UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA: Tehnologia informației
CADRU DIDACTIC COORDONATOR: Catalin Mironeanu

Game Review Platform
Proiect la disciplina Baze de Date

Studenți:
Hartan Mihai Silviu 1305B
Timofti Gabriel 1305B

Cuprins

Capitolul 1. Introducere

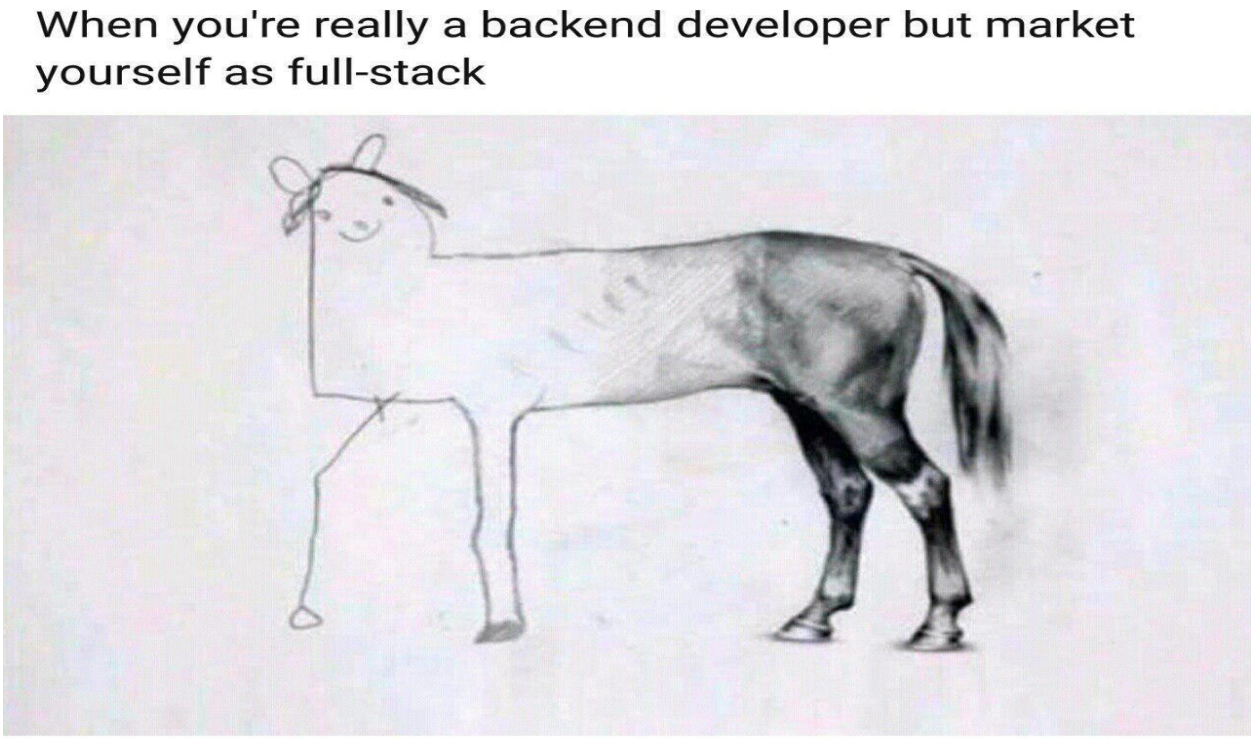
Titlu Proiect: Game Review Platform

Descrierea si motivatia proiectului:

Industria de gaming este una de miliarde de dolari, si milioane de oameni se joaca in fiecare moment. Nevoia pentru o aplicatie care sa furnizeze in timp real informatii legate de cele mai noi jocuri este una imperioasa. Platforma este accesibila pentru toti cei care sunt pasionati de aceasta industrie. Site-ul web permite alegerea mai multor roluri printre care se numara cel de moderator, admin si user normal. Jucatorii care vor sa salveze o anumita sesiune de joc, au aceasta posibilitate, si de asemenea sa scrie recenzii si comentarii. Celelalte doua roluri permit administrarea bazei de date prin intermediul functionalitatilor din front-end.

Scopul platformei noastre este permite utilizatorilor sa gestioneze si sa vizualizeze informatiile legate de jocuri.

Capitolul 2. Tehnologii folosite



Frontend:

Pentru partea de front-end am decis sa folosim:
• HTML5
• CSS
• JavaScript

Interfata cu utilizatorul este una minimalista, acesta putand accesa foarte usor informatiile de care are nevoie.

