Modelarea unor procese de resource management folosind oetpn

PROIECT DE DIPLOMĂ

Autor: **Ioana Petronela BOGDAN**

Conducător științific: **ȘL. dr. ing. Octavian CUIBUS**

|  |  |
| --- | --- |
| DECAN  **Prof.dr.ing. Liviu MICLEA** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof.dr.ing. Honoriu VĂLEAN** |

Autor: **Ioana Petronela BOGDAN**

Modelarea unor procese de resource management folosint OETPN

1. **Enunțul temei:** *O scurtă descriere a temei proiectului de diplomă*
2. **Conținutul proiectului:** *(enumerarea părților componente) Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Titlul capitolului 1, Titlul capitolului 2,… Titlul capitolului n, Bibliografie, Anexe.*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, alte locuri dacă este cazul*
4. **Consultanți:** *ing. Prenume Nume (dacă este cazul)*
5. **Data emiterii temei:**
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**Declaraţie pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

Subsemnatul(a) **Prenume NUME**  , legitimat(ă) cu CI/BI seria nr. , CNP ,

autorul lucrării:

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată,** din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie 2021 a anului universitar 2020-2021, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data Prenume NUME

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Titlul lucrării

Autor: **Ioana Petronela Bogdan**

Conducător științific: **ȘL. dr. ing. Octavian CUIBUS**

1. Cerințele temei:

2. Soluții alese:

3. Rezultate obținute:

4. Testări și verificări:

5. Contribuții personale:

6. Surse de documentare:

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 4](#_Toc105846463)

[1.1 Context general 4](#_Toc105846464)

[1.2 Prezentarea pe scurt a tehnologiilor folosite 4](#_Toc105846465)

[1.3 Motivația 4](#_Toc105846466)

[1.4 Conținutul lucrării 5](#_Toc105846467)

[2 Obiectivele proiectului 6](#_Toc105846468)

[2.1 Obiectivul principal 6](#_Toc105846469)

[2.2 Obiective funcționale 7](#_Toc105846470)

[2.3 Obiective non- funcționale: 8](#_Toc105846477)

[2.3.1.1 **Cerințe non-funcționale** 8](#_Toc105846478)

[2.3.2 Accesabilitatea 8](#_Toc105846479)

[2.3.3 Performanța 9](#_Toc105846480)

[2.3.4 Mentenanța 9](#_Toc105846481)

[2.3.5 Independență 9](#_Toc105846482)

[2.3.6 Fiabilitate 9](#_Toc105846483)

[2.4 Cerințe funcționale 9](#_Toc105846484)

[**2.4.1.1** **Tabel cerințe funcționale ale sistemului realizat** 10](#_Toc105846485)

[2.5 Specificații 10](#_Toc105846486)

[3 Studiu bibliografic 11](#_Toc105846487)

[3.1 Studiul instrumentelor pentru testarea și validarea serviciilor 11](#_Toc105846488)

[3.2 Instrumente ce vizează testarea performanței serviciilor Web 11](#_Toc105846489)

[3.3 Concepte de bază 11](#_Toc105846490)

[3.3.1 Arhitectura REST 11](#_Toc105846491)

[3.3.1.1 Arhitectura Client-Server [19] 11](#_Toc105846492)

[3.3.2 OETPN (Object Enhanced Time Petri Nets) 12](#_Toc105846493)

[3.3.2.1 [16] Rețea OETPN 12](#_Toc105846494)

[3.4 Tehnologii și concepte utilizate pentru dezvoltarea aplicației web 14](#_Toc105846495)

[3.4.1 Spring Boot 14](#_Toc105846496)

[3.4.1.1 Arhitectura Spring 15](#_Toc105846497)

[3.4.2 React JS 15](#_Toc105846498)

[3.4.2.1 Arhitectura MVC 16](#_Toc105846499)

[3.4.2.2 Arhitectura React, bazată pe servicii și componente 16](#_Toc105846500)

[3.4.3 SQL Workbench 16](#_Toc105846501)

[3.4.3.1 Operațiile CRUD efectuate de către SQL Server 17](#_Toc105846502)

[3.4.4 Bootstrap 17](#_Toc105846503)

[3.4.5 JSON- JavaScript Objectiv Notation 17](#_Toc105846504)

[4 Analiză și Fundamentare Teoretica 19](#_Toc105846505)

[5 Proiectare de Detaliu și Implementare 19](#_Toc105846506)

[5.1 Descrierea ficărei pagini prezente în aplicația web 19](#_Toc105846507)

[5.1.1 Pagina APLICANT 19](#_Toc105846508)

[5.1.2 Pagina HR Manager 20](#_Toc105846509)

[5.1.2.1 Descriere particularități ale paginii OETPN 20](#_Toc105846510)

[5.2 Cazuri de utilizare 22](#_Toc105846511)

[5.2.1 Actorii aplicației 22](#_Toc105846512)

[**5.2.1.1** **Caz de utilizare pentru Admin** 23](#_Toc105846513)

[**5.2.1.2** **Caz de utilizare pentru Aplicant** 23](#_Toc105846514)

[**5.2.1.3** **Caz de utilizare pentru HR Manager** 24](#_Toc105846515)

[5.2.2 Descrierea tuturor cazurilor de utilizare 24](#_Toc105846516)

[**5.2.2.1** **Caz de utilizare 1** 24](#_Toc105846517)

[**5.2.2.2** **Caz de utilizare 2** 25](#_Toc105846518)

[**5.2.2.3** **Caz de utilizare 3** 26](#_Toc105846519)

[**5.2.2.4** **Caz de utilizare 4** 27](#_Toc105846520)

[5.3 Structura aplicației(diagrama bloc cu frontend, backend+db, protocol de comunicare intre ele) 29](#_Toc105846521)

[5.3.1 Proiectare front-end 29](#_Toc105846522)

[5.3.1.1 React Architecture [20] 29](#_Toc105846523)

[30](#_Toc105846524)

[5.3.1.2 Structura front-end-ului 30](#_Toc105846525)

[5.3.2 Proiectare pentru back-end 30](#_Toc105846526)

[5.3.2.1 Structura back-end 31](#_Toc105846527)

[5.3.3 Structura bazei de date 32](#_Toc105846528)

[5.3.3.1 Tabela User 33](#_Toc105846529)

[5.3.3.2 Tabela Email 33](#_Toc105846530)

[5.3.3.3 Tabel UserJob 33](#_Toc105846531)

[5.3.3.4 Tabela Form 34](#_Toc105846532)

[5.3.3.5 Tabela FileInfo 34](#_Toc105846533)

[5.3.3.6 Tabela Job 35](#_Toc105846534)

[5.4 Implementare (!! Ss cu aplicația si cod relevant ) 35](#_Toc105846535)

[6 Testare și validare 36](#_Toc105846536)

[6.1 Rezultate obținute 36](#_Toc105846537)

[6.1.1 Testarea funcțională a serviciilor Web 36](#_Toc105846538)

[6.1.2 Testare înregistrare și logare utilizatori 37](#_Toc105846539)

[6.1.2.1 Validare login 37](#_Toc105846540)

[6.2 Direcții de dezvoltare 37](#_Toc105846541)

[7 Manual de Instalare și Utilizare 38](#_Toc105846542)

[7.1 Instalarea tuturor resurselor necesare aplicației 38](#_Toc105846543)

[7.2 Manual de utilizare 39](#_Toc105846544)

[7.2.1 Înregistrare și logare în aplicație 39](#_Toc105846545)

