Raport de Dezvoltare - Proiect ASP.NET Core MVC

Echipa:

- **Ionel Otilia (IO)**
- **Andra Chirițoiu (AC)**

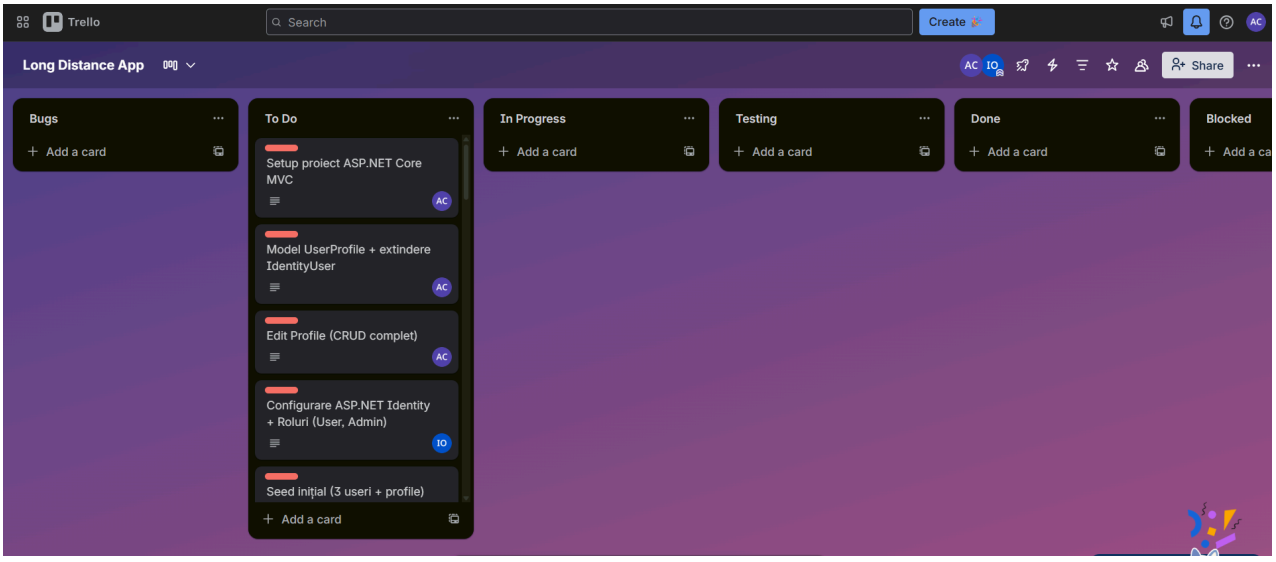
Grupa: 234

Proiect: MicroSocialPlatform

1. Metodologia de lucru și Planificarea (Trello)

Pentru organizarea eficientă a proiectului, am adoptat o metodologie de lucru **Agile**, utilizând platforma **Trello** pentru managementul sarcinilor (task-urilor). Am împărțit dezvoltarea aplicației în **3 Sprint-uri** distincte, marcate prin etichete colorate (Labels) pe board-ul nostru, pentru a avea o vizibilitate clară asupra progresului.

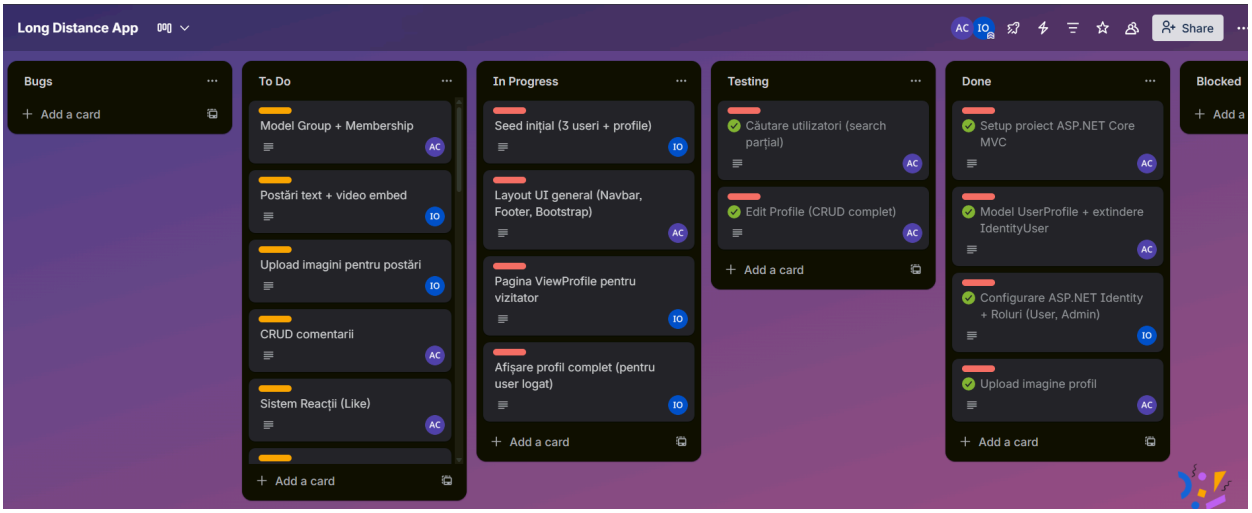
Fiecare membru al echipei a avut responsabilități clare, marcate pe cartonașe cu inițialele noastre (**IO** - Ionel Otilia și **AC** - Andra Chirițoiu), asigurând astfel o împărțire echilibrată a muncii (Back-End și Front-End).