Backend:

Pentru partea de back-end am decis sa folosim:
• Java
• Spring Boot
• Oracle SQL
• Thymeleaf

Ecosistemul Spring Boot faciliteaza accesul la baza de date si manipularea informatiilor. De asemenea ne permite comunicarea dintre partea de interfata cu utilizatorul si partea de logica a aplicatiei.

Capitolul 3. Structura si organizarea aplicatiei

Am decis sa organizam aplicatia intr-o maniera MVC.

Asadar am separat aplicatia in main multe componente:

• Controllers(API)
• Models
• Services
• DAO's
• Tests
• Resources

Controllers(API):

Scopul acestei componente este de a prelua informatii provenite de la utilizatori de pe paginile web si de a-l ruta pe alte pagini in functie de flow-ul aplicatiei. Ca in orice aplicatie de tip MVC acesta doar apeleaza functionalitatile din componentele de logica(Services), conferindu-i astfel functionalitate de API. Actiunile din front-end sunt direct mapate pe functii din Controllere.

Services(Business logic):

In cadrul acestor componente se regaseste partea logica si de business aplicatiei. Functionalitatile din services sunt apelate direct de functiile din care este alcatuit API-ul. De asemenea, ele se

folosesc de elemente de Data Access pentru a manipula informatiile din baza de date.

Models(Set processing si reference processing result models):

Modelele reprezinta modalitatea prin care se mapeaza informatii abstracte provenite din exterior(fie o baza de date, fie informatii provenite dintr-un site web in cadrul aplicatiei noastre) in obiecte concrete din Java. Fiecare entitate din BD, are drept corespondent o clasa in Java.

DAO's

Ne dorim ca accesul la date sa fie facut de niste componente separate pentru a facilita eventuala interschimbare a unei noi baze de date in aplicatei. Datorita acestui design pattern separam si logica de acces si manipulare a datelor dar si izolam layer-ul de business de layer-ul de persistenta print intermediul unor clase care in final constituie un API.

Test's

Termenul de „testare unitară" se referă la testarea individuală a unor unități separate dintr-un sistem software. În sistemele orientate pe obiecte, aceste „unități" sunt de regulă clase și metode. Uneltele de testare unitară pot înregistra testele pentru ca ele să poată fi repetate ușor mai târziu (de regulă când se schimbă o parte din sistem), astfel încât dezvoltatorul să fie convins că noile modificări nu au stricat vechea funcționalitate. Fiecare functionalitate din fiecare componenta de acces a datelor este testata unitar in cadrul aplicatiei noastre.

Resources

Toate resursele statice din cadrul aplicatiei noastre (CSS, HTML templates, JS scripts, Images etc) sunt tinute intr-un folder separat.

Prin aceasta organizare, am reusit sa organizam cod-ul si arhitectura aplicatiei, facilitandu-ne foarte mult munca in echipa si impartatirea responsabilitatilor.

Pentru munca la comun si versionarea proiectului am folosit git si github.com. Utilizand mecanismul de fork & merge am reusit cu usurinta sa lucram pe proiect cu modificarile la zi.

Capitolul 4. Structura si inter-relationarea tabelelor

Baza noastră de date cuprinde un numar de 10 tabele:

- Group
- User
- Role
- RolePermission
- GameSession
- Review
- Permission
- Genre
- GenreGame
- Game

Informatii si specificatii:

User: Este identificat unic prin id generat prin autoincrement, are nume, prenume, parola hashed, email, data crearii contului, id group(Fk), id Rol(fk)

Group(Aceasta functionalitate a fost omisa in cadrul aplicatiei noastre) : fiecare group are un id unic; reprezentand o categorie: newbie, rising-star, legend, etc..; o descriere;

Rol: Exista 3 roluri --> useri normali, moderatori si admini

Permisiune: Tipuri de Permisiuni--> adauga joc, adauga comentariu, sterge comentariu, sterge joc;

Genre: Genul de joc--> action, simulation, rpg etc;

Game: Este identificat unic prin id generat prin autoincrement, are nume, descriere, data aparitie, numar jucatori, Rating(1-5), etc

Review: Este identificat unic id generat prin autoincrement, are titlu, continut(text).

GameSession: Este identificat unic id generat prin autoincrement, titlu, continut, o data.

