Propunere tehnică

Sistemul Informațional “Registrul Prețurilor Bunurilor Imobile”
RFP NR. SI RPBI (ocds-b3wdp1-MD-1773831984484)

S&T Mold
Str. Calea Ieșilor 8
MD-2069, Moldova
t: +373 22 837960
e: office@snt.md; tenders@snt.md
w: www.snt.md

Digitally signed by Cojocari Andrei
Date: 2026.04.22 09:57:16 EEST
Reason: MoldSign Signature
Location: Moldova
MOLDOVA EUROPEANĂ

Cuprins

1. Rezumat executiv

S&T Mold își exprimă prin prezenta interesul față de proiect și prezintă abordarea și expertiza tehnică ce vor fi utilizate pentru executarea și finalizarea cu succes a proiectului.

S&T Mold activează în Republica Moldova din anul 1995 și dispune de un portofoliu amplu de soluții livrate atât pentru sectorul public, cât și pentru cel privat, inclusiv sisteme enterprise critice pentru misiune.

Compania deține competențele și abilitățile necesare pentru finalizarea cu succes a acestui proiect și este pregătită să le valorifice în maniera și stilul descrise în continuare în prezentul document.

2. Abordare tehnică

Abordarea noastră în proiectarea și dezvoltarea sistemelor informaționale constă în echilibrarea tehnologiilor fundamentale și validate cu metode moderne și de ultimă generație de construire și scalare a software-ului. Valorificăm, de asemenea, experiența noastră vastă de două decenii în dezvoltarea și livrarea de soluții software pentru organizații mari, lecțiile învățate, competențele dobândite, expertiza tehnică și experiența considerabilă de colaborare cu echipe ale donatorilor internaționali și cu reprezentanți ai autorităților publice, care ne-au permis să construim un istoric solid de proiecte livrate cu succes.

În capitolele următoare vom prezenta principiile și tehnologiile pe care intenționăm să le utilizăm pentru implementarea proiectului curent.

2.1. Arhitectură și principii de proiectare

2.1.1. Arhitectură orientată pe servicii

Sistemul va fi proiectat și construit ca o arhitectură orientată pe servicii, utilizând microservicii.

2.1.2. SOLID

Sistemul va fi proiectat pe baza principiilor SOLID, ceea ce îl va face ușor de întreținut și extins.

2.1.3. Domain Driven Design

Vom utiliza Domain-Driven Design pentru a colabora continuu cu experții din domeniu, în vederea îmbunătățirii modelului aplicației și a abstractizării corecte a entităților și proceselor.

2.1.4. Microservicii

Arhitectura bazată pe microservicii extinde principiul responsabilității unice asupra unor servicii slab cuplate, care pot fi dezvoltate, implementate și întreținute independent. Microserviciile pot fi scalate independent, spre deosebire de aplicațiile monolitice mari, care se scalează împreună. Aceasta înseamnă că o anumită zonă

funcțională, care necesită mai multă putere de procesare sau mai multă lățime de bandă pentru a susține cererea, poate fi scalată fără a extinde inutil și alte zone ale aplicației. De asemenea, microserviciile oferă o izolare îmbunătățită a defectelor, astfel încât, în cazul unei erori într-un serviciu, întreaga aplicație nu își încetează neapărat funcționarea.

2.1.5. Containere (Docker)

Containerizarea microserviciilor va permite livrarea acestora împreună cu toate fișierele de configurare asociate, bibliotecile și dependențele necesare pentru rularea eficientă și corectă a unui microserviciu în diferite medii de calcul.

Avantajul unei abordări orientate pe containere pentru dezvoltare și implementare este că elimină majoritatea problemelor generate de configurările neuniforme ale mediilor și de efectele asociate acestora. În plus, containerele permit scalarea rapidă a aplicației prin instanțierea de noi containere, după necesitate.

2.1.6. Orchestrarea containerelor și auto-scalarea prin Kubernetes

Vom configura clustere Kubernetes pentru găzduirea și auto-scalarea sistemului. Kubernetes este un sistem open-source pentru automatizarea implementării, scalării și administrării aplicațiilor containerizate. Acesta oferă un cadru pentru rularea rezilientă a sistemelor distribuite. Asigură scalarea automată pe baza consumului de memorie și/sau CPU, auto-vindecare prin repornirea automată a containerelor care cedează sau nu răspund la verificările de sănătate definite de utilizator, nu expune containerele către clienți până când acestea nu sunt gata să deservească cereri, echilibrare automată a traficului și descoperirea serviciilor, precum și multe altele.

2.1.7. Procesarea asincronă a fluxurilor de date

Microserviciile care vor colabora pentru procesarea datelor vor schimba mesaje prin fluxuri persistente de date asincrone, care vor decupla consumatorii și producătorii de date, vor permite proiectarea și implementarea facilă a unor fluxuri complexe de procesare și vor permite sistemului să se scaleze natural prin realocarea resurselor de la fluxurile inactive către cele încărcate.

2.1.8. Limbaje de programare și tehnologii fundamentale

Vom construi sistemul folosind limbaje de programare consacrate și de vârf în industrie pentru aplicații enterprise:

- Java pentru serviciile back-end și logica de bază a aplicației
- JavaScript pentru interfețele utilizator front-end
- Python pentru procesare avansată a datelor, scripturi de automatizare și integrare cu instrumente analitice

Toate framework-urile și bibliotecile utilizate vor fi open-source, larg adoptate și întreținute activ. Vom evita tehnologiile de nișă sau cu nivel redus de adopție, pentru a asigura mentenabilitate pe termen lung, suport din partea comunității și conformitate cu bunele practici din industrie.

2.1.9. Aplicație de tip Single Page

O aplicație de tip single-page funcționează în browser și nu necesită reîncărcarea paginii, nici timp suplimentar de așteptare. Majoritatea resurselor (HTML/CSS/scripturi) sunt încărcate o singură dată pe durata de viață a aplicației. Deoarece aplicațiile single-page nu actualizează întreaga pagină, ci doar conținutul necesar, acestea îmbunătățesc semnificativ viteza aplicației. Se transmit doar datele, în ambele sensuri. Aplicațiile web single-page sunt ideale pentru construirea de platforme dinamice și aplicații enterprise.

2.1.10. Jurnal complet de audit

Jurnalele de audit înregistrează practic fiecare acțiune a utilizatorului în cadrul unui sistem, oferind o evidență completă a operațiunilor din aplicație. Prin urmare, jurnalele de audit reprezintă o resursă valoroasă pentru administratori și auditori, care doresc să diagnosticheze și să depaneze probleme sau să analizeze activitatea utilizatorilor.

Jurnalele de audit contribuie și la securitate, deoarece oferă înregistrări ale întregii activități a utilizatorilor, inclusiv activitate suspectă. Acestea pot sprijini monitorizarea datelor și sistemelor pentru identificarea eventualelor breșe sau vulnerabilități de securitate și pot ajuta la depistarea utilizării abuzive interne a datelor. Ele pot furniza înregistrări care pot servi drept dovezi în cazul unor litigii.

2.1.11. Securizat prin proiectare

Sistemul va implementa cele mai noi standarde de securitate și va urma recomandările OWASP la toate nivelurile. De asemenea, va utiliza cele mai noi tehnologii și va fi menținut la zi cu toate patch-urile de securitate nou emise pentru bibliotecile de bază utilizate. Prin utilizarea instrumentelor automate de tip Security Compliance Tracker în cadrul procesului de integrare continuă, echipa de dezvoltare poate monitoriza securitatea aplicației în raport cu vulnerabilitățile cunoscute și poate remedia problemele înainte de lansarea aplicației.

2.1.12. Interconectivitate cu alte sisteme

Sistemul va putea comunica ușor cu alte sisteme, atât în calitate de consumator, cât și de furnizor de servicii web, utilizând protocoalele standard: REST și SOAP.

De asemenea, echipa noastră are o experiență vastă în integrarea cu MPass, MSign, MLog, MNotify, MPay, MConnect, MConnect-Events și a realizat integrări cu majoritatea serviciilor MCloud.

2.2. Design de nivel înalt

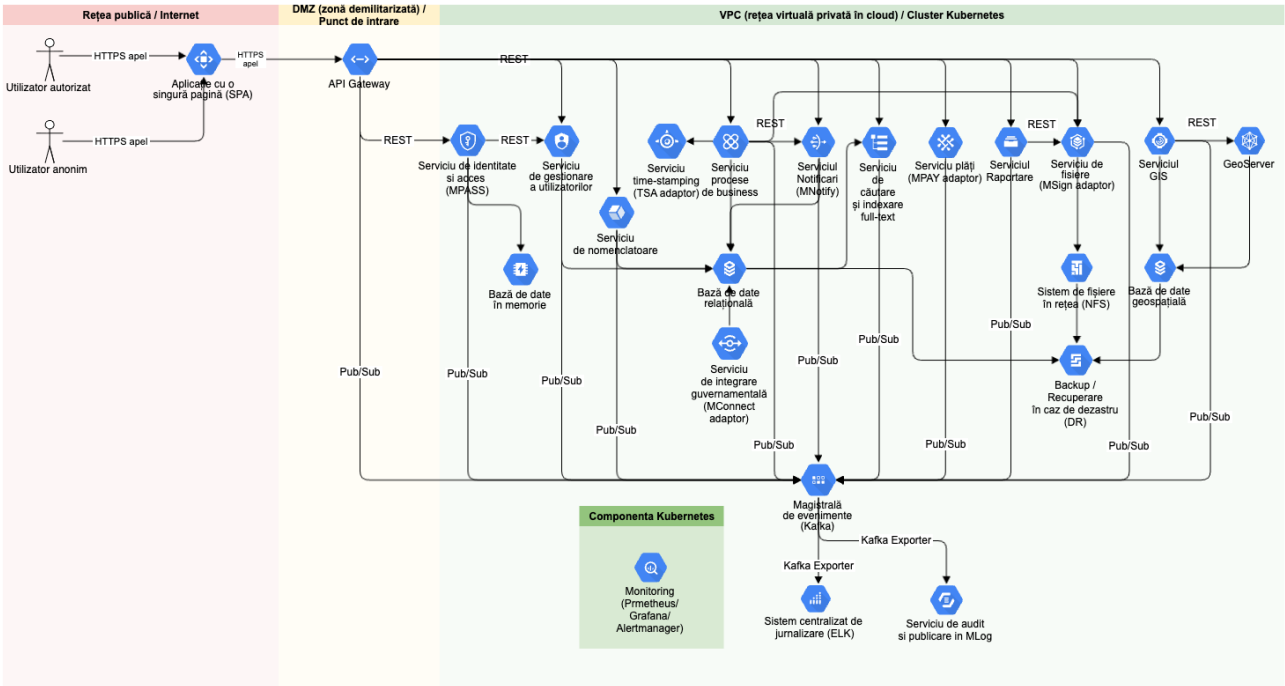


Diagrama de mai sus prezintă o imagine de ansamblu, la nivel înalt, a arhitecturii soluției propuse și modul în care containerele comunică între ele. Fiecare componentă reprezintă o unitate containerizată separată, executabilă/implementabilă independent, cum ar fi o aplicație single-page, un microserviciu, o bază de date etc.

Prin utilizarea serviciilor containerizate, software-ul poate fi implementat în cloud, într-un cluster Kubernetes sau pe gazde bare-metal/mașini virtuale (folosind un runtime de containere).

2.3. Tehnologii și platforme

Componentă	Descriere succintă	Stivă tehnologică
Aplicație cu o singură pagină (SPA)	Interfață web (Single Page Application)	JavaScript, ReactJS 19.2.0
API Gateway	Punct de intrare echilibrat la încărcare, responsabil de filtrarea interogărilor invalide și redirecționarea cererilor către serviciile necesare	Java 25 / Cloud API Gateway
Serviciu de identitate si acces (MPass)	Serviciu responsabil de autentificarea utilizatorilor prin MPass, inclusiv aplicarea regulilor de acces pe roluri și a politicilor de securitate pentru funcționalitățile sistemului.	Java 25, Spring Boot 4.0.3
Serviciu de gestionare a utilizatorilor	Serviciu responsabil de administrarea profilurilor utilizatorilor, a rolurilor interne și a relațiilor acestora cu entitățile relevante din sistem, inclusiv actualizarea datelor de bază și gestionarea statutului contului. Serviciul oferă celorlalte componente informațiile necesare despre utilizator pentru autorizare, procese de business și audit.	Java 25, Spring Boot 4.0.3
Serviciu de nomenclatoare	Serviciu responsabil de gestionarea nomenclatoarelor și a datelor de referință utilizate în sistem, inclusiv clasificări, categorii, coduri și alte valori standard necesare funcționării proceselor de business.	Java 25, Spring Boot 4.0.3
Serviciul Notificari (MNotify)	Serviciu responsabil de generarea și transmiterea notificărilor către utilizatori prin integrarea cu platforma guvernamentală MNotify, în funcție de evenimentele și stările proceselor din sistem.	Java 25, Spring Boot 4.0.3
Serviciu procese de business	Serviciul procese de business implementează regulile de business ale SI RPBI și orchestrează fluxurile operaționale principale ale sistemului. Acesta gestionează ciclul de viață al dărilor de seamă, validarea și procesarea datelor privind prețurile de vânzare, plățile contractuale, ofertele imobiliare și rapoartele. Totodată, coordonează interacțiunea cu celelalte componente interne și declanșează integrările necesare cu serviciile guvernamentale și sistemele externe.	Java 25, Spring Boot 4.0.3

Serviciu de cautare si indexare full-text	Serviciul de căutare și indexare full-text asigură căutarea unificată în SI RPBI, atât prin căutare simplă în text integral, cât și prin căutări contextuale și criterii multiple, pentru a facilita găsirea rapidă a imobilelor, tranzacțiilor, ofertelor, rapoartelor și altor date relevante. Acesta indexează informațiile din sursele operaționale și oferă funcții avansate de căutare, precum suport pentru diacritice, fuzzy search, sugestii type-ahead, corectare ortografică, sortare, highlight și stemming/synonyms.	Java 25, Spring Boot 4.0.3, Elasticsearch
Serviciu plăți (Mpay Adaptor)	Serviciul plăți (MPay adaptor) este responsabil de integrarea SI RPBI cu serviciul guvernamental MPay, pentru inițierea și gestionarea plăților electronice aferente taxelor sau serviciilor publice, dacă astfel de scenarii sunt activate în sistem. Acesta transmite către MPay cererile de plată, recepționează răspunsurile și statusurile tranzacțiilor și pune la dispoziția proceselor de business informațiile necesare pentru confirmarea, urmărirea și reconcilierea plăților.	Java 25, Spring Boot 4.0.3
Serviciul Raportare	Serviciul Raportare asigură generarea rapoartelor statistice și analitice din SI RPBI pe baza datelor privind prețurile de vânzare, plățile contractuale, ofertele, dările de seamă și rapoartele de evaluare. Acesta permite parametrizarea rapoartelor prin filtre și agregări, previzualizarea rezultatelor în interfață și exportul acestora în formatele PDF și XLS(X).	Java 25, Spring Boot 4.0.3
Serviciu de fișiere(MSign adaptor)	Serviciu responsabil de gestionarea fișierelor și documentelor din sistem, inclusiv încărcarea, stocarea, descărcarea și transmiterea acestora către serviciul de semnare electronică prin adaptorul MSign.	Java 25, Spring Boot 4.0.3
Serviciu time-stamping (TSA adaptor)	Serviciu responsabil de integrarea SI RPBI cu un serviciu extern de marcare temporală, compatibil RFC 3161, pentru obținerea, validarea și persistarea token-urilor de timp asociate operațiunilor critice din sistem.	Java 25, Spring Boot 4.0.3, Bouncy Castle (RFC 3161)
Serviciul GIS	Serviciul GIS este responsabil de furnizarea funcționalităților geospațiale ale SI RPBI și de publicarea informațiilor pe hartă pentru geoportalul sistemului. Acesta utilizează datele din baza de date geospațială pentru interogări și selecții spațiale, heatmap/cluster, geocodare și normalizare de adrese, precum și pentru expunerea sau consumul de straturi și servicii de tip WMS/WMTS/WFS și vector tiles. Totodată, componenta deservește interfața web și rapoartele geografice, inclusiv exportul datelor și al hărților în formate precum GeoJSON și PDF.	Java 25, Spring Boot 4.0.3

