ARHITECTURA SOLUȚIEI

Platforma Națională de Voluntariat „voluntar.md”

Prezentul document descrie arhitectura tehnică a Platformei Naționale de Voluntariat voluntar.md, în conformitate cu Secțiunile 5 și 6 ale Caietului de Sarcini. Documentul cuprinde diagrama de context, arhitectura pe componente, structura Clean Architecture, modelul de date (ERD), fluxurile de stare și arhitectura de implementare (deployment).

Digitally signed by Morozan Victor
Date: 2026.06.09 11:30:32 EEST
Reason: MoldSign Signature
Location: Moldova
MOLDOVA EUROPEANĂ

Cuprins

1. Privire de ansamblu

Soluția voluntar.md este o aplicație web modernă pe trei niveluri, construită integral pe stack-ul tehnologic impus de Autoritatea Contractantă:

- **Client (Frontend):** Angular 19+ cu Standalone Components și Server-Side Rendering (SSR), stilizat cu Tailwind CSS 4+, reactiv prin Angular Signals;
- **API (Backend):** ASP.NET Core (.NET 10 LTS) organizat conform Clean Architecture, cu Entity Framework Core 9+;
- **Bază de date:** PostgreSQL 16+, cu connection pooling pentru scalare.

Comunicarea se realizează prin API REST versionat (/api/v1), securizat cu JWT Bearer. Sistemul este containerizat (Docker) și livrat prin pipeline CI/CD către trei medii (Development, Staging, Production).

1.1 Principii arhitecturale

- Separarea responsabilităților prin Clean Architecture (dependențe unidirecționale spre Domain);
- API stateless (autentificare prin JWT) care permite scalarea orizontală (minim 2 instanțe);
- Validare centralizată (FluentValidation + pipeline MediatR) și audit imutabil al operațiunilor critice;
- Securitate în profunzime (hashing bcrypt, HTTPS, security headers, rate limiting, CORS restrictiv).

2. Diagrama de context

Diagrama de context prezintă platforma ca un sistem unic și interacțiunile cu actorii externi (cele patru roluri de utilizator și publicul larg) și cu infrastructura de găzduire a autorității contractante.

