**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**

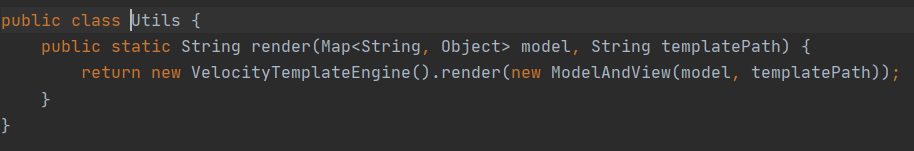
Baza de date folosită în această aplicație este SQLPlus. Pentru partea de front-end s-a folosit JavaSpark ca server web. Pentru randarea paginilor web, am folosit motorul Velocity.

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

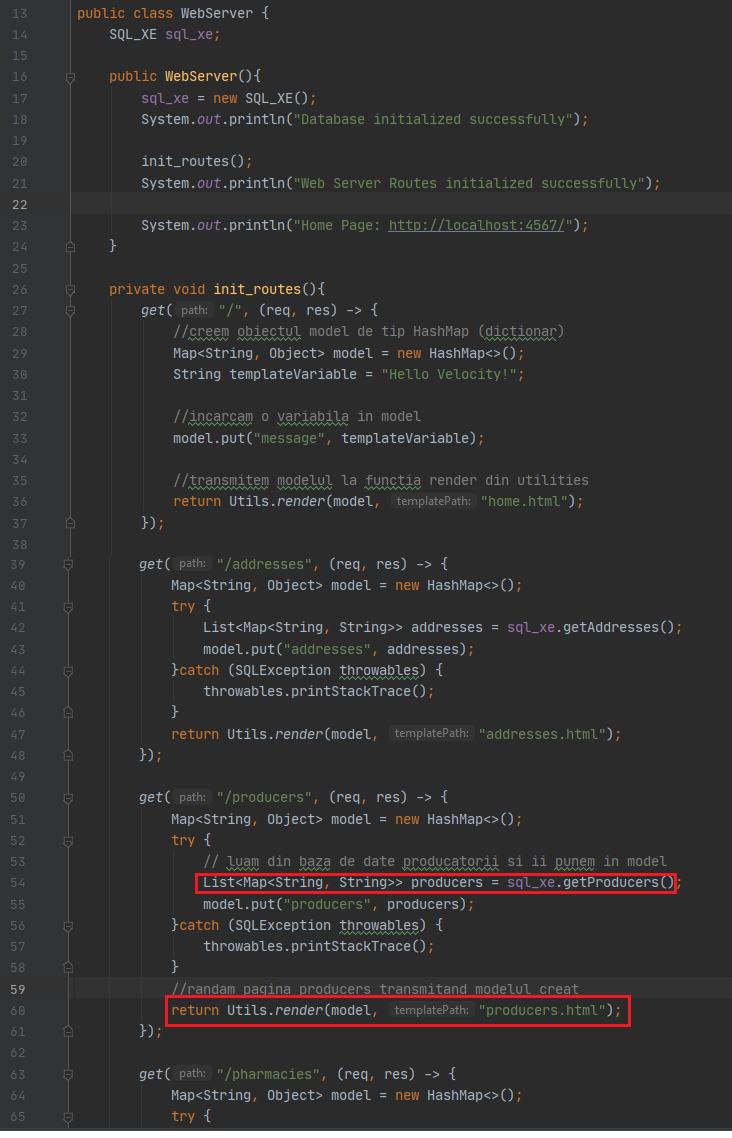
Pentru a include cele enunțate mai sus, am inclus următoarele dependențe Maven în fișierul **pom.xml**.



