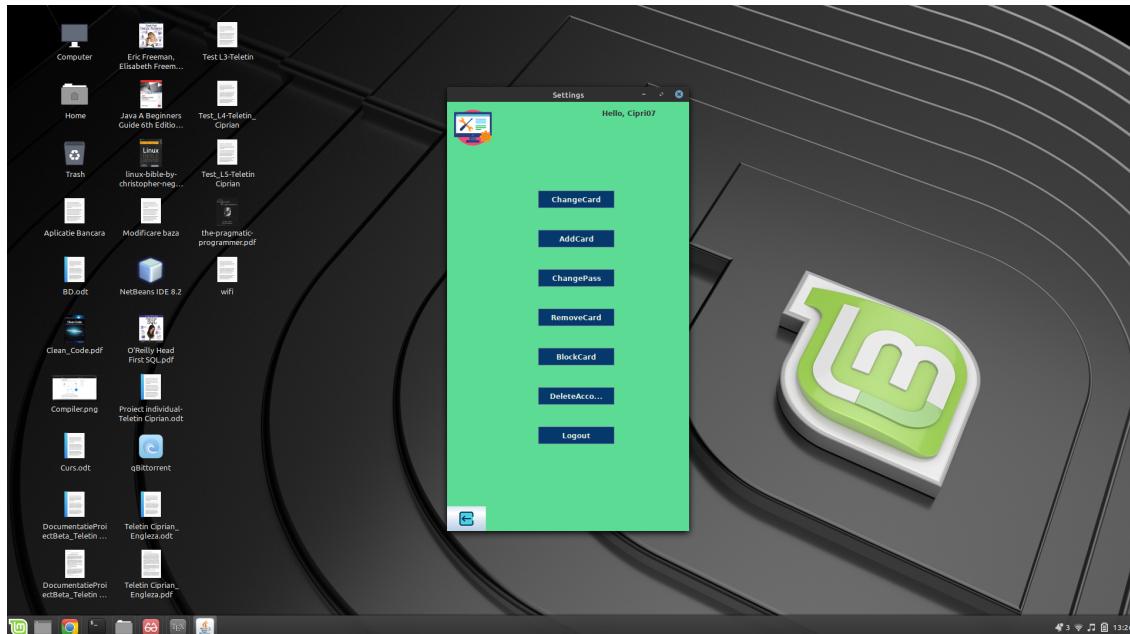
Istoric: Buton ce va deschide o interfata ce va avea ca si rol listarea pe ecran(in cadrul unei zone de text) a unui numar de tranzactii ales de catre utilizator.

IBAN: Buton ce va deschide o interfata unde ne vor fi prezentate date specifice contului, cat si posibilitatea de a salva pe calculator in format pdf aceste date, in cazul dorintei de a printa datele.

SoldCurrent: Buton ce ne va deschide o interfata in cadrul caruia vom putea vizualiza soldul curent, moneda contului, comisioanele aplicate, limita de transfer etc.

SearchTranz: Buton ce ne va lista toate tranzactiile ce respecta anumite criterii prestabilite de utilizator(in functie de nume, pret, data calendaristica etc).

Butoanele **SwitchPage** si **Rotita settings** discutate in cadrul interfetei anterioare.



(Settings interface)

Prezentare facilitati butoane:

ChangeCard: Buton ce face trecerea catre un alt card(la alegerea utilizatorului) in cazul existentei a multiple carduri bancare.

AddCard: Buton ce permite adaugarea unui nou card in aplicatie, asociat contului utilizator curent. Fiecare utilizator poate sa aiba in contul sau maxim 3 carduri, respectiv 1 card daca este student.

ChangePassword: Buton ce schimba si retine in baza de date noua parola introdusa.

RemoveCard: Buton ce permite eliminarea uneia dintre cardurile asociate contului introdus

BlockCard: Buton ce are ca si rol blocarea cardului curent, pe baza introducerii parolei. In momentul blocarii, butoanele din cadrul interfetelor Application si SecondApplication sunt blocate, neputand fi utilizate pana la deblocarea cardului.

DeleteAccount: Buton care stergerea contului si a tuturor datelor si activitatilor asociate acestuia din baza de date a aplicatiei(tranzactii etc)

Logout: Buton ce ne scoate din contul curent si ne trimita inapoi la interfata LogIn.

Back Button: Ce are ca si rol deschiderea interfetei anterior utilizate(cea din care s-a facut trecerea spre interfata Settings)

3 Descrierea procesului de implementare

In cele ce urmeaza voi prezenta din cadrul fiecarei interfete principale cateva dintre implementarile realizate in cadrul butoanelor descrise mai sus, librariile si tehnologiile utilizate pentru realizarea acestei aplicatii, diverse bug-uri intampinate pe parcursul crearii acestei aplicatii cat si rezolvările acestora, acolo unde a fost posibil acest lucru. Din cadrul fiecarei interfete voi descrie doar butoanele in cadrul careia au fost utilizate anumite api-uri sau care sunt mai complicate din punct de vedere logistic.