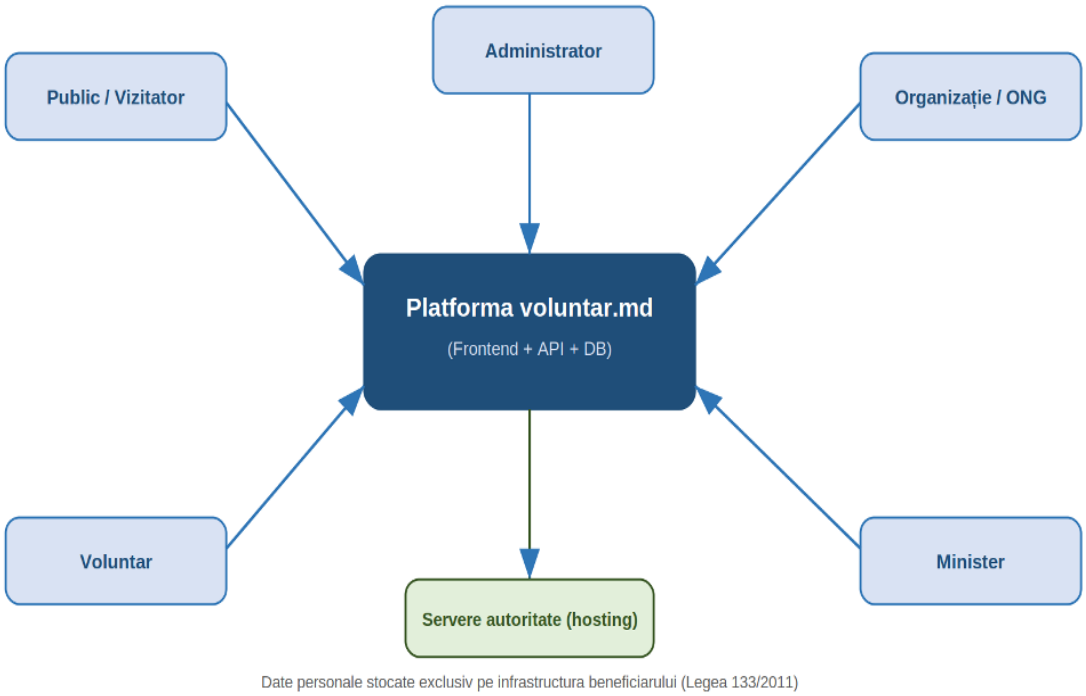

Figura 1. Diagrama de context — actori și sistem

** Toate datele cu caracter personal sunt stocate exclusiv pe serverele Autorității Contractante, în conformitate cu Legea nr. 133/2011.*

3. Arhitectura pe componente

Arhitectura pe niveluri ilustrează separarea între client, API și baza de date, precum și componentele majore din fiecare nivel.

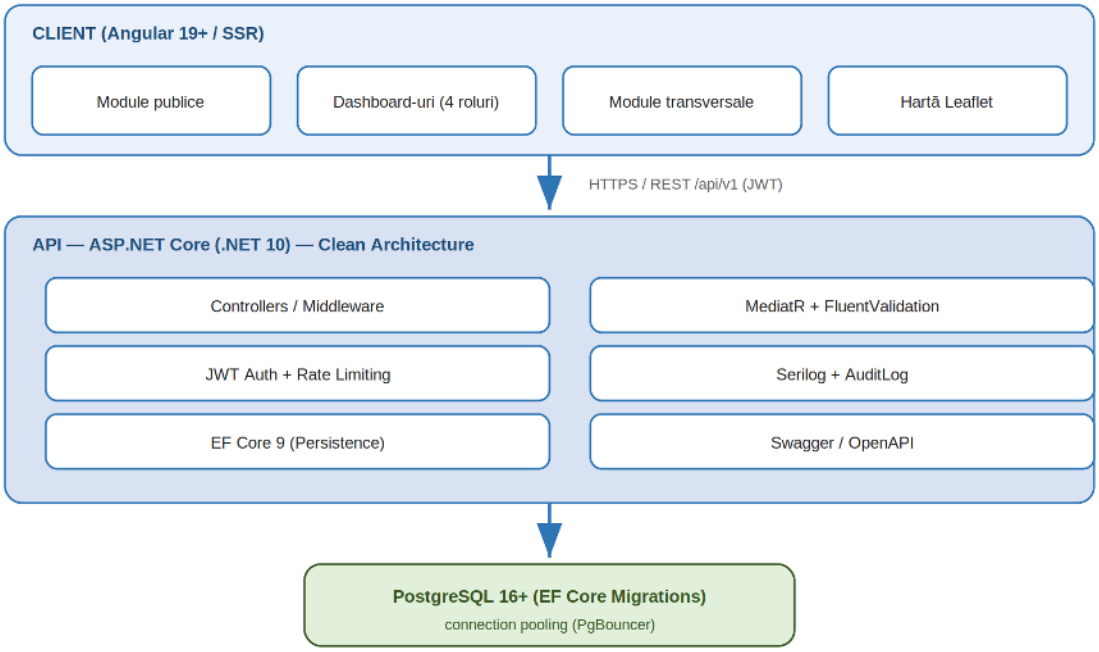

Figura 2. Arhitectura pe componente (client — API — bază de date)

3.1 Nivelul Client (Angular)

Aplicația Angular este structurată în trei zone: core (guards role-based, interceptors JWT, modele, servicii), features (16 module încărcate lazy prin loadComponent) și shared (componente reutilizabile: Nav, StatusBadge, EmptyState, LoadingSpinner, OpportunityCard). SSR (@angular/ssr) asigură SEO și performanță, iar hărțile Leaflet se randează exclusiv în browser (afterNextRender).