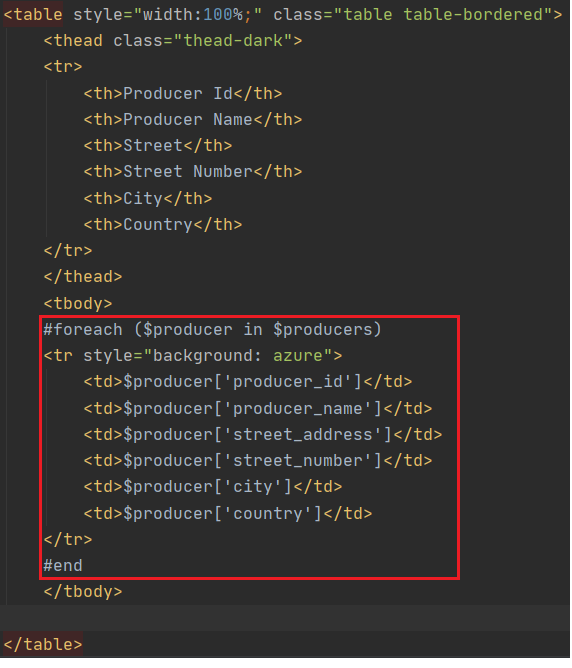
A fost creată o clasă WebServer. În constructorul său sunt definite rutele aplicației. Pentru fiecare rută sunt extrase datele din baza de date și puse într-un obiect de tip HashMap și transmise ca parametru în funcția render dintr-o clasa Utils, creată pentru a ușura randarea paginii.



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



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

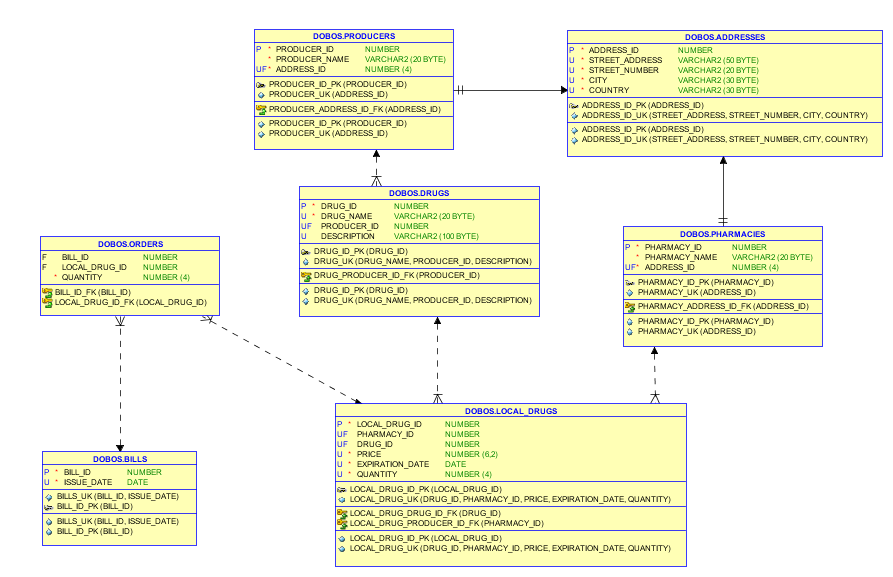


**Capitolul 3. Structura și inter-relaționarea tabelelor**

* 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.