GeoServer	GeoServer este responsabil de publicarea și expunerea straturilor cartografice ale SI RPBI sub formă de servicii GIS standard, necesare pentru geoportal și pentru consumul în interfața web. El deservește cerințele de interoperabilitate GIS prin furnizarea serviciilor de tip WMS/WMTS/WFS, a hărților tematice, a straturilor de bază și a accesului la datele geospațiale stocate în PostGIS. Totodată, susține funcționalitățile GIS cerute în caiet, precum afișarea straturilor, heatmap/cluster, selecția spațială și publicarea datelor geografice către aplicație și către alte sisteme.	GeoServer 2.28.3
Bază de date în memorie	Bază de date Redis utilizată pentru stocarea temporară în memorie a datelor necesare pentru sesiuni, cache, token-uri și alte operațiuni care necesită acces rapid și latență redusă.	Redis
Bază de date relațională	Baza de date relațională este responsabilă de stocarea și gestionarea datelor operaționale ale SI RPBI, precum și a informațiilor necesare funcționării proceselor de business ale sistemului. Ea asigură integritatea, consistența și accesul concurent sigur la date, oferind suport pentru operațiuni tranzacționale și pentru generarea rapoartelor statistice.	PostgreSQL
Bază de date geospațială	Baza de date geospațială stochează și gestionează datele spațiale utilizate de SI RPBI pentru reprezentarea pe hartă a informațiilor privind bunurile imobile, prețurile, ofertele și indicatorii teritoriali. Aceasta asigură suport pentru interogări geospațiale, agregări pe zone, heatmap/cluster, selecție spațială și publicarea datelor în formate standard precum GeoJSON, precum și prin servicii de tip WMS/WMTS/WFS.	PostgreSQL+PostGIS
Sistem de fișiere în rețea (NFS)	Stocarea fișierelor încărcate / generate	MinIO
Serviciu de integrare guvernamentală (MConnect adaptor)	Serviciu responsabil de integrarea sistemului cu platformele și registrele guvernamentale prin adaptorul MConnect, asigurând schimbul securizat de date și interoperabilitatea cu sisteme externe relevante.	Java 25, Spring Boot 4.0.3
Backup / Recuperare în caz de dezastru (DR)	Componentă responsabilă de realizarea copiilor de siguranță, păstrarea acestora conform politicilor de retenție și restaurarea datelor și serviciilor în caz de incident major sau indisponibilitate a infrastructurii.	Pentru baza de date – CloudNativePG; pentru fisiere - Velero File System Backup
Magistrală de evenimente (Kafka)	Componentă utilizată pentru transportul asincron al evenimentelor de audit și al fluxurilor tehnice între componentele sistemului, asigurând decuplarea	Kafka

	serviciilor și integrarea cu mecanismele de jurnalizare centralizată.	
Monitoring (Prometheus/Grafana/Alertmanager)	Componentă responsabilă de colectarea și vizualizarea metricilor de funcționare (CPU, memorie, latențe, erori), precum și de generarea alertelor automate prin Prometheus/Grafana/Alertmanager pentru monitorizarea proactivă a sistemului.	kube-prometheus-stack
Sistem centralizat de jurnalizare (ELK)	Componentă responsabilă de colectarea, agregarea, stocarea, indexarea și vizualizarea centralizată a jurnalelor generate de componentele sistemului, pentru analiză operațională, depanare și trasabilitate.	Elasticsearch, Logstash, Kibana
Serviciu de audit și publicare în MLog	Serviciu responsabil de colectarea, înregistrarea și publicarea evenimentelor de audit generate de componentele sistemului, asigurând trasabilitatea acțiunilor și transmiterea acestora către platforma guvernamentală MLog.	Java 25, Spring Boot 4.0.3

3. Abordare de dezvoltare

Abordarea de bază a echipelor S&T este de a urma bunele practici din industrie, metodologiile validate și know-how-ul extins pe care Compania l-a dobândit în proiecte similare de-a lungul anilor și care s-a transpus în principii organizaționale.

3.1. Agile/SCRUM

Intenționăm să utilizăm o metodologie Agile, denumită SCRUM, pentru executarea acestui proiect, care promovează următoarele principii:

- Toate sarcinile sunt formalizate conform planului de proiect și adăugate în Backlog, care reprezintă o colecție neordonată a tuturor sarcinilor ce urmează a fi executate.
- Activitatea este împărțită în Sprinturi, fiecare sprint având o durată de 2 săptămâni. La începutul sprintului sunt stabilite obiectivele pentru livrabilele ce urmează a fi prezentate clientului la sfârșitul celor 2 săptămâni. Sarcinile sunt prioritizate, estimate și mutate din backlog în sprinturi.
- Sarcinile sunt alocate membrilor echipei.
- În fiecare dimineață au loc ședințe stand-up, în cadrul cărora fiecare își actualizează echipa cu privire la progresul său.

- La sfârșitul sprintului, livrabilele sunt prezentate clientului, iar feedbackul este transformat, dacă este necesar, în activități pentru sprintul următor.

Această abordare prezintă avantaje importante pentru client, principalul fiind acela că acesta poate monitoriza îndeaproape progresul și poate primi livrabile funcționale la fiecare 2 săptămâni, pe baza cărora poate ajusta direcția, dacă este necesar. Acest lucru contrastează puternic cu abordarea iterativă planificată, care are cicluri mult mai lungi și prezintă riscul ca rezultatul final să fie diferit de cel anticipat de client.

3.2. Metoda Waterfall

Uneori, metoda Agile de management al proiectelor este dificil de implementat cu anumiți clienți. Pentru astfel de cazuri, putem configura împreună cu Managerul de Proiect al Clientului/Beneficiarului un proces Waterfall. Toți pașii și întreaga metodologie pot fi aplicați cu ușurință de echipa noastră. Managerii noștri de proiect au pregătire formală PMI atât în metode Agile, cât și în metode clasice.

3.3. Tehnici și abordări pentru dezvoltarea software

Practica noastră de dezvoltare software a fost consolidată și perfecționată de-a lungul mai multor ani, iar în aceasta am integrat toate bunele practici relevante, standardele din industrie și lecțiile învățate de noi înșine. În prezent, dispunem de un proces complex și sigur de dezvoltare software, cu mari părți complet automatizate și cu numeroase verificări de siguranță menite să detecteze problemele din cod cu mult înainte ca acestea să ajungă în producție sau să afecteze planificarea.

Printre bunele noastre practici se numără următoarele:

1. Revizuire de cod în mai multe etape. Niciun cod nu ajunge în repository fără a fi revizuit și îmbunătățit de cel puțin un membru senior al echipei.

2. Refactorizare continuă, o dată la 3-5 sprinturi. Periodic, oprim dezvoltarea de funcționalități noi și îmbunătățim ceea ce a fost deja realizat, pentru a preveni anumite probleme și a evita acumularea datoriei tehnice.

3. Praguri minime pentru acoperirea cu teste unitare. Avem praguri obligatorii pentru procentele de acoperire cu teste unitare, iar dacă vreun commit se situează sub acestea, nu este acceptat în repository.

4. Test-Driven-Design. Aplicăm TDD acolo unde este relevant și garantăm astfel că funcționalitățile dezvoltate corespund criteriilor de acceptanță aferente.

5. Avem configurate pipeline-uri de Continuous Integration / Continuous Delivery pentru fiecare tip de proiect pe care îl dezvoltăm. Acest lucru garantează o livrare sigură a tuturor componentelor sistemului către mediile relevante.

6. Avem o practică front-end foarte bine consolidată, care se traduce în cele din urmă printr-un UI/UX excelent, apreciat de clienți. Interfețele noastre sunt concepute să fie intuitive, moderne și responsive. Acesta a fost întotdeauna un punct forte care ne-a diferențiat de concurență.

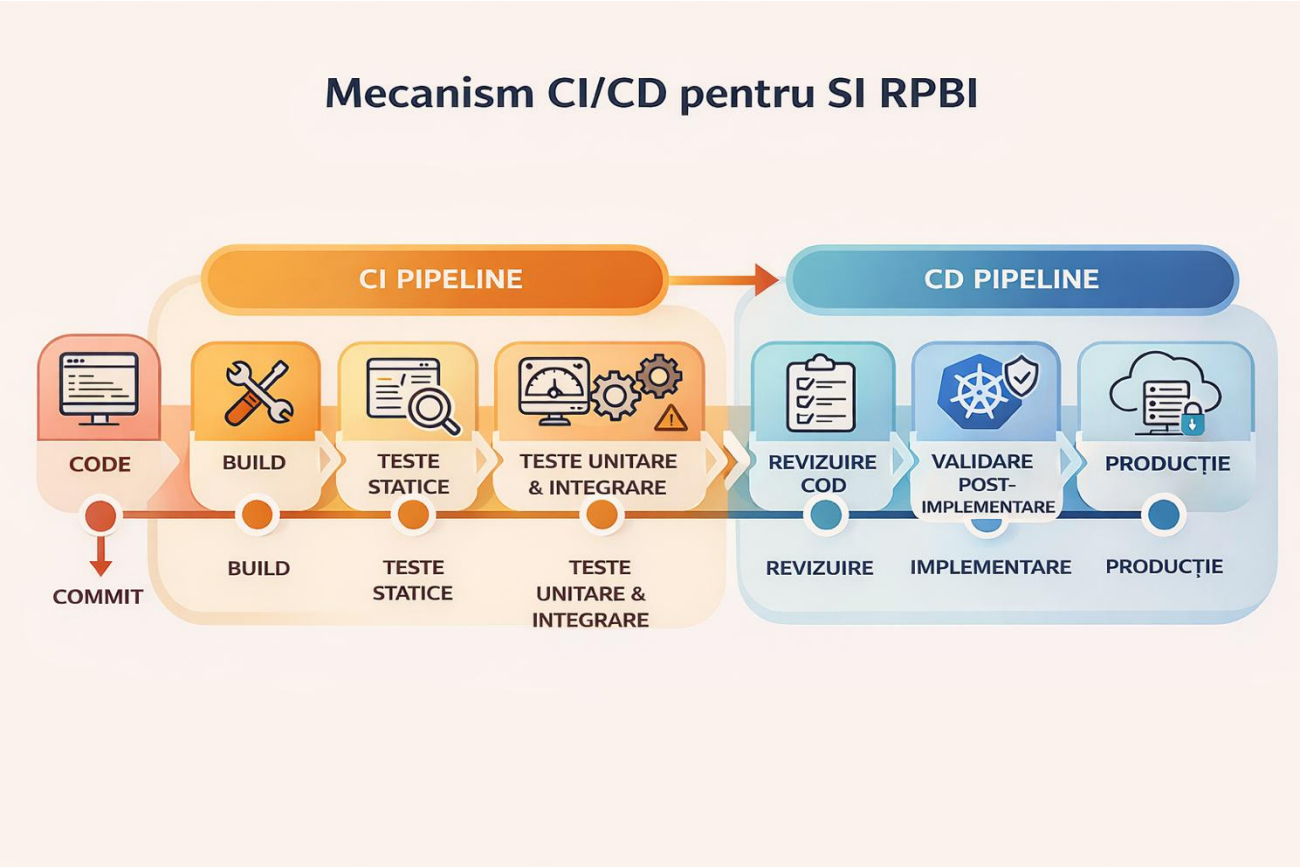
7. Folosim instrumente de prototipare foarte puternice și proiectăm interfețele împreună cu clienții noștri, trecând la implementare numai după ce clientul aprobă prototipurile.

8. Procese riguroase QA pe mai multe niveluri - testarea de regresie, testarea de sistem, testarea de acceptanță și testarea de acceptanță a utilizatorului sunt toate procese bine stabilite care ne permit să livrăm produse de înaltă calitate.

9. Bune practici de securitate: urmăm îndeaproape ghidurile OWASP, codul nostru este securizat prin proiectare și avem o serie de instrumente automate care rulează verificări de securitate pentru fiecare build și identifică vulnerabilități. revizuiri săptămânale, Prototyping, Jira, Confluence

3.4. Mecanismul CI/CD

Pentru SI RPBI, mecanismul CI/CD propus are rolul de a asigura livrări controlate, repetabile, securizate și complet auditabile pentru toate componentele soluției, de la interfețele web și serviciile backend până la procesele asincrone, migrările de bază de date și configurațiile de infrastructură. Prin această abordare, modificările de cod sunt integrate continuu, validate automat și promovate gradual între medii, până la producție, pe baza unor criterii clare de calitate și securitate.



Lanțul CI/CD va fi organizat astfel încât fiecare schimbare să parcurgă aceleași etape standardizate:

- **CODE → COMMIT**: modificările sunt versionate în repository, pe baza unei strategii de branching agreate pentru proiect, cu utilizarea obligatorie a pull request-urilor și a code review-ului înainte de integrarea în ramura principală;
- **CI pipeline**: la fiecare commit sau pull request se rulează automat build-ul componentelor aplicației, verificările statice de calitate, testele unitare, testele de integrare și validările necesare pentru a confirma că schimbarea poate fi promovată în siguranță;
- **CD pipeline**: artefactele validate sunt promovate controlat în medii succesive, de tip Development / Test / UAT / Staging / Production, cu aprobări și verificări suplimentare pentru mediile superioare;
- **Release control**: publicarea în producție se realizează exclusiv pe baza unor „quality gates” și a principiului „4-eyes”, astfel încât promovarea finală să fie verificată și aprobată formal.

În cadrul SI RPBI, mecanismul CI/CD va fi aplicat după următoarele principii:

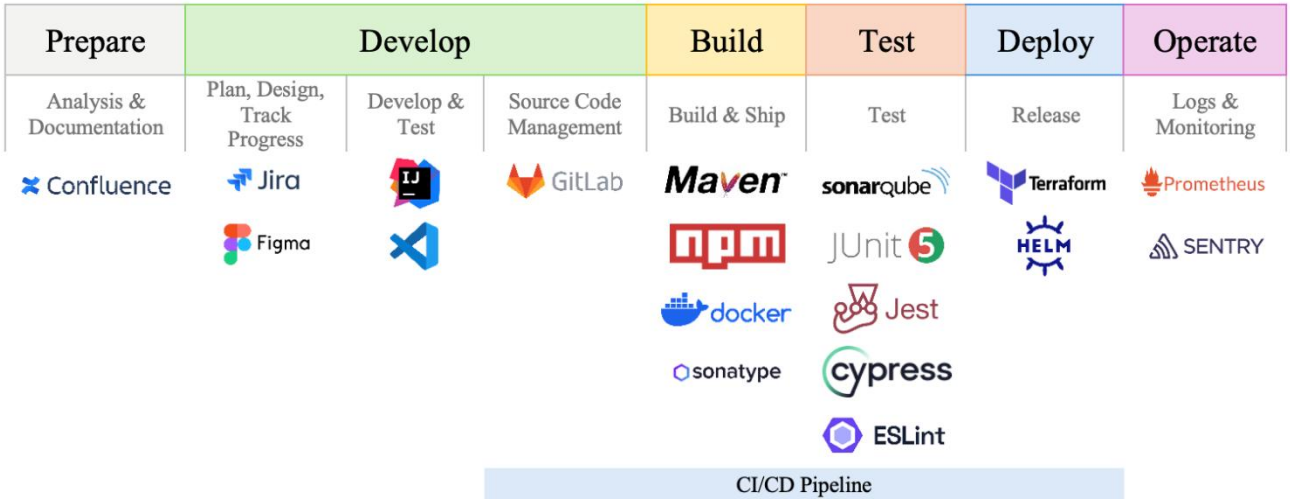
- **Artefact unic și imuabil.**
  După finalizarea build-ului, pipeline-ul va genera artefacte unice, versionate și imuabile, în mod uzual sub forma imaginilor container. Aceleași artefacte vor fi utilizate în toate mediile, fără rebuild între etape, pentru a elimina diferențele dintre Test/UAT/Prod și pentru a garanta predictibilitatea livrărilor.
- **Automatizare completă a build-ului și testării.**
  Pipeline-ul va executa automat verificările minime obligatorii pentru fiecare livrare: compilare, analiză statică de cod, testare unitară, testare de integrare și, acolo unde este relevant, testare de contract, smoke tests și validări funcționale pe fluxurile critice. Pentru SI RPBI, aceste fluxuri critice includ în special autentificarea prin MPass, semnarea prin MSign, integrarea prin MConnect, notificările prin MNotify, jurnalizarea prin MLog, precum și funcțiile esențiale de încărcare, validare, expediere și raportare.
- **Pipeline securizat.**
  În concordanță cu cerințele de securitate ale proiectului, lanțul CI/CD va include controale automate de securitate asupra codului și artefactelor livrate: scanări SAST/SCA, verificarea dependențelor, generarea SBOM, scanarea imaginilor container pentru vulnerabilități cunoscute și aplicarea politicilor de promovare. Artefactele publicate vor putea fi semnate digital, iar promovarea către medii superioare va fi blocată automat în cazul identificării unor probleme critice sau majore.
- **Configurare externă și separarea strictă a mediilor.**
  Toate configurațiile de mediu, inclusiv endpoint-urile, secretele, cheile, parametrii de integrare și politicile operaționale, vor fi externalizate față de codul sursă și injectate la rulare prin mecanismele platformei de execuție. Această abordare permite utilizarea aceluiași artefact în toate mediile și reduce riscul unor diferențe necontrolate între instalări.
- **Deploy determinist în MCloud.**
  Livrarea aplicației în mediile non-producție și producție va fi realizată pe infrastructura guvernamentală MCloud, utilizând containere Docker orchestrate prin Kubernetes. Deployment-ul va fi automatizat și reproductibil, pe baza manifestelor versionate și, după caz, a unei abordări GitOps / Infrastructure as Code pentru configurările de infrastructură și aplicație. Astfel, fiecare versiune instalată va putea fi reconstituită și urmărită exact până la codul și configurația care au generat-o.
- **Migrări controlate de bază de date.**
  Schimbările de schemă vor fi gestionate prin migrări versionate, executate automat în pipeline, într-o ordine controlată. Migrările vor fi proiectate incremental, astfel încât actualizările să poată fi aplicate în siguranță, cu risc

minim asupra datelor și cu posibilitate de revenire controlată atunci când scenariul o permite. Inițializarea și alimentarea bazelor de date cu date de referință se vor face tot automatizat, în mod idempotent.