Descrierea functionala a aplicatiei

Principalele functii ale platformei:

- Ofera Userilor ultimele stiri cu privire la o varietate larga de jocuri
- Userii isi pot descrie experienta legate de o sesiune de joc
- Userii pot adauga recenzii sau comentarii la jocuri
- Se formeaza o comunitate, unde userii au diferite roluri

Prin aceasta aplicație am incercat sa implementam vizual functionalitatea principalelor funcții de interogare a unei baze de date. Astfel putem modifica/ accesa o baza de date cu ușurința datorită bibliotecilor din Java.

Principalele funcții folosite sunt:

- Select
- Insert
- Update
- Delete

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

• Group-User: Relatie 1:n. Un grup poate avea mai multi useri. Dar un user poate apartine unui singur grup la un moment dat. Legatura se stabilesti prin campul Group_User_FK

• User-Role relatie n:1. Mai multi useri pot avea un singur rol. Si un rol poate fi detinut de mai multi useri. Legatura prin camplul Role_User_FK. Aceasta Relatie insa nu este de ajuns, pt ca fiecare tip de User poate avea diferite permisiuni.

• Relatia Role-Permission este de tipul n:n, insa este sparta in 2 relatii: 1:n si n:1 si cream o noua tabela RolePermission. Practic un rol poate avea mai multe permisiuni(ex: rolul moderator are permisiunea de a edita, cenzura, sterge review, adauga joc, etc), si o permsiune poate fi prezenta la mai multe roluri(ex: Post a review, e prezenta atat la simple user, cat si la moderator, si la admin).

• Relatia Genre-Game este de tipul n:n. Un gen de joc poate contine mai multe jocuri, iar un joc se poate incadra la randul lui in mai multe genuri. O noua tabela GenreGame este creata iar relatia este sparta in 2 relatii: 1:n si n:1.

• Relatia User-GameSession este o relatia 1:n. Un user poate adauga una sau mai multe sesiune de joc. Relatia GameSessionGame este de tipul n:1. Una sau mai multe sesiuni de joc pot fi despre un joc. Practic tabela GameSession este spargerea relatiei de tipul n:n User-Game.

• Relatia User-Review si Review-Game functioneaza pe acelasi principiu ca relatiile dintre User-GameSession si GameSessionGame. O mentiune ar fi ca tabelele GameSession si Review au cheie

primara dedicata, pentru ca un user poate posta mai multe review-uri la acelasi joc, si nu este de ajuns doar cheile straine pentru unicitate.

