# TaskManager (https://github.com/BayraktarD/TaskManager)

Aplicația web *TaskManager* este destinată managementului sarcinilor. Utilizatorii aplicației pot crea, modifica diverse sarcini care pot fi atribuite altor angajați sau nu.

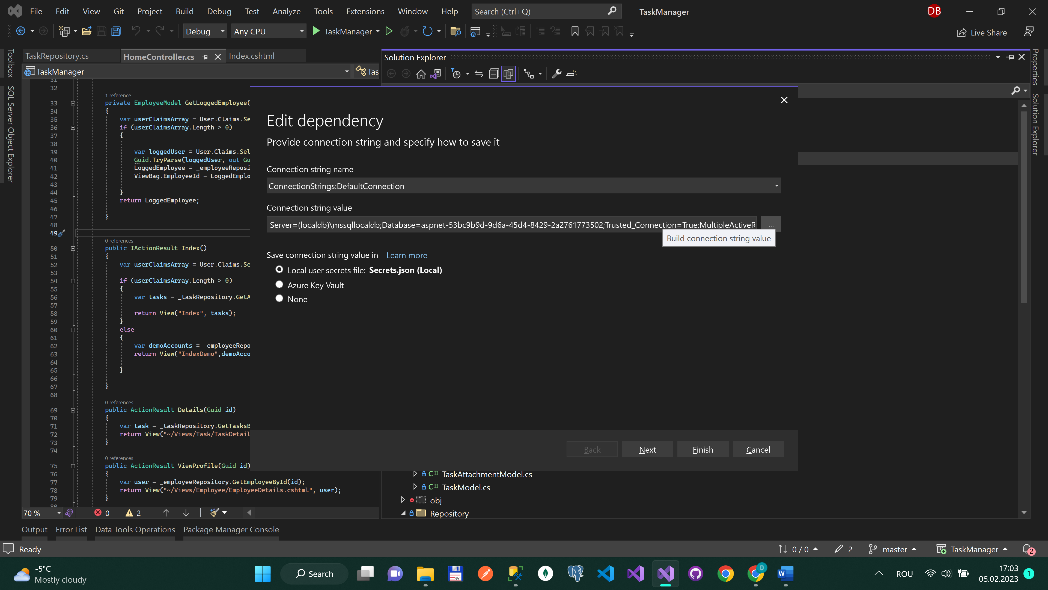
Pentru realizarea aplicației a fost realizată o bază de date ce cuprinde următoarele tabele:

* Departments;
* Employees;
* JobTitles;
* TaskAttachment;
* Tasks.

Diagram

Description automatically generatedDiagrama bazei de date este următoarea:

A screenshot of a computer

Description automatically generatedConectarea la baza de date a fost realizată prin secțiunea *Connected Services*, dând click pe *Edit*.

A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface

Description automatically generatedCrearea tabelelor a fost realizată utilizând secțiunea *SQL Server Object Explorer*. Tabelele create sunt prezentate în pozele următoare:

Graphical user interface

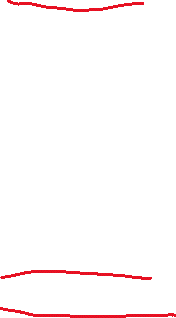
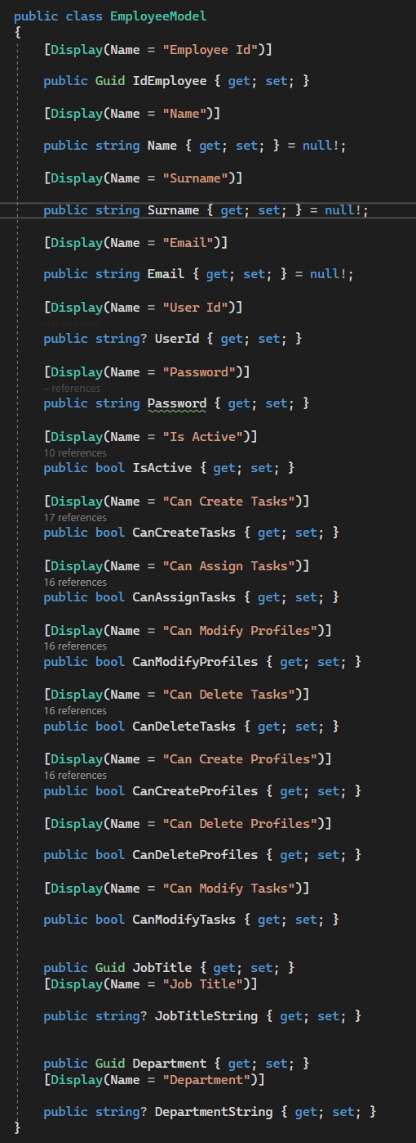
Description automatically generated

