UNIVERSITATEA TEHNICĂ “GHEORGHE ASACHI” IAŞI

FACULTATEA AUTOMATICĂ ŞI CALCULATOARE

SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA BAZE DE DATE

BĂNCĂ DE INVESTIȚII LA BURSĂ

Coordonator: Cătălin Mironeanu

Student: Toma Alexandru-Ionuț

Iași, 2021

Titlu proiect: Gestiunea clientilor unei banci de actiuni la bursa

Analiza, proiectarea si implementarea unei baze de date si a aplicatiei aferente care sa modeleze activitatea unei banci cu privire la gestiunea clientilor, si a investitiilor acestora la Bursa de Valori Bucuresti.

Descrierea cerintelor si modul de organizare al proiectului

Aceasta este o banca care se axeaza pe tranzactii de actiuni la bursa.

Clientii pot sa isi faca un cont la banca, iar dupa ce contul lor este creat, acestia pot sa realizeze tranzactii simple in cadrul bancii. Pot sa isi depuna sau sa isi retraga bani, si mai pot sa transfere bani de la un cont la altul pe baza unui IBAN.

Tranzactiile sunt putine si simple, ideea principala fiind de a cumpara actiuni la diverse companii si de a obtine profit prin vanzarea lor ulterioara.

Pretul actiunilor este variabil in timp, iar fiecare companie o sa aiba un numar standard de actiuni, care pot fi cumparate de clientii bancii respective. Cand se doreste, clientii isi pot vinde o parte sau chiar toate actiunile la pretul curent al pietei.

Ca oricare alta banca, va avea un commission de tranzactii. Acesta se aplica la tranzactii de tip transfer si la vanzare de actiuni. Comisionul este unul standard de 2%.

Descrierea functionala a aplicatiei

Banca de actiuni dispune de urmatoarele functionalitati:

- Creare cont unic pentru fiecare client
- Depunere, retragere si transfer de numerar
- Evidenta tranzactiilor efectuate de clienti
- Cumparare si vanzare de actiuni listate la Bursa de Valori Bucuresti
- Preturi la actiuni generate semi-random folosind Perlin Noise
- Evidenta investitiilor efectuate de clienti
- Statistici pentru actiuni:
    - Evolutia pretului
    - Actionar majoritar
    - Procent detinere
- Statistici evolutie cont:
    - Profit/Pierderi
    - Preturi medii de cumparare/vanzare

Tehnologii folosite in cadrul realizarii aplicatiei

Aplicatia este destinat utilizatorilor windows. Aceasta a fost creata folosind .NET si C#.

Pentru criptarea si decriptarea parolelor am folosit functiile:

https://www.c-sharpcorner.com/blogs/encrypt-and-decrypt-a-string-in-asp-net1

Pentru generearea unui Perlin Noise am folosit:

https://gist.github.com/Flafla2/1a0b9ebef678bbce3215

Conectarea la baza de date Oracle se face folosind libraria Oracle.ManagedDataAccess.Client.

```
37 references
public static class DataBase
{

    private static OracleConnection DBconnection =
        new OracleConnection(ConfigurationManager.ConnectionStrings["DBconnection"].ConnectionString);

    1 reference
    private static bool ConnectToDatabase()
    {
        try
        {
            DBconnection.Open();
        }
        catch
        {
            return false;
        }
        return true;
    }
    0 references
```

Exemplu de functie care obtine procentajul de detinere al unei companii de catre un client:

```
2 references
public static float GetOwnedPercentStock(uint clientId, string stockId)
{
    try
    {
        OracleCommand command = DBconnection.CreateCommand();
        command.CommandText =
            "WITH stock_temp AS" +
            "(SELECT client_id, stock_id, SUM(ammount) AS ammount " +
            "FROM invest GROUP BY client_id, stock_id ) " +
            "SELECT(s.ammount / b.initial_ammount * 100) " +
            "FROM stock_temp s, client c, bvb b " +
            "WHERE s.client_id = c.client_id AND b.stock_id = s.stock_id " +
            $"AND c.client_id = {clientId} AND s.stock_id = '{stockId}'";

        OracleDataReader dataReader = command.ExecuteReader();

        while (dataReader.Read())
        {
            float percent = dataReader.GetFloat(0);
            return percent;
        }
    }
    catch
    {
        return 0f;
    }
    return 0f;
}
```

