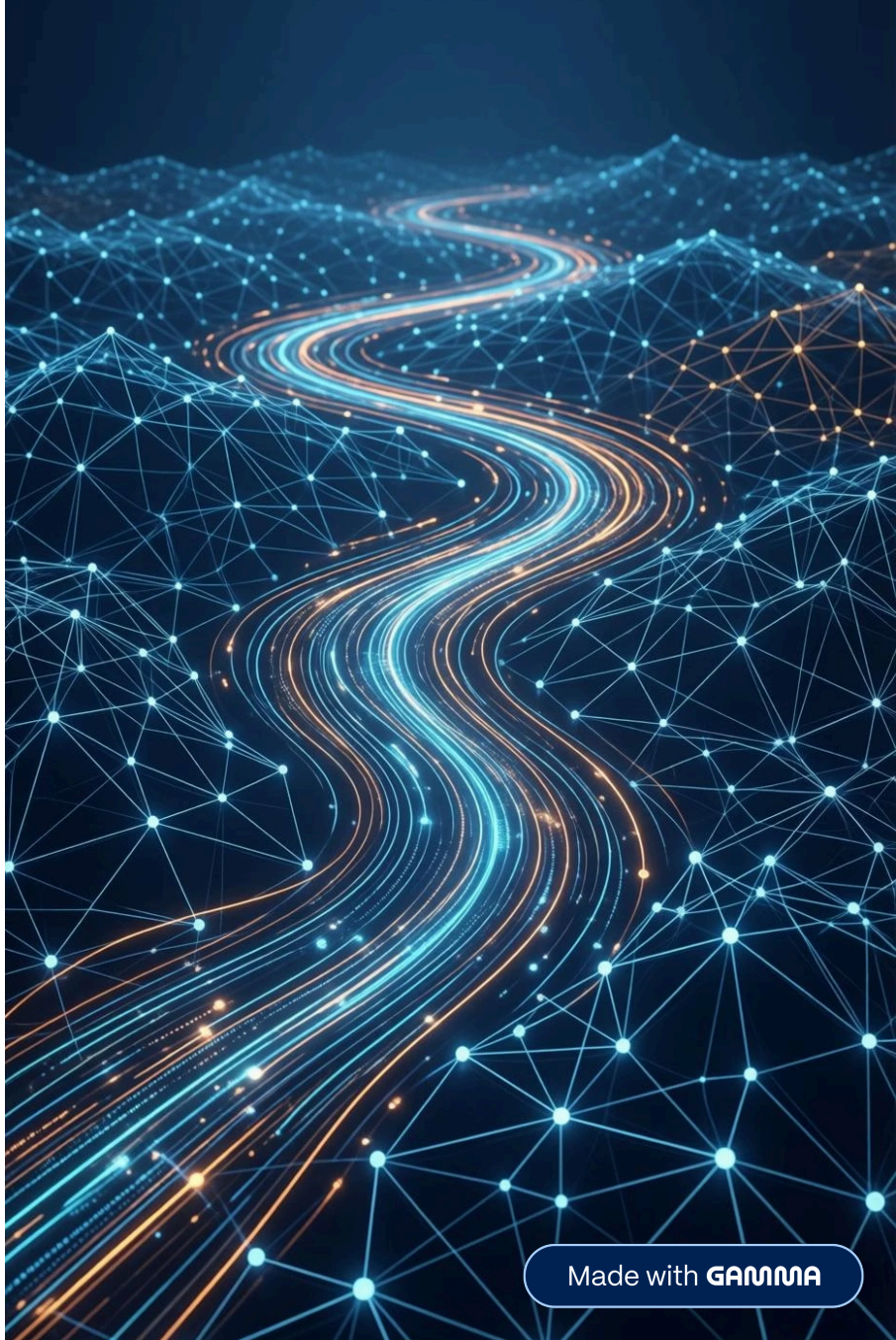
Arhitectura unui Sistem de Mesagerie P2P



O privire de ansamblu asupra arhitecturii

Frontend: Interfața Utilizator

Frontend-ul, construit cu React, Vite și TypeScript, oferă o interfață rapidă și reactivă, esențială pentru o aplicație de mesagerie.