După realizarea conectarea la baza de date creată și generarea tabelelor, au fost create modelele într-un nou folder din *Models*, denumit *DBObjects*, prin rularea în *Package Manager Console* a comenzii următoare:

*Scaffold-DbContext Name=DefaultConnection Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models\DBObjects -ContextDir Data -Context ApplicationDbContext -f*

Text

Description automatically generatedDupă au fost create manual modelele utilizate în aplicație, acestea fiind asemănătoare celor create automat. Totuși există anumite diferențe necesare pentru interpretarea datelor.



În cazul prezentat mai sus, adăugarea definițiilor pentru *JobTitleString* și *DepartmentString* au fost necesare pentru afișarea acestora în tabele.

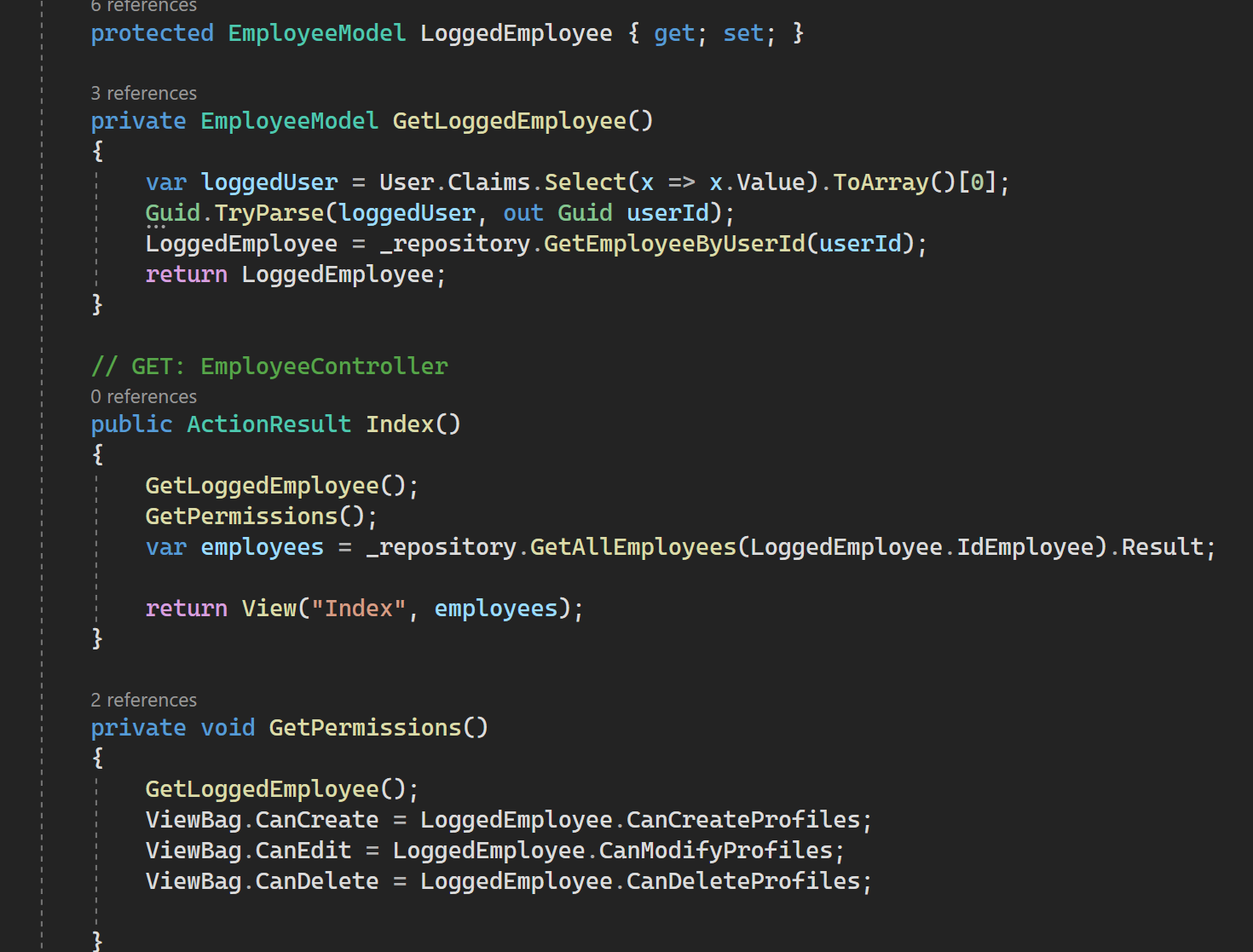
După crearea modelelor, au fost create clasele repository pentru fiecare tabel al bazei de date, acestea cuprinzând metodele necesare pentru manipularea datelor.