Clasa din care are loc deschiderea aplicatiei(metoda main) este aflata in clasa Launch, de unde prin SwingUtilities.invokeLater se apeleaza metoda start al clasei Login.

3.1 LogIn

Reprezinta punctul de pornire al aplicatiei, de unde avem 2 posibilitati: sa ne inregistram sau sa ne conectam. Am ales sa descriu in cadrul acestui subcapitol butonul de logIn, deoarece cel de signUp doar face trimiterea spre o alta interfata(apel *SwingUtilities.invokeLater(Runnable)*).

In realizarea butonului de login, am folosit API-ul pentru conectarea catre o baza de date, prin driverul JDBC, API prezent in libraria java.sql.* (Pentru documentatie, click aici); Pentru realizarea conexiunii, am folosit clasa DriverManager, metoda getConnection in cadrul caruia trimit ca parametrii URL-ul bazei de date, numele si parola de utilizator.

In cadrul interfetei de mai jos este prezentat url-ul catre baza de date locala, ce accepta si caractere unicode, folosite in procesul de criptare a

parolei.

```
package com;

public interface URL {
    //URL of my connection; this interface has only
    //this purpose, to export this to all packages;
    String url="jdbc:mysql://localhost:3306/
bank_application?useUnicode=yes&characterEncoding=UTF-8";
    //can accept utf characters
}
```

Cum functioneaza butonul?

In cadrul acestuia este adaugat un actionListener, care in momentul aparii acestuia executa metoda atasata; In cadrul metodei, se verifica existenta si corectitudinea datelor prin intermediul unui automat finit, cu diverse stari, fiecare dintre aceste stari reprezentand un pattern in care se poate incadra litera curenta din sirul primit sau un pattern de tranzitie spre o alta stare. In frame-ul de mai jos este prezentata clasa FiniteAutomata, ce are scopul mentionat anterior;

```
package com.automata;

public class FiniteAutomata {
    private Transition transition;
    private String verifyData;

    public FiniteAutomata(String verifyData, State... states){
        transition=new Transition(states);
        this.verifyData=verifyData;
    }

    public boolean validateData(){
        int i=0;
        for(;i<verifyData.length();++i){
            if(transition.nextState(verifyData.charAt(i)))
                continue; //Do nothing at all;
        }
    }
}
```

```

    else if (!transition.verifyCurrentState(
        verifyData.charAt(i)))
        return false;
    }
    return (i==verifyData.length() &&
        transition.currentState());
}

```

Imediat dupa procesul de verificare a datelor, metoda realizeaza conexiunea la baza de date din cadrul careia extrage numele de utilizator corespondent celui introdus(daca acesta exista, din tabela **users**) si parola asociata utilizatorului(din tabela **passwords**). Dupa extragerea datelor din tabela, se decripteaza parola pastrata in baza de date si se verifica cu cea actuala. Daca cele doua parole corespund in urma decriptarii, atunci se va verifica numarul de carduri asociate contului si se va oferi posibilitatea de alegere a acestuia in cazul existentei a 2 sau 3 carduri(prin interfetele TwoCards si ThreeCards). Dupa alegerea cardului, se actualizeaza baza de date pentru a marca cardul ales si se va deschide interfata Application.

3.2 SignUp

Interfata prezinta 9 campuri de text in care utilizatorul este rugat sa introduca diverse date, conform unui format, detalii despre acest lucru regasindu-se in capitolul *Manual de utilizare*. Dintre cele doua butoane disponibile, am ales spre prezentare butonul de Confirm, deoarece cel de Reset sterge datele din campuri pe baza raspunsului utilizatorului.

Cum functioneaza butonul? Aceasta verifica corectitudinea datelor prin intermediul automatelor finite(unul specific pentru fiecare camp in parte). In cazul in care unul dintre campuri este invalid, va afisa un mesaj de eroare spre campul corespunzator erorii.

```

boolean checkUsername(){
    State s0=new State("","[A-Z]");
    State s1=new State("[A-Z]","[a-z]");
    State s2=new State("[a-z]","[0-9]");
    State s3=new State("[0-9]",true);
    FiniteAutomata automate=new
    FiniteAutomata(username.getText(),s0,s1,s2,s3);
}

```

```
    return automate.validateData();}
```

In cazul in care datele introduse respecta formatul specific, se va trece la procesul de verificare a existentei datelor unice in cadrul bazei de date(precum username,email,CNP si numar de telefon, date ce trebuie sa fie unice). In cazul in care datele introduse nu exista in baza de date, vor fi introduse in tabele specifice tipului de informatii pe care o reprezinta (**users**,**passwords**,**info** si **adress**), dupa care se face trecerea la ChooseAccount interface.

3.3 ChooseAccountInterface

In cadrul acestei interfete, utilizatorul va introduce datele legate de cardul bancar al acestuia. In cadrul ei se regaseste un singur buton, **Confirm**; Aceasta interfata este utilizata si in cadrul butonului **AddCard** din cadrul interfetei *Settings*.