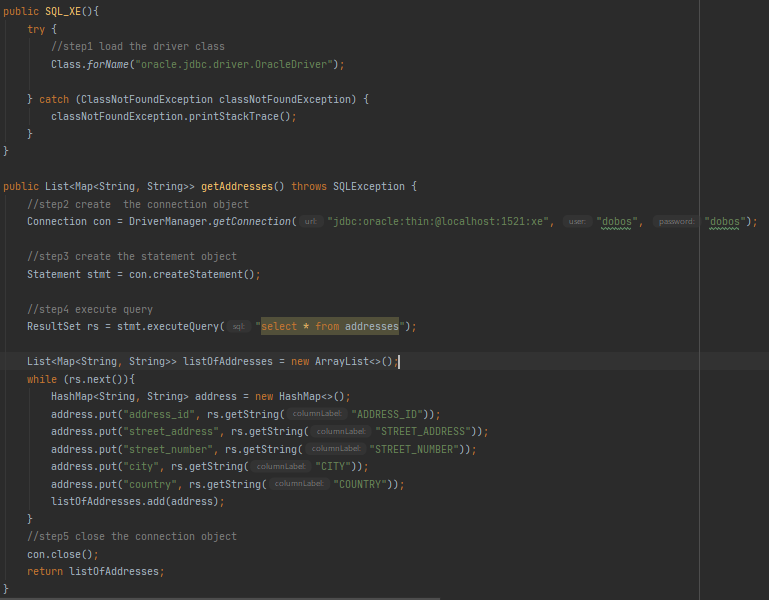
De asemenea avem și tipuri de chei de tip Unique Key in tabelele **Addresses, Producers, Drugs, Pharmacies** și **Local\_Drugs.**

**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.

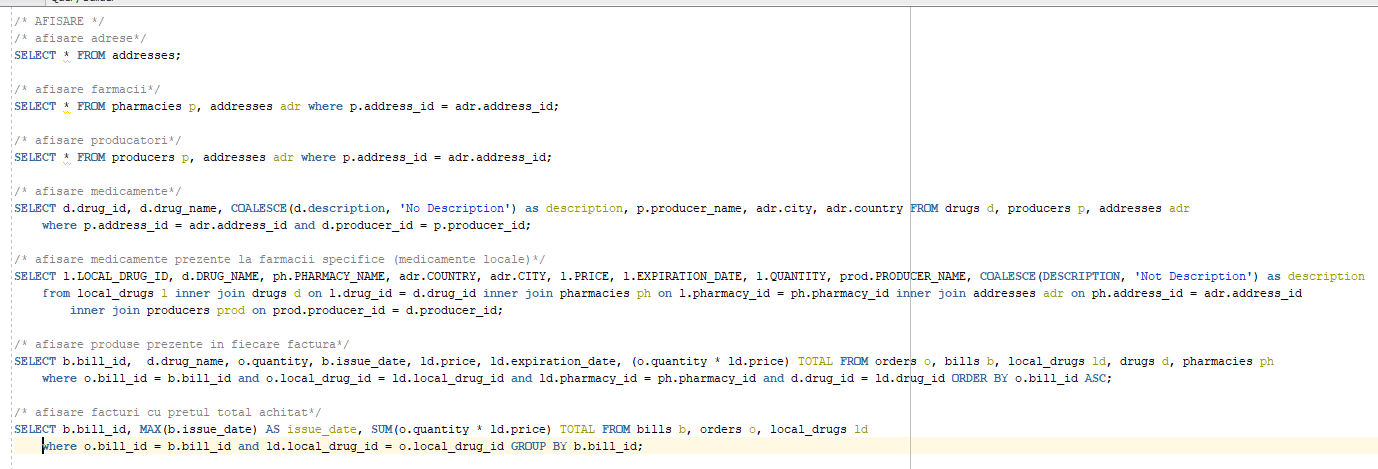
S-a creeat o clasă SQL\_XE care oferă funcționalitatea de extragere a datelor din fiecare tabelă.





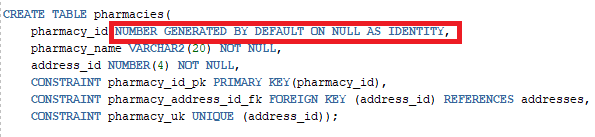
Mai întăi, în constructorul clasei se încarcă driver-ul ojdbc, precizat mai sus. Ulterior, în fiecare metodă de extragere a datelor se creează o conexiune, urmat de un obiect de tip statement prin interdiul căruia se execută query-ul și se preiau datele din baza de date. Aceste date sunt ulterior puse 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.