3.2 Nivelul API (ASP.NET Core)

API-ul expune operațiunile prin controllere documentate Swagger/OpenAPI, cu un pipeline MediatR pentru comenzi și interogări, validare FluentValidation, autentificare JWT, rate limiting și logging Serilog. Toate operațiunile CRUD critice sunt înregistrate în tabelul AuditLog.

3.3 Nivelul de date (PostgreSQL)

Persistența este gestionată prin EF Core 9+ cu migrații aplicate automat la pornire. Indexurile pe câmpurile frecvent filtrate (status, role, categoryId) și connection pooling-ul (PgBouncer) asigură performanța și scalarea.

4. Clean Architecture

Backend-ul respectă principiile Clean Architecture, cu dependențe orientate exclusiv spre interior (către Domain). Acest model izolează regulile de business de detaliile de infrastructură, facilitând testarea și mentenanța.

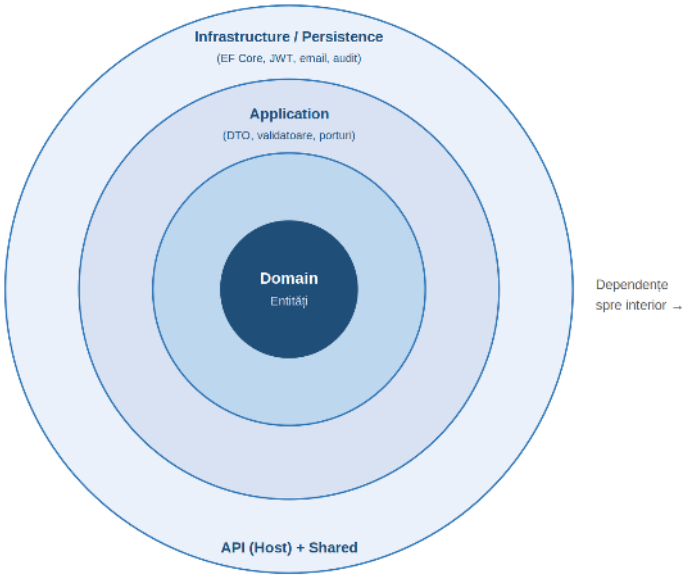

Figura 3. Straturile Clean Architecture (dependențe spre interior)

4.1 Structura pe proiecte (.NET)

Soluția este organizată în 6 proiecte, conform structurii din Caietul de Sarcini:

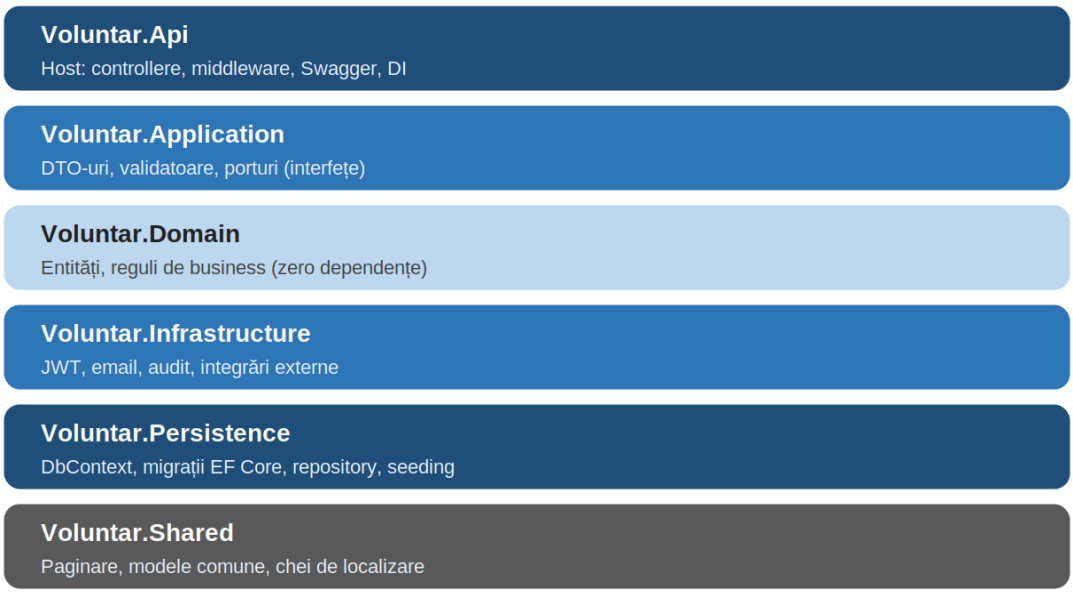

Figura 4. Cele 6 proiecte ale soluției backend

5. Modelul de date (ERD)

Diagrama entitate-relație prezintă entitățile principale ale domeniului și legăturile dintre ele, conform secțiunii 6.2 a Caietului de Sarcini.

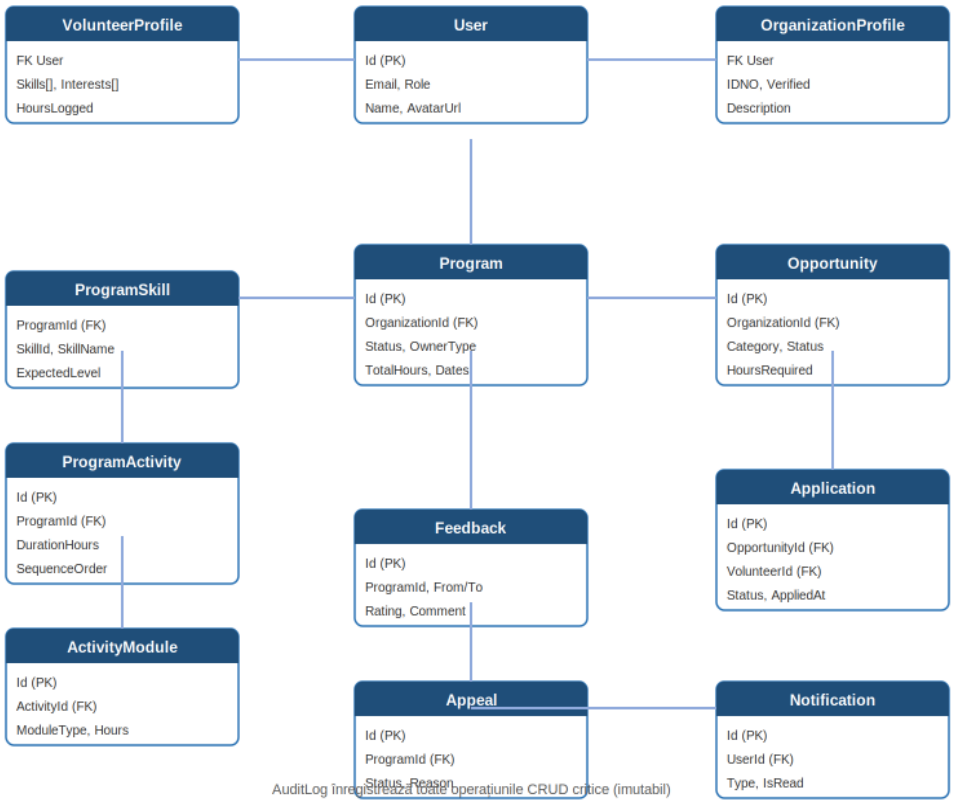

Figura 5. Modelul de date — entitățile domeniului și relațiile

5.1 Entități principale

Modelul cuprinde 13 entități: User (cu rolurile Volunteer/Organization/Ministry/Admin), VolunteerProfile, OrganizationProfile, Program (cu fluxul de stare), ProgramSkill, ProgramActivity, ActivityModule, Opportunity, Application, Notification, Feedback, Appeal și AuditLog. Tabelul AuditLog înregistrează automat operațiunile CRUD critice și este imutabil.