[7.2.1.1 Pagina principală a aplicației 39](#_Toc105846546)

[7.2.1.2 Înregistrare user nou în sistem 39](#_Toc105846547)

[7.2.1.3 Pagină login 40](#_Toc105846548)

[7.2.2 Detalierea funcționalităților 40](#_Toc105846549)

[7.2.2.1 Pagină pentru vizualizarea statusului aparținând aplicanților 40](#_Toc105846550)

[7.2.2.2 Pagina principală a utilizatorului HR Manager 41](#_Toc105846551)

[7.2.2.3 Pagină trimitere email 41](#_Toc105846552)

[7.2.2.4 Pagină anunțuri job-uri 42](#_Toc105846553)

[8 Concluzii 44](#_Toc105846554)

[9 Bibliografie 45](#_Toc105846555)

# Introducere

## Context general

Noțiunea de sistem software reprezintă un concept foarte des întâlnit în viețile noastre de zi cu zi. De-a lungul timpului, această tehnologie a evoluat într-un ritm alert. Acest fapt, a dus la necesitatea creării unor sisteme tot mai performante, care reușesc cu ușurință să țină pasul cu cerințele fiecăruia dintre noi.

Ritmul continuu în care s-a dezvoltat resursa software are ca si merit un acces mai facil si mai rapid la orice categorie de interes.

Cu ajutorul tehnologiei, lucrările noastre sunt realizate mai rapid, economic si convenabil. Fie ca vorbim de acasă sau la birou, tehnologia ne face mai eficienți și mai productivi.

Proiectul propus spre realizare este constituit dintr-o aplicație web care are la bază management pentru întregul proces de recrutare pentru un job. Acesta vine atât in sprijinul aplicanților, cât și în sprijinul angajaților de la departamentul de Resurse Umane. Întreaga aplicație este realizată astfel încât să ofere fiecărui utilizator o interfață prietenoasă, ușor de navigat și totodată una optimă pentru fiecare nevoie.

## Prezentarea pe scurt a tehnologiilor folosite

Întreaga aplicație realizată se bazează doar pe componente software, iar acestea sunt dupa cum urmează:

* MySQL Workbench- constituie baza de date, acolo unde toate datele utilizatorilor care folosesc aplicația software sunt stocate. Comunicarea între FrontEnd si BackEnd se realizareză prin servicii diferite.
* Partea de backend este realizată in limbajul de programare Java, cu ajutorul framework-ului Spring Boot.
* Frontend-ul aplicației este construit cu ajutorul framework-ului React JS. Acesta este constituit din acea interfață grafică, ce este vizibilă oricărui client.

## Motivația

Un motiv al alegerii acestei teme, a venit din dorința de a aprofunda mai mult acest domeniu al aplicațiilor web, un domeniu in continuă dezvoltare si cu un mare impact asupra fiecăruia dintre noi. Analizând mai multe platforme software, am constatat că majoritatea acestora se ocupa cu managementul intern al unei companii, ci nu pe întregul proces de recrutare. Dezvoltarea unui astfel de software, atrage după sine minimizarea sarcinilor manuale care necesită timp. În tot acest timp, HR Manager-ul poate să se concentreze pe atragerea celor mai potriviți candidați, pentru posturile disponibile.

## Conținutul lucrării

În acest capitol, am realizat prezentarea per ansamblu atât a naturii teoretice, cât și a celei funcționale a acestei lucrări. In capitolele care urmează, vor fi expuse o serie de concepte care au dus la conturarea întregului proiect. Acestea sunt precum urmează:

* **Capitolul 1- Introducere-** Primul capitol reprezintă în linii mari atât natura funcțională cât și cea funcțională a lucrării.
* **Capitolul 2- Obiectivul Proiectului –** Acest capitol va descrie principalele obiective pe care dorește să le atingă proiectul, impreună cu detaliile semnificative care vor servi ca bază pentru construcția proiectului înca de la primele faze
* **Capitolul 3- Studiu Bibliografic-** Acest capitol constă in reexaminarea materialului bibliografic existent cu privire la subiectul care urmează a fi studiat și vine în sprijinul realizării proiectului. Seria de documentări din revistele de specialitate a fost necesară a fi realizată, asfel conturându-se întregul ansamblu.
* **Capitolul 4- Analiză și Fundamentare Teoretică-** În acest capitol se va analiza atât proiectarea, cât și implementarea proiectului. În același capitol se vor descrie pe scurt și detaliile care țin de tehnologiile utilizate pe parcursul realizării proiectului.
* **Capitolul 5- Proiectare de Detaliu si Implementare-** Pe parcursul acestui capitol, se va detalia arhitectura conceptuală a proiectului. Acest capitol va conține detaliile care au stat la baza elaborării codului aplicatiei, schemele descriptive, dar si diagrama bazei de date.
* **Capitolul 6- Testare, Validare și Evaluare-** Acest capitol va conține o serie de moduri prin care s-au realizat testăriile pentru fiecare componentă prezentă în proiect. Pentru a fi pusă în funcțiune, o aplicație are nevoie de mai multe validări pentru a fi considerată potrivită pentru a fi pus în producție
* **Capitolul 7- Manual de Instalare si Utilizare-** În cadrul acestui capitol se vor evidenția pașii necesari pentru instalarea si utilizarea aplicației efectuate, precum si configurările aferente acesteia.
* **Capitolul 8- Concluzii-** Acest capitol este predestinat concluziilor deduse dupa efectuarea aplicației, dar și posibilele îmbunătățiri care ar putea fi aduse aplicației

# Obiectivele proiectului

În cadrul acestui capitol, se vor prezenta toate obiectivele propuse pentru realizarea intregului sistem software. Scopul acestei aplicații fiind acela de a facilita întregul proces de recrutare pentru un job dorit. Aceasta aplicație este compusă din 2 părți: o parte care conține serverul (Java Spring) și cea de a doua parte, partea de client (React JS). Toate acestea vor atrage spre sine o îmbunătățire a întregului proces, atât intern cât și extern.

## Obiectivul principal

Principalul obiectiv al acestei aplicații este de a implementa un sistem care să fie util și totodată să mențină mereu in legătura aplicatul, cel care dorește să aplice la un job, cu partea de resurse umane a unei anumite firme. Pentru îndeplinirea acestora, sistemul creat va pune la dispoziția utilizatorului o aplicație software. În momentul în care un utilizator va deschide aplicația și se conectează cu credențialele specifice, acesta va fi introdus automat ca si utilizator în aplicație. Acest proces de selectare a candidaților necesita mai multe etape si totodată mai multe date care ar trebui să fie accesate. În această privință, realizarea unei aplicații web, este cea mai benefică opțiune.

Principalul avantaj pe care îl deține această aplicatie software, este acela că faciliteaza tot procesul de recrutare pentru un post dorit. Astfel va exista o buna comunicare între aplicant și departamentul de resurse umane, care se ocupă de întregul proces de recrutare.

Un alt avantaj este reprezentat de faptul că la sfârșitul aplicării pentru un job, partea de HR-ul, va putea genera o ierarhie a aplicanților în funcție de CV-ul încărcat de către aceștia.

Prin această implementare, se urmărește dezvoltarea unei aplicații care sa fie in același timp una concretă, dar și prietenoasă cu utilizatorul.

Sistemele software se caracterizează atât prin funcționalitate(ceea ce realizează sistemul), cat si prin non-funcționalitate(comportamentul sistemului cu privire la unele atribute observabile). Ambele aspecte sunt importante in dezvoltarea unui software.