**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”

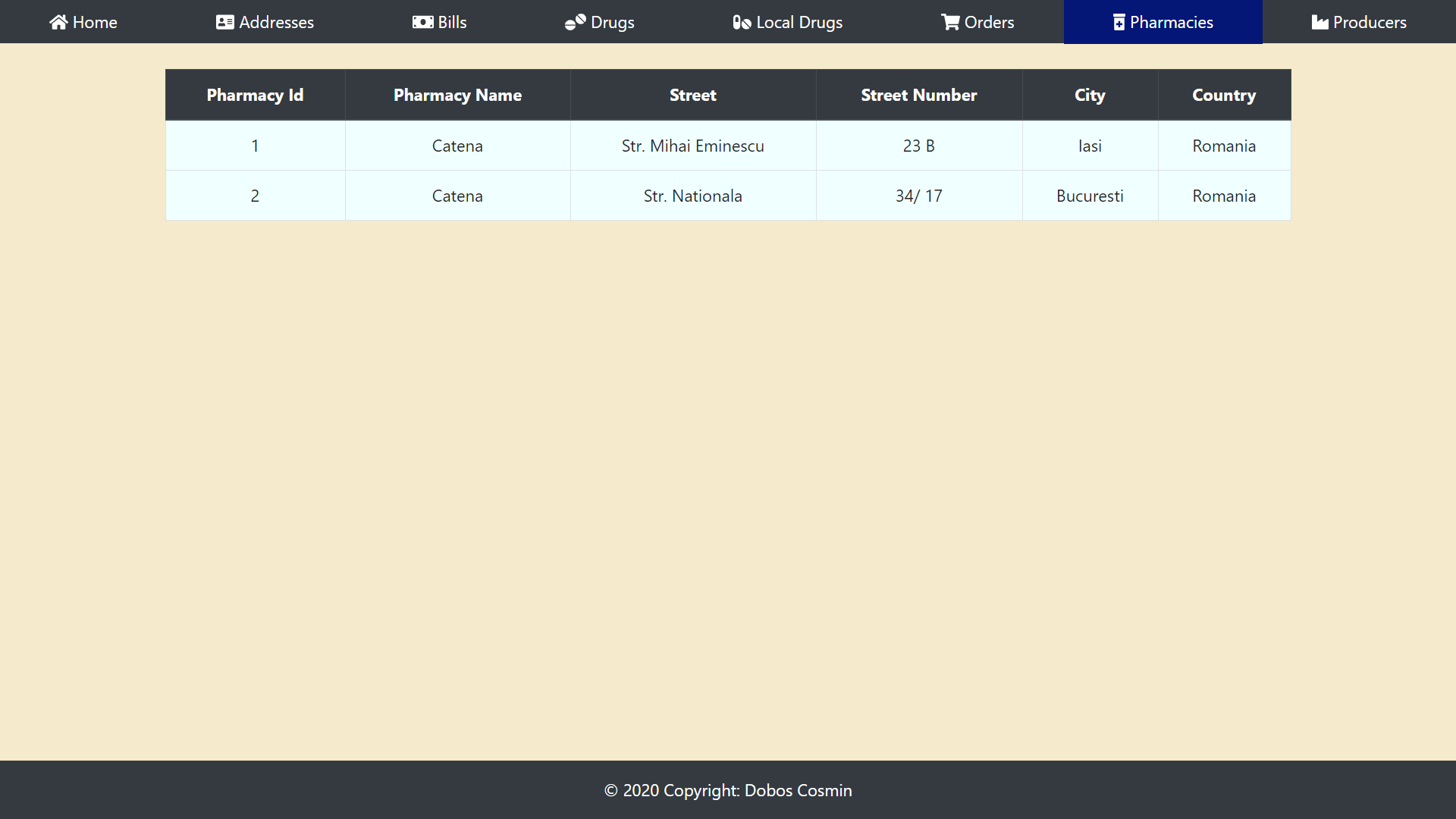


**Capitolul 6. Capturi de ecran**

* **Interfața grafică**

****





* **Exemple cod SQL**