6. Fluxuri de stare

Entitățile cu ciclu de viață (Program, Application, Appeal) sunt implementate ca mașini de stare validate la nivel de domeniu, conform secțiunii 6.3 a Caietului de Sarcini.

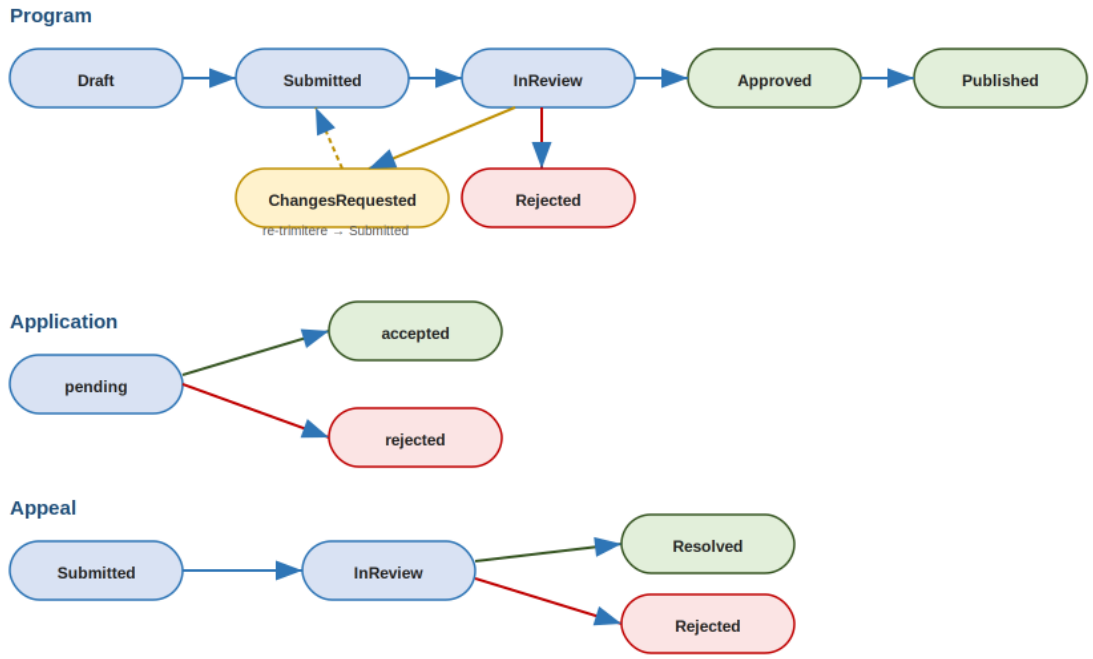

Figura 6. Fluxurile de stare pentru Program, Application și Appeal

Fluxul Program permite re-trimiterea după solicitarea de modificări (ChangesRequested → Submitted) și publicarea după aprobare (Approved → Published). Tranzițiile nepermise sunt blocate la nivel de domeniu și înregistrate în audit.

7. Arhitectura de implementare (Deployment)

Soluția este livrată în trei medii, cu promovarea codului prin pipeline CI/CD (GitHub Actions / GitLab CI) și Coolify. Migrațiile EF Core sunt aplicate automat la pornirea aplicației.

