Universitatea Tehnică „Gheorghe Asachi” din Iași

Facultatea de Automatică și Calculatoare

Proiect la disciplina

Baze de Date

Student: Doboș Cosmin

Anul: 3

Grupa: 1308A

Capitolul 1. Introducere

Proiectul dat imită funcționalitatea unei baze de date în sfera domeniului farmaceutic.

API-ul din proiectul curent oferă utilizatorului posibilitatea de a naviga în tabela adreselor, farmaciilor, producătorilor, medicamentelor și a facturilor. Pentru început user-ul este redirecționat pe pagina de 'home'.

În aplicație a fost implementat doar procedeul de READ asupra tabelelor din baza de date.

Capitolul 2. Tehnologiile folosite pentru front-end și back-end

Baza de date folosită în această aplicație este SQLPlus. Pentru partea de front-end s-a folosit JavaSpark ca server web. Pentru randarea paginilor web, am folosit motorul Velocity.

Pentru paginile web, am folosit Bootstrap 4 și Font Aweseome.

Pentru a include cele enunțate mai sus, am inclus următoarele dependențe Maven în fișierul **pom.xml**.

```
<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.5</version>
</dependency>
<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-template-velocity</artifactId>
    <version>2.7.1</version>
</dependency>
```

A fost creată o clasă WebServer. În constructorul său sunt definite rutele aplicației. Pentru fiecare rută sunt extrase datele din baza de date și puse într-un obiect de tip HashMap și transmise ca parametru în funcția render dintr-o clasa Utils, creată pentru a ușura randarea paginii.

```
public class Utils {
    public static String render(Map<String, Object> model, String templatePath) {
        return new VelocityTemplateEngine().render(new ModelAndView(model, templatePath));
    }
}
```

Mai jos este prezentat constructorul clasei WebServer, precum și câteva exemple de definire a rutelor.

```java
public class WebServer {
    SQL_XE sql_xe;

    public WebServer(){
        sql_xe = new SQL_XE();
        System.out.println("Database initialized successfully");

        init_routes();
        System.out.println("Web Server Routes initialized successfully");

        System.out.println("Home Page: http://localhost:4567/");
    }

    private void init_routes(){
        get( path: "/", (req, res) -> {
            //creem obiectul model de tip HashMap (dictionar)
            Map<String, Object> model = new HashMap<>();
            String templateVariable = "Hello Velocity!";

            //incarcam o variabila in model
            model.put("message", templateVariable);

            //transmitem modelul la functia render din utilities
            return Utils.render(model, templatePath: "home.html");
        });

        get( path: "/addresses", (req, res) -> {
            Map<String, Object> model = new HashMap<>();
            try {
                List<Map<String, String>> addresses = sql_xe.getAddresses();
                model.put("addresses", addresses);
            }catch (SQLException throwables) {
                throwables.printStackTrace();
            }
            return Utils.render(model, templatePath: "addresses.html");
        });

        get( path: "/producers", (req, res) -> {
            Map<String, Object> model = new HashMap<>();
            try {
                // luam din baza de date producatorii si ii punem in model
                List<Map<String, String>> producers = sql_xe.getProducers();
                model.put("producers", producers);
            }catch (SQLException throwables) {
                throwables.printStackTrace();
            }
            //randam pagina producers transmitand modelul creat
            return Utils.render(model, templatePath: "producers.html");
        });

        get( path: "/pharmacies", (req, res) -> {
            Map<String, Object> model = new HashMap<>();
            try {
```

Pentru afișarea datelor am folosit funcțiile din utilitarul Velociy, după cum se vede și în continuare:

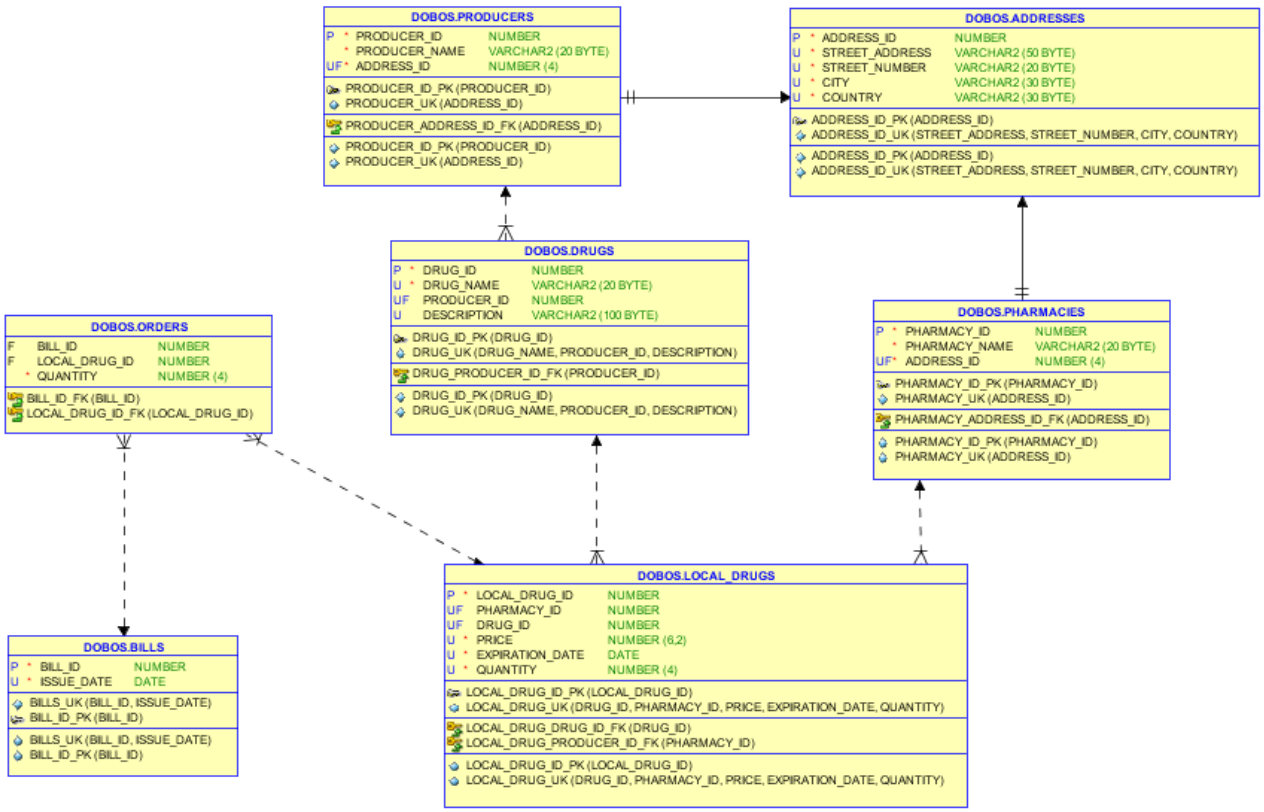
```
<table style="width:100%;" class="table table-bordered">
    <thead class="thead-dark">
    <tr>
        <th>Producer Id</th>
        <th>Producer Name</th>
        <th>Street</th>
        <th>Street Number</th>
        <th>City</th>
        <th>Country</th>
    </tr>
    </thead>
    <tbody>
    #foreach ($producer in $producers)
    <tr style="background: azure">
        <td>$producer['producer_id']</td>
        <td>$producer['producer_name']</td>
        <td>$producer['street_address']</td>
        <td>$producer['street_number']</td>
        <td>$producer['city']</td>
        <td>$producer['country']</td>
    </tr>
    #end
    </tbody>

</table>
```

Capitolul 3. Structura și inter-relaționarea tabelelor

- O farmacie se poate afla la o singură adresă (One-To-One între **Pharmacies** și **Addresses**)
- Un producător se poate afla la o singură adresă (One-To-One între **Producers** și **Addresses**)
- Un medicament poate fi creeat de un singur producător, iar un producător poate creea mai multe tipuri de medicamente (One-To-Many înre **Producer** și **Drugs**).
- Un medicament local se poate afla într-o singură farmacie, iar o farmacie poate conține mai multe medicamente locale (One-To-Many între **Local_Drugs** și **Pharmacies**). Un medicament local reprezintă un medicament al unui producător, iar acel medicament al producătorului îi poate corespunde mai multe medicamente locale de la mai multe farmacii (One-To-Many între **Local_Drugs** și **Drugs**) cu prețuri (cel mai probabil) diferite. Prin intermediul tabelei **Local_Drugs** se realizează relația Many-To-Many dintre farmacii (**Pharmacies**) și medicamentele de la producători (**Drugs**) .
- O factură poate conține mai multe produse, astfel se realizează One-To-Many între **Bills** și **Orders.**
- Un produs este format dintr-un singur medicament și o cantitate a acestuia, astfel se realizează relația One-One **Orders** și **Local_Drugs.**

Capitolul 4. Descrierea constrângerilor folosite

Constrângerile, atributele și cheile tabelelor sunt reprezentate în mai jos.



Atributele, constrăngerile și cheile tabelelor

Pentru toate atributele din fiecare tabel, cu excepția atributului **Description** din tabelul **Drugs** s-a impus constrângerea de tip check – **Not Null**.