## Obiective funcționale

Obiectivele funcționale au rolul de a descrie interacțiunile prezente intre sistem si mediu acestuia.

Principalele funcționalități implementate in cadrul aplicație sunt următoarele:

* Posibilitatea unui user de a își crea un cont în aplicație
* Posibilitatea ca în momentul in care un utilizator și-a creat un cont, si s-a logat, să poată intra in cont si să poată aplica pentru job-ul dorit
* Adăugarea joburilor disponibile împreună cu data până la care se poate aplica de catre admin
* Ștergerea unui job de catre admin
* Actualizarea unui job de catre “admin”
* Vizualizarea tuturor joburilor
* Trimitere email catre aplicant pentru cazul de accept/respins
* Realizarea unui rating al aplicanților
* Existența a doua roluri pentru utilizatori
* Vizualizare date cu privire la fiecare aplicant
* Cu ajutorul interfeței web prezentate, utilizatorul poate să încarce datele cu privire la job-ul dorit.
* Utilizatorul poate naviga facil prin toate paginile aplicației

**Posibilitatea unui user de a se înregistra în aplicație**

Cu ajutorul acestei funcționalități, utilizatorul are posibilitatea de a-și crea cont, introducând datele personale în câmpurile prezente.

**Login si Logout**

Pentru a putea efectua operații in aplicația existenta, toți utilizatorii trebuie in primul rând să fie autentificati. Întregul proces se realizeaza prin introducerea de către fiecare utilizator a unor credentiale (email, parolă). Aceste credențiale provin de la pasul anterior al înregistrării. O alta funcție prezentă aici, este aceea de logout a utilizatorului.

**Existența a trei roluri pentru utilizatori**

In cadrul acestei funcționalități, vom avea prezente doua tipuri de utilizator: **HR Manger** (membrii echipei de HR al firmei prezentate), **user** (aplicantul care doreste să aplice pentru job-ul dorit), **admin** (cel care acces la întreaga gestiune a informațiilor aplicației)

**Vizualizare date cu privire la fiecare aplicant, impreuna cu realizarea operațiilor de baza (CRUD) pentru fiecare anunț in parte**

Utilizatorii care detin rolul de “admin” au posibilitatea de a vizualiza intregul panou cu aplicanti, la ce job au aplicat, pot adauga o nota specifica CV-ului incarcat si pot face o ierarhie a tuturor acestora.

**Trimitere e-mail cu detaliile cu privință la acceptarea sau respingerea pentru postul selectat**

Fiecare aplicant va primi un mail în care este informat cu privință la statusul aplicării.

## Obiective non- funcționale:

Cerințele non-funcționale sunt acele comportamente ale sistemului cu privire la unele atribute observabile. Acestea descriu toate capacitățile sistemelor, dar si a serviciilor prezente intr-o aplicatie.

Diagram

Description automatically generated

#### **Cerințe non-funcționale**

### Accesabilitatea

* Un sistem trebuie să fie unul accesibil, adică un sistem care garantează faptul ca toate persoanele cu o gamă cât de larga de capacități pot accesa facil informațiile prezente in aplicație, iar această aplicație este disponibilă pe Internet. Scopul accesabilității software este acela de a asigura disponibilitatea si capacitatea de utilizare a aplicație software pentru o gamă cât mai largă de utilizatori.

### Performanța

* această cerință are ca și scop măsurarea vitezei și eficienței sistemului realizat.

### Mentenanța

* constă în inspectarea frecventă a aplicației. Scopul acestei inspectări este acela de a preveni situațiile de nefuncționare a anumitor componente și totodată remedierea defectelor constante.

### Independență

* în cadrul acestei cerințe se asigură faptul că aplicația va avea doar dependințe proprii

### Fiabilitate

* Este reprezentată de acea abilitate a sistemului de a reuși sa îndeplinească toate funcțiile cerute în condițiile stabilite.

## Cerințe funcționale

|  |  |  |
| --- | --- | --- |
| Identificator | Descriere a cerinței funcționale | Tipul utilizatorului |
| CF1 | Creare cont/Conectare către un cont existent | Toți utilizatorii |
| CF2 | Gestionare conturilor pentru **utilizatori** | Admin |
| CF2.1 | Gestionare conturi pentru **Aplicant** | Admin |
| CF2.2 | Gestionare conturi pentru **personalul HR** | Admin |
| CF2.3 | Gesionare conturi pentru **Admin** | Admin |
| CF3 | Gestionarea procesului de aplicare către un job dorit | HR manager |
| CF3.1 | Adăugare job | HR manager |
| CF3.2 | Ștergere job | HR manager |
| CF3.3 | Actualizare detalii privind un job | HR manager |
| CF3.4 | Vizualizare joburi | HR manager, Aplicant |
| CF3.5 | Alegerea unui job dorit | Aplicant |
| CF3.6 | Completarea unui form cu date personale si incarcarea CV-ului aferent aplicantului | Aplicant |
| CF4 | Administrarea tututor aplicanților | HR manager |
| CF4.1 | Vizualizarea CV-urilor aferente fiecărui aplicant | HR manager |
| CF4.2 | Acordarea unei note de la 1-10, in funcție de CV, pentru fiecare aplicant si ordonarea acestora într-un tabel | HR manager |
| CF4.3 | Vizualizarea unui grafic care indica numărul de aplicanți către un anumit job | HR manager |
| CF4.4 | Vizualizarea numărului total de aplicanti respinsi | HR manger |
| CF4.5 | Vizualizarea numărului total de aplicanti acceptați | HR manger |
| CF4.6 | Trimitere mail pentru acceptare/respingere pentru job | HR manger |
| CF4.7 | Vizualizare tabel cu datele personale ale aplicaților | HR manger |
| CF4.8 | Vizualizarea pe baza unei retele OETPN a progresului in procesul de aplicare a fiecărui user | HR manger |

#### **Tabel cerințe funcționale ale sistemului realizat**

## Specificații

# Studiu bibliografic

O altă etapă cheie în realizarea acestei lucrări este constituită de literatura bibliografică în sine.

## Studiul instrumentelor pentru testarea și validarea serviciilor

## Instrumente ce vizează testarea performanței serviciilor Web

## Concepte de bază

### Arhitectura REST

Arhitectura REST (Representational State Transfer) reprezintă un stil arhitectural software predestinat sistemelor hipermedia distribuite. XML. Un exemplu de astfel de sistem ar fi cel Word Wide Web. Cu ajutorul acesteia, se dezvoltă cele mai multe aplicații web. Se vor realiza mesaje între client-server, însă starea acestora nu va putea fi reținută. Datele între cei care comunică se transmit folosind protocolul HTTP și sunt reprezentate în format XML sau în orice alt format.

Diagram

Description automatically generated

#### Arhitectura Client-Server [19]

Așa cum ne este specificat în [11], principalele metode ale serviciilor REST sunt:

* **PUT:** va concepe o nouă resursă
* **POST:** realizează o actualizare pentru o nouă resursă. Este folosită pentru a afla dacă serverul acceptă o entitate înglobată în cerere.
* **GET:** citește o resursă, dar nu o modifică, realizeaza o cerere de accesare cu privire la accesarea unor informații
* **DELETE:** are ca effect stergerea unei resurse

Printre aspectele esențiale amintite în cadrul [11], se enumără:

* Față de un model SOAP, acest stil are la baza standardele Web si HTTP
* Clienții au la dispoziție mai multe tehnici legate de caching care au ca scop creșterea de scalabilitate, dar și de performanță
* Sunt reprezentări diferite pentru resurse, cum ar fi: XML, JSON, dar pot fi definite chiar si de un user

Exemple de adnotări prezente în această arhitectură:

### OETPN (Object Enhanced Time Petri Nets)

Implementarea acesteia a fost realizata cu ajutorul limbajului de programare Java.