- **Promovare graduală și rollback rapid.**
  Actualizările vor fi livrate gradual, cu verificări post-deploy și cu posibilitatea de rollback rapid în cazul apariției unor probleme. După fiecare instalare, pipeline-ul va executa validări automate asupra endpoint-urilor critice, conectivității cu serviciile externe și sănătății componentelor principale, pentru a confirma că sistemul este funcțional și coerent operațional.
- **Trasabilitate completă a fiecărui release.**
  Fiecare versiune livrată va fi identificată prin tag sau număr de release și va fi corelată cu: commit-urile sursă, rezultatele build-ului, testele executate, aprobările primite, artefactele publicate și jurnalul de deployment. În acest fel, orice versiune instalată în producție va putea fi urmărită complet din punct de vedere tehnic și operațional, în concordanță cu cerințele de audit și control.
- **Aliniere cu modelul de implementare al proiectului.**
  Deși implementarea SI RPBI urmează contractual un model de tip waterfall cu livrări incrementale, mecanismul CI/CD va susține această abordare prin integrare continuă, livrări repetabile și demo-uri periodice pe medii controlate. Astfel, fiecare increment funcțional va putea fi construit, validat, demonstrat și acceptat într-o manieră predictibilă, fără a compromite stabilitatea generală a soluției.

3.5. Lanțul de instrumente DevOps

Acestea sunt câteva dintre instrumentele pe care le utilizăm în procesele noastre de dezvoltare software. Printre beneficiile utilizării acestor instrumente se numără frecvența crescută a implementărilor, rata mai scăzută de eșec a noilor versiuni, timpul mediu mai redus de recuperare în cazul eșecurilor de implementare și un timp mai scurt până la lansarea pe piață.



3.6. Tehnici și abordări pentru asigurarea calității

Testarea este o parte integrantă și importantă a ciclului de viață al dezvoltării, care asigură funcționarea corectă a întregului produs și îndeplinirea atât a cerințelor funcționale, cât și a celor nefuncționale.

Pentru a asigura cea mai înaltă calitate a produsului și a minimiza timpul petrecut pentru remedieri, echipa implementează controlul continuu al calității, și nu doar testare simplă.

În faza de codare, calitatea este asigurată prin aplicarea principiilor TDD, scrierea de teste unitare, peer review continuu și sesiuni de refactorizare a codului. Echipa utilizează, de asemenea, instrumente automate de inspecție a codului și de evaluare a calității acestuia, precum Sonarcube și ESLint.

În faza de testare, sunt executate atât strategii de testare manuală, cât și automată, care includ: testare unitară, testare pe componente, testare de integrare, testare de acceptanță și testare de performanță, pentru a acoperi atât cerințele funcționale, cât și cele nefuncționale.

Testarea manuală acoperă consistența logică și vizuală a produsului și permite, de asemenea, explorarea unor scenarii de utilizare nestandard.

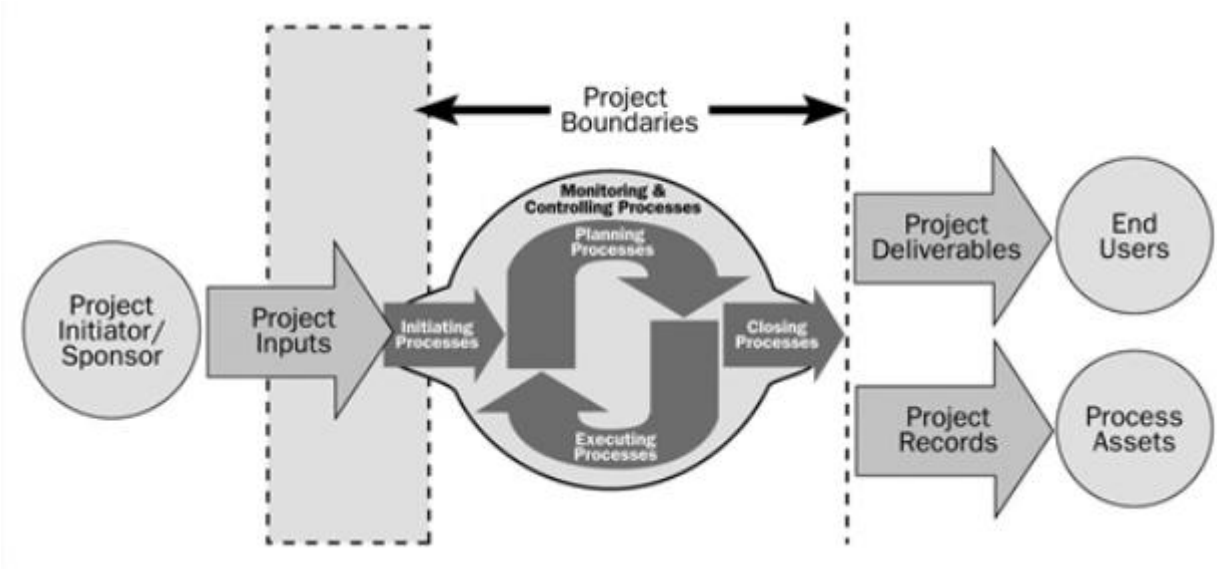
Testarea automată permite executarea rapidă, în paralel, a mai multor scenarii standardizate cu seturi de date variate.

Pentru fiecare proiect este elaborat un plan de testare individual și cuprinzător, care acoperă cerințele funcționale și nefuncționale ale aplicației, strategiile de testare, instrumentele, documentele, seturile de date și alte resurse necesare pentru a asigura funcționarea corectă a acesteia.

4. Abordare privind managementul proiectului

Proiectele sunt gestionate în conformitate cu ghidurile și recomandările elaborate de Project Management Institute - PMI și cu corpul său de cunoștințe PMBOK, monitorizând în mod continuu domeniul de aplicare, jaloanele, costul proiectului, precum și rolurile și responsabilitățile părților interesate.

În cadrul unui proiect, toate procesele pot fi împărțite în 5 grupuri distincte: Inițiere, Planificare, Execuție, Monitorizare și control, Închidere.



4.1. Inițiere

Acest grup de procese include fundamentul de bază necesar pentru crearea proiectului și definirea liniilor directoare și criteriilor în baza cărora acesta va funcționa.

În etapa de inițiere are loc semnarea contractului, sunt identificate părțile interesate atât din interiorul, cât și din exteriorul organizațiilor implicate, este definit domeniul proiectului și sunt alocate resursele financiare necesare pentru demararea acestuia.

În etapa de licitație este constituită echipa care va livra proiectul și este stabilită data începerii oficiale a proiectului (ședința de kick-off), la care vor participa reprezentanți-cheie din ambele părți.

4.2. Planificare

După autorizarea proiectului, acesta trebuie planificat. Această etapă produce un document numit Plan de Management al Proiectului. Acesta este documentul principal de planificare, care stabilește așteptările părților interesate și clarifică modul în care va fi gestionat proiectul. El trebuie să descrie domeniul de aplicare al proiectului, costurile, resursele, termenele, jaloanele, nevoile de comunicare și orice alte documente necesare pentru livrarea cu succes a proiectului.

4.3. Prototipare

Această etapă presupune elaborarea, împreună cu părțile interesate, a unor prototipuri interactive de înaltă fidelitate. Echipa va realiza mai multe iterații de prototipare UI/UX, demonstrând soluțiile părților interesate, colectând feedback și aplicându-l asupra prototipului. La final, prototipul rezultat va fi o reprezentare de înaltă fidelitate a produsului final, permițând beneficiarului și celorlalte părți interesate să își formeze o imagine foarte clară asupra rezultatului și să poată aplica din timp orice corecții asupra foii de parcurs.

4.4. Execuția proiectului

Aceasta este etapa în care se desfășoară activitatea tehnică a proiectului. Echipa de proiect este constituită și pusă la lucru, iar producerea livrabilelor proiectului este demarată. http://www.projectengineer.net/25-example-project-deliverables/

Execuția proiectului necesită coordonarea resurselor umane, gestionarea așteptărilor părților interesate și tratarea modificărilor apărute în proiect. Managerul de proiect trebuie să fie la curent cu problemele care apar și, de asemenea, să realizeze periodic previziuni privind eventualele probleme de calendar și cost, pentru a gestiona schimbările cât mai din timp. Cererile de modificare trebuie tratate și documentate pe tot parcursul acestei etape, iar părțile interesate trebuie informate.http://www.projectengineer.net/process-groups/project-execution/

Actualizările de stare și alte comunicări de proiect sunt transmise părților interesate relevante conform planului de management al proiectului. Documentele sunt stocate și arhivate, iar părțile interesate sunt gestionate potrivit planului.http://www.projectengineer.net/knowledge-areas/project-communication/

4.5. Monitorizare și control

Pe tot parcursul proiectului, managerul de proiect trebuie să monitorizeze și să controleze activitatea proiectului pentru a se asigura că livrabilele sunt realizate la timp, în limitele bugetului și la un nivel acceptabil de calitate. De asemenea, părțile interesate trebuie menținute satisfăcute, iar echipa de proiect trebuie păstrată motivată și coerentă. Activitățile de monitorizare și control se desfășoară concomitent cu faza de execuție, prin urmare cele două grupuri de procese au loc în paralel.http://www.projectengineer.net/the-4-parts-of-project-control/http://www.projectengineer.net/planning-for-project-quality/

Monitorizarea timpului (termene, jaloane etc.) și a costurilor se realizează, de regulă, prin analiza valorii câștigate (Earned Value Analysis), care oferă un semnal de avertizare timpurie puternic în cazul abaterilor din aceste zone. Calitatea livrabilelor, comunicarea cu părțile interesate și problemele potențiale cu risc ridicat reprezintă alte domenii monitorizate în mod regulat. În orice moment, monitorizarea poate conduce la modificări în proiect.http://www.projectengineer.net/introduction-to-project-management/project-control/

Dacă sunt necesare modificări în orice parte a proiectului, așa cum este documentată în planul de management al proiectului, acestea trebuie documentate și trebuie să conducă la un plan actualizat. Acest lucru include modificări ale termenelor, costurilor, livrabilelor și orice altă schimbare a proiectului, așa cum a fost acesta conceput.

4.6. Închiderea proiectului

Există aproape întotdeauna o serie de activități implicate în închiderea proiectului și trecerea la etapa următoare, iar acestea sunt de regulă foarte vizibile pentru conducere și sponsorii proiectului. Obligațiile contractuale trebuie îndeplinite și contractele închise, detaliile finale transmise, iar cerințele de finanțare finalizate.

5. Controale specifice proiectului

Echipa noastră înțelege pe deplin importanța dezvoltării unui Sistem Informațional modern, fiabil și sustenabil pentru Registrul Prețurilor Bunurilor Imobile (SI RPBI), capabil să centralizeze, valideze, stocheze, analizeze și publice date relevante privind piața imobiliară din Republica Moldova. Soluția finală pe care o vom livra va fi construită pe o arhitectură robustă, pregătită pentru cloud, orientată spre

interoperabilitate, securitate, performanță și scalabilitate, astfel încât să răspundă cerințelor operaționale ale Agenției Geodezie, Cartografie și Cadastru și să permită evoluția ulterioară a platformei. Sistemul va susține atât procesele interne de evidență, verificare și raportare, cât și publicarea controlată a informațiilor prin interfețe moderne și prin componenta GIS/geoportal dedicată.

Următoarele module și funcționalități vor fi rafinate în strânsă colaborare cu beneficiarul, cu accent pe implementarea lor modernă, bazată pe standarde:

- O arhitectură modulară, cloud-ready, implementată și operată în infrastructură containerizată, capabilă să asigure disponibilitate ridicată, mentenabilitate, securitate și extindere facilă a componentelor sistemului
- Gestionarea utilizatorilor, rolurilor și drepturilor de acces, inclusiv autentificare prin MPass și control al accesului pe bază de roluri
- Gestionarea nomenclatoarelor și a datelor de referință necesare clasificării și operării uniforme a proceselor din sistem
- Procesarea și orchestrarea fluxurilor de business aferente dărilor de seamă, inclusiv creare, numerotare, salvare, semnare, expediere, verificare, acceptare, respingere și urmărirea istoricului documentelor
- Evidența prețurilor de vânzare ale bunurilor imobile, pe baza contractelor relevante și a corelării acestora cu datele cadastrale oficiale
- Evidența plăților contractuale aferente contractelor de locațiune, arendă și superficie, inclusiv verificarea și corectarea controlată a datelor
- Evidența ofertelor listate pe platformele online, inclusiv identificarea anomaliilor și marcarea ofertelor extreme
- Generarea rapoartelor statistice și analitice, inclusiv exporturi în formate uzuale și suport pentru analiza multidimensională a datelor
- Interoperabilitate și schimb de date cu platforme și sisteme externe relevante prin MConnect, inclusiv integrarea cu IP CBI, SFS, e-Notar și alte surse oficiale necesare
- Integrare cu serviciile guvernamentale electronice precum MPass, MSign, MNotify, MLog și, după necesitate, MPay
- Publicarea și consumul datelor geospațiale printr-o componentă GIS/geoportal, cu suport pentru hartă interactivă, heatmap/cluster, selecție spațială, straturi tematice și servicii geospațiale standard
- API-uri securizate pentru integrarea cu sisteme externe, consumatori autorizați și componente interne ale ecosistemului aplicațional
- Audit și trasabilitate completă prin jurnalizarea evenimentelor de business și a acțiunilor utilizatorilor, inclusiv integrarea cu MLog

- Stocare sigură a documentelor și fișierelor asociate sistemului, inclusiv documente semnate electronic, rapoarte și atașamente, cu politici adecvate de retenție, backup și recuperare în caz de dezastru
- Mecanisme de monitorizare proactivă, jurnalizare centralizată, observabilitate operațională și măsuri de continuitate pentru operare fiabilă în regim permanent

În plus, întregul sistem va fi dezvoltat în conformitate cu bunele practici de securitate, interoperabilitate și guvernanță digitală, inclusiv cerințele privind protecția datelor, auditabilitatea, reziliența operațională și integrarea cu infrastructura tehnologică guvernamentală. Prin această abordare, SI RPBI va deveni o platformă modernă și sustenabilă, capabilă să ofere suport real pentru transparența pieței imobiliare, fundamentarea deciziilor și digitalizarea proceselor instituționale aferente domeniului.

6. Abordarea și implementarea cerințelor funcționale

Stack propus:

- **Frontend**: ReactJS, Ant Design, React Router, TanStack Query, Fetch/Axios, dayjs; componentă GIS în interfață cu OpenLayers pentru hartă, straturi, heatmap/cluster și interacțiuni spațiale.
- **Backend**: Java 25 + Spring Boot 4.0.3, Spring Web, Spring Security, Spring Validation, Spring Data JPA, Hibernate, OpenAPI/Swagger, Flyway; API-uri REST/JSON securizate, orchestrate în Kubernetes.
- **Bază de date operațională**: PostgreSQL pentru date relaționale; Redis pentru cache și accelerarea interogărilor frecvente.
- **GIS / Geoportal**: PostgreSQL + PostGIS pentru date geospațiale, GeoServer pentru publicarea serviciilor GIS standard (WMS/WMTS/WFS), OpenLayers în frontend.
- **Căutare full-text:** Elasticsearch pentru indexare și căutare full-text, sugestii, fuzzy search, highlight și filtrare rapidă; sincronizare din PostgreSQL prin joburi/evenimente.
- **Fișiere și documente**: MinIO pentru stocarea obiectelor și a documentelor generate sau semnate, inclusiv atașamente, exporturi și fișiere asociate fluxurilor operaționale ale sistemului; integrare cu serviciile aplicaționale pentru încărcare, acces securizat, retenție și recuperare.
- **Integrare eGov și sisteme externe:** adaptoare dedicate pentru MPass, MSign, MNotify, MConnect, MLog, opțional MPay; interoperabilitate cu IP CBI, e-Notar și SFS prin MConnect.