Diagrama Entitate Relatie

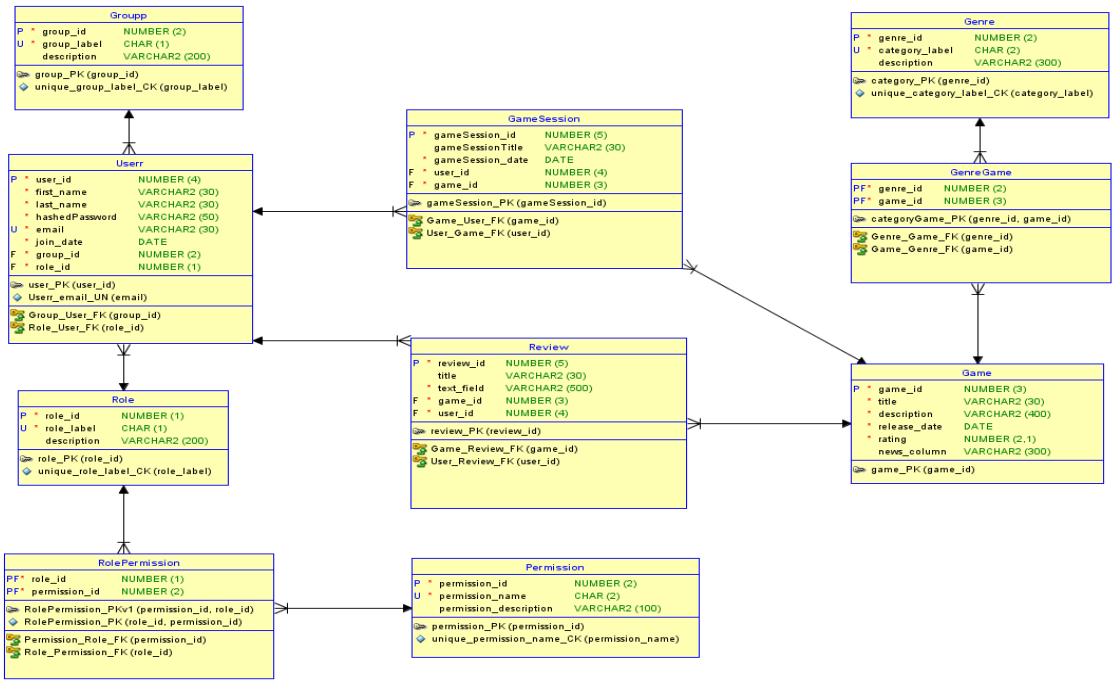
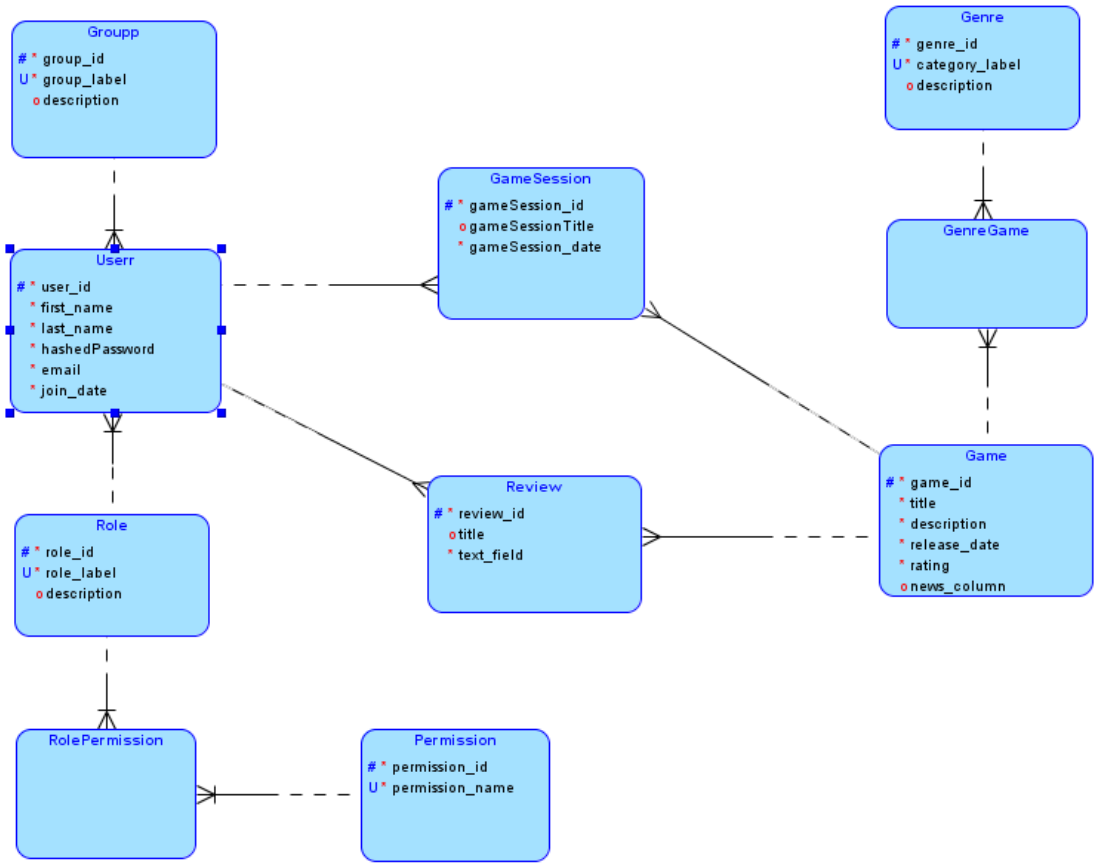


Diagrama Logica



Aplicatia se conecteaza si manipuleaza baza de date prin ajutorul obiectelor de tip DAO(Data Access Object) care se folosesc de JbdcTemplate si adnotatia @Repository pentru operatii de tip CRUD.

(Spring @Repository annotation is used to indicate that the class provides the mechanism for storage, retrieval, search, update and delete operation on objects. - documentatia oficiala).

De asemenea informatiile de configurare si de conectare la BD se afla in fisierul application.properties:

```
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
server.port=8083
spring.datasource.username=game_rev_db_v3
spring.datasource.password=bunica
```

Capitolul 5. Functionalitatea Aplicatiei

Prin aceasta aplicație am incercat sa implementam vizual functionalitatea principalelor funcții de interogare a unei baze de date. Astfel putem modifica/ accesa o baza de date cu ușurința datorita interfetei vizuale.