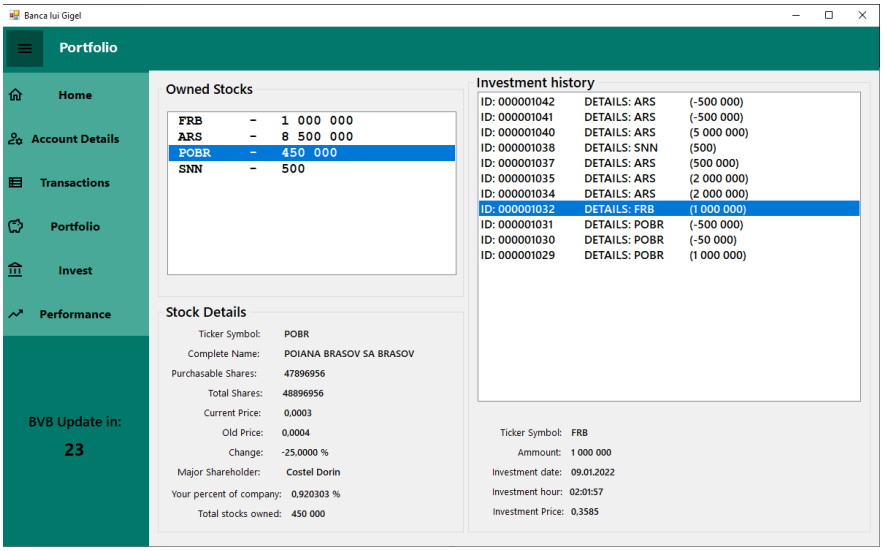
Exemplu de tranzactie pentru inserarea unui nou client in baza de date:

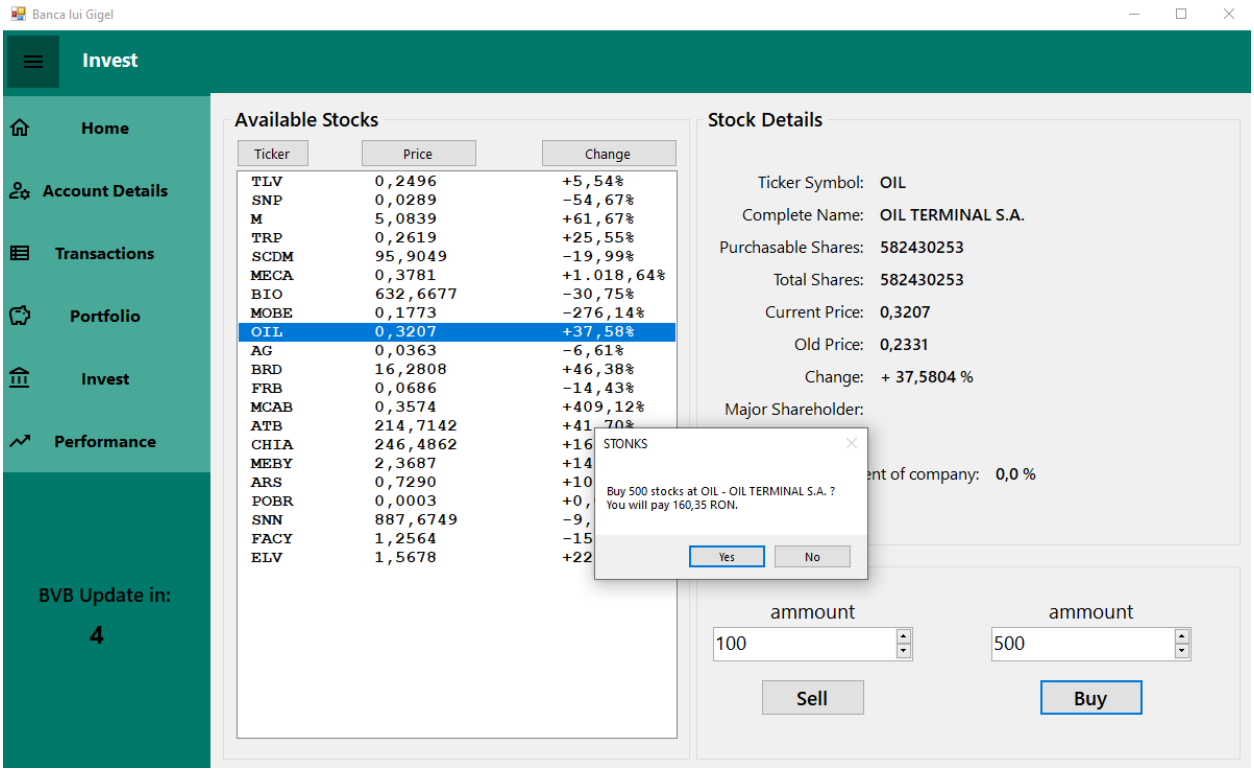
```
1 reference
public static string InsertNewClientAccount(Account account)
{


    OracleTransaction transaction = DBconnection.BeginTransaction(System.Data.IsolationLevel.ReadCommitted);
    OracleCommand command = DBconnection.CreateCommand();

    try
    {
        command.CommandText =
            "INSERT INTO client(client_id, first_name, last_name, cnp) " +
            $"VALUES({account.ClientId}, " +
            $"'{ account.ClientProp.FirstName }', " +
            $"'{ account.ClientProp.LastName }', " +
            $"'{ account.ClientProp.Cnp }')";
        command.ExecuteNonQuery();

        command.CommandText =
        "INSERT INTO client_details(client_id, birth_date, genre, city, address, phone, email) " +
            $"VALUES({account.ClientId}, " +
            $"TO_DATE('{account.ClientProp.ClientDetailsProp.BirthDate}', 'DD.MONTH.YYYY')," +
            $"'{account.ClientProp.ClientDetailsProp.Genre}', " +
            $"'{ account.ClientProp.ClientDetailsProp.City }', " +
            $"'{ account.ClientProp.ClientDetailsProp.Address }', " +
            $"'{ account.ClientProp.ClientDetailsProp.Phone }', " +
            $"'{ account.ClientProp.ClientDetailsProp.Email }')";
        command.ExecuteNonQuery();

        string comm = account.Commission.ToString();
        comm.Replace(",", ".");
        command.CommandText =
            "INSERT INTO account(client_id, iban, balance, username, password, commission) " +
            $"VALUES({account.ClientId}, " +
            $"'{ account.Iban }', " +
            $" { account.Balance }, " +
            $"'{ account.Username }', " +
            $"'{ Cryptography.Encrypt(account.Password) }', " +
            $" 0.02 )";
        command.ExecuteNonQuery();

        transaction.Commit();
    }
    catch(OracleException e)
    {
        transaction.Rollback();

        command.Dispose();
        transaction.Dispose();
        return "db_insert_err: " + e.ToString();
    }

    command.Dispose();
    transaction.Dispose();
    return "success";
}