Chiar dacă noțiunea de OEPTN nu este des folosită, acasta este un real ajutor in detalierea dezvoltării de orice aplicație. Așa cum ne este explicat în [16] modelele OETPN, pot fi realizate să fie cât mai aproape de gândirea umană, dar totodată și mai aproape de programele în care urmează să fie implementate. OETPN poate fi folosit pe parcursul tuturor etapelor de dezvoltare a unei aplicații.

Exemplu de rețea **OETPN** împreună cu principiul după care funcționează o astfel de rețea:

Diagram, schematic

Description automatically generated

#### [16] Rețea OETPN

Se consideră:

* u,v – locații de intrare
* y ieșire
* x1,x2,x3 locații
* t\_1, t\_2, t\_3 tranziții

**Grd** reprezintă acel set de condiții de execuție pentru tranziții, bazate pe locațiile de intrare.

**Tranziția t\_1:**



Tranziția t\_1 va putea transmite jetonul mai departe doar dacă în x1 vom avea jeton. Odată ce locația x1 va avea jeton, tranziția t\_1 se va valida, iar locația x2 va lua valoarea locației x1+ intrarea u.

**Tranziția t\_2:**



Comportamentul prezent la tranziția 1 este același ca și în acest caz.

**Tranziția t\_3:**



Validarea acestei tranziții se realizează în momentul în care jetonul a ajuns in locația x3. După această validare, unul dintre jetoane va merge in locația x1, iar celălalt in ieșirea y .

## Tehnologii și concepte utilizate pentru dezvoltarea aplicației web

### Spring Boot

Framework-ul Spring se numără printre cele mai populare framework-uri de tipul open source și se adresează aplicațiilot de tipul Java enterprise.

Acest framework, acceptă operații CRUD (Create, Read, Update, Delete), operații care sunt considerate necesare pentru implementarea oricărei aplicații. Toate aceste operații ajută aplicația pentru a avea o stocare persistentă (orice dispozitiv de stocare a datelor care chiar si dupa ce dispozitivul este oprit, se păstreaza alimentarea).

Seria de pași pentru folosirea acestui framework ar fi:

* Crearea de entități(@Entity)- reprezintă un obiect persistent care ne informează cu privire la faptul că o clasă adnotată este un depozit.
* Crearea de depozite(@Repository)- extind interfața JPA Repository și au rolul de depozita specificația metodelor care vor fi implementate ulterior. Acestă adnotare poate fi de două feluri: CRUD(Create, Read, Update, Delete) sau JPA.
* Crearea de controllere(@Controller)- rol în a acumula implementările operațiilor dorite, dar și se a face o mapare către serviciile REST, folosindu-se de metodele implementate in clasele care aparțin Service-ului.
* @Service- adnotare care ne pune la dispoziție toată logica necesară pentru gestiunea datelor

Pentru transferul datelor din partea de BackEnd spre partea de FrontEnd se folosesc următoarele adnotații:

* @PostMapping- rol în inserarea anumitor date in baza de date
* @GetMapping- obținem toate datele dintr-un anumit câmp din baza de date
* @PutMapping- realizăm update informațiilor din baza de date
* @DeleteMapping- ștergem date din baza de date

Realizarea legăturilor între tabelele bazei de date:

* Crearea de celor patru tipuri de legături între entități:
* One-to-One(@OneToOne)
* One-to-Many(@OneToMany)
* Many-to-One(@ManyToOne)
* Many-to-Many(@ManyToMany)

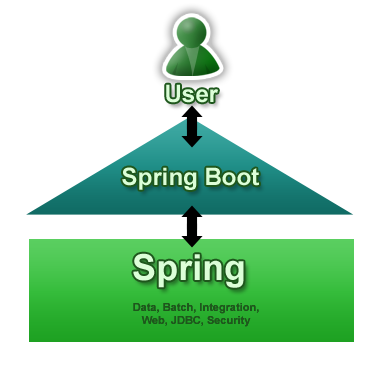
Direcția acestor legături poate fi atât unidirectională, cât și bidirectional. Împreună cu aceste tipuri de legături, vom avea nevoie si de atributul mappedBy cu rol de a indica acea proprietate de acolo de unde este referită.

Pentru a asigura o persistență între legăturile dintre date și obiecte, vom folosi și adnotările @Id (realizează cheia primară), @Column (mapează un câmp din tabelă, cu o anumită proprietate a unei clase), @Query (scop în asigurarea implementării unei interogări).

@GeneratedValue- rol in stabilirea ordinii generării de id-uri în cadrul cheii primare

@Column- asocierea unui câmp al clasei către o coloană din tabelă

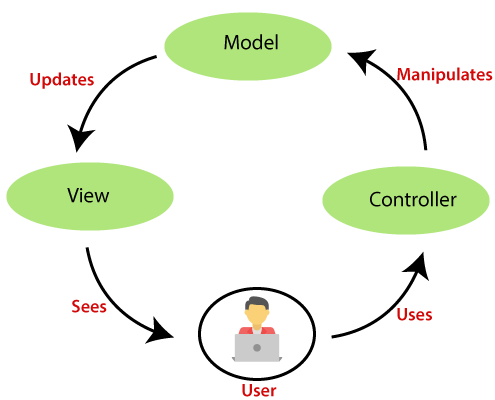
Integrarea cu ajutorul JPA va crea funcționalități mai puternice oricărei aplicații.



#### Arhitectura Spring

### React JS

După cum se specifică si în [7], React JS este o bibliotecă din categoria JavaScript de tipul open source. Cu ajutorul acestei biblioteci, se facilitează întregul proces de construție de interfețe utilizator, interactive. În cadrul acestei biblioteci, se regaseste și componenta MVC(Model-View-Controller). Acest framework este printre cele mai folosite până în present și împletește cu ușurință performanța, complexitatea, dar și accesibilitatea.



#### Arhitectura MVC

Printre caracteristicile care sunt de remarcat se numără:

* Componente- întregul cod este format din entități numite componente. Toate aceste componente pot fi redate pe un anumit element din DOM, cu ajutorul bibliotecii React DOM
* DOM virtual- model de obiect de document virtual (DOM). Cu ajutorul acestei componente, React va creea pentru structura de date, in memorie, o memorie cache.

