**JT HR**

**Admin**

**Echipa *ULBS20*:**

* **Bugner Ioana**
* **Cozma Alexandru**
* **Ghip Sabrina** – Cerinte personale

**Informatizarea procesului de recrutare**

**Echipa: ULBS20**

**Membrii:**

- Bugner Ioana (Gr. 231/2)

- Cozma Alexandru (Gr. 231/3)

- Ghip Sabrina (Gr. 231/1) – Cerinte personale

1. **Introducere**
2. **Cerinte functionale**

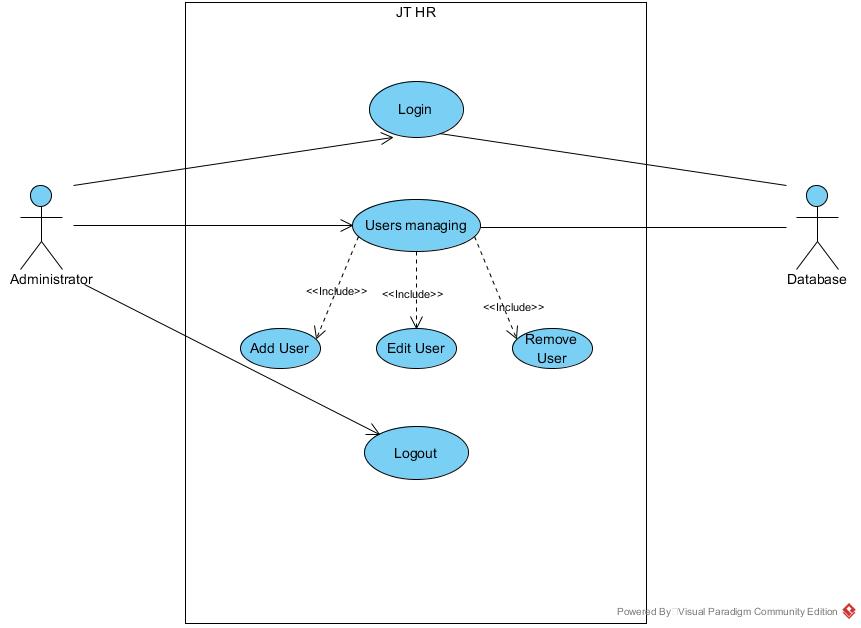
**Operatii CRUD implementate**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rol**  **Operatie** | D. General | D. Dep. | D. HR | Recruiter | Admin | Viewer |
| Add user | x |  |  |  | x |  |
| Edit user | x |  |  |  | x |  |
| Delete user | x |  |  |  | x |  |
| Add pozitie | x | x | x | X |  |  |
| Edit pozitie | X | x | x | x |  |  |
| Close pozitie |  |  |  |  |  |  |

**Membru trei – Ghip Sabrina**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rol**  **Operatie** | D. General | D. Dep. | D. HR | Recruiter | Admin | Viewer |
| Add User | X |  |  |  | x |  |
| Edit User | X |  |  |  | x |  |
| Delete User | X |  |  |  | x |  |
| Add Pozitie | X | x | x | x |  |  |
| Edit Pozitie | X | x | x | x |  |  |
| Close pozitie |  |  |  |  |  |  |

1. **Diagrame Use-Case**



**Use-Case List:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID Autor + nume** | **Rol** | **Use-Case ID** | **Use-Case Name** |
| **2** –Bugner Ioana | **Administrator** | **UC1** | **Login** |
| **1** - Cozma Alexandru | **UC2** | **Add User** |
| **3** - Ghip Sabrina | **UC3** | **Edit User** |
| **1** - Cozma Alexandru | **UC4** | **Remove User** |
| **2** –Bugner Ioana | **UC5** | **Logout** |