```

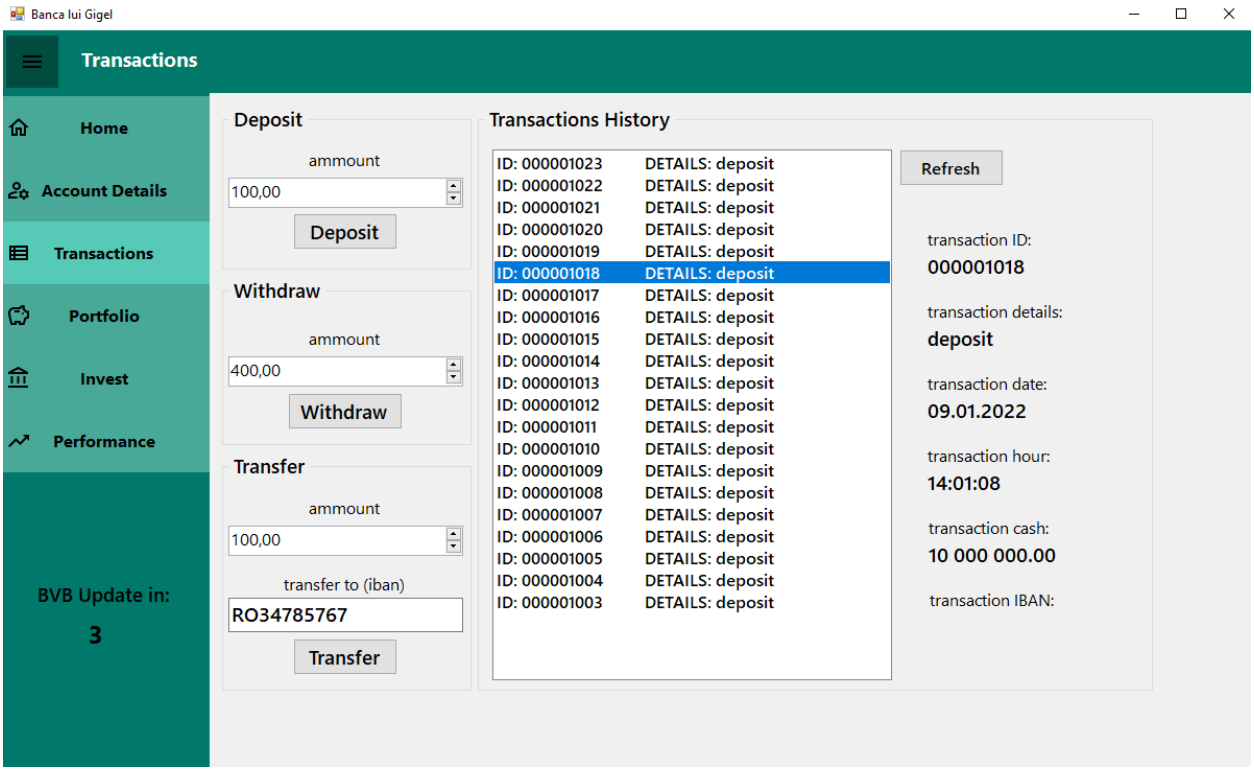
Exemplu de functie pentru obtinerea unor statistici legate de pret mediu de achizitie/vanzare si profit:

```
1 reference
836 public static Statistic GetStatistic(uint clientId, string stockId)
837 {
838     try
839     {
840         OracleCommand command = DBconnection.CreateCommand();
841
842         command.CommandText =
843             $" WITH pret_mediu_achizitie AS( " +
844             $"  SELECT SUM(ammount * aquisition_price) / SUM(ammount) AS pret " +
845             $"  FROM invest WHERE stock_id = '{stockId}' " +
846             $"  AND client_id = {clientId} AND ammount > 0), " +
847             $"pret_mediu_vanzare AS( " +
848             $"  SELECT SUM(ammount* aquisition_price)/ SUM(ammount) AS pret " +
849             $"  FROM invest WHERE stock_id = '{stockId}' " +
850             $"  AND client_id = {clientId} AND ammount< 0), " +
851             $"actiuni_vandute AS( " +
852             $"  SELECT SUM(-ammount) AS numar " +
853             $"  FROM invest WHERE stock_id = '{stockId}' " +
854             $"  AND client_id = {clientId} AND ammount< 0) " +
855             $"" +
856             $"SELECT a.numar * pv.pret * (1 - a.commission) - a.numar * pa.pret, " +
857             $"a.numar * (pv.pret - pa.pret) * a.commission, " +
858             $"pv.pret, pa.pret, a.numar " +
859             $"FROM actiuni_vandute a, pret_mediu_achizitie pa, pret_mediu_vanzare pv, account a " +
860             $"WHERE a.client_id = {clientId} ";
861
862         OracleDataReader dataReader = command.ExecuteReader();