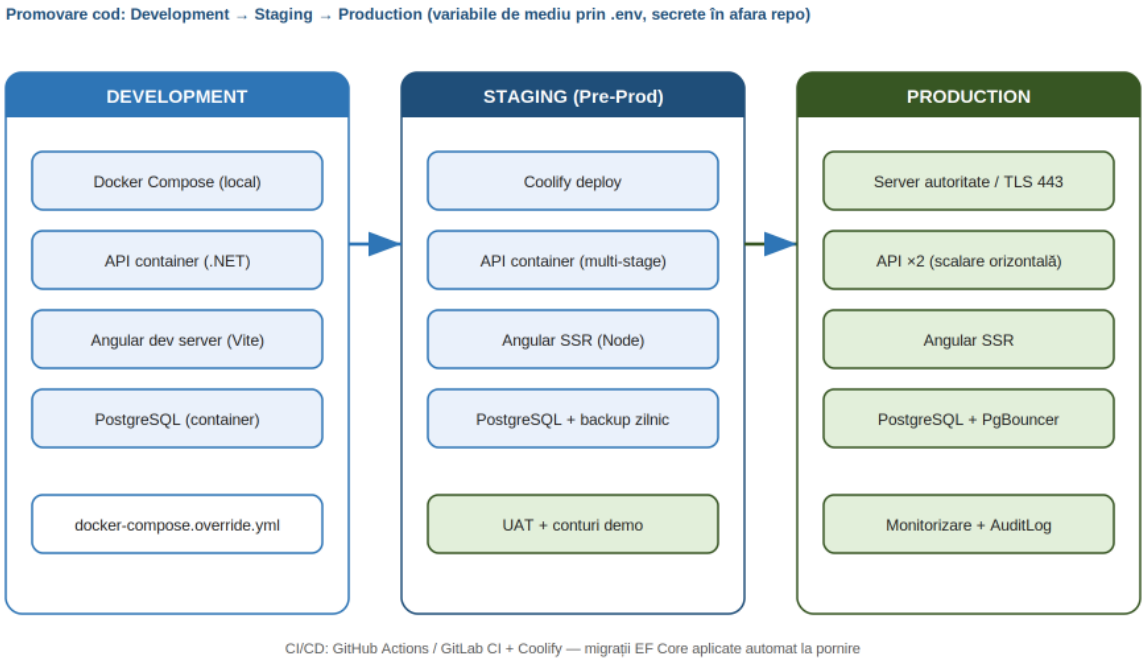

Figura 7. Arhitectura de implementare — medii și flux CI/CD

7.1 Containerizare și configurare

- Dockerfile multi-stage (build + runtime separate); imaginea finală conține doar runtime-ul .NET;
- docker-compose.yml de bază, docker-compose.override.yml (dev) și docker-compose.prod.yml (producție);
- Secrete configurate exclusiv prin variabile de mediu (.env exclus din repo; .env.example documentat);
- În producție: HTTPS obligatoriu (TLS 443, Let's Encrypt acceptat), API scalat orizontal (minim 2 instanțe).

7.2 Backup și recuperare

Strategia de backup PostgreSQL este automatizată (zilnic), asigurând RPO ≤ 24 ore și RTO ≤ 4 ore. Procedura de restaurare este documentată și testată înainte de recepția finală.

8. Acoperirea cerințelor non-funcționale

Arhitectura este proiectată să îndeplinească țintele non-funcționale din Caietul de Sarcini:

Cerință	Soluție arhitecturală
Performanță API (p95 ≤ 300/500ms)	Paginare server-side, indexare DB, query-uri optimizate EF Core, caching unde e relevant.
FCP ≤ 2s, Lighthouse ≥ 80/90	SSR Angular, lazy loading, bundle optimizat prin Vite.
Uptime ≥ 99,5%	API scalat orizontal, monitorizare, ferestre de mentenanță planificate.
RTO ≤ 4h / RPO ≤ 24h	Backup zilnic automatizat, procedură de restaurare documentată.
Scalabilitate	API stateless (JWT), connection pooling (PgBouncer), containere replicabile.
Securitate	bcrypt, JWT cu rotation, HTTPS, security headers, rate limiting, CORS, audit imutabil.

Detaliile complete de implementare a cerințelor sunt prezentate în Oferta Tehnică (capitolele 5–8) și în Matricea de Conformitate.