

UrHomie - Platformă web distribuită destinată serviciilor de Home Maintenance

Coordonator științific:
ș.I. dr.ing. Aflori Cristian

Absolvent:
Gherasim Dragoș-George

Iași, 2025

Cuprins

Capitolul 1. Introducere	1
1.1. Contextul și importanța temei alese.....	1
1.2. Obiectivele generale și specifice ale lucrării	1
1.2.1. Obiectiv general.....	1
1.2.2. Obiective specifice.....	1
1.3. Metodologia de cercetare utilizată.....	2
1.4. Structura generală a lucrării.....	2
Capitolul 2. Fundamentarea teoretică și soluții similare.....	3
2.1. Fundamente teoretice și concepte cheie privind platformele de servicii pentru întreținerea locuinței	3
2.2. Evaluarea abordărilor actuale din literatura de specialitate	5
2.3. Explorarea soluțiilor existente în cadrul platformelor de servicii casnice.....	5
2.4. Lacune identificate în stadiul actual de dezvoltare și în soluțiile existente	6
2.5. Conturarea cerințelor aplicației în raport cu nevoile identificate	6
Capitolul 3. Soluția propusă.....	8
3.1. Reiterarea problemei și a strategiei de rezolvare	8
3.2. Idei originale, soluții noi.....	9
3.3. Cerințele utilizatorului	9
3.4. Arhitectura sistemului.....	10
3.4.1. Componenta Frontend.....	11
3.4.2. Componente proxy în arhitectura sistemului	14
3.4.3. Componenta UserAuth Microservice	15
3.4.4. Componenta UserManagement Microservice	17
Bibliografie	20
Anexe	21
Anexa 1: Diagrama de activitate pentru procesul de autentificare și acces bazat pe tokenuri ..	21
Anexa 2: Diagrama de activitate pentru procesul de înregistrare distribuit	22

Rezumat

Într-o societate tot mai digitalizată, utilizatorii caută soluții rapide și sigure pentru accesarea diverselor servicii, inclusiv în sfera *home maintenance* (întreținere, reparații, curățenie etc.). În România, numărul platformelor specializate dedicate acestui domeniu este redus, iar cele existente nu oferă o experiență digitală suficient de personalizată, intuitivă și sigură. Motivația dezvoltării proiectului *UrHomie* derivă din această realitate și vizează realizarea unei soluții digitale adaptate contextului local, care să sprijine atât clienții, cât și micii furnizori de servicii.

Obiectivele principale constau în dezvoltarea unei platforme web moderne, fiabile și ușor de utilizat, care facilitează accesul utilizatorilor la servicii specializate și oferă vizibilitate online furnizorilor independenți. *UrHomie* se bazează pe o arhitectură distribuită, compusă din microservicii dezvoltate în *C#*, *Python* și *Java*, utilizând framework-uri precum *ASP.NET Core*, *Spring Boot* și *FastAPI*. Comunicarea între componente este realizată printr-o combinație de tehnologii: *WebSockets* (pentru comunicație bidirecțională în timp real), *gRPC* (framework pentru apeluri de proceduri la distanță, de înaltă performanță) și *REST* (stil arhitectural utilizat pentru proiectarea serviciilor web).

Platforma integrează baze de date relaționale (*MariaDB*) și non-relaționale (*MongoDB*), oferind flexibilitate în stocarea și procesarea datelor. Pentru containerizare se utilizează *Docker*, iar rutarea și conversia protocoalelor este asigurată prin *NGINX* și *Envoy*. Până în prezent, au fost implementate și validate modulele de autentificare, autorizare, înregistrare și gestionare a profilelor utilizatorilor.

Prin arhitectura sa modulară și scalabilă, *UrHomie* are potențialul de a deveni un ecosistem digital complet dedicat serviciilor de home maintenance.

Cuvinte-cheie: Home Maintenance, Platformă Web Distribuită, Microservicii, gRPC, WebSockets, REST API.

Capitolul 1. Introducere

1.1. Contextul și importanța temei alese.

Într-o societate în continuă dezvoltare tehnologică, accesul rapid și sigur la diverse servicii, inclusiv cele de home maintenance, devine o necesitate tot mai presantă. În România, oferta de platforme specializate în acest domeniu este limitată, iar cele existente nu reușesc să ofere o experiență digitală personalizată, intuitivă și sigură. Această situație evidențiază o oportunitate semnificativă pentru dezvoltarea unor soluții adaptate contextului local, care să sprijine atât clienții, cât și micii furnizori de servicii.

Digitalizarea serviciilor de întreținere a locuinței poate aduce multiple beneficii, cum ar fi eficientizarea proceselor, reducerea timpului de așteptare și creșterea transparenței în relația dintre clienți și furnizori. În plus, o astfel de platformă poate contribui la formalizarea activităților desfășurate de furnizorii independenți, oferindu-le acestora o vizibilitate mai mare și acces la o bază mai largă de clienți. Prin urmare, dezvoltarea unei platforme web în acest domeniu răspunde unei nevoi reale a pieței și sprijină incluziunea digitală și dezvoltarea economică locală. [1]

Așadar, importanța temei alese rezidă în potențialul său de a transforma modul în care serviciile de home maintenance sunt accesate și furnizate în România, prin intermediul unei soluții moderne, eficiente și adaptate nevoilor actuale ale utilizatorilor.

1.2. Obiectivele generale și specifice ale lucrării

Lucrarea de față își propune să contribuie la digitalizarea serviciilor de home maintenance în România, prin dezvoltarea unei platforme web inovatoare care să faciliteze conexiunea între clienți și furnizori de servicii. Această inițiativă răspunde unei nevoi reale identificate pe piața locală, unde oferta de platforme specializate este limitată, iar cele existente nu oferă o experiență digitală suficient de personalizată și sigură. [2]

1.2.1. Obiectiv general

Dezvoltarea unei platforme web scalabile și securizate, bazată pe sisteme distribuite și arhitectura pe microservicii, care să permită interacțiunea eficientă între clienți și furnizori de servicii de întreținere a locuinței, oferind o experiență digitală intuitivă și personalizată.

1.2.2. Obiective specifice

1. Identificarea și documentarea nevoilor clienților și a furnizorilor de servicii, pentru a asigura o aliniere optimă între funcționalitățile platformei și așteptările acestora.
2. Elaborarea unei arhitecturi distribuite, bazată pe microservicii, care să asigure modularitate, scalabilitate și ușurință în mentenanță. [3]
3. Dezvoltarea microserviciilor utilizând tehnologii precum ASP.NET Core, Spring Boot și FastAPI, asigurând o comunicare eficientă prin intermediul gRPC și REST.
4. Integrarea bazelor de date relaționale (MariaDB) și non-relaționale (MongoDB) pentru stocarea și gestionarea eficientă a datelor utilizatorilor și a serviciilor oferite.

5. Implementarea unui sistem robust de autentificare și autorizare, utilizând JWT pentru transmiterea securizată a datelor între participanți.
6. Utilizarea Docker pentru containerizarea aplicației și NGINX ca reverse proxy, facilitând astfel scalabilitatea și distribuția eficientă a traficului.
7. Crearea unei interfețe web intuitive și responsive, care să ofere o experiență plăcută și accesibilă pentru utilizatori. [4]
8. Realizarea testelor unitare și de integrare pentru a asigura funcționarea corectă a componentelor și a întregului sistem.
9. Analiza performanței platformei în condiții variate de încărcare, pentru a identifica și implementa îmbunătățiri necesare.
10. Redactarea unei documentații detaliate care să acopere toate aspectele tehnice și funcționale ale platformei, facilitând astfel utilizarea și extinderea ulterioară a acesteia.

Prin atingerea acestor obiective, lucrarea își propune să ofere o soluție digitală viabilă și eficientă pentru facilitarea accesului la servicii de home maintenance, contribuind astfel la modernizarea și digitalizarea acestui sector în România.

1.3. Metodologia de cercetare utilizată

În etapa inițială a proiectului, am analizat diverse proiecte similare disponibile pe platforma GitHub, concentrându-mă pe cele care utilizează arhitecturi bazate pe microservicii și tehnologii moderne. Această analiză mi-a permis să înțeleg mai bine structura și funcționalitățile necesare pentru o platformă de acest tip. De asemenea, am efectuat o cercetare extensivă în mediu online pentru a identifica cele mai bune practici în proiectarea arhitecturilor scalabile, documentându-mă din surse valide și actualizate noilor cerințe pe piața IT.

Pe parcursul dezvoltării, am aplicat concepte fundamentale de proiectare software, precum principiile SOLID, care promovează o arhitectură robustă și ușor de întreținut. Totodată, am integrat diverse design patterns, cum ar fi Saga, Repository, Dependency Injection, pentru a asigura o structură modulară și scalabilă a aplicației. Am consultat resurse academice și documentații tehnice pentru a mă asigura că soluțiile implementate sunt aliniate cu cele mai bune practici în domeniul dezvoltării software.