863
864         while (dataReader.Read())
865         {
866             float profit = 0f;
867             float comm = 0f;
868             float buy = 0f;
869             float sell = 0f;
870             long sold = 0;
871
872             try
873             {
874                 profit = dataReader.GetFloat(0);
875             }
876             catch { }
877             try
878             {
879                 comm = dataReader.GetFloat(1);
880             }
881             catch { }
882             try
883             {
884                 sell = dataReader.GetFloat(2);
885             }
886             catch { }
887             try
```

Screenshot-uri din interfata grafica:




Banca lui Gigel
Invest
Home
Account Details
Transactions
Portfolio
Invest
Performance
BVB Update in:
4
Available Stocks
Ticker
Price
Change
TLV 0,2496 +5,54%
SNP 0,0289 -54,67%
M 5,0839 +61,67%
TRP 0,2619 +25,55%
SCDM 95,9049 -19,99%
MECA 0,3781 +1.018,64%
BIO 632,6677 -30,75%
MOBE 0,1773 -276,14%
OIL 0,3207 +37,58%
AG 0,0363 -6,61%
BRD 16,2808 +46,38%
FRB 0,0686 -14,43%
MCAB 0,3574 +409,12%
ATB 214,7142 +41,70%
CHIA 246,4862 +16
MEBY 2,3687 +14
ARS 0,7290 +10
POBR 0,0003 +0,
SNN 887,6749 -9,
FACY 1,2564 -15
ELV 1,5678 +22
Stock Details
Ticker Symbol: OIL
Complete Name: OIL TERMINAL S.A.
Purchasable Shares: 582430253
Total Shares: 582430253
Current Price: 0,3207
Old Price: 0,2331
Change: + 37,5804 %
Major Shareholder:
nt of company: 0,0 %
STONKS
Buy 500 stocks at OIL - OIL TERMINAL S.A. ?
You will pay 160,35 RON.
Yes
No
ammount
100
ammount
500
Sell
Buy