Principalele funcții folosite sunt:

5.1 Select

```
String sql = "SELECT * FROM game";
```

Functia are rolul de a extrage informatii din baza de date. Rezultatul expresiei de mai sus este:

Title	Description	Release	Rating	Genre	Actions
sdd	sd	2020-12-01	1.2	moba rpg	Edit Delete
sdsd	sdsd	2020-12-29	1.2	generic	Edit Delete
sdd	sdsd	2020-12-27	1.1	rpg	Edit Delete
sdsd	sdsd	2020-12-28	1.1	generic	Edit Delete
sdsd	sdsd	2021-01-05	1.3	generic	Edit Delete
Assassins Creed: Valhalla	With an action that takes place in 873 AD, the game presents an alternate histor ...	2020-11-12	4.2	action generic rpg	Edit Delete
Cyberpunk 2077	Adapted from the Cyberpunk franchise, the story takes place in dystopian Night C ...	2020-11-20	4.4	action shoting	Edit Delete
Call of Duty: Cold War	Call of Duty: Cold War is a 2020 first-person shooter video game developed by Tr ...	2020-10-30	4.2	action shoting	Edit Delete
FIFA 21	FIFA 21 is a football simulation video game published by Electronic Arts as part ...	2020-10-09	3.4	generic simulation	Edit Delete
Microsoft Flight Simulator	Microsoft Flight Simulator is a series of amateur flight simulator programs for ...	2020-07-14	4.2	simulation	Edit Delete
Watch Dogs Legion	Watch Dogs: Legion is a 2020 action-adventure game published by Ubisoft. It is t ...	2020-10-12	4.1	action	Edit Delete

```
String   sql  = "SELECT * FROM Userr WHERE EMAIL = ? and HASHEDPASSWORD = ?"
```

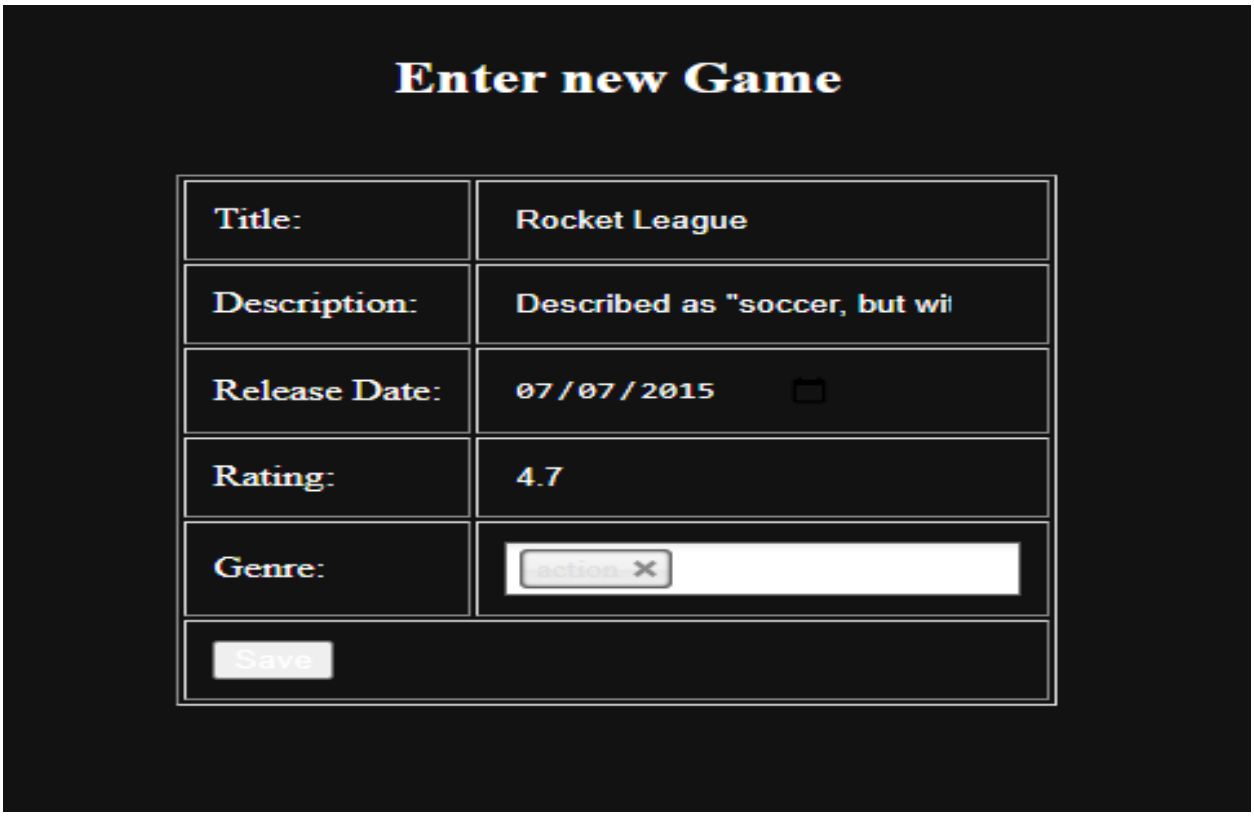
Rezultatul expresiei se foloseste pentru a verifica daca un utilizator este deja inregistrat, iar in caz afirmativ acesta se poate loga pe platforma.