React

Componentizare eficientă (ex: Composer, MessageList).

Vite

Timp de build redus și feedback instant prin Hot Reload.

TypeScript

Tipare declarative pentru un cod robust și ușor de întreținut.

Comunicații P2P cu WebRTC Data Channels

WebRTC Data Channels este coloana vertebrală a comunicațiilor P2P, gestionând conexiunile directe dintre clienți pentru un chat decentralizat.

- **Latență Redusă:** Mesajele circulă direct între utilizatori, eliminând intermedierea serverului.
- **Trafic Distribuit:** Serverul este utilizat doar pentru semnalizare (descoperirea și inițierea conexiunilor P2P).
- **Securitate Îmbunătățită:** Conținutul mesajelor nu tranzitează prin server.

Backend: Serverul de Semnalizare și API

Backend-ul, implementat cu Node.js și Express, servește ca un gateway de semnalizare, facilitând stabilirea conexiunilor P2P fără a intermedia conținutul.

- **Node.js + Express:** Un server ușor și eficient pentru API-uri și Evenimente Trimise de Server (SSE).
- **WebSocket:** Utilizat pentru gateway-ul de semnalizare, permițând comunicarea în timp real necesară pentru stabilirea conexiunilor WebRTC.
- **TypeScript:** Asigură consistența tipurilor între frontend și backend, reducând erorile și îmbunătățind mentenabilitatea.

Persistență Locală cu SQLite

Stocarea locală a datelor este gestionată prin SQLite și biblioteca `better-sqlite3`, asigurând o soluție simplă și rapidă pentru persistența mesajelor și a stării aplicației.

SQLite

Bază de date ușoară, ideală pentru stocare locală.

Better-SQLite3

Performanță și API intuitiv pentru interacțiunea cu SQLite.

Repository Pattern

(MessageRepository) izolează accesul la date pentru o arhitectură curată.

Servicii Auxiliare și Implementare

Pentru a asigura funcționalitatea completă și scalabilitatea, au fost dezvoltate servicii auxiliare esențiale și strategii de implementare robuste.

1

PeerRegistry

Urmărește și gestionează starea peer-ilor activi în rețea.

2

MessageService

Orchestrează validarea și logica de afaceri pentru mesaje.

3

Containerizare

Întreaga aplicație este împachetată pentru Docker și Kubernetes.

4

Lansare Ușoară

Asigură o lansare fluidă în diverse medii de producție.

Docker, Kubernetes și Distribuire

Pentru a asigura o implementare scalabilă și rezilientă a sistemului de mesagerie P2P, integrarea Docker și Kubernetes este esențială. Aceste tehnologii permit o gestionare eficientă a resurselor și o operare fiabilă a aplicației în medii distribuite, asigurând disponibilitatea și performanța necesare pentru o experiență fluidă a utilizatorului.

Docker: Containerizare

Împachetează aplicația și dependințele sale într-un container izolat, asigurând portabilitate și consistență în diferite medii. Facilitează distribuția rapidă a componentelor sistemului P2P, inclusiv a backend-ului și a bazelor de date.

Kubernetes: Orchestrere

Gestionează automat implementarea, scalarea și operarea containerelor Docker. Asigură disponibilitate ridicată prin repornirea automată a componentelor eșuate și balansează încărcarea traficului între instanțe.