**Use-Case: Edit User – Ghip Sabrina**

|  |  |
| --- | --- |
| **Use Case ID:** | UC3 |
| **Use Case Name:** | Edit User |
| **Created By:** | Ghip Sabrina |
| **Date Created:** | January 18, 2016 |
| **Actors:** | Administrator |
| **Description:** | To ensure the possibility to modify(update) user information in the database |
| **Trigger:** | There is a need/demand for an update to a user information |
| **Preconditions:** | There must be at least a user in the database |
| **Postconditions:** |  |
| **Normal Flow:** | Administrator choose option to edit user  JT HR displays the list of all users  Admin selects the user wanted to be edit and clicks the edit button  JT HR displays a editable form with the user information  Admin edits the last and the first name,(or) mobile phone no,(or) email address (mandatory fields)  Admin completes the user position, (or) description, (or) the list of roles  JT HR autocompletes the username based on the last and first name (if first and/or last name were modify)  Admin. Selects to update the user with the information from the form  JT HR validates the given information  JT HR updates the user in the database  JT HR restores the user management menu |
| **Alternative Flows:** | NA |
| **Exceptions:** | 7a. in the step 7 of the normal flow, if the email address and/or phone number is/are no valid:  JT HR displays a warning message which tells that email and/or phone number is/are not valid  Use case resumes from step 5  7b. in the step 7 of the normal flow, if a user with the same username(generated) already exists:  JT HR displays a warning message which tells that email and/or phone number is/are not valid  Use case resumes from step 5 |
| **Includes:** | NA |
| **Frequency of Use:** |  |
| **Special Requirements:** | NA |
| **Assumptions:** | NA |
| **Notes and Issues:** | What’s the best way to generate the username from last and first name? |

1. **Modelul de date si alte informatii**

**Fisierele JSP (Java Server Pages) – Ghip Sabrina**

**Fisierele JSP realizate personal:**

* **AllUsers.jsp** 
  + Afisarea listei de utilizatori din baza de date intr-un tabel pe pagina web
* **editUser.jsp**
  + Afisarea interfetei corespunzatoare pentru editarea unui anumit utilizator pe pagina web

**Servleti**

**Fisiere Servlet realizate personal:**

* Afisarea tuturor utilizatorilor (AllUsersServlet.java)
* Editarea unui utilizator (EditUserServlet.java)

**Validator Email**

“EmailValidator.java” – respectarea pattern-ului a unei expresii regulate, precum emailui care in cazul nostru urmareste structura:

“nume@k.m’

* nume – trebuie sa inceapa cu o litera, cifra sau caracterul “\_”, contine mai multe cifre, litere, caractere
* trebuie sa contina neaparat caracterul “@”
* k – un camp de litere sau cifre
* m – camp format din cel putin 3 caractere

**Implementarea Bazei de Data in MySQL**

**Nume tabela: „jthr\_admin“**

**Atributele tabelei:**

* **„userid“**  - reprezinta un id unic, *“autoincrement”* care ramane ascuns pe parcursul rularii aplicatiei, avand ca scop accesarea din spatele codului a tuplurilor din baza de date
* **„username“** – este de asemenea un atribut unic la un moment de timp, format din primele 5 litere ale “firstname-ului” (sau intregul camp atunci cand acesta are o lungime mai mica de 5) si prima litera din *“lastname”.*  In cazul in care *“username-ul”* exista pentru o altă inregistrare el va continua sa adauge litere din campul *“lastname”* pana va devein unic.
* **„firstname“** – numele de familie al utilizatorului
* **„lastname“** – prenumele utilizatorului
* **„phone“**
* **„cellphone“**
* **„userfunction“** – functia utilizatorului
* **„rolelist“** – lista de roluri pe care trebuie sa o indeplineasca, dupa caz
* **„description“** – descrierea in amanunt a functiei

**„password“** – parola utilizatorului care la inceput va fi formata din concatenearea *“username-ului”* cu anul prezent “2016”

**DAO Pattern (Data Access Object)** (+ Cozma Alexandru)

“UserDao.java”

DAO este un obiect care oferă o interfață abstractă pentru un anumit tip de baze de date sau alt mecanism de persistență . Prin cererea de cartografiere cheamă la stratul de persistența , DAO oferă unele operațiuni specifice de date , fără a expune detalii ale bazei de date.

In aceasta aplicatie nu am respectat in totalitate pattern-ul de DAO intrucat nu am implementat interfata abstracta.

**Conectarea la Baza de Date**

Pentru conectarea la baza de date MySQL s-a implementat clasa *“SQLdb”*

Membri:

final String db\_driver;

final String db\_url;

final String db\_user;

final String db\_password;

Connection conn;

Statement statem;

PreparedStatement ps;

ResultSet result;

Metode:

public void Connect() – Pentru conectare

* + Se face instantarea driver-ului de MySql
  + Se incearca realizarea conexiunii prin apelarea functiei ***DriverManager.getConnection(url, user, password);***
  + Se creaza statement-ul conexiunii

public void executeSQL(String sql) – Pentru trimiterea unei cereri catre baza de date

* + Se verifica daca este o cerere de “update”, “insert” sau “delete”
    - In acest caz se pregateste statement-ul apoi se executa
  + Daca este un simplu *“query”* se executa cererea care returneaza rezultatul in membru *“result”*