2. Detalierea Sprint-urilor

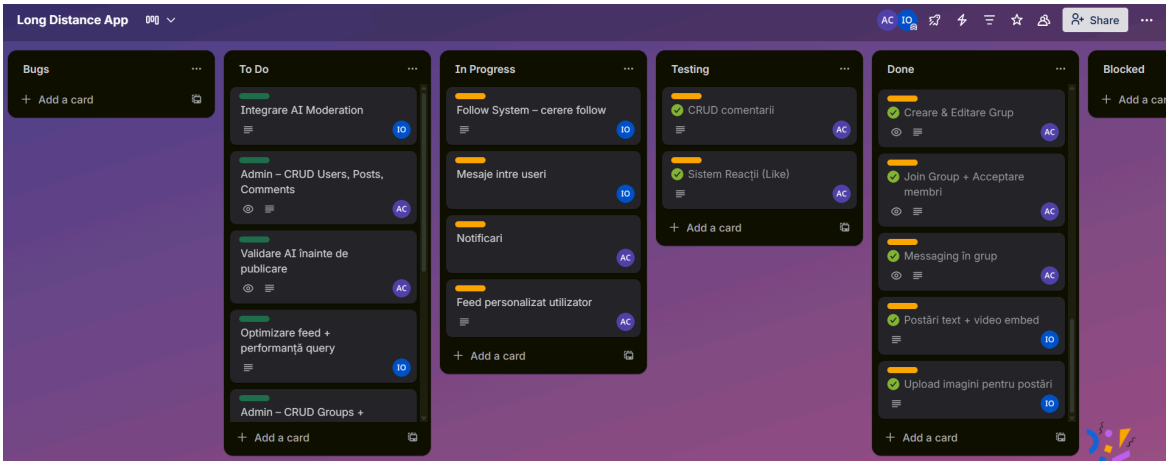
Sprint 1 (Eticheta Roz) – Arhitectura și Configurația de Bază În prima etapă, ne-am concentrat pe "fundament". Prioritatea a fost configurarea corectă a proiectului și a sistemului de identitate.

- **Realizări:** Am configurat **ASP.NET Core Identity**, am extins clasa **IdentityUser** prin modelul UserProfile pentru a adăuga date specifice (descriere, poză) și am implementat mecanismul de **Seed Data** pentru popularea inițială a bazei de date cu roluri (Admin, User) și utilizatori de test.
- **UI:** Am stabilit Layout-ul general folosind **Bootstrap** (Navbar, Footer).



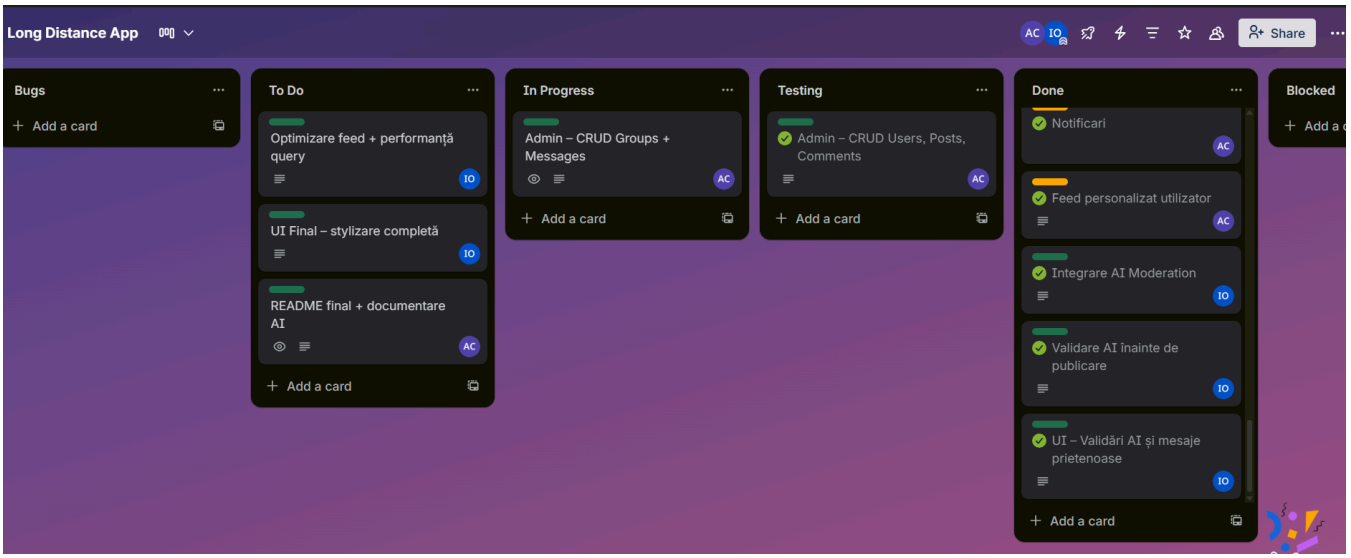
Sprint 2 (Eticheta Galbenă) – Funcționalitățile Sociale (Core Features) Acesta a fost cel mai complex sprint, dedicat interacțiunii dintre utilizatori.