- **Messaging / evenimente**: Kafka pentru event bus intern și procesare asincronă a evenimentelor de business, notificărilor, auditului și sincronizărilor.
- **DevOps / infrastructură**: Docker, Kubernetes, GitLab CI/CD sau similar, GitOps cu ArgoCD, IaC cu Terraform/Ansible, găzduire în MCloud.
- **Observabilitate și audit**: OpenTelemetry, Prometheus, Grafana, ELK; jurnalizare centralizată și publicare în MLog.

6.1. UC01: RECEPȚIONARE NOTIFICĂRI

Cerință	Descrierea cerinței
FR 01.01	Evaluatorul certificat și Operatorul AGCC vor recepționa notificare prin *e-mail* privind parvenirea spre examinare, verificare și semnare a dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism de notificare electronică bazat pe evenimente, declanșat la expedierea dării de seamă spre examinare. Backend-ul va genera evenimentul de business aferent și îl va publica, printr-un mecanism outbox și Kafka, către serviciul intern de notificări. Acest serviciu va compune mesajul pe baza șabloanelor configurabile și va transmite notificarea prin MNotify către destinatarii stabiliți de regulile de business. Toate notificările vor fi persistate în SI RPBI, cu urmărirea stării de livrare, și vor fi jurnalizate în MLog. Interfața ReactJS va expune și un centru intern de notificări, sincronizat cu backend-ul, pentru vizualizarea istoricului mesajelor expediate și recepționate.	
FR 01.02	Evaluatorul certificat și Operatorul AGCC vor recepționa notificare prin *e-mail* privind acceptarea sau respingerea dării de seamă cu indicarea motivului de refuz.
Implementare propusă:	
Va fi implementat un mecanism de notificare automată declanșat de schimbarea statutului dării de seamă în urma examinării de către Operatorul AGCC. La acceptare sau respingere, backend-ul va genera evenimentul de business corespunzător, îl va persista tranzacțional și îl va transmite printr-un flux outbox + Kafka către serviciul intern de notificări. Acest serviciu va compune mesajul pe baza șabloanelor configurabile și îl va expedia prin MNotify, cu urmărirea completă a stării livrării. În cazul respingerii, completarea motivului va fi obligatorie înainte de emiterea notificării. Toate notificările vor fi salvate și în SI RPBI, afișate în centrul intern de notificări și jurnalizate în MLog.	
FR 01.03	Sistemul va furniza mecanisme de stocare a tuturor notificărilor parvenite în adresa utilizatorilor sistemului SI RPBI.
Implementare propusă:	
Va fi implementat un subsistem intern de notificări, în care fiecare notificare transmisă către utilizatorii SI RPBI va fi persistată ca entitate distinctă în baza de date operațională PostgreSQL. Notificările vor fi asociate documentului și destinatarului, vor avea stări urmărite pe întreg ciclul de viață și vor fi accesibile printr-un Centru de notificări disponibil în interfața web ReactJS. Serviciul intern de notificări, va actualiza aceste stări pe baza răspunsurilor primite de la MNotify, iar toate evenimentele vor fi jurnalizate în MLog. Soluția va include retenție configurabilă, căutare, filtrare, marcaj citit/necitit și raportare operațională asupra volumului și livrării notificărilor.	

6.2. UC02: EXPEDIERE NOTIFICĂRI

Cerință	Descrierea cerinței
FR 02.01	Sistemul va integra mecanismul de notificare MNotify
Implementare propusă:	
Integrarea cu serviciul guvernamental MNotify va fi realizată printr-un subsistem dedicat de notificări, care va include un adapter tehnic pentru comunicarea securizată cu MNotify, un serviciu intern de orchestrare a notificărilor și un mecanism de template management. Soluția va externaliza parametrii de integrare, va persista fiecare notificare și starea ei într-o structură internă proprie, va procesa asincron trimiterea prin Kafka și va sincroniza rezultatul livrării prin callback sau interogare de status. Vor fi implementate retry automat, deduplicare, timeout, circuit breaker, audit complet în MLog și monitorizare operațională a integrării.	
FR 02.02	Sistemul va notifica Operatorul AGCC extern despre parvenirea pentru examinare și aprobare a dării de seamă.
Implementare propusă:	
Va fi implementat un flux asincron de notificare, declanșat la emiterea evenimentului intern de tip DS_SUBMITTED. SI RPBI va persista notificarea într-un outbox tranzacțional din PostgreSQL, va publica evenimentul în Kafka, iar serviciul dedicat de notificări va transmite mesajul către Operatorul AGCC prin adaptorul MNotify. Soluția va include template management, rezolvare configurabilă a destinatarilor, retry automat, deduplicare, sincronizare a stării de livrare, audit în MLog și monitorizare operațională a integrării.	
FR 02.03	Sistemul va notifica Evaluatorul despre respingerea dării de seamă și vizualizarea motivului respingerii
Implementare propusă:	
Va fi implementat un flux asincron declanșat de evenimentul intern DS.REJECTED. La salvarea respingerii, SI RPBI va persista notificarea într-un outbox tranzacțional din PostgreSQL și va publica evenimentul în Kafka. Serviciul dedicat de notificări va prelua evenimentul, va valida existența motivului respingerii, va compune mesajul pe baza șablonului configurabil și îl va transmite către Evaluator prin adaptorul MNotify. Soluția va urmări complet starea livrării, va persista notificarea în RPBI și va jurnaliza toate operațiunile în MLog.	
FR 02.04	Sistemul va afișa motivul respingerii dării de seamă.
Implementare propusă:	
Va fi implementat prin persistarea explicită a motivului respingerii în modelul de date al documentului și prin expunerea acestuia în toate componentele relevante ale sistemului: API REST, pagina de detalii a documentului, Centrul de notificări și istoricul documentului. Motivul respingerii va fi validat server-side ca informație obligatorie, va fi păstrat ca parte a deciziei de workflow și va fi afișat integral în interfața web ReactJS. Soluția va asigura coerența dintre datele afișate în UI, conținutul notificărilor și jurnalizarea operațiunilor în MLog.	
FR 02.05	Sistemul va jurnaliza totalitatea evenimentelor de expediere a notificărilor.
Implementare propusă:	
Va fi implementat printr-un mecanism de jurnalizare completă a ciclului de viață al notificărilor, bazat pe persistența fiecărei tranziții de stare în SI RPBI și publicarea asincronă a evenimentelor de audit în MLog. Soluția va păstra pentru fiecare notificare informațiile privind crearea,	

expedierea, livrarea, eșecul, retry-urile, destinatarul, tipul evenimentului, șablonul utilizat, payload-ul minimal și corelarea cu documentul aferent. Jurnalizarea va fi structurată, imuabilă logic, corelată prin correlation id și completată cu metrici operaționale și monitorizare tehnică.

6.3. UC03: CREAREA DARE DE SEAMĂ

Cerință	Descrierea cerinței
FR 03.01	Evalatorul va selecta tipul dării de seamă: • Pentru evaluarea bunurilor imobile proprietate publică: ○ Darea de seamă privind rapoartele de evaluare a bunurilor imobile proprietate publică, întocmite de evaluatorii bunurilor imobile prin intermediul întreprinderilor de evaluare. • Pentru evaluarea bunurilor imobile proprietate privată: ○ Darea de seamă privind rapoartele de evaluare a bunurilor imobile proprietate privată, întocmite de evaluatorii bunurilor imobile prin intermediul întreprinderilor de evaluare.
Implementare propusă:	
Va fi implementat printr-un mecanism configurabil de selecție a tipului dării de seamă, bazat pe nomenclatoare persistate în PostgreSQL și expuse prin API REST. Interfața ReactJS va încărca dinamic tipurile disponibile pentru utilizatorul autentificat și va transmite selecția către backend-ul, care va valida codul ales, va crea draftul și va persista tipul documentului ca atribut structural al acestuia. Soluția va include control de acces prin Spring Security, jurnalizare în MLog și suport pentru încărcarea ulterioară a schemelor dinamice de formular în funcție de tipul selectat.	
FR 03.02	Evaluatorul pentru Darea de seamă privind rapoartele de evaluare a bunurilor imobile proprietate publică, întocmite de evaluatorii bunurilor imobile prin intermediul întreprinderilor de evaluare va anexa Raportul de evaluare.
Implementare propusă:	
Va fi implementat printr-un mecanism dedicat de gestionare a atașamentelor pentru dările de seamă de tip evaluare a bunurilor imobile proprietate publică. Soluția va stoca metadatele fișierelor în PostgreSQL, iar documentul fizic în MinIO, prin API-uri controlate de încărcare și acces, dezvoltate în Java 25 și Spring Boot 4.0.3. Backend-ul va impune încărcarea exclusiv în format PDF, limite configurabile de dimensiune, scanare antivirus, generarea checksum-ului și validarea unicității în raport cu numărul raportului. Interfața ReactJS va expune atașamentul ca un câmp obligatoriu pentru tipul relevant de dare de seamă.	
FR 03.03	Sistemul SI RPBI va afișa formularul de inserare a datelor aferente dării de seamă.
Implementare propusă:	
Va fi implementat printr-un mecanism de formulare dinamice, în care backend-ul va furniza definiția structurată a formularului aferent tipului de dare de seamă selectat, iar interfața ReactJS va randă automat câmpurile și secțiunile corespunzătoare. Soluția va utiliza scheme versionate de formular, persistate în PostgreSQL, și va permite afișarea unitară, configurabilă și extensibilă a formularelor pentru toate tipurile de dări de seamă. Renderer-ul din frontend va interpreta schema primită de la backend și va construi interfața de introducere a datelor fără a depinde de pagini hardcodate distincte pentru fiecare tip de document.	

FR 03.04	Formularul furnizat de sistem va permite inserarea datelor în câmpurile de introducere a informației, selectarea valorilor din nomenclatoare și completarea automată a compartimentelor formularului pentru informația identificată în Mpass sau din profilul companiei.
Implementare propusă:	
Va fi implementat un mecanism de formular dinamic în care sistemul va permite introducerea valorilor în câmpurile editabile, selectarea datelor standardizate din nomenclatoare și completarea automată a anumitor compartimente pe baza informațiilor identificate în MPass sau în profilul companiei. Backend-ul va furniza schema formularului, seturile de nomenclatoare și modelul inițial de date precompletate, iar interfața ReactJS va randă aceste componente într-un mod unitar și configurabil. Soluția va asigura validarea server-side a tuturor valorilor, controlul editabilității pentru câmpurile precompletate și persistența datelor în draft, cu jurnalizarea operațiunilor relevante în MLog.	
FR 03.05	Sistemul va permite redactarea dării de seamă completate de Evaluator până la aplicarea semnăturii electronice.
Implementare propusă:	
Va fi implementat un mecanism de editare controlată a dării de seamă în regim de draft, disponibil până la aplicarea semnăturii electronice MSign. Backend-ul va permite actualizarea conținutului doar pentru documentele aflate în stări premergătoare semnării, va aplica optimistic locking pentru prevenirea conflictelor de editare și va păstra istoricul versiunilor intermediare. Interfața ReactJS va permite redeschiderea și modificarea formularului atât timp cât documentul nu este semnat, iar după aplicarea cu succes a semnăturii electronice documentul va deveni needitabil la nivel de backend și frontend, cu jurnalizarea tuturor modificărilor în MLog.	
FR 03.06	Sistemul va permite ștergerea dărilor de seamă din statut „Nou”.
Implementare propusă:	
Va fi implementat un mecanism controlat de ștergere a dărilor de seamă aflate în statut „Nou”, disponibil prin endpoint REST dedicat și validat integral în backend. Soluția va verifica server-side eligibilitatea documentului pentru ștergere, drepturile utilizatorului și starea curentă a documentului, iar ștergerea va fi realizată implicit în regim de soft delete, cu păstrarea trasabilității și a integrității referențiale. Interfața ReactJS va afișa funcționalitatea de ștergere doar pentru documentele eligibile, cu confirmare explicită din partea utilizatorului, iar toate operațiunile vor fi jurnalizate în MLog.	
FR 03.07	Sistemul va notifica automat utilizatorii cu rol de Operator AGCC cu privire la expedierea dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism automat de notificare declanșat la expedierea dării de seamă. La schimbarea stării documentului în statut expediat, backend-ul va persista tranzacțional evenimentul de notificare într-un outbox din PostgreSQL și îl va publica asincron în Kafka. Serviciul dedicat de notificări va rezolva destinatarii cu rol de Operator AGCC, va compune mesajul pe baza unui șablon configurabil și îl va transmite prin adaptorul MNotify. Soluția va include urmărirea completă a stării notificării, persistența internă a acesteia în SI RPBI și jurnalizarea tuturor operațiunilor în MLog.	

FR 03.08	Sistemul va jurnaliza totalitatea activităților de creare/modificare/ștergere/ expediere a dărilor de seamă.
Implementare propusă:	
Va fi implementat printr-un mecanism automat de audit funcțional, care va jurnaliza integral operațiunile de creare, modificare, ștergere și expediere a dărilor de seamă. Backend-ul va genera evenimente de audit structurate pentru fiecare acțiune relevantă, le va persista în PostgreSQL într-un jurnal dedicat și le va transmite asincron către MLog pentru jurnalizare centralizată. Soluția va asigura imuabilitatea logică a istoricului, corelarea prin identificatori unici, trasabilitate completă asupra ciclului de viață al documentului și integrare cu mecanismele de monitorizare și observabilitate ale platformei.	

6.4. UC04: ELIBERARE NUMĂR DARE DE SEAMĂ

Cerință	**Descrierea cerinței**
FR 04.01	Sistemul va elibera un număr unic pentru fiecare dare de seamă creată.
Implementare propusă:	
Va fi implementat un serviciu dedicat de numerotare, dezvoltat în backend, care va aloca automat un număr unic fiecărei dări de seamă create. Serviciul va utiliza reguli de numerotare și secvențe persistate în PostgreSQL, va genera atomic seria și numărul conform configurației active și va salva rezultatul ca atribut imuabil al documentului. Soluția va include constrângeri de unicitate la nivel de bază de date, control tranzacțional pentru concurență ridicată și jurnalizarea evenimentului de atribuire în MLog.	
FR 04.02	Sistemul va conține mecanism care va permite configurarea principiului de generare a numărului dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism configurabil de numerotare, bazat pe reguli persistate și versionate în PostgreSQL, administrate printr-un modul securizat disponibil Administratorului tehnic. Soluția va permite definirea pattern-ului numărului, a scope-ului de unicitate și a politicii de reset, utilizând un set controlat de token-uri și validări stricte de configurare. La runtime, serviciul de numerotare va selecta regula activă în funcție de context, va obține atomic secvența corespunzătoare și va genera numărul final conform principiului configurat, cu păstrarea versiunii regulii utilizate și jurnalizarea completă în MLog.	
FR 04.03	Sistemul va apela la serviciul time stamping pentru a fixa momentul atribuirii numărului.
Implementare propusă:	
Cerința va fi implementată prin integrarea unui serviciu TSA (Time-Stamp Authority), apelat în mod obligatoriu la momentul atribuirii numărului dării de seamă. Backend-ul va invoca serviciul de time-stamping conform standardului RFC 3161, va valida răspunsul primit și va persista împreună seria și numărul documentului, tokenul TSA, serialul și identificatorul politicii aplicate. Soluția va considera atribuirea numărului finalizată doar dacă marcajul temporal a fost obținut și salvat cu succes; în caz de indisponibilitate a serviciului TSA, operația va fi blocată și nu va persista numărul. Toate evenimentele aferente vor fi jurnalizate în MLog și vor fi disponibile pentru verificare ulterioară.	