Cum functioneaza butonul? Se verifica datele introduse pe baza automatelor finite;Se verifica numarul de carduri asociate contului, pentru a nu se depasi limita admisa de carduri per cont. In cazul in care numarul de carduri limita nu este depasit, se continua prin verificarea existentei datelor in baza de date(a numarului de cont si a cvv-ului, ce trebuie sa fie unice). In cazul unicitatii datelor, acestea vor fi introduse in baza de date, date specifice cardului(moneda,suma card,data in care se vireaza salariul, salariul de baza, IBAN etc) vor fi generate automat si introduce in tabele specifice din cadrul bazei de date iar controlul va fi transferat interfetei de logare(in cazul in care nu exista card asociat) respectiv catre interfata Settings. In cazul in care se produce vreo eroare in legatura cu datele introduse, se vor afisa mesaje specifice spre solutarea problemei;

O alta functionalitate a interfetei este aceea ca in cazul apasarii butonului de x in timpul procesului de inregistrare(nu si la cel de logare), datele introduse anterior in interfata **SignUp** vor fi eliminate din baza de date, deoarece nu au un card asociat. Mai jos se regaseste o bucată de cod din cadrul listener-ului pentru inchiderea aplicatiei.

```

if (! rs . next () && ! noCards) {
    pst = connection . prepareStatement
    ("DELETE FROM adress WHERE user=?");
    pst . setString (1, username);
    pst . executeUpdate ();
    pst = connection . prepareStatement
    ("DELETE FROM info WHERE user=?");
    pst . setString (1, username);
    pst . executeUpdate ();
    pst = connection . prepareStatement
    ("DELETE FROM passwords WHERE user=?");
    pst . setString (1, username);
    pst . executeUpdate ();
    pst = connection . prepareStatement
    ("DELETE FROM users WHERE user=?");
    pst . setString (1, username);
    pst . executeUpdate ();
}

```

3.4 Application

Este interfata care este deschisa imediat dupa ce logarea a avut success; In momentul deschiderii interfetei, acesteia i se ataseaza un obiect de tip Card, ce preia datele specifice cardului curent si pe care se vor efectua diversele operatii. Interfata are in componenta sa 8 butoane, dintre care eu voi prezenta **PlataOnline** si **Transfer**.

PlataOnline:

Acesta are la baza un API din cadrul librariei org.jsoup (Documentatie aici) ce primeste o pagina web, se conecteaza la acesta, extrage codul HTML(Dom-ul) dupa care ne permite parsarea acestuia si revendicarea datelor de care noi avem nevoie. Pentru acest buton, am ales sa extrag date(numele si pretul produsului) de pe 3 site-uri romanesti: *Emag*, *PCGarage* si *Cel.ro*; Clasele folosite pentru parsarea si extragerea informatiilor dorite au la baza clasa abstracta Scrapper, descrisa mai jos:

```

public abstract class Scrapper {
    final protected String url;

```

```

protected String price;
protected String productName;
protected String valueType;
protected int prc;

public Scrapper(String url){
    this.url=url;
    this.price=null;
    this.productName=null;
    this.valueType=null;
    this.prc=0;
}

public abstract void scrape();

public String getProductName(){
    return productName;
}

public int getPret(){
    return prc;
}

public String getMoneda(){
    return valueType;
}
}

```

Cum functioneaza butonul?

In momentul apasarii acestuia, afiseaza un mesaj instructional dupa care deschide o noua interfata: **Online**, ce contine un text field de introducere a link-ului produsului si un buton **pay**. In momentul apasarii butonului pay, urmatorii pasi sunt executati: se verifica link-ul si se creeaza obiect corespunzator acestuia, dupa care se afiseaza si retine numele produsului si pretul acestuia. Se continua prin verificarea daca suma de plata(cu tot cu comisioane, in moneda curenta a cardului) poate fi achitata si daca limita nu este depasita de aceasta plata. In cazul in care totul este in regula, se actualizeaza suma si limita de pe card prin metoda updateDB(specifica

obiectului de tip card) si se introduce in baza de date tranzactia respectiva.

Transfer:

In momentul apasarii butonului de transfer, se lanseaza interfata **Transfer**, in cadrul careia regasim 3 zone de text, in care se vor introduce IBAN-ul contului in care dorim sa transferam, suma de transfer si numele de utilizator catre care efectuam transferul, cat si un buton **Transfer**; in momentul apasarii butonului de transfer, se va verifica daca iban-ul corespunde numelui de utilizator introdus si daca transferul se va putea efectua in urma comisioanelor aplicate. In cazul in care transferul este efectuat cu success, se vor adauga cele doua tranzactii catre ambele carduri: cel de pe care transferam si cel primitor.

3.5 SecondApplication

In momentul trecerii spre aceasta interfata, la fel ca in cazul interfetei descrise anterior, se creeaza un obiect de tip Card pe care se vor executa diversele operatii.

