DOCUMENTAȚIA PROIECTULUI

Platforma de Joburi

STUDENȚI

Trofin George Ionut

Schipor Lucian

**Cuprins**

1. Descriere platforma de joburi
2. Tehnologiile folosite pentru crearea platformei
3. Despre backend- request-uri folosite
4. Despre arhitectura platformei
5. Utilizarea aplicatiei pentru consumator
6. Concluzii

**Descriere platforma de joburi**

Acesta este un proiect de facultate creat de Lucian Schipor si Trofin George Ionut in care orice persoana dornica sa-si caute un loc de munca poate folosi aceasta platforma, ea fiind o pagina web care cauta, in functie de diverse criterii de selectie alese de utilizator, precum industria sau tipul meseriei , orice meserie disponibila la momentul curent. Acest proiect isi propune sa ajute astfel pe orice individ doritor sa-si gaseasca un loc de munca mult mai repede si cu sanse mai mari de succes in aceasta actiune. Se poate reduce astfel rata somajului si a timpului lung de cautare a unui job intr-un domeniu anume. Simplul fapt ca pe aceasta platforma oricine poate sa vada toate ofertele de munca de pe piata indiferent de locatia persoanei respective si chiar sa aplice virtual la acel loc de munca direct de pe platforma este o evolutie mare fata de trecut, acceleranduse astfel procesul de recrutare la nivel global. Cu cat se accelereaza mai mult procesul de recrutare, cu atat mai mult va creste economia fiecarei tari , rezultand astfel in salarii mai mari. De aceea am adaugat si o optiune de a selecta sariile dupa intervalul in care se incadreaza (mari,medii si mai mici).Se vor gasi si cerintele joburilor respective, fiecare stiind dinainte ce daca se poate califica pentru meseria respectiva sau nu. Utilizatorul se poate chiar loga cu un cont de aplicator daca este cazul. Nu mai mentionez ca sunt si conturi de recrutori si de admini ai platfomei care au diverse permisiuni in minus sau in plus asupra diverselor facilitati existente si chiar ascunse.

**Tehnologiile folosite pentru crearea platformei**

Pentru acest proiect s-au folosit urmatoarele tehnologii:

* C#
* ASP.NET CORE
* Visual Studio 2022
* SQL Express 2022
* Microsoft SQL Server
* EF Core

In limbajul de programare C# s-a scris codul necesar pentru partea de back-end a platfomei de joburi, fiecare procedura fiind eficient rezolvata de cu acest limbaj.

ASP.NET a ajutat mult la crearea platfomei prin rutele si comenzile HTTP ce fac posibile ,odata, gasirea unui port disponibil care sa poata fi ocupat de platforma si,

pe de alta parte, o fundatie pentru o posibila legatura intre back-end so baza de date.

Visual Studio 2022 a fost IDE-ul in care s-au creat toate acestea.

Microsoft SQL Server, folosing SQL Express 2022, a contribuit la construirea bazei de date in care sau creat,citit,modificat si sters informatii din ea.

EF Core a fost unealta care a facut legatura esentiala pentru comunicarea intre back-end si baza de date, permitand chiar facerea de migratii, care usureaza mult viata.

**DESPRE BACK-END REQUEST-URI FOLOSITE**

Pentru partea de back-end, s-au folosit request-uri de tip HTTP pentru citire,scriere,updatare si stergere a datelor din baza de date.

In mare parte s-au folosit request-urile HttpGet si HttpPost pentru tot ce am avut nevoie pentru platforma, restul de request-uri, cumva rezolvandule cu ajutorul acestor 2 request-uri. HttpGet a fost folosit pentru extragerea a diverse seturi de date, fie despre conturile aplicatorilor, fie despre cele ale recrutorilor sau chiar cele ale adminilor pentru a fi afisate persoanei cu autoritatea necesara. S-au extras si informatii despre joburi, cate sunt, ce descriere au, din ce industrie sunt, ce salarii ofera, in ce raza salariala se afla, etc, extragandule chiar dupa un criteriu de selectie anume prin console (vorbim mai tarziu despre ele). Request-urile de HttpPost au fost folosite pentru a scrie sau suprascrie date in baza de date. Fie sa adaugat un nou job in lista de joburi, fie cineva si-a creat un nou cont de aplicator sau de recrutor sau chiar de admin. Datele s-au introdus direct in baza de date dupa ce utilizatorul a apasat butonul de trimitere si acolo au ramas stocate. Modificarile se pot face momentan doar manual.

DESPRE ARHITECTURA PLATFOMEI

Platforma, la crearea sa, a fost impartita pe straturi. Odata pe DataLayer , pe Core layer si stratul principal ramand proiectul in sine.

In Core Layer se afla DTO-urile (data transfer objects) si serviciile (Services). In fisierul de DTO s-au folosit doar 4 clase, in care doar li s-a scris structura si eventual constructorul, ele urmand a fi instantiate si folosite in actiunile viitoare ale aplicatiei. Fisierul ”Services” sunt 3 clase: pentru joburi,autentificare si conturi.

In ele se afla functiile aferente pentru request-urile ce se afla implementate in controlere. Serviciile se injecteaza in proiectele( componentele) respective pentru a putea transporta metodele necesare realizarii cererii respective, fapt pentru care eu si Lucian am structurat astfel proiectul pe straturi. De asemenea, serviciul de autentificare este facut cu un generator de chei criptate (s-a ocupat Lucian de el). Eu facut mare parte requesturile si metodele.