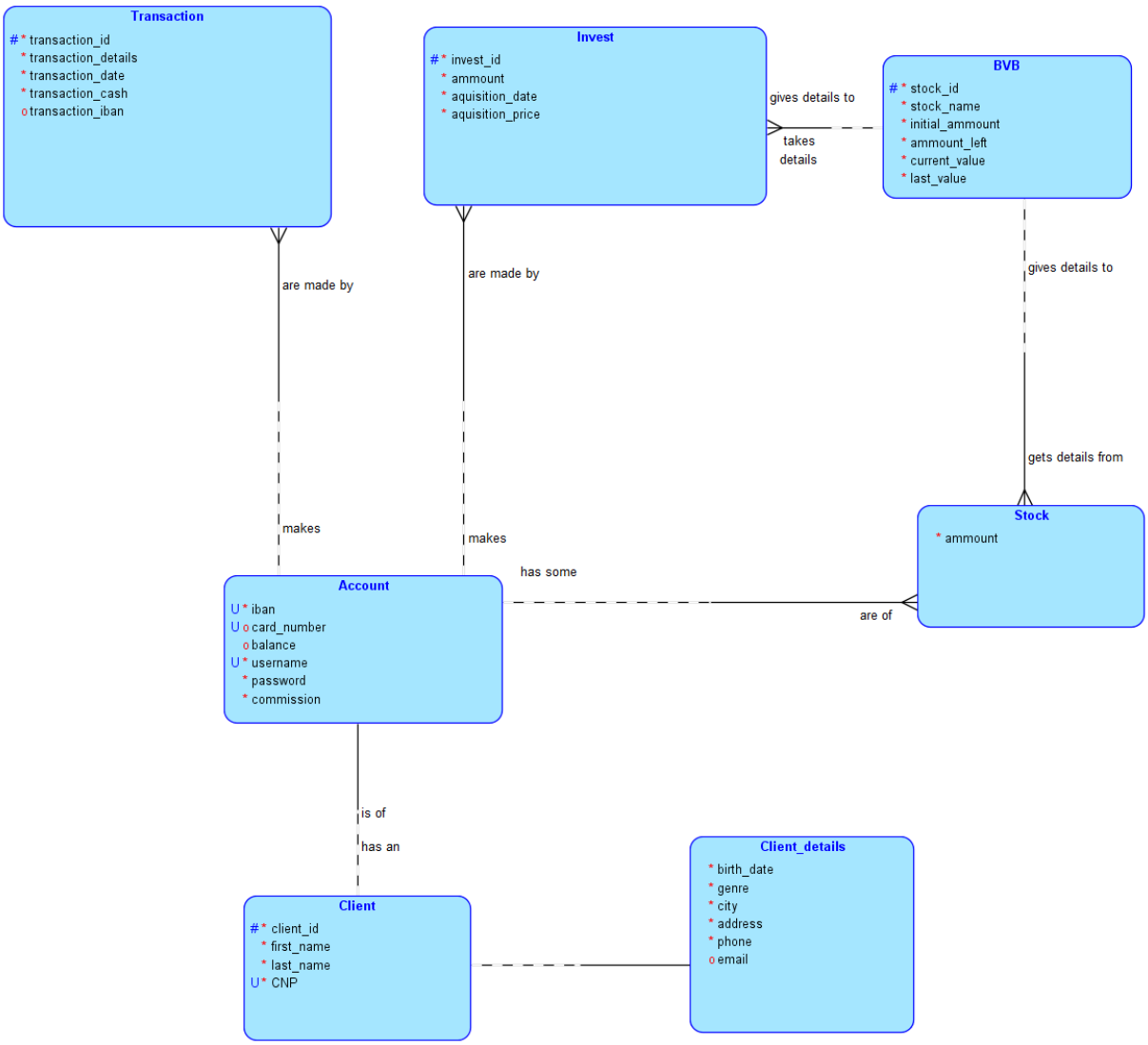

Banca lui Gigel
Transactions
Home
Account Details
Transactions
Portfolio
Invest
Performance
BVB Update in:
3
Deposit
ammount
100,00
Deposit
Withdraw
ammount
400,00
Withdraw
Transfer
ammount
100,00
transfer to (iban)
RO34785767
Transfer
Transactions History
ID: 000001023 DETAILS: deposit
ID: 000001022 DETAILS: deposit
ID: 000001021 DETAILS: deposit
ID: 000001020 DETAILS: deposit
ID: 000001019 DETAILS: deposit
ID: 000001018 DETAILS: deposit
ID: 000001017 DETAILS: deposit
ID: 000001016 DETAILS: deposit
ID: 000001015 DETAILS: deposit
ID: 000001014 DETAILS: deposit
ID: 000001013 DETAILS: deposit
ID: 000001012 DETAILS: deposit
ID: 000001011 DETAILS: deposit
ID: 000001010 DETAILS: deposit
ID: 000001009 DETAILS: deposit
ID: 000001008 DETAILS: deposit
ID: 000001007 DETAILS: deposit
ID: 000001006 DETAILS: deposit
ID: 000001005 DETAILS: deposit
ID: 000001004 DETAILS: deposit
ID: 000001003 DETAILS: deposit
Refresh
transaction ID:
000001018
transaction details:
deposit
transaction date:
09.01.2022
transaction hour:
14:01:08
transaction cash:
10 000 000.00
transaction IBAN:

Descrierea detaliata a entitatilor si a relatiilor dintre tabele

Entitatile din cadrul aplicatiei sunt:

- **Client**
- **Client_details**
- **Account**
- **Transaction**
- **Invest**
- **Stock** (redundant)
- **BVB**

Diagrama cu entitatile:



Client va contine informatiile de baza ale unui client al bancii, cum ar fi numele, CNP-ul si un ID specific bancii.

Client_details va contine informatiile detaliate ale clientului: data nasterii, genul, orasul si adresa, numarul de telefon si un email.

Account va contine informatiile legate de contul clientului. Aici avem un IBAN, un username cu o parola pentru logare, comisionul afferent bancii si desigur, numerarul. Optional se mai gaseste si un numar de card.

Transaction va contine inregistrarile tuturor tranzactiilor effectuate de clienti, aceasta contine un id de tranzactie, detaliile tranzactiei, data la care a fost efectuata, numerarul care a fost procesat si eventual un iban daca tranzactia este de tip transfer.

Invest va contine inregistrarile tuturor investitiilor facute de clienti, aici vom avea un id de investitie, un numar de actiuni care au fost cumparate/vandute, pretul cu care au fost cumparate, si data la care a avut loc investitia. Restul detaliilor, precum pretul si numele vor fi date de entitatea **BVB**.

