SISTEME DISTRIBUITE

Sistem de Licitații Live

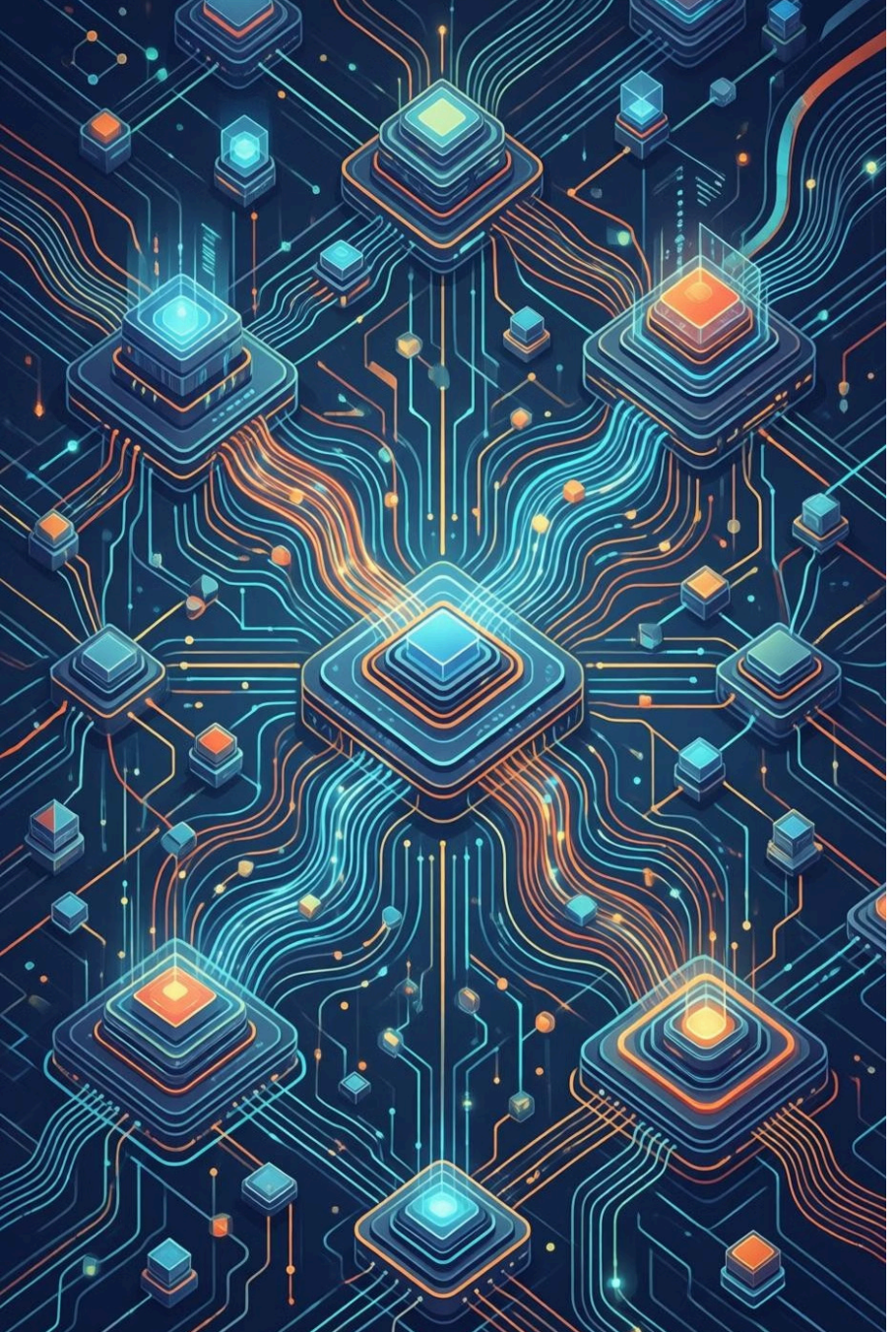
Platformă distribuită de licitații în timp real bazată pe microservicii și Kubernetes

Arhitectura Distribuită

Sistemul implementează o **arhitectură de microservicii** cu separarea responsabilităților între componente independente:

- Auction Service - API REST pentru licitații
- Notification Service - WebSocket în timp real
- Worker de expirare - procesare asincronă
- Frontend React - interfață utilizator

Fiecare serviciu rulează ca **proces independent**, comunicând prin protocoale bine definite și permițând scalare specifică nevoilor.



Comunicare Asincronă și Cuplaj Redus

Publish/Subscribe

Auction Service publică evenimente în Redis fără a cunoaște consumatorii

Canal Redis

Canalul **auction.events.v1** transportă evenimente între servicii

Broadcast WebSocket

Notification Service traduce evenimente în mesaje Socket.io către clienți

Fluxul Distribuit de Date

01

Ingress Kubernetes

Utilizatorii accesează prin Ingress care rutează traficul către servicii

02

Validare și Persistență

Auction Service validează licitații, salvează în MongoDB și programează expirarea

03

Publicare Evenimente

Evenimentele sunt publicate în Redis cu envelope semnate (eventId, traceId, occurredAt)

04

Notificări Real-Time

Notification Service consumă evenimente și emite mesaje Socket.io către browsere

Consistență și Coordonare Distribuită

Control Optimist al Concurenței

Actualizarea licitațiilor folosește **versiunea documentului Mongoose** pentru a preveni suprascrierea ofertelor concurente și garantarea monotoniei prețului

Reconciliere la Pornire

Worker-ul verifică licitațiile expirate la restart pentru a evita **stări divergente** între baza de date și coada de joburi

Coada BullMQ Întârziată

Programarea expirării elimină dependența de cronuri centralizate și permite **redimensionarea worker-ilor** în funcție de volum

Scalare Orizontală

Socket.io cu Adaptor Redis

Replicile multiple ale Notification Service partajează aceeași **magazie de sesiuni**, astfel orice replică poate trimite notificări către orice client conectat.

Worker-i Paraleli

BullMQ permite rularea **mai multor worker-i** pentru a procesa simultan expirări în scenarii cu volum ridicat.

Servicii Stateless

API-ul și serviciul de notificări sunt **stateless**, facilitând înlocuirea rapidă a podurilor fără pierdere de date.

2+

Replici Notification

Scalare automată

∞

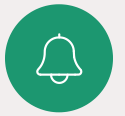
Worker-i BullMQ

Procesare paralelă

Separarea Responsabilităților

Auction Service

Expune API REST pentru autentificare, gestiunea licitațiilor și plasarea ofertelor. Rulează pe portul 3000.



Notification Service

Creează conexiunea WebSocket către clienți și traduce evenimente Redis în notificări live. Portul 4000.

Worker Expirare

Procesează joburi BullMQ pentru expirarea automată a licitațiilor și reconcilierea stărilor.

Infrastructura Distribuită

Kubernetes

Orchestrare și networking pentru toate serviciile

Ingress

Rutare trafic cu cookie affinity pentru WebSocket

MongoDB

Stare persistentă pentru utilizatori și licitații

Redis

Pub/Sub pentru evenimente și backend BullMQ

Docker

Containerizare pentru portabilitate și izolare

Rezultate

Realizări Cheie

- Arhitectură **complet distribuită** și scalabilă
- Fluxuri de evenimente robuste cu tratament erori
- Sincronizare corectă în scenarii concurente
- Experiență utilizator reactivă în timp real