Stratul Data Layer se imparte in fisierele “Entities” si “ Repositories”.

In entitati sunt clase cu un enumerator pentru usurarea selectarii criteriului de selectie a joburilor, in mare parte, precum cel de Role, Industry, etc. Mai contine si un fisier “User” cu clasa ” account”, care va fi folosit pentru request-urile aplicatiei si inca 2 clase separate de fisierele “Enum” si “User” pentru uz la nivel de assembly. Al doilea fisier principal este “Repository” , in care se afla un BaseRepository cu CRUD-urile necesare (facute generice) pentru a fi mostenite de celelalte 2 clase de tip de repository pentru si joburi si una pentru conturi. Ele vor fi folosite intr-o clasa “UnitOfWork” care va functiona ,precum un serviciu din Angular , la nivelul solutiei platformei, usurand astfel circulatia datelor in assembli-ul in care trebuie sa ajunga datele. Tot in Data Layer se afla si connection string-ul.

In stratul principal se afla controllerere cu request-urile pentru baza de date, care e axat pe joburi si pe conturi, avand diverse tipuri de logare si diverse cai de afisare a joburilor.Acestea sunt in fisierul “Controllers”. In fisierul “Migrations”, a facut Lucian o migratie cu EF Core, iar in “Dependencies” s-a facut un depencency injection ca sa se poate asigura circulatia datelor. In clasa Program, plina de exception handling, de care m-am ocupat, se creaza controller-ere ,asa zise “repositories” si validarea lor.

Navigarea platfomei e destul de simpla, dai click pe ce intereseaza, scrii detalii daca este cazut si dai click “Execute”

**Utilizarea aplicatiei pentru consumator**

Exista 4 tipuri de utilizatori : None (anonimi), Applier(aplicanti), Employer (angajatori) si Admin.

* Utilizatorul anonim:
  + ar trebui, in mod normal, sa vada doar obtiunile de logare si, eventual, cateva variante de selectare anume pentru joburi.
* Utilizatorul aplicator:
  + Poate sa caute toate variantele de selectare a joburilor, dar nu poate sa vada niciun angajator.
  + Acesta va avea posibilitatea sa aplice la un job, pe baza ID-ului pe care il detine job-ul. Pasii sunt urmatorii:
    - Se cauta toate job-urile disponibile
    - In functie de preferinte (salariu, tip, locatie etc) se va alege un job pe placul aplicantului
    - Aplicantul selecteaza id-ul jobului, retinut de variabila ‘JobID’
    - Aplicantul utilizeaza ruta ‘create-application’, unde va introduce date sale (experienta, studiile etc) iar aplicatia va fi inregistrata in baza de date, cu o legatura catre job-ul la care a aplicat
    - Angajatorul va putea urmari aplicatiile ulterior.
* Utilizatorul angajator:
  + Poate sa-si posteze ofertele de angajare, dar nu poate sa vada ofertele celorlalti.
  + Acesta va posta o oferta de angajare, care va contine detaliile relevante despre locul de munca (industrie, tip, descriere, locatie etc)
  + Acesta va utiliza ruta “add-job” pentru a face asta
  + De asemenea, un angajator isi va sterge optiunea de angajare atunci cand s-a gasit un angajat potrivit, folosindu-se de ruta “Delete-Job”
* Utilizatorul admin:
  + poate sa faca tot ce pot ceilalti mai din urma, dar poate sa si modifice conturile cu autoritate inferioara, sa adauage sau sa stearga joburi si chiar si alte conturi. El poate modifica aproape orice.

Autentificarea este la fel pentru toata lumea. Nu trebuie sa fii logat ca sa te poti loga la admin de exemplu. Poti si de la titlul “None” sa te loghezi la admin sau la employer, daca nu-ti place la applier.

Iar navigarea platformei este permisa intr-o masura cat iti permite autoritatea contului tau.

**Arhitectura bazei de date**

Baza de date contine trei tabele principale: Accounts, Jobs si Applications. Fiecare tabela are detalii relevante despre rolul pe care il ofera, Accounts (email, parola), Jobs (salariu, experienta), Applications (studii). Legatura dintre tabele se face prin ID-ul tabelului, preluat de la entitatea de baza ‘BaseEntity’. De exemplu, un job isi va accesa aplicatiile prin tabela ‘Applications’, unde va cauta aplicatiile care au acelasi Job\_ID ca si job-ul. Diagrama bazei de date, arata in felul urmator:

A screenshot of a computer

Description automatically generated with medium confidence

**Concluzii**

Platforma de joburi este un loc ideal pentru absolut orice individ sa isi caute un loc de munca, dar si pentru un angajator sa-si posteze ofertele de angajare pe ea, iar adminii sa le verifice si sa intretina platforma. Este foarte securizata datorita parolelor criptate si ,datorita C#-ului, va avea o longevitate continua si va satisface multi utilizatori care o for folosii in scopurile de mai sus.

ASP.NET-ul mereu va gasi un port liber pentru platforma, ea mereu fiind deschisa si valabila pentru oameni, Microsoft SQL Server cu SQL Express este o baza de date foarte securizata si ea, iar datele nu vor fi in veci furate, iar la coruptia sau pierderea lor, mereu va fi un back-up pentru orice eventualitate. Folositi platforma cu incredere.