



PLATFORMĂ INTEGRATĂ PENTRU GESTIONAREA ȘI VIZUALIZAREA CURSURILOR VALUTARE

PROIECT

Contribuitori: **Bogdan-Alexandru BÂRGĂOANU**
Diana-Susana FORRAI

Coordonator **Prof. dr. ing. Ioan-Valentin SITA**
științific:

2024



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ

**PLATFORMĂ INTEGRATĂ PENTRU GESTIONAREA ȘI
VIZUALIZAREA CURSURILOR VALUTARE**

1. **Enunțul temei:** *Dezvoltarea unei aplicații web full stack care să permită monitorizarea în timp real și gestionarea cursurilor de schimb valutar, incluzând un API pentru preluarea și stocarea datelor, o interfață client pentru utilizatori finali și un panou de administrare pentru gestionarea resurselor și configurărilor sistemului.*
2. **Conținutul lucrării:** *Pagina de prezentare, introducere, obiectivele lucrării, studiu bibliografic, analiză teoretică și sinteză, Proiectare de detaliu și implementare, manual de instalare și utilizare, concluzii, bibliografie, anexe.*
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca
4. **Data emiterii temei:** 30 septembrie 2024
5. **Data predării:** 20 decembrie 2024

Bârgăoanu Bogdan-Alexandru

Contribuitori:

Forrai Diana-Susana

Coordonator științific:

Cuprins

Capitolul 1. Introducere	1
Capitolul 2. Obiectivele proiectului.....	2
Capitolul 3. Studiu bibliografic	4
Capitolul 4. Analiză teoretică și sinteză.....	5
4.1. Aplicația Server / API.....	5
4.1.1. Structura bazei de date.....	5
4.1.2. Relații între tabele:.....	5
4.1.3. Securitatea datelor	6
4.1.4. Funcționalități suplimentare	7
4.2. Aplicația Administrator	8
4.2.1. Structura aplicației	8
4.2.2. Principalele componente React	8
4.2.3. Funcționalități suplimentare	8
4.3. Aplicația Client.....	9
4.3.1. Structura aplicației	9
4.3.2. Principalele componente React	9
4.3.3. Funcționalități suplimentare	10
Capitolul 5. Proiectare de detaliu si implementare	11
5.1. Plan de proiectare și implementare.....	11
5.2. Aplicația Server / API.....	12
5.3. Aplicația Administrator	14
5.4. Aplicația Client.....	19
Capitolul 6. Manual de instalare și utilizare	24
Capitolul 7. Concluzii	26
Bibliografie	28
Anexe	29

Capitolul 1. Introducere

Acest proiect propune dezvoltarea unei aplicații web full-stack pentru monitorizarea și gestionarea cursurilor de schimb valutar. Soluția integrează trei componente principale: un API backend care gestionează cererile și interacțiunile cu baza de date, o interfață client destinată utilizatorilor finali pentru vizualizarea cursurilor valutare în timp real și o interfață de administrare pentru gestionarea resurselor și configurarea sistemului.

Aplicația utilizează tehnologii moderne precum *Node.js* și *Express.js* pentru backend, *React.js* pentru componentele frontend, și *MySQL* pentru stocarea datelor. Funcționalitățile sunt extinse prin utilizarea *JWT* pentru securizarea API-urilor și a autentificării, precum și prin integrarea de grafice și hărți pentru o experiență interactivă a utilizatorului.

Sistemul trebuie să ofere un grad ridicat de configurabilitate, permițând personalizarea aplicației în funcție de nevoile fiecărei societăți utilizatoare. Administratorii pot configura locațiile în mod dinamic, asociind fiecare rată valutară unei locații specifice. Această flexibilitate permite urmărirea precisă a cursurilor valutare în diferite regiuni, asigurând o gestionare detaliată a datelor financiare.

Scopul principal al platformei este de a oferi o soluție simplificată și accesibilă pentru orice utilizator, având ca obiectiv să funcționeze ca o platformă gratuită dedicată clienților. Utilizatorii pot căuta și selecta cursurile valutare de interes, beneficiind de opțiuni de filtrare avansate. Platforma permite afișarea cursurilor valutare disponibile, fie în ansamblu, fie în funcție de proximitatea față de locația utilizatorului, utilizând coordonatele GPS pentru a identifica punctele de schimb valutar din apropiere. În plus, aplicația facilitează accesul la cele mai avantajoase rate de schimb, oferind o experiență intuitivă și eficientă.

Această soluție adresează necesitatea unei platforme centralizate pentru urmărirea cursurilor de schimb valutar, oferind atât accesibilitate, cât și o modalitate eficientă de administrare a datelor financiare.

Capitolul 2. Obiectivele proiectului

În contextul globalizării economice și al interdependenței financiare, cunoașterea și monitorizarea cursurilor de schimb valutare reprezintă un aspect esențial pentru instituțiile financiare, companiile de comerț internațional și chiar pentru utilizatorii individuali care doresc să își gestioneze mai eficient tranzacțiile financiare. De asemenea, în era digitală, accesul rapid și ușor la aceste informații devine din ce în ce mai important, iar tehnologiile moderne oferă posibilitatea de a construi platforme care să furnizeze date actualizate, ușor accesibile și personalizabile. Astfel, tema acestui proiect se concentrează pe dezvoltarea unei platforme complete, capabile să urmărească și să stocheze datele referitoare la cursurile de schimb valutar, să ofere un sistem de administrare al acestora și să permită vizualizarea acestora prin intermediul unei interfețe client intuitivă și interactivă.

Obiectivele proiectului:

1. Dezvoltarea unei aplicații de tip full-stack:

- Crearea unui backend robust care să gestioneze API-ul de acces la datele valutare și să interacționeze eficient cu baza de date *MySQL*.
- Crearea unui frontend intuitiv, utilizând *React.js*, care să permită utilizatorilor să vizualizeze cursurile valutare actualizate și să le selecteze în funcție de locație și de preferințele lor.

2. Gestionarea și actualizarea dinamică a datelor valutare

- Permite administratorilor să adauge, modifice sau să elimine cursurile de schimb și locațiile asociate acestora printr-o aplicație de administrare construită cu *React.js*.
- Asigurarea unui proces de actualizare în timp real a cursurilor de schimb, cu date provenite din surse externe și actualizări periodice automate.

3. Implementarea funcționalității de geolocalizare și personalizare a experienței utilizatorului

- Permite utilizatorilor să vizualizeze cursurile de schimb cele mai apropiate de locația lor, utilizând funcționalitățile de geolocalizare din cadrul aplicației frontend client.
- Oferirea posibilității de a selecta cele mai bune rate de schimb, având în vedere atât locația, cât și preferințele de monedă ale utilizatorului.

4. Securizarea aplicației

- Implementarea unui sistem de autentificare și autorizare bazat pe *JWT* (JSON Web Tokens) pentru a proteja accesul la API-uri și pentru a asigura un nivel ridicat de securitate a datelor.
- Protejarea parolelor utilizatorilor prin criptare folosind algoritmi de tip *MD5*.

5. Crearea unui sistem de rapoarte și vizualizări

- Integrarea de grafice și hărți interactive pentru vizualizarea evoluției cursurilor de schimb pe perioade determinate, utilizând biblioteci precum *React Google Charts* și *Google Maps*.

- Oferirea posibilității de a genera rapoarte personalizate pe baza locației și monedei alese de utilizator sau administrator.

6. Scalabilitate și performanță

- Asigurarea unei arhitecturi scalabile, capabile să susțină un număr mare de utilizatori și tranzacții simultane, prin optimizarea interacțiunilor cu baza de date și a performanței aplicației frontend.
- Utilizarea unui sistem performant de stocare a datelor pentru gestionarea și procesarea eficientă a cursurilor de schimb și a altor informații financiare.

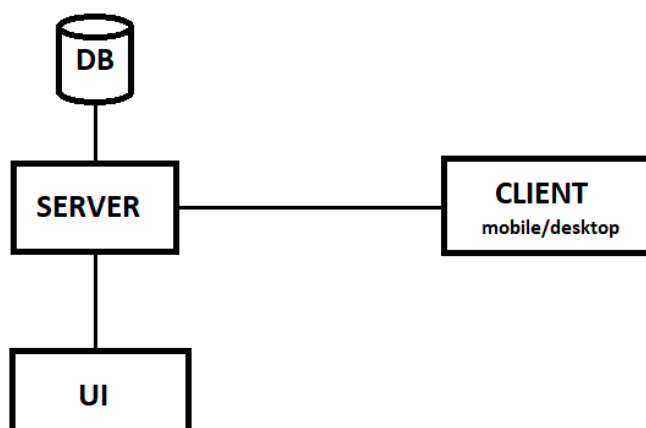


Figura 2.1. Arhitectura platformei

Metodologia de dezvoltare:

Proiectul va fi realizat folosind metodologia Agile, cu sprinturi de dezvoltare scurte și obiective clare. Fiecare componentă a aplicației va fi dezvoltată în paralel, urmând ca la final să fie integrate într-o soluție unitară. Testarea va fi realizată continuu, atât la nivel de unități, cât și prin teste de integrare și de performanță, pentru a asigura funcționarea corectă a întregii platforme. De asemenea, se va urmări integrarea eficientă a platformei cu surse externe de date pentru actualizarea cursurilor de schimb și a altor informații financiare relevante.

Rezultatele așteptate:

La finalizarea proiectului, se va obține o platformă complet funcțională, care va permite gestionarea și vizualizarea cursurilor de schimb valutare într-un mod ușor de utilizat, sigur și scalabil. Utilizatorii vor putea accesa datele în timp real, iar administratorii vor avea la dispoziție un sistem eficient pentru actualizarea și gestionarea acestora. Platforma va contribui la îmbunătățirea accesului la informații financiare relevante, oferind în același timp o soluție completă pentru monitorizarea și analiza cursurilor de schimb pe baza locației și a altor parametri relevanți.

Capitolul 3. Studiu bibliografic

Tema monitorizării și gestionării cursurilor de schimb valutar prin intermediul aplicațiilor software este de o relevanță deosebită în contextul economiei globale moderne. Într-un mediu economic caracterizat de volatilitate și interconectivitate, accesul rapid la informații despre cursurile de schimb valutar și capacitatea de a analiza aceste date devin esențiale pentru utilizatori individuali și organizații financiare.

Literatura de specialitate subliniază importanța dezvoltării de platforme care integrează tehnologii moderne pentru a sprijini utilizatorii în luarea deciziilor financiare. Conform lucrării lui Goodhart și Tsang (2020) [1], aplicațiile financiare bazate pe web și mobil au crescut exponențial, alimentate de progresele în tehnologiile cloud și în procesarea datelor în timp real. Aceste soluții permit utilizatorilor să obțină informații precise și actualizate, reducând astfel riscurile asociate tranzacțiilor internaționale.

O caracteristică esențială în proiectarea acestor aplicații este utilizarea interfețelor programabile de aplicații (API-uri) pentru a conecta surse externe de date. Potrivit lui Zhou și alții (2019) [2], utilizarea API-urilor pentru extragerea datelor valutare și integrarea acestora într-un sistem centralizat simplifică procesul de agregare și analiză a informațiilor financiare. În plus, securitatea devine un aspect critic, iar utilizarea tokenurilor de autentificare precum JWT asigură accesul controlat și protecția datelor utilizatorilor.

Din perspectiva utilizabilității, interfețele interactive bazate pe tehnologii precum React.js, menționate de Kumar și Sharma (2021) [3], sunt preferate datorită capacității lor de a oferi o experiență fluidă și adaptabilă pe diverse dispozitive. În același timp, stocarea și gestionarea datelor financiare necesită soluții robuste, iar baze de date relaționale precum MySQL sunt adesea alese datorită capacității lor de a gestiona volume mari de date structurate.

Un alt aspect evidențiat de Lee și Park (2022) [4] este integrarea funcționalităților legate de geolocalizare în aplicațiile financiare. Utilizarea coordonatelor GPS pentru a oferi utilizatorilor cele mai apropiate puncte de schimb valutar adaugă valoare serviciilor, făcând aplicațiile mai relevante și personalizate. De asemenea, funcționalitățile de generare a rapoartelor personalizate sunt considerate critice pentru utilizatorii care doresc să analizeze evoluția cursurilor valutare pe baza locației și a monedei.

Acest proiect își propune să integreze cele mai bune practici identificate în literatura de specialitate, combinând securitatea, accesibilitatea și flexibilitatea într-o soluție completă pentru gestionarea cursurilor de schimb valutar. Având în vedere creșterea cererii pentru astfel de soluții, dezvoltarea unei platforme centralizate, gratuite și configurabile reprezintă un pas important în sprijinirea utilizatorilor finali în procesul de luare a deciziilor financiare.

Capitolul 4. Analiză teoretică și sinteză

Platforma dezvoltată este un sistem distribuit bazat pe o arhitectură client-server care separă logic backend-ul (serverul) de componentele frontend. Această separare permite scalabilitate, securitate și o interfață de utilizare performantă. Fiecare componentă a sistemului utilizează protocoale moderne și algoritmi optimizați pentru a asigura funcționarea corectă, manipularea eficientă a datelor și protejarea acestora împotriva accesului neautorizat.

4.1. Aplicația Server / API

Aplicația server reprezintă nucleul sistemului și este responsabilă de gestionarea logicii de business, a interacțiunilor cu baza de date și a comunicării prin API-uri securizate cu aplicațiile frontend.

4.1.1. Structura bazei de date

Baza de date utilizată în cadrul platformei este *MySQL* și este proiectată să reflecte relațiile logice dintre parteneri, locații, cursurile de schimb și monede. Principalele tabele și relațiile dintre acestea sunt:

- *Partner*: Stochează informații despre parteneri (instituții sau companii) care operează schimburi valutare. Fiecare partener poate avea mai multe locații.
- *Location*: Reprezintă locațiile fizice asociate partenerilor. Fiecare locație poate oferi mai multe schimburi valutare.
- *Rate*: Înregistrează cursurile de schimb valutar pentru diverse locații, cu detalii precum valoarea ratei, moneda de bază și moneda țintă.
- *Currency*: Stochează informații despre monedele disponibile. Monedele sunt configurabile și pot fi adăugate în funcție de cerințele platformei.

4.1.2. Relații între tabele:

- Un partener (*Partner*) poate avea mai multe locații (*Location*).
- Fiecare locație (*Location*) poate oferi mai multe rate de schimb (*Rate*).
- Fiecare rată de schimb (*Rate*) implică o locație (*Location*), și o valoare corespunzătoare acesteia.

Această structură relațională asigură flexibilitate și extensibilitate, facilitând gestionarea eficientă a datelor valutare și asocierea acestora cu locații și parteneri specifici.

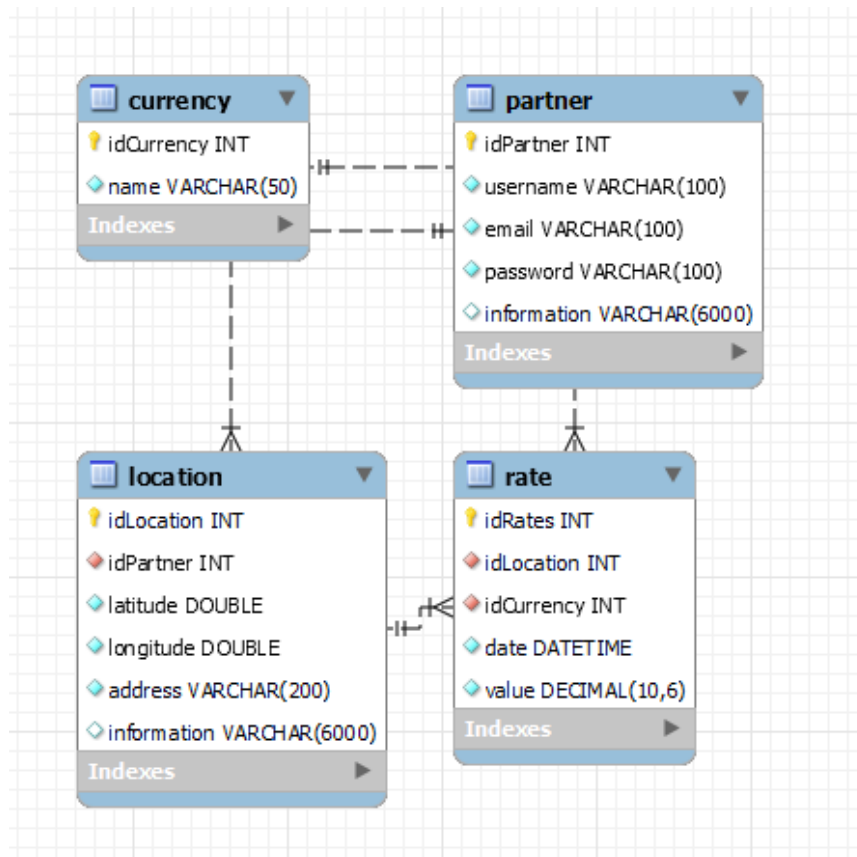


Figura 4.1: Diagrama bazei de date

4.1.3. Securitatea datelor

Pentru protejarea datelor utilizatorilor de tip administrator și a operațiunilor efectuate în platformă, au fost implementate următoarele măsuri:

- *Criptarea parolelor utilizatorilor:*
 - Parolele utilizatorilor de tip administrator sunt criptate folosind algoritmul *MD5* înainte de a fi stocate în baza de date.
 - Alegerea *MD5* a fost făcută pentru simplitate și viteza sa de execuție, având în vedere că această platformă are un nivel controlat de utilizatori administratori și nu procesează date critice personale ale clienților finali. Totuși, pentru medii mai stricte, poate fi considerată trecerea la algoritmi mai moderni, cum ar fi *bcrypt*.
 - Avantajul criptării este că niciun utilizator rău-intenționat nu poate obține direct parolele chiar dacă accesul la baza de date ar fi compromis.
- *Endpoint-uri securizate:*
 - Accesul la API-urile aplicației server este protejat prin utilizarea *JWT* (JSON Web Tokens).
 - *JWT* este configurat cu o cheie secretă unică, care este stocată în mod sigur și utilizată pentru a semna și valida token-urile. Aceasta permite autentificarea și autorizarea utilizatorilor, asigurând că doar cererile validate au acces la resursele API-ului.
 - Toate cererile către API necesită un token valid pentru a accesa endpoint-urile securizate, reducând riscul de acces neautorizat.

4.1.4. Funcționalități suplimentare

- Sistem de Paginație

Pentru a optimiza răspunsurile API și a evita supraîncărcarea aplicației cu volume mari de date, s-a implementat un sistem de paginație la nivelul endpoint-urilor API. Acest mecanism permite livrarea datelor în blocuri controlate, îmbunătățind performanța aplicației și experiența utilizatorului.

Paginația este configurată prin intermediul parametrilor *page* (pagina curentă) și *limit* (numărul de înregistrări pe pagină). Logica de paginație constă în:

- Calcularea offset-ului pe baza paginii curente:

$$offset = (page - 1) \times limit \quad (4.1.1)$$

- Adăugarea unei limite fixe (*LIMIT*) și a offset-ului (*OFFSET*) în cadrul interogării SQL pentru a returna doar setul de date corespunzător paginii solicitate.
- Structurarea răspunsului API astfel încât să includă nu doar rezultatele, ci și metainformații despre pagina curentă și starea operațiunii (succes/eroare).

- Gestionarea Geolocației

O altă funcționalitate importantă implementată în aplicația server este suportul pentru geolocație. Aceasta permite identificarea cursurilor de schimb valutar disponibile în locațiile cele mai apropiate de utilizator, folosind coordonatele GPS furnizate (latitudine și longitudine).

Acest concept este realizat cu ajutorul formulei **Haversine**, care calculează distanța dintre două puncte pe suprafața Pământului utilizând coordonatele lor geografice. Formula este definită matematic astfel:

$$d = 2 \times R \times \sin^{-1} \sqrt{\sin^2 \frac{(\phi_1 - \phi_2)}{2} + \cos \phi_1 \times \cos \phi_2 \times \sin^2 \frac{(\lambda_1 - \lambda_2)}{2}} \quad (4.1.2)$$

Unde:

d : distanța dintre cele două puncte.

R: raza Pământului (aproximativ 6371 km).

ϕ : latitudinile punctelor.

λ : longitudinile punctelor.

Aceste măsuri de securitate, combinate cu o structură bine gândită a bazei de date, asigură un sistem sigur și fiabil, protejând datele și funcționalitatea platformei împotriva amenințărilor.

4.2. Aplicația Administrator

Aplicația dedicată administratorilor este componenta frontend a platformei care permite utilizatorilor de tip partener să gestioneze datele asociate activităților lor, cum ar fi locațiile, cursurile de schimb valutar și monedele disponibile. Această aplicație este implementată folosind *React.js*, fiind o aplicație de tip *Single Page Application (SPA)* ce utilizează un router pentru navigare eficientă între componente.

4.2.1. Structura aplicației

Aplicația administrator include următoarele funcționalități principale:

- *Pagina de autentificare (Login/Register):*
O interfață simplistă permite utilizatorilor să se autentifice sau să își creeze un cont nou. După autentificare, utilizatorii primesc un token *JWT* care este utilizat pentru gestionarea sesiunilor și autorizarea operațiunilor ulterioare.
- *Interfața principală:*
După autentificare, aplicația afișează o interfață cu un **navbar** pentru navigarea prin principalele secțiuni ale aplicației.

4.2.2. Principalele componente React

- *Dashboard:*
Această secțiune oferă o imagine de ansamblu asupra activității partenerului.
Include grafice interactive utilizând *React Google Charts* care afișează:
 1. Locațiile asociate partenerului logat.
 2. Valorile cursurilor valutare corespunzătoare locațiilor.
- *Location:*
Permite adăugarea și gestionarea locațiilor asociate partenerului logat.
Include funcționalitatea de generare automată a unui *preview Google Maps* pe baza adresei introduse. Oferă coordonatele GPS (latitudine și longitudine) ale locației după validarea adresei.
- *Partners:*
Afișează date publice despre toți partenerii înscrși pe platformă. Este o secțiune informativă utilă pentru partenerii care doresc să colaboreze sau să își facă o idee despre concurență.
- *Currency:*
Administrează monedele disponibile pe platformă. Oferă funcționalități de vizualizare, adăugare, modificare și ștergere a monedelor.
- *Rates:*
Permite adăugarea și gestionarea cursurilor de schimb valutar. Cursurile sunt configurate pentru locațiile asociate partenerului logat și monedele existente.

4.2.3. Funcționalități suplimentare

- *Single Page Application (SPA):*
Utilizarea unui *router* React pentru navigarea între componente asigură o experiență de utilizator fluidă, fără reîncărcări ale paginii.

- *Sistem de notificări Toast:*

Toate operațiile realizate (ex. adăugarea unei locații sau actualizarea unei monede) sunt însoțite de notificări de tip *toast*. Aceste notificări oferă feedback rapid utilizatorului, contribuind la îmbunătățirea interacțiunii cu aplicația.

Aplicația administrator este proiectată pentru a fi intuitivă și eficientă, oferind toate instrumentele necesare unui partener pentru a-și gestiona activitățile legate de schimbul valutar. Aceasta integrează tehnologii moderne și componente interactive, oferind o experiență profesională utilizatorilor.

4.3. Aplicația Client

Aplicația client reprezintă interfața dedicată utilizatorilor finali, oferindu-le o experiență simplă și intuitivă pentru vizualizarea cursurilor de schimb valutar. Implementată ca o *Single Page Application (SPA)* utilizând *React.js*, aceasta integrează funcționalități cheie pentru a ajuta utilizatorii să acceseze rapid informațiile de care au nevoie.

4.3.1. Structura aplicației

Aplicația este organizată într-o interfață minimalistă și eficientă, fiind ușor de utilizat chiar și pentru cei fără cunoștințe tehnice avansate.

Componenta principală a aplicației client, *Dashboard*, include butoane intuitive care permit utilizatorilor să acceseze rapid una dintre cele trei opțiuni:

- **Nearest Rates**
- **Best Rates**
- **All Rates**

4.3.2. Principalele componente React

- *Dashboard:*

Componenta principală a aplicației client.

- *NearestRates:*

Această componentă afișează cele mai apropiate rate de schimb valutar în funcție de geolocația utilizatorului.

La accesare:

- Aplicația solicită permisiunea utilizatorului pentru a obține coordonatele GPS (latitudine și longitudine).
- În cazul în care geolocația nu este activată sau permisă, un mesaj informativ este afișat pentru utilizator, explicând problema și oferind soluții.
- Datele sunt preluate din *API* pe baza coordonatelor GPS, iar rezultatele sunt afișate împreună cu un preview *Google Maps* pentru fiecare schimb valutar.

- *BestRates:*

Această componentă listează cursurile de schimb valutar ordonate în funcție de cea mai bună valoare (raport favorabil între cumpărare și vânzare).

- Similar componentei *NearestRates*, fiecare rată este însoțită de un preview *Google Maps* al locației unde aceasta este disponibilă.

- *AllRates:*

Această componentă oferă o vizualizare completă a tuturor cursurilor de schimb valutar disponibile pe platformă.

- Similar pot accesa informații detaliate despre toate locațiile și cursurile configurate în sistem, împreună cu harta aferentă fiecărui punct de schimb valutar.

4.3.3. Funcționalități suplimentare

- *Google Maps Preview:*

Toate cele trei componente includ o hartă interactivă generată cu ajutorul *Google Maps API*, oferind utilizatorilor o reprezentare vizuală a locațiilor asociate schimburilor valutare.

- *Feedback și mesaje pentru utilizatori:*

Aplicația este proiectată să gestioneze cazurile de eroare, cum ar fi lipsa accesului la geolocație, afișând notificări clare și informative pentru utilizator.

- *Accesibilitate și performanță:*

Datorită arhitecturii de tip SPA și utilizării optimizate a componentelor React, aplicația oferă timpi de încărcare rapizi și o experiență de utilizator fluidă.

Aplicația client este concepută pentru a fi prietenoasă, practică și accesibilă. Funcționalitățile acesteia răspund nevoilor utilizatorilor de a găsi rapid și eficient cele mai bune sau cele mai apropiate rate de schimb valutar, contribuind la o experiență digitală de calitate superioară.

Capitolul 5. Proiectare de detaliu si implementare

Acest capitol prezintă procesul de proiectare și implementare al platformei, detaliind arhitectura modulară și componentele esențiale ale fiecărei aplicații: Server/API, aplicația administrator și aplicația client. Se vor descrie obiectivele, funcționalitățile principale și modul de integrare a acestora, oferind o privire detaliată asupra fiecărui modul în parte, alături de un plan clar pentru implementare.

5.1. Plan de proiectare și implementare

Planul de proiectare urmărește o arhitectură modulară care să permită extinderea și întreținerea facilă a aplicației. Principalele etape ale proiectării sunt:

1. *Definirea cerințelor*: Identificarea funcționalităților esențiale pentru fiecare modul.
2. *Arhitectura aplicației*: Stabilirea unei arhitecturi bazate pe trei niveluri:
 - Backend (server/API) pentru manipularea datelor și securitatea accesului.
 - Frontend-ul aplicației administrator pentru gestionarea resurselor platformei.
 - Frontend-ul aplicației client pentru oferirea unei experiențe interactive utilizatorilor finali.
3. *Implementarea modulară*: Divizarea fiecărei aplicații în componente autonome pentru reducerea complexității.
4. *Testare și integrare*: Asigurarea unei funcționări corecte prin testări unitare și de integrare.

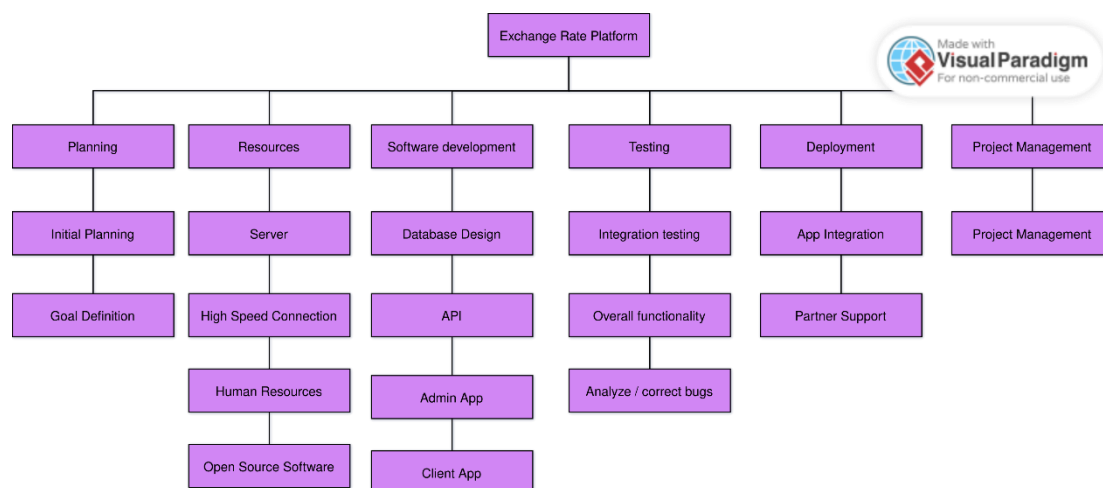


Figura 5.1: Diagrama fluxului de lucru

5.2. Aplicația Server / API

- **Descriere generală**

Serverul gestionează accesul la baza de date și implementează logica de business a platformei. Este construit folosind *Node.js* și *Express.js*, cu date stocate într-o bază de date *MySQL*.

- **Funcționalități cheie**

1. *Managementul partenerilor și locațiilor*

Partenerii pot avea mai multe locații asociate, fiecare locație fiind identificată prin coordonate GPS, adresă și informații suplimentare.

Relația partener-locație este gestionată printr-o legătură între tabelele partner și location.

2. *Configurarea și gestionarea monedelor*

Sistemul permite configurarea unei liste personalizate de monede, fiecare identificată printr-un nume unic.

3. *Adăugarea și administrarea ratelor de schimb valutar*

Ratele de schimb sunt asociate atât unei locații specifice, cât și unei monede. Utilizatorii pot adăuga, actualiza și șterge ratele, respectând constrângerile de integritate.

4. *Securitatea datelor*

Parolele utilizatorilor sunt criptate folosind algoritmul *MD5*, oferind un nivel de protecție.

Endpoint-urile API sunt securizate cu JSON Web Tokens (*JWT*), utilizând o cheie secretă configurabilă, pentru a garanta accesul exclusiv utilizatorilor autorizați.

- **Structura bazei de date**

1. *Tabela partner*

- 1.1. Coloane:

- *idPartner*: cheie primară, identifică unic fiecare partener.
- *username*: unic, folosit pentru autentificare.
- *email*: unic, pentru identificare suplimentară.
- *password*: stochează parola criptată.
- *information*: detalii adiționale despre partener.

- 1.2. Constrângeri

- Toți partenerii au nume de utilizator și email-uri unice.

2. *Tabela location*

- 2.1. Coloane:

- *idLocation*: cheie primară, identifică unic fiecare locație.
- *idPartner*: cheie străină, face legătura cu un partener.
- *latitude și longitude*: coordonatele GPS ale locației.
- *address*: adresa locației.
- *information*: detalii adiționale despre locație.

2.2. Constrângeri

- Relația dintre locații și parteneri este definită printr-o cheie străină (*FK_PartnerLocation*), cu opțiunea de ștergere în cascadă.

3. *Tabela currency*

3.1. Coloane:

- *idCurrency*: cheie primară, identifică unic fiecare monedă.
- *name*: unic, denumirea monedei (ex.: USD, EUR).

3.2. Constrângeri

- Fiecare monedă are o denumire unică.

4. *Tabela rate*

4.1. Coloane:

- *idRates*: cheie primară, identifică unic fiecare curs valutar.
- *idLocation*: cheie străină, leagă rata de o locație specifică.
- *idCurrency*: cheie străină, leagă cursul de o monedă.
- *date*: data și ora la care a fost introdus cursul.
- *value*: valoarea cursului de schimb.

4.2. Constrângeri

- Relațiile cu locațiile și monedele sunt definite prin chei străine (*FK_LocationRate* și *FK_CurrencyRate*), asigurând consistența datelor.
- În cazul ștergerii unei locații sau a unei monede, ratele asociate sunt șterse automat.

- **Documentație Swagger**

Aplicația server utilizează *Swagger (OpenAPI)* pentru a genera și gestiona documentația API-ului. Swagger oferă o interfață interactivă care permite dezvoltatorilor și utilizatorilor să vizualizeze și să testeze endpoint-urile disponibile direct dintr-un browser web.

Endpoint-urile aplicației server sunt organizate în *module de tip fișier componentă*, fiecare fișier reprezentând o colecție de rute asociate unei funcționalități specifice. Această abordare modulară facilitează organizarea codului, întreținerea și extinderea funcționalităților aplicației.

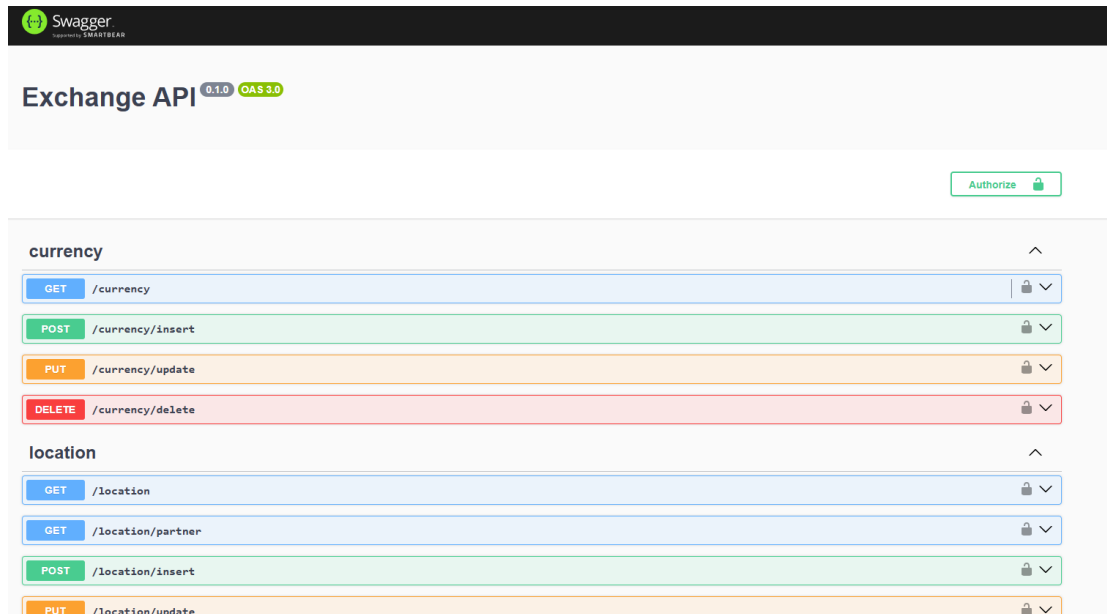


Figura 5.2: Swagger UI pentru API

5.3. Aplicația Administrator

- **Descriere generală**

Aplicația Adminstrator este un *Single Page Application (SPA)* dezvoltată folosind *React.js* și este dedicată gestionării datelor platformei. Aceasta oferă funcționalități precum gestionarea locațiilor, cursurilor valutare și a monedelor însoțite de opțiuni avansate de filtrare pentru o experiență de utilizare optimizată. Aplicația este modulară, organizată pe componente reutilizabile, și integrează notificări pentru operațiunile efectuate, utilizând o navigare bazată pe *React Router*.

- **Structura aplicației**

1. *Pagina de autentificare*

- Oferă acces securizat pentru utilizatorii de tip administrator.
- Parola utilizatorilor este stocată în baza de date folosind algoritmul *MD5*, iar autentificarea este gestionată prin *JWT* pentru securizarea sesiunilor.

