Documentatie proiect PPAW

Proiectare

Paradigme utilizate

API

Web API în .NET Core este un framework pentru construirea de servicii HTTP care pot fi accesate de o varietate de clienți, cum ar fi browsere web, dispozitive mobile și dispozitive IoT. API-urile sunt construite în mod obișnuit folosind stilul arhitectural REST (Representational State Transfer) și utilizarea metodelor HTTP: GET, POST, PUT și DELETE.

Folosind .NET Core pentru a construi API-uri web, dezvoltatorii pot crea servicii scalabile, de înaltă performanță, care sunt ușor de întreținut și testat. Aceste API-uri pot fi utilizate pentru a expune datele și funcționalitățile altor aplicații, permițând integrarea fără întreruperi și interoperabilitatea între diferite sisteme software.

ORM Code First

ORM (Object-Relational Mapping) Code First este o metodă a dezvoltării bazei de date în care modelul obiect este proiectat mai întâi, iar schema bazei de date este generată din cod.

Cu Code First, se definesc clasele de model de domeniu și relațiile lor în cod, folosind C# sau alt limbaj acceptat. Aceste clase sunt apoi utilizate pentru a genera schema de bază de date corespunzătoare. Această abordare permite un mod mai natural și mai orientat pe obiecte de a gândi despre modelul de date, deoarece dezvoltatorii pot lucra cu constructe de limbaj familiar și modele de design.

Cache

Memorarea în cache este actul de stocare a datelor într-un strat intermediar, ceea ce face ca extragerile ulterioare ale datelor să fie mai rapide. Din punct de vedere conceptual, memorarea în cache este o strategie de optimizare a performanței și o considerație de proiectare. Memorarea în cache poate îmbunătăți semnificativ performanța aplicației, făcând mai ușor disponibile datele care se schimbă rar (sau sunt costisitoare de preluat).

Depencency Injection

.NET acceptă modelul de proiectare software de injecție de dependență (DI), care este o tehnică pentru realizarea inversării controlului (Inversion of Control - IoC) între clase și dependențele acestora.

Modelul de proiectare a injecției de dependență în C# ne permite să dezvoltăm componente software cuplate liber.

Arhitectura aplicație

Aplicația este imparțită în mai multe niveluri:

- Repository — aici este situat contextul.

- LibrarieModele — aici sunt situate modelurile pentru ORM.
- NivelAccessDate — aici sunt situaţi accesorii pentru accesare bazei de date.
- BusinessLayer — aici sunt definit business layer.
- ApiForum — aici este situat proiectul web api. Nivelu de prezentare.

Fiecare nivel este despărţit în proiect separat.

Implementare

Business layer – explicat

Business layer este o componentă cheie a arhitecturii generale a aplicației. Se află între stratul de acces la date și stratul de prezentare și este responsabil pentru implementarea logicii de afaceri a aplicației.

Unul dintre avantajele cheie ale stratului de afaceri este că oferă o separare a preocupărilor între nivelul de acces la date și nivelul de prezentare. Prin încapsularea logicii de afaceri într-un strat separat, se poate modifica și menține mai ușor aplicația în timp. Modificările nivelului de acces la date sau ale stratului de prezentare pot fi făcute fără a afecta logica afacerii și invers.

Librarii suplimentare utilizate

Microsoft.Extensions.Caching.Memory — a fost folosită pentru păstrarea cache-uli în memorie.

Npgsql Entity Framework Core Provider. Npgsql are un furnizor de bază Entity Framework (EF). Se comportă ca alți furnizori EF Core (de exemplu, SQL Server), astfel încât documentele generale EF Core se aplică și aici. Dacă tocmai ați început să utilizați EF Core, acele documente sunt cel mai bun loc pentru a începe.

Nlog.Web.AspNetCore — librarie pentru logare.

Secțiuni de cod sau abordări deosebite

Verificarea dacă erau folosite cuvinte nedorite:

```
private bool HasBadWords(string text)
{
    // List of discriminatory words
    string[] discriminatoryWords = { "word1", "word2", "word3" }; // Add your dis

    // Create the regular expression pattern
    string pattern = @"\b(" + string.Join("|", discriminatoryWords) + @")\b";

    // Check if the input text contains any discriminatory words
    bool hasDiscriminatoryWords = Regex.IsMatch(text, pattern);

    return hasDiscriminatoryWords;
}
```

Verificarea numărului de imagini maximale pentru un post pentru fiecare user.

```
1 reference
private bool IsAllowedNumberOfImages(int userId, int imageCount)
{
    int maxCount = 0;

    User? user = userService.GetUser(userId);
    if (user != null)
    {
        maxCount = user.AuthType.Limitation;
        return false;
    }
    return imageCount <= maxCount;
}
```

Topicul poate fi creat doar de admin.

```
public int Create(int userId, Topic topic)
{
    string userRole = userService.GetUserRole(userId);
    Console.WriteLine("Role: "+userRole);
    if (userRole != "Admin")
        return 0;

    topicsAccessor.CreateTopic(topic);
    return 1;
}
```

Pașii de instalare

Pentur programator

`cd forum`

`cd ApiForum`

`dotnet run` — cu această comandă automatic se efectuează instalarea dependeților.

`dotnet publish` — compilarea aplicației.

`dotnet ApiForum.dll` — aplicație startează pe kestrel server.

`sudo apt install postgresql postgresql-contrib` — installarea bazei de date

`psql -U postgres`

`create database forum;` - se crează baza de date.

Pentru beneficiar

`dotnet ApiForum.dll` — aplicație startează pe kestrel server.

`sudo apt install postgresql postgresql-contrib` — installarea bazei de date

`psql -U postgres`

`create database forum;` - se crează baza de date.