Scalabilitate Orizontală

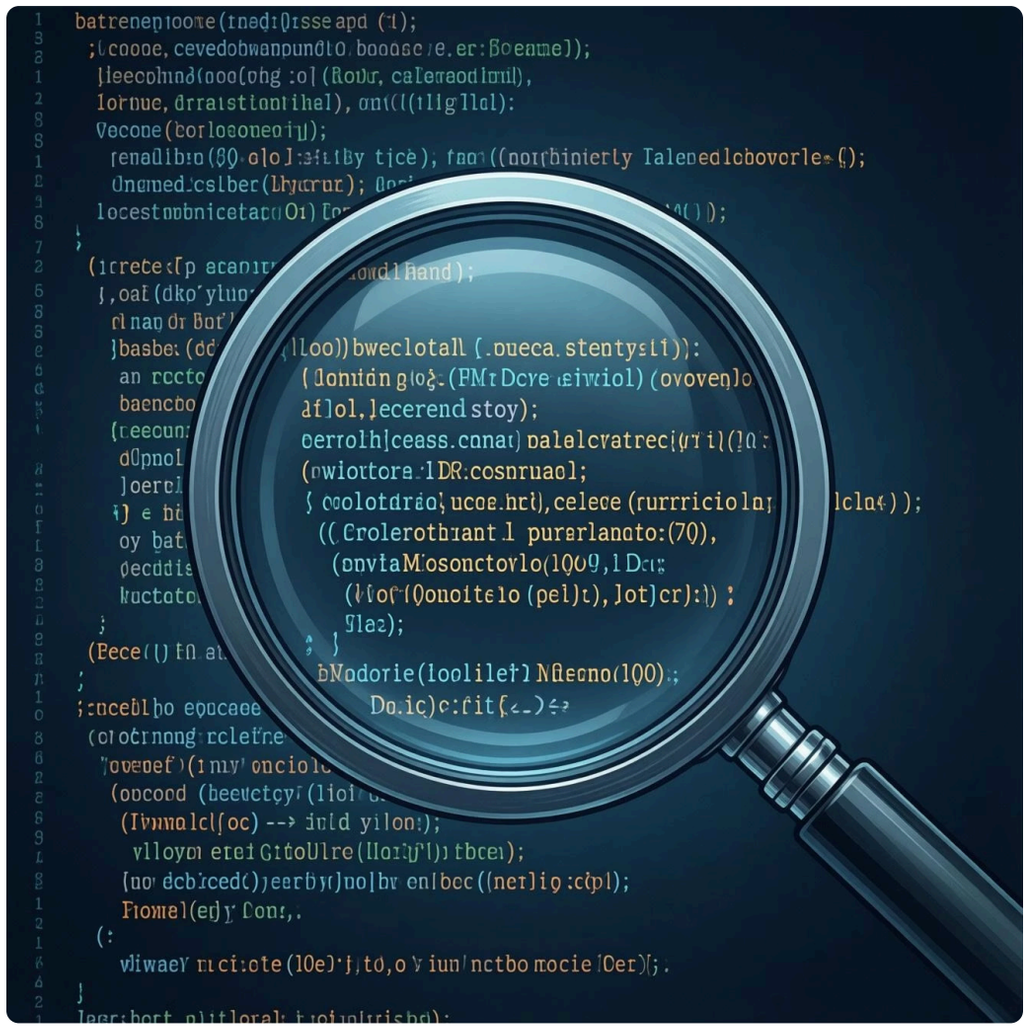
Permite adăugarea dinamică de noi instanțe ale serviciilor (e.g., servere de semnalizare, baze de date) pentru a gestiona un volum crescut de utilizatori sau trafic, fără a afecta performanța sistemului P2P.

Toleranță la Erori

Design-ul sistemului distribuit, susținut de Kubernetes, asigură funcționalitatea continuă chiar și în cazul defectării unor componente individuale. Sistemul se recuperează automat, menținând serviciul disponibil.