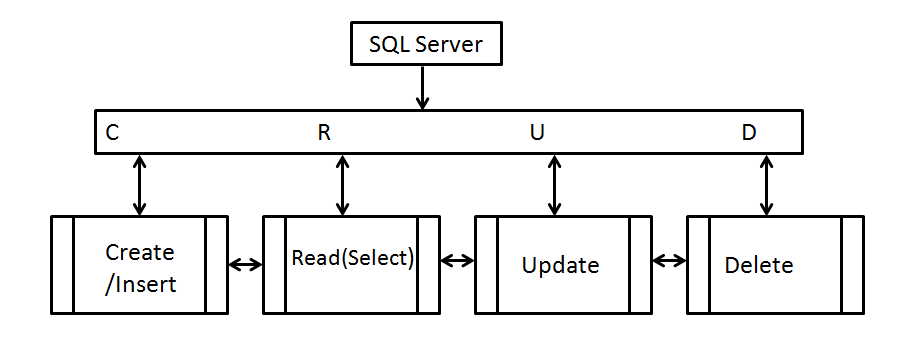
Graphical user interface, text

Description automatically generated

#### **Arhitectura React, bazată pe servicii și componente**

### SQL Workbench

Baza de date este una importantă, deoarece cu ajutorul acesteia vom stoca toate informațiile și datele specifice aplicației realizate. Datele prezente in bazele de date sunt organizate in rânduri, coloane, tabele, fiind indexate pentru facilitarea găsirii informațiilor. SQL Workbench este o bază de date și un instrument pentru toți ahitecții de baze de date, dezvoltatori și DBA. Acest instrument ajută la configurarea serverului, administrarea utilizatorilor, backup, fiind disponibil pe toate sistemele de operare.



#### Operațiile CRUD efectuate de către SQL Server

### Bootstrap

Bootstrap reprezintă un framework de tipul open source și constituie o componentă importantă in crearea atât a site-urilor web, cât si a aplicațiilor web. Acest framework este construit pe HTML, CSS si JavaScript(JS) și are ca scop facilitarea dezvoltării de site-uri și aplicații. Design-ul responsive face posibilă ca o pagină web sau o aplicație să detecteze dimensiunea si orientarea ecranului și totodată să adapteze automat afișarea.

Principalele avantaje care le constituie acest framework ar fi:

* Utilizare ușoară și personalizată
* Folosirea mai multor plugin-uri JavaScript care folosesc JQuery
* Un framework care acceptă majoritatea browser-elor, dar și o compatibilitate CSS

### JSON- JavaScript Objectiv Notation

JSON [17] este un format care este apropiat de codul care se folosește pentru crearea obiectelor JavaScript. Fiindcă formatul acestuia este unul exclusiv text, datele pot fi transmise între pc și totodată pot si utilizate de orixce fel de limbaj de programare.

Text

Description automatically generated

##### Exemplu de interogare JSON

# Analiză și Fundamentare Teoretica

Cu ajutorul acestui capitol, vom parcurge toate tehnologiile și principiile legate de funcționare utilizate pe tot parcursul lucrării. Toate acestea se vor examina din punctul de vedere al cerințelor non-funcționale și se vor urmări modul de funcționare a fiecărei componente. Din toate acestea, vom extrage cerințele funcționale și vom realiza și o detaliere a tehnologiilor care au stat la baza creării întregii aplicații.

# Proiectare de Detaliu și Implementare

Întreaga aplicație a fost realizată din cele două părți mari: atât front-end, cât și back-end. Fiecare dintre aceste doua părți, la rândul lor au fost realizate cu ajutorul mai multor tehnologii și module. În cele ce urmează, voi descrie, pe rând părțile importante din cod, dar și unele componente.

Arhitectura specifică aplicației realizate este cea Client-Server. Prin această arhitectură înțelegem că un server oferă pentru unul sau mai mulți clienți informații. Clientul si serverul pot comunica de la distanță, prin protocolul HTTP, dar cu condiția să fie conectați la internet. După cum specifică si figura de mai jos, clientul face apeluri către serverul web, iar acesta ii returnează un răspuns prin JSON, in cazul aplicației dezvoltate.

Diagram

Description automatically generated

## Descrierea ficărei pagini prezente în aplicația web

### Pagina APLICANT

După ce fiecare aplicant se va înregistra cu propriile credențiale, va avea la dispoziție o pagină care va conține o listă cu toate posturile disponibile din firmă, impreuna cu data scadenta depunerii cererilor. După selectarea postului dorit, utilizatorul va fi direcționat catre o pagină în care va avea posibilitatea sa completeze un formular cu privire la datele personale, impreuna cu CV-ul.

În cazul in care utilizatorul va fi acceptat, va primi un mail de confirmare cu data si ora interviului, link catre interviu.

### Pagina HR Manager

Pagina principală destinată HR-ului va conține diferite opțiuni pe care le are cu redirecționare către pagina specifică opțiunii alese.

Primul tab, “Dashboard” a acestei pagini este împărțit după cum urmează:

* Componente in care ne sunt contorizate detaliile privind numărul aplicanților acceptați, respinși, dar si numărul total al candidaților.
* Grafic care ne prezintă numărul total de candidați pentru un anumit job
* Tabel cu fiecare aplicant, butoane de admis/respins care vor si trimite email candidatului, dar și o parte în care HR Manager-ul acordă o notă fiecărui candidat, în funcție de CV-ul încărcat de fiecare
* Tabel cu aplicanții și notele obținute de către aceștia

Al doilea tab este alcătuit dintr-un tabel care conține următoarele detalii:

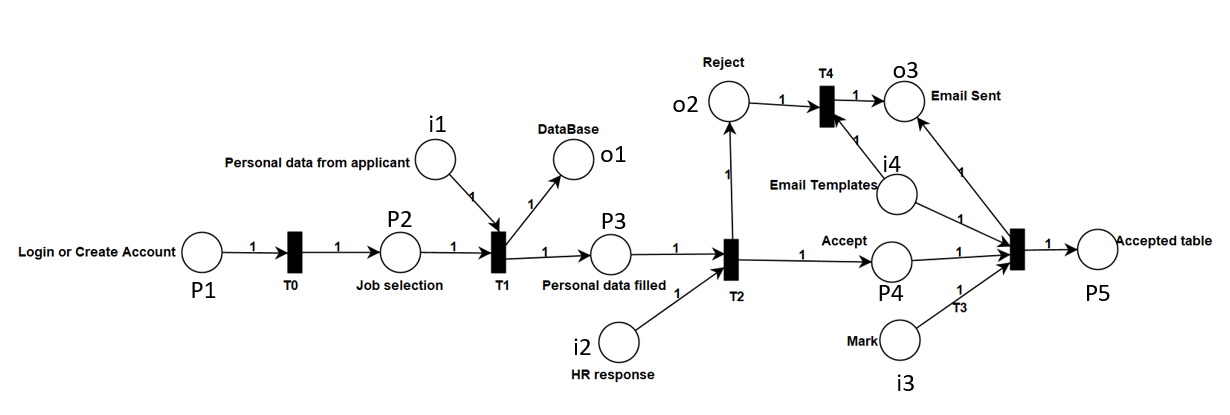
* Numele aplicantului
* Email-ul la care poate fi contactat
* Număr de telefon
* Job-ul la care a aplicat
* Calificativul de admis/respins

Al treilea tab, va contine toate posturile disponibile, împreuna cu data până la care se poate aplica. HR Manager-ul va putea face operațiile CRUD (Create, Read, Update, Delete) pentru fiecare anunț.

Al patrulea tab, va conține o retea OETPN, în care pentru fiecare user ni se va ilustra câți pași s-au realizat din tot procesul de recrutare. Aici vom vedea unde anume a ajuns jetonul, în funcție de pașii efectuați.

#### **Descriere particularități ale paginii OETPN**

Așa cum am explicat și în paragraful anterior, cu ajutorul acestei pagini, Hr Manager-ul poate monitoriza statusul aplicării fiecarui aplicant.

Rețeaua OETPN pentru statusul aplicanților

În cele ce urmează voi detalia principalele operații ale rețelelor OETPN (Location Type, Grd, Map, Init, End).

