Propunere tehnică

Dezvoltarea și implementarea Sistemului Informațional Automatizat “Managementul deşeurilor”
Mtender ID NR. ocds-b3wdp1-MD-1770204825070

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

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, MPower, MConnect și a realizat integrări cu majoritatea serviciilor MCloud.

2.2. Design de nivel înalt

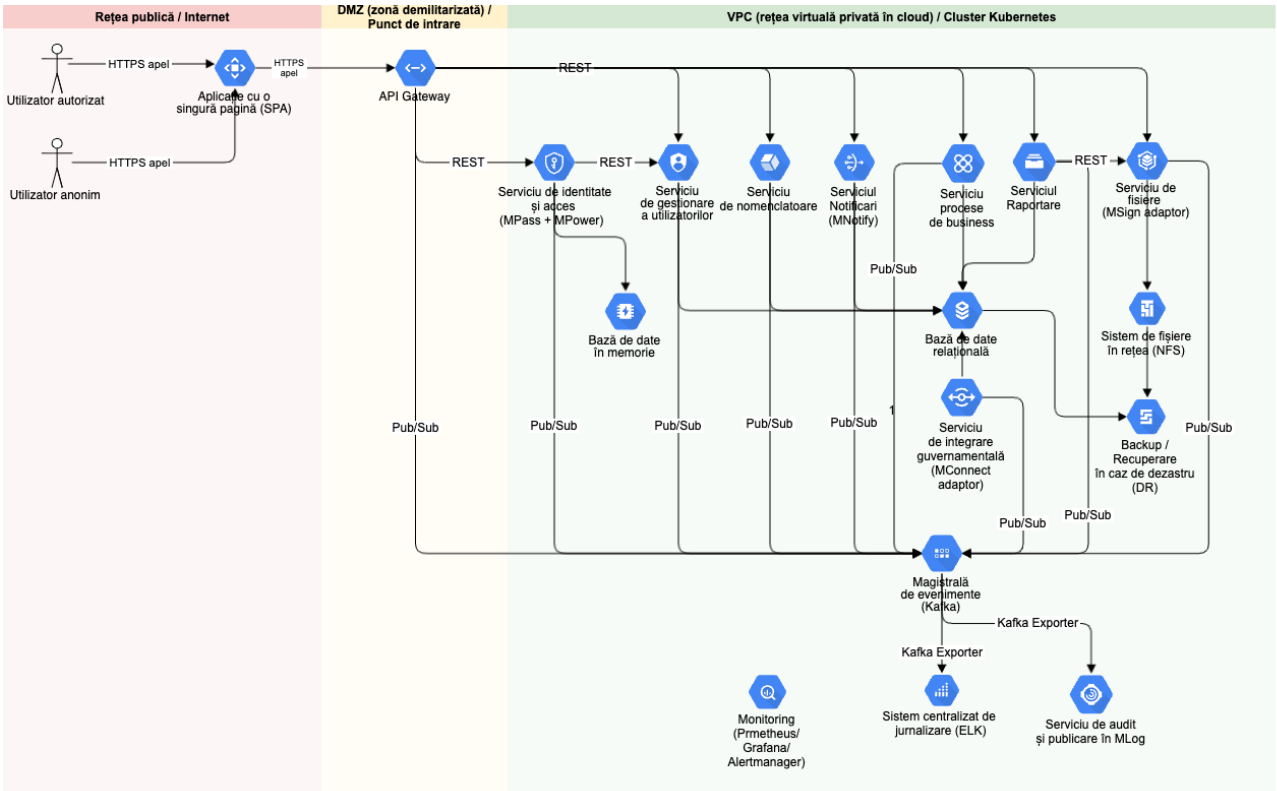


Diagrama de mai sus prezintă o imagine de ansamblu, la nivel înalt, a arhitecturii soluției propuse și modul în care containerele comunică între ele. Fiecare componentă

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 + MPower)	Serviciu responsabil de autentificarea utilizatorilor prin MPass și de verificarea drepturilor/împuternicirilor prin MPower, 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	Serviciu responsabil de orchestrarea și executarea proceselor de business ale sistemului, inclusiv aplicarea	Java 25, Spring Boot 4.0.3

	regulilor operaționale, validarea fluxurilor și coordonarea interacțiunii dintre serviciile funcționale și sursele de date.	
Serviciul Raportare	Serviciu responsabil de colectarea, validarea, procesarea și gestionarea rapoartelor din sistem (inclusiv generarea de rapoarte/ieșiri și exporturi), în conformitate cu cerințele și fluxurile de raportare aplicabile.	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
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	Bază de date relațională utilizată pentru stocarea persistentă și gestionarea structurată a datelor operaționale, a tranzacțiilor și a informațiilor de business din sistem.	PostgreSQL
Sistem de fișiere în rețea (NFS)	Stocarea fișierelor încărcate / generate	Cloud Storage
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 serviciilor și integrarea cu mecanismele de jurnalizare centralizată.	Kafka
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.

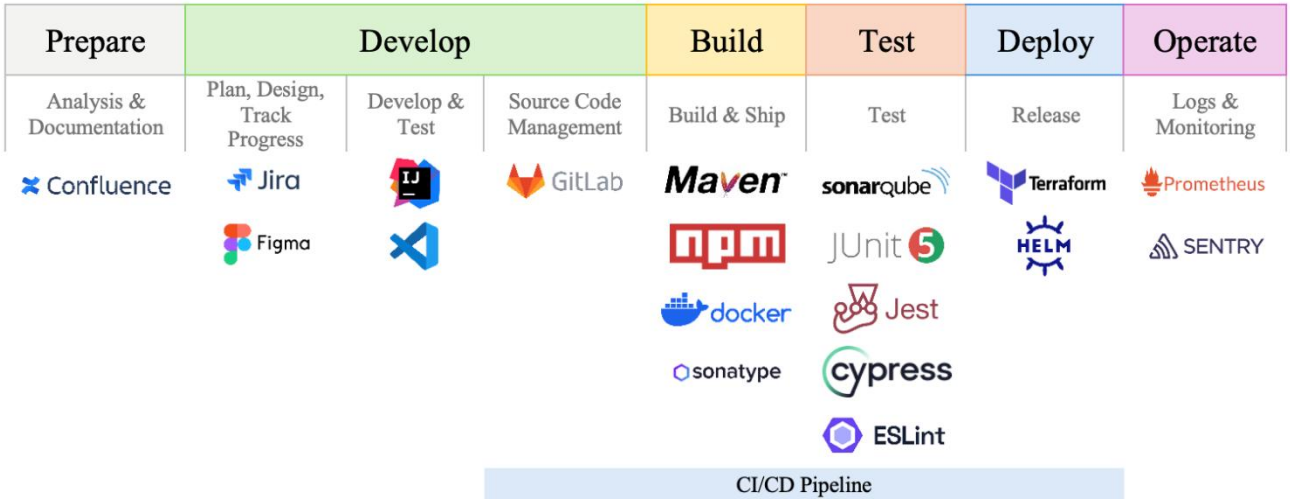
7. Folosim instrumente de prototipare foarte puternice și proiectăm interfețele împreună cu clienții noștri, trecând la implementare numai după ce clientul aprobă prototipurile.

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. 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.



3.5. 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.

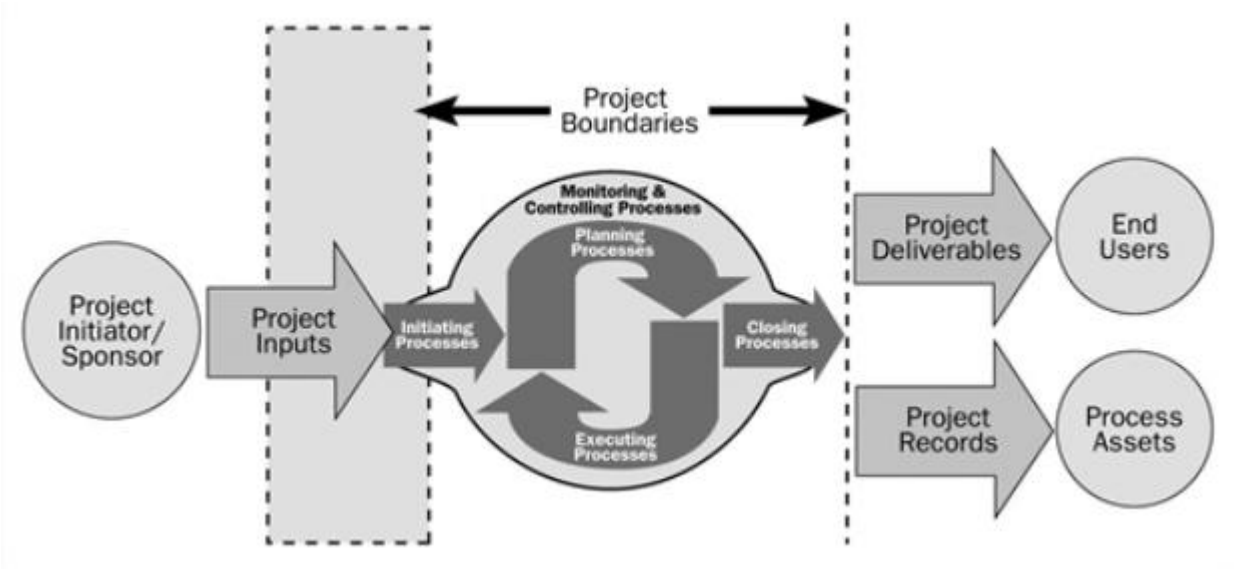
Testarea automată permite executarea rapidă, în paralel, a mai multor scenarii standardizate cu seturi de date variate.

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.

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.

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.

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.

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.

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 și urgența dezvoltării unui Sistem Informațional modern, fiabil și sustenabil – **Sistemul Informațional Automatizat „Managementul Deșeurilor” (SIA MD)**. Soluția finală pe care o vom livra se bazează pe o arhitectură robustă, pregătită pentru cloud, care asigură scalabilitate, performanță, securitate și adaptabilitate la nevoile în continuă evoluție ale proceselor de evidență, raportare și monitorizare a fluxurilor de deșeuri, precum și a mecanismelor aferente de conformare și control.

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

- **O arhitectură pe microservicii** scalabilă și modulară, care asigură disponibilitate ridicată, mentenabilitate și securitatea datelor, implementată și operată în infrastructură containerizată (Kubernetes)
- **Gestionarea utilizatorilor, rolurilor și accesului**, inclusiv autentificare și autorizare prin servicii guvernamentale (MPass) și verificarea împuternicirilor (MPower)
- **Gestionarea nomenclatoarelor și datelor de referință**, necesare clasificării și operării fluxurilor de raportare și evidență
- **Procesarea și orchestrarea proceselor de business** aferente sistemului, inclusiv fluxuri de depunere, verificare, corectare și aprobare a datelor/rapoartelor
- **Raportarea și evidența fluxurilor de deșeuri** și generarea rapoartelor/ieșirilor necesare, inclusiv exporturi și rapoarte statistice/analitice
- **Interoperabilitate și schimb de date** cu platforme și sisteme externe relevante prin **MConnect**, inclusiv integrarea cu registre și sisteme guvernamentale
- **Interoperabilitate cu serviciile guvernamentale electronice** precum **MPass, MSign, MLog și MNotify**, inclusiv semnarea electronică a documentelor și transmiterea notificărilor
- **API-uri REST securizate** pentru integrarea cu sisteme externe și consumatori autorizați, precum și pentru integrarea cu aplicații web și mobile
- **Audit și trasabilitate** prin colectarea evenimentelor de business și publicarea acestora în **MLog**, cu procesare asincronă și mecanisme de retry pentru reziliență
- **Stocare sigură a documentelor și fișierelor** (atașamente, documente semnate), inclusiv integrare cu adaptor **MSign** și păstrare pe infrastructură de tip NFS, conform politicilor de retenție

În plus, întregul sistem va fi dezvoltat în conformitate cu recomandările de securitate **OWASP Top 10**, asigurând reziliență împotriva vulnerabilităților comune și a amenințărilor cibernetice, precum și cu măsuri de monitorizare proactivă, jurnalizare centralizată și planuri de backup/recuperare în caz de dezastru, necesare operării în regim 24/7.

6. Abordarea și implementarea cerințelor funcționale

Stack propus:

- **Frontend:** ReactJS (JavaScript), **Ant Design** (Layout/Menu/Table/Form/Dropdown/Tabs/Collapse), React Router, Fetch, React Query (cache + retry), dayjs.

- **Backend:** Java 25 + Spring Boot 4.0.3, Spring Web (REST), Spring Security, PostgreSQL, Flyway, OpenAPI/Swagger.
- **Export:** XLSX (Apache POI), CSV (stream), PDF (OpenPDF).
- **Integrare GEAP:** adaptor MConnect + serviciu de sincronizare (job) + cache/DB local pentru afișare/filtrare/export.

6.1. Interfața publică

Cerință	Implementare propusă
FRQ001	**FE:** rută publică /public/* în React Router, fără provider de autentificare. **BE:** Spring Security permitAll() pentru /api/public/** și resurse statice. Cache HTTP (ETag/Cache-Control) pentru pagini publice.
FRQ002	**BE:** API public **read-only** (doar GET). Orice endpoint de scriere rămâne în /api/private/** și necesită token/rol; validări server-side + rate-limit (ex. bucket4j) pe endpoint-uri publice ca să prevenim abuz. **FE:** nu renderăm butoane de submit/încărcare în zona publică.
FRQ003	**FE:** meniu AntD Menu + Layout, rute dedicate per secțiune (Home/About/Legal/Lists/Costs/Help). Config de meniu din backend (feature flag) ca să putem activa/dezactiva secțiuni fără redeploy.
FRQ004	**FE:** Header sticky + meniu responsive (desktop: orizontal, mobil: Drawer). Navigarea rapidă prin React Router + scroll-to-top on route change.
FRQ005	**FE:** pagină "Landing" construită din componente AntD (Hero, Cards, Quick links). **BE:** conținutul text (descrieri, blocuri) servit dintr-un "mini-CMS" (public_pages), ca să poată fi editat de Administrator.
FRQ006	**FE:** pagină "About" compusă din secțiuni (obiective/actori/fluxuri) + posibilitate de embed pentru diagrame (imagine/PDF). **BE:** conținut versionat în DB (audit cine/când a modificat).
FRQ007	**BE:** tabel legal_acts (type: lege/HG/regulament, titlu, descriere, url, pdf_id). **FE:** listă (AntD Table/List) cu filtrare pe tip + căutare full-text (LIKE/tsvector).
FRQ008	**FE:** click pe act → window.open(url) dacă există link oficial; altfel buton "Descarcă PDF". **BE:** endpoint /api/public/legal/{id}/file stream-uiește PDF (din storage), cu headers corecte + antivirus scan la upload (admin).
FRQ009	**FE:** pagină "Lista Producătorilor" cu AntD Table (paginare/sortare server-side). **BE:** endpoint GET /api/public/producers cu page,size,sort,filters.
FRQ010	**FE:** definim coloane AntD conform mapping-ului; coloana "denumire sistem" este **calculată**: dacă tip=colectiv afișăm numele sistemului colectiv, dacă tip=individual afișăm denumirea companiei (logică în BE ca să fie identică în UI+export).
FRQ011	**FE:** panou "Filtre avansate" (AntD Form + Input/Select/DatePicker). **BE:** construire dinamică query (Spring Data Specification) pe câmpurile filtrabile; index-uri DB pe IDNO/număr înregistrare/data.
FRQ012	**BE:** endpoint GET /api/public/systems care agregă: (1) sisteme individuale derivat din producători cu tip=individual; (2) sisteme colective din registrul cererilor

	aprobate. Opțional: materialized view pentru performanță. **FE:** tabel identic ca UX cu producătorii.
FRQ013	**FE:** coloane + render special: website ca link, email mailto:, telefon cu format. **BE:** DTO dedicat PublicSystemDTO cu normalizare date (ex. website fără "http" → adăugat).
FRQ014	**FE:** filtre pe denumire + tip responsabilitate (Select). **BE:** Specification simplă + collation/case-insensitive.
FRQ015	**FE:** implementăm ca **Tabs** pe categorie REP (sau rute /producers?category=DEEE etc.). **BE:** category devine parametru obligatoriu când e ales tab-ul → query + export pe categorie.
FRQ016	**Integrare:** serviciu GeapMconnectClient care extrage listele de autorizații din SIA GEAP prin MConnect. **Strategie:** stocăm rezultatul în DB local (geap_permits) ca să permitem filtrare rapidă + export consistent; job de sincronizare (cron configurabil) + fallback on-demand dacă lipsesc date. **FE:** click pe rând → deschidem documentul oficial într-un tab nou (link/redirect generat de BE).
FRQ017	**BE:** mapare strictă pe câmpurile cerute în DTO (denumire, IDNO, adresă, activități, telefon, email). **FE:** coloană "Activități" cu tooltip/expand (AntD Popover) dacă textul e lung.
FRQ018	**FE:** filtre (denumire/IDNO/activități) cu debounce. **BE:** index pe IDNO + căutare text (ILIKE/tsvector) pe denumire/activități; limitare + paginare.
FRQ019	**Integrare:** același pattern ca FRQ016, dar pentru notificări export/tranzit (tabel geap_notifications). **FE:** tabel public cu refresh automat (cache invalidation după sync).
FRQ020	**BE:** DTO PublicNotificationDTO cu câmpuri cerute; normalizare tip transfer/tip deșeuri. **FE:** coloană "Tip deșeuri" cu wrap + tooltip.
FRQ021	**FE:** filtre combinate (Select pentru tip transfer + Input pentru rest). **BE:** query dinamic + paginare.
FRQ022	**BE:** nomenclator rep_targets (category, waste_type, target_value, source_doc_ref). Seed inițial din anexele/actele normative; pentru VSU setăm flag not_defined=true și afișăm text "nu sunt stabilite". **FE:** pagină cu Tabs per categorie REP, tabel simplu.
FRQ023	**FE:** filtrare multiplă simultan (Form → payload complet). **BE:** combinare filtre cu AND (implicit), cu suport pentru liste (ex. mai multe categorii) prin IN (...).
FRQ024	**BE:** endpoint comun de export per listă, ex.: `/api/public/producers/export?format=xlsx
FRQ025	**BE:** parametru `mode=full
FRQ026	**FE:** buton AntD Dropdown.Button "Export" deasupra tabelului; opțiuni XLSX/CSV/PDF; indicator Spin/disabled în timpul exportului.
FRQ027	**BE:** (1) pentru date interne: tabelele publice citesc direct din DB (status "validat"); (2) pentru GEAP: job @Scheduled + "last_sync" per tip listă; posibil "near real-time" prin TTL mic + refresh la cerere. **Observabilitate:** log + metrică de succes/eroare la sincronizare.

FRQ028	**FE:** pagină “Cost operațional” cu secțiuni pe categorie (Tabs/Collapse) și câte un AntD Table per secțiune. **BE:** date în operational_costs (category, waste_type, lei_per_ton, year, status).
FRQ029	**BE:** validare că fiecare rând are doar waste_type + lei_per_ton (schema fixă), iar UI afișează exact 2 coloane. Opțional: suport multi-an cu filtrare year.
FRQ030	**BE:** endpoint public GET /api/public/costs (read-only) + export. **FE:** fără editare, doar vizualizare și descărcare.
FRQ031	**Admin UI:** pagină în zona admin (React + AntD EditableTable) pentru editare/adăugare/ștergere. **BE:** endpoint-uri /api/admin/costs (POST/PUT/DELETE) cu validări + audit (cine/când/ce a schimbat).
FRQ032	**Securitate:** zona admin protejată prin autentificare (SSO/MPass conform sistemului) + rol ADMIN. **FE:** route guard (dacă nu ai rol → redirect). **BE:** Spring Security hasRole('ADMIN').
FRQ033	**FE:** pagina Asistență: FAQ ca AntD Collapse + Contact ca Card (date: telefon/email/adresă/program). **BE:** conținut administrabil din DB (faq_items, contact_info) + posibilitate de ordonare și publicare.

6.2. Cabinetul personal

Cerință	Implementare propusă
FRQ034	**Accesul în Cabinetul personal** va fi permis doar după autentificare prin **MPass**. În backend vom integra un modul de autentificare bazat pe token/asserțiune primită de la MSign, iar în Spring Security vom proteja toate rutele /api/private/profile/**. În frontend, după login reușit, utilizatorul va fi redirecționat automat către /cabinet-personal.
FRQ035	La prima logare, utilizatorul va fi creat automat în tabela users cu rol inițial **VISITOR**. În React vom implementa un **guard de onboarding**, astfel încât utilizatorul fără companie activă să fie direcționat către tab-ul **Companii**. În backend vom avea o regulă centrală de autorizare care permite doar acces vizual la alte module până la completarea profilului minim.
FRQ036	Vom implementa un mecanism de **access gate**: dacă utilizatorul nu are nicio companie activă asociată, API-urile funcționale pentru cereri/raportări/notificări vor răspunde cu 403 / business restriction, iar în frontend modulele respective vor apărea dezactivate sau read-only, cu mesaj contextual.
FRQ037	În secțiunea **Companii** vom permite asocierea uneia sau mai multor companii per utilizator. Modelul de date va include tabelele companies și user_company_roles, astfel încât un utilizator să poată avea drepturi pe mai multe entități juridice. În UI, lista companiilor va fi afișată într-un **Ant Design Table** cu acțiuni de adăugare, vizualizare și editare.
FRQ038	Cabinetul personal va fi implementat ca pagină unică cu **Ant Design Tabs** sau meniu lateral, cu 4 secțiuni distincte: **Date personale**, **Companii**, **Delegare persoană**, **Mesagerie**. Fiecare secțiune va fi încărcată lazy pentru performanță, iar starea va fi gestionată prin React Query + store local.

FRQ039	Secțiunea **Date personale** va prelua automat datele utilizatorului din **MPass** la prima autentificare și/sau la refresh de profil. În backend vom expune GET /api/private/profile/me, care va agrega datele din MSign și cele locale. În UI, toate câmpurile vor fi read-only, cu excepția **telefon** și **email**, care vor fi editabile prin AntD Form + Input.
FRQ040	Pentru telefon și email vom implementa un formular de actualizare cu **validare client-side + server-side**. La apăsarea butonului **Salvează**, frontend-ul va apela PUT /api/private/profile/me/contact, iar backend-ul va salva datele în tabela user_profiles, păstrând separat valorile locale editabile față de atributele provenite din MSign.
FRQ041	Secțiunea **Companii** va conține un formular complet de înregistrare companie, implementat în React cu **Ant Design Form**, grupat pe secțiuni logice: identificare, date juridice, contacte, reprezentare. În backend vom defini DTO-uri separate pentru CompanyCreateRequest și CompanyResponse, iar datele vor fi persistate în companies.
FRQ042	Butonul **Salvare** va fi activ doar când toate câmpurile obligatorii sunt completate și bifa GDPR/consimțământ este selectată. În FE vom folosi Form.useWatch() pentru activare/dezactivare în timp real, iar în BE vom dubla această regulă prin validatori @NotBlank, @NotNull, @AssertTrue.
FRQ043	Pentru că toate câmpurile formularului de companie sunt obligatorii, vom defini schema de validare completă atât în frontend, cât și în backend. În PostgreSQL, câmpurile critice vor avea și constrângeri NOT NULL, pentru a evita inconsistențe dacă formularul este apelat direct prin API.
FRQ044	Înainte de salvare, backend-ul va executa validarea completă a payload-ului și va întoarce erori structurate pe câmpuri. Frontend-ul va mapa aceste erori direct în AntD Form.Item help/validateStatus, astfel încât utilizatorul să primească **mesaje contextuale** lângă fiecare câmp invalid.
FRQ045	La introducerea **IDNO**, sistemul va interoga automat **RSUD** printr-un serviciu backend dedicat, de tip RsudClient. Vom implementa un buton sau trigger onBlur pe IDNO, care apelează GET /api/private/rsud/company/{idno}; datele găsite vor popula automat formularul. Câmpurile neîntoarse de RSUD vor rămâne editabile pentru completare manuală.
FRQ046	Dacă IDNO nu este găsit în RSUD, backend-ul va întoarce un răspuns business de tip COMPANY_NOT_FOUND_IN_RSUD, iar frontend-ul va afișa un mesaj explicit în dreptul câmpului IDNO și/sau un Alert AntD.
FRQ047	Câmpurile completate automat din RSUD vor fi marcate în UI ca **read-only/disabled**, pentru a preveni modificarea manuală. În backend vom păstra și un set de metadate source=RSUD per atribut, astfel încât să fie clar ce vine din registru și ce vine de la utilizator.
FRQ048	Câmpurile completate manual vor putea fi actualizate oricând. Vom implementa PUT /api/private/companies/{id} pentru actualizare incrementală, cu audit al modificărilor în tabela company_audit_log și afișare în UI a datei ultimei modificări.

FRQ049	Pentru a preveni înregistrarea dublă a aceleiași companii, vom defini o **constrângere unică** pe idno în tabela companies și, suplimentar, o verificare business la nivel de serviciu. Dacă compania există deja, frontend-ul va primi un mesaj clar și, opțional, va propune asocierea la compania existentă dacă utilizatorul are drepturi pe ea.
FRQ050	Validarea pentru **email** și **telefon** va fi făcută în două straturi: regex/normalizare în frontend și validare strictă în backend. Pentru telefon vom aplica și normalizare format (ex. eliminare spații, prefix standardizat), astfel încât căutarea și compararea să fie coerente.
FRQ051	Câmpul **Reprezentant legal** va fi precompletat automat cu numele persoanei autentificate prin MSign. În backend, atributul va fi extras din identitatea autentificată și injectat în răspunsul formularului de companie; în UI va apărea prepopulat și blocat sau semiblocat, conform regulii finale de business.
FRQ052	Secțiunea **Delegare persoană** va fi implementată ca formular separat, cu câmpuri: **IDNP**, **Companie** (selectată doar din companiile utilizatorului), **Persoana autorizată** (autocompletată), plus bifa obligatorie. Pentru selectarea companiei vom folosi Select AntD alimentat din GET /api/private/companies/my.
FRQ053	La salvarea delegării, backend-ul va apela serviciul guvernamental **MPower** pentru verificarea împuternicirii persoanei pentru compania selectată. Implementarea va fi făcută printr-un adaptor MpowerClient, iar fluxul de salvare va fi **sincron**: mai întâi verificare, apoi persistare delegare dacă răspunsul este pozitiv.
FRQ054	Dacă MPower confirmă că persoana nu deține împuternicire, backend-ul va compune mesajul de eroare din datele reale ale persoanei și ale companiei, iar frontend-ul îl va afișa ca eroare blocantă. Vom păstra și log tehnic al răspunsului MPower pentru audit și troubleshooting.
FRQ055	Dacă **IDNP** nu este găsit în **RSP**, backend-ul va returna un cod business separat, iar UI va marca explicit câmpul IDNP ca invalid. Pentru acest flux vom avea un client dedicat RspClient, separat de logica MPower, ca să putem diferenția clar erorile de existență persoană vs. lipsă împuternicire.
FRQ056	După confirmarea împuternicirii, sistemul va crea o asociere în tabela user_company_roles cu rol de tip **AUTHORIZED_REPRESENTATIVE** sau echivalent. Din acel moment, compania va deveni disponibilă în toate modulele unde utilizatorul poate depune raportări/cereri pentru entitatea respectivă.
FRQ057	Secțiunea **Mesagerie** va fi implementată ca un **inbox intern de notificări**. Vom avea tabela notifications cu atribute precum: utilizator, companie, tip obiect, id obiect, titlu, mesaj, status, timestamp, link de navigare. Orice modul din sistem va putea publica notificări printr-un serviciu comun NotificationService, astfel încât toate evenimentele funcționale să fie centralizate aici.
FRQ058	O notificare își va pierde statutul de activ/necitit în momentul accesării. În backend vom expune PATCH /api/private/notifications/{id}/read, iar frontend-ul va face actualizarea la click pe notificare și va recalcula badge-ul din meniu în timp real.

6.3. Raportare deșeuri

Cerință	Implementare propusă
FRQ059	**Gate de acces**: în React Router, ruta /raportare-deseuri este disponibilă doar utilizatorilor autentificați prin **MPASS** și care au **cel puțin o companie activă** în Cabinetul personal. În backend, endpoint-urile /api/private/waste-reports/** verifică existența unei companii active asociate utilizatorului și întorc 403 cu mesaj business dacă această condiție nu este îndeplinită.
FRQ060	**Multi-companie, raport separat**: modelul de date va include tabela waste_reports(report_id, company_id, year, org_type, reporter_profile, status, ...), cu constrângere unică pe combinația relevantă, pentru a forța raportarea separată per companie și profil. În UI vom avea selector pentru companie/municipalitate și listă de rapoarte filtrată per entitate.
FRQ061	**Formular dinamic** realizat cu **Ant Design Form**, care va include câmpurile de bază: tip organizare, compania/municipalitatea, profil raportor, adresă amplasament, existența colectării pe bază de contract etc. Backend-ul va expune endpoint-uri pentru metadate și nomenclatoare (GET /api/private/waste-reports/form-metadata, GET /api/private/companies/my), astfel încât frontend-ul să încarce dinamic opțiunile disponibile.
FRQ062	**Un singur profil raportor per raport**: în frontend, câmpul va fi implementat ca Select simplu, fără selecție multiplă. În backend, validarea va impune existența unui singur reporterProfile și va preveni dublarea rapoartelor pentru aceeași combinație companie + an + profil.
FRQ063	**Autocompletare adresă, dar cu posibilitate de editare**: dacă utilizatorul indică faptul că amplasamentul coincide cu adresa juridică, frontend-ul va prelua automat valorile din profilul companiei și le va completa în formular, fără a le bloca. În backend, datele de adresă vor fi salvate în raport ca snapshot, pentru a păstra istoricul exact al valorilor la momentul depunerii.
FRQ064	**Validare automată a autorizației de mediu** prin integrare cu **SIA GEAP via MConnect**. La introducerea numărului autorizației, frontend-ul va apela un endpoint de validare, iar backend-ul va verifica existența și valabilitatea acesteia printr-un GeapMconnectClient. Dacă autorizația este validă, datele relevante vor fi completate automat; dacă nu, utilizatorul va primi un mesaj explicit de eroare.
FRQ065	**Butonul "Trimite spre verificare"** va fi activ doar după validarea completă a formularului și confirmarea utilizatorului. În backend, schimbarea statusului în IN_REVIEW se va face doar după revalidarea tuturor regulilor de business, nu doar pe baza validării din frontend.
FRQ066	**Câmpuri obligatorii și reguli condiționale**: în frontend se vor implementa validări cu AntD rules, iar în backend cu Bean Validation și validatori custom. Exemplu: numărul autorizației va fi obligatoriu doar dacă utilizatorul declară că deține autorizație, iar anumite câmpuri geografice vor deveni obligatorii doar dacă este selectat un anumit tip de activitate sau tip de colectare.

FRQ067	**Interfață Inspector AM**: va exista un ecran dedicat de monitorizare a rapoartelor, cu tabel server-side și acțiuni precum **Validează**, **Trimite spre corectare** și **Respinge**. În backend, acțiunile vor fi expuse printr-un endpoint de workflow, accesibil doar utilizatorilor cu rolul de inspector, și vor necesita comentariu obligatoriu unde impune fluxul de business.
FRQ068	**Istoric și vizualizare a stadiilor**: vom implementa tabela waste_report_history, care va înregistra toate tranzițiile de status, actorul, data și comentariile. În frontend, istoricul va fi afișat printr-un Timeline sau Steps, astfel încât utilizatorul și inspectorul să poată vedea clar parcursul raportului.
FRQ069	**Re-editare înainte de preluarea de către inspector**: cât timp raportul este trimis spre verificare, dar nu a fost preluat de un inspector, utilizatorul va putea reveni în modul de editare. În backend, această regulă va fi controlată prin verificarea unui câmp precum pickedAt sau assignedInspectorId; dacă acestea nu sunt setate, raportul poate fi mutat înapoi în status de completare.
FRQ070	**Blocarea editării după preluare**: în momentul în care un inspector a preluat raportul, frontend-ul va dezactiva acțiunea de editare, iar backend-ul va bloca orice încercare de modificare prin validare de status. Mesajul afișat utilizatorului va fi unul clar și contextual.
FRQ071	**Foldere/categorii pe status pentru inspector**: interfața inspectorului va fi organizată pe categorii de tip “trimise spre verificare”, “trimise spre corectare”, “verificare IPM”, “luarea deciziei”, “aprobate”, “respinse”. În frontend vom utiliza Tabs sau meniu lateral, iar backend-ul va oferi atât liste filtrate pe status, cât și contori pentru fiecare categorie.
FRQ072	**Contorizare, paginare și sortare**: backend-ul va returna rezultate paginate, implicit cu maximum 50 de înregistrări pe pagină și sortare descrescătoare după data creării. În tabel vor fi afișate câmpurile esențiale: ID raport, companie, data depunerii, anul, persoana responsabilă, profil și statusul curent.
FRQ073	**Escaladare la IPM și încărcarea actului de inspecție**: dacă inspectorul AM identifică neconcordanțe, raportul va fi transmis în etapa IPM_REVIEW, iar sistemul va genera notificare către inspectorul IPM. În interfața IPM va exista o componentă Upload pentru atașarea actului de inspecție, fișierul fiind salvat în object storage, iar metadatele în tabela de atașamente.
FRQ074	**Luarea deciziei finale**: când raportul ajunge în etapa de decizie, inspectorul AM va putea aproba, respinge sau retrimite raportul înapoi în verificare, în funcție de rezultat. Toate aceste acțiuni vor fi gestionate prin același mecanism central de workflow și vor fi logate în istoricul raportului.

6.4. Cerere de înregistrare a sistemului colectiv

Cerință	Implementare propusă
FRQ075	Accesul la modulul **Cerere de înregistrare a sistemului colectiv** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă înregistrată în secțiunea **Companii**. În frontend, ruta va fi protejată prin route guard,

	iar în backend endpoint-urile /api/private/collective-systems/** vor verifica existența unei companii active asociate utilizatorului.
FRQ076	Modelul de date va permite mai multe companii per utilizator, dar fiecare cerere va fi legată de **o singură companie**. Vom introduce tabela collective_system_requests cu relație many-to-one către companies, iar la nivel de business vom impune completarea separată a cererilor per companie și, dacă va fi necesar ulterior, per categorie operațională. În UI, compania se va selecta explicit dintr-un Select alimentat din companiile utilizatorului.
FRQ077	Formularul va fi implementat cu **Ant Design Form**, împărțit în blocuri logice: date companie, autorizație de mediu, documente atașate, persoană autorizată, confirmare finală. Câmpul **Numele companiei** va fi selectabil doar din lista companiilor existente în cabinet, **Numărul autorizației** va fi introdus manual și validat prin integrare cu **SIA GEAP via MConnect**, iar câmpurile **Data emiterii** și **Valabil până la** vor fi completate automat și afișate read-only după validare. Pentru **Atașare fișiere** vom folosi componenta Upload, iar fișierele vor fi stocate în object storage, cu metadate în DB. **Persoana autorizată** va fi completată automat din profilul utilizatorului și/sau din companie.
FRQ078	Butonul **Trimite spre verificare** va fi activ doar după ce toate câmpurile obligatorii sunt completate și checkbox-ul de confirmare este bifat. În frontend vom folosi Form.useWatch() pentru activare în timp real, iar în backend vom dubla regula cu validări Bean Validation și business validation înainte de schimbarea statusului în IN_REVIEW.
FRQ079	Câmpurile obligatorii vor fi modelate explicit în DTO-ul CollectiveSystemRequestCreateRequest și validate atât în UI, cât și în backend. Pentru consistență, în baza de date vom aplica constrângeri NOT NULL pe companie, număr autorizație, persoană autorizată și confirmarea de consimțământ.
FRQ080	Verificarea **Numărului autorizației de mediu** va fi implementată printr-un adaptor dedicat GeapMconnectClient. La onBlur sau la apăsarea unui buton de validare, frontend-ul va apela un endpoint intern, iar backend-ul va interoga SIA GEAP prin MConnect. Dacă autorizația există și este valabilă, sistemul va completa automat datele auxiliare; dacă nu există, utilizatorul va primi un mesaj de eroare contextual, afișat lângă câmp și într-un Alert de formular.
FRQ081	Pentru **Inspectorul Agenției de Mediu** vom implementa o interfață server-side cu acțiuni de workflow: **Acceptă cererea**, **Trimite spre corectare**, **Respinge/Invalidează**. Aceste acțiuni vor fi expuse printr-un endpoint de tip /api/private/collective-system-requests/{id}/actions, protejat prin roluri. La acțiunile care necesită justificare, comentariul va fi obligatoriu și salvat în istoric.
FRQ082	Pentru fiecare cerere vom păstra un istoric complet în tabela collective_system_request_history, care va conține statusul anterior, statusul nou, actorul, timestamp-ul și comentariul. În frontend, istoricul va fi afișat prin Timeline sau Steps, astfel încât utilizatorul și inspectorii să poată vedea clar traseul cererii: completare, verificare, corectare, aprobare, respingere.

FRQ083	Utilizatorul va putea reactiva cererea în mod de editare atât timp cât aceasta nu a fost preluată de inspector în procesare. În backend vom folosi un câmp de tip pickedBy / pickedAt sau processingStartedAt; dacă acesta nu este setat, acțiunea **Editare** va fi permisă și va muta automat statusul din IN_REVIEW în DRAFT.
FRQ084	După ce inspectorul a accesat și a început procesarea cererii, sistemul va bloca editarea. Frontend-ul va dezactiva butonul de editare, iar backend-ul va întoarce eroare business dacă totuși se încearcă modificarea prin API. Mesajul de eroare va fi exact unul contextual și ușor de înțeles pentru utilizator.
FRQ085	În interfața inspectorului și a conducerii Agenției de Mediu vom organiza cererile pe **foldere logice / categorii de status**, implementate prin Tabs sau meniu lateral cu badge-uri de număr. Statusurile vor fi mapate într-un enum de backend și vor alimenta direct atât lista curentă de cereri, cât și contorii per categorie.
FRQ086	Fiecare folder, în afară de **Lista sistemelor colective**, va afișa un tabel server-side cu paginare și sortare descrescătoare după data creării. Backend-ul va returna maximum 50 de înregistrări pe pagină, iar frontend-ul va afișa coloanele cerute: ID unic cerere, companie, data depunerii, număr autorizație, persoană responsabilă, statusul curent. Pentru performanță, vom indexa în PostgreSQL câmpurile status, created_at, company_id și permit_number.
FRQ087	Folderul **Lista sistemelor colective** va fi alimentat din cererile aprobate și va utiliza o tabelă dedicată sau o vedere materializată, de tip collective_system_registry. În UI vom afișa un tabel paginat cu număr de înregistrare, companie, persoană responsabilă, data obținerii statutului și valabilitatea. Valabilitatea va fi calculată din starea autorizației și va fi afișată sub formă de badge Activ / Expirat.
FRQ088	După aprobarea cererii, sistemul va genera automat două notificări: una internă în **Mesagerie** și una prin **email**. Vom implementa un NotificationService comun și un EmailService asincron, astfel încât aprobarea să nu fie blocată de timpul de trimitere al emailului.
FRQ089	Numărul de înregistrare al sistemului colectiv va fi generat automat în backend în momentul acceptării cererii de către inspector. Vom implementa un serviciu dedicat RegistrationNumberGenerator, care va construi identificatorul conform regulilor din cerințele generale / nefuncționale și va garanta unicitatea prin secvență DB și constrângere unică.
FRQ090	Sistemul va executa **verificări periodice** ale valabilității autorizației de mediu pentru sistemele colective aprobate. Vom implementa un job programat (@Scheduled) care va reinteroga SIA GEAP prin MConnect sau va consuma un mecanism de sincronizare periodică. Dacă autorizația este expirată sau retrasă, statutul sistemului colectiv va fi mutat automat în WITHDRAWN, compania va fi notificată, iar toate funcționalitățile de raportare REP vor fi blocate pentru acel sistem. Tot aici vom aplica și blocarea selecției acestui sistem colectiv în alte fluxuri, cum este cererea pentru Lista Producătorilor. Pentru entitățile deja afiliate la acest sistem vom genera notificări în masă și vom restricționa drepturile de raportare conform regulii de business.

FRQ091	Inspectorul de mediu va avea o acțiune explicită de **radiere / retragere** a unui sistem colectiv din lista aprobată. Această acțiune va muta entitatea în categoria WITHDRAWN, va solicita motiv obligatoriu și va fi înregistrată complet în istoric și audit log.
FRQ092	După obținerea statutului de sistem colectiv, compania va fi adăugată automat în **Lista sistemelor colective** și va avea o pagină de detaliu proprie. Pagina de detaliu va fi implementată ca un ecran dedicat, cu tab-uri pentru: date de identificare, date de contact, informații despre autorizație, activitatea în sistem, lista agenților economici afiliați și documente asociate. Pentru încărcarea documentelor rezultate din controalele IPM vom implementa o zonă separată de upload, disponibilă rolului IPM, cu stocare în object storage și jurnalizare completă a tuturor fișierelor încărcate.

6.5. Cerere de acordare a numărului de înregistrare în Lista Producătorilor

Cerință	Implementare propusă
FRQ093	Accesul la modulul **Cerere de acordare a numărului de înregistrare în Lista Producătorilor** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă înregistrată în secțiunea **Companii** din Cabinetul personal. În frontend, ruta va fi protejată prin guard, iar în backend endpoint-urile /api/private/producer-registration-requests/** vor verifica existența unei companii active asociate utilizatorului.
FRQ094	Vom modela cererea astfel încât să fie legată de **o singură companie** și **o singură categorie REP**. În baza de date vom avea tabela producer_registration_requests cu câmpuri precum company_id, rep_category, responsibility_type, status, registration_number, valid_until, iar la nivel de business vom preveni depunerea simultană a două cereri active pentru aceeași combinație companie + categorie REP.
FRQ095	Formularul va fi implementat în **ReactJS + Ant Design Form**, cu secțiuni distincte: selecția companiei, categoria REP, tip de responsabilitate, selectarea sistemului colectiv, tabelul de estimări, atașamente, persoana autorizată și confirmarea finală. Câmpul **Numele companiei** va fi un Select alimentat din companiile utilizatorului. **Categoria REP** va fi un Select single-choice. **Tip de responsabilitate** va fi un Radio.Group sau Select, iar dacă este ales colectivă, frontend-ul va afișa condiționat câmpul **Selectați sistemul colectiv**, alimentat din registrul sistemelor colective active. Tabelul **Informații generale estimate pentru anul în care se face înregistrarea** va fi randat dinamic în funcție de categoria REP selectată, pe baza unui configurator de formulare și a unor șabloane de tabel predefinite. Pentru fiecare categorie REP vom avea o configurație proprie în backend, de tip rep_form_templates, astfel încât frontend-ul să poată genera tabelul fără hardcodare excesivă. Pentru **DEEE + sistem individual**, sistemul va calcula automat **garanția financiară** pe baza datelor estimate și a costurilor operaționale preluate din tabela de costuri operaționale DEEE. Câmpul **Plan operațional** va fi implementat

	prin Upload, cu stocare în object storage și metadate în DB. **Persoana autorizată** va fi preluată automat din profilul utilizatorului și/sau din compania selectată.
FRQ096	Butonul **Trimite spre verificare** va deveni activ doar după validarea completă a formularului și bifarea consimțământului. În frontend vom folosi validări reactive prin Form.useWatch() și form.validateFields(), iar în backend vom repeta toate validările înainte de schimbarea statusului în IN_REVIEW, astfel încât să nu existe dependență exclusivă de validarea din UI.
FRQ097	Câmpurile obligatorii vor fi implementate prin reguli explicite atât în frontend, cât și în backend. Pentru tabelul de estimări vom aplica o regulă de tip business: trebuie să existe **cel puțin o cantitate > 0** într-un rând relevant. **Valoarea garanției financiare** nu va fi introdusă manual, ci va fi generată automat de backend și returnată ca valoare calculată. **Persoana autorizată** va fi autocompletată și validată ca prezență. **Planul operațional** va fi obligatoriu prin verificarea existenței unui attachment de tip OPERATIONAL_PLAN.
FRQ098	În tabelul de estimări, toate câmpurile de cantitate lăsate necompletate vor fi convertite automat la valoarea 0. În frontend, la serializarea formularului, valorile null/undefined/empty string vor fi transformate în zero pentru coloanele de cantitate. În backend, vom mai aplica încă o normalizare înainte de persistare, astfel încât datele salvate și cele utilizate în calcule să fie consistente.
FRQ099	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran de lucru dedicat, în zona **Monitorizare și Raportare → Raportare REP → Cereri de acordare a numărului de înregistrare**. Acesta va afișa cererile într-un tabel server-side și va permite acțiuni precum **Acceptă**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de workflow, de tip /api/private/producer-registration-requests/{id}/actions, care va valida permisiunile, statusul curent și existența comentariului obligatoriu la corectare sau respingere.
FRQ100	Pentru fiecare cerere vom menține un istoric complet într-o tabelă producer_registration_request_history, cu câmpuri pentru status vechi, status nou, actor, dată, comentariu și eventual document asociat. În frontend, istoricul va fi afișat sub formă de Timeline sau Steps, cu etapele: **Completare cerere**, **Verificare cerere**, **Trimis spre corectare**, **Trimis spre semnare**, **Cerere aprobată**, **Cerere respinsă**.
FRQ101	Utilizatorul va putea relua în editare cererea cât timp aceasta nu a fost preluată în procesare de către inspector. În backend vom folosi câmpuri precum assigned_inspector_id, picked_at sau processing_started_at. Dacă acestea sunt goale, acțiunea de editare va fi permisă și va muta cererea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat condiționat pe baza stării și a flagului editable=true venit din API.
FRQ102	Dacă inspectorul a preluat cererea, editarea va fi blocată. Frontend-ul va dezactiva butonul de editare și va afișa mesajul contextual stabilit. Backend-ul va refuza orice update dacă cererea este deja în procesare, întorcând un cod business specific, de exemplu REQUEST_ALREADY_IN_PROCESSING.

FRQ103	În interfața inspectorului și a conducerii Agenției de Mediu, cererile vor fi organizate în **foldere logice pe status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri de număr. În backend, statusurile vor fi mapate într-un enum și vom avea endpoint-uri pentru listă și contori. Pentru **Lista Producătorilor**, vom crea și subcategorii pe categorie REP: DEEE, ambalaje, baterii și acumulatori, anvelope uzate, uleiuri uzate, vehicule scoase din uz, produse textile.
FRQ104	Folderele de tip **Cereri trimise spre verificare**, **Cereri trimise spre corectare**, **Cereri trimise spre semnare** și **Cereri respinse** vor afișa tabele server-side, paginate la maximum 50 de înregistrări pe pagină, ordonate descrescător după data creării. Coloanele vor fi: ID cerere, companie, data depunerii, categorie REP, persoană responsabilă, tip de responsabilitate și status. În PostgreSQL vom indexa status, created_at, company_id, rep_category pentru performanță.
FRQ105	Toate subfolderele din **Lista Producătorilor** vor fi alimentate din înregistrările active aprobate. Vom avea o tabelă producer_registry_entries sau o vedere materializată generată din cererile aprobate și semnate. UI-ul va afișa maximum 50 de înregistrări pe pagină, cu sortare descrescătoare și coloanele: număr de înregistrare, companie, persoană responsabilă, tip de responsabilitate, valabilitate. Câmpul **valabilitate** va fi calculat automat pe baza datei de expirare și afișat sub formă de badge Activ / Expirat.
FRQ106	După acceptarea cererii de către inspector, sistemul va muta automat cererea în **Cereri trimise spre semnare** și va genera documentul oficial conform anexei aplicabile. Tehnic, vom implementa un serviciu de generare documente, care va popula un șablon DOCX/HTML cu datele cererii și îl va converti în PDF pentru previzualizare și semnare. Documentul generat va fi salvat în storage, iar metadatele în tabela generated_documents.
FRQ107	**Conducerea Agenției de Mediu** va avea un ecran dedicat pentru vizualizarea detaliilor cererii și a documentului generat. În frontend vom afișa metadatele cererii și un viewer PDF integrat. Va exista opțiune de **previzualizare**, **descărcare** și **inițiere semnare**. Endpoint-urile vor fi separate pentru datele cererii și pentru documentul generat, pentru a menține clară responsabilitatea componentelor.
FRQ108	Semnarea documentului se va face prin **MSIGN**, fără descărcarea fișierului de către utilizator. Backend-ul va iniția fluxul de semnare prin integrarea cu serviciul de semnare, transmiterea documentului hash-uit sau a pachetului de semnare și preluarea rezultatului semnat. După semnare, PDF-ul semnat va fi salvat ca versiune oficială. Conducerea Agenției de Mediu va putea și să retrimită cererea înapoi spre verificare, printr-o acțiune de workflow ce mută statusul din PENDING_SIGNATURE în IN_REVIEW.
FRQ109	După semnare, documentul devine oficial și sistemul va declanșa automat trei acțiuni: actualizarea stării cererii în APPROVED, activarea numărului de înregistrare și notificarea companiei. Vom implementa un NotificationService pentru inbox-ul intern și un EmailService asincron pentru trimiterea notificării pe email. Documentul

	semnat va fi disponibil pentru descărcare din detaliul cererii și din profilul entității înregistrate.
FRQ110	Numărul de înregistrare va fi generat automat de sistem în momentul în care inspectorul acceptă cererea. Vom implementa un serviciu RegistrationNumberGenerator, bazat pe secvență DB și reguli de formatare configurabile conform cerințelor generale de numerotare. Numărul va fi rezervat tranzacțional, pentru a evita dublurile în caz de concurență.
FRQ111	Dacă aceeași companie este înregistrată pe mai multe categorii REP, fiecare categorie va primi propriul număr de înregistrare. În modelul de date, înregistrarea va fi legată de combinația companie + categorie REP, nu doar de companie. Astfel, fiecare intrare aprobată va exista ca entitate distinctă în registru.
FRQ112	După semnarea cererii, sistemul va actualiza automat toate interfețele relevante: lista internă a cererilor, **Lista Producătorilor** din zona internă și zona publică, detaliile companiei și eventualele dropdown-uri sau filtre care folosesc acest registru. Vom implementa această actualizare prin persistarea intrării în registru și invalidarea cache-urilor aferente, astfel încât datele noi să devină imediat vizibile.
FRQ113	Odată aprobată și semnată, compania va fi plasată automat în subfolderul categoriei REP pentru care a fost depusă cererea. În backend, câmpul rep_category va determina folderul logic și filtrarea. În frontend, aceasta va apărea automat în secțiunea corespunzătoare, fără intervenție manuală.
FRQ114	Folderul **Lista Producătorilor** va reprezenta registrul tuturor agenților economici înregistrați pe diferite categorii REP. Tehnic, acesta va fi o agregare a intrărilor active din producer_registry_entries, grupate și filtrabile după categorie. Vom putea extinde ulterior această zonă și cu filtre suplimentare după tip responsabilitate, status, companie, IDNO și perioadă de valabilitate.
FRQ115	Pentru fiecare agent economic din Lista Producătorilor va exista o **pagină de detaliu**. Aceasta va fi implementată ca ecran dedicat, cu tab-uri pentru: date identificare, contacte, activitate, cereri depuse, rapoarte transmise, notificări, documente și istoric. În această pagină va exista și o zonă de **upload documente** disponibilă rolului **IPM**, pentru atașarea documentelor rezultate din controale. Fișierele vor fi salvate în object storage, iar jurnalizarea se va face într-o tabelă entity_documents.
FRQ116	Toată informația din **Lista Producătorilor** va fi expusă către **SFS** prin **MConnect**. Vom implementa un adaptor de interoperabilitate și un set de endpoint-uri sau servicii expuse conform contractului convenit. Datele vor fi oferite în format structurat, cu posibilitate de filtrare după categorie, companie sau status, în funcție de cerințele de integrare aprobate.
FRQ117	Cu 2 luni înainte de expirarea termenului de 5 ani, sistemul va trimite notificări automate în inbox și pe email. Vom implementa un job programat (@Scheduled) care identifică înregistrările ce se apropie de expirare și generează notificările aferente. În mesaj va fi inclusă acțiunea recomandată: depunerea unei noi cereri pentru aceeași categorie REP.

FRQ118	Dacă în intervalul de 2 luni nu este depusă o nouă cerere, înregistrarea veche va fi marcată automat ca **expirată**, iar compania va fi scoasă din Lista Producătorilor active. În modelul de date vom separa statusurile ACTIVE, EXPIRED, WITHDRAWN, iar interfața publică și internă va afișa doar înregistrările active în listele operaționale.
FRQ119	Inspectorul de Mediu și/sau conducerea Agenției de Mediu vor avea posibilitatea de a **radia** un agent economic din Lista Producătorilor în cazurile prevăzute de business. Vom implementa o acțiune explicită de workflow care mută înregistrarea în categoria **Numere de înregistrare retrase**, solicită motiv obligatoriu și jurnalizează complet evenimentul. În plus, sistemul va declanșa notificări, va actualiza interfața publică și va bloca utilizarea înregistrării retrase în fluxurile operaționale viitoare.

6.6. Notificare de intenție

Cerință	Implementare propusă
FRQ120	Accesul la modulul **Notificare de intenție** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă înregistrată în secțiunea **Companii** din Cabinetul personal. În frontend, ruta va fi protejată prin route guard, iar în backend endpoint-urile /api/private/intention-notifications/** vor valida existența unei companii active asociate utilizatorului.
FRQ121	Modelul de date va permite mai multe companii per utilizator, însă fiecare notificare va fi legată de **o singură companie** și **o singură categorie REP**. Vom introduce tabela intention_notifications cu câmpuri precum company_id, rep_category, notification_type, collective_system_id, status, submitted_at, responsible_person_id. La nivel de business vom preveni existența simultană a două notificări active pe aceeași combinație companie + categorie + tip notificare, dacă regula operațională va impune unicitate în aceeași perioadă.
FRQ122	Înainte de accesarea formularului, sistemul va verifica două condiții: existența a cel puțin unei companii în Cabinetul personal și existența unei înregistrări active în **Lista Producătorilor** pentru cel puțin o categorie REP. În backend vom implementa un EligibilityService care va valida aceste condiții și va întoarce motive clare de blocare, iar în frontend utilizatorul va vedea un mesaj contextual și link direct către secțiunea care trebuie completată mai întâi.
FRQ123	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form**, cu validare dinamică și afișare condiționată a câmpurilor. **Numele companiei** va fi selectat exclusiv din companiile utilizatorului. **Categoria REP** va fi Select obligatoriu, cu opțiunile configurate în nomenclator. **Tip de notificare** va fi Select sau Radio.Group, iar dacă utilizatorul selectează **aderare la sistem colectiv**, se va afișa dinamic câmpul **Selectați sistemul colectiv**, alimentat din registrul sistemelor colective active. **Atașare fișiere** va fi implementată prin componenta Upload, cu stocare în object storage și metadate în DB. **Persoana autorizată** va fi preluată automat din profilul utilizatorului și/sau din compania selectată. Textul de confirmare va fi implementat prin checkbox obligatoriu și salvat în audit odată cu transmiterea formularului.

FRQ124	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea confirmării. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend vom repeta toate validările înainte de schimbarea statusului în IN_REVIEW, astfel încât sistemul să nu depindă exclusiv de validarea din UI.
FRQ125	Toate câmpurile formularului vor fi tratate ca obligatorii în DTO-urile backend și în schema formularului frontend. Pentru câmpurile condiționale, cum este **Selectați sistemul colectiv**, regula de obligativitate se va activa doar când tipul notificării o cere. În PostgreSQL, câmpurile cheie vor avea constrângeri NOT NULL, iar pentru atașamente vom valida existența lor doar dacă fluxul de business o cere.
FRQ126	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Notificări de intenție**, cu tabel server-side și acțiuni de workflow: **Acceptă notificarea**, **Trimite spre corectare**, **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/intention-notifications/{id}/actions, care va valida rolul utilizatorului, starea curentă și existența comentariului obligatoriu pentru corectare sau respingere.
FRQ127	Pentru fiecare notificare vom menține un istoric complet într-o tabelă intention_notification_history, cu câmpuri precum notification_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare notificare**, **Verificare notificare**, **Trimis spre corectare**, **Notificare aprobată**, **Notificare respinsă**.
FRQ128	Utilizatorul va putea relua în editare notificarea transmisă atât timp cât aceasta nu a fost preluată în procesare de către inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul de editare va fi afișat pe baza unui flag editable=true returnat de API.
FRQ129	Dacă notificarea a fost deja accesată și preluată în procesare, editarea va fi blocată. Frontend-ul va dezactiva butonul și va afișa mesajul contextual cerut, iar backend-ul va bloca orice încercare de update și va întoarce un cod business dedicat, de exemplu NOTIFICATION_IN_PROCESSING, pentru a evita modificările concurente.
FRQ130	În interfața inspectorului, notificările vor fi grupate pe **foldere logice / categorii de status**, implementate în React prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per status, astfel încât fiecare tab să afișeze rapid numărul de notificări aflate în etapa respectivă.
FRQ131	Fiecare folder va afișa notificările într-un tabel server-side, paginat cu maximum 50 de înregistrări pe pagină și ordonat descrescător după data depunerii sau creării. Coloanele vor fi: **ID notificare**, **numele companiei**, **tip notificare**, **data depunerii**, **categorie REP**, **persoană responsabilă**, **etapa notificării**. În PostgreSQL vom indexa status, company_id, rep_category, notification_type, submitted_at pentru

	performanță, iar în frontend vom folosi Ant Design Table cu sortare, filtrare și navigare spre ecranul de detalii.

6.7. Declarație pe propria răspundere pentru categoriile exceptate

Cerință	Implementare propusă
FRQ132	**Precondiție companie**: în FE (React Router) ruta /declaratii-exceptate este protejată (MPASS + hasActiveCompany). În BE, toate endpoint-urile /api/private/exempt-declarations/** verifică existsActiveCompany(userId) și întorc 403 cu motiv business dacă nu este îndeplinit.
FRQ133	**O declarație = o companie + o configurație de produse**: DB: exempt_declarations(id, company_id, declaration_type, product_type, payload_json, payload_hash, status, submitted_at, picked_at, picked_by, ...). payload_hash (fingerprint) permite reguli: pentru aceeași companie, dacă lista produselor/valorilor se modifică → se creează o declarație nouă (nu se suprascrie cea veche). FE: selector companie + wizard/form; BE: validare că o declarație este asociată unei singure companii și logica de versionare (create new + history).
FRQ134	**Acces doar după înregistrare companie**: aceeași regulă ca FRQ132, dar aplicată și în UI (banner + buton “Mergi la Cabinet → Companii”), plus blocare buton “Creează declarație” dacă nu există companii disponibile în GET /api/private/companies/my.
FRQ135	**Formular complet + text dinamic**: FE: AntD Form cu câmpuri: companie (Select din Cabinet), tip declarație (Select obligatoriu: <100kg ambalaje/an vs import consum propriu), telefon/email autocompletate din companie, atașamente (Upload), persoana autorizată autocompletată din profil, funcția persoanei autorizate (Input/Select), tip produs importat (Select: ambalaje/vehicule/baterii/EEE/uleiuri/anvelope/textile). **Date specifice produsului**: pentru **ambalaje** și **baterii** se afișează un tabel editabil (AntD Table cu celule editabile) conform anexei relevante; pentru celelalte tipuri (vehicule/EEE/uleiuri/anvelope/textile) se afișează un câmp numeric “masă (kg)”. **Text declarație**: se generează dintr-un “template engine” intern (BE) + placeholders (companie, an, cantități, kg), astfel încât UI să afișeze previzualizarea declarației (HTML) în timp real. BE păstrează și snapshot-ul textului generat (pentru audit/istoric). Endpoint-uri tipice: GET /api/private/exempt-declarations/form-metadata, POST /api/private/exempt-declarations/draft, GET /api/private/exempt-declarations/{id}/preview.
FRQ136	**“Trimite spre verificare” doar după validare + confirmare**: FE: form.validateFields() + checkbox obligatoriu (confirmare) înainte de enable; BE: POST /api/private/exempt-declarations/{id}/submit revalidează complet DTO + reguli condiționale, apoi mută status în IN_REVIEW și setează submitted_at.
FRQ137	**Obligatoriu peste tot, cu excepție pentru atașamente**: FE: regula required pentru toate câmpurile; Upload devine **opțional** doar când tip declarație = <100kg ambalaje/an. BE: aceeași regulă (business validation) + constrângeri DB NOT NULL

	pe câmpurile critice; atașamente validate condiționat (există cel puțin un fișier) doar pentru tipurile unde e cerut.
FRQ138	**Workflow Inspector AM (Acceptă / Corectare / Respinge)**: FE inspector: tabel + ecran detaliu cu acțiuni. BE: endpoint unificat POST /api/private/exempt-declarations/{id}/actions cu actionType (APPROVE, SEND_BACK, REJECT) + comment obligatoriu pentru SEND_BACK/REJECT. Roluri: ROLE_AM_INSPECTOR. La APPROVE se finalizează declarația; la SEND_BACK se întoarce în NEEDS_CORRECTION; la REJECT → REJECTED.
FRQ139	**Istoric etape**: DB: exempt_declaration_history(id, declaration_id, from_status, to_status, actor_id, comment, created_at). FE: Timeline/Steps cu etapele: Completare → Verificare → Trimis spre corectare → Aprobat → Respins.
FRQ140	**Editare permisă până la preluare**: BE: dacă picked_at este null și status este IN_REVIEW, utilizatorul poate "Editare", iar sistemul mută status în DRAFT (sau COMPLETION) și scrie în istoric. FE: buton "Editare" vizibil doar când API returnează editable=true. Endpoint: POST /api/private/exempt-declarations/{id}/back-to-edit.
FRQ141	**Blocare editare după preluare**: BE: dacă picked_at != null atunci update/întoarcere în editare este refuzată cu eroare business (409) și mesaj clar; FE: dezactivează acțiunea și afișează mesaj contextual.
FRQ142	**Foldere pe status (Inspector)**: FE inspector: Tabs/meniu lateral cu categorii: "Trimise spre verificare", "Trimise spre corectare", "Aprobate", "Respinse". BE: GET /api/private/exempt-declarations?status=... + GET /api/private/exempt-declarations/counts pentru badge-uri.
FRQ143	**Listare max 50/pagină, sort desc + coloane standard**: BE: paginare server-side (size=50 default) + sort created_at desc/submitted_at desc. FE: AntD Table cu coloane: ID unic declarație, companie, tip declarație, data depunerii, persoană responsabilă, etapa (status). Indexare DB pe status, submitted_at/created_at, company_id.
FRQ144	**Prevenire dublă înregistrare (<100kg → REP ADA)**: BE introduce o regulă transversală în fluxul de **înregistrare REP ADA** (din 9.1.5): la inițiere/aprobare se verifică dacă agentul economic are o declarație activă în registrul "<100kg ambalaje/an". Dacă există, sistemul: (1) blochează aprobarea automată sau marchează cazul ca "needs attention", (2) creează o notificare internă către inspector ("există în lista <100kg; trebuie radiat înainte de ADA"). FE inspector: banner/flag pe cererea ADA + link către declarația <100kg.
FRQ145	**Radiere din lista <100kg (Inspector)**: acțiune dedicată în UI inspector pe declarațiile/înregistrările <100kg: "Radiază (trecere la EPR)", cu motiv obligatoriu. BE: POST /api/private/exempt-declarations/{id}/withdraw → status WITHDRAWN, scriere în istoric + notificare către companie (inbox + email). Această acțiune deblochează ulterior aprobarea cererii REP ADA.
FRQ146	**Transmitere către SFS și SV prin MConnect**: implementăm un adaptor de interoperabilitate care expune seturi de date pentru: (a) lista agenților <100kg ambalaje/an, (b) lista importatorilor pentru consum propriu, cu câmpuri

	standardizate (IDNO, denumire, tip declarație, status, perioadă, date contact, produse/cantități). Susținem consum **incremental** (ex. “de la lastUpdatedAt”) și audit pentru livrări. Opțional: un job de sincronizare/replicare dacă se cere model push.

6.8. Calculul garanției financiare pentru DEEE

Cerință	Implementare propusă
FRQ147	Accesul la modulul **Calculul garanției financiare pentru DEEE** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și o înregistrare activă în **Lista Producătorilor** pentru categoria **DEEE**. În backend vom implementa un EligibilityService care verifică aceste precondiții înainte de a permite accesul la endpoint-urile /api/private/deee-financial-guarantees/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat mai întâi.
FRQ148	Pentru fiecare companie se va completa un formular separat. În modelul de date vom avea tabela deee_financial_guarantee_requests cu legătură many-to-one către companies. În UI, compania va fi selectată explicit din lista companiilor utilizatorului, iar fiecare înregistrare va avea propriul istoric, propriile calcule și propriul flux de aprobare.
FRQ149	Formularul pentru anul curent va putea fi completat doar dacă agentul economic a depus **Raportarea DEEE pentru anul precedent**. În backend, accesarea formularului și acțiunea de submit vor verifica existența unei raportări DEEE pentru anul anterior, ideal într-un status aprobat sau cel puțin existent conform regulii finale de business. Dacă lipsesc datele sursă, modulul va fi blocat și utilizatorul va fi redirecționat către raportarea necesară.
FRQ150	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va conține: **Numele companiei**, **Tabelul Calculul garanției financiare**, **Suma calculată**, **Balanța contului**, **Atașare fișiere**, **Persoana autorizată** și checkbox-ul de confirmare. Tabelul principal va fi randat dinamic pe baza unei configurații de structură corespunzătoare anexei, folosind Ant Design Table cu celule calculate automat. Atașamentele vor fi încărcate prin Upload, salvate în object storage, iar metadatele lor în DB. **Persoana autorizată** va fi preluată automat din profilul utilizatorului și/sau din compania selectată.
FRQ151	Pentru **primul an de activitate**, după selectarea companiei, singurul câmp activ va fi **Suma calculată**. Backend-ul va prelua automat valoarea din câmpul **Valoarea garanției financiare** generat anterior în **Cererea de acordare a numărului de înregistrare pentru DEEE**. În frontend, tabelul de calcul și celelalte coloane dependente vor fi afișate read-only sau ascunse, în funcție de UX-ul ales.
FRQ152	Pentru anii următori, **Suma calculată** va fi determinată automat de un serviciu backend dedicat, de tip FinancialGuaranteeCalculationService, care va aplica formula **GE = Σ(N'Ei × Ci)**. Vom păstra logica de calcul exclusiv în backend, pentru a

	evita discrepanțe între UI și persistență. Frontend-ul va afișa rezultatul calculat în timp real, dar valoarea oficială va fi cea returnată de backend.
FRQ153	În al doilea an de activitate, coloana **Cantitatea de EEE estimate a fi introduse pe piața națională pentru anul precedent** va fi populată automat din tabelul **Informații generale estimate pentru anul în care se face înregistrarea** din cererea de înregistrare DEEE. În backend vom implementa un serviciu de agregare care citește datele istorice din modulul de înregistrare și le mapează pe categoriile DEEE corespunzătoare tabelului actual.
FRQ154	Pentru anii următori, aceeași coloană va fi completată automat din cererea de **Calcul al garanției financiare** din anul anterior, mai exact din coloana **Cantitatea de EEE estimate a fi introduse pe piața națională în anul în curs**. Vom implementa versionare anuală pe aceeași companie și categorie, astfel încât sistemul să poată identifica clar documentul precedent și să preia automat valorile relevante.
FRQ155	Coloana **Cantitatea de EEE efectiv introduse pe piața națională în anul precedent** va fi completată automat din **Raportare DEEE**, ca sumă a coloanelor **Echipamente produse proprii** și **Echipamente importate** pentru fiecare categorie. Backend-ul va expune un serviciu de agregare anuală, iar în frontend aceste valori vor apărea read-only.
FRQ156	Coloana **Costul operațional de gestionare a deșeurilor** va fi preluată automat din modulul **Costul operațional de gestionare a deșeurilor**, pentru categoria DEEE. Vom păstra aceste valori într-o tabelă anuală dedicată, de exemplu operational_costs, versionată pe an și categorie, astfel încât fiecare calcul să folosească exact costurile aprobate pentru anul de gestiune.
FRQ157	Câmpul **Balanța contului** va fi calculat automat în backend conform formulei **GT = GE + GV - GC**. Vom implementa formula într-un engine dedicat, unde GE este calculul pentru anul curent, GV reprezintă diferența dintre cantitatea efectiv introdusă și cea estimată pentru anul precedent, iar GC reprezintă garanția aferentă cantităților colectate și gestionate în anul precedent. În UI, balanța va fi recalculată după fiecare modificare relevantă, dar valoarea finală oficială va fi returnată de backend.
FRQ158	Sistemul va permite și **valori negative** pentru câmpul **Balanța contului**. Tehnic, toate câmpurile de calcul vor fi stocate în tipuri numerice semnate (NUMERIC/DECIMAL), iar în frontend valorile negative vor fi afișate explicit, eventual cu evidențiere vizuală. Dacă balanța este negativă, UI-ul poate afișa și un mesaj contextual că suma respectivă poate deveni eligibilă pentru retragere în fluxul următor.
FRQ159	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea confirmării. În frontend vom folosi form.validateFields() și Form.useWatch() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ160	Toate câmpurile formularului vor fi tratate ca obligatorii. În frontend vom defini reguli de validare pentru toate controalele, iar în backend DTO-ul de submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelele de calcul. În DB,

	câmpurile critice vor avea constrângeri NOT NULL, iar pentru tabelul anexă vom salva valorile într-o structură normalizată sau într-un JSONB validat.
FRQ161	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Cereri de calcul a garanției financiare pentru DEEE**, cu tabel server-side și acțiuni de workflow: **Acceptă cererea**, **Trimite spre corectare**, **Respinge/Invalidează**. Backend-ul va expune un endpoint unificat de tip /api/private/deee-financial-guarantees/{id}/actions, care va valida rolul, statusul curent și comentariul obligatoriu acolo unde este necesar.
FRQ162	Pentru fiecare cerere vom menține un istoric complet într-o tabelă deee_financial_guarantee_history, cu status vechi, status nou, actor, dată și comentariu. În frontend, istoricul va fi afișat prin Timeline sau Steps, astfel încât utilizatorul și inspectorul să poată urmări clar etapele: **Completare cerere**, **Verificare cerere**, **Trimis spre corectare**, **Cerere aprobată**, **Cerere respinsă**.
FRQ163	Utilizatorul va putea relua în editare cererea transmisă atât timp cât aceasta nu a fost preluată în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, cererea va putea reveni din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ164	Dacă cererea a fost deja preluată în procesare, editarea va fi blocată. Frontend-ul va dezactiva funcționalitatea de editare și va afișa mesajul contextual cerut, iar backend-ul va respinge orice încercare de modificare printr-un cod business dedicat, de exemplu REQUEST_IN_PROCESSING.
FRQ165	În interfața inspectorului, cererile de calcul a garanției financiare vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contori per categorie.
FRQ166	Fiecare folder va afișa numărul de cereri și un tabel server-side, cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID cerere**, **numele companiei**, **data depunerii**, **persoana responsabilă**, **etapa cererii**. În PostgreSQL vom indexa status, company_id, submitted_at, created_at pentru performanță bună la filtrare și sortare.

6.9. Cerere de retragere a garanției financiare pentru DEEE

Cerință	Implementare propusă
FRQ167	Accesul la modulul **Cerere de retragere a garanției financiare pentru DEEE** va fi permis doar utilizatorilor autentificați prin **MPASS** care îndeplinesc simultan patru condiții: au companie activă în Cabinetul personal, au înregistrare activă în **Lista Producătorilor** pe categoria **DEEE**, au **Raportarea DEEE pentru anul precedent** aprobată și au **Calculul garanției financiare pentru anul în curs** aprobat. În backend vom implementa un EligibilityService care verifică toate aceste precondiții înainte de accesul la endpoint-urile /api/private/deee-guarantee-withdrawals/**. În frontend,

	dacă una dintre condiții nu este îndeplinită, utilizatorul va primi un mesaj contextual și link direct către modulul care trebuie completat înainte.
FRQ168	Pentru fiecare companie înregistrată în Cabinetul personal se va completa un formular separat. În modelul de date vom introduce tabela deee_guarantee_withdrawal_requests, cu relație many-to-one către companies, plus câmpuri precum reason_type, current_guarantee_value, return_amount, remaining_guarantee_value, status, submitted_at, picked_at, picked_by. În UI, compania va fi selectată explicit din lista companiilor utilizatorului, iar fiecare cerere va avea propriul istoric și propriul flux de aprobare.
FRQ169	Formularul pentru anul în curs va putea fi deschis și transmis doar după ce există atât **Raportarea DEEE pentru anul precedent**, cât și **Calculul garanției financiare pentru anul curent**. În backend, această regulă va fi verificată atât la încărcarea formularului, cât și la submit, pentru a evita salvarea unor cereri neeligibile.
FRQ170	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va conține: **Numele companiei**, **Motivul solicitării**, **Tabelul Retragerea garanției financiare**, **Cantitatea de EEE pentru care a trecut termenul de garantare**, **Valoarea garanției calculate pentru anul de gestiune**, **Suma ce urmează a fi returnată**, **Valoarea garanției rămase**, **Atașează fișiere**, **Persoana autorizată** și checkbox-ul de confirmare. Tabelul principal va fi randat pe baza unei configurații backend corespunzătoare anexei 3, folosind Ant Design Table cu rânduri și coloane configurabile. **Valoarea garanției calculate pentru anul de gestiune** va fi preluată automat din modulul **Calculul garanției financiare**, iar **Suma ce urmează a fi returnată** și **Valoarea garanției rămase** vor fi calculate automat în backend. Formula standard va fi implementată într-un serviciu dedicat de calcul. Atașamentele vor fi încărcate prin Upload, salvate în object storage și jurnalizate în tabela de metadate. **Persoana autorizată** va fi preluată automat din profilul utilizatorului și/sau din compania selectată.
FRQ171	Dacă utilizatorul selectează motivul **încetarea activității**, frontend-ul va afișa toate câmpurile formularului, iar acestea vor fi disponibile pentru completare și recalcul. Vom implementa logica prin afișare condiționată pe baza valorii câmpului reasonType, astfel încât UI-ul să poată comuta instant între modurile de formular.
FRQ172	Pentru motivul **încetarea activității**, câmpul **Cantitatea de EEE rămasă negestionată la moment** va fi calculat automat ca diferență dintre **Cantitatea de EEE efectiv introdusă pe piața națională în anul curent până la încetarea activității** și **Cantitatea de EEE colectate/reciclate până la moment**. Vom implementa formula în backend într-un GuaranteeWithdrawalCalculationService, iar frontend-ul va afișa valoarea în regim read-only. Datele sursă vor fi agregate din raportările și calculele existente în sistem.
FRQ173	Dacă utilizatorul selectează motivul **pentru anul precedent de activitate**, sistemul nu va cere completarea altor câmpuri de calcul și va afișa doar câmpul **Suma care urmează a fi returnată**, completat automat din câmpul **Balanța contului** din modulul **Calculul garanției financiare**. În backend vom expune un serviciu de

	preluare a valorii finale aprobate din anul precedent, astfel încât utilizatorul să nu introducă manual această sumă.
FRQ174	Dacă utilizatorul selectează motivul **sfârșitul perioadei de garantare**, sistemul va afișa doar câmpurile necesare pentru acest scenariu: **Cantitatea de EEE pentru care a trecut termenul de garantare**, **Valoarea garanției calculate pentru anul de gestiune**, **Suma care urmează a fi returnată**, **Valoarea garanției rămase**, **Atașează fișiere**, **Persoana autorizată**. Formula de calcul pentru **Suma care urmează a fi returnată** va fi implementată în backend ca **GR = GE - (NGi × Ci)**, unde GE este valoarea garanției aprobate, NGi este numărul/cantitatea de EEE pentru care a trecut termenul de garantare, iar Ci este costul operațional de gestionare a deșeurilor. Frontend-ul va afișa rezultatul în timp real, dar valoarea oficială va rămâne cea calculată de backend.
FRQ175	După aprobarea cererii, sistemul va actualiza automat câmpul **Suma calculată** din modulul **Calculul garanției financiare** cu valoarea din câmpul **Valoarea garanției rămase**. Tehnic, la aprobarea finală vom executa o actualizare tranzacțională asupra înregistrării active din deee_financial_guarantee_requests, astfel încât noua valoare rămasă să devină baza oficială pentru perioadele următoare. Vom jurnaliza această modificare într-un audit log separat, pentru trasabilitate financiară.
FRQ176	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea confirmării. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ177	Toate câmpurile formularului vor fi tratate ca obligatorii, cu excepția câmpului **Atașează fișiere**. În frontend vom defini reguli explicite pentru toate controalele, iar în backend DTO-ul de submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelul de calcul. În PostgreSQL vom marca drept obligatorii câmpurile-cheie precum company_id, reason_type, current_guarantee_value, return_amount, remaining_guarantee_value, authorized_person.
FRQ178	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Cereri retragere a garanției financiare pentru DEEE**. Inspectorul va putea efectua acțiunile **Acceptă cererea**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de workflow de tip /api/private/deee-guarantee-withdrawals/{id}/actions, cu validare de rol, status și comentariu obligatoriu pentru corectare sau respingere. În cazul acțiunii **Acceptă**, cererea va fi mutată în starea PENDING_SIGNATURE sau echivalentul operațional stabilit, pentru a reflecta faptul că ea trece în etapa **Cereri trimise spre semnare**.
FRQ179	Pentru fiecare cerere vom păstra un istoric complet într-o tabelă deee_guarantee_withdrawal_history, cu câmpuri precum from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare cerere**, **Verificare cerere**, **Trimis spre corectare**, **Cerere aprobată**, **Cerere respinsă**.

FRQ180	Utilizatorul va putea relua în editare cererea transmisă atât timp cât aceasta nu a fost preluată în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, cererea va putea reveni din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ181	Dacă cererea a fost deja accesată de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva acțiunea de editare și va afișa mesajul contextual cerut, iar backend-ul va respinge orice update printr-un cod business dedicat, de exemplu REQUEST_IN_PROCESSING.
FRQ182	În interfața inspectorului și a conducerii Agenției de Mediu, cererile de retragere a garanției financiare vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contori per categorie. Folderele vor corespunde etapelor: **Cereri trimise spre verificare**, **Cereri trimise spre corectare**, **Cerere aprobată**, **Cerere respinsă**.
FRQ183	Fiecare folder va afișa numărul de cereri și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID cerere**, **numele companiei**, **motivul solicitării**, **data depunerii**, **persoana responsabilă**, **etapa cererii**. În PostgreSQL vom indexa status, company_id, reason_type, submitted_at, created_at pentru performanță bună la filtrare și sortare.

6.10. Planul operațional

Cerință	Implementare propusă
FRQ184	Accesul la modulul **Planul operațional** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și cel puțin o înregistrare activă în **Lista Producătorilor**. În backend vom implementa un EligibilityService care verifică aceste condiții înainte de accesul la endpoint-urile /api/private/operational-plans/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat înainte.
FRQ185	Fiecare **Plan operațional** va fi asociat unei singure companii. Vom introduce tabela operational_plans cu relație many-to-one către companies, plus atribute precum rep_category, responsibility_type, status, submitted_at, picked_at, picked_by, authorized_person_id. Dacă utilizatorul gestionează mai multe companii, în UI va selecta explicit compania din listă, iar pentru fiecare companie și categorie REP se va crea un document separat.
FRQ186	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va include: **Numele companiei**, **Tipul de responsabilitate REP**, **Categoria REP**, **Atașare Plan operațional**, **Persoana autorizată** și checkbox-ul de confirmare. **Numele companiei** va fi un Select alimentat exclusiv din companiile utilizatorului. **Tipul de**

	responsabilitate REP va fi implementat ca Radio.Group sau Select cu valorile sistem individual și sistem colectiv. **Categoria REP** va fi un Select obligatoriu, alimentat din nomenclatorul intern: VSU, DBA, UU, AU, DEEE, ADA, DPT. **Atașare Plan operațional** va folosi componenta Upload, cu stocare în object storage și metadate în DB. **Persoana autorizată** va fi completată automat din profilul utilizatorului și/sau din compania selectată. Pentru fiecare combinație companie + categorie REP vom genera un plan separat, fără agregarea mai multor categorii în același document.
FRQ187	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea câmpului de confirmare. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ188	Toate câmpurile formularului vor fi tratate ca obligatorii. În frontend vom defini reguli de validare pentru fiecare control, iar în backend DTO-ul de creare/submit va avea validări @NotNull, @NotBlank și validatori custom pentru fișierul atașat. În PostgreSQL, câmpurile critice vor avea constrângeri NOT NULL, iar pentru atașament vom valida existența unui document de tip OPERATIONAL_PLAN.
FRQ189	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Planul operațional**, cu tabel server-side și acțiuni de workflow: **Acceptă planul**, **Trimite spre corectare**, **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/operational-plans/{id}/actions, care va valida rolul utilizatorului, statusul curent și comentariul obligatoriu la corectare sau respingere. Pentru acțiunea **Acceptă planul**, sistemul va muta cererea în PENDING_SIGNATURE sau echivalentul operațional pentru etapa **trimise spre semnare**, astfel încât fluxul ulterior să poată fi gestionat prin mecanismul standard de semnare cu **MSIGN**, dacă acest pas este inclus în implementarea finală a workflow-ului.
FRQ190	Pentru fiecare formular vom păstra un istoric complet într-o tabelă operational_plan_history, cu câmpuri precum plan_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare Plan operațional**, **Verificare Plan operațional**, **Trimis spre corectare**, **Plan operațional aprobat**, **Plan operațional respins**.
FRQ191	Utilizatorul va putea relua în editare formularul transmis atât timp cât acesta nu a fost preluat în procesare de inspector. În backend vom utiliza câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ192	Dacă formularul a fost deja accesat de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva acțiunea de editare și va afișa mesajul contextual cerut. Backend-ul va respinge orice încercare de actualizare printr-un cod business dedicat, de exemplu PLAN_IN_PROCESSING, pentru a evita modificările concurente.

FRQ193	În interfața inspectorului, planurile operaționale vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per categorie. Folderele vor corespunde etapelor: **Planuri operaționale trimise spre verificare**, **Planuri operaționale trimise spre corectare**, **Planuri operaționale aprobate**, **Planuri operaționale respinse**.
FRQ194	Fiecare folder va afișa numărul de formulare și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID unic cerere**, **numele companiei**, **data depunerii cererii**, **categoria REP**, **persoana responsabilă**, **etapa cererii**. În PostgreSQL vom indexa status, company_id, rep_category, submitted_at, created_at pentru performanță bună la filtrare și sortare.
FRQ195	După ce **Planul operațional** trece în starea **aprobat**, sistemul va publica automat documentul în pagina de detaliu a agentului economic din **Lista Producătorilor**, într-un tab dedicat de tip **Planuri operaționale**. În backend vom salva fiecare plan aprobat în tabela operational_plan_documents, legată de entitatea din registrul producătorilor, cu metadate precum company_id, rep_category, approved_at, file_id, version_no, is_active. În frontend, la accesarea agentului economic, utilizatorii autorizați vor vedea lista ultimelor planuri aprobate și vor putea deschide sau descărca documentul. Pentru regula de retenție, vom implementa un serviciu backend care, la aprobarea unui nou plan, verifică numărul de documente aprobate existente pentru aceeași entitate și păstrează doar **ultimele 3** după approved_at; dacă limita este depășită, documentul cel mai vechi va fi eliminat automat în ordinea **FIFO**. Ștergerea va însemna atât marcarea logică în DB, cât și eliminarea fizică din object storage, după validarea că documentul nu mai este referit de alt obiect. Pentru audit și trasabilitate, fiecare operațiune de publicare și eliminare va fi jurnalizată într-o tabelă document_retention_audit

6.11. Costul operațional de gestionare a deșeurilor

Cerință	Implementare propusă
FRQ196	**Eligibilitate (doar sisteme colective):** în FE, ruta /cost-operational este protejată (MPASS + hasActiveCompany). În BE, endpoint-urile /api/private/operational-cost/** verifică: (1) compania există în Cabinet (asociere user–companie activă) și (2) compania are statut **Sistem colectiv** (din registrul sistemelor colective aprobate). Dacă nu, răspuns business 403 + motiv (“nu aveți statut de sistem colectiv”).
FRQ197	**Raport 1x/an + editare până la 25 decembrie:** DB: operational_cost_reports(id, company_id, rep_category, year, status, version, submitted_at, picked_at, picked_by, approved_at, ...) cu UNIQUE(company_id, rep_category, year, version) și regulă business: există max 1 raport “activ”/“curent” pe (company, category, year). UI: listă rapoarte pe an (implicit anul curent). Submit permis conform regulilor; după validare (aprobare) raportul rămâne referința oficială, dar până la **25 decembrie**

	poate intra pe “Revizuire” (vezi FRQ208). Deadline-ul de completare (20 noiembrie) îl tratăm ca **regulă configurabilă** (config server) pentru atenționări și, dacă se cere, blocare la submit după termen.
FRQ198	**Formular (React + AntD Form):** (1) Select companie (doar din Cabinet), (2) Select categorie REP (VSU, DBA/DAB, UU, AU, DEEE, ADA, DPT) → **1 raport per categorie**; (3) tabel editabil AntD Table “Costul operațional…” cu rânduri generate din nomenclator rep_waste_types (seed din anexe) și coloană cost_lei_per_ton (editable numeric); (4) persoana autorizată autocompletată (GET /api/private/profile/me + date companie), read-only; (5) atașamente opționale (AntD Upload) cu stocare în object storage + metadate în attachments; (6) checkbox confirmare obligatoriu.
FRQ199	**Buton “Trimite spre verificare” cu gating:** FE ține butonul disabled până trec form.validateFields() și checkbox confirmare e bifat. BE: POST /api/private/operational-cost/{id}/submit revalidează complet (DTO + reguli) și schimbă status în IN_REVIEW.
FRQ200	**Validări obligatorii:** (1) companie selectată, (2) categorie REP selectată, (3) **cel puțin un cost completat** în tabel (minim un rând cu cost > 0 sau cost != null înainte de normalizare), (4) persoana autorizată completată automat, (5) checkbox confirmare. Implementare: FE rules + BE Bean Validation + validator custom pentru “minim un cost introdus”.
FRQ201	**Celule necompletate → inactive + 0:** FE: la submit (și la autosave) normalizăm null/undefined/"" în 0 și setăm flag is_active=false pentru rândurile necompletate; BE repetă normalizarea pentru consistență. DB: operational_cost_report_items(report_id, waste_type_code, cost_lei_per_ton, is_active) (când e 0 din “necompletat”, is_active=false).
FRQ202	**Acțiuni Inspector AM (workflow):** ecran inspector cu detalii raport + acțiuni: VALIDATE, SEND_BACK, REJECT. BE: POST /api/private/operational-cost/{id}/actions cu actionType + comment obligatoriu la SEND_BACK/REJECT. La VALIDATE setăm approved_at, status APPROVED și marcăm valorile “definitive” pentru calcule.
FRQ203	**Calcul cost mediu (Inspector):** BE: modul OperationalCostAggregationService care calculează agregări din rapoarte APPROVED pe **an + categorie** (și, unde e relevant, pe tip de deșeu). UI inspector: pagină “Calculează medii” cu filtre (an, categorie) și tabel cu rezultate (ex. medie, mediană, min/max, număr rapoarte). Acțiunea poate fi rulată oricând; pentru performanță, putem salva rezultat în operational_cost_aggregates(year, rep_category, waste_type_code, avg_cost, computed_at, ...) și recalcula la cerere.
FRQ204	**Istoric etape (audit):** DB: operational_cost_report_history(id, report_id, from_status, to_status, actor_id, comment, created_at). FE: Timeline/Steps cu etape: Completare → Verificare → Trimis spre corectare → Acceptat → Respins, cu date și comentarii.
FRQ205	**Foldere pe status (Inspector):** FE inspector: Tabs/meniu lateral (“Rapoarte în verificare”, “Trimise spre corectare”, “Acceptate”, “Respinse”) cu badge count. BE:

	GET /api/private/operational-cost?status=...&page&size&sort + GET /api/private/operational-cost/counts.
FRQ206	**Editare permisă până la preluare:** BE: dacă status = IN_REVIEW și picked_at este null → POST /api/private/operational-cost/{id}/back-to-edit mută în DRAFT și scrie în history. FE: buton "Editare" vizibil doar când API întoarce editable=true.
FRQ207	**Blocare editare după preluare:** la primul acces "de lucru" al inspectorului setăm picked_by/picked_at. Din acel moment, orice update/back-to-edit este refuzat cu 409 + mesajul cerut; FE dezactivează editarea și afișează exact mesajul contextual.
FRQ208	**Revizuire după aprobare (până la 25 decembrie):** FE: buton "Revizuire" disponibil doar când status = APPROVED și data curentă ≤ 25 decembrie (config server). BE: POST /api/private/operational-cost/{id}/revise creează o **versiune nouă** (copy snapshot din items + increment version), status DRAFT, păstrează istoricul și permite resubmit. După 25 decembrie, endpoint-ul returnează eroare business ("raport definitiv").
FRQ209	**Listare/număr rapoarte + paginare 50 + sort desc:** FE: fiecare folder afișează count + tabel AntD Table cu paginare server-side size=50 default și sort created_at/ submitted_at desc. Coloane: ID raport, companie, categorie REP, data depunerii, persoană responsabilă, etapa (status). DB indexare: status, submitted_at, company_id, rep_category, year pentru performanță.
FRQ210	**Interfață Admin pentru editarea categoriilor + valorilor unitare:** vom implementa un modul Admin (ReactJS + Ant Design) cu două zone: (1) **Management categorii / tipuri de deșeuri** (adăugare/ștergere/activare-dezactivare rânduri din tabel) și (2) **Editare valori unitare (lei/tonă)** pe fiecare categorie REP și tip de deșeu. În backend (Java 25 + Spring Boot 4.0.3) expunem endpoint-uri /api/admin/operational-cost/catalog/** și /api/admin/operational-cost/reference-values/** pentru CRUD, cu validări business (ex.: nu poți șterge un tip de deșeu dacă e referit de rapoarte istorice; se face **soft-delete** + active=false). Persistență recomandată: rep_waste_types (nomenclator) + operational_cost_reference_values (valori unitare, versionate pe an/"effectiveFrom"), plus **audit** (*_audit_log) pentru trasabilitate (cine/când/ce a modificat).
FRQ211	**Acces doar după autentificare ca Administrator:** ruta Admin în frontend va fi protejată prin guard de rol (ROLE_ADMIN) și nu va fi afișată în meniu pentru utilizatorii non-admin. În backend, Spring Security va proteja toate endpoint-urile /api/admin/** cu hasRole('ADMIN'), iar acțiunile de modificare (POST/PUT/DELETE) vor cere token MPASS valid + vor fi logate (audit + IP + userId). Opțional: restricție suplimentară (allowlist IP/VPN) pentru zona Admin.

6.12. Raportare UU

Cerință	Implementare propusă
FRQ212	Accesul la modulul **Raportare UU** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și o înregistrare activă în **Lista Producătorilor** pentru categoria **UU**. În backend vom implementa un

	EligibilityService care verifică aceste condiții înainte de accesul la endpoint-urile /api/private/uu-reports/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat mai întâi.
FRQ213	Sistemul va permite gestionarea mai multor companii per utilizator, dar fiecare raport UU va fi completat și transmis **separat pentru fiecare companie**. În modelul de date vom introduce tabela uu_reports, legată de companies, cu câmpuri precum company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by. În UI, compania va fi selectată explicit, iar fiecare raport va avea propriul istoric și propriul flux de aprobare.
FRQ214	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va include: **Numele companiei**, **Numărul de înregistrare în Lista Producătorilor**, **Localitatea**, **Adresa**, **Telefon**, **E-mail**, **Genurile principale de activitate**, **Numărul mediu de angajați**, **Anul de raportare**, **Atașare Raport narativ**, **Atașare fișiere**, **Persoana autorizată**, **Funcția persoanei autorizate**, plus tabelul **Cantitatea de uleiuri introduse pe piață cât și cele colectate** și checkbox-ul de confirmare. Pentru **sistem individual**, câmpul **Numele companiei** va fi selectat exclusiv din companiile din Cabinetul personal. Pentru **sistem colectiv**, acesta va fi selectat din lista companiilor afiliate acelui sistem colectiv. Câmpurile **Numărul de înregistrare**, **Localitatea**, **Adresa**, **Telefonul**, **E-mail-ul** și **Genurile principale de activitate** vor fi autocompletate după selectarea companiei, pe baza datelor deja existente în profil și în registru. **Raportul narativ** va fi încărcat prin Upload, cu stocare în object storage și metadate în DB. Tabelul principal va fi implementat prin Ant Design Table cu rânduri dinamice, pe baza categoriilor din anexa aplicabilă, și coloane numerice validate server-side.
FRQ215	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea checkbox-ului de confirmare. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ216	Toate câmpurile formularului vor fi tratate ca obligatorii, cu excepția câmpului **Atașare fișiere**. În frontend vom defini reguli de validare explicite pentru toate controalele, iar în backend DTO-ul de creare/submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelul de raportare și pentru raportul narativ. În PostgreSQL vom aplica constrângeri NOT NULL pentru câmpurile-cheie și vom valida existența atașamentului de tip NARRATIVE_REPORT.
FRQ217	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Raportare UU**, cu tabel server-side și acțiuni de workflow: **Validează raportul**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/uu-reports/{id}/actions, care va valida rolul, statusul curent și comentariul obligatoriu pentru corectare sau respingere.

FRQ218	Pentru fiecare raport vom păstra un istoric complet într-o tabelă uu_report_history, cu câmpuri precum report_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare raport**, **Verificare raport**, **Trimis spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Raport aprobat**, **Raport respins**.
FRQ219	Utilizatorul va putea relua în editare raportul transmis atât timp cât acesta nu a fost preluat în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ220	Dacă raportul a fost deja accesat de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva funcționalitatea de editare și va afișa mesajul contextual cerut, iar backend-ul va respinge orice actualizare printr-un cod business dedicat, de exemplu REPORT_IN_PROCESSING.
FRQ221	În interfața inspectorului, rapoartele UU vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per categorie. Folderele vor corespunde etapelor: **Rapoarte trimise spre verificare**, **Rapoarte trimise spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Rapoarte aprobate**, **Rapoarte respinse**.
FRQ222	Fiecare folder va afișa numărul de rapoarte și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID raport**, **numele companiei**, **data depunerii**, **anul de raportare**, **persoana responsabilă**, **numărul de înregistrare**, **etapa raportului**. În PostgreSQL vom indexa status, company_id, reporting_year, submitted_at, created_at, registration_number pentru performanță bună la filtrare și sortare.
FRQ223	Dacă inspectorul Agenției de Mediu are suspiciuni privind corectitudinea datelor, raportul va fi mutat în etapa **Verificare IPM**. În acel moment, sistemul va genera o notificare către rolul **Inspector IPM** și va expune raportul în interfața IPM. În UI-ul IPM vom adăuga o secțiune specială pentru atașarea documentelor de control, în special **actul de inspecție**, folosind componenta Upload. Fișierele vor fi stocate în object storage, iar metadatele în tabela report_attachments. După încărcarea documentelor, inspectorul IPM va putea muta raportul în etapa **Luarea deciziei**, printr-un endpoint dedicat de workflow.
FRQ224	Când raportul ajunge în etapa **Luarea deciziei**, inspectorul AM va putea alege una dintre acțiunile: **aprobă raportul**, **trimite spre verificare** sau **respinge raportul**. Tehnic, aceasta va fi implementată prin același mecanism central de workflow, cu validări suplimentare pe statusul curent. Acțiunea **trimite spre verificare** va readuce raportul într-o etapă intermediară de reverificare, iar **respingerea** va necesita justificare obligatorie. Toate deciziile vor fi jurnalizate în istoric și vor genera notificări interne și, opțional, email-uri către companie.

6.13. Raportare VSU

Cerință	Implementare propusă
FRQ225	Accesul la modulul **Raportare VSU** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și o înregistrare activă în **Lista Producătorilor** pentru categoria **VSU**. În backend vom implementa un EligibilityService care verifică aceste condiții înainte de accesul la endpoint-urile /api/private/vsu-reports/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat mai întâi.
FRQ226	Sistemul va permite gestionarea mai multor companii per utilizator, dar fiecare raport **VSU** va fi completat și transmis **separat pentru fiecare companie**. În modelul de date vom introduce tabela vsu_reports, legată de companies, cu câmpuri precum company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by. Vom aplica și o regulă de unicitate business pentru combinația company_id + reporting_year + rep_category, astfel încât să nu existe dubluri active pentru același an.
FRQ227	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va include: **Numele companiei**, **Numărul de înregistrare în Lista Producătorilor**, **Localitatea**, **Adresa**, **Telefon**, **E-mail**, **Genurile principale de activitate**, **Numărul mediu de angajați**, **Anul de raportare**, **Atașare Raport narativ**, **Atașare fișiere**, **Persoana autorizată**, **Funcția persoanei autorizate**, plus **Tabelul nr. 1**, **Tabelul nr. 2**, **Tabelul nr. 3** și **Tabelul nr. 4**. Pentru **sistem individual**, câmpul **Numele companiei** va fi selectat exclusiv din companiile din Cabinetul personal. Pentru **sistem colectiv**, acesta va fi selectat din lista companiilor afiliate acelui sistem colectiv, pe baza relațiilor deja definite în cererea de înregistrare sau notificarea de trecere la sistem colectiv. Câmpurile **Numărul de înregistrare**, **Localitatea**, **Adresa**, **Telefonul**, **E-mail-ul** și **Genurile principale de activitate** vor fi autocompletate după selectarea companiei, pe baza datelor din profil și registru. **Raportul narativ** va fi încărcat prin Upload, cu stocare în object storage și metadate în DB. Cele patru tabele vor fi generate dinamic pe baza unui șablon/configurator derivat din anexa relevantă și vor fi afișate pe **aceeași pagină**, sub formă de secțiuni expandabile. În UI vom implementa o listă de bifare (Checkbox.Group) pentru selectarea tabelelor ce urmează a fi completate, iar secțiunile corespunzătoare vor fi expandate condiționat. Datele tabelare vor fi persistate fie în tabele child dedicate (vsu_report_table1_items, vsu_report_table2_items, etc.), fie într-un model generic report_sections + report_rows dacă vrem o arhitectură reutilizabilă și pentru celelalte categorii REP.
FRQ228	Coloanele din tabele care conțin formule vor fi calculate automat de către sistem. Vom implementa un serviciu backend dedicat, de tip VsuReportCalculationService, care va recalcula valorile derivate la salvare draft, la submit și la orice actualizare de rând. În frontend, celulele calculate vor fi afișate read-only, iar utilizatorul va edita

	doar câmpurile sursă. În acest mod evităm discrepanțele dintre valorile vizibile în UI și cele persistate în sistem.
FRQ229	Pentru **Tabelul nr. 4** vom introduce explicit prima coloană **Categorii de deșeuri**, alimentată dintr-un nomenclator intern versionat, astfel încât ordinea și valorile să fie consistente în toate rapoartele. În frontend, această coloană va fi read-only și va fi randată ca prima coloană fixată (fixed: 'left') pentru lizibilitate mai bună. În backend, valorile vor fi mapate prin coduri standardizate, nu doar text liber.
FRQ230	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea checkbox-ului de confirmare. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ231	Toate câmpurile formularului vor fi tratate ca obligatorii, cu excepția câmpului **Atașare fișiere**. În frontend vom defini reguli de validare explicite pentru toate controalele, iar în backend DTO-ul de creare/submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelele raportului și pentru raportul narativ. În PostgreSQL vom aplica constrângeri NOT NULL pentru câmpurile-cheie și vom valida existența atașamentului de tip NARRATIVE_REPORT.
FRQ232	Vor avea caracter obligatoriu doar tabelele care au fost **bifate** din listă. Dacă un tabel nu este bifat, acesta nu intră în validare și nu este obligatoriu. Dacă un tabel este bifat, atunci sistemul va impune regula ca **cel puțin un rând** să fie completat. În frontend vom implementa această regulă prin validatori dinamici pe secțiune, iar în backend printr-un validator custom care verifică prezența unui minim de date nenule în rândurile tabelului selectat.
FRQ233	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Raportare VSU**, cu tabel server-side și acțiuni de workflow: **Validează raportul**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/vsu-reports/{id}/actions, care va valida rolul, statusul curent și comentariul obligatoriu pentru corectare sau respingere.
FRQ234	Pentru fiecare raport vom păstra un istoric complet într-o tabelă vsu_report_history, cu câmpuri precum report_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare raport**, **Verificare raport**, **Trimis spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Raport aprobat**, **Raport respins**.
FRQ235	Utilizatorul va putea relua în editare raportul transmis atât timp cât acesta nu a fost preluat în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ236	Dacă raportul a fost deja accesat de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva funcționalitatea de editare și va afișa mesajul contextual cerut, iar backend-ul va respinge orice actualizare printr-un cod business

	dedicat, de exemplu REPORT_IN_PROCESSING. Pentru a evita concurența, la primul acces efectiv în procesare vom seta picked_by și picked_at într-o tranzacție atomică.
FRQ237	În interfața inspectorului, rapoartele VSU vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per categorie. Folderele vor corespunde etapelor: **Rapoarte trimise spre verificare**, **Rapoarte trimise spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Rapoarte aprobate**, **Rapoarte respinse**.
FRQ238	Fiecare folder va afișa numărul de rapoarte și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID raport**, **numele companiei**, **data depunerii**, **anul de raportare**, **persoana responsabilă**, **numărul de înregistrare**, **etapa raportului**. În PostgreSQL vom indexa status, company_id, reporting_year, submitted_at, created_at, registration_number pentru performanță bună la filtrare și sortare.
FRQ239	Dacă inspectorul Agenției de Mediu are suspiciuni privind corectitudinea datelor, raportul va fi mutat în etapa **Verificare IPM**. În acel moment, sistemul va genera o notificare către rolul **Inspector IPM** și va expune raportul în interfața IPM. În UI-ul IPM vom adăuga o secțiune specială pentru atașarea documentelor de control, în special **actul de inspecție**, folosind componenta Upload. Fișierele vor fi stocate în object storage, iar metadatele în tabela report_attachments. După încărcarea documentelor, inspectorul IPM va putea muta raportul în etapa **Luarea deciziei**, printr-un endpoint dedicat de workflow.
FRQ240	Când raportul ajunge în etapa **Luarea deciziei**, inspectorul AM va putea alege una dintre acțiunile: **aprobă raportul**, **trimite spre verificare** sau **respinge raportul**. Tehnic, aceasta va fi implementată prin același mecanism central de workflow, cu validări suplimentare pe statusul curent. Acțiunea **trimite spre verificare** va readuce raportul într-o etapă intermediară de reverificare, iar **respingerea** va necesita justificare obligatorie. Toate deciziile vor fi jurnalizate în istoric și vor genera notificări interne și, opțional, email-uri către companie.

6.14. Raportare DBA

Cerință	Implementare propusă
FRQ241	**Eligibilitate / acces (MPASS + prerechizite):** în FE (React Router) ruta /raportare-dba este protejată (MPASS + hasActiveCompany + hasProducerRegistryEntry(DBA)). În BE, toate endpoint-urile /api/private/dba-reports/** trec prin EligibilityService care verifică: (1) companie activă în Cabinet, (2) înregistrare activă în Lista Producătorilor pe categoria **DBA**. Dacă nu e eligibil → 403 cu motiv business + link-uri/“next steps” returnate în payload.
FRQ242	**Multi-companie, raport separat:** DB: dba_reports(id, company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by, ...) cu regulă de unicitate business pentru company_id + reporting_year

	(și, dacă e cazul, pe combinații suplimentare). FE: selector companie + listă rapoarte per companie/an.
FRQ243	**Formular (ReactJS + AntD Form) + autocompletare:** FE construiește formularul cu câmpuri: companie (Select), nr. înregistrare (read-only, autocompletat), localitate/adresă/telefon/email/CAEN (autocompletate), nr. mediu angajați (InputNumber), anul de raportare (Select), raport narativ (Upload – obligatoriu), atașare fișiere (Upload – opțional), persoana autorizată (autocompletată), funcția (Input). **Sistem individual:** compania se alege doar din Cabinet. **Sistem colectiv:** compania se alege din lista companiilor afiliate sistemului colectiv (BE: GET /api/private/collective-systems/{id}/members). **Tabelul principal (Anexa 11):** AntD Table cu rânduri predefinite din nomenclator (seed în DB) + coloane numerice validate. Persistență recomandată: dba_report_items(report_id, category_code, introduced_qty, collected_qty, treated_qty, recycled_qty, disposed_qty, exported_qty, ...) (coloanele exacte după anexa).
FRQ244	**"Trimite spre verificare" cu gating:** FE ține butonul disabled până trec form.validateFields() și checkbox-ul de confirmare este bifat. BE: POST /api/private/dba-reports/{id}/submit revalidează DTO + regulile pe tabel + existența raportului narativ și schimbă status în IN_REVIEW, setând submitted_at.
FRQ245	**Obligatoriu peste tot, cu excepția "Atașare fișiere":** FE: rules required pe toate câmpurile, iar "Atașare fișiere" rămâne opțional. BE: Bean Validation + validator custom (ex.: cel puțin un rând relevant completat; cantitățile ≥ 0; consistență pe coloane). În DB: NOT NULL pe câmpurile cheie; atașamente opționale stocate în attachments + legături în dba_report_attachments.
FRQ246	**Acțiuni Inspector AM (workflow):** UI inspector (React + AntD) în "Monitorizare și raportare → Raportare REP → Raportare DBA" cu acțiuni: VALIDATE, SEND_BACK, REJECT. BE: POST /api/private/dba-reports/{id}/actions (rol ROLE_AM_INSPECTOR), cu comment obligatoriu pentru SEND_BACK/REJECT. La VALIDATE → status APPROVED, setare approved_at, notificare către companie (inbox + email opțional).
FRQ247	**Istoric etape (audit complet):** DB: dba_report_history(id, report_id, from_status, to_status, actor_id, comment, created_at); FE: Timeline/Steps cu etape: Completare → Verificare → Trimis spre corectare → Verificare IPM → Luarea deciziei → Aprobat → Respins.
FRQ248	**Editare permisă până la preluare:** BE: dacă status IN_REVIEW și picked_at este null → POST /api/private/dba-reports/{id}/back-to-edit mută în DRAFT și scrie în history; FE: buton "Editare" vizibil doar când API întoarce editable=true.
FRQ249	**Blocare editare după preluare:** la primul "open for processing" inspectorul setează atomic picked_by/picked_at (optimistic lock/version). De atunci, orice update/back-to-edit este respins (409) cu mesaj business; FE dezactivează editarea și afișează mesajul contextual.
FRQ250	**Foldere pe status (Inspector):** FE inspector: Tabs/meniu lateral ("trimise spre verificare", "trimise spre corectare", "verificare IPM", "luarea deciziei", "aprobate",

	"respinse") + badge counts. BE: GET /api/private/dba-reports?status=...&page&size&sort + GET /api/private/dba-reports/counts.
FRQ251	**Listare în foldere: max 50/pagină + sort desc + coloane standard:** BE: paginare server-side size=50 default, sort created_at desc/submitted_at desc. FE: AntD Table cu coloane: ID raport, companie, data depunerii, anul de raportare, persoană responsabilă, număr de înregistrare, etapa (status). Indexare DB: status, company_id, reporting_year, submitted_at, created_at, registration_number.
FRQ252	**Escaladare la IPM + încărcare act inspecție:** dacă inspectorul AM are suspiciuni → acțiune SEND_TO_IPM care mută raportul în IPM_REVIEW și generează notificare către ROLE_IPM_INSPECTOR. UI IPM: câmp nou Upload pentru **actul de inspecție**; fișierul se stochează în object storage (S3/MinIO) + metadate în attachments, legate de raport. După upload, IPM execută acțiunea SEND_TO_DECISION (status DECISION).
FRQ253	**Luarea deciziei (Inspector AM):** când status = DECISION, inspectorul AM poate APPROVE, SEND_BACK_TO_REVIEW (înapoi la verificare) sau REJECT (justificare obligatorie). Implementare prin același endpoint de acțiuni, cu reguli stricte pe status + history + notificări către companie.

6.15. Raportare AU

Cerință	Implementare propusă
FRQ254	**Eligibilitate / acces:** în FE (React Router) ruta /raportare-au este protejată (MPASS + hasActiveCompany + hasProducerRegistryEntry(AU)). În BE, endpoint-urile /api/private/au-reports/** trec prin EligibilityService care verifică: (1) companie activă în Cabinet, (2) înregistrare activă în Lista Producătorilor pe categoria **AU**. Dacă nu → 403 cu motiv business.
FRQ255	**Multi-companie, raport separat:** DB: au_reports(id, company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by, ...) + regulă de unicitate business pentru (company_id, reporting_year) (și eventual (company_id, reporting_year, responsibility_type) dacă se cere). FE: selector companie + listă rapoarte per companie/an.
FRQ256	**Formular (React + AntD Form) + autocompletare + tabel anexă:** FE construiește formularul cu: companie (Select), nr. înregistrare (read-only, auto), localitate/adresă/telefon/email/CAEN (auto), nr. mediu angajați (InputNumber), anul raportării (Select), **Raport narativ (Upload – obligatoriu)**, atașare fișiere (Upload – opțional), persoana autorizată (auto), funcția (Input), **Tabel "Cantitatea de anvelope introduse pe piață, cât și cele colectate"** (AntD Table editabil). • **Sistem individual:** compania se alege doar din Cabinet. • **Sistem colectiv:** compania se alege din lista companiilor afiliate sistemului colectiv (BE: GET /api/private/collective-systems/{id}/members). • **Categorii (coloana "Categorii"):** Select alimentat din nomenclator (seed din Anexa 1) – stocat ca cod, nu text liber. Persistență: au_report_items(report_id, category_code, qty_introduced, qty_collected, qty_treated, qty_recycled, qty_disposed, qty_exported, ...) (coloane

	conform anexei). Atașamentele se stochează în object storage + metadate în attachments.
FRQ257	**“Trimite spre verificare” cu gating:** FE ține butonul disabled până trec form.validateFields() și checkbox-ul de confirmare este bifat. BE: POST /api/private/au-reports/{id}/submit revalidează DTO + regulile pe tabel + existența raportului narativ și schimbă status în IN_REVIEW, setând submitted_at.
FRQ258	**Obligatoriu peste tot, excepție “Atașare fișiere”:** FE: rules required pe toate câmpurile, iar “Atașare fișiere” rămâne opțional. BE: Bean Validation + validator custom (ex.: cel puțin un rând relevant completat; cantități ≥ 0; consistență pe coloane). În DB: NOT NULL pe câmpuri cheie; raportul narativ validat ca attachment obligatoriu de tip NARRATIVE_REPORT.
FRQ259	**Acțiuni Inspector AM (workflow):** UI inspector în “Monitorizare și raportare → Raportare REP → Raportare AU” cu acțiuni: VALIDATE, SEND_BACK, REJECT. BE: POST /api/private/au-reports/{id}/actions (rol ROLE_AM_INSPECTOR), cu comment obligatoriu pentru SEND_BACK/REJECT. La VALIDATE → status APPROVED, setare approved_at, notificare către companie (inbox + email opțional).
FRQ260	**Istoric etape (audit):** DB: au_report_history(id, report_id, from_status, to_status, actor_id, comment, created_at). FE: Timeline/Steps cu etape: Completare → Verificare → Trimis spre corectare → Verificare IPM → Luarea deciziei → Aprobat → Respins.
FRQ261	**Editare permisă până la preluare:** BE: dacă status IN_REVIEW și picked_at este null → POST /api/private/au-reports/{id}/back-to-edit mută în DRAFT și scrie în history; FE: buton “Editare” vizibil doar când API întoarce editable=true.
FRQ262	**Blocare editare după preluare:** la primul acces “de lucru” inspectorul setează atomic picked_by/picked_at (optimistic lock/version). De atunci, orice update/back-to-edit este respins (409) cu mesaj business; FE dezactivează editarea și afișează mesajul contextual.
FRQ263	**Foldere pe status (Inspector):** FE inspector: Tabs/meniu lateral (“trimise spre verificare”, “trimise spre corectare”, “verificare IPM”, “luarea deciziei”, “aprobate”, “respinse”) + badge counts. BE: GET /api/private/au-reports?status=...&page&size&sort + GET /api/private/au-reports/counts.
FRQ264	**Listare: max 50/pagină + sort desc + coloane standard:** BE: paginare server-side size=50 default, sort created_at desc/submitted_at desc. FE: AntD Table cu coloane: ID raport, companie, data depunerii, anul de raportare, persoană responsabilă, număr de înregistrare, etapa (status). Indexare DB: status, company_id, reporting_year, submitted_at, created_at, registration_number.
FRQ265	**Escaladare la IPM + act inspecție:** acțiune SEND_TO_IPM mută raportul în IPM_REVIEW și notifică ROLE_IPM_INSPECTOR (inbox + email opțional). UI IPM: câmp Upload pentru **actul de inspecție** + alte documente; stocare în object storage + metadate în attachments legate de raport. După upload, IPM execută acțiunea SEND_TO_DECISION (status DECISION).

FRQ266	**Luarea deciziei (Inspector AM):** când status = DECISION, inspectorul AM poate APPROVE, SEND_BACK_TO_REVIEW (înapoi la verificare) sau REJECT (justificare obligatorie). Implementare prin același endpoint de acțiuni, cu reguli stricte pe status + history + notificări către companie.

6.16. Raportare ADA

Cerință	Implementare propusă
FRQ267	Accesul la modulul **Raportare ADA** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și o înregistrare activă în **Lista Producătorilor** pentru categoria **ADA**. În backend vom implementa un EligibilityService care verifică aceste condiții înainte de accesul la endpoint-urile /api/private/ada-reports/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat mai întâi.
FRQ268	Sistemul va permite gestionarea mai multor companii per utilizator, dar fiecare raport **ADA** va fi completat și transmis **separat pentru fiecare companie**. În modelul de date vom introduce tabela ada_reports, legată de companies, cu câmpuri precum company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by. Vom aplica și o regulă de unicitate business pentru combinația company_id + reporting_year + rep_category, astfel încât să nu existe două rapoarte active pentru aceeași companie și același an de raportare.
FRQ269	Pentru accesarea formularului, în frontend vom crea ruta dedicată **/rapoarte/raportare-ada**, inclusă în meniul principal al zonei autentificate, în secțiunea **Rapoarte**. Navigarea va fi realizată cu **React Router**, iar încărcarea inițială a formularului va apela endpoint-uri pentru metadate, companiile disponibile și configurația tabelelor ADA.
FRQ270	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va include: **Numele companiei**, **Numărul de înregistrare în Lista Producătorilor**, **Localitatea**, **Adresa**, **Telefon**, **E-mail**, **Genurile principale de activitate**, **Numărul mediu de angajați**, **Anul de raportare**, **Atașare Raport narativ**, **Atașare fișiere**, **Persoana autorizată**, **Funcția persoanei autorizate**, plus **Tabelul nr. 1 – 7**. Pentru **sistem individual**, câmpul **Numele companiei** va fi selectat exclusiv din companiile din Cabinetul personal. Pentru **sistem colectiv**, acesta va fi selectat din lista companiilor afiliate acelui sistem colectiv, pe baza relațiilor deja înregistrate în sistem. Câmpurile **Numărul de înregistrare**, **Localitatea**, **Adresa**, **Telefonul**, **E-mail-ul** și **Genurile principale de activitate** vor fi autocompletate după selectarea companiei, pe baza datelor din profil și din registru. **Raportul narativ** va fi încărcat prin Upload, cu stocare în object storage și metadate în DB. Toate cele **7 tabele** vor fi construite pe baza unei configurații versionate derivate din Anexa nr. 8 din HG nr. 561/2020 și vor fi afișate pe **aceeași pagină**, sub formă de secțiuni expandabile. În UI vom implementa o listă de bifare (Checkbox.Group) pentru selectarea tabelelor ce urmează a fi completate; doar secțiunile bifate vor fi expandate și activate pentru

	editare. Persistența poate fi făcută printr-un model generic report_sections + report_rows + report_cells, pentru a permite flexibilitate pe structura celor 7 tabele și eventuale modificări viitoare fără migrarea masivă a codului.
FRQ271	În tabelele nr. 1–7, coloana **„Denumirea valorificatorului autorizat + IDNO”** va fi modelată în sistem ca **3 câmpuri distincte**: valorificator_idno, valorificator_name, destination_country_code. În frontend, acestea vor fi randate ca trei coloane separate, folosind Input, Input și Select. Pentru **Țara de destinație** vom folosi un nomenclator de țări versionat, ideal bazat pe ISO 3166. În backend, aceste trei valori vor fi persistate separat, pentru a permite filtrare, export și raportare corectă.
FRQ272	Dacă pentru același tip de material există mai mulți valorificatori autorizați, sistemul va permite introducerea a **mai multor grupuri de câmpuri** pentru același rând logic. Tehnic, în frontend vom folosi o structură de tip Form.List, astfel încât utilizatorul să poată adăuga dinamic mai multe intrări pentru același material. În backend, modelul de date va avea o relație one-to-many între rândul tabelului și valorificatori, de exemplu ada_report_row_value_recipients.
FRQ273	Pentru coloana **operațiunea de valorificare la care a fost supus deșeul**, frontend-ul va permite **selecție multiplă** din lista operațiunilor **R1–R13**, folosind Select cu mode="multiple" sau o structură repetabilă dacă e nevoie de corelare 1:1 cu fiecare valorificator. În backend vom păstra aceste valori într-o tabelă child separată, de tip ada_report_row_recovery_operations, cu codul operațiunii și legătura către rândul raportului. Dacă pentru același material există mai multe operațiuni distincte, sistemul va permite adăugarea mai multor înregistrări pentru același material.
FRQ274	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea checkbox-ului de confirmare. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ275	Toate câmpurile formularului vor fi tratate ca obligatorii, cu excepția câmpului **Atașare fișiere**. În frontend vom defini reguli de validare explicite pentru toate controalele, iar în backend DTO-ul de creare/submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelele raportului și pentru raportul narativ. În PostgreSQL vom aplica constrângeri NOT NULL pentru câmpurile-cheie și vom valida existența atașamentului de tip NARRATIVE_REPORT.
FRQ276	Vor avea caracter obligatoriu doar tabelele care au fost **bifate** din listă. Dacă un tabel nu este bifat, acesta nu intră în validare și nu este obligatoriu. Dacă un tabel este bifat, sistemul va impune regula ca **cel puțin un rând** să fie completat. În frontend vom implementa această regulă prin validatori dinamici pe secțiune, iar în backend printr-un validator custom care verifică prezența unui minim de date nenule în rândurile tabelului selectat.
FRQ277	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Raportare ADA**, cu tabel server-side și acțiuni de workflow: **Validează raportul**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/ada-

	reports/{id}/actions, care va valida rolul, statusul curent și comentariul obligatoriu pentru corectare sau respingere.
FRQ278	Pentru fiecare raport vom păstra un istoric complet într-o tabelă ada_report_history, cu câmpuri precum report_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare raport**, **Verificare raport**, **Trimis spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Raport aprobat**, **Raport respins**.
FRQ279	Utilizatorul va putea relua în editare raportul transmis atât timp cât acesta nu a fost preluat în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ280	Dacă raportul a fost deja accesat de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva funcționalitatea de editare și va afișa mesajul contextual cerut, iar backend-ul va respinge orice actualizare printr-un cod business dedicat, de exemplu REPORT_IN_PROCESSING. Pentru a evita concurența, la primul acces efectiv în procesare vom seta picked_by și picked_at într-o tranzacție atomică.
FRQ281	În interfața inspectorului, rapoartele ADA vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per categorie. Folderele vor corespunde etapelor: **Rapoarte trimise spre verificare**, **Rapoarte trimise spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Rapoarte aprobate**, **Rapoarte respinse**.
FRQ282	Fiecare folder va afișa numărul de rapoarte și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID raport**, **numele companiei**, **data depunerii**, **anul de raportare**, **persoana responsabilă**, **numărul de înregistrare**, **etapa raportului**. În PostgreSQL vom indexa status, company_id, reporting_year, submitted_at, created_at, registration_number pentru performanță bună la filtrare și sortare.
FRQ283	Dacă inspectorul Agenției de Mediu are suspiciuni privind corectitudinea datelor, raportul va fi mutat în etapa **Verificare IPM**. În acel moment, sistemul va genera o notificare către rolul **Inspector IPM** și va expune raportul în interfața IPM. În UI-ul IPM vom adăuga o secțiune specială pentru atașarea documentelor de control, în special **actul de inspecție**, folosind componenta Upload. Fișierele vor fi stocate în object storage, iar metadatele în tabela report_attachments. După încărcarea documentelor, inspectorul IPM va putea muta raportul în etapa **Luarea deciziei**, printr-un endpoint dedicat de workflow.
FRQ284	Când raportul ajunge în etapa **Luarea deciziei**, inspectorul AM va putea alege una dintre acțiunile: **aprobă raportul**, **trimite spre verificare** sau **respinge raportul**. Tehnic, aceasta va fi implementată prin același mecanism central de workflow, cu validări suplimentare pe statusul curent. Acțiunea **trimite spre verificare** va readuce raportul într-o etapă intermediară de reverificare, iar **respingerea** va necesita

	justificare obligatorie. Toate deciziile vor fi jurnalizate în istoric și vor genera notificări interne și, opțional, email-uri către companie.

6.17. Raportare DEEE

Cerință	Implementare propusă
FRQ285	Accesul la modulul **Raportare DEEE** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și o înregistrare activă în **Lista Producătorilor** pentru categoria **DEEE**. În backend vom implementa un EligibilityService care verifică aceste condiții înainte de accesul la endpoint-urile /api/private/deee-reports/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat mai întâi.
FRQ286	Sistemul va permite gestionarea mai multor companii per utilizator, dar fiecare raport **DEEE** va fi completat și transmis **separat pentru fiecare companie**. În modelul de date vom introduce tabela deee_reports, legată de companies, cu câmpuri precum company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by. Vom aplica și o regulă de unicitate business pentru combinația company_id + reporting_year + rep_category, astfel încât să nu existe două rapoarte active pentru aceeași companie și același an de raportare.
FRQ287	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va include: **Numele companiei**, **Numărul de înregistrare în Lista Producătorilor**, **Localitatea**, **Adresa**, **Telefon**, **E-mail**, **Genurile principale de activitate**, **Numărul mediu de angajați**, **Anul de raportare**, **Atașare Raport narativ**, **Atașare fișiere**, **Persoana autorizată**, **Funcția persoanei autorizate**, plus **Tabelul nr. 1 – 7**. Pentru **sistem individual**, câmpul **Numele companiei** va fi selectat exclusiv din companiile din Cabinetul personal. Pentru **sistem colectiv**, acesta va fi selectat din lista companiilor afiliate acelui sistem colectiv, pe baza relațiilor deja definite în cererea de înregistrare sau notificarea de trecere la sistem colectiv. Câmpurile **Numărul de înregistrare**, **Localitatea**, **Adresa**, **Telefonul**, **E-mail-ul** și **Genurile principale de activitate** vor fi autocompletate după selectarea companiei, pe baza datelor din profil și registru. **Raportul narativ** va fi încărcat prin Upload, cu stocare în object storage și metadate în DB. Toate cele **7 tabele** vor fi construite pe baza unei configurații versionate derivate din anexele HG nr. 212/2018 și vor fi afișate pe **aceeași pagină**, sub formă de secțiuni expandabile. În UI vom implementa o listă de bifare (Checkbox.Group) pentru selectarea tabelelor ce urmează a fi completate; doar secțiunile bifate vor fi expandate și activate pentru editare. Coloana **Categorii** din fiecare tabel va fi alimentată din nomenclatoare versionate corespunzătoare anexelor **1A** și **1B** din HG nr. 212/2018. Persistența recomandată este un model generic de tip report_sections + report_rows + report_cells, pentru a putea administra flexibil structurile diferite ale celor 7 tabele DEEE.

FRQ288	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea checkbox-ului de confirmare. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ289	Toate câmpurile formularului vor fi tratate ca obligatorii, cu excepția câmpului **Atașare fișiere**. În frontend vom defini reguli de validare explicite pentru toate controalele, iar în backend DTO-ul de creare/submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelele raportului și pentru raportul narativ. În PostgreSQL vom aplica constrângeri NOT NULL pentru câmpurile-cheie și vom valida existența atașamentului de tip NARRATIVE_REPORT.
FRQ290	Vor avea caracter obligatoriu doar tabelele care au fost **bifate** din listă. Dacă un tabel nu este bifat, acesta nu intră în validare și nu este obligatoriu. Dacă un tabel este bifat, sistemul va impune regula ca **cel puțin un rând** să fie completat. În frontend vom implementa această regulă prin validatori dinamici pe secțiune, iar în backend printr-un validator custom care verifică prezența unui minim de date nenule în rândurile tabelului selectat.
FRQ291	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Raportare DEEE**, cu tabel server-side și acțiuni de workflow: **Validează raportul**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/deee-reports/{id}/actions, care va valida rolul, statusul curent și comentariul obligatoriu pentru corectare sau respingere.
FRQ292	Pentru fiecare raport vom păstra un istoric complet într-o tabelă deee_report_history, cu câmpuri precum report_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare raport**, **Verificare raport**, **Trimis spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Raport aprobat**, **Raport respins**.
FRQ293	Utilizatorul va putea relua în editare raportul transmis atât timp cât acesta nu a fost preluat în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ294	Dacă raportul a fost deja accesat de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva funcționalitatea de editare și va afișa mesajul contextual cerut, iar backend-ul va respinge orice actualizare printr-un cod business dedicat, de exemplu REPORT_IN_PROCESSING. Pentru a evita concurența, la primul acces efectiv în procesare vom seta picked_by și picked_at într-o tranzacție atomică.
FRQ295	În interfața inspectorului, rapoartele DEEE vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per categorie. Folderele vor

	corespunde etapelor: **Rapoarte trimise spre verificare**, **Rapoarte trimise spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Rapoarte aprobate**, **Rapoarte respinse**.
FRQ296	Fiecare folder va afișa numărul de rapoarte și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID raport**, **numele companiei**, **data depunerii**, **anul de raportare**, **persoana responsabilă**, **numărul de înregistrare**, **etapa raportului**. În PostgreSQL vom indexa status, company_id, reporting_year, submitted_at, created_at, registration_number pentru performanță bună la filtrare și sortare.
FRQ297	Dacă inspectorul Agenției de Mediu are suspiciuni privind corectitudinea datelor, raportul va fi mutat în etapa **Verificare IPM**. În acel moment, sistemul va genera o notificare către rolul **Inspector IPM** și va expune raportul în interfața IPM. În UI-ul IPM vom adăuga o secțiune specială pentru atașarea documentelor de control, în special **actul de inspecție**, folosind componenta Upload. Fișierele vor fi stocate în object storage, iar metadatele în tabela report_attachments. După încărcarea documentelor, inspectorul IPM va putea muta raportul în etapa **Luarea deciziei**, printr-un endpoint dedicat de workflow.
FRQ298	Când raportul ajunge în etapa **Luarea deciziei**, inspectorul AM va putea alege una dintre acțiunile: **aprobă raportul**, **trimite spre verificare** sau **respinge raportul**. Tehnic, aceasta va fi implementată prin același mecanism central de workflow, cu validări suplimentare pe statusul curent. Acțiunea **trimite spre verificare** va readuce raportul într-o etapă intermediară de reverificare, iar **respingerea** va necesita justificare obligatorie. Toate deciziile vor fi jurnalizate în istoric și vor genera notificări interne și, opțional, email-uri către companie.

6.18. Raportare DPT

Cerință	Implementare propusă
FRQ299	Accesul la modulul **Raportare DPT** va fi permis doar utilizatorilor autentificați prin **MPASS** care au cel puțin o companie activă în Cabinetul personal și o înregistrare activă în **Lista Producătorilor** pentru categoria **DPT**. În backend vom implementa un EligibilityService care verifică aceste condiții înainte de accesul la endpoint-urile /api/private/dpt-reports/**. În frontend, dacă una dintre condiții nu este îndeplinită, utilizatorul va vedea un mesaj contextual și link către modulul care trebuie completat mai întâi.
FRQ300	Sistemul va permite gestionarea mai multor companii per utilizator, dar fiecare raport **DPT** va fi completat și transmis **separat pentru fiecare companie**. În modelul de date vom introduce tabela dpt_reports, legată de companies, cu câmpuri precum company_id, reporting_year, status, responsibility_type, registration_number, submitted_at, picked_at, picked_by. Vom aplica și o regulă de unicitate business pentru combinația company_id + reporting_year + rep_category, astfel încât să nu existe două rapoarte active pentru aceeași companie și același an de raportare.
FRQ301	Formularul va fi implementat în **ReactJS + JavaScript + Ant Design Form** și va include: **Numele companiei**, **Numărul de înregistrare în Lista Producătorilor**,

	Localitatea, **Adresa**, **Telefon**, **E-mail**, **Genurile principale de activitate**, **Numărul mediu de angajați**, **Anul de raportare**, **Atașare Raport narativ**, **Atașare fișiere**, **Persoana autorizată**, **Funcția persoanei autorizate**, plus **Tabelul Raportarea datelor privind gestionarea deșeurilor textile rezultate din produsele textile plasate pe piață**. Pentru **sistem individual**, câmpul **Numele companiei** va fi selectat exclusiv din companiile din Cabinetul personal. Pentru **sistem colectiv**, acesta va fi selectat din lista companiilor afiliate acelui sistem colectiv, pe baza relațiilor deja definite în sistem. Câmpurile **Numărul de înregistrare**, **Localitatea**, **Adresa**, **Telefonul**, **E-mail-ul** și **Genurile principale de activitate** vor fi autocompletate după selectarea companiei, pe baza datelor din profil și registru. **Raportul narativ** va fi încărcat prin Upload, cu stocare în object storage și metadate în DB. Tabelul DPT va fi generat dinamic pe baza structurii din **Anexa nr. 6**, folosind un model configurabil de tip report_sections + report_rows + report_cells, astfel încât să putem adapta ușor structura dacă anexa se modifică ulterior.
FRQ302	Butonul **Trimite spre verificare** va fi activ doar după completarea tuturor câmpurilor obligatorii și bifarea checkbox-ului de confirmare. În frontend vom folosi Form.useWatch() și form.validateFields() pentru activare în timp real, iar în backend submit-ul va revalida toate regulile înainte de schimbarea statusului în IN_REVIEW.
FRQ303	Toate câmpurile formularului vor fi tratate ca obligatorii, cu excepția câmpului **Atașare fișiere**. În frontend vom defini reguli de validare explicite pentru toate controalele, iar în backend DTO-ul de creare/submit va avea validări @NotNull, @NotBlank și validatori custom pentru tabelul raportului și pentru raportul narativ. În PostgreSQL vom aplica constrângeri NOT NULL pentru câmpurile-cheie și vom valida existența atașamentului de tip NARRATIVE_REPORT.
FRQ304	Pentru **Inspectorul Agenției de Mediu** vom implementa un ecran dedicat în zona **Monitorizare și Raportare → Raportare REP → Raportare DPT**, cu tabel server-side și acțiuni de workflow: **Validează raportul**, **Trimite spre corectare** și **Respinge/Invalidează**. Backend-ul va expune un endpoint de tip /api/private/dpt-reports/{id}/actions, care va valida rolul, statusul curent și comentariul obligatoriu pentru corectare sau respingere.
FRQ305	Pentru fiecare raport vom păstra un istoric complet într-o tabelă dpt_report_history, cu câmpuri precum report_id, from_status, to_status, actor_id, comment, created_at. În frontend, istoricul va fi afișat prin Timeline sau Steps, cu etapele: **Completare raport**, **Verificare raport**, **Trimis spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Raport aprobat**, **Raport respins**.
FRQ306	Utilizatorul va putea relua în editare raportul transmis atât timp cât acesta nu a fost preluat în procesare de inspector. În backend vom folosi câmpuri precum picked_at și picked_by; dacă acestea nu sunt setate, acțiunea **Editare** va fi permisă și sistemul va muta automat starea din IN_REVIEW în DRAFT. În frontend, butonul **Editare** va fi afișat doar dacă API-ul returnează editable=true.
FRQ307	Dacă raportul a fost deja accesat de inspector și a intrat în procesare, editarea va fi blocată. Frontend-ul va dezactiva funcționalitatea de editare și va afișa mesajul

	contextual cerut, iar backend-ul va respinge orice actualizare printr-un cod business dedicat, de exemplu REPORT_IN_PROCESSING. Pentru a evita concurența, la primul acces efectiv în procesare vom seta picked_by și picked_at într-o tranzacție atomică.
FRQ308	În interfața inspectorului, rapoartele DPT vor fi grupate pe **foldere logice / categorii de status**. În React vom implementa această structură prin Tabs sau meniu lateral cu badge-uri numerice. În backend, statusurile vor fi modelate prin enum și vor exista endpoint-uri pentru listă și pentru contorii per categorie. Folderele vor corespunde etapelor: **Rapoarte trimise spre verificare**, **Rapoarte trimise spre corectare**, **Verificare IPM**, **Luarea deciziei**, **Rapoarte aprobate**, **Rapoarte respinse**.
FRQ309	Fiecare folder va afișa numărul de rapoarte și un tabel server-side cu maximum **50 de înregistrări pe pagină**, sortate descrescător după data depunerii sau creării. Coloanele tabelului vor fi: **ID raport**, **numele companiei**, **data depunerii**, **anul de raportare**, **persoana responsabilă**, **numărul de înregistrare**, **etapa raportului**. În PostgreSQL vom indexa status, company_id, reporting_year, submitted_at, created_at, registration_number pentru performanță bună la filtrare și sortare.
FRQ310	Dacă inspectorul Agenției de Mediu are suspiciuni privind corectitudinea datelor, raportul va fi mutat în etapa **Verificare IPM**. În acel moment, sistemul va genera o notificare către rolul **Inspector IPM** și va expune raportul în interfața IPM. În UI-ul IPM vom adăuga o secțiune specială pentru atașarea documentelor de control, în special **actul de inspecție**, folosind componenta Upload. Fișierele vor fi stocate în object storage, iar metadatele în tabela report_attachments. După încărcarea documentelor, inspectorul IPM va putea muta raportul în etapa **Luarea deciziei**, printr-un endpoint dedicat de workflow.
FRQ311	Când raportul ajunge în etapa **Luarea deciziei**, inspectorul AM va putea alege una dintre acțiunile: **aprobă raportul**, **trimite spre verificare** sau **respinge raportul**. Tehnic, aceasta va fi implementată prin același mecanism central de workflow, cu validări suplimentare pe statusul curent. Acțiunea **trimite spre verificare** va readuce raportul într-o etapă intermediară de reverificare, iar **respingerea** va necesita justificare obligatorie. Toate deciziile vor fi jurnalizate în istoric și vor genera notificări interne și, opțional, email-uri către companie.

6.19. Cerințe generale

Cerință	Implementare propusă
FRQ312	Vom implementa un **motor generic de filtrare avansată** reutilizabil în toate tabelele din interfețele inspectorilor și ale altor organe de control. În frontend, vom folosi **ReactJS + Ant Design** cu o componentă de tip **Filter Builder**, care permite adăugarea și eliminarea dinamică a regulilor, gruparea lor logică și selecția operatorilor în funcție de tipul câmpului. Pentru câmpurile text vom suporta operatori precum contains, equals, startsWith; pentru câmpuri numerice =, >, <, between; pentru date on, before, after, between; pentru status și enum-uri in, not in. Pentru combinarea multiplă a filtrelor vom utiliza grupuri logice de tip **AND/OR**, cu posibilitatea de a construi expresii nested. În backend, vom implementa un layer generic de query

	building, bazat pe **Spring Data Specifications**, care va transforma expresiile de filtrare în query-uri SQL sigure și optimizate. Pentru performanță, filtrele vor fi executate server-side, iar coloanele utilizate frecvent la căutare vor fi indexate în PostgreSQL.
FRQ313	Vom implementa un **serviciu unificat de export** pentru toate tabelele vizibile în interfața inspectorilor. Frontend-ul va transmite către backend starea curentă a gridului: filtrele active, sortarea, coloanele afișate și ordinea lor, precum și formatul dorit (xlsx, csv, docx, pdf). Backend-ul va genera exportul strict pe baza dataset-ului filtrat curent, astfel încât utilizatorul să primească exact rezultatul vizibil în interfață. Pentru generare vom folosi: **Apache POI** pentru XLSX, stream text pentru CSV, bibliotecă de generare DOCX pentru Word și bibliotecă PDF pentru PDF. Sistemul va include în export doar coloanele selectate și în aceeași ordine ca în UI. La finalul fiecărui export va fi adăugat automat un rând de **total** pentru coloanele numerice unde suma este aplicabilă. Pentru fișiere mari, exportul va rula asincron pe backend cu status polling sau descărcare directă prin stream, în funcție de volum.
FRQ314	Vom implementa un mecanism de **șabloane de filtrare salvate per utilizator**. În frontend va exista o zonă dedicată cu acțiuni: **Salvează șablon**, **Aplică șablon**, **Redenumește**, **Șterge**. La salvare, sistemul va persista în backend un obiect ce conține: numele șablonului, codul ecranului/tabelului, setul complet de filtre, logica dintre filtre, sortarea și, opțional, configurația coloanelor vizibile. În PostgreSQL vom folosi o tabelă de tip user_filter_templates(id, user_id, screen_code, template_name, filter_payload_json, created_at, updated_at). filter_payload_json va fi salvat în format JSONB, astfel încât structura filtrelor să fie flexibilă și extensibilă. La aplicarea unui șablon, frontend-ul va reconstrui automat regulile în builder și va reîncărca datele. Accesul la aceste șabloane va fi strict per utilizator, fără partajare implicită între inspectori.
FRQ315	Pentru toate tabelele din interfața inspectorilor și a altor organe de control vom implementa un sistem dual: **coloane implicite** și **configurare personalizabilă**. Fiecare ecran va avea un set de coloane defaultVisible, definit conform cerințelor funcționale specifice acelui contur. În plus, utilizatorul va putea deschide un **Column Manager** pentru a selecta ce coloane suplimentare dorește să afișeze, ascundă sau reordoneze. Filtrele avansate vor putea fi aplicate nu doar pe coloanele vizibile, ci pe **toate coloanele disponibile** ale tabelului, inclusiv cele ascunse. În backend, fiecare ecran va avea o definiție metadata a coloanelor, de tip screen_table_columns(screen_code, column_code, label, data_type, default_visible, filterable, sortable, exportable). Această abordare ne permite să controlăm centralizat ce coloane sunt disponibile, filtrabile și exportabile, fără hardcodare excesivă în frontend. Opțional, putem salva și preferințele utilizatorului privind coloanele vizibile într-o tabelă user_table_preferences, astfel încât fiecare inspector să își regăsească interfața exact cum a configurat-o anterior.
FRQ316	În toate tabelele din interfața inspectorilor vom extinde componenta generică de grid astfel încât utilizatorul să poată controla complet modul de vizualizare și selecție a datelor. În frontend, pe baza **Ant Design Table**, vom activa selecția

	dimensiunii paginii cu opțiuni configurabile de tip **10 / 25 / 50 / 100 / 200**, căutare globală full-text și selecția multiplă a rândurilor prin checkbox-uri. Vom adăuga și acțiunea **„Selectează tot”** pentru toate rândurile vizibile din pagina curentă. Pentru căutarea globală, frontend-ul va trimite un parametru unic search către backend, iar backend-ul va implementa un strat generic de căutare text pe toate coloanele marcate ca searchable în metadatele tabelului. Pentru performanță, căutarea full-text va fi executată server-side, folosind fie ILIKE controlat pe coloane configurate, fie indexare text dedicată în PostgreSQL pentru ecranele cu volume mari. Selecția rândurilor va fi păstrată în starea locală a tabelului și va putea fi utilizată ulterior pentru export selectiv.
FRQ317	Vom implementa un **mecanism centralizat de notificări** care va reacționa la orice schimbare de status pentru cereri, notificări și rapoarte. În backend vom avea un NotificationService comun, apelat de toate modulele de workflow, care va genera simultan: (1) o notificare internă în inbox-ul din platformă și (2) un email către utilizatorul sau utilizatorii relevanți. Vom salva notificările în tabela notifications, cu atribute precum object_type, object_id, old_status, new_status, recipient_user_id, title, message, is_read, created_at. Pentru email vom folosi un EmailService asincron, astfel încât schimbarea de status să nu fie blocată de timpul de transmitere. În frontend, utilizatorul va vedea noile notificări în tabul **Mesagerie** și, opțional, într-un badge de notificări din header.
FRQ318	După fiecare schimbare de status, sistemul va muta automat obiectul în **folderul logic corespunzător** noului status, fără intervenție manuală. Tehnic, acest lucru nu va însemna mutarea fizică a datelor, ci schimbarea atributului status în DB și recalcularea listelor/folderelor pe baza acestuia. În frontend, fiecare folder este de fapt o vizualizare filtrată pe status, implementată prin Tabs sau meniuri dedicate. Astfel, după modificarea statusului și invalidarea cache-ului, obiectul va dispărea automat din lista curentă și va apărea în categoria corespunzătoare noii etape.
FRQ319	Vom implementa o **validare automată centralizată** pentru toate formularele din sistem, astfel încât niciun buton de tip **Trimite spre verificare**, **Salvează** sau altă acțiune de validare să nu poată fi utilizat dacă lipsesc date obligatorii. În frontend vom folosi validarea standard a formularelor Ant Design (rules, validatori custom, form.validateFields()), iar în backend vom dubla toate regulile prin Bean Validation și validatori business specifici fiecărui modul. Dacă lipsesc date, sistemul va întoarce erori structurate pe câmpuri, iar UI-ul va afișa mesaje contextuale clare direct lângă câmpurile problematice sau la nivel de secțiune/tabel. Pentru tabele, validatorii vor verifica inclusiv regula de tip **„cel puțin o valoare completată”** acolo unde este cazul.
FRQ320	Pentru obiectele trimise spre corectare, vom permite utilizatorului să modifice **toate câmpurile completate manual**, păstrând în același timp valorile provenite automat din registre sau calcule în regim read-only, dacă regula de business o cere. În frontend, după trecerea într-un status de tip NEEDS_CORRECTION, formularul va fi reactivat și va afișa suplimentar un câmp nou **Comentarii privind modificările**, implementat ca TextArea, dar fără caracter obligatoriu. În backend vom păstra acest

	comentariu în istoricul obiectului, într-o tabelă de tip *_history sau *_revision_comments. La retransmitere, sistemul va reaplica întregul flux de validare: completarea câmpurilor obligatorii, verificarea tabelelor și bifarea declarației de veridicitate. Numai după trecerea acestor validări obiectul va putea reveni în IN_REVIEW.
FRQ321	Vom implementa un tab dedicat **„Rapoarte ADA”** în zona inspectorului, orientat pe elaborarea rapoartelor pentru resursa proprie bazată pe deșeurile de ambalaje din plastic nereciclate. În frontend, acest modul va avea un ecran separat cu buton **„Generează raport”**, selector de an și afișarea tuturor tabelelor prevăzute în Anexa nr. 13. La generare, backend-ul va crea o nouă entitate de tip ada_plastic_reports, va popula automat toate câmpurile calculabile din datele deja existente în SIA MD și va marca distinct câmpurile care trebuie completate manual de către **Inspectorul AM 2**. În UI, aceste câmpuri manuale vor fi evidențiate vizual și vor rămâne editabile, în timp ce restul valorilor vor fi read-only. Structura tabelară va fi configurabilă, bazată pe metadate de tip report_templates, pentru a permite actualizarea anexei fără refactorizare majoră. Modulul va include și salvare draft, recalculare la regenerare, istoric de versiuni și export în formatele standard folosite și în celelalte tabele ale sistemului.

7. Implementare

Sistemul de producție al SIA MD va fi implementat și găzduit în cadrul platformei guvernamentale comune MCloud, utilizând containere Docker orchestrate prin Kubernetes, într-o arhitectură bazată pe microservicii, elastică 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, demonstrare, testare intermediară și acceptanță, vor fi utilizate medii separate non-producție, în care vor fi livrate iterativ versiuni demo, versiuni parțiale și versiuni corective, conform procesului de testare și acceptanță stabilit 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, configurare controlată și livrare repetabilă între medii.

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 8GB, HDD 100GB. Configurație worker: CPU 8 nuclee, RAM 32GB, HDD 100GB.
- Server NFS (pentru stocarea documentelor): CPU 2 nuclee, RAM 4GB, HDD 500GB.
- Server bază de date: CPU 8 nuclee, RAM 32GB, HDD 1TB.

8. 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.

9. 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

10. 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.

11. Servicii post-implementare în perioada de garanție

Serviciile post-implementare furnizate de compania noastră în perioada de garanție includ:

- Suport tehnic 24/7: O persoană dedicată de suport tehnic va fi disponibilă permanent pentru a răspunde prompt la orice probleme tehnice sau întrebări legate de sistem. Această persoană va monitoriza continuu sistemul și va asigura remedierea rapidă a oricărei disfuncționalități, pentru a minimiza întreruperile operaționale.

- Dezvoltator alocat: Un programator desemnat special pentru acest proiect va fi disponibil 8 ore pe zi, inclusiv în weekend. Acest dezvoltator va gestiona remedierile critice de defecte și va efectua orice intervenții necesare pentru menținerea performanței și funcționalității sistemului.

- Implicarea echipei de dezvoltare: În funcție de complexitatea problemelor întâlnite, vor fi implicați și alți dezvoltatori din echipa noastră. Acești membri ai echipei își vor aduce expertiza pentru a asigura rezolvarea rapidă și eficientă a oricăror provocări tehnice.

12. Servicii post-garanție

Serviciile post-garanție oferite de compania noastră sunt concepute pentru a asigura stabilitatea continuă și performanța optimă a sistemului după expirarea perioadei de garanție. Aceste servicii includ:

- Mentenanță corectivă: În cazul apariției unor erori sau disfuncționalități ale sistemului, echipa noastră de dezvoltare va interveni pentru a remedia prompt problemele. Mentenanța corectivă este furnizată pe baza unui Acord privind nivelul serviciilor (SLA), agreat cu clientul, pentru a asigura timpi optimi de răspuns și rezolvare.

- Mentenanță evolutivă: Pe lângă remedierea problemelor, oferim servicii de dezvoltare pentru adaptarea sistemului la noi cerințe de business sau pentru îmbunătățirea funcționalităților existente. Acestea includ adăugarea de module noi, optimizarea performanței și integrarea cu alte sisteme.

- Suport tehnic extins: Clienții beneficiază de suport tehnic dedicat, care poate fi oferit prin telefon, email sau platforme de ticketing, în funcție de preferințele și nevoile lor. Suportul este disponibil în timpul programului standard de lucru sau în intervale extinse, în funcție de pachetul de servicii ales.

- Monitorizare proactivă: Continuăm să oferim servicii de monitorizare a sistemului pentru a detecta problemele potențiale înainte ca acestea să devină critice. Echipa noastră utilizează soluții avansate de monitorizare pentru a urmări performanța și stabilitatea sistemului.

- Consultanță tehnică: Pe măsură ce nevoile de business evoluează, oferim servicii de consultanță pentru a adapta soluția software la noi provocări și oportunități. Acestea pot include analize tehnice, recomandări de îmbunătățire și planificarea dezvoltărilor viitoare.

13.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.