Testare și Calitate Cod

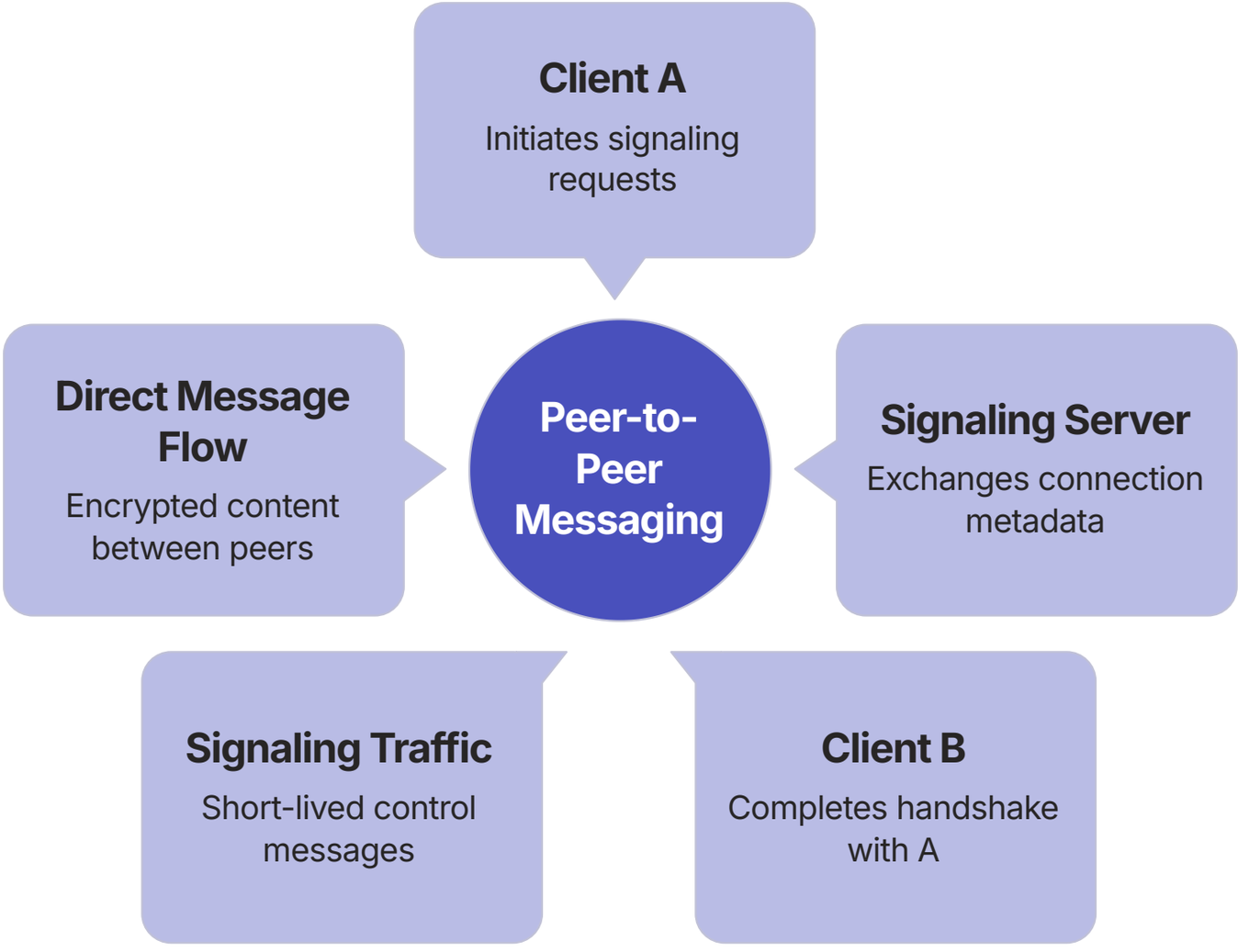
Un accent deosebit a fost pus pe testare și menținerea calității codului, asigurând stabilitatea și fiabilitatea aplicației.



- **Jest (Backend):** Testează logica serverului și serviciile backend.
- **Vitest + React Testing Library (Frontend):** Asigură funcționalitatea componentelor UI și a fluxurilor utilizatorilor.
- **ESLint/Prettier:** Mențin stilul codului consecvent și uniform, prevenind erorile.

Motivația din Spatele Aplicației

Acest proiect demonstrează fezabilitatea unui sistem de mesagerie distribuit, unde serverul are un rol minim, axându-se pe descoperirea peer-ilor.



Serverul nu intermediază conținutul mesajelor, ci doar ajută la descoperirea și inițierea conexiunilor directe între utilizatori.

Concluzii și Perspective

Am explorat o arhitectură de mesagerie P2P care maximizează decentralizarea și reduce dependența de serverele centrale pentru conținut.

1

Autonomie Crescută

Utilizatorii dețin controlul asupra datelor lor.

2

Performanță Îmbunătățită

Latență redusă și scalabilitate prin distribuția traficului.

3

Model Scalabil

Poate fi extins pentru o varietate de aplicații descentralizate.