- **Realizări:** Am implementat sistemul de Grupuri (Creare, Editare, Join), sistemul de Postări (inclusiv upload de imagini și embed video), Comentarii și Reacții (Like).
- **Interacțiune:** Am adăugat funcționalitatea de "Follow" și mesagerie privată între utilizatori sau în grupuri.



Sprint 3 (Eticheta Verde) – Administrare, Optimizare și AI Ultima etapă este dedicată rafinării și funcțiilor avansate.

- **Realizări:** Ne-am concentram pe zona de Administrare (CRUD complet pentru Admin asupra conținutului), validări avansate și, ca element de inovație, integrarea unor funcționalități AI pentru moderarea conținutului și validarea mesajelor înainte de publicare.



3. Impedimente Întâlnite și Soluții Tehnice

Pe parcursul dezvoltării, am întâmpinat diverse probleme tehnice care ne-au ajutat să înțelegem mai bine funcționarea Entity Framework Core:

1.Problema datelor obligatorii (NULL Constraints):

- *Situație*: La rularea **SeedData** și la înregistrarea utilizatorilor noi, primeam eroarea “Cannot insert the value NULL into column 'Description' “.
- *Cauză*: Coloanele noi adăugate în **ApplicationUser** erau **NOT NULL** în baza de date, dar codul nu le popula.

- *Soluție*: Am actualizat clasa **SeedData.cs** și **Register.cshtml.cs** pentru a atribui valori implicite (ex: o imagine "default.png" și o descriere generică) la crearea oricărui cont nou.

2.Duplicarea Rolurilor la Seed:

- *Situație*: Primeam eroarea “There is already an object named 'AspNetRoles' in the database” sau erori de cheie primară duplicată.
- *Soluție*: Am adăugat o **verificare logică if (context.Roles.Any()) return;** la începutul metodei de inițializare, pentru a preveni rularea codului de Seed dacă baza de date este deja populată.

3. Problema Încărcării Datelor Relaționale (Eager Loading):

- *Situație*: În pagina de **Feed** (afișarea postărilor), primeam frecvent **NullReferenceException** atunci când încercam să afișăm numele utilizatorului care a creat postarea (**item.User.FirstName**).
- *Cauză:* Am înțeles că Entity Framework Core nu încarcă automat datele din tabelele relaționate (comportamentul default nu este **Lazy Loading** automat pentru toate proprietățile).
- *Soluție*: Am optimizat interogările **LINQ** folosind metoda **.Include(p => p.User)** și **.Include(p => p.Comments)**, trecând astfel la **Eager Loading**. Aceasta a redus numărul de interogări către baza de date și a rezolvat erorile de nulitate.

4. Eroarea "Multiple Cascade Paths" în SQL Server :

- *Situație*: La implementarea sistemului de **Grupuri** și **Comentarii**, am întâmpinat eroarea critică: “ **Introducing FOREIGN KEY constraint may cause cycles or multiple cascade paths** “.

- *Cauză*: SQL Server nu permite ștergerea în cascadă pe mai multe căi. De exemplu, ștergerea unui Grup șterge postările, dar și ștergerea unui User șterge postările. Când un User este și în Grup, se creează un conflict de ștergere (ciclu).
- *Soluție*: A trebuit să intervenim în fișierul **ApplicationDbContext.cs** și să suprascriem metoda **OnModelCreating**. Am configurat manual relațiile folosind **Fluent API,** schimbând comportamentul de ștergere din **Cascade** în **Restrict** pentru una dintre relații **(.OnDelete(DeleteBehavior.Restrict))**, rupând astfel ciclul infinit.

4. Concluzii

Proiectul ne-a oferit ocazia să lucrăm cu un flux complet de dezvoltare Web: de la proiectarea bazei de date relaționale, la logica de backend în C# și interfața cu utilizatorul. Utilizarea Trello a fost esențială pentru a nu ne suprapune pe fișiere și pentru a ști mereu în ce stadiu se află colegul de echipă.