Avem constrăngere de tip check asupra coloane **Price** din tabelul **Drugs**, și asupra coloanei **Quantity** din tabelele **Local_Drugs** și **Orders**.

În fiecare tabelă, cu excepția tabelei **Addresses** avem consrăngeri de tip cheie străină.

De asemenea avem și tipuri de chei de tip Unique Key in tabelele **Addresses, Producers, Drugs, Pharmacies** și **Local_Drugs.**

Capitolul 5. Descrierea modalității de conectare la baza de date

Conectarea la baza de date a fost făcută prin intermediul pachetului java.sql, care oferă API-ul pentru accesarea și procesarea datelor stocate într-o sursă de date (de obicei, o bază de date relațională) folosind limbajul de programare Java. Pentru a folosi acest pachet am adăugat următoarea depedență Maven în fișierul pom.xml.

S-a creeat o clasă SQL_XE care oferă funcționalitatea de extragere a datelor din fiecare tabelă.

```xml
<dependency>
    <groupId>com.oracle.ojdbc</groupId>
    <artifactId>ojdbc8</artifactId>
    <version>19.3.0.0</version>
</dependency>
```

```java
public SQL_XE(){
    try {
        //step1 load the driver class
        Class.forName("oracle.jdbc.driver.OracleDriver");

    } catch (ClassNotFoundException classNotFoundException) {
        classNotFoundException.printStackTrace();
    }
}

public List<Map<String, String>> getAddresses() throws SQLException {
    //step2 create  the connection object
    Connection con = DriverManager.getConnection( url: "jdbc:oracle:thin:@localhost:1521:xe", user: "dobos", password: "dobos");

    //step3 create the statement object
    Statement stmt = con.createStatement();

    //step4 execute query
    ResultSet rs = stmt.executeQuery( sql: "select * from addresses");

    List<Map<String, String>> listOfAddresses = new ArrayList<>();
    while (rs.next()){
        HashMap<String, String> address = new HashMap<>();
        address.put("address_id", rs.getString( columnLabel: "ADDRESS_ID"));
        address.put("street_address", rs.getString( columnLabel: "STREET_ADDRESS"));
        address.put("street_number", rs.getString( columnLabel: "STREET_NUMBER"));
        address.put("city", rs.getString( columnLabel: "CITY"));
        address.put("country", rs.getString( columnLabel: "COUNTRY"));
        listOfAddresses.add(address);
    }
    //step5 close the connection object
    con.close();
    return listOfAddresses;
}
```

Mai întâi, în constructorul clasei se încarcă driver-ul ojdbc, precizat mai sus. Ulterior, în fiecare metodă de extragere a datelor se creează o conexiune, urmat de un obiect de tip statement prin interdiul căruia se execută query-ul și se preiau datele din baza de date. Aceste date sunt ulterior puse într-o listă de hashMap-uri, care este returnată. Apoi se închide conexiunea.

Pentru afișarea tabelelor farmacii, producători, medicamente, medicamente locale, facturi, comenzi se folosesc join-uri dintre tabele.

```sql
/* AFISARE */
/* afisare adrese*/
SELECT * FROM addresses;

/* afisare farmacii*/
SELECT * FROM pharmacies p, addresses adr where p.address_id = adr.address_id;

/* afisare producatori*/
SELECT * FROM producers p, addresses adr where p.address_id = adr.address_id;

/* afisare medicamente*/
SELECT d.drug_id, d.drug_name, COALESCE(d.description, 'No Description') as description, p.producer_name, adr.city, adr.country FROM drugs d, producers p, addresses adr
    where p.address_id = adr.address_id and d.producer_id = p.producer_id;

/* afisare medicamente prezente la farmacii specifice (medicamente locale)*/
SELECT l.LOCAL_DRUG_ID, d.DRUG_NAME, ph.PHARMACY_NAME, adr.COUNTRY, adr.CITY, l.PRICE, l.EXPIRATION_DATE, l.QUANTITY, prod.PRODUCER_NAME, COALESCE(DESCRIPTION, 'Not Description') as description
    from local_drugs l inner join drugs d on l.drug_id = d.drug_id inner join pharmacies ph on l.pharmacy_id = ph.pharmacy_id inner join addresses adr on ph.address_id = adr.address_id
        inner join producers prod on prod.producer_id = d.producer_id;

/* afisare produse prezente in fiecare factura*/
SELECT b.bill_id,  d.drug_name, o.quantity, b.issue_date, ld.price, ld.expiration_date, (o.quantity * ld.price) TOTAL FROM orders o, bills b, local_drugs ld, drugs d, pharmacies ph
    where o.bill_id = b.bill_id and o.local_drug_id = ld.local_drug_id and ld.pharmacy_id = ph.pharmacy_id and d.drug_id = ld.drug_id ORDER BY o.bill_id ASC;

/* afisare facturi cu pretul total achitat*/
SELECT b.bill_id, MAX(b.issue_date) AS issue_date, SUM(o.quantity * ld.price) TOTAL FROM bills b, orders o, local_drugs ld
    where o.bill_id = b.bill_id and ld.local_drug_id = o.local_drug_id GROUP BY b.bill_id;
```