FR 04.04	Redactarea câmpurilor serie și număr nu poate fi efectuată de către utilizatorii sistemului.
Implementare propusă:	
Va fi implementat prin tratarea seriei și numărului dării de seamă ca atribute imuabile ale documentului după momentul alocării. Backend-ul va bloca orice tentativă de modificare a acestor câmpuri în API, iar interfața ReactJS le va afișa exclusiv în regim read-only după generare. Soluția va separa responsabilitatea de scriere a seriei și numărului în serviciul dedicat de numerotare, va respinge explicit tentativele de editare și le va jurnaliza ca evenimente de securitate în MLog, în conformitate cu cerințele de audit și imuabilitate ale sistemului.	

6.5. UC05: SALVARE DARE DE SEAMĂ

Cerință	Descrierea cerinței
FR 05.01	Sistemul SI RPBI va permite salvarea dărilor de seamă în zona temporară (până la aplicarea semnăturii electronice).
Implementare propusă:	
Va fi implementat un mecanism de salvare în zona temporară a dărilor de seamă, disponibil până la aplicarea semnăturii electronice. Backend-ul va persista conținutul documentului în PostgreSQL în regim de draft, cu statut intermediar, versiune proprie și moment al ultimei salvări, fără a condiționa salvarea de completitudinea finală a documentului. Interfața ReactJS va permite atât salvarea manuală, cât și autosave, iar soluția va include control de concurență prin optimistic locking, persistența rezultatelor validării și jurnalizarea evenimentelor de salvare în MLog.	
FR 05.02	Sistemul SI RPBI va permite utilizatorului să vizualizeze, modifice și să șteargă dările de seamă salvate temporar până la semnarea dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism complet de gestionare a drafturilor, care va permite utilizatorului să vizualizeze, redeschidă, modifice, resalveze și șteargă dările de seamă salvate temporar până la aplicarea semnăturii electronice. Backend-ul va expune API-uri REST dedicate pentru listarea și editarea drafturilor, va aplica reguli stricte de acces și de stare a documentului și va utiliza optimistic locking pentru prevenirea conflictelor de versiune. Interfața ReactJS va afișa lista drafturilor disponibile și va permite operarea asupra acestora până la semnare, iar toate acțiunile relevante vor fi jurnalizate în MLog.	
FR 05.03	Sistemul va notifica utilizatorul despre salvarea cu succes a dării de seamă în zona temporară.
Implementare propusă:	
Va fi implementat printr-un mecanism de notificare in-app, afișat direct în interfața ReactJS după salvarea reușită a dării de seamă în zona temporară. Backend-ul va confirma persistarea draftului în PostgreSQL și va returna momentul ultimei salvări, iar interfața va afișa atât un mesaj explicit de succes la salvarea manuală, cât și un indicator persistent de tip „Ultima salvare la HH:MM” pentru autosave și stare curentă. Soluția va separa clar succesul, eșecul și starea de salvare în curs și va corela feedback-ul vizual cu jurnalizarea operației în MLog.	

FR 05.04	Sistemul va verifica automat structura și corectitudinea completării dării de seamă la momentul salvării temporare.
Implementare propusă:	
Va fi implementat un mecanism automat de validare executat la fiecare salvare temporară a dării de seamă. Backend-ul va rula un set de reguli de validare structurale și de coerență, inclusiv verificarea câmpurilor obligatorii, a formatelor, a domeniilor de valori și a referințelor din nomenclatoare, precum și a relațiilor logice dintre câmpuri. Rezultatul validării va fi persistat împreună cu draftul în PostgreSQL și va fi returnat interfeței ReactJS sub formă de erori și avertismente afișate în UI. Soluția va permite salvarea draftului chiar și în prezența erorilor blocante, urmând ca acestea să blocheze doar etapele ulterioare de semnare și expediere, iar toate operațiunile vor fi jurnalizate în MLog.	

6.6. UC06: SEMNARE DARE DE SEAMĂ

Cerință	**Descrierea cerinței**
FR 06.01	Sistemul SI RPBI va oferi funcțional utilizatorilor de aplicare a semnăturii electronice.
Implementare propusă:	
Va fi implementat printr-un serviciu dedicat de inițiere și orchestrare a semnării electronice, disponibil direct în interfața SI RPBI pentru utilizatorii și documentele eligibile. Interfața ReactJS va expune acțiunea „Semnează electronic” doar pentru documentele aflate în stare validă de semnare, iar backend-ul va valida eligibilitatea, va genera hash-ul conținutului, va pregăti pachetul canonic de semnare și va coordona aplicarea semnăturii asupra documentului. După finalizarea cu succes a operației, documentul va trece în stare semnată, semnătura și metadatele tehnice vor fi persistate, iar după eșec documentul va rămâne în zona temporară, cu posibilitatea reluării procesului. Toate tentativele și rezultatele vor fi jurnalizate în MLog.	
FR 06.02	Sistemul SI RPBI va integra serviciul de semnătură electronică Msign.
Implementare propusă:	
Va fi implementat prin integrarea nativă a SI RPBI cu serviciul guvernamental MSign, utilizat în mod exclusiv pentru aplicarea semnăturii electronice asupra dărilor de seamă. Backend-ul va pregăti payload-ul canonic de semnare, va calcula hash-ul documentului, va iniția cererea către MSign printr-un adapter dedicat și va recepționa semnătura electronică rezultată, pe care o va atașa documentului împreună cu metadatele tehnice aferente. Soluția va suporta atât semnarea inițială, cât și contra-semnarea, va trata controlat erorile de integrare fără a persista semnături parțiale și va jurnaliza complet toate operațiunile în MLog.	
FR 06.03	Sistemul va verifica validitatea certificatului digital înainte de aplicarea semnăturii electronice.
Implementare propusă:	
Va fi implementat un mecanism obligatoriu de validare a certificatului digital înainte de aplicarea semnăturii electronice. Backend-ul va verifica valabilitatea temporală a certificatului, lanțul de încredere, starea OCSP/CRL, algoritmii acceptați și corelarea identității din certificat cu utilizatorul autentificat prin MPass. Semnarea va fi permisă doar dacă toate verificările sunt îndeplinite cu succes; în caz contrar, fluxul va fi întrerupt, documentul va rămâne în zona temporară, iar utilizatorul va primi un mesaj explicit privind cauza refuzului. Metadatele certificatului și rezultatul validării vor fi persistate și jurnalizate în MLog.	

FR 06.04	Sistemul va notifica utilizatorul cu privire la succesul sau eșecul aplicării semnăturii electronice.
Implementare propusă:	
Va fi implementat un mecanism de feedback in-app privind rezultatul operației de semnare electronică. Backend-ul va întoarce un răspuns structurat după finalizarea fluxului de semnare, indicând explicit succesul sau eșecul operației, iar interfața ReactJS va afișa imediat mesajul corespunzător utilizatorului. În caz de succes, documentul va fi actualizat în stare semnată și utilizatorul va primi confirmarea vizuală a operației; în caz de eșec, utilizatorul va primi un mesaj explicit, iar documentul va rămâne în zona temporară pentru reluarea procesului. Toate rezultatele vor fi jurnalizate în MLog.	
FR 06.05	În cazul în care semnarea eșuează, sistemul SI RPBI va păstra darea de seamă în zona temporară și va permite reluarea procesului.
Implementare propusă:	
Va fi implementat un mecanism de tratare controlată a eșecurilor din fluxul de semnare, astfel încât documentul să rămână în zona temporară și procesul să poată fi reluat fără pierderea conținutului. Backend-ul va separa tentativa de semnare de semnarea finalizată, va promova documentul în stare semnată doar după finalizarea cu succes a tuturor verificărilor și integrărilor necesare și nu va persista semnături parțiale în cazul unui eșec. Interfața ReactJS va afișa un mesaj explicit privind rezultatul nereușit și va permite reluarea procesului, iar toate tentativele și eșecurile vor fi jurnalizate în MLog.	
FR 06.06	În momentul semnării dării de seamă sistemul va apela serviciul de time stamping pentru salvarea marcajului de timp când a fost efectuată acțiunea.
Implementare propusă:	
Va fi implementat prin integrarea obligatorie a unui serviciu TSA (Time-Stamp Authority) în fluxul de semnare electronică al SI RPBI. Backend-ul va apela serviciul de time-stamping conform standardului RFC 3161 imediat după obținerea semnăturii electronice, va valida răspunsul primit și va persista împreună semnătura, tokenul TSA, serialul și identificatorul politicii aplicate. Soluția va considera semnarea finalizată doar dacă marcajul temporal a fost obținut și salvat cu succes; în caz de eșec al serviciului TSA, documentul va rămâne în zona temporară, procesul va putea fi reluat, iar toate operațiunile vor fi jurnalizate în MLog.	

6.7. UC07: EXPEDIERE DARE DE SEAMĂ

Cerință	Descrierea cerinței
FR 07.01	Sistemul SI RPBI va pune la dispoziția utilizatorului funcțional de expediere spre acceptare a dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism dedicat de expediere spre acceptare a dării de seamă, disponibil utilizatorului printr-o operație controlată de backend. Interfața ReactJS va afișa acțiunea „Expediază spre acceptare” doar pentru documentele eligibile, pe baza metadatelor operaționale furnizate de API, iar backend-ul va valida autoritativ drepturile utilizatorului, starea documentului și posibilitatea inițierii fluxului de submit. Soluția va actualiza statusul documentului în	

PostgreSQL, va proteja împotriva expedierilor duplicate prin mecanisme de idempotență și va jurnaliza operația în MLog.	
FR 07.02	Sistemul va verifica dacă darea de seamă este semnată electronic înainte de expedierea spre aprobare.
Implementare propusă:	
Va fi implementat un mecanism obligatoriu de verificare a semnării electronice înainte de expedierea dării de seamă spre aprobare. Backend-ul va valida existența unei semnături electronice valide asociate versiunii curente a documentului și va permite inițierea fluxului de submit numai dacă această condiție este îndeplinită. Interfața ReactJS va expune acțiunea de expediere doar pentru documentele semnate, iar în cazul în care utilizatorul încearcă transmiterea unui document nesemnat, sistemul va bloca operația, va afișa un mesaj explicit și va păstra documentul în starea anterioară. Toate tentativele și refuzurile aferente vor fi jurnalizate în MLog.	
FR 07.03	Sistemul va oferi utilizatorului un mesaj de confirmare a expedierii cu succes.
Implementare propusă:	
Va fi implementat un mecanism de confirmare in-app a expedierii reușite, afișat utilizatorului imediat după finalizarea cu succes a operației de submit. Backend-ul va returna un răspuns structurat după executarea verificărilor finale, aplicarea marcajului temporal și schimbarea statusului documentului, iar interfața ReactJS va afișa un mesaj explicit de confirmare și va actualiza starea documentului în „Expediat” / „În examinare AGCC”. Opțional, soluția va putea transmite aceeași confirmare și prin e-mail, utilizând mecanismul general de notificări al platformei. Toate operațiunile aferente vor fi jurnalizate în MLog.	
FR 07.04	Sistemul va notifica Operatorul AGCC despre expedierea dării de seamă spre aprobare.
Implementare propusă:	
Va fi implementat printr-un mecanism automat de notificare către Operatorul AGCC, declanșat după finalizarea cu succes a operației de expediere a dării de seamă spre aprobare. Backend-ul va persista tranzacțional evenimentul de notificare într-un outbox din PostgreSQL, îl va publica asincron în Kafka, iar serviciul dedicat de notificări va compune mesajul și îl va transmite prin adaptorul MNotify către destinatarii cu rol de Operator AGCC. Soluția va urmări complet starea notificării, va permite reîncercarea automată în caz de indisponibilitate a serviciului extern și va jurnaliza toate operațiunile în MLog, în corelație cu evenimentul de submit al documentului.	
FR 07.05	În momentul expedierii dării de seamă sistemul va apela serviciul de time stamping pentru salvarea marcajului de timp când a fost efectuată acțiunea.
Implementare propusă:	
Va fi implementat prin integrarea obligatorie a unui serviciu TSA (Time-Stamp Authority) în fluxul de expediere a dării de seamă spre aprobare. Backend-ul va apela serviciul de time-stamping conform standardului RFC 3161 înainte de finalizarea submit-ului, va valida răspunsul primit și va persista împreună statusul de expediere, momentul operației, tokenul TSA, serialul și identificatorul politicii aplicate. Soluția va considera expedierea finalizată doar dacă marcajul temporal a fost obținut și salvat cu succes; în caz de eșec al serviciului TSA, documentul va rămâne în starea semnată, operația va putea fi reluată, iar toate evenimentele aferente vor fi jurnalizate în MLog.	

FR 07.06	În cazul care Evaluatorul nu va reuși să depună darea de seamă până la finele trimestrului, sistemul îi va permite să depună darea de seamă în decurs de 20 zile după care va bloca posibilitatea de depunere a dării de seamă pentru acel trimestru.
Implementare propusă:	
Va fi implementat un mecanism de control al termenului de depunere pe trimestru, executat server-side în backend la fiecare tentativă de expediere a dării de seamă. Soluția va calcula automat trimestrul de raportare, sfârșitul perioadei și fereastra de grație de 20 zile (parametru configurabil), permițând submit-ul numai în interiorul acestor limite. După expirarea perioadei de grație, sistemul va bloca expedierea pentru trimestrul respectiv, va returna un mesaj explicit utilizatorului și va păstra evenimentul în jurnalul de audit MLog. Interfața ReactJS va afișa starea termenului și va reflecta vizual posibilitatea sau imposibilitatea expedierii, fără a înlocui validarea autoritativă din backend.	
FR 07.07	În cazul care darea de seama a fost respinsă pentru corectare, sistemul îi va oferi posibilitatea de ajustare a dării de seamă în decurs de 5 zile după care funcționalul va fi blocat.
Implementare propusă:	
Va fi implementat un mecanism de control al ferestrei de corectare după respingere, în care backend-ul va calcula automat și va aplica un termen de 5 zile (parametru configurabil) de la momentul respingerii dării de seamă. În această perioadă, documentul va putea fi redeschis, ajustat și reexpediat de către evaluator, iar după expirarea termenului funcționalul va fi blocat server-side, cu refuz explicit al operațiilor nepermise. Interfața ReactJS va afișa statusul de respingere, termenul-limită de corectare și va reflecta vizual posibilitatea sau imposibilitatea reexpedierii, fără a înlocui validarea autoritativă din backend. Toate evenimentele relevante vor fi jurnalizate în MLog.	
FR 07.08	Odată ce Evaluatorul a semnat si trimis darea de seamă, acesta nu mai poate efectua careva modificări. Modificările pot fi efectuate doar în cazul care darea de seamnă a fost respinsă pentru corectare sau depune dare de semană corectată.
Implementare propusă:	
Va fi implementat un mecanism de imuabilitate post-expediere, aplicat server-side în backend. După semnarea și transmiterea cu succes a dării de seamă, documentul va fi trecut într-o stare în care evaluatorul nu va mai putea efectua modificări asupra conținutului sau atașamentelor, iar toate operațiile de editare vor fi refuzate prin API. Interfața ReactJS va reflecta această stare prin afișarea documentului în regim read-only, cu excepțiile prevăzute de fluxul de corectare după respingere și de depunerea unei dări de seamă corectate. Soluția va păstra trasabilitatea completă a acestor tranziții și va jurnaliza în MLog atât activarea imuabilității, cât și orice tentativă nepermisă de modificare.	
FR 07.09	În cazul depunerii dării de seamă corectate sistemul va lua în considerație ultima dare de seamă depusă.
Implementare propusă:	
Va fi implementat un mecanism de versionare a depunerilor corectate, în care fiecare reexpediere validă a unei dări de seamă va fi păstrată în istoric, iar ultima depunere finalizată cu succes va fi marcată drept versiune în vigoare. Backend-ul va actualiza în mod controlat	

indicatorul versiunii curente, va păstra trasabilitatea tuturor depunerilor anterioare și va utiliza ultima depunere corectată în toate fluxurile operaționale, de afișare și de raportare. Interfața ReactJS va evidenția clar versiunea activă și istoricul versiunilor anterioare, iar toate tranzițiile relevante vor fi jurnalizate în MLog.	
FR 07.10	Dările de seamnă nu vor putea fi semnate și transmise dacă nu au fost completate toate câmpurile obligatorii, sau nu a fost atașat raportul aferent dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism de validare blocantă a completitudinii documentului înainte de semnare și înainte de expediere. Backend-ul va verifica existența tuturor câmpurilor obligatorii definite de schema formularului și prezența raportului aferent, acolo unde acesta este obligatoriu, și va permite continuarea fluxului numai dacă documentul este complet. În caz contrar, operația de semnare sau expediere va fi refuzată, statusul documentului va rămâne neschimbat, iar interfața ReactJS va afișa explicit lista erorilor și a elementelor lipsă. Toate refuzurile și verificările relevante vor fi jurnalizate în MLog.	