**Legenda locațiilor**

P1- utilizatorul și-a creat cont sau s-a logat un credențialele specifice

P2- utilizatorul și-a selectat job-ul dorit

I1(locație de intrare)- datele personale pentru aplicare au fost completate

O1(locație de ieșire)- formularul a fost înregistrat in baza de date

I2(locație de intrare)- răspunsul dat de catre HR Manager printr-un email specific

I3(locație de intrare)- nota acordată aplicantului

P4- accesare buton accept în urma verificării CV-ului

O2(locație de ieșire)- accesare buton respingere aplicant

O3(locație de ieșire)- trimitere email

P5- tabelul care conține datele personale ale fiecărui aplicant

* **Location types**- un set de clase software reprezentând jetoanele din rețeaua OETPN [16]

type(P1) = P1{int x1}

type(P2) = P2{int i1}

type(P3) = P3{int x3}

type(P4) = P4{int x4}

type(P5) = P5{int x5}

type(I1) = I1{int i1}

type(I2) = I2{int i2}

type(I3) = I3{int i3}

type(I4) = I4{int i4}

type(O1) = O1{int o1}

type(O2) = O2{int o2}

type(O3) = O3{int o3}

* **grd** - condiții de executie pentru tranziții bazate pe locații de intrare [16]



* **map**- condiții de executie pentru tranziții bazate pe locații de ieșire [16]



𝑔𝑟𝑑0∶=((𝑚(𝑝1)≠𝜑) →𝑚𝑎𝑝0∶=(𝑥2=x1=1)

eet() = let() = 0

𝑔𝑟𝑑1∶=((𝑚(𝑝2)≠𝜑 𝑎𝑛𝑑 (𝑚(i1)≠𝜑)) →𝑚𝑎𝑝1∶=(x3=𝑥2+i1); (o1=x2-1)

eet() = let() = 0

𝑔𝑟𝑑2∶=((𝑚(𝑝3)≠𝜑 𝑎𝑛𝑑 (𝑚(i2)≠𝜑)) →𝑚𝑎𝑝2∶=(x4=𝑥3+i2); (o2=x3-1)

eet() = let() = 0

𝑔𝑟𝑑3∶=((𝑚(𝑝4)≠𝜑 𝑎𝑛𝑑 (𝑚(i3)≠𝜑) 𝑎𝑛𝑑 (𝑚(i4)≠𝜑)) →𝑚𝑎𝑝3∶=(x5=𝑥4+i3+i4); (o3=x4-1)

eet() = let() = 0

𝑔𝑟𝑑4∶=((𝑚(𝑝2)≠𝜑 𝑎𝑛𝑑 (𝑚(i4)≠𝜑)) →𝑚𝑎𝑝4∶= (o3=o2+i4)

eet() = let() = 0

* **init**- folosită pentru a inițializa un marcaj M, dar si pentru pornirea modelului OETPN [16]
* **end**- folosită pentru terminarea/ oprirea OETPN [16]

## Cazuri de utilizare

### Actorii aplicației

Întregul sistem are în componență trei tipuri de utilizatori:

* **Admin**- este acel tip de utilizator care se ocupă de intreaga gestionare a aplicației.
* **HR Manager**- utilizatorul care administrează toate etapele recrutării pentru un job dorit.
* **Aplicant**- utilizatorul cu cele mai putine acțiuni, cel care poate vizualiza întreg panoul de posturi disponibile și cel care poate aplica la unul din job-uri.

În figura următoare am reprezentat actorul **ADMIN** cu fiecare acțiune care îi aparține:

Diagram

Description automatically generated

#### **Caz de utilizare pentru Admin**

În figura care urmează este ilustrat profilul de **Aplicant:**

Diagram

Description automatically generated

#### **Caz de utilizare pentru Aplicant**

În următoarea figura am prezentat actorul principal, **HR Manager:**

Diagram, schematic

Description automatically generated

#### **Caz de utilizare pentru HR Manager**

### Descrierea tuturor cazurilor de utilizare

În componența acestui subcapitol, se va evidenția o scurta descriere a cazurilor de utilizare a aplicației

#### **Caz de utilizare 1**

**Numele cazului de utilizare**: Logarea unui utilizator în aplicație

**Actorul**: Aplicant, HR Manager, Admin

Această parte de adreseaza utilizatorului care s-a înregistrat cu succes. Prin inregistrare, fiecare actor va putea fi direcționat către pagina asociată acestuia. Doar parcurgând acest pas, se va putea merge mai departe.

**Precondiții**: accesarea de către utilizator a paginii de login

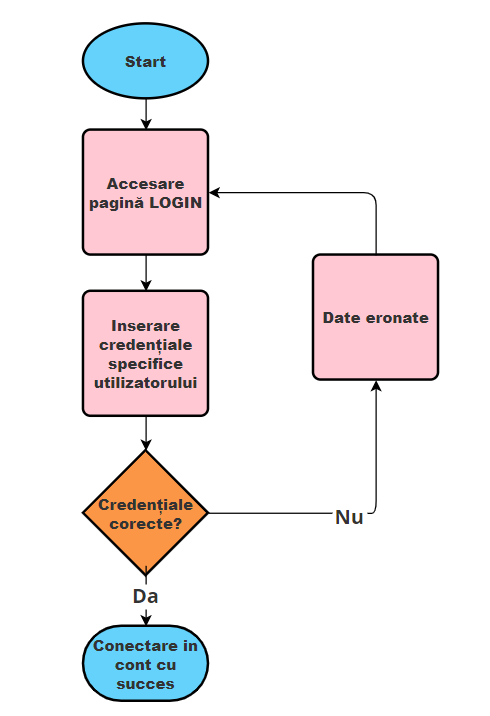
**Postconditii**: inregistrarea utlizatorului respectiv în sistem

**Scenariu favorabil**:

* Accesare pagină login
* Introducere date personale(credențiale proprii)
* Dupa intregistrare, se vor trimite datele catre baza de date
* Autentificare utilizator

**Scenariu nefavorabil**:

Datele menționate de către utilizator conțin unele greșeli și drept urmare va apărea pe ecran un mesaj specific care nu permite trecerea mai departe



##### **Diagrama flow-chart pentru logarea utilizatorilor**

#### **Caz de utilizare 2**

**Numele cazului de utilizare**: Adăugare/Ștergere/Actualizare anunțuri privind job-urile disponibile

**Actorul**: HR Manager

Această parte de adresează personalului HR Manager care va realiza operații de tipul CRUD asupra anunțurilor.