Informații despre Id-uri

Primary Key-urile de tip id sunt generate de baza de date după tipul AUTO-Incremet folosind secvența „id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY”

```sql
CREATE TABLE pharmacies(
    pharmacy_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    pharmacy_name VARCHAR2(20) NOT NULL,
    address_id NUMBER(4) NOT NULL,
    CONSTRAINT pharmacy_id_pk PRIMARY KEY(pharmacy_id),
    CONSTRAINT pharmacy_address_id_fk FOREIGN KEY (address_id) REFERENCES addresses,
    CONSTRAINT pharmacy_uk UNIQUE (address_id));
```

Capitolul 6. Capturi de ecran

- **Interfața grafică**

Home | Addresses | Bills | Drugs | Local Drugs | Orders | Pharmacies | Producers

Proiect Baze de date

Realizat de Dobos Cosmin

Home | Addresses | Bills | Drugs | Local Drugs | Orders | Pharmacies | Producers

Pharmacy Id	Pharmacy Name	Street	Street Number	City	Country
1	Catena	Str. Mihai Eminescu	23 B	Iasi	Romania
2	Catena	Str. Nationala	34/ 17	Bucuresti	Romania

© 2020 Copyright: Dobos Cosmin

- **Exemple cod SQL**

  - **Creare Tabele**

```
CREATE TABLE drugs(
    drug_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    drug_name VARCHAR2(20) NOT NULL,
    producer_id NUMBER,
    description VARCHAR2(100),
    CONSTRAINT drug_id_pk PRIMARY KEY(drug_id),
    CONSTRAINT drug_producer_id_fk FOREIGN KEY(producer_id) REFERENCES producers,
    CONSTRAINT drug_uk UNIQUE (drug_name, producer_id, description));


CREATE TABLE local_drugs(
    local_drug_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    pharmacy_id NUMBER,
    drug_id NUMBER,
    price NUMBER(6, 2) NOT NULL,
    expiration_date DATE NOT NULL,
    quantity NUMBER(4) NOT NULL,
    CONSTRAINT local_drug_id_pk PRIMARY KEY(local_drug_id),
    CONSTRAINT local_drug_producer_id_fk FOREIGN KEY(pharmacy_id) REFERENCES pharmacies,
    CONSTRAINT local_drug_drug_id_fk FOREIGN KEY(drug_id) REFERENCES drugs,
    CONSTRAINT price_range_ch CHECK (price > 0 and price < 1000000),
    CONSTRAINT local_drugs_quantity_range_ch CHECK (quantity > 0 and quantity < 1000000),
    CONSTRAINT local_drug_uk UNIQUE (drug_id, pharmacy_id, price, expiration_date, quantity));
```

  - **Inserare Date**

```
/* producers */
INSERT INTO producers (producer_name, address_id) VALUES ('Novartis', (select address_id from addresses where street_address = 'Champs Elysees Street'
    and street_number = '34' and city = 'Paris' and country='France'));


/* drugs */
INSERT INTO drugs (drug_name, producer_id) VALUES ('Crema de maini',
    (SELECT producer_id FROM producers p, addresses adr WHERE p.address_id = adr.address_id AND p.producer_name = 'Novartis' AND adr.street_address = 'Champs Elysees Street'
        AND adr.street_number = '34' AND adr.city = 'Paris' AND adr.country='France'));


/* local drugs */
INSERT INTO local_drugs (pharmacy_id, drug_id, price, expiration_date, quantity) VALUES (
(SELECT pharmacy_id FROM pharmacies f, addresses adr WHERE f.address_id = adr.address_id AND f.pharmacy_name = 'Catena' AND adr.street_address = 'Str. Mihai Eminescu'
        AND adr.street_number = '23 B' AND adr.city = 'Iasi' AND adr.country='Romania'),
(SELECT drug_id FROM drugs d, producers p, addresses adr WHERE d.drug_name = 'Crema de maini' and d.producer_id = p.producer_id AND p.address_id = adr.address_id
        AND p.producer_name = 'Novartis' AND adr.street_address = 'Champs Elysees Street' AND adr.street_number = '34' AND adr.city = 'Paris' AND adr.country='France'),
100, to_date('2020-08-29', 'YYYY-MM-DD'), 100);
```