Aici regasim 6 butoane, dintre care voi explica si prezenta in amanunt butoanele **IBAN** si **SearchTranz**.

IBAN: Cum functioneaza butonul?

In momentul apasarii butonului, se va deschide o noua interfata intitulata IbanDisplay, unde vor fi afisate in cate un label informatii referitoare la cardul curent, cum ar fi Iban-ul contului, numele proprietarului, tipul de card cat si banca la care se regaseste. De mentionat faptul ca in cadrul acestei interfete regasim un buton ce are aspectul unei imprimante; in momentul apasarii acestuia, o noua interfata va fi lansata si anume FileChooser, din cadrul careia putem crea si salva in computerul personal un PDF cu datele precizate. Pentru alegerea locatiei de salvare a datelor, am utilizat un JFileChooser ce are punctul de start in /home/user/(dat de simbolul) si ne permite sa alegem directorul in care dorim sa stocam fisierul. In momentul alegeriei unui director, calea path este updatata si text-field-ul corespunzator numelui fisierului deblocat. Dupa introducerea unui nume, butonul **Print** poate fi apasat, moment in care, prin intermediul API-ului iText (Documentatie aici) este creat si salvat un pdf cu datele precizate anterior. In cazul in care un fisier cu acelasi nume exista, utilizatorul este intrebat daca se doreste suprascrierea lui sau nu.

```

private void makePDF(){
    print . addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            File f=new File(pathName.getText()+" .pdf");
            boolean create=true;
            if(f.exists()){
                int result=JOptionPane.showConfirmDialog(null ,
                    "The _file _already _exists !Do _" +
                    "you _want _to _override _it ?" , "Existen _file" ,
                    JOptionPane.YES_NO_OPTION);
                if(result==JOptionPane.YES_OPTION){
                    if(!f.delete()){
                        JOptionPane.showMessageDialog(null ,
                            " Failed _to _delete _the _actual _file , " +
                            " _please _choose _another _name _for _your _PDF" );
                }
            }else{
                JOptionPane.showMessageDialog(null ,
                    "Choose _another _name!" );
                create=false ;
            }
            if(create) {
                PDF pdf = new PDF(IBAN, fullname, bank,
                    type, pathName.getText()+" .pdf");
                pdf.createDoc();
                JOptionPane.showMessageDialog(null ,
                    "Your _pdf _is _ready _to _be _printed!" );
                display.dispose();
            }
        }
    });
}
}

```

SearchTranz:

In momentul apasarii butonului, o noua interfata va fi lansata: Search, in cadrul careia regasim un JComboBox ce are urmatoarea lista de obiecte: Date, Pret si Nume(de tip String) prin intermediul caruia utilizatorul poate selecta criteriile pentru afisarea tranzactiilor. In cazul selectarii campului Date, se va putea deschide un **datePicker** din API-ul org.jdesktop.swingx.impl.* (Documentatie aici). Dupa selectarea criteriilor, in momentul apasarii butonului **Confirm** se vor extrage din baza de date, tranzactiile respective ce respecta criteriile impuse de utilizator. Pentru afisarea tranzactiilor, se utilizeaza un JScrollPane pentru afisarea tranzactiilor in mod convenabil.

3.6 Settings

Interfata de setari prezinta butoane ce au ca rol management-ul contului si al cardurilor asociate acestora. Ca si in cazul celor 2 interfete prezentate anterior, la pornirea interfetei se va crea un obiect de tip Card ce reprezinta cardul curent utilizat. Din cadrul acestei interfete voi prezenta urmatoarele 2 butoane: **DeleteAccount** si **RemoveCard**.

DeleteAccount:

In momentul apasarii butonului, utilizatorul va primi un mesaj de avertizare daca este sigur sau nu sa ia decizia asta. In cazul in care utilizatorul doreste acest lucru, in cadrul metodei asociate butonului se vor sterge datele din cadrul fiecarei tabele legate de cont, inclusiv fiecare card si tranzactiile sale. Stergerea datelor din tabele se vor face intr-o ordine stabilita, deoarece tabelele sunt conectate intre ele prin intermediul unor chei straine, stergerea intr-o ordine aleatoare ducand la erori fatale. In frame-ul de mai jos, este prezentata o bucatica de cod ce se ocupa cu stergera contului. Se poate observa ca in momentul stergerii, se reseteaza valoarea cardID care se incrementeaza automat.

```
pst=conn . prepareStatement (  
"DELETE FROM users WHERE user=?");  
pst . setString (1 ,username );  
pst . executeUpdate ();  
pst=conn . prepareStatement (  
"ALTER TABLE card_data AUTOINCREMENT=1" );  
pst . executeUpdate ();  
conn . close ();
```

```

if (card!=null)
    card . closeConnection ();
Settings . this . dispose ();
SwingUtilities . invokeLater(()->new LoginInterface () . start ());

```

RemoveCard:

In momentul apasarii butonului se vor numara cardurile asociate contului dupa care, pe baza acestui numar se vor lua diferite actiuni(daca numarul este mai mare decat 1, se va deschide interfata TwoCards sau ThreeCards de unde se va elimina cardul selectat). In cazul in care numarul de carduri este 1, se va sterge si ultimul card asociat contului iar utilizatorul va fi intrebat daca isi doreste sau nu sa pastreze contul curent. Daca se doreste acest lucru, butoanele din interfetele Application si SecondApplication vor fi blocate(cu exceptia butonului cursValutar, ce ne afiseaza cursul valutar din ziua curenta). Mai jos este prezentata o bucată de cod din cadrul metodei de eliminare a cardurilor.

```

PreparedStatement pst=conn . prepareStatement (
"SELECT_cardID _FROM_card_data _WHERE_user=?");
pst . setString (1 , username );
ResultSet rs=pst . executeQuery ();
int cardNr=0 , ID=0;
while (rs . next ()) {
    ++cardNr ;
    ID=rs . getInt (1);
}
if (cardNr==0){
    JOptionPane . showMessageDialog (null ,
    "You don 't have a card !",
    "No card" , JOptionPane . WARNING_MESSAGE);
}
else if (cardNr==1){
pst=conn . prepareStatement (
"DELETE _FROM_tranzactii _WHERE_cardID=?");
pst . setInt (1 , ID );
pst . executeUpdate ();
pst=conn . prepareStatement (""
DELETE _FROM_financiar _WHERE_cardID=?");
pst . setInt (1 , ID );

```

```

pst . executeUpdate ();
pst=conn . prepareStatement (
"DELETE FROM card_type WHERE cardID=?");
pst . setInt (1 , ID );
pst . executeUpdate ();
pst=conn . prepareStatement (
"DELETE FROM data_limit WHERE cardID=?");
pst . setInt (1 , ID );
pst . executeUpdate ();
pst=conn . prepareStatement (
"DELETE FROM card_data WHERE cardID=?");
pst . setInt (1 , ID );
pst . executeUpdate ();
int result=JOptionPane . showConfirmDialog (
null , "You have no card left !
Do you want to delete your account?" ,
"No cards remaining" , JOptionPane . YES_NO_OPTION );

```

3.7 Bug-uri intalnite

-La pornirea interfetei ChooseAccount prin interfata SignUp, daca utilizatorul dorea sa inchida aplicatia in acel punct, datele introduse anterior ar fi ramas salvate, contul fiind asociat fara un card, ceea ce nu este permis(problema rezolvata)

-la addCard, datorita modificarii anterioare, daca nu mai doream adaugarea unui nou card, datele contului ar fi fost sterse, ceea ce nu era de dorit(problema rezolvata)

-se puteau deschide prea multe frame-uri de acelasi tip sau tipuri diferite simultan, problema rezolvata printr-un event transmis la apelul noilor interfete(problema rezolvata)

-datele de pe card nu se actualizau tot timpul, datorita omiterii apelurilor metodei(rezolvat)

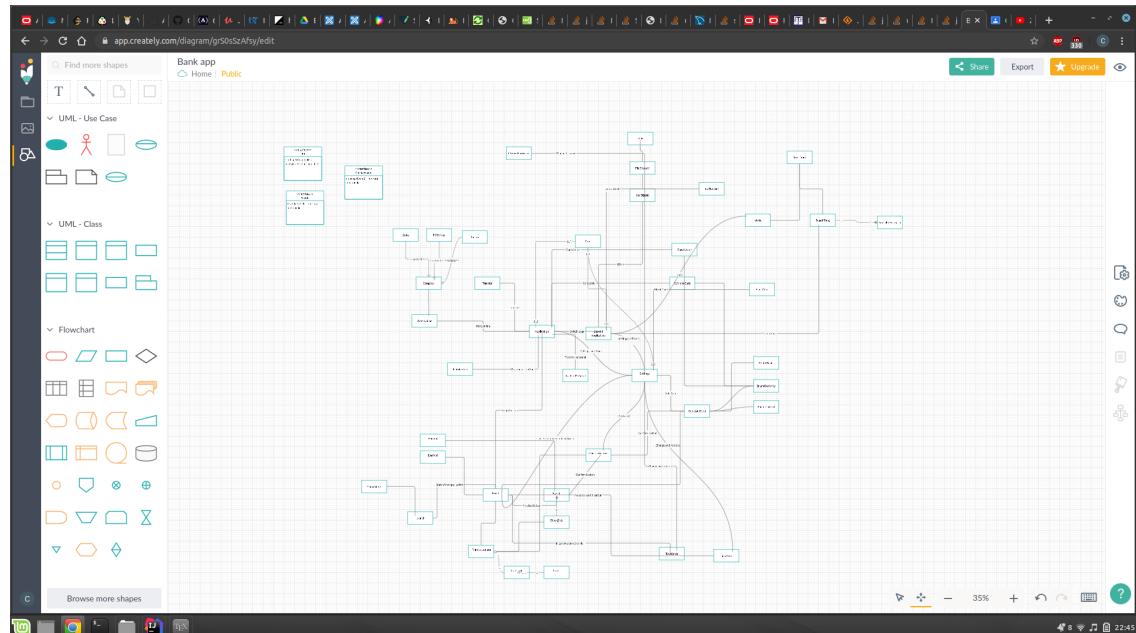
-daca unul dintre site-urile de pe care datele sunt preluate se modifica, aplicatia nu va mai functiona cum trebuie(nu se poate rezolva)