6.8. UC08: ȘTERGERE DARE DE SEAMĂ

Cerință	Descrierea cerinței
FR 08.01	Sistemul SI RPBI va dispune de funcțional de ștergere a dării de seamă.
Implementare propusă:	
Va fi implementat un mecanism dedicat de ștergere controlată a dărilor de seamă aflate în zona temporară, disponibil prin API REST și validat integral în backend. Soluția va verifica server-side eligibilitatea documentului, drepturile utilizatorului și starea curentă a acestuia, iar ștergerea va fi realizată implicit în regim de soft-delete, cu păstrarea trasabilității, a retenției și a integrității referențiale. Interfața ReactJS va afișa funcționalitatea de ștergere doar pentru documentele eligibile și va solicita confirmare explicită înainte de executare, iar toate operațiunile relevante vor fi jurnalizate în MLog.	
FR 08.02	O dare de seamă poate fi ștearsă doar dacă nu a parcurs integral ciclul (semnare, expediere,acceptare)
Implementare propusă:	
Va fi implementat printr-un mecanism server-side de verificare a eligibilității documentului pentru ștergere, bazat pe starea acestuia în ciclul de viață. Backend-ul va permite ștergerea numai pentru dările de seamă aflate în statut „Nou” sau „Salvat (draft)” și va refuza explicit operația dacă documentul a parcurs etapele de semnare, expediere sau acceptare. Soluția va aplica această regulă autoritativ în API, va proteja împotriva conflictelor de stare prin verificări tranzacționale și optimistic locking și va jurnaliza în MLog atât ștergerile permise, cât și tentativele respinse de ștergere a documentelor neeligibile.	

6.9. UC09: EXTRAGERE RAPOARTE

Cerință	Descrierea cerinței
FR 09.01	Sistemul va oferi un set de rapoarte statistice destinate utilizatorilor. La această categorie de rapoarte pot fi următoarele: • Raport privind ....

	• Raport privind ....
Implementare propusă:	
Va fi implementat un serviciu dedicat de raportare statistică, în care SI RPBI va pune la dispoziția utilizatorilor autorizați un catalog de rapoarte parametrizabile asupra datelor operaționale și analitice ale sistemului. Serviciul de raportare va expune definițiile de rapoarte, va valida parametrii și drepturile de acces și va executa agregările necesare pentru generarea rezultatelor, iar interfața ReactJS va permite selectarea raportului, filtrarea datelor și previzualizarea rezultatelor în format tabelar și, unde este relevant, grafic sau cartografic. Soluția va include control RBAC, jurnalizarea rulărilor în MLog și optimizări de performanță prin structuri agregate și execuții asincrone pentru rapoarte complexe.	
FR 09.02	Rapoartele generate în cadrul sistemului trebuie să dispună de funcțional de exportare cel puțin în următoarele formate: PDF, XLS(X).
Implementare propusă:	
Va fi implementat un mecanism de export al rapoartelor generate în formatele PDF și XLS(X), integrat în serviciul de raportare al SI RPBI. Backend-ul va genera fișierele de export pe baza rezultatului raportului și a parametrilor aplicați, păstrând în conținutul acestora metadatele relevante, precum perioada, data generării, versiunea și sursa. Interfața ReactJS va permite exportul direct al rapoartelor după previzualizare, iar pentru rapoartele voluminoase soluția va suporta procesare asincronă și descărcare ulterioară. Exporturile vor respecta regulile de acces, confidențialitate și audit ale platformei și vor fi jurnalizate în MLog.	

6.10. UC010: VIZUALIZAREA ISTORICULUI DOCUMENTULUI/DĂRII DE SEAMĂ

Cerință	Descrierea cerinței
FR 10.01	Sistemul trebuie să permită utilizatorului autorizat vizualizarea istoricului complet al acțiunilor efectuate asupra unei dări de seamă.
Implementare propusă:	
Va fi implementat un mecanism dedicat de vizualizare a istoricului complet al acțiunilor efectuate asupra unei dări de seamă, disponibil exclusiv utilizatorilor autentificați și autorizați. Backend-ul va colecta evenimentele documentului din serviciul de audit și le va expune prin API într-o formă structurată, iar interfața ReactJS va afișa acest istoric într-o listă sau tabel cronologic descrescător, cu posibilități de filtrare și consultare a detaliilor fiecărui eveniment. Soluția va trata istoricul ca informație generată automat, imuabilă și read-only, va asigura control RBAC asupra accesului și va jurnaliza separat evenimentele de vizualizare a istoricului în MLog.	
FR 10.02	Sistemul va afișa o listă cronologică a acțiunilor efectuate asupra documentului.
Implementare propusă:	
Va fi implementat un mecanism de ordonare cronologică descrescătoare a istoricului documentului, aplicat autoritativ în backend și reflectat unitar în interfața ReactJS. Backend-ul va extrage evenimentele asociate documentului din subsistemul de audit, le va sorta implicit după data și ora producerii în ordine descrescătoare și le va expune prin API într-o formă pregătită pentru afișare tabelară sau listă cronologică. Soluția va păstra aceeași regulă de ordonare și la filtrare, căutare sau export, va optimiza interogarea prin indexuri și caching pentru ultimele evenimente și va jurnaliza accesarea istoricului în MLog.	
FR 10.03	Fiecare acțiune va include următoarele informații:

	• Tipul acțiunii (ex. „Document creat”, „Document modificat”„Document semnat”, „Document expediat”) • Data și ora efectuării acțiunii • Numele complet al utilizatorului care a efectuat acțiunea • Modificarea statusului documentului (de ex. din „Nou” în „Salvat”).
Implementare propusă:	
Va fi implementat un model standardizat de eveniment istoric, generat automat de sistem pentru fiecare acțiune relevantă asupra unei dări de seamă. Backend-ul va persista și expune pentru fiecare înregistrare de istoric tipul acțiunii, data și ora producerii, numele complet al utilizatorului care a executat operația și tranziția de statut a documentului, acolo unde aceasta există. Interfața ReactJS va afișa aceste informații într-o formă unitară și ușor de urmărit, iar soluția va păstra coerența dintre evenimentele de business, audit și istoricul vizibil utilizatorilor, cu respectarea regulilor de confidențialitate și jurnalizare în MLog.	
FR 10.04	Informațiile vor fi prezentate sub formă de tabel sau listă cronologică descrescătoare (cea mai recentă acțiune prima). Coloanele afișate: • Data/Ora acțiunii • Utilizator • Acțiunea efectuată (ex. Documentul a fost creat) • Statut anterior și Statut nou (ex. nou>salvat, semnat>expediat)
Implementare propusă:	
Va fi implementat un mecanism standardizat de prezentare a istoricului documentului în format tabelar, cu posibilitatea de afișare alternativă ca listă cronologică, ambele ordonate descrescător astfel încât cea mai recentă acțiune să fie afișată prima. Backend-ul va furniza evenimentele istorice într-o structură pregătită pentru afișare, iar interfața ReactJS va prezenta pentru fiecare înregistrare coloanele Data/Ora acțiunii, Utilizator, Acțiunea efectuată, Statut anterior și Statut nou, cu filtre rapide și acces la metadate suplimentare. Soluția va asigura performanță prin indexare și caching, va păstra coerența dintre afișare și export și va jurnaliza accesarea istoricului în MLog.	
FR 10.05	Funcționalitatea este disponibilă doar pentru utilizatorii autentificați care dețin dreptul de acces la documentul respectiv.
Implementare propusă:	
Va fi implementat un mecanism de control al accesului la istoricul documentului, bazat pe autentificare prin MPass și autorizare server-side la nivel de document. Backend-ul va permite accesul la istoricul unei dări de seamă numai utilizatorilor autentificați care dețin drepturile necesare asupra documentului respectiv, aplicând validări RBAC și măsuri anti-enumerare. Interfața ReactJS va expune funcționalitatea de istoric doar pentru documentele accesibile utilizatorului curent, iar toate accesările permise vor fi jurnalizate în MLog.	
FR 10.06	Istoricul documentului este generat automat de sistem și nu poate fi modificat manual de către utilizatori.
Implementare propusă:	
Va fi implementat un mecanism de generare automată și imuabilă a istoricului documentului, bazat pe evenimente write-once persistate de sistem la fiecare acțiune relevantă asupra dării de	

seamă. Backend-ul va crea automat înregistrări istorice standardizate pentru operațiile de business și le va păstra într-o structură dedicată, separată de datele curente ale documentului, fără a expune funcționalități de editare sau ștergere manuală. Interfața ReactJS va permite doar vizualizarea, filtrarea și, după caz, exportul istoricului, în regim strict read-only. Soluția va asigura integritatea logică a jurnalului, va permite asocierea de marcaje temporale pentru evenimente critice și va jurnaliza separat accesările istoricului în MLog.

6.11. UC11: IMPRIMARE DARE DE SEAMĂ

Cerință	Descrierea cerinței
FR 11.01	Funcționalitatea de imprimare a dărilor de seamă trebuie să fie accesibilă doar pentru documentele acceptate.
Implementare propusă:	
Va fi implementat un mecanism de validare a eligibilității documentului pentru imprimare, aplicat autoritativ în backend. Soluția va permite inițierea fluxului de imprimare numai pentru dările de seamă aflate în starea „Acceptat”, va bloca server-side orice tentativă de generare pentru documente aflate în alte stări și va reflecta această regulă și în interfața ReactJS prin afișarea controlată a acțiunii „Imprimă / PDF”. Documentele imprimate vor reflecta exact versiunea acceptată, iar operațiile de imprimare și refuzurile relevante vor fi jurnalizate în MLog.	
FR 11.02	Sistemul SI RPBI va genera pentru imprimare un fișier extern în format PDF.
Implementare propusă:	
Va fi implementat un mecanism dedicat de generare a unui fișier extern în format PDF, orchestrat de backend în cadrul subsistemului de imprimare. Soluția va selecta șablonul activ, va mapa datele documentului acceptat pe structura de layout, va aplica formatările necesare și va produce un PDF pregătit pentru previzualizare, descărcare și tipărire. Fișierul generat va include metadate relevante, precum identificatorul documentului, versiunea șablonului, data și ora generării și hash-ul de integritate, iar pentru documentele voluminoase sistemul va suporta generare asincronă. Toate operațiunile de generare vor fi jurnalizate în MLog.	
FR 11.03	Darea de seamă destinată imprimării va arăta similar cu exemplarul aprobat de către AGCC.
Implementare propusă:	
Va fi implementat un mecanism de generare a documentului imprimabil pe baza unor șabloane oficiale versionate, astfel încât PDF-ul rezultat să fie conform, din punct de vedere structural și vizual, cu exemplarul aprobat de către AGCC. Backend-ul va selecta la runtime șablonul activ și valid, va mapa datele documentului acceptat pe layout-ul aprobat și va aplica regulile de formatare privind antetul, structura secțiunilor, tabelarea, numerotarea și paginarea. Soluția va păstra trasabilitatea versiunii de șablon utilizate, va bloca generarea în lipsa unui șablon valid și va jurnaliza fiecare operațiune de imprimare în MLog.	

6.12. UC12: JURNALIZARE EVENIMENTE

Cerință	Descrierea cerinței
FR 12.01	SI RPBI va conține mecanism de jurnalizare a tuturor evenimentelor aferente utilizării.
Implementare propusă:	

Va fi implementat un mecanism complet de jurnalizare a tuturor evenimentelor relevante generate de utilizarea SI RPBI, bazat pe generarea automată de evenimente de audit în backend, persistența lor într-un jurnal intern și propagarea ulterioară către subsistemul guvernamental MLog. Soluția va trata jurnalizarea ca un subsistem transversal al platformei, aplicat tuturor fluxurilor de utilizare, va păstra evenimentele într-o structură standardizată și imuabilă, va asigura transmiterea non-blocantă și reziliență a acestora și va permite consultarea contextuală a jurnalelor de către utilizatorii autorizați.	
FR 12.02	În calitate de mecanism de jurnalizare trebuie integrat subsistemul de jurnalizare din cadrul MLog.
Implementare propusă:	
Va fi implementat prin integrarea nativă al serviciului de audit cu platforma guvernamentală MLog, utilizată ca mecanism centralizat de jurnalizare a evenimentelor. Fiecare serviciu din cadrul SI RPBI va persista fiecare eveniment de audit în jurnalul intern și îl va transmite asincron către MLog printr-un modul dedicat de publicare, cu urmărirea stării de livrare pentru fiecare înregistrare. Soluția va utiliza cozi interne, retry automat, backoff și circuit breaker pentru a asigura transmiterea non-blocantă și reziliență către MLog, iar în caz de indisponibilitate a subsistemului extern, evenimentele vor fi buffer-izate local și retransmise automat după restabilire.	
FR 12.03	Sistemul va jurnaliza cel puțin următoarele categorii de evenimente: • Autentificarea și autorizarea utilizatorilor • Accesarea, vizualizarea, modificarea sau ștergerea documentelor • Crearea, semnarea, expedierea documentelor • Schimbarea statutului documentelor
Implementare propusă:	
Va fi implementat prin definirea și utilizarea unui set standardizat de categorii minime de evenimente de audit, generate automat de serviciile SI RPBI în toate fluxurile relevante ale platformei. Soluția va jurnaliza cel puțin autentificarea, autorizarea și deautentificarea utilizatorilor, accesarea, vizualizarea, modificarea și ștergerea documentelor, precum și crearea, semnarea, expedierea și schimbările de statut ale documentelor, precum si orice tip de acces la bazele de date. Evenimentele vor fi persistate într-o structură unitară de audit, vor fi transmise către MLog și vor putea fi filtrate și analizate în panoul de audit și în vizualizările contextuale ale sistemului.	
FR 12.04	Fiecare eveniment jurnalizat va conține următoarele câmpuri obligatorii: • Data și ora exactă a evenimentului (timestamp) • Tipul evenimentului • Descrierea evenimentului • ID-ul unic al utilizatorului • Numele complet al utilizatorului • Adresa IP (dacă este disponibilă)
Implementare propusă:	
Va fi implementat un model standardizat al evenimentelor de audit, utilizat unitar în toate fluxurile aplicației și populat automat de catre serviciile SI RPBI. Fiecare eveniment jurnalizat va conține în mod obligatoriu data și ora exactă a producerii, tipul evenimentului, descrierea standardizată a acțiunii, ID-ul unic al utilizatorului, numele complet al utilizatorului și adresa IP, dacă aceasta este disponibilă. Soluția va include un mecanism de îmbogățire și validare internă a	

metadatelor de audit înainte de persistență și transmitere către MLog, precum și suport pentru câmpuri suplimentare relevante, cum ar fi tranziția de statut a documentului.