**Precondiții**:

* Logare în sistemul prezent, cu credențialele specifice
* Utilizatorul trebuie să fie în rolul de HR Manager

**Postconditii**: Adăugare/Ștergere/Actualizare/Vizualizare job-uri în sistem

**Scenariu favorabil**:

* Înregistrare utilizator
* Realizare operații CRUD pentru job-uri
* Trimitere date disponibile către sistem

**Scenariu nefavorabil**:

* HR Manager-ul nu va putea transmite datele cu privire la job-uri

Diagram

Description automatically generated

##### **Diagrama flow-chart pentru efectuarea operațiilor CRUD asupra anunțurilor privind job-urile**

#### **Caz de utilizare 3**

**Numele cazului de utilizare**: Accesare tablou cu datele privind aplicanții și realizarea unor serii de operații pentru o evidență clară a fiecărui aplicant

**Actorul**: HR Manager

Această parte de adresează tot personalului HR Manager și are scopul de a obține toate informțiile cu privire la aplicant, dar si realizarea unei statistici

**Precondiții**:

* Logare în sistemul prezent, cu credențialele specifice
* Utilizatorul trebuie să fie în rolul de HR Manager
* Accesare pagină specifică HR Manager-ului

**Postconditii**: Logare pentru rolul de HR Manager

**Scenariu favorabil**:

* Înregistrare utilizator
* Realizare statistici aplicant(acordare notă aplicant în funcție de CV-ul încărcat, realizare diagrama chart in raport cu numărul aplicanților la un anumit job, vizualizare număr total de aplicanți, respinși, acceptați, trimitere email pentru acceptare/respingere )
* Vizualizare date importante aplicanți

**Scenariu nefavorabil**:

* În cazul în care user-ul nu se logheaza cu credențialele de HR Manager, acesta nu va putea avea access la cele menționate mai sus

Diagram

Description automatically generated

##### **Diagrama flow-chart pentru pagina principală a HR Manager-ului**

#### **Caz de utilizare 4**

**Numele cazului de utilizare**: Adăugare candidatură pentru un job dorit

**Actorul**: Aplicant

Această parte de adreseaza aplicantului care va avea la dispoziție un form pentru înregistrarea pentru un post dorit.

**Precondiții**:

* Logare utilizator
* Acest utilizator trebuie să dețina rolul de aplicant
* Accesare pagină pentru alegerea și completarea form-ului de aplicare

**Postconditii**: Crearea unei candidaturi pentru un job

**Scenariu favorabil**:

* Logare utilizator cu credențialele specifice
* Alegere job dorit din lista prezentă
* Trimitere datelor din form-ul asociat, prin apăsarea unui buton care va stoca toate aceste informații în baza de date

**Scenariu nefavorabil**:

Daca utilizatorul nu completează toate câmpurile obligatorii, sistemul ii va afișa un mesaj si drept urmare nu se va putea trece la pasul următor.

Diagram

Description automatically generated

##### **Diagrama flow-chart pentru adăugarea aplicantului a candidaturii pentru un job**

## Structura aplicației(diagrama bloc cu frontend, backend+db, protocol de comunicare intre ele)

### Proiectare front-end

Diagram

Description automatically generated

#### React Architecture [20]

### Text Description automatically generated

#### Structura front-end-ului

### Proiectare pentru back-end

În acest capitol se va prezenta structura modulului backend, folosind framework-ul Spring Boot. Acest framework este divizat in pachete (Controllers, Dtos, Service, Repository, Entities), așa cum este prezentat și în figura de mai jos.

Text

Description automatically generated

#### Structura back-end

* Pachetul Controller- în antetul fiecărei clase Controller, se găsește adnotarea @RestController cu rol în a oferi acces la comportamentul aplicației realizat în pachetul Service
* Pachetul Dtos- Data Transfer Object este reprezentat de un model conceput special pentru a reduce numărul de apeluri atunci când se lucrează cu interfețe. Este util, deoarece se extrag doar atributele necesare din baza de date
* Pachetul Entities- Adnotarea @Entity este reprezentată de POJO-uri( Plain Old Java Object) care au rolul de a păstra în baza de date, datele introduse. Fiecare instanță a unei entități este reprezentată în baza de date ca un rând nou.
* Pachetul Repository- Adnotarea @Repository indică faptul că o clasă este un depozit
* Pachetul Service- adnotare care ne pune la dispoziție toată logica necesară pentru gestiunea datelor

În aplicația dezvoltată, avem două rute dedicate autentificării:

* , /user/addNewUser- rută care ne va adăuga în baza de date un cont nou
* , /user/loginUser- rută care ne va permite să accesăm contul după ce ne-am creat unul

### Structura bazei de date

Tipuri de constrângeri pentru tabelele bazei de date:

* PRIMARY KEY – identifică în mod unic fiecare înregistrare în parte
* FOREIGN KEY- realizează o legătură între coloane din diferite tabele pe baza unei chei externe
* UNIQUE- specifică coloana care o valoare unică pentru întregile înregistrări din tabel
* NOT NULL- indică faptul că o coloană nu va putea conține o valoare de tipul NULL

Diagram

Description automatically generated with medium confidence

**5.1 Diagrama bazei de date**

#### **Tabela User**

În interiorul acestei tabele se colectează toate informațiile utilizatorilor, legate de crearea unui cont nou, iar mai apoi sunt salvate în baza de date. Relațiile existente între acest tabel si tabelele Email, Form sunt relații 1:1.

Text

Description automatically generated

#### **Tabela Email**

Această tabelă este responsabilă de trimiterea email-urilor către aplicanți pentru cazurile de admis/respins. De asemenea, vom avea și o relație între aceasta și tabela User.

Graphical user interface, text, application

Description automatically generated

#### **Tabel UserJob**

#### Tabelă care are rolul de a crea legăturile necesare între id-ul user-ului din formular și id-ul job-ului din baza de date

Text

Description automatically generated

#### **Tabela Form**

Scopul acestei tabele este acela de a stoca întreaga aplicare a tuturor userilor. Vom avea legături între aceasta și tabelele: User, Job, UserJob, FileInfo.

Text

Description automatically generated

#### **Tabela FileInfo**

Folosind aceasta tabelă, utilizatorul va avea la dispozitie un câmp în care își va încărca CV-ul, iar mai apoi acesta va fi înregistrat și în baza de date. Legătura va fi între aceasta si tabela Form

Graphical user interface, text, application

Description automatically generated

#### **Tabela Job**

Cu ajutorul acestei tabele, numele job-urilor vor putea fi păstrate în baza de date. Vor exista conexiuni între aceasta și tabelele Form, UserJob

Graphical user interface, text

Description automatically generated

În realizarea acestei baze de date, s-a dorit abordarea datelor cât se poate de corect pentru a evita cazul în care informațiile ajung eronate sau se pierd pe parcurs.

## Implementare (!! Ss cu aplicația si cod relevant )

# Testare și validare

## Rezultate obținute

### Testarea funcțională a serviciilor Web

Acest pas este unul important, deoarece ni se garantează faptul că toate operațiile dezvoltate au fost corect implementate. Astfel pot fi verificate oricare dintre date prin trimiterea unui simplu request de tipul HTTP.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Descriere a testării funcționale | URL necesar | Rezultat așteptat |
| TF1 | Afișarea tuturor user-ilor | <http://localhost:8082/user/getAll> | Mesaj sub forma JSON în care ne sunt afișați toți user-ii aplicației |
| TF2 | Vizualizare status pentru fiecare aplicant | <http://localhost:8082/user/get-status> | Mesaj sub forma JSON în care ne sunt afișați pașii pe care au reușit să-i parcurgă fiecare aplicant al aplicației |
| TF3 | Vizualizare detalii aplicant (firstName, lastName) | <http://localhost:8082/form/getAllForm> | Afișarea sub forma JSON a numelui si prenumelui fiecărui aplicant |
| TF4 | Vizualizare detalii aplicant (firstName, lastName, email, telefon, job-ul la care a aplicat) | <http://localhost:8082/form/getAllForm1> | Afișarea sub forma JSON a datelor: nume, prenume, email, telefon, job-ul la care a aplicat fiecare user |
| TF5 | Vizualizare CV-uri aplicanți | <http://localhost:8082/upload/files> | Afișare sub forma JSON a unor link-uri care ne vor duce către fiecare CV în parte, în functie de numele aplicantului |
| TF6 | Vizualizare CV după numele acestuia | <http://localhost:8082/upload/files/filename> | Afișare CV sortat după numele acetuia |
| TF7 | Vizualizare job în funcție de id | http://localhost:8082/userJobs/job/{id} | Vizualizare job-uri i]n funcție de id-ul acestora |

### Testare înregistrare și logare utilizatori

Aplicația dezvoltată permite oricărui utilizator să își creeze un cont bazat pe adresă de email și parolă proprie. Acestea două trebuie să fie unele valide și complete. Dacă utilizatorul a uitat sa completeze unul dintre câmpuri, nu se va putea finaliza procesul de înregistrare.