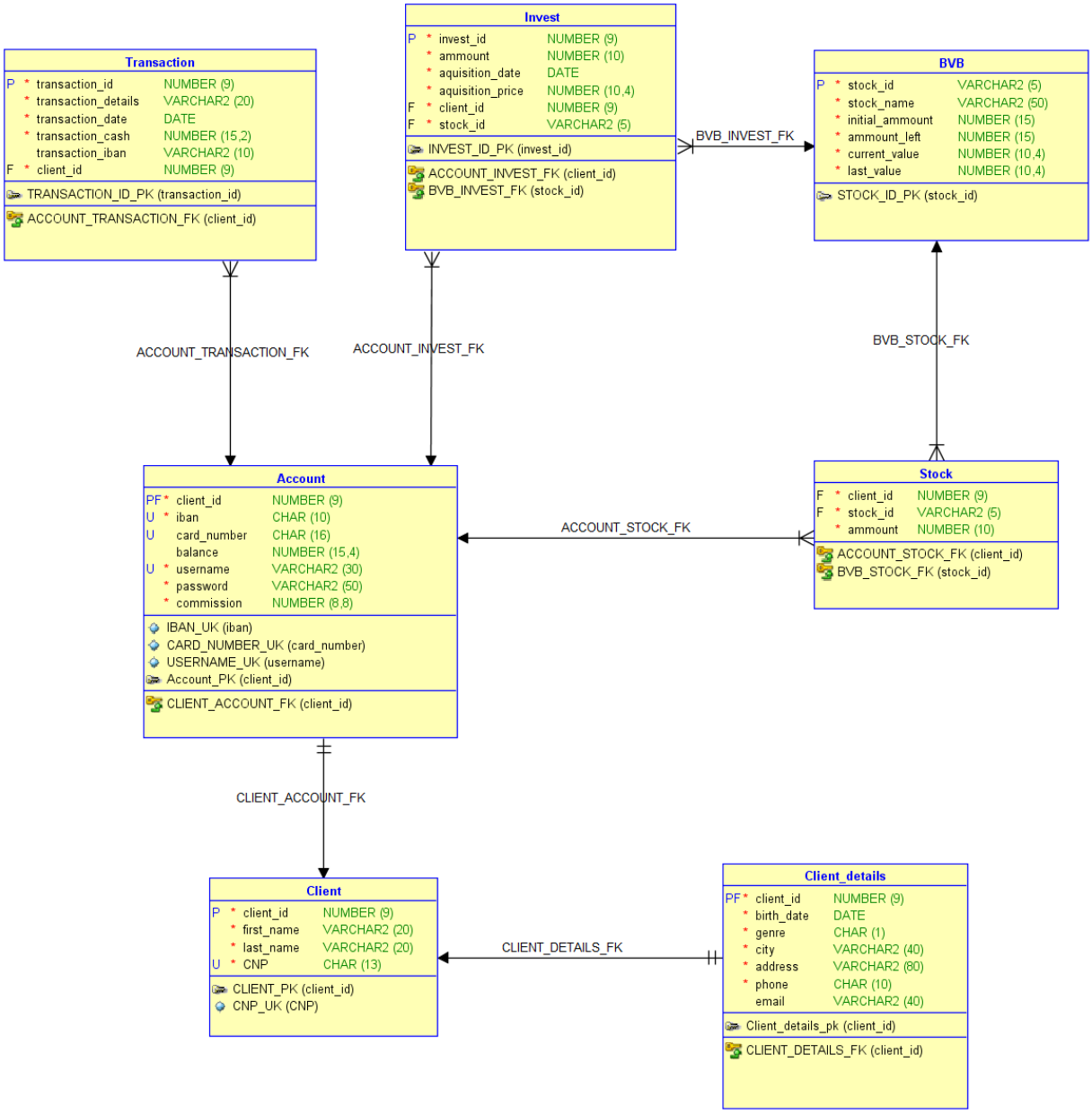
Stock va contine informatiile legate de numarul actiunilor detinute de fiecare client. Dupa un update a devenit redundanta, deoarece toate informatiile pe care le avea se pot afla cu usurinta folosind grupari din tabela **Invest.**

BVB va contine inregistrarile actiunilor listate. Aici vom avea un id specific actiunii, numele, totatul de actiuni listate, numarul de actiuni ramase, pretul current si pretul vechi.

Tabelele din cadrul aplicatiei sunt:

- **Client**
- **Client_details**
- **Account**
- **Transaction**
- **Invest**
- **Stock** (redundanta)
- **BVB**

Diagrama cu tabelele:



Descrierea relatiilor dintre tabele:

In proiectarea acestei baze de date s-au identificat tipurile de **relatii: 1:n**, **n:1** si **1:1**.

Intre tabela **Client** si **Client_details** este o relatie de **1:1**, iar legatura se realizeaza prin campul **client_id**.

Intre tabela **Client** si **Account** este o relatie tot de **1:1**, deoarece un client poate avea un singur cont, iar un cont nu poate fi detinut de mai multi clienti. Legatura este realizata de campul **client_id**.

Tabelele **Account** si **Transaction** se afla intr-o relatie de **1:n**. Acest lucru se datoreaza faptului ca un cont poate avea mai multe tranzactii diferite, dar o tranzactie nu poate fi asignata mai multor conturi. Intre cele doua tabele, legatura se face tot cu **client_id**.

Intre tabela **Account** si tabela **Invest** avem o relatie foarte asemanatoare cu cea dinte **Account** si **Transaction**. Ele se afla de asemenea intr-o relatie de **1:n**, tot datorita faptului ca un cont poate avea mai multe inregistrari cu investitii, iar o investitie este a unui singur cont. Legatura este data tot de **client_id**.