După crearea claselor repository, au fost create controller-ilor pentru fiecare repository, iar pentru fiecare controller – o serie de view-uri.

## EmployeeController

Pentru EmployeeController au fost create următoarele view-uri:

* *EmployeeCreate*
* *EmployeeDelete*
* *EmployeeDetails*
* *EmployeeEdit*
* *Index*

View-ul *Index* cuprinde lista angajaților, pagina fiind populată cu ajutorul metodei *Index*, care arată în felul următor:

Metoda *GetLoggedEmployee* este utilizată pentru determinarea angajatului logat, astfel determinând și permisiunile acestuia. Lista de angajați afișată în pagina *Index* nu cuprinde angajatul logat.

User-ul logat este eliminat la selectarea tuturor user-ilor, în metoda *GetAllEmpoyees.*

Text

Description automatically generatedPagina arată în felul următor:

Graphical user interface, text, application, email

Description automatically generatedÎn funcție de drepturile utilizatorului, acesta poate crea, edita, șterge datele unui angajat. Pentru fiecare dintre acțiunile respective există câte un view.

Pentru editare utilizatorul trebuie să dea click pe *Edit*, după care va fi redirecționat spre un formular ce cuprinde datele existente ale angajatului.

A screenshot of a computer

Description automatically generated with medium confidenceMetoda *SelectCategory* are ca scop selectarea valorilor pentru dropdown-urile *Department* și *Job Title.*

Graphical user interface, text, application, email

Description automatically generatedText

Description automatically generatedFormularul de editare arată în felul următor:

La fiecare modificare a datelor angajatului trebuie introdusă parola, aceasta fiind actualizată în tabela *AspNetUsers*.

Text

Description automatically generatedSalvarea datelor se face prin click pe *Save*:

Pentru salvare, este invocată metoda *UpdateEmployee* care arată în felul următor:

Text

Description automatically generated

*Text

Description automatically generated*Până la salvarea noilor date a angajtului, se face un update a datelor de logare prin metoda *UpdateUser*:

Pentru metodele care cuprind comenzi SQL pentru accesarea bazei de date a fost definit un alt connection string pentru a evita eventualele erori.

Text

Description automatically generatedÎn cadrul metodei *UpdateUser* are loc identificarea și extragerea user-ului care urmează a fi modificat, aceasta realizându-se prin metoda *GetUserFromDb*, aceasta returnând un datatable cu datele necesare.

Text

Description automatically generatedDupă extragerea user-ului are loc maparea acestuia pe model prin metoda *MapDbUserToUserModel*, care primește ca parametri acel datatable evidențiat anterior și id-ul de user și returnează un obiect de tip *IdentityUser*. Metoda arată în felul următor:

Text

Description automatically generatedFolosing obiectul de *user* și parola inserata în momentul modificării, are loc crearea adaptarea user-ului pentru introducerea acestuia in tabela *AspNetUsers* și, astfel, făcând posibil utilizatea credențialelor acestuia pentru logare. Acest fapt se realizează prin metoda *HashPassword* care primește ca parametri un obiect de tip IdentityUser și noua parolă.

Pentru validarea rezultatului metodei HashPassword este utilizată metoda *VerifyHasher* care returnează o variabilă de tip bool, astfel metoda returnează *true* dacă parola hash coincide cu datele user-ului și noua parolă.

Graphical user interface, text

Description automatically generated