-in caz de eroare sql, datele pot fi doar parcial introduse(nerezolvat, dar se poate remedia)

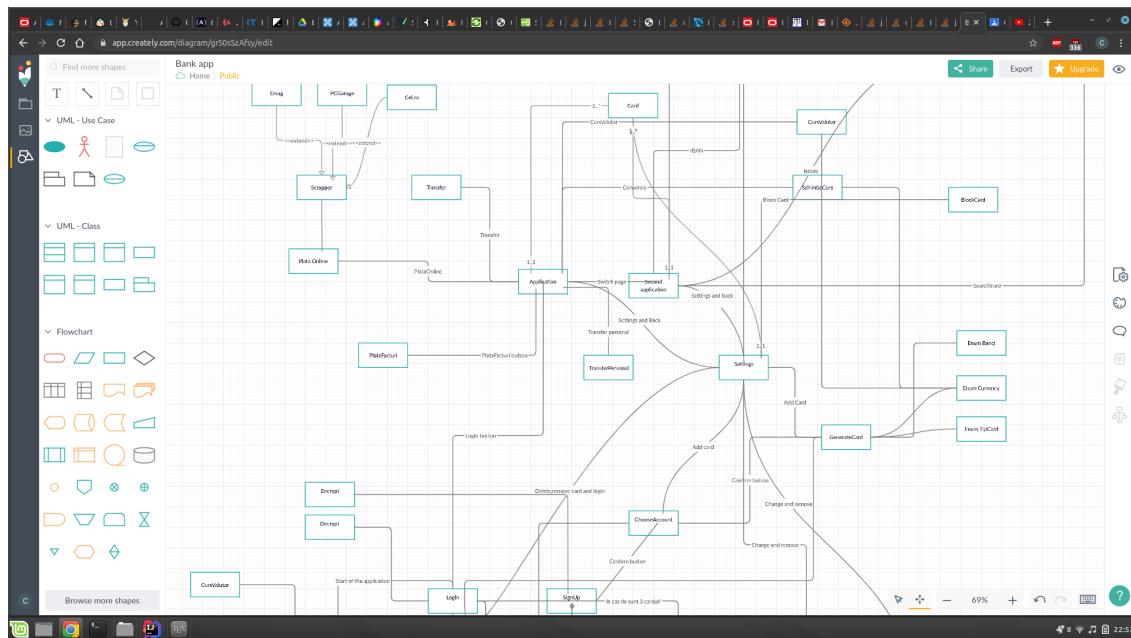
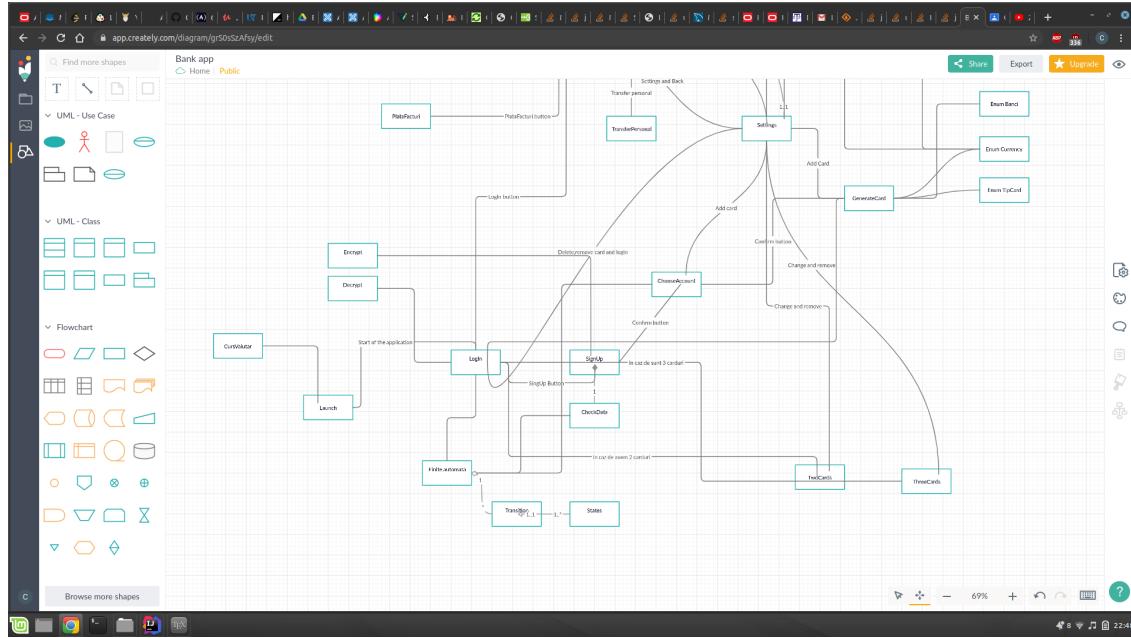
-adaugarea in lista de transfere a salarului cu suma 0, pentru carduri fara
salar, lucru ce nu este de dorit(problema rezolvata)