6.13. UC13: EVIDENȚA PLĂȚILOR CONTRACTUALE

Cerință	Descrierea cerinței
FR 13.01	Sistemul va afisa datele din două categorii de contracte: 1. Contracte de vânzare-cumpărare 2. Contracte de locațiune, arendă, superficie
Implementare propusă:	
Va fi implementat un mecanism de afișare separată a datelor contractuale în două categorii distincte: contracte de vânzare-cumpărare și contracte de locațiune, arendă, superficie. Backend-ul va agrega și expune datele contractuale într-o structură clasificată pe categorie, iar interfața ReactJS va prezenta aceste informații în secțiuni sau tab-uri distincte, astfel încât utilizatorul să poată naviga clar între cele două tipuri de contracte. Soluția va păstra separarea logică și funcțională a surselor de date, va respecta regulile de acces ale modulului și va asigura trasabilitatea operațiunilor prin integrarea cu mecanismele de audit ale platformei.	
FR 13.02	Datele contractelor de vânzare-cumpărare se vor afișa în forma tabelară și vor conține următoarele date: - numărul cadastral; - adresa completă; - tipul bunului; - modul de folosință a bunului imobil; - categoria de destinație (în cazul terenurilor); - suprafața bunului; - parametrii tehnici ai bunului; - prețul tranzacției; - valuta tranzacției; - data încheierii contractului de vânzare-cumpărare; - statut contract; - vizualizarea contractul de vânzare-cumpărare.
Implementare propusă:	
Va fi implementat un mecanism de afișare tabelară a contractelor de vânzare-cumpărare, bazat pe un read model dedicat expus de backend și randat în interfața ReactJS. Soluția va prezenta pentru fiecare contract toate câmpurile obligatorii prevăzute în caietul de sarcini, respectiv numărul cadastral, adresa completă, tipul bunului, modul de folosință, categoria de destinație pentru terenuri, suprafața, parametrii tehnici ai bunului, prețul tranzacției, valuta, data contractului, statutul și acțiunea de vizualizare a contractului. Tabelul va fi optimizat pentru listare rapidă, filtrare și navigare către documentul contractual asociat și va respecta cerințele de trasabilitate și audit ale platformei.	
FR 13.03	Datele contractelor de locațiune, arendă, superficie se vor afișa în formă tabelară și vor conține următoarele date: - numărul cadastral; - adresa completă; - tipul bunului;

	- modul de folosință a bunului; - categoria de destinație (în cazul terenurilor); - suprafața bunului; - parametri tehnici ai bunului; - plata contractuală; - valuta plății; - tipul plații contractulae (lunare, anuale); - data încheierii contractului; - statut contract; - vizualizarea contractului de locațiune, arendă, superficie.
Implementare propusă:	
Va fi implementat un mecanism de afișare tabelară a contractelor de locațiune, arendă și superficie, bazat pe un read model dedicat expus de backend și randat în interfața ReactJS. Soluția va prezenta pentru fiecare contract toate câmpurile obligatorii prevăzute în caietul de sarcini, respectiv numărul cadastral, adresa completă, tipul bunului, modul de folosință, categoria de destinație pentru terenuri, suprafața, parametrii tehnici ai bunului, plata contractuală, valuta plății, tipul plății contractuale, data încheierii contractului, statutul și acțiunea de vizualizare a contractului. Tabelul va fi optimizat pentru listare rapidă, filtrare și acces la documentul contractual asociat și va utiliza date sincronizate prin MConnect din sursele oficiale relevante, cu trasabilitate și audit în mecanismele platformei.	
FR 13.04	Informația privind contractele de locațiune, arendă, superficie va fi preluată prin platforma de interoperabilitate MConnect din: - Sistemul informațional al SFS, informațiile despre contractele încheiate de către persoanele fizice, care nu desfășoară activitate de întreprinzător și care transmit persoanelor specificate la art.54 din Codul fiscal nr. 1163/1997, precum și altor persoane decât cele specificate la art. 90 din codul prenotat, în posesie și/sau folosință (locațiune/arendă/uzufruct/superficie) proprietate imobiliară; - Sistemul informațional Cadastrul Bunurilor Imobile.
Implementare propusă:	
Va fi implementat un mecanism de integrare și preluare automată a informațiilor privind contractele de locațiune, arendă și superficie prin platforma guvernamentală de interoperabilitate MConnect. Serviciu de integrare guvernamentală (MConnect adaptor) va interoga, recepționa, valida și normaliza datele provenite din Sistemul informațional al Serviciului Fiscal de Stat și din Sistemul informațional Cadastrul Bunurilor Imobile, persistându-le într-un model intern unificat utilizabil de SI RPBI. Soluția va suporta ingestie inițială, refresh periodic, corelare între sursele externe și tratarea controlată a erorilor de integrare, asigurând trasabilitatea operațiunilor prin mecanismele de audit ale platformei.	
FR 13.05	Sistemul va diviza datele din contractele de vânzare-cumpărare și contractele de locațiune, arendă, superficie.
Implementare propusă:	
Va fi implementat un mecanism de divizare explicită a datelor contractuale în două categorii distincte: contracte de vânzare-cumpărare și contracte de locațiune, arendă, superficie. Backend-ul va clasifica și expune datele contractuale pe categorii logice separate, iar interfața ReactJS va prezenta aceste informații în secțiuni distincte, cu separare clară între cele două tipuri de contracte. Soluția va păstra această divizare atât la nivel de model intern și servicii API, cât și la	

nivel de afișare, filtrare și raportare, asigurând coerența dintre sursele de date, câmpurile specifice fiecărei categorii și regulile de audit ale platformei.	
FR 13.06	Sistemul va afisa o lista cronologica a numerelor cadastrale si datelor aferente acestora
Implementare propusă:	
Va fi implementat printr-un mecanism de afișare cronologică a informațiilor contractuale pe număr cadastral, bazat pe un read model dedicat expus de backend. Soluția va agrega contractele și datele aferente fiecărui număr cadastral, le va ordona temporal și le va pune la dispoziția interfeței ReactJS într-o formă pregătită pentru afișare tabelară sau tip timeline. Sistemul va permite utilizatorului să urmărească succesiunea contractelor asociate aceluiași număr cadastral, cu legături către documentele relevante, va ține cont de scenariile cu mai multe coduri cadastrale pe același contract și va asigura performanță și trasabilitate prin indexare, paginare și audit al accesărilor.	
FR 13.07	Sistemul va permite operatorului aparat central adăugarea numerelor cadastrale și datelor aferente acestora, în cazul depistării contractelor care au indicate în același contract mai multe numere cadastrale.
Implementare propusă:	
Va fi implementat un mecanism controlat de asociere multiplă între contract și numere cadastrale, care va permite Operatorului aparatului central să adauge, pentru același contract, numere cadastrale suplimentare și datele aferente acestora. Backend-ul va modela explicit relația dintre contract și bunurile imobile asociate, va valida formatul și unicitatea numărului cadastral în contextul contractului și va persista separat fiecare înregistrare cadastrală, împreună cu metadatele operației. Interfața ReactJS va pune la dispoziție un formular dedicat pentru adăugarea numerelor cadastrale suplimentare, iar noile înregistrări vor fi integrate în listele cronologice, în fluxurile de corecție și în istoricul modificărilor, cu audit complet în mecanismele platformei.	
FR 13.08	Sistemul va permite corectarea datelor înregistrate pentru numerele cadastrale
Implementare propusă:	
Va fi implementat un mecanism de corecții controlate asupra datelor contractuale aferente unui număr cadastral, disponibil exclusiv utilizatorilor autorizați și aplicat asupra unui set limitat de câmpuri configurabile, precum suma, valuta și alte câmpuri selectate. Backend-ul va valida server-side toate corecțiile, va păstra versiunea anterioară a datelor, va înregistra istoricul complet al modificărilor și va jurnaliza operațiile în MLog. Interfața ReactJS va pune la dispoziție un editor controlat cu validări inline și previzualizare a modificărilor, asigurând trasabilitatea integrală a remediilor aplicate asupra datelor contractuale.	
FR 13.09	Sistemul va permite corectarea sumei contractului și valutei
Implementare propusă:	
Va fi implementat un mecanism de corectare controlată a sumei contractului și a valutei, disponibil exclusiv utilizatorilor autorizați și aplicat la nivelul fiecărui număr cadastral. Backend-ul va valida server-side noile valori financiare, va păstra versiunea anterioară a datelor, va înregistra istoricul complet al modificărilor și va jurnaliza fiecare corecție în mecanismele de audit ale platformei. Interfața ReactJS va pune la dispoziție un editor controlat pentru aceste câmpuri, cu validări inline, selecție asistată a valutei și previzualizare a modificărilor înainte de salvare, asigurând trasabilitatea integrală a ajustărilor aplicate datelor contractuale.	

FR 13.10	Sistemul va permite vizualizarea contractului de vânzare-cumpărare pentru fiecare număr cadastral în parte
Implementare propusă:	
Va fi implementat un mecanism de vizualizare securizată a documentului contractual asociat fiecărui număr cadastral, disponibil direct din modulul contractual al SI RPBI. Backend-ul va rezolva documentul aferent contractului, va valida drepturile utilizatorului și va furniza acces controlat la contractul de vânzare-cumpărare sau la documentul de locațiune/arendă/superficie, în funcție de categoria înregistrării selectate. Interfața ReactJS va afișa acțiunea de vizualizare în tabelele și ecranele de detaliu ale contractelor, iar soluția va asigura trasabilitatea accesărilor prin jurnalizare completă în MLog.	
FR 13.11	Sistemul va permite generarea rapoartelor statistice
Implementare propusă:	
Va fi implementat un serviciu de raportare dedicat modulului de evidență a plăților contractuale, care va permite generarea de rapoarte predefinite și parametrizabile asupra datelor contractuale din SI RPBI. Serviciul va expune catalogul de rapoarte disponibile, va valida parametrii de filtrare și va executa agregările statistice necesare pe baza modelului intern consolidat al datelor contractuale, iar interfața ReactJS va permite selectarea raportului, introducerea criteriilor precum perioada, teritoriul, tipul bunului, valuta și statutul și previzualizarea rezultatelor. Soluția va respecta regulile de acces pe roluri, va utiliza structuri optimizate pentru performanță și va asigura trasabilitatea execuției rapoartelor prin mecanismele de audit ale platformei.	
FR 13.12	Sistemul va salva istoricul modificarilor efectuate pentru fiecare număr cadastral în parte
Implementare propusă:	
Va fi implementat un mecanism de păstrare a istoricului complet al modificărilor pentru fiecare număr cadastral, bazat pe versionarea și salvarea separată a tuturor corecțiilor aplicate datelor contractuale. Backend-ul va înregistra pentru fiecare modificare valorile anterioare și valorile noi, utilizatorul care a efectuat schimbarea, momentul operației și tipul modificării, păstrând astfel trasabilitatea integrală a evoluției datelor asociate fiecărui număr cadastral. Interfața ReactJS va expune acest istoric sub forma unui timeline sau a unei liste cronologice în detaliul contractului, iar toate modificările vor fi corelate cu mecanismele generale de audit și jurnalizare ale platformei.	
FR 13.13	Sistemul va oferi posibilitatea de filtrare a informației apărute după: - număr cadastral; - perioadă; - modul de folosintă a bunului; - tipul bunului; - valuta; - adresă; - statut.
Implementare propusă:	
Va fi implementat un mecanism de filtrare avansată a datelor contractuale, executat server-side în backend și expus în interfața ReactJS prin filtre rapide și criterii definite. Soluția va permite restrângerea rezultatelor după număr cadastral, perioadă, mod de folosință, tip de bun, valută, adresă și statut, va valida parametrii de filtrare, va utiliza paginare și indexare pentru performanță și va integra comportamente explicite pentru scenariile fără rezultate, inclusiv	

resetarea sau relaxarea criteriilor. Filtrarea va funcționa unitar pentru cele două categorii de contracte și va respecta regulile de acces și audit ale platformei.	
FR 13.14	Pentru fiecare număr cadastral sistemul va afișa statutul acestuia care poate fi: - Nou (este atrubuit implicit de sistem pentru acele numere cadastrale care nu au fost prelucrate de către operatorul SCT sau operatorul aparatului central); - Salvat (au fost efectuate ajustari dar nu au fost finalizate); - Acceptat (datele au fost verificate și corespund sau datele au fost ajustate pentru a corespunde); - Incident (sunt depistate contracte cu anomalii); - Anomalie (datele contractului nu corespund realității).
Implementare propusă:	
Va fi implementat un mecanism de clasificare a fiecărui număr cadastral, bazat pe un set determinist de stări operaționale: Nou, Salvat, Acceptat, Incident și Anomalie. Backend-ul va persista și actualiza statutul fiecărei înregistrări cadastrale pe baza regulilor de business ale modulului contractual, iar interfața ReactJS va afișa vizual această stare prin badge-uri și indicatori dedicați în tabelele și ecranele de detaliu. Soluția va permite și filtrarea după statut, va păstra coerența între operațiile de corecție, validare și incident management și va jurnaliza toate schimbările de stare în mecanismele de audit ale platformei.	
FR 13.15	Sistemul va trebuie sa calculeze extremele conform formulelor stabilite de către AGCC.
Implementare propusă:	
Va fi implementat un mecanism dedicat de calcul al extremelor asupra datelor contractuale, bazat pe formule configurabile stabilite de AGCC. Backend-ul va utiliza un serviciu specializat pentru evaluarea seturilor de date relevante, calcularea pragurilor statistice sau a intervalelor acceptabile și marcarea automată a valorilor extreme la nivel de contract și număr cadastral. Soluția va păstra regulile într-un model configurabil și versionabil, va salva rezultatele într-o structură dedicată de flag-uri și va expune aceste marcări în interfața ReactJS, în rapoartele statistice și în mecanismele ulterioare de analiză teritorială și GIS.	
FR 13.16	Sistemul va amplasa pe hartă informațiile referitoare la prețurile și valorile bunurilor imobile, oferind o reprezentare detaliată a zonelor valorice și a tendințelor pieței imobiliare.
Implementare propusă:	
Va fi implementat un serviciu GIS dedicat reprezentării tematice a prețurilor și valorilor bunurilor imobile, integrat în SI RPBI și bazat pe PostgreSQL+PostGIS, GeoServer și OpenLayers. Serviciul va corela datele contractuale și valorice cu geometriile cadastrale și teritoriale, va genera layer-e GIS distincte pentru prețuri și valori și va susține agregări spațiale, zone valorice și indicatori de tendință ai pieței imobiliare. Interfața ReactJS va permite navigarea interactivă pe hartă, filtrarea spațială, clusterizarea, afișarea de tooltip-uri cu detalii și vizualizarea pe Unitate Administrativ-Teritorială sau număr cadastral, iar soluția va respecta cerințele de performanță, audit și control al accesului definite pentru platformă.	
FR 13.17	Sistemul va afișa pe hartă informațiile referitoare la plățile pentru locațiune, arendă și redevențele pentru constituirea superficiilor.
Implementare propusă:	
Va fi implementat un serviciu GIS dedicat reprezentării tematice a plăților pentru locațiune, arendă și redevențelor pentru constituirea superficiilor, integrat în SI RPBI și bazat pe	

PostgreSQL/PostGIS, GeoServer și OpenLayers. Serviciul va corela datele contractuale din categoria LEASE/ARREND/SUPERFICIE cu geometriile cadastrale și teritoriale, va calcula niveluri medii și distribuții spațiale și va publica un layer GIS distinct pentru plăți contractuale. Interfața ReactJS va permite vizualizarea interactivă pe hartă, filtrarea spațială, afișarea de legende și tooltip-uri de detaliu și navigarea pe Unitate Administrativ-Teritorială sau număr cadastral, cu respectarea regulilor de acces, performanță și audit ale platformei.

7. Abordarea și implementarea cerințelor nefuncționale

Vom implementa cerințele nefuncționale ale SI RPBI ca un set coerent de măsuri tehnice, arhitecturale și operaționale, aplicate uniform pe toate componentele soluției: interfața publică, back-office-ul operațional, serviciile de integrare, motorul de procesare, componenta GIS, stocarea documentelor și stratul de date. Abordarea propusă urmărește respectarea integrală a cerințelor obligatorii din Caietul de sarcini, în special celor referitoare la modularitate, interoperabilitate, performanță, securitate, accesibilitate și operare în MCloud.

7.1. Cerințe privind licențierea și proprietatea intelectuală (LIPR 001 – LIPR 008)

Vom asigura cerințele de licențiere și proprietate intelectuală prin utilizarea preponderentă a unei stive bazate pe componente open-source, mature și larg adoptate, ceea ce reduce dependența de produse COTS cu licențiere restrictivă și elimină riscul unor limitări pe utilizatori, tranzacții sau accesări simultane. În implementarea SI RPBI propunem ReactJS, Java 25 + Spring Boot 4.0.3, PostgreSQL, PostgreSQL + PostGIS, Elasticsearch, Redis, Kafka, MinIO, GeoServer, OpenTelemetry, Prometheus, Grafana și ELK, toate integrate într-o arhitectură containerizată în MCloud. În consecință, modelul nostru răspunde direct cerințelor LIPR privind furnizarea licențelor necesare fără costuri suplimentare, lipsa restricțiilor de scalare pentru utilizatori/documente/tranzacții și accesibilitatea API-urilor pentru sisteme externe autorizate.

Vom transfera către AGCC întregul cod sursă dezvoltat pentru SI RPBI, împreună cu configurațiile, personalizările, scripturile de build și deployment, definițiile IaC, schemele de migrare a bazei de date, definițiile API și documentația tehnică aferentă. Practic, nu vom livra doar binare sau imagini container, ci întregul pachet de active tehnice necesare pentru operare, mentenanță și dezvoltare ulterioară. Acest lucru răspunde explicit cerințelor LIPR privind transferul drepturilor asupra dezvoltărilor și asupra codului sursă, precum și faptului că datele și structurile informaționale ale SI RPBI rămân proprietatea AGCC.

7.2. Cerințe privind arhitectura sistemului IT (ARH 001 – ARH 029)

Vom implementa SI RPBI ca o platformă modulară, orientată pe servicii, rulată în containere Docker și orchestrată prin Kubernetes în MCloud. În oferta tehnică, soluția este împărțită în servicii specializate pentru acces și identitate, utilizatori, nomenclatoare, procese de business, notificări, căutare, fișiere, raportare, GIS, integrare guvernamentală și audit, plus componente dedicate de persistență, căutare, cache, event bus și monitorizare. Această separare ne permite să scalăm independent componentele încărcate, să izolăm defectele, să livrăm incrementale și să menținem sistemul fără a afecta transversal toate funcțiile.

