***DOCUMENTATIE***

***Cerințe proiect***

1.Proiectul este individual. Structurile propuse ale bazelor de date sunt orientative,

ele pot fi modificate

2. Limbajul de programare pentru dezvoltarea aplicației este la alegere (ex. Java,

PHP, Python, Visual Studio C# etc.).

3. Sistemul de Gestiune a Bazei de Date (SGBD) va fi relațional in ORACLE și va

trebui să suporte un limbaj de programare procedural.

4. Se vor defini constrângeri de integritate pe structurile tabelare.

5. Pentru a putea fi acceptată prezentarea proiectului, aplicația trebuie să conțină o

interfață grafică ce permite introducerea de date, modificarea lor si afisarea lor

si eventualal cerinte suplimentare, precum și descrierea aplicației pe scurt

(tehnologii folosite, tabele, captura de ecran a interfetelor).

6.Documentația. Va fi structurată astfel:

(a) Descrierea bazei de date:

i. Diagrama bazei de date

ii. Structura tabelelor

(b) Descrierea aplicației:

i. Prezentarea modului în care se face conexiunea cu baza de date

ii. Diagrama de clase

(c) Capturi de ecran.

(d) Concluzii

***Proiect 6 - cerinte suplimentare***

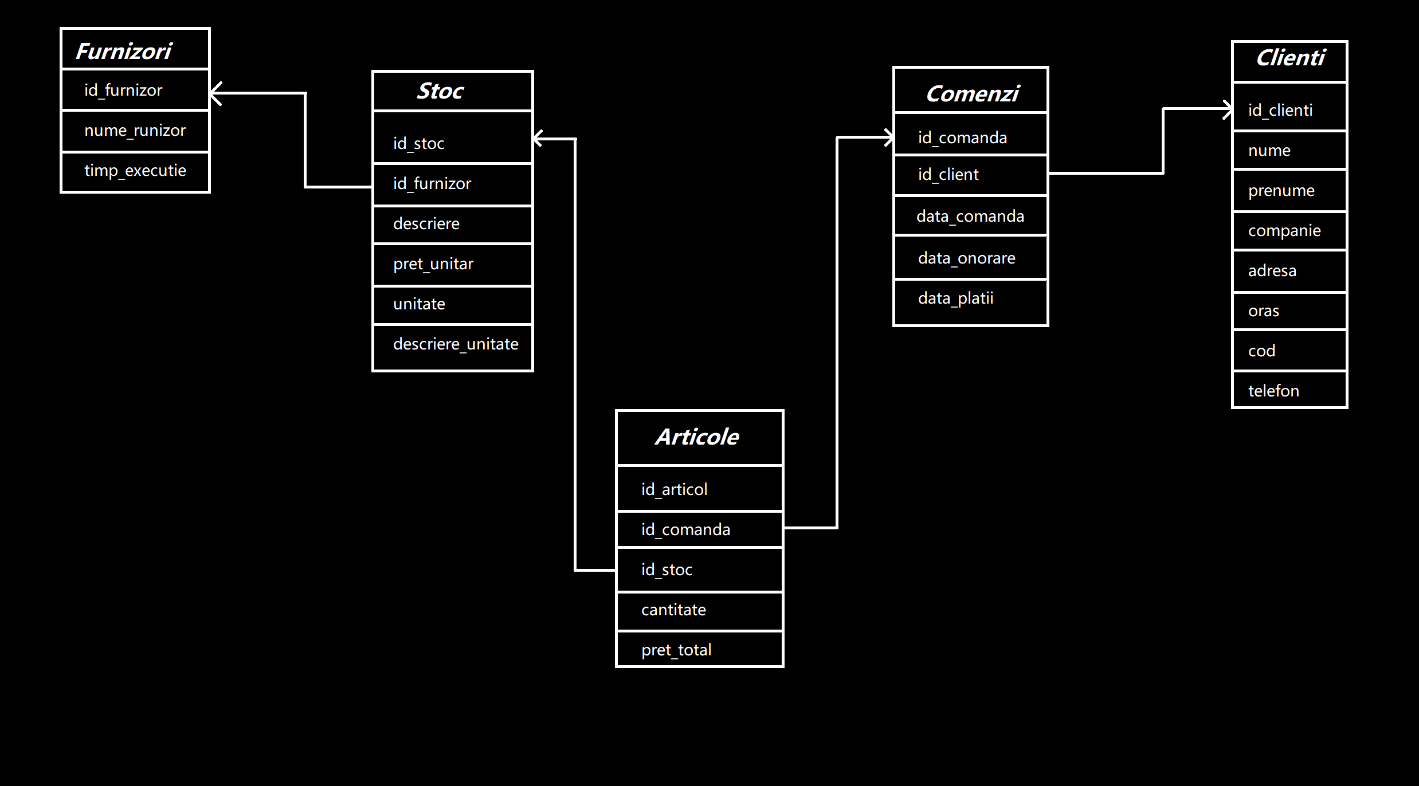
1. creati si instantiati baza de date cu toate restrictiile de integritate

3. Afisati separat Lista cu nr\_comenzi, nr\_articole si valoarea comenzii, articole (GROUP BY)

6. Determinati si afisati produsele care nu au fost niciodata comandate(nu se gasesc in articole si se gasesc in stoc), folosind un buton de pe interfata