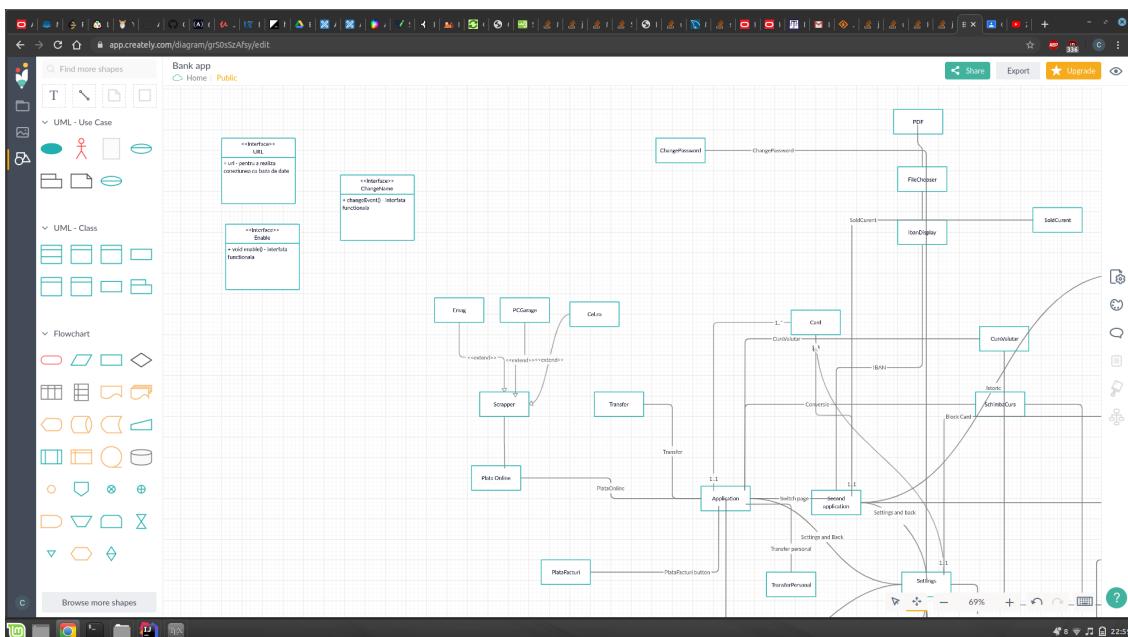
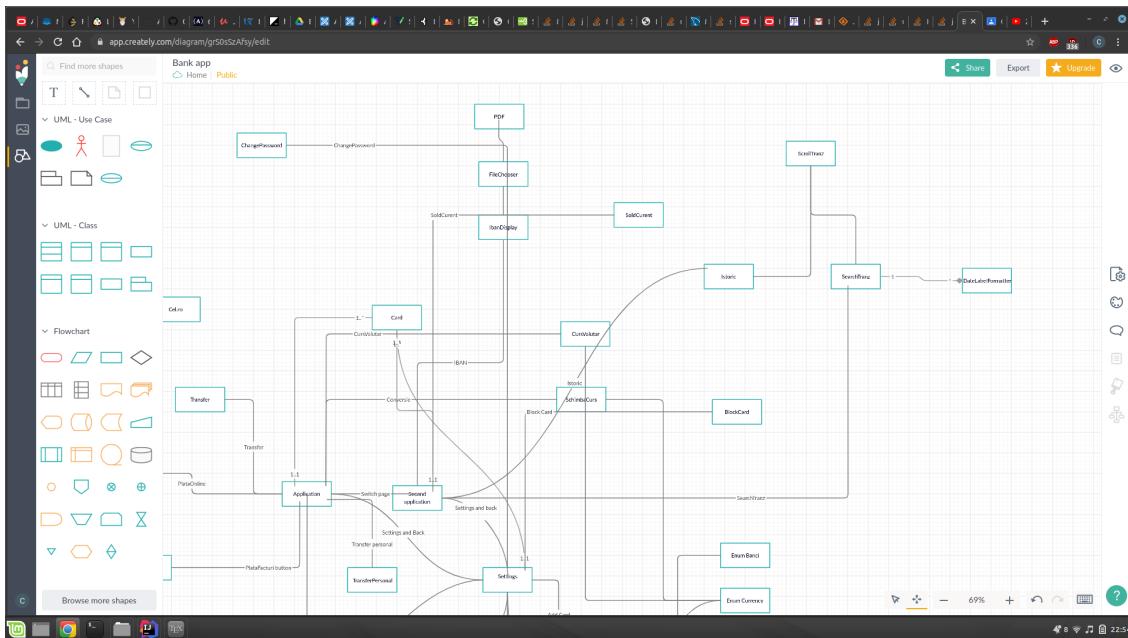
4 Diagrama UML

Pentru această parte a lucrării, am realizat, folosind un tool online (click aici pentru tool) ce cuprinde o diagramă UML pentru clasele ce alcătuiesc proiectul.