Arhitectura este susținută și de infrastructura propusă: cluster Kubernetes cu 6 noduri, server dedicat PostgreSQL/PostGIS cu 12 CPU, 64 GB RAM și SSD de 1 TB, plus stocare dedicată pentru MinIO. Asta înseamnă că nu bazăm cerințele nefuncționale doar pe design teoretic, ci și pe o topologie tehnică explicită, dimensionată pentru separarea rolurilor infrastructurale și pentru susținerea încărcării operaționale. În plus, PostgreSQL gestionează tranzacțiile operaționale, PostGIS gestionează sarcinile geospațiale, MinIO stocarea documentelor, iar Elasticsearch și Kafka decuplează și accelerează funcțiile de căutare și procesare.

7.3. Cerințe privind stiva tehnologică a sistemului IT (TS 001 – TS 025)

Vom asigura cerințele privind stiva tehnologică prin utilizarea unei platforme moderne, coerente și specializate pe roluri. ReactJS 19.2.0, împreună cu Ant Design, React Router și TanStack Query, va fi utilizat pentru interfața SPA și pentru un UI rapid, modular și responsive. Java 25 + Spring Boot 4.0.3 vor asigura serviciile backend, API-urile REST/JSON, validările de business, securitatea, integrarea și orchestration-ul proceselor. PostgreSQL va fi sursa de adevăr pentru datele relaționale, Redis va accelera accesul la date frecvente, Elasticsearch va deservi căutarea full-text și filtrările rapide, iar MinIO va separa stocarea de fișiere de baza de date operațională. Pentru componenta GIS vom utiliza PostGIS, GeoServer și OpenLayers, ceea ce oferă suport nativ pentru interogări spațiale, heatmap, clustering, WMS/WMTS/WFS și publicarea straturilor geografice.

Această alegere de stivă nu este doar compatibilă cu TOR, ci și justificată tehnic prin specializarea componentelor: full-text search nu este forțat în RDBMS, GIS nu este simulat în frontend, iar fișierele nu sunt stocate ca blob-uri în baza operațională. Fiecare clasă de cerințe este susținută de o tehnologie potrivită: căutare prin Elasticsearch, date geospațiale prin PostGIS/GeoServer, procesare asincronă prin Kafka, cache prin Redis, și observabilitate prin OpenTelemetry + Prometheus/Grafana

+ ELK. Astfel, mentenabilitatea, scalabilitatea și performanța sunt obținute prin design și alegerea corectă a componentelor, nu doar prin optimizări ulterioare.

7.4. Cerințe privind interoperabilitatea sistemului IT (INT 001 – INT 008)

Vom asigura interoperabilitatea prin adaptoare dedicate dezvoltate în Java 25 + Spring Boot 4.0.3 pentru MPass, MSign, MNotify, MConnect, MLog și, după caz, MPay, precum și pentru integrarea cu IP CBI, e-Notar și SFS prin MConnect. Acest model ne permite să izolăm tehnic integrarea de logica de business, să versionăm contractele de interfață, să tratăm separat erorile de interoperabilitate și să menținem stabilitatea nucleului aplicației când se schimbă un sistem extern. În plus, API-urile interne și externe vor fi documentate prin OpenAPI/Swagger, ceea ce susține atât integrarea, cât și testarea și acceptanța tehnică.

Pentru a face interoperabilitatea robustă, nu ne vom baza pe apeluri sincrone brute între sisteme, ci pe un model cu timeouts, retry controlat, jurnalizare, corelare și, unde este necesar, procesare asincronă prin Kafka și outbox. Astfel, un incident temporar într-un sistem extern nu va compromite integritatea internă a SI RPBI. Pentru componenta GIS, interoperabilitatea este susținută suplimentar prin GeoServer și servicii standard WMS/WMTS/WFS, iar în browser prin OpenLayers, ceea ce permite atât consum, cât și publicare de date geospațiale în formate și protocoale standard.

7.5. Cerințe privind performanța și scalabilitatea sistemului IT (PSR 001 – PSR 012)

Vom asigura țintele de performanță și scalabilitate printr-o combinație explicită de arhitectură, tehnologii specializate și dimensionare de infrastructură. Pentru cerințele de procesare asincronă, suport pentru 1000 de utilizatori concurenți și timp de răspuns sub 2 secunde pentru 95% din tranzacții, oferta ta deja propune exact mecanismele potrivite: separarea operațiilor sincrone de cele asincrone, scalare orizontală în Kubernetes, utilizarea Redis pentru cache și validarea prin teste de sarcină JMeter/k6.

Concret, pentru operațiile interactive vom utiliza ReactJS + TanStack Query în frontend pentru reducerea apelurilor redundante, încărcare incrementală și reutilizarea controlată a datelor deja citite. În backend, Spring Boot va expune API-uri specializate, cu payload-uri limitate la datele necesare ecranului curent, iar în PostgreSQL vom proiecta explicit indexuri pentru filtre, sortări și căutări recurente. Paginarea, filtrarea la sursă și evitarea dataset-urilor masive în UI vor fi reguli de

proiectare obligatorii. Pentru nomenclatoare, rezultate frecvent accesate și date relativ stabile vom utiliza Redis, astfel încât solicitările repetitive să nu lovească inutil baza operațională.

Pentru căutările full-text și căutările cu mai multe criterii nu vom folosi interogări de tip LIKE direct în PostgreSQL, ci Elasticsearch, care este inclus explicit în stiva propusă. Asta este important pentru SI RPBI, unde există filtrări, sugestii, fuzzy search, highlight și căutări rapide pe volume mari. Prin mutarea acestei clase de sarcini în Elasticsearch, eliberăm baza operațională de interogări costisitoare și menținem timpul de răspuns al fluxurilor tranzacționale.

Pentru operațiile costisitoare, cum ar fi generarea de rapoarte, exporturi, recalculări, sincronizări și alte procese I/O intensive, vom utiliza Kafka pentru procesare asincronă și decuplare între producătorii și consumatorii de evenimente. Cu alte cuvinte, nu vom bloca request-ul utilizatorului până la finalizarea unei operații grele; vom crea joburi și fluxuri de procesare independente, astfel încât interfața să rămână rapidă și predictibilă chiar și sub încărcare. Acesta este unul dintre mecanismele principale prin care asigurăm simultan timp bun de răspuns pentru UI și capacitate de procesare pentru sarcini grele.

Pentru componenta geospațială nu vom încărca calcule GIS în backend-ul generic, ci vom utiliza PostgreSQL + PostGIS pentru interogări spațiale și GeoServer pentru publicarea și servirea straturilor. Asta înseamnă că operațiile de heatmap, cluster, selecție spațială și filtrare teritorială vor fi executate în componente specializate, nu simulate ineficient la nivel aplicațional. În acest mod, și performanța GIS rămâne predictibilă, și scalarea se face separat față de nucleul tranzacțional al sistemului.

La nivel de infrastructură, Kubernetes ne permite să scalăm independent frontend-ul, serviciile backend, consumatorii Kafka, Elasticsearch și componentele auxiliare, în funcție de presiunea reală din sistem. În plus, oferta ta propune deja un cluster Kubernetes cu 6 servere, un server dedicat de bază de date puternic și resurse dedicate pentru MinIO, în timp ce resursele pentru Elasticsearch, Kafka și Redis vor fi dimensionate în etapa tehnică după volumele reale estimate. Cu alte cuvinte, cerințele de performanță nu sunt tratate declarativ, ci sunt susținute printr-o topologie clară de rulare și prin separarea rolurilor între componente.

În final, performanța nu va fi doar „presupusă”, ci demonstrată. Vom valida explicit scenariile de încărcare și concurență prin teste JMeter și/sau k6, exact pe fluxurile critice ale SI RPBI, și vom monitoriza în exploatare latențele, ratele de eroare, consumul de resurse și comportamentul la vârf prin OpenTelemetry,

Prometheus și Grafana. Astfel, țintele NFR vor fi susținute de măsurare continuă, nu doar de argumentare arhitecturală.

7.6. Cerințe privind interfața cu utilizatorul și ergonomia sistemului IT (UI 001 - UI 030)

Vom asigura cerințele UI printr-o aplicație SPA în ReactJS 19.2.0, cu Ant Design pentru consistență vizuală și funcțională, React Router pentru navigare coerentă și TanStack Query pentru interacțiune rapidă cu backend-ul. Această combinație ne permite să construim o interfață modulară, responsive și rapidă, fără reload complet de pagină, cu validări în timp real, feedback contextual și o experiență uniformă între ecranele operaționale și portalul public. Pentru componenta GIS, OpenLayers va permite hartă interactivă, straturi, heatmap, cluster și interacțiuni spațiale direct în browser.

Pentru a răspunde cerințelor de accesibilitate și ergonomie, vom implementa un design system unitar, cu pattern-uri coerente, contrast, focus states, navigare clară și compatibilitate cu WCAG 2.1 AA. Nu vom trata accesibilitatea ca strat cosmetic de final, ci ca regulă de proiectare a componentelor încă din prototipare și din implementarea frontend-ului. În plus, prin separarea clară dintre backend, GeoServer și frontend, fiecare ecran va primi exact datele și serviciile de care are nevoie, ceea ce îmbunătățește nu doar performanța, ci și claritatea utilizării.

7.7. Cerințe privind securitatea și protecția sistemului IT (SEC 001 – SEC 072)

Vom asigura securitatea SI RPBI printr-o combinație de controale la nivel de aplicație, infrastructură, supply chain și operare. În aplicație, accesul va fi controlat prin MPass și Spring Security, cu autorizare RBAC granulară pe roluri și acțiuni. Toate API-urile vor fi expuse prin HTTPS/TLS, iar datele sensibile vor fi protejate atât în tranzit, cât și la rest. Evenimentele critice vor fi jurnalizate și publicate în MLog, iar logurile tehnice vor fi centralizate în ELK. În plus, observabilitatea completă va fi asigurată prin OpenTelemetry, Prometheus și Grafana, astfel încât anomaliile, erorile și degradările să fie detectate rapid.

Pentru protecția datelor și continuitate, oferta ta include CloudNativePG pentru backup și recuperare la nivel de bază de date și Velero File System Backup pentru fișiere, ceea ce oferă o bază tehnică reală pentru RPO/RTO și DR. Documentele și fișierele vor fi stocate în MinIO, separat de baza operațională, ceea ce simplifică retenția, restaurarea și controlul accesului. În plus, prin rularea în Kubernetes și prin CI/CD automatizat cu GitLab CI/CD sau similar, ArgoCD și

Terraform/Ansible, configurarea și deployment-ul devin repetabile, auditate și mai puțin vulnerabile la erori umane.

Vom aplica și măsuri de securitate în lanțul de livrare: scanări automate în pipeline, verificări asupra dependențelor, control al artefactelor și politici de release. Acest model este deja aliniat atât cu cerințele din Caiet privind OWASP, supply-chain control, jurnalizare și managementul vulnerabilităților, cât și cu secțiunea ta de CI/CD și QA. Astfel, securitatea nu este doar un set de reguli de infrastructură, ci o proprietate verificată continuu pe tot ciclul de viață al sistemului.

8. Implementare

Sistemul de producție al SI RPBI va fi implementat și găzduit în cadrul platformei guvernamentale comune MCloud, utilizând containere Docker orchestrate prin Kubernetes, într-o arhitectură bazată pe microservicii, elastică, scalabilă și pregătită pentru disponibilitate ridicată. Soluția nu include furnizarea de componente hardware, iar dimensionarea finală a resurselor de infrastructură va fi confirmată în etapa de proiectare tehnică și coordonată cu mediul operațional administrat de STISC/MCloud.

Pentru dezvoltare, testare, demonstrare, validare intermediară și acceptanță, vor fi utilizate medii separate non-producție, în care vor fi livrate iterativ versiuni demo, versiuni intermediare, versiuni candidate pentru acceptanță și versiuni corective, conform planului de testare și procesului de livrare agreat în proiect. Toate implementările vor fi realizate prin pipeline-uri automate de CI/CD și printr-o abordare infrastructure-as-code, cu suport pentru build automat al imaginilor container, versionare controlată a configurațiilor și livrare repetabilă între medii.

Soluția SI RPBI va utiliza o platformă tehnologică compusă din servicii containerizate pentru frontend, backend, baze de date relaționale și geospațiale, căutare full-text, integrare guvernamentală, stocare documente, procesare asincronă și componente GIS, toate operate într-un model unitar și securizat în cadrul clusterului Kubernetes. Componente precum serviciile aplicaționale Java/Spring Boot, interfața ReactJS, GeoServer, Elasticsearch, Redis, Kafka, adaptoarele pentru MPass, MSign, MNotify, MConnect și serviciile auxiliare vor fi livrate și administrate în regim containerizat, cu monitorizare, jurnalizare centralizată și politici de backup/restore.

Pentru funcționarea soluției sunt necesare următoarele cerințe tehnice de infrastructură:

- Cluster Kubernetes: 6 servere (trei master și trei worker). Configurație master: CPU 6 nuclee, RAM 16 GB, HDD 100 GB. Configurație worker: CPU 8 nuclee, RAM 32 GB, HDD 150 GB.
- Server bază de date PostgreSQL/PostGIS: CPU 12 nuclee, RAM 64 GB, HDD 1 TB SSD.
- Server de stocare obiecte / documente (MinIO) pentru documente generate, atașamente și exporturi: CPU 2 nuclee, RAM 8 GB, HDD 1 TB.
- Spațiu de stocare și resurse pentru componentele de căutare și procesare asincronă (Elasticsearch, Kafka, Redis) vor fi alocate în cadrul clusterului Kubernetes, cu dimensionare finală stabilită în etapa de proiectare tehnică, în funcție de volumele estimate de date, încărcarea operațională și cerințele de performanță.
- Serviciile GIS vor utiliza baza de date geospațială PostgreSQL/PostGIS și GeoServer, publicate și operate în cadrul platformei containerizate, cu suport pentru layer-e tematice, agregări spațiale, filtrare geografică și integrare cu interfața web bazată pe OpenLayers.

9. Cerințe pentru instruirea utilizatorilor sistemului

Ca parte a implementării sistemului, vor fi elaborate și furnizate materiale de instruire. Instruirea utilizatorilor-cheie va fi desfășurată pe baza materialelor dezvoltate. Procesul de instruire va acoperi subiectele necesare și suficiente pentru ca utilizatorii și administratorii să își poată îndeplini funcțiile.

Va fi elaborat și agreat cu Beneficiarul un Plan de Instruire. Planul de Instruire va include toate acțiunile necesare, programele, scenariile de instruire și alte cerințe.

10. Sisteme de testare automată

Toate cerințele pentru sistemele de testare sunt asigurate de procesul nostru de dezvoltare; vă rugăm să consultați descrierea detaliată din capitolul 3.5 Tehnici și abordări pentru asigurarea calității

11. Procedura de control și recepție a sistemului

Echipa noastră va prelua coordonarea organizării recepției sistemului. Va fi constituită o comisie pentru recepția punerii în funcțiune a sistemului, care va include reprezentanți ai Clientului și ai Contractantului.

Software-ul va fi furnizat împreună cu codul sursă, iar Clientul/Beneficiarul va avea dreptul de a modifica sistemul și de a angaja alți furnizori pentru a-l modifica. Cu toate acestea, în perioada de garanție se va acorda o atenție deosebită, iar dacă alți furnizori vor modifica software-ul, se presupune că aceștia vor deține calificările și competențele necesare.

Testele de acceptanță vor fi efectuate pe baza unui protocol care va fi agreat cu Beneficiarul.

Următoarea listă de documente va fi furnizată la punerea în funcțiune a sistemului:

- Ghiduri de utilizare;
- Ghidul administratorului de sistem;
- Ghid de instalare și configurare a sistemului;
- Document de proiectare detaliată a software-ului (SDDD);
- Documentația API-urilor implementate;
- Rapoarte de testare.

12. Concluzie

Echipa noastră este pregătită să preia această misiune și garantează livrarea unui produs de înaltă calitate, care va ajuta Beneficiarul să își automatizeze și modernizeze procesele.

Pentru orice întrebări sau clarificări privind detaliile abordării noastre tehnice pentru acest proiect, vă rugăm să nu ezitați să ne contactați.

Semnat: _____

Andrei Cojocari
Director Comercial
S&T MOLD SRL
Republica Moldova, municipiul Chișinău, str. Calea Ieșilor 8, MD-2069