|  |  |
| --- | --- |
| **Autor:** | *Hodorog Radu* |
| **Use Case UC1:** | *Information about us* |
| **Actors:** | *Director HR* |
| **Preconditions:** | *Vizualizeaza informatii despre candidatii care sunt recomandati pentru functiile din departamentul lor.* |
| **Postconditions:** | *Poate deschide o pozitie pentru care se cauta angajati.*  *Are sarcina de a selecta candidatul acceptat pentru pozitia deschisa.* |
| **Trigger:** | *Autentificat ca Director HR* |
| **Main Scenario:** | 1. *Acceseaza pagina Candidati.* 2. *Citeste informatii despre candidatii recomandati.* 3. *Adauga o pozitie pentru candidatul acceptat.* 4. *Daca pozitia candidatului a fost adaugata eronat in sensul ca unii candidati care sunt inferiori altora au o pozitie superioara aceasta poate fi modificata.* 5. *Poate inlocui o pozitie candidatului cu alta.* |
| **Alternative Flows:** | *Are posibilitatea de a refolosi datele unei pozitii inactive, la deschiderea unei noi pozitii.* |
| **Exceptions:** |  |

Arhitectura aplicatiei

In ceea ce privesc entitatile acestea sunt legate de contextul de persistenta. Deci daca modificam valorile din obiecte modificarile sunt propagate si in baza de date. Pentru a evita acest lucru am creat clasa **CandidateDetails** care este o entitate separata ce ne ajuta la protejarea modificarilor nedorite in afara containerului EJB.

De asemenea aceasta entitate ajuta utilizatorii sa foloseasca aceeasi entitate in acelasi timp.

De aceea am creat aceasta clasa pentru a transmite date in afara EJB-ului nostru.

Tot in aceasta aplicatie am creat clasa **CandidateBean** pentru a inlocui clasa **Candidate** cu clasa **CandidateDetails**. In CandidateBean am creat metoda

*“private List<CandidateDetails> copyCandidatesToDetails(List<Candidate> candidates)*

*{*

*} “* pentru a mapa lista Candidate la lista CandidateDetails.

Pentru a salva pozitiile candidatilor in baza de date am creat 2 functii:

1. Am creat functia numita *“copyCandidateDetailsListToCandidateList”* care copiaza lista CandidateDetails la lista Candidate.
2. Am creat functia numita *“copyCandidateDetailToCandidate”* care copiaza obiectul de tip CandidateDetail la obiectul de tip Candidate.

Apoi am creat functia *“addPosition”* pentru a adauga pozitia unui candidat. In aceasta functie prima data am setat pozitia candidatului **“(candidateDetails.setPosition(position.getPosition()**) **)”** dupa care am creat un nou obiect de tip Candidate unde in acel obiect am apelat metoda *“copyCandidateDetailToCandidate”* pentru a salva obiectul candidate in baza de date in urma adaugarii pozitiei candidatului. In urma apasarii butonului pentru a adauga pozitia candidatului pe langa faptul ca acesta e salvat in baza de date este adaugata si pozitia acestuia.

Apoi am creat functia *“editPosition”* care contine aceleasi functionalitati ale functiei *“addPosition”* doar ca functia aceasta este folosita in cazul in care pozitia unui candida teste adaugata eronat.

De asemenea am creat metoda *“removePosition”* pentru a inlocui pozitia Candidat cu orice alta pozitie de tip String din clasa enum Position.

De asemenea am creat metoda *“saveAllCandidates”* pentru a salva toti candidatii in baza de date.Practic in aceasta metoda am creat am creat un obiect de tip ArrayList() numit candidateList apoi in obiectul candidateList am apelat metoda *“copyCandidateDetailsListToCandidateList(candidateDetailsList)”* .

De asemenea in metodele *“addPosition”, “editPosition”,”removePosition” si “saveAllCandidates”*  mi-am propus sa salvez obiectul candidate in baza de date cu ajutorul unui Repository in Java sau cu ajutorul DAO.(nu e gata)