Avem o realtie de **n:1** intre tabela **Invest** si tabela **BVB**. Tabela **BVB** ofera tabelei **Invest** detalii, iar relatia este cat se poate de logica. In tabela **Invest** avem multiple inregistrari alea actiunilor din **BVB**, iar inregistrarile din **BVB** sunt de sine statatoare si nu ar avea sens sa fie in legatura multipla cu cele din tabela **Invest**. Legatura se face prin **stock_id**.

Mai avem inca o relatie de **n:1** intre tabela **Stock** si **BVB**, relatie identica cu cea dintre **Invest** si **BVB**. Legatura este in mod evident facuta tot prin **stock_id**.

Constrangeri folosite:

- **PRIMARY KEY**
  - CLIENT_PK(client_id): fiecare client trebuie sa aiba o cheie primara cu care sa se identifice si sa se faca legatura cu alte tabele prin intermediul acesteia, pentru aceasta s-a implementant mecanism de auto-increment
  - STOCK_ID_PK(stock_id): fiecare actiune se identifica printr-o cheie primara formata din 3, 4 sau 5 caractere.
  - TRANSACTION_ID_PK(transaction_id): fiecare tranzatie este iregistrata cu o cheie primara si unica, pentru aceasta s-a implementant mecanism de auto-increment
  - INVEST_ID_PK(invest_id): fiecare investitie este iregistrata cu o cheie primara si unica, pentru aceasta s-a implementant mecanism de auto-increment

- **FOREIGN KEY**
  - BVB_STOCK_FK(stock_id): cheie care face legatura dintre tabela BVB si tabela Stock
  - BVB_INVEST_FK(stock_id): cheie care face legatura dintre tabela BVB si tabela Invest
  - CLIENT_ACOUNT_FK(client_id): leaga tabela Client de tabela Account
  - CLIENT_DETAILS_FK(client_id): leaga tabela Client de tabela Client_details
  - ACOUNT_STOCK_FK(client_id): leaga tabela Account de tabela Stock
  - ACOUNT_INVEST_FK(client_id) : leaga tabela Account de tabela Invest
  - ACOUNT_TRANSACTION_FK(client_id) : leaga tabela Account de tabela Transaction

- **UNIQUE**
  - CNP_UK(CNP): CNP-ul clientului este unic pentru fiecare client, nu pot exista doi clienti cu acelasi CNP
  - IBAN_UK(iban): IBAN-ul unui cont este unic, nu vom putea avea acelasi IBAN pentru doua sau mai multe conturi distincte
  - CARD_NUMBER_UK(card_number): fiecare cont va avea un numar de card unic.
  - USERNAME_UK(username): pentru a accesa contul este nevoie de un username, in mod evident acesta trebuie sa fie unic pentru a nu exista conflicte intre conturi

- **CHECK**
  - Client_first_name_ck: s-a folosit un regex pentru validarea numelui, pentru a nu avea cifre sau alte caractere fara sens in nume
  - Client_last_name_ck: identic ca cel precedent
  - Client_cnp_ck: s-a folosit tot un regex pentru a valida CNP-ul, acesta trebuie sa contina doar 13 cifre
  - Client_details_genre_ck: validare pentru genul persoanei, se admit doar valori de ‘M’ sau ‘F’

- Client_details_phone_ck: validarea numarului de telefon ca acesta sa contina doar 10 cifre
- Client_details_email_ck: validarea adresei de email
- Account_iban_ck: validarea IBAN-ului, acesta sa fie de forma ‘RO99999999’
- Account_card_number_ck: validarea numarului cardului pentru a fi compus doar din 16 cifre
- Account_balance_ck: validarea numererului pentru a nu ajunge la valori negative
- Transaction_details_ck: validare pentru detaliile tranzactiei, aceasta poate fi: ‘deposit’, ‘withdraw’ sau ‘transfer’
- Transaction_iban_ck: validarea IBAN-ului, acesta sa fie de forma ‘RO99999999’
- BVB_initial_ammount_ck: validare pentru numarul de actiuni initiale, acestea trebuie sa fie mai mari decat 0
- BVB_ammount_left_ck: validare pentru numarul de actiuni ramase, acestea trebuie sa fie mai mari decat 0 sau egale cu 0
- BVB_current_value_ck: valoarea unei actiuni nu poate fi negativa
- BVB_last_value_ck: valoarea unei actiuni nu poate fi negativa