Graphical user interface, application

Description automatically generated

#### Validare login

În cazul în care credențialele sunt corecte, contul va fi introdus automat în baza de date, iar utilizatorul va avea acces la toate anunțurile privind job-urile.

## Direcții de dezvoltare

Descrieți direcțiile posibile de dezvoltare.

# Manual de Instalare și Utilizare

Acest capitol este predestinat detaliilor ce privesc instalarea și cum anume se va folosi aplicația dezvoltată. Tot aici sunt prezenți și pașii care trebuie parcurși pentru instalarea tuturor componentelor sistemului împreună cu manualul de utilizare a aplicației.

## Instalarea tuturor resurselor necesare aplicației

Aplicația descrisă în capitolele precedente, este dezvoltată pe partea de back-end în IntelliJ IDEA, o aplicație pentru limbajul Java. Pentru partea de front-end, am folosit limbajul de programare REACT JS, iar pentru stocarea integrală a datelor am folosit SQL WORKBENCH( s-a creat automat cu ajutorul codului din IntelliJ).

* Instalare IntelliJ IDEA [18]
* Se va descărca fișierul de instalare al Java IDE ca Java IDE, de la adresa https://www.jetbrains.com/idea/download/#section=windows
* Se va parcurge întregul fișier de instalare după ce s-a ales opțiunea Run as Administrator
* Activare licență IntelliJ pe baza unui JetBrains account
* După finalizarea tuturor pașilor, se va alege opțiunea de Lunch
* Instalare Visual Studio Code (pentru ReactJS)
* Se descarcă fișierul sursă
* Se parcurg toate instrucțiunile spre instalare
* După instalare, ni se va deschide IDE-ul

Comenzile necesare pentru pornirea aplicației sunt:

* npx create-react-app my-app
* npm install
* npm start
* Instalare SQL Workbench
* Descărcare sursă de pe site-ul mysql.com/products/workbench
* După descărcare și executare, va fi necesar sa configurăm portul pe care dorim să lucrăm, un username și o parolă. Acestea ne vor fi necesare mereu atunci când vom folosi bază de date
* După toate configurările, vom putea accesa cu ușurință baza de date

## Manual de utilizare

### Înregistrare și logare în aplicație

A group of people sitting around a table

Description automatically generated with medium confidence

#### Pagina principală a aplicației

Graphical user interface

Description automatically generated with medium confidence

#### Înregistrare user nou în sistem

Graphical user interface, application

Description automatically generated

#### Pagină login

### Detalierea funcționalităților

#### Pagină pentru vizualizarea statusului aparținând aplicanților

A picture containing graphical user interface

Description automatically generated

##### Vizualizare status al aplicantului

În figura de mai sus este reprezentată pagina în care HR Manager-ul are posibilitatea de a căuta un aplicant, iar mai apoi pe rețeaua OETPN să vizualizeze câți pași a reușit să realizeze aplicantul. Tot acest progres ne va fi afișat cu ajutorul jetoanelor în locațiile specifice.

În exemplul de mai sus, se observă faptul că utilizatorul “Vlad Mihai” a parcurs pașii de login, alegere și aplicare job, dar în urma analizei de către HR Manager, acesta a fost respins.

#### Pagina principală a utilizatorului HR Manager

Graphical user interface

Description automatically generated

În cadrul acestei pagini, HR Manager-ul are o evidență clară cu numărul job-urilor, numarul aplicanților admiși/respiși. Pe baza CV-ului încarcat de fiecare aplicant, acesta poate oferi o notă in câmpul din partea dreaptă, iar in partea stânga se va realiza o ordonare în funcție de această notă. Tot aici, prin apăsarea butonului de accept sau respins, pagina ne va direcționa către un form care permite trimiterea de email catre aplicant.

Graphical user interface, application

Description automatically generated

#### Pagină trimitere email

Graphical user interface

Description automatically generated

#### Pagină anunțuri job-uri

Prin apăsarea butonului Icon

Description automatically generated din partea sus a paginii, ni se va deschide un dialog în care HR Manager-ul va putea adauga un job, împreună cu detaliile acetuia și data până la care este valabil.

Graphical user interface, text, application

Description automatically generated

Prin apăsarea butonului  din partea sus a paginii, ni se va deschide un dialog în care HR Manager-ul va putea selecta din lista disponibilă de job-uri, unul pentru eliminare.

Graphical user interface, application

Description automatically generated

Prin apăsarea butonului  din partea sus a paginii, ni se va deschide un dialog în care HR Manager-ul va putea actualiza un job dorit.

Aici vom putea modifica denumirea jobului sau detaliile care îl privesc pe acesta.

Graphical user interface, application, email

Description automatically generated

# Concluzii

# Bibliografie

|  |  |
| --- | --- |
| [1] | P. Nume, "Titlul capitolului," în *Titlul cartii*, Oras, Editura, 2016, pp. 1-24. |
| [2] | P. Nume, "Titlul articolului," *Titlul revistei,* vol. 1, no. 2, pp. 22-30, 2016. |
| [3] | P. Nume, "Titlul articolului," în *Numele conferintei*, Oras, 2015. |
| [4] | "IEEE Citation Reference," 2009. [Online]. Available: https://www.ieee.org/documents/ieeecitationref.pdf. |
| [5] | "IEEE Editorial Style Manual," 2016. [Online]. Available: https://www.ieee.org/documents/style\_manual.pdf.  [6] https://www.sumologic.com/glossary/crud/  https://ro.education-wiki.com/9050114-what-is-react  [7] https://www.tutorialspoint.com/reactjs/  [8]https://koaha.org/wiki/React\_(web\_framework)  [9] https://www.mysql.com/products/workbench/  [10] Tiberiu S. Leția, Dahlia Al- Janabi- The Development of Complex Data Structures Using Object Enhanced Time Petri Nets, IEEE International Conference on System Theory, Control and Computing 2018  [11] T.Leția- Programarea concurentă în limbajul Java standard  [12] • Leonard Richardson, RESTful Web Services, O'Reilly, 2007  [12] Leonard Richardson, RESTful Web Services, O'Reilly, 2007  [13] https://www.todaysoftmag.ro/article/81/restful-web-services-folosind-jersey  [14] B. S. Alboaie Lenuța, Servicii Web: concepte de bază și implementări, Iași: Editura POLIROM, 2006.  [15] Borcan Dumitru - Marius - Integrarea Spring cu Java Persistence API.pdf  [16] Programarea concurentă în limbajul Java standard- Tiberiu Leția  [17] https://www.w3schools.com/js/js\_json\_intro.asp  [18] http://www.cs.ubbcluj.ro/~sergiu/MAP/  [19] https://medium.com/@nityachampion/client-server-architecture-e9806f7538a9  [20] [ReactJS App Architecture. Our CEO now posts directly in his… | by Xourse | Medium](https://medium.com/@Xourse/reactjs-app-architecture-7a33d7ae9834) |