5.2 Insert

Cu ajutorul functiei de mai jos se introduce un nou joc in baza de date.

```
public void insertGame(Game game) {
   // when inserting a new game we must know the autoincrement id.

   String insertSql = "INSERT INTO Game (title, description, release_date, rating, news_column) " +
         "VALUES (?, ?, ?, ?, ?)";
   int game_id = 0;
```

```
    // use KeyHolder and PreparedStatement to get the autoincrement id from oracle db
    KeyHolder keyHolder = new GeneratedKeyHolder();

    jdbcTemplate.update(conn -> {
        PreparedStatement ps = conn.prepareStatement(insertSql, new String[] { "game_id" });
        ps.setString(1, game.getTitle());
        ps.setString(2, game.getDescription());
        ps.setDate(3,  new java.sql.Date(game.getRelease_date().getTime()));
        ps.setDouble(4, game.getRating());
        ps.setString(5, game.getNews_column());
        return ps;
    }, keyHolder);
    game_id = keyHolder.getKey().intValue();

    game.setGame_id(game_id);
}
```



Functia aferenta din controller

```
@RequestMapping("/newGame")
public String viewEnterGamePage(Model model) {
    FrontPageGame game = new FrontPageGame();
    List<Genre> genres = genreService.getAllGenres();

    model.addAttribute("game", game);
    model.addAttribute("genres", genres);
```

```
    return "new_game_form";
}
```

Tabela corespondenta in HTML5 si binding-ul facut de Thymeleaf

```
<tbody>
<tr th:each="game : ${games}">
    <td><a th:text="${game.title}" th:href="@{'/game/' + ${game.game_id}}"> </a></td>
    <td th:text="${game.description}"></td>
    <td th:text="${game.release_date_string}"></td>
    <td th:text="${game.rating}"></td>
    <td th:text="${game.genre}"></td>

    <!-- action column -->
    <td>
        <a th:href="@{'/edit/' + ${game.game_id}}">Edit</a>
            
        <a th:href="@{'/delete/' + ${game.game_id}}">Delete</a>
    </td>

</tr>
</tbody>
```

```
String insertSql = "INSERT INTO Userr(first_name, last_name, hashedPassword, email, join_date, group_id,
role_id) " +
        "VALUES (?, ?, ?, ?, ?, ?, ?)";
```