(a) Descrierea bazei de date

i. Diagrama bazei de date



ii. Structura tabelelor

Furnizori ( id\_furnizori, nume\_furnizor, timp\_executie )

Stoc ( id\_stoc, id\_furnizor, descriere, pret\_unitar, unitate, descriere\_unitate )

Clienti ( id\_client, nume, prenume, companie, adresa, oras, cod, telefon )

Comenzi ( id\_comanda, id\_client, data\_comanda, data\_onorare, data\_platii )

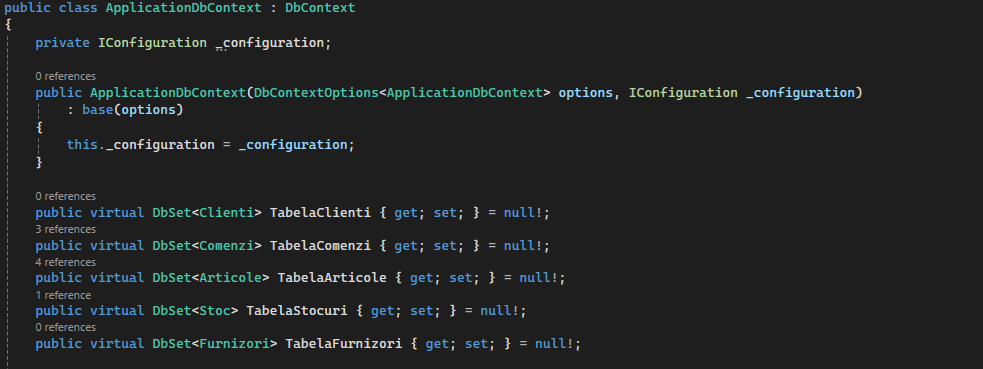
Articole ( id\_articol, id\_comanda, id\_stoc, cantitate, pret\_total )

(b) Descrierea aplicației

i. Modul în care se face conexiunea cu baza de date :

Conexiunea efectiva, deschiderea si oprirea canalului de comunicatii dintre .NET si Oracle este gestionata de EntityFrameworkCore. Packet necesar : Oracle.EntityFrameworkCore.

Acesta framework functioneaza pe baza unei clase de configurare, ce trebuie sa extinda clasa DbContext, in cazul acesta este numita ApplicationDbContext si a entitatilor.

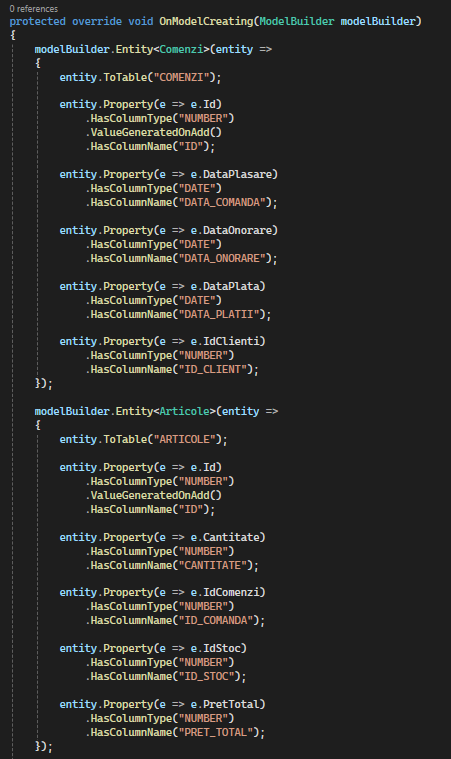


Proiectul este realizat prin abordarea ‚ database first’, simplitatea de conectare de back-end ii este specifica : Scriptul pentru crearea structurii bazei de date este creat si executat intai.

Aceasta abordare permite generarea clasei de configurare a bazei de date si a entitatilor / modelelor automat, ele fiind deja stabilite la implementarea bazei de date.

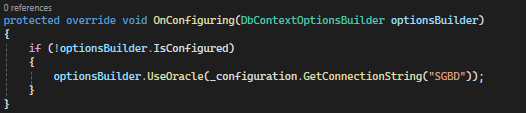
Comanda folosita pentru generare : Scaffold-DbContext "User Id=system;Password=231;Data Source=//localhost:1521/xe" -Provider Oracle.EntityFrameworkCore -OutputDir Models -Context ApplicationDbContext

Exemplu de configurari generate automat, ele folosind Fluent API :



Ceea ce vedem in poza de mai sus reprezinta informatia stiuta de EntityFramework despre clasele ce reprezinta tabelele ‚Comenzi’ si ‚Articole’ din Oracle.

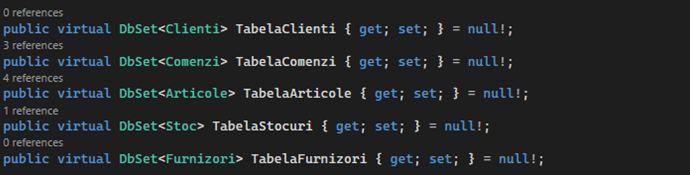
Pentru a configura conexiunea, trebuie suprascrisa metoda OnConfiguring. O configurare vitala este setarea unui connection string :



Prin metoda „UseOracle” setam un sir de caractere de conectare, in exemplul expus, acesta este localizat in appsettings.json :



Pentru a accesa tabelele, se folosesc clasele template DbSet :



ii. Diagrama de clase