In imaginea prezentata mai sus este prezentata diagrama completa, aceasta urmand sa fie reprezentat in mai multe imagini cu dimensiune marita.





In cadrul imaginii de mai sus se pot observa 3 interfețe(2 funcționale). Acestea nu au legături deoarece servesc mai multor clase(precum interfața URL ce are link-ul de conectare la baza de date). De asemenea, și clasa

Card serveste ca utilizare catre multiple clase, aceasta avand legaturi doar cu clasele principale(Application, Second Application si Settings) de unde pornesc si restul claselor ce au legatura cu aceasta.

Pentru vizualizare online a diagramei, click aici.

5 Manual de utilizare al aplicatiei

5.1 Cerinte de utilizare

Pentru a putea utiliza acest program, sunt necesare urmatoarele: posezia unui calculator personal sau laptop, cu un mediu de rulare Java(Java Runtime Environment sau JRE). De asemenea, utilizatorul trebuie sa aiba imprimantă minim 18 ani si un card bancar activ.

5.2 Creeare cont

In momentul in care utilizatorul doreste sa isi creeze un cont prin apasarea butonului de signUp, acesta va fi intampinat de o interfata cu unsprezece campuri de text, fiecare dintre acestea avand un format specific, explicit si de fiecare ToolTip atasat fiecarui camp in parte; cerintele de completare sunt urmatoarele:

Username: Primele n litere majuscule, urmatoarele n litere minuscule, iar la final sa existe cel putin 2 cifre(n mai mare sau egal cu 1). Lungimea maxima a username-ului este de 12 caractere.

password, respectiv **confirmPassword**: Cele 2 campuri destinate parolei pot avea o lungime de maxim 15 caractere, singura cerinta este ca parolele introduse sa fie identice si sa contina minim 2 cifre, oriunde in text.

First name: Poate contine doar litere(majuscule sau nu) si liniuta despartitoare intre nume(-). Conditia impusa este ca fiecare nume sa inceapa cu litera mare.

Last name: Nume de familie, scris cu litera mare la inceput.

CNP: Alcatuit din 14 cifre ce reprezinta codul numeric personal

birthday: Data de nastere valida, in formatul zz/ll/aaaa(unde / va fi adaugat automat)

adress: Campul de adresa necesita urmatoarele: Tara,Oras,Strada; datele vor fi completate la fel ca in descrierea formatului, acestea fiind separate prin ”,”.

email: Aceleasi reglementari ca si in cazul unei adrese de e-mail(de la yahoo, gmail etc).

Job: Accepta litere majuscule, mici si spatii.

phoneNumber: Zece cifre ce reprezinta numarul de telefon al utilizatorului.

Acest format cerut pentru completarea datelor va fi cel utilizat de fiecare data cand aplicatia solicita una dintre informatiile mentionate mai sus.

5.3 Completarea datelor de card

Dupa ce au fost introduse datele personale ale utilizatorului, aplicatia va cere sa introduceti datele de pe cardul bancar. Formatul campurilor in care se introduc aceste date este cel corespunzator celui stabilit de banca ce a emis cardul bancar respectiv.

Acest format este respectat si in momentul adaugarii unui nou card pe cont, interfata de adaugare a cardului fiind aceeasi.

5.4 Utilizarea aplicatiei

Dupa ce crearea contului s-a realizat cu success, utilizatorul se va putea loga cu acesta din cadrul interfetei de logare LogIn. In cadrul fiecarui buton din aplicatie este introdus un mesaj de dialog ce este declansat in momentul apasarii butonului in care se vor regasi informatii referitoare la modul de functionare al butonului si ce fel de date este necesar sa introduceti pentru realizarea cu success a operatiunii bancare dorite.

6 Bibliografie

Pentru realizarea acestei aplicatii, am folosit ca si material de documentare urmatoarele:

Java A Beginners Guide 6th Edition - McGraw Hill, de unde am invatat bazele limbajului Java si secretele acestuia;

O'Reilly Head First SQL - De unde am invatat cum sa creez si sa utilizez o baza de date, impreuna cu legaturile dintre tabele.

Stack OverFlow - Pentru diverse probleme intampinate pe parcursul crearii acestei aplicatii

+Toate site-urile mentionate pe parcursul lucrarii, cum ar fi documentatia referitoare la Swing si Layout-urile acestuia. Aceste site-uri se gasesc mentionate prin (Documentatie aici) sau alte forme de exprimare ce contin aceste 2 cuvinte.