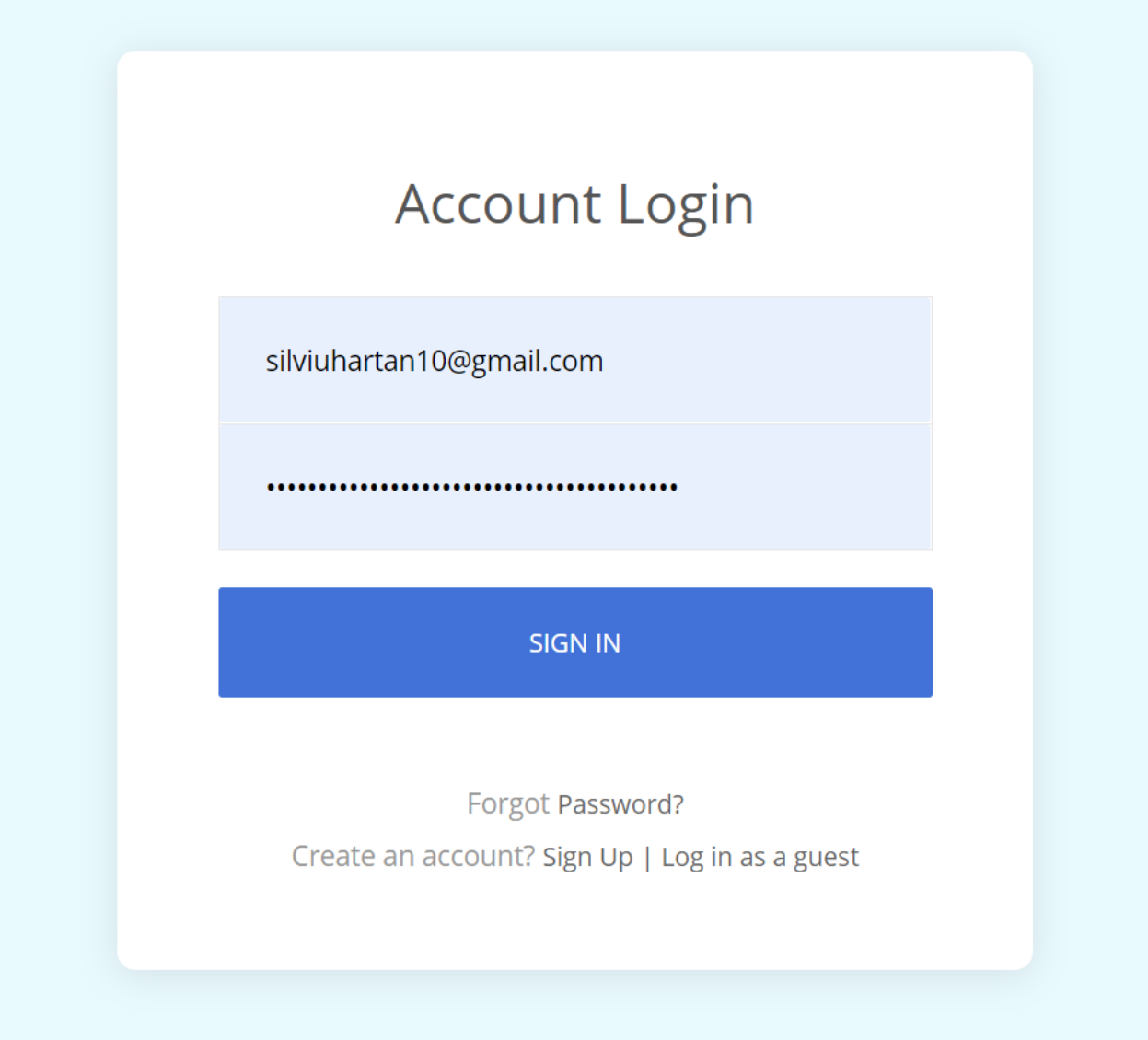
Dupa ce se executa aceasta operatie de insert, user-ul este inregistrat in baza de date.

```
public void insertUserr(Userr userr) { // for register purposes
    String insertSql = "INSERT INTO Userr(first_name, last_name, hashedPassword, email, join_date, group_id,
role_id) " +
            "VALUES (?, ?, ?, ?, ?, ?, ?)";
    int      userr_id  = 0;
    KeyHolder keyHolder = new GeneratedKeyHolder();

    jdbcTemplate.update( conn -> {
        PreparedStatement ps = conn.prepareStatement(
                insertSql,
                new String[]{ "user_id" }
        );

        ps.setString( 1, userr.getFirstName() );
        ps.setString( 2, userr.getLastName() );
        ps.setString( 3, userr.getHashedPassword() );
        ps.setString( 4, userr.getEmail() );
        ps.setDate( 5, userr.getJoinDate() );
        ps.setInt( 6, userr.getGroup_id() );
        ps.setInt( 7, userr.getRole_id() );        return ps;
    }, keyHolder );
```

```
    userr_id = keyHolder.getKey().intValue();
    userr.setUser_id( userr_id );
}
```

Functia aferenta din controller

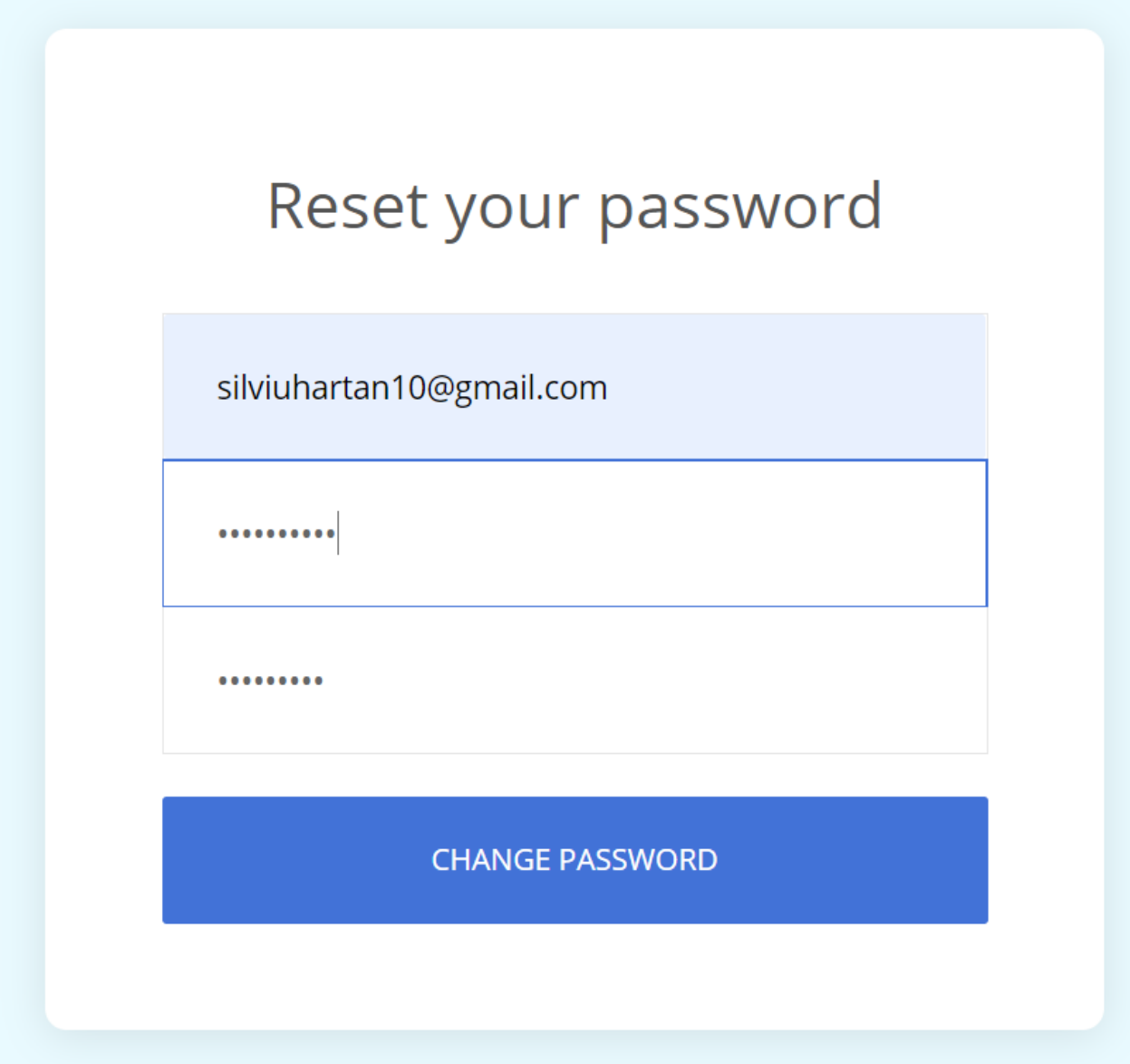
```
@RequestMapping("/sign_up")
public String signUp(Model model) {
    Userr userr1 = new Userr();
    model.addAttribute( "userr", userr1 );

    return "register_account_form";
}
```



5.3 Update

Un exemplu de update este atunci cand un user isi schimba parola din diferite motive. (si-a uitat-o sau doreste sporirea securitatii)

```
public int updatePassword(String email, String hashedPassword) {
   String   sql    = "UPDATE Userr set HASHEDPASSWORD = ? where EMAIL = ?";
   Object[] args   = { hashedPassword, email };
   int      result = 0;

   try {
      result = jdbcTemplate.update( sql, args );
   } catch ( EmptyResultDataAccessException e ) {
      System.out.println( e.getMessage() );
      return result;
   }

   return result;
}
```



5.4 Delete

Pentru a sterge un joc se foloseste functia de mai jos.

```
public void deleteGame(int game_id) {
    String sql = "DELETE from Game WHERE game_id = ?";
    jdbcTemplate.update(sql, game_id);
}
```

Actions
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete
Edit Delete