Pe tot parcursul proiectului, am menținut o colaborare strânsă cu domnul profesor îndrumător, discutând periodic despre progresul realizat și deciziile tehnice adoptate. Feedback-ul oferit de acesta a fost esențial în validarea alegerilor stack-ului și în orientarea dezvoltării platformei, colaborarea noastră asigurând alinierea proiectului la cerințele academice și la nevoile reale ale utilizatorilor.

1.4. Structura generală a lucrării

Lucrarea de față este structurată în mod logic și progresiv, astfel încât să reflecte analiza teoretică și implementarea practică a platformei UrHomie. Primul capitol, Introducere, oferă contextul și motivația alegerii temei, definind obiectivele urmărite și metodologia de cercetare adoptată. În continuare, capitolul Fundamentarea teoretică și soluții similare explorează conceptele relevante pentru arhitectura sistemelor distribuite, designul orientat pe microservicii și experiența

utilizatorului (UI/UX), punând accent și pe soluțiile deja existente în domeniul home maintenance, identificând limitările și oportunitățile neacoperite.

Capitolul central, Soluția propusă, constituie nucleul lucrării și detaliază procesul de proiectare și implementare al aplicației, de la cerințele utilizatorilor până la alegerea tehnologiilor și arhitectura sistemului. Sunt incluse modelarea bazelor de date, descrierea componentelor software dezvoltate, justificarea deciziilor tehnice și capturi de ecran din aplicația funcțională. Urmează Testarea soluției și rezultate experimentale, unde sunt prezentate metodele de validare aplicate sistemului, inclusiv testarea funcțională, de performanță și scenarii de utilizare, susținute prin tabele, grafice și interpretări.

Ultima parte a lucrării cuprinde Concluziile, unde sunt sintetizate contribuțiile personale, impactul soluției dezvoltate și posibile direcții de extindere viitoare. În încheiere, Bibliografia conține sursele academice și tehnice consultate, iar Anexele includ diagrame UML, fragmente de cod relevante și alte resurse care susțin înțelegerea și replicabilitatea proiectului.

Capitolul 2. Fundamentarea teoretică și soluții similare

2.1. Fundamente teoretice și concepte cheie privind platformele de servicii pentru întreținerea locuinței

În contextul dezvoltării platformei *UrHomie*, adoptarea unei arhitecturi web distribuite reprezintă o alegere strategică, menită să asigure scalabilitate, reziliență și performanță în gestionarea relațiilor dintre clienți și furnizorii de servicii de home maintenance. O astfel de arhitectură permite distribuirea sarcinilor între multiple componente autonome, facilitând adaptarea rapidă la cerințele variate ale utilizatorilor și la volumul fluctuant de solicitări.

Unul dintre avantajele majore ale sistemelor distribuite este capacitatea de scalare orizontală, prin adăugarea de noi noduri sau servicii pentru a gestiona creșterea cererii, fără a compromite performanța generală a sistemului. [5] Această flexibilitate este esențială pentru platforme precum *UrHomie*, care trebuie să răspundă eficient atât nevoilor clienților, cât și ale furnizorilor. De asemenea, arhitectura distribuită oferă toleranță la erori, asigurând continuitatea serviciilor chiar și în cazul unor defecțiuni ale anumitor componente.

Arhitectura bazată pe microservicii

Arhitectura bazată pe microservicii reprezintă o abordare modernă în dezvoltarea aplicațiilor, în care sistemul este divizat în componente mici, autonome și independente, numite microservicii. Fiecare microserviciu este responsabil pentru o funcționalitate specifică și poate fi dezvoltat, implementat și scalat independent de celelalte, acest tip de arhitectură oferind avantaje precum modularitatea, scalabilitatea și flexibilitatea în gestionarea aplicațiilor complexe.[6]

Principiile fundamentale ale microserviciilor includ: [7]

- Autonomie, mai exact fiecare serviciu funcționează independent, având propriul ciclu de viață și propriile date;

- Cuplare slabă, deoarece microserviciile interacționează între ele prin intermediul unor interfețe bine definite, reducând dependențele și facilitând modificările locale fără a afecta întregul sistem;
- Toleranță la erori prin faptul că sistemul este proiectat astfel încât eșecurile unui microserviciu să nu afecteze funcționarea celorlalte componente;
- Compozabilitate, ceea ce înseamnă că microserviciile pot fi combinate pentru a crea funcționalități mai complexe, permițând reutilizarea componentelor existente.

Principiile SOLID în proiectarea software

Principiile SOLID reprezintă un set de cinci reguli fundamentale în programarea orientată pe obiect, menite să îmbunătățească calitatea codului și să faciliteze întreținerea și extensibilitatea aplicațiilor. Aceste principii sunt: [8]

- S (Single Responsibility Principle) - o clasă ar trebui să aibă o singură responsabilitate, adică să fie responsabilă pentru un singur aspect al funcționalității sistemului;
- O (Open/Closed Principle) - entitățile software (clase, module, funcții) ar trebui să fie deschise pentru extensie, dar închise pentru modificare;
- L (Liskov Substitution Principle) - obiectele unei clase derivate ar trebui să poată înlocui obiectele clasei de bază fără a afecta corectitudinea programului;
- I (Interface Segregation Principle) - este mai bine să se creeze interfețe specifice pentru fiecare client decât o interfață generală care să conțină metode inutile pentru unii clienți;
- D (Dependency Inversion Principle) - modulele de nivel înalt nu ar trebui să depindă de modulele de nivel jos; ambele ar trebui să depindă de abstracții.

Design Patterns în dezvoltarea software

Design patterns sunt soluții general acceptate pentru probleme recurente în dezvoltarea software ce oferă un cadru standardizat care facilitează comunicarea între dezvoltatori și promovează reutilizarea codului. În dezvoltarea platformei *UrHomie*, aplicarea unor design patterns consacrate a fost esențială pentru asigurarea unei arhitecturi flexibile și scalabile. Printre aceste șabloane de proiectare se numără:

- Dependency Injection (DI), un pattern care promovează inversarea controlului prin injectarea dependențelor unei clase din exterior, reducând astfel cuplajul între componente și facilitând testarea și întreținerea codului;
- Repository Pattern ce reprezintă un șablon structural care oferă o abstractizare a accesului la date, separând logica de acces la date de logica de business a aplicației, ceea ce contribuie la o arhitectură mai curată și modulară;
- Saga Pattern, utilizat pentru gestionarea tranzacțiilor distribuite în arhitecturile bazate pe microservicii, pattern ce permite menținerea consistenței datelor prin definirea unei serii de tranzacții locale, fiecare cu o acțiune de compensare în caz de eșec.

Stilul arhitectural REST

REST (Representational State Transfer) este un stil arhitectural pentru dezvoltarea serviciilor web, care se bazează pe un set de constrângeri și principii pentru a crea sisteme scalabile și ușor de întreținut. Principiile fundamentale ale REST includ: [9]

- Statelessness, adică fiecare cerere de la client către server trebuie să conțină toate informațiile necesare pentru a înțelege și procesa cererea;
- Uniform Interface, ce presupune ideea că se utilizează o interfață uniformă pentru a interacționa cu resursele, de obicei prin metodele HTTP standard (GET, POST, PUT, PATCH, DELETE);
- Client-Server Architecture, deoarece separarea responsabilităților între client și server, permițând dezvoltarea și scalarea independentă a fiecărei componente;
- Cacheability, răspunsurile trebuie să fie explicit marcate ca fiind cacheable sau non-cacheable pentru a îmbunătăți performanța.

2.2. Evaluarea abordărilor actuale din literatura de specialitate

Pentru a înțelege mai bine contextul și direcțiile actuale în dezvoltarea platformelor de servicii pentru întreținerea locuinței, este importantă o analiză a cercetărilor academice și studiilor relevante din domeniu. Aceasta permite evidențierea tendințelor emergente, a tehnologiilor frecvent utilizate și a principalelor provocări întâlnite în proiectarea acestor soluții.

Un exemplu notabil este aplicația *Home Repairs*, dezvoltată de cercetători de la Middle East College. Aceasta oferă o gamă largă de servicii de întreținere, precum reparații electrice, montaj de mobilier și curățenie. Aplicația permite utilizatorilor să selecteze serviciile dorite, să specifice preferințele și să evalueze calitatea serviciilor primite, facilitând astfel o interacțiune eficientă între clienți și furnizori. Studiul evidențiază importanța aplicațiilor mobile în facilitarea accesului la servicii de întreținere și în îmbunătățirea experienței utilizatorilor. [10]

Astfel, literatura de specialitate subliniază tendința acescendentă de integrare a tehnologiilor digitale în domeniul serviciilor de home maintenance. Aplicațiile mobile, în special, joacă un rol important în conectarea eficientă a clienților cu prestatorii de servicii, oferind soluții rapide și personalizate pentru nevoile de întreținere.

2.3. Explorarea soluțiilor existente în cadrul platformelor de servicii casnice

Acest subcapitol vizează o analiză a platformelor existente în domeniul serviciilor de întreținere a locuinței, atât la nivel național, cât și internațional, cu scopul de a evidenția inovațiile propuse de acestea, precum și limitările întâmpinate în practică.

În contextul platformelor naționale dedicate întreținerii locuinței, *Home Maintenance România* se remarcă drept o companie specializată în furnizarea de servicii de mentenanță pentru imobile și cartiere rezidențiale. Printre serviciile oferite se numără amenajările peisagistice, instalarea sistemelor de irigații automatizate, precum și lucrările de construcții civile [11]. O altă

inițiativă relevantă este *Servicii24.ro*, o platformă care funcționează ca un marketplace digital, facilitând conexiunea directă între furnizorii de servicii și clienți. Aceasta oferă un cadru eficient pentru identificarea și contractarea rapidă a specialiștilor în diverse domenii de întreținere și reparații [12].

Pe plan internațional, *TaskRabbit* este una dintre cele mai cunoscute platforme online care conectează utilizatorii cu furnizori locali pentru o gamă variată de servicii. Printre acestea se numără reparațiile casnice, montajul de mobilier și serviciile de curățenie. Utilizatorii pot alege furnizorii în funcție de criterii precum prețul, recenziile altor clienți și disponibilitatea, toate operațiunile fiind gestionate facil prin intermediul aplicației dedicate [13]. O altă platformă relevantă este *Handy*, care oferă servicii de curățenie și reparații pentru locuințe, punând accent pe accesibilitatea și eficiența procesului de rezervare. Utilizatorii pot programa rapid intervențiile unor profesioniști verificați pentru sarcini diverse, de la montarea televizoarelor la instalarea corpurilor de iluminat. Platforma se distinge printr-un sistem de rezervare simplificat și prin opțiunea de plată securizată online [14].

2.4. Lacune identificate în stadiul actual de dezvoltare și în soluțiile existente

O analiză critică a limitărilor și provocărilor cu care se confruntă platformele și aplicațiile actuale din domeniul serviciilor de întreținere a locuinței se dovedește necesară pentru evidențierea nevoii de inovație. Această evaluare fundamentează justificarea dezvoltării unei soluții noi, capabile să răspundă mai eficient cerințelor utilizatorilor.

Analiza platformelor locale existente în domeniul serviciilor de întreținere a locuinței relevă o serie de limitări importante. *Home Maintenance România*, de exemplu, oferă servicii variate în acest sector, însă prezentarea lor se realizează într-un mod tradițional, lipsind o platformă digitală interactivă care să faciliteze comunicarea directă și eficientă între clienți și furnizori. Pe de altă parte, *Servicii24.ro*, deși acoperă o gamă extinsă de servicii, inclusiv cele legate de întreținerea locuinței, nu reușește să ofere o experiență digitală complet personalizată. Platforma nu dispune de funcționalități avansate, precum gestionarea programărilor sau urmărirea în timp real a progresului serviciilor, ceea ce limitează nivelul de interacțiune și control al utilizatorilor.

În ceea ce privește aplicațiile mobile existente, un exemplu relevant este aplicația *Home Repairs*, dezvoltată de cercetători de la Middle East College. Aceasta oferă servicii diverse, cum ar fi reparații electrice, montaj de mobilier și curățenie. Totuși, studiile de specialitate subliniază că aplicația nu include funcționalități moderne, precum programarea automată a serviciilor sau integrarea cu sisteme inteligente de monitorizare a locuinței. Aceste lipsuri evidențiază o oportunitate reală de dezvoltare a unor soluții mai avansate, care să răspundă mai bine nevoilor utilizatorilor și să crească eficiența operațională.

2.5. Conturarea cerințelor aplicației în raport cu nevoile identificate

Cerințe funcționale

Un prim set de cerințe funcționale vizează *gestionarea utilizatorilor*. Aplicația trebuie să includă un sistem de autentificare securizat, care să permită accesul atât clienților, cât și furnizorilor de servicii. Înregistrarea va putea fi realizată prin mai multe metode, inclusiv adresă de e-mail, număr de telefon sau conturi sociale. După autentificare, fiecare utilizator va beneficia de un profil personalizat, unde vor fi afișate informații relevante precum istoricul comenzilor, evaluările primite

și preferințele exprimate în cadrul aplicației.

În ceea ce privește *catalogul de servicii*, aplicația va pune la dispoziție o interfață intuitivă pentru listarea și căutarea serviciilor disponibile. Utilizatorii vor putea aplica filtre după criterii precum categoria serviciului, preț, rating sau disponibilitate. Fiecare serviciu listat va beneficia de o pagină dedicată care va conține o descriere detaliată, informații privind prețul și durata estimativă a intervenției, precum și recenzii din partea clienților anteriori.

Componenta de *programare și gestionare a comenzilor* va fi susținută de un sistem de calendar integrat, care le va permite utilizatorilor să selecteze intervalele orare disponibile pentru prestarea serviciilor. În completare, aplicația va trimite notificări automate pentru confirmarea programărilor, mementouri înainte de data prestabilită, precum și actualizări în timp real cu privire la statusul comenzii.

În zona de *plăți și facturare*, aplicația va fi integrată cu gateway-uri de plată securizate, astfel încât utilizatorii să poată efectua tranzacții online în condiții de siguranță, folosind metode variate de plată. Totodată, sistemul va genera automat facturi și chitanțe pentru fiecare comandă finalizată, documentele fiind disponibile pentru descărcare din contul utilizatorului.

Pentru a asigura transparența și calitatea serviciilor, aplicația va include un sistem de *evaluări și recenzii*. După prestarea unui serviciu, clienții vor putea acorda o notă furnizorului și vor putea lăsa comentarii, aceste informații contribuind la formarea unei reputații bazate pe experiențe reale.

Nu în ultimul rând, aplicația va pune accent pe o *interfață personalizată și prietenoasă cu utilizatorul*. Navigarea va fi intuitivă, cu meniuri clar structurate și acces rapid la funcționalitățile principale. Elemente vizuale coerente, cum ar fi o paletă cromatică echilibrată, fonturi lizibile și iconuri sugestive, vor contribui la orientarea ușoară a utilizatorului. De asemenea, aplicația va furniza feedback vizual și auditiv pentru a semnaliza acțiunile efectuate sau eventualele erori întâmpinate în timpul utilizării.

Cerințe tehnice

Pentru a garanta o experiență optimă, aplicația trebuie să îndeplinească o serie de cerințe tehnice esențiale. În ceea ce privește *performanța și scalabilitatea*, sistemul trebuie să asigure timpi de răspuns rapizi pentru toate operațiunile, inclusiv în condiții de trafic intens sau de utilizare simultană de către un număr mare de utilizatori. Arhitectura tehnică va fi proiectată astfel încât să permită scalarea orizontală, oferind posibilitatea extinderii capacității aplicației fără a compromite performanța.

Din perspectiva *securității*, aplicația va integra mecanisme avansate pentru protejarea datelor personale și financiare ale utilizatorilor, respectând în același timp reglementările legale în vigoare, precum cele impuse de GDPR. Autentificarea și autorizarea se vor baza pe protocoale robuste, menite să prevină accesul neautorizat și eventualele breșe de securitate.

Disponibilitatea și fiabilitatea sistemului sunt, de asemenea, esențiale. Aplicația va trebui să funcționeze cu un timp de disponibilitate de cel puțin 99.9%, reducând la minimum perioadele de indisponibilitate. În plus, vor fi implementate strategii de backup și mecanisme eficiente de recuperare în caz de dezastru, astfel încât pierderile de date sau funcționalitate să fie prevenite sau remediate rapid.

Capitolul 3. Soluția propusă

3.1. Reiterarea problemei și a strategiei de rezolvare

În contextul digital actual, intermedierea eficientă între cerere și ofertă în domenii diverse – de la servicii casnice la activități profesionale specializate – este esențială pentru a răspunde rapid și adecvat nevoilor utilizatorilor. Cu toate acestea, multe dintre platformele actuale dedicate serviciilor de întreținere a locuinței se confruntă cu limitări precum lipsa unei interfețe moderne și interactive, absența unui mecanism eficient de programare și urmărire a serviciilor, personalizare redusă a experienței utilizatorului și integrare deficitară cu tehnologii contemporane, precum notificările în timp real sau monitorizarea digitală a progresului solicitărilor.

Această lucrare de licență propune dezvoltarea unei platforme web distribuite, denumită *UrHomie*, destinată facilitării interacțiunii dintre clienți și furnizorii de servicii din domeniul întreținerii locuinței. Platforma oferă o infrastructură digitală completă, care acoperă aspecte precum gestionarea utilizatorilor, administrarea serviciilor disponibile, căutarea și filtrarea acestora, precum și programarea și rezervarea serviciilor într-un mod intuitiv și eficient.

Problema abordată vizează gestionarea coerentă a unor procese critice, precum înregistrarea, validarea și coordonarea componentelor distribuite, într-un ecosistem scalabil. Totodată, se urmărește înlocuirea mecanismelor clasice de interogare repetitivă (polling) cu notificări în timp real, menite să îmbunătățească atât experiența utilizatorului, cât și eficiența operațională a sistemului.

Strategia de rezolvare se fundamentează pe o arhitectură orientată pe microservicii și integrează următoarele principii și mecanisme:

- separarea clară a responsabilităților între componenta de autentificare și cea de gestionare a profilului, fiecare fiind susținută de infrastructură și bază de date distincte;
- utilizarea unui broker de mesaje (RabbitMQ) pentru coordonarea evenimentelor dintre componente, inclusiv gestionarea tranzacțiilor distribuite printr-un model de tip Saga;
- integrarea aplicației web cu microserviciul responsabil de autentificare, înregistrare și autorizare prin intermediul unui proxy compatibil gRPC-Web (ex. Envoy), care permite frontend-ului să comunice eficient și securizat cu serviciile gRPC expuse, în ciuda limitărilor impuse de mediul browser;
- utilizarea unei stocări temporare (precum Redis) pentru menținerea stării fluxurilor asincrone identificate printr-un `correlation_id`, care asigură corelarea exactă a operațiilor distribuite;
- notificarea utilizatorului în timp real, prin intermediul unui canal WebSocket, eliminând necesitatea interogărilor recurente și reducând latențele în procesul de feedback.

3.2. Idei originale, soluții noi

Soluția propusă în cadrul platformei *UrHomie* se remarcă printr-o serie de decizii arhitecturale și funcționale care răspund nevoilor actuale ale utilizatorilor, dar și prin introducerea unor elemente inovatoare care diferențiază această aplicație de majoritatea platformelor similare existente pe piață. Una dintre principalele idei originale constă în separarea proceselor de autentificare și gestionare a profilului în microservicii distincte, cu baze de date izolate și logici proprii. Această abordare oferă o modularitate crescută, permițând dezvoltarea, testarea și scalarea independentă a fiecărei componente, reducând în același timp gradul de cuplare între subsisteme.

Un alt element de noutate este implementarea unui mecanism asincron de orchestrare a procesului de înregistrare a utilizatorilor, utilizând un model de tip Saga. Acesta gestionează tranzacțiile distribuite între microserviciile implicate (autentificare și user management) cu posibilitatea de compensare în cazul unor eșecuri, asigurând astfel consistența sistemului fără blocarea componentelor.

Pentru a evita utilizarea polling-ului – o practică frecventă în aplicațiile web, dar inefficientă în contexte distribuite – platforma propune un sistem de notificare în timp real prin WebSocket, care informează frontend-ul asupra statusului final al unei operațiuni complexe (de exemplu, finalizarea procesului de înregistrare sau apariția unei erori).

De asemenea, o decizie semnificativă este integrarea serviciilor de autentificare gRPC într-un context frontend, cu ajutorul unui proxy gRPC-Web, permițând comunicarea directă din browser cu backend-ul, integrare ce asigură performanță ridicată și o compatibilitate extinsă, fără a compromite securitatea sau flexibilitatea dezvoltării.

3.3. Cerințele utilizatorului

Pentru definirea direcției de dezvoltare a platformei *UrHomie*, a fost realizată o analiză a nevoilor și așteptărilor utilizatorilor finali – atât clienți, cât și furnizori de servicii. Aceste cerințe au fost formulate în urma evaluării limitărilor soluțiilor existente și a tendințelor actuale în interacțiunea utilizator-software. Scopul este de a crea o aplicație care oferă nu doar funcționalitate, ci și o experiență coerentă, accesibilă și relevantă.

Cerințe ale clienților (consumatori de servicii)

- Posibilitatea de a crea rapid un cont și de a se autentifica în siguranță;
- Acces la un catalog organizat de servicii, cu opțiuni clare de filtrare și căutare;
- Acces la informații detaliate despre fiecare serviciu și furnizor (preț, durată, recenzii);
- Posibilitatea de a rezerva online un serviciu într-un interval convenabil;
- Confirmări clare și notificări despre starea solicitărilor și rezervărilor;
- Modalități simple și sigure de plată;
- Acces la un istoric al comenzilor și posibilitatea de a evalua experiențele anterioare;
- O interfață simplă și prietenoasă, adaptată și dispozitivelor mobile.

Cerințe ale furnizorilor de servicii

- Posibilitatea de a crea un cont specializat și de a furniza informații complete despre serviciile oferite;
- Un spațiu dedicat pentru administrarea serviciilor disponibile (adăugare, modificare, ștergere);
- Vizualizarea și gestionarea comenzilor primite, inclusiv posibilitatea de a accepta/refuza programări;
- Acces la evaluările primite și la statistici despre activitatea proprie;
- Flexibilitate în setarea disponibilității în calendar și a condițiilor de lucru;
- Protecția datelor proprii și transparență în relația cu clienții.

Cerințe generale pentru toți utilizatorii

- Interacțiune intuitivă și feedback vizual imediat pentru fiecare acțiune efectuată;
- Timp redus de așteptare pentru răspunsurile din aplicație;
- Siguranța datelor personale și confidențialitatea tranzacțiilor.

3.4. Arhitectura sistemului

Platforma *UrHomie* este concepută pe baza unei arhitecturi distribuite de tip microservicii, care permite separarea clară a responsabilităților, scalabilitate pe orizontală și o dezvoltare modulară, independentă între componente. În stadiul actual, sistemul include următoarele module principale: interfața cu utilizatorul (frontend), un reverse proxy de tip NGINX, un proxy de conversie gRPC-Web (Envoy), două microservicii (UserAuth și UserManagement), un broker de mesaje RabbitMQ și mai multe baze de date (MariaDB, Redis).

Arhitectura sistemului este reprezentată în diagrama de mai jos, care evidențiază interacțiunile directe dintre componente și fluxurile de date corespunzătoare:

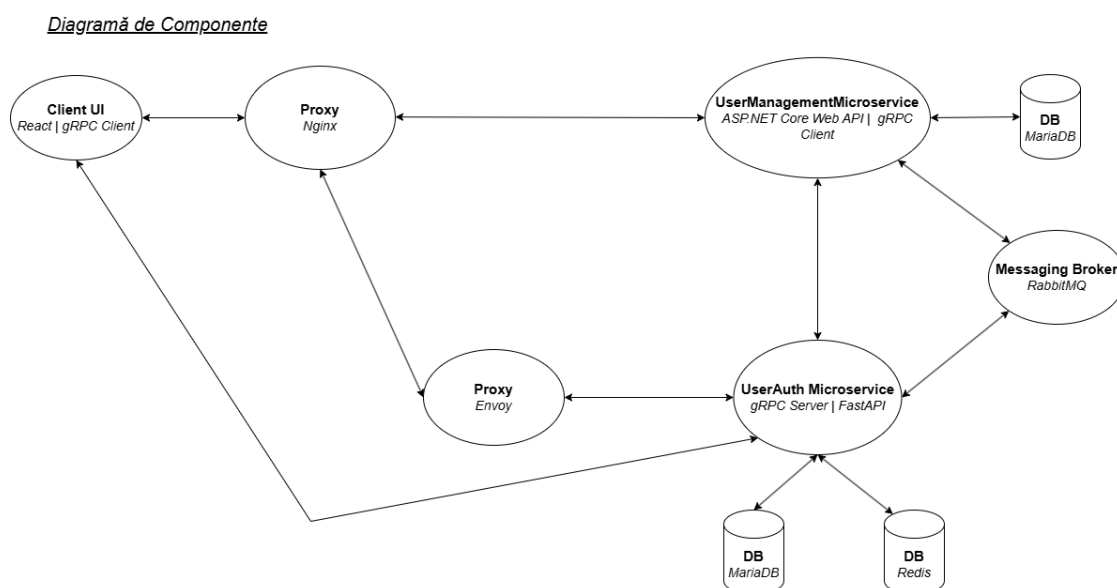


Figura 1: Diagrama logică a componentelor din platforma *UrHomie*

3.4.1. Componenta Frontend

Interfața utilizator (frontend-ul) din cadrul platformei *UrHomie* este dezvoltată sub forma unei aplicații single-page, construită utilizând biblioteca React. Aceasta permite o dezvoltare modulară și rapidă a componentelor, fiind ideală pentru aplicații web moderne care necesită interactivitate crescută și integrare strânsă cu servicii backend.

Organizare generală

Aplicația este structurată în funcție de funcționalități și rute:

- Landing page: punctul de pornire, cu opțiuni pentru login și creare cont;

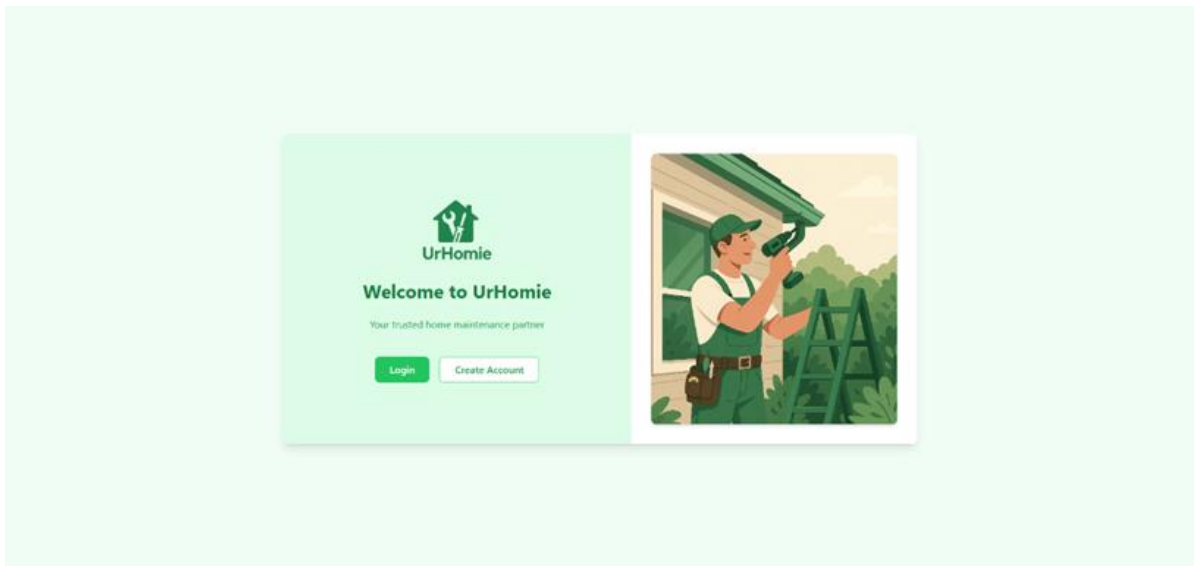


Figura 2: Landing Page

- Login și înregistrare: fluxuri distincte pentru clienți și furnizorii de servicii, în două etape, cu validări și sincronizări gRPC;

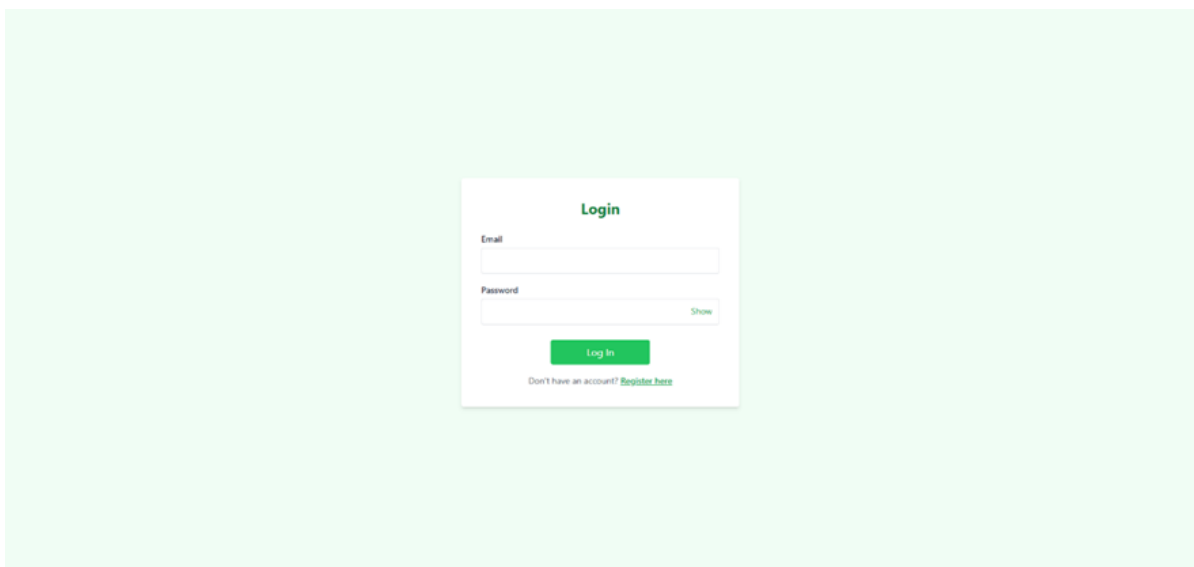


Figura 3: Login Form

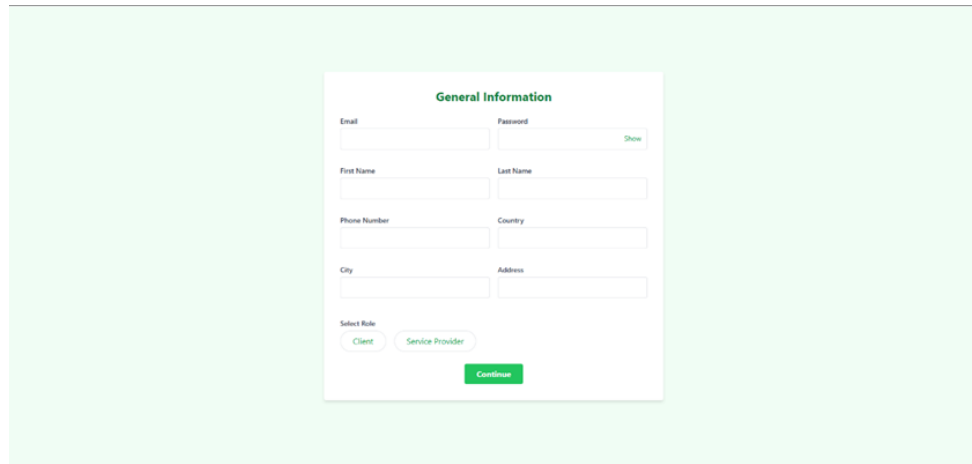
A screenshot of a 'General Information' form. The form is white with a green border and is centered on a light green background. It contains several input fields: Email, Password (with a 'Show' link), First Name, Last Name, Phone Number, Country, City, and Address. At the bottom, there are two radio buttons for 'Select Role': 'Client' and 'Service Provider', and a green 'Continue' button.

Figura 4: General Information Form

- HomePage: personalizată în funcție de rol (Client sau Service Provider), oferind funcționalități și conținut contextual;

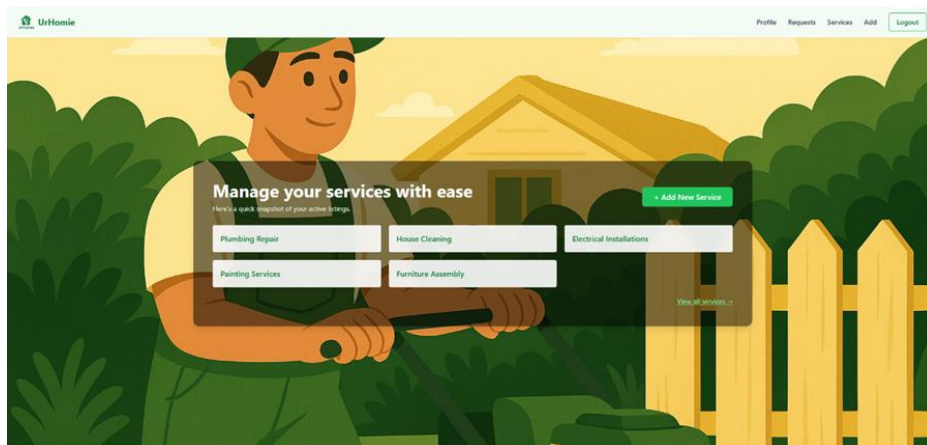


Figura 5: Service Provider Home Page

- Profilul utilizatorului: secțiune protejată, unde utilizatorul își poate vizualiza și edita datele.

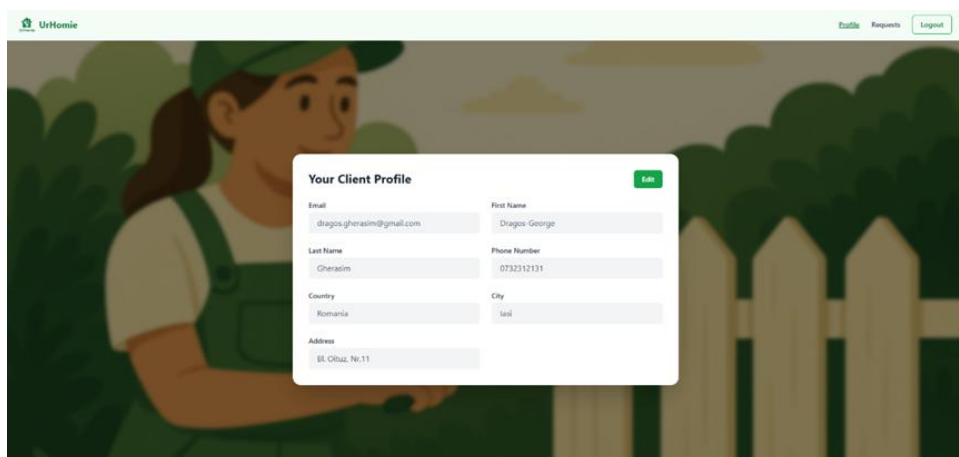
A screenshot of the 'Client Profile Page'. The page shows a profile card for 'Your Client Profile' with an 'Edit' button. The card contains the following information: Email (dragos.gherasim@gmail.com), First Name (Dragos-George), Last Name (Gherasim), Phone Number (0732312131), Country (Romania), City (Iasi), and Address (Bl. Olbia, Nr.11). The background features a blurred illustration of a woman in a green cap and overalls.

Figura 6: Client Profile Page

Tehnologii și comunicare

Aplicația utilizează un set diversificat de tehnologii moderne pentru a asigura o comunicare eficientă între componente și o experiență de utilizare fluidă:

- *gRPC-Web* este folosit pentru interacțiunea dintre frontend și microserviciul de autentificare (UserAuth). Comunicarea se face printr-un reverse proxy NGINX, iar conversia între protocoalele HTTP/1.1 și HTTP/2 este realizată de Envoy, plasat între NGINX și serviciu;
- *REST API* este utilizat pentru gestionarea profilului utilizatorului, prin cereri trimise cu Axios către microserviciul UserManagement. Accesul este securizat cu ajutorul unui token JWT.
- *WebSocket* asigură actualizarea în timp real a stării înregistrării utilizatorului. Frontend-ul se conectează la un canal WebSocket dedicat, iar fiecare sesiune este identificată printr-un *correlationId*, permițând recepționarea notificărilor legate de finalizarea procesului;
- *Tailwind CSS* este responsabil pentru stilizarea interfeței aplicației, oferind un design modern, coerent și responsiv, adaptabil pentru orice tip de dispozitiv;
- *Rutele protejate* (Protected Routes) garantează accesul controlat la anumite pagini. Acestea sunt disponibile doar utilizatorilor autentificați, iar accesul este validat prin verificarea prezenței unui token valid și a unui rol activ în contextul aplicației.

Context global de autentificare

Aplicația gestionează autentificarea utilizatorilor printr-un context global numit *AuthContext*, care centralizează și oferă acces facil la informațiile esențiale ale sesiunii. Acest context menține în memorie tokenul de acces (*accessToken*), ID-ul utilizatorului și rolul asociat, permițând componentelor aplicației să utilizeze aceste date fără redundanță. La fiecare pornire a aplicației, *AuthContext* verifică automat validitatea sesiunii și, dacă este cazul, reîmprospătează tokenul folosind *refresh token*-ul stocat în cookie, printr-un apel *gRPC* către microserviciul de autentificare. De asemenea, contextul expune metode globale pentru *login* și *logout* și asigură redirectionarea automată către pagina de start (*/landing*) în situațiile în care autentificarea este pierdută sau expirată.

Fluxuri funcționale majore

Procesul de autentificare începe cu introducerea adresei de email și a parolei de către utilizator, urmată de o validare locală a datelor. Dacă acestea sunt corecte, se trimite un apel *gRPC* către metoda *LogIn*, iar în cazul unui răspuns valid, tokenul este stocat și utilizatorul este redirectionat către pagina principală (*/home*).

Înregistrarea unui client presupune completarea unui formular general, urmată de trimiterea datelor prin apelul *SignUp*. După primirea unui *correlationId*, aplicația deschide o conexiune *WebSocket* pentru a asculta în timp real notificarea privind finalizarea procesului de înregistrare. În funcție de rezultat, utilizatorul este redirectionat sau i se afișează un mesaj de eroare.

Pentru înregistrarea ca service provider, procesul se desfășoară în doi pași: mai întâi se completează formularul general, apoi cel cu detalii suplimentare specifice rolului. După trimiterea completă a datelor printr-un apel *gRPC*, aplicația ascultă finalizarea înregistrării prin același mecanism *WebSocket* bazat pe *correlationId*. (Vezi Anexa 1)

Accesarea paginii de profil declanșează o cerere GET care utilizează tokenul JWT pentru autorizare. Datele utilizatorului sunt apoi afișate într-un formular editabil, iar modificările efectuate sunt trimise la server printr-un request PATCH la salvarea formularului.

Funcționalități vizuale

Aplicația oferă o interfață vizuală interactivă, prietenoasă și ușor de navigat, construită pentru a răspunde nevoilor utilizatorilor în funcție de rolul acestora. Formularele sunt dinamice și includ validări în timp real, oferind o experiență fluidă și intuitivă la completare.

Pagina de start (Home) este personalizată în funcție de rol: clienții pot căuta și explora prestatori de servicii, în timp ce prestatorii au acces la funcționalități pentru gestionarea serviciilor oferite. Secțiunea de profil este complet editabilă, cu afișarea și structurarea câmpurilor în funcție de tipul utilizatorului. De asemenea, aplicația oferă feedback instant pentru toate acțiunile importante, prin elemente vizuale precum indicatori de încărcare (loading spinner) și mesaje clare de succes sau eroare.

Securitate și protecție

Aplicația implementează măsuri clare de securitate pentru protejarea sesiunii utilizatorilor. Tokenul de acces este stocat exclusiv în memorie, în timp ce refresh token-ul este salvat în cookie, o abordare menită să reducă riscul atacurilor de tip XSS. Accesul la rutele sensibile este controlat prin mecanismul ProtectedRoute, care verifică în mod constant prezența unui token valid și a unui rol autorizat. În plus, interfața previne executarea acțiunilor nepermise și redirecționează automat utilizatorul către pagina de autentificare în cazul în care sesiunea expiră sau tokenul devine invalid.

3.4.2. Componente proxy în arhitectura sistemului

În arhitectura propusă, aplicația beneficiază de un mecanism de rutare și conversie format din două componente proxy: NGINX și Envoy, fiecare având un rol specific în facilitarea comunicării dintre interfața frontend și microserviciile backend.

NGINX – Reverse Proxy principal și handler CORS

NGINX reprezintă punctul central de acces pentru toate cererile provenite din browser, acționând ca un reverse proxy care facilitează comunicarea dintre frontend și serviciile backend. Acesta se ocupă de direcționarea cererilor către serviciile corespunzătoare, aplică politicile CORS necesare pentru a permite interacțiunea între domenii (întrucât aplicația React este separată de backend) și asigură compatibilitatea cu conexiunile WebSocket utilizate în procesul de înregistrare. Prin intermediul NGINX, aplicația beneficiază de o separare clară între interfața utilizatorului și logica de business, obținând astfel o gestionare centralizată și controlată a rutelor.

Envoy – Convertor de protocol pentru gRPC-Web

Pentru a permite apeluri gRPC din browser, care nu suportă în mod direct protocolul HTTP/2, aplicația utilizează Envoy ca proxy intermediar specializat în conversia de protocoale. Rolul său principal este de a transforma cererile gRPC-Web trimise de frontend, compatibile cu browserele moderne, în cereri gRPC native care pot fi procesate de microserviciul de autentificare, acesta din urmă funcționând exclusiv pe HTTP/2.

Astfel, Envoy asigură o compatibilitate transparentă între client și server, păstrând avantajele gRPC precum performanța ridicată și respectarea strictă a contractelor definite prin proto. Datorită filtrului integrat pentru gRPC-Web, Envoy realizează automat conversia cererilor și răspunsurilor fără a necesita modificări în logica serviciului backend. Această componentă devine esențială pentru funcționarea fiabilă a comunicației gRPC într-un context bazat pe browser.

3.4.3. Componenta UserAuth Microservice

UserAuth Microservice oferă o interfață gRPC nativă, expusă printr-un server grpc.aio, responsabilă de operațiile de autentificare și înregistrare. Separat, componenta de notificare în timp real pentru confirmarea înregistrării este implementată folosind FastAPI, care gestionează conexiunile WebSocket. Procesul de creare a conturilor este orchestrat asincron, pe baza unui pattern de tip Saga, cu ajutorul RabbitMQ (message broker) și Redis (pentru urmărirea stării).

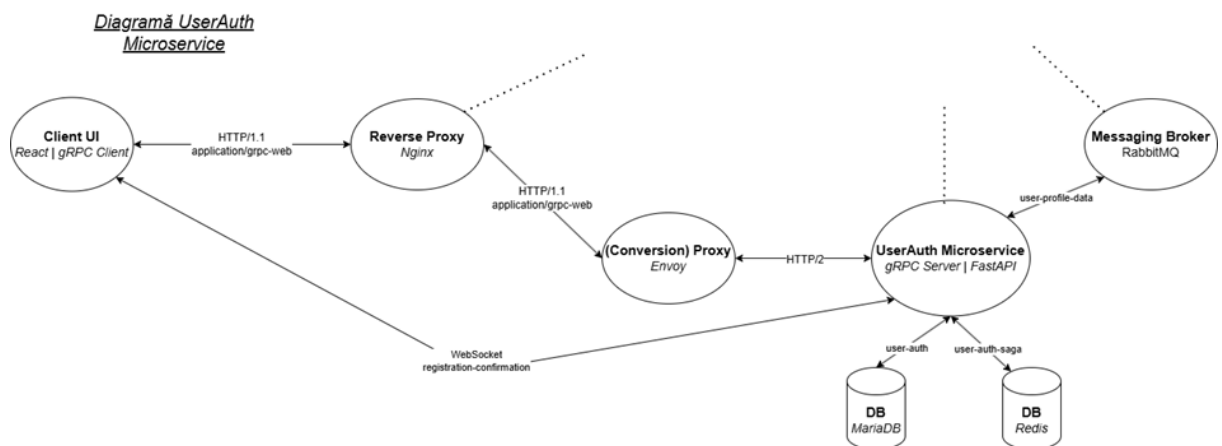


Figura 7: Diagrama UserAuth Microservice

Structura și responsabilități

Microserviciul UserAuth este responsabil de gestionarea autentificării și a sesiunii utilizatorilor, expunând cinci metode gRPC esențiale.

Metoda *LogIn* se ocupă de validarea credențialelor și returnează un access token (JWT), în timp ce refresh token-ul este livrat ca un cookie securizat, marcat HTTP-only pentru a preveni accesul JavaScript-ului.

SignUp inițiază procesul de creare a unui nou cont de utilizator și publică un mesaj în RabbitMQ, semnalând microserviciului UserManagement necesitatea de a crea profilul asociat.

Prin metoda *ValidateJwt*, aplicația poate verifica dacă un token de acces este valid, inclusiv identitatea utilizatorului și rolul acestuia.

RefreshToken extrage și validează refresh token-ul din cookie și generează un nou access token în mod automat.

Metoda *LogOut* asigură închiderea sesiunii prin ștergerea explicită a cookie-ului care conține token-ul de reîmprospătare. Întregul proces de înregistrare este coordonat printr-un *correlationId* unic, generat la începutul operației, persistat în Redis și utilizat ulterior pentru urmărirea statusului acesteia în timp real.

Orchestrarea înregistrării și gestiunea statusului (Saga)

Procesul de orchestrare a înregistrării este gestionat printr-un mecanism de tip Saga, care coordonează pașii necesari creării complete a unui utilizator și a profilului său. După crearea inițială a contului, microserviciul inițiază o instanță Saga în Redis, setând statusul operațiunii ca „pending”. Ulterior, publică un mesaj *CreateUserProfileMessage* pe un exchange de tip fanout în RabbitMQ, conform specificației oferite de MassTransit, pentru a notifica microserviciul UserManagement despre necesitatea creării profilului.

În continuare, microserviciul ascultă evenimentele de răspuns, care pot fi fie *UserProfileCreated*, fie *UserProfileCreationFailed*. În funcție de mesajul primit, Saga-ul este actualizat în Redis, fie marcând statusul ca „completed”, fie ca „failed”. În caz de eșec, sistemul declanșează automat ștergerea contului de utilizator creat anterior, pentru a evita inconsistențele.

Pe partea de frontend, aplicația ascultă aceste modificări de stare printr-un canal WebSocket expus de microserviciul de autentificare, la ruta */user-auth/registration-status*, filtrând notificările primite în funcție de *correlationId*, pentru a identifica sesiunea relevantă fiecărui utilizator. (Vezi Anexa 2)

Modelarea datelor

Tabelul principal gestionat de acest microserviciu este *user_account*, iar structura sa este ilustrată mai jos. Acesta include următoarele atribute:

- *id*: identificator unic de tip BIGINT;
- *email*: adresa utilizatorului (unică);
- *password*: parolă hash-uită cu bcrypt;
- *role*: valoare ENUM (CLIENT, SERVICE_PROVIDER), folosită în autorizare.

Diagramă Entitate-Relație
UserAuth Microservice

user-account	
PK	ID:BIGINT
VARCHAR	email
VARCHAR	password
ENUM (CLIENT, SERVICE_PROVIDER)	role

Figura 8: Diagrama Entitate-Relație UserAuth Microservice

Securitate

Microserviciul UserAuth integrează mai multe mecanisme de securitate pentru a proteja sesiunea utilizatorului și a preveni accesul neautorizat. La autentificare, access token-ul este returnat direct în răspunsul gRPC și este stocat temporar în memorie pe partea de frontend, evitând astfel expunerea în medii nesigure. Refresh token-ul este transmis separat, sub forma unui cookie securizat (HTTP-only), configurat cu attributele *SameSite=Lax* și *Path=/*, pentru a reduce riscul de atacuri de tip CSRF. Validarea token-urilor este implementată riguros, cu tratamente distincte pentru access token și refresh token, în funcție de scopul fiecăruia. În situațiile în care un token este expirat, invalid sau corupt, microserviciul răspunde automat cu un status *UNAUTHENTICATED*, semnalizând necesitatea reautentificării.

3.4.4. Componenta UserManagement Microservice

Microserviciul UserManagement este responsabil pentru gestionarea datelor extinse de profil ale utilizatorilor, inclusiv pentru clienți și furnizori de servicii. Acesta expune un API REST construit în ASP.NET Core și acționează ca consumator de mesaje RabbitMQ pentru procesul asincron de creare a profilului, precum și ca client gRPC pentru validarea token-urilor JWT prin microserviciul UserAuth.

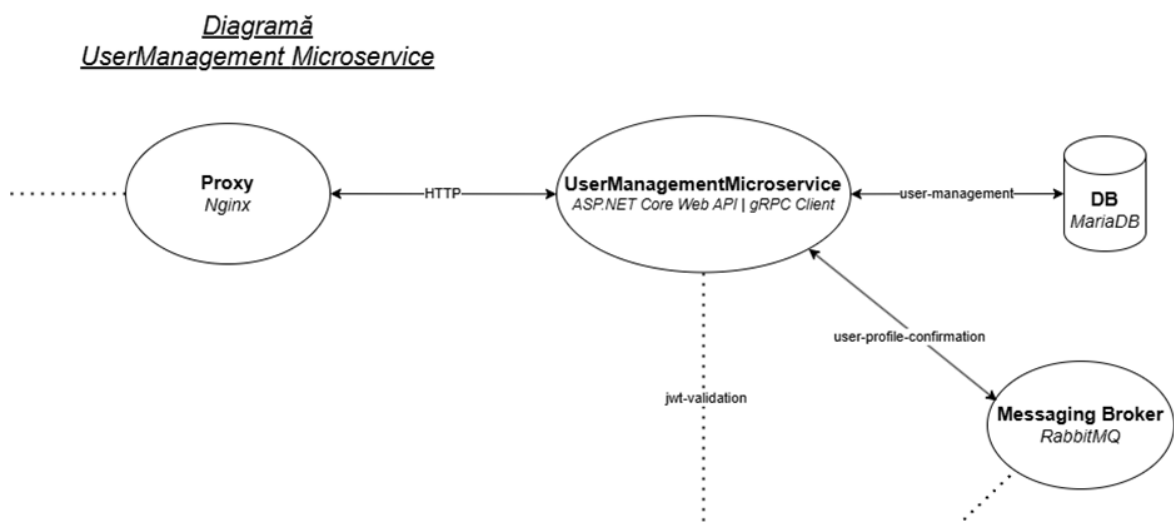


Figura 9: Diagrama UserManagement Microservice

Funcționalități principale

Microserviciul UserManagement oferă funcționalități esențiale pentru gestionarea profilurilor de utilizator, integrând atât operații sincrone, cât și asincrone. Crearea profilului se realizează automat, ca răspuns la un mesaj primit de la UserAuth în urma înregistrării unui nou utilizator. Interfața aplicației poate accesa aceste profiluri prin endpointuri REST securizate, de tip GET și PATCH, disponibile atât pentru clienți, cât și pentru prestatori de servicii. Pentru validarea autentificării, UserManagement utilizează un client gRPC care apelează metoda *ValidateJwt* din microserviciul UserAuth. Autorizarea este implementată granular, pe baza rolului și identității utilizatorului, folosind politici personalizate precum *SameClientOnly* și *SameServiceProviderOnly*.

Integrare RabbitMQ și MassTransit

Integrarea cu RabbitMQ și MassTransit permite procesarea asincronă a fluxului de creare a profilului. La primirea mesajului *CreateUserProfileMessage* din partea UserAuth, microserviciul determină tipul utilizatorului (Client sau ServiceProvider), creează entitatea corespunzătoare în baza de date și transmite înapoi un eveniment de răspuns, fie *UserProfileCreated*, fie *UserProfileCreationFailed*, utilizând același *CorrelationId*. Pentru această comunicare, microserviciul se abonează la exchange-ul *create_user_profile_event* printr-un *ReceiveEndpoint* numit *create_user_profile_queue*, iar procesarea mesajelor este realizată prin consumatorul *CreateUserProfileConsumer*. Răspunsurile sunt publicate în exchange-uri de tip fanout, *user_profile_created_event* și *user_profile_failed_event*, asigurând suport complet pentru un pattern asincron de tip Request-Reply. Toate mesajele sunt marcate cu un *CorrelationId*, care permite microserviciului UserAuth să coreleze statusul înregistrării cu sesiunea corespunzătoare din frontend.

Modelarea datelor

La nivel de modelare a datelor, baza de date este centrată în jurul entității *user_profile*, care conține attribute comune tuturor utilizatorilor, precum email, nume, prenume, număr de telefon, țară, oraș și adresă. Această entitate este extinsă într-o relație de tip unu-la-unu de către entitățile *client* și *service_provider*. Pentru prestatori, profilul include câmpuri specifice precum educație, certificări, descrieri ale experienței, program de lucru, arie de acoperire și lista de categorii de servicii (*category_ids*). În cazul utilizatorilor de tip client, structura este deja pregătită pentru a susține viitoare extensii, însă nu conține momentan attribute suplimentare.

*Diagramă Entitate-Relație
UserManagement
Microservice*

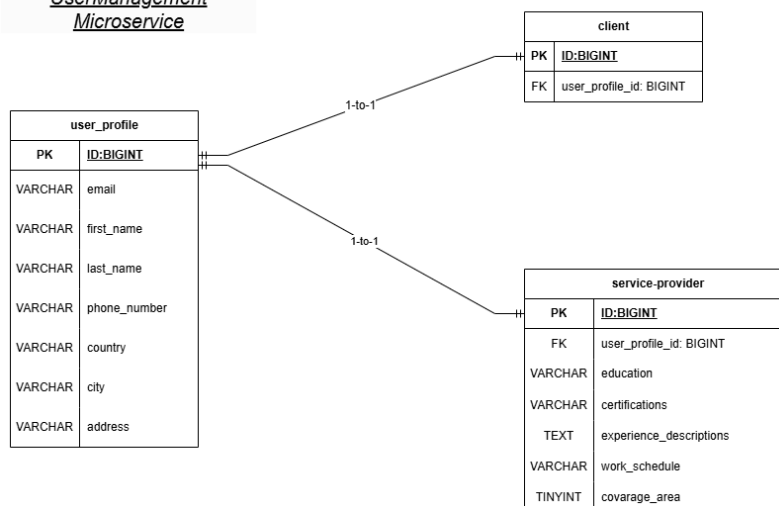


Figura 13: Diagrama Entitate-Relație UserManagement Microservice

Validări și actualizări parțiale (PATCH)

Actualizarea profilului de utilizator este realizată prin metoda HTTP PATCH, utilizând DTO-uri dedicate precum *UpdateClientDto* și *UpdateServiceProviderDto*. Sistemul aplică o strategie flexibilă de validare, concentrându-se exclusiv pe câmpurile efectiv furnizate în request, prin folosirea atributului *ValidateIfPresent*. Astfel, utilizatorii pot modifica doar datele dorite, fără a fi obligați să trimită întregul profil. În cazul câmpurilor sensibile precum email sau număr de telefon, sistemul impune unicitatea valorilor. Dacă se detectează un conflict cu valorile existente în

baza de date, este returnat un răspuns de tip HTTP 409 Conflict. De asemenea, se asigură că request-ul include cel puțin un câmp valid de actualizat, printr-o extensie custom numită *HasAtLeastOnePopulatedField*.

Autentificare și autorizare

Pentru autentificare și autorizare, microserviciul nu validează local token-urile JWT. În schimb, fiecare token este transmis către microserviciul UserAuth printr-un apel gRPC intern. Acest proces este gestionat de un middleware personalizat, *ExternalJwtValidationMiddleware*, care injectează un obiect *ClaimsPrincipal* în contextul cererii doar dacă token-ul este valid. Pe baza acestor claims, se aplică politici de autorizare granulară. Politica *SameClientOnly* permite accesul doar dacă identificatorul (*sub*) din token corespunde ID-ului specificat în URL și rolul este *CLIENT*. O regulă similară, *SameServiceProviderOnly*, se aplică în cazul prestatorilor de servicii (*SERVICE_PROVIDER*), asigurând că doar utilizatorii autorizați pot efectua modificări asupra propriului profil.

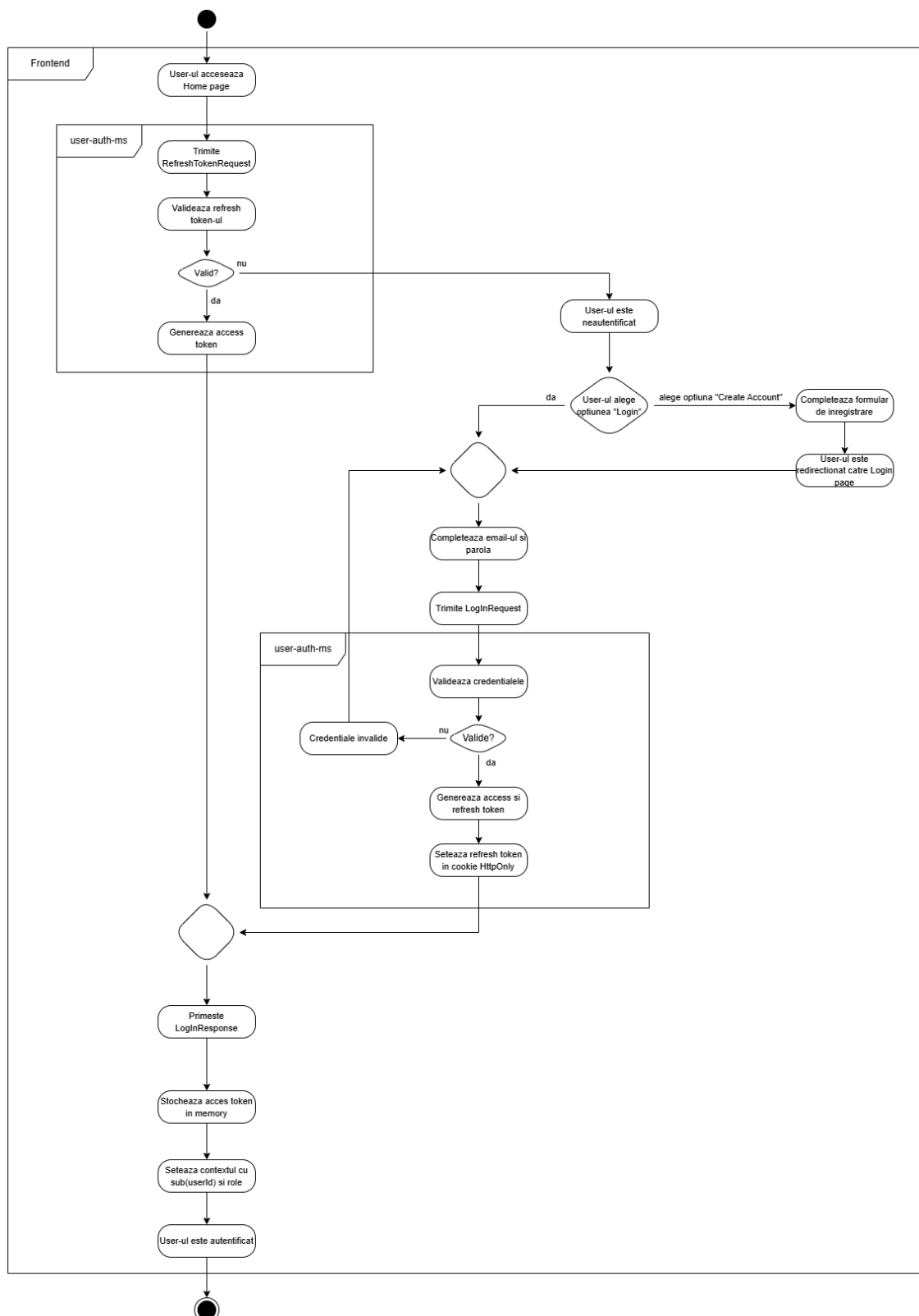
Bibliografie

- [1] Issue Monitoring, „Romania’s Digital Environment: Navigating the Path to a Tech-DrivenFuture”, <https://issuemonitoring.eu/en/romantias-digital-environment-navigating-the-path-to-a-tech-driven-future/>
- [2] OECD, „Digital Government Review of Romania”, https://www.oecd.org/en/publications/digital-government-review-of-romania_68361e0d-en.html
- [3] Atlassian, „Advantages of microservices and disadvantages to know”, <https://www.atlassian.com/microservices/cloud-computing/advantages-of-microservices>
- [4] plego, „The Importance of User Interface Design”, <https://plego.com/blog/importance-of-user-interface-design/>
- [5] TRAILHEAD, „When to Use a Distributed Architecture—And When Not”, <https://trailheadtechnology.com/when-to-use-a-distributed-architectureand-when-not/>
- [6] OSO, „13 Microservices Best Practices”, <https://www.osohq.com/learn/microservices-best-practices>
- [7] salesforce, „6 Fundamental Principles of Microservice Design”, <https://www.salesforce.com/blog/microservice-design-principles/>
- [8] DigitalOcean, „SOLID: The First 5 Principles of Object Oriented Design”, <https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>
- [9] komprise, „REST (Representational State Transfer)”, https://www.komprise.com/glossary_terms/rest-representational-state-transfer/
- [10] Journal of Student Research, „Home Repairs: Mobile Application for Home Maintenance Services”, <https://www.jsr.org/index.php/path/article/view/1500/>
- [11] <https://home-maintenance.ro/>
- [12] <https://servicii24.ro/>
- [13] <https://www.taskrabbitt.com/>
- [14] <https://www.handy.com/>

Anexe

Anexa 1: Diagrama de activitate pentru procesul de autentificare și acces bazat pe tokenuri

Diagrama de activitate pentru procesul de autentificare și acces bazat pe tokenuri



Anexa 2: Diagrama de activitate pentru procesul de înregistrare distribuit

Diagrama de activitate pentru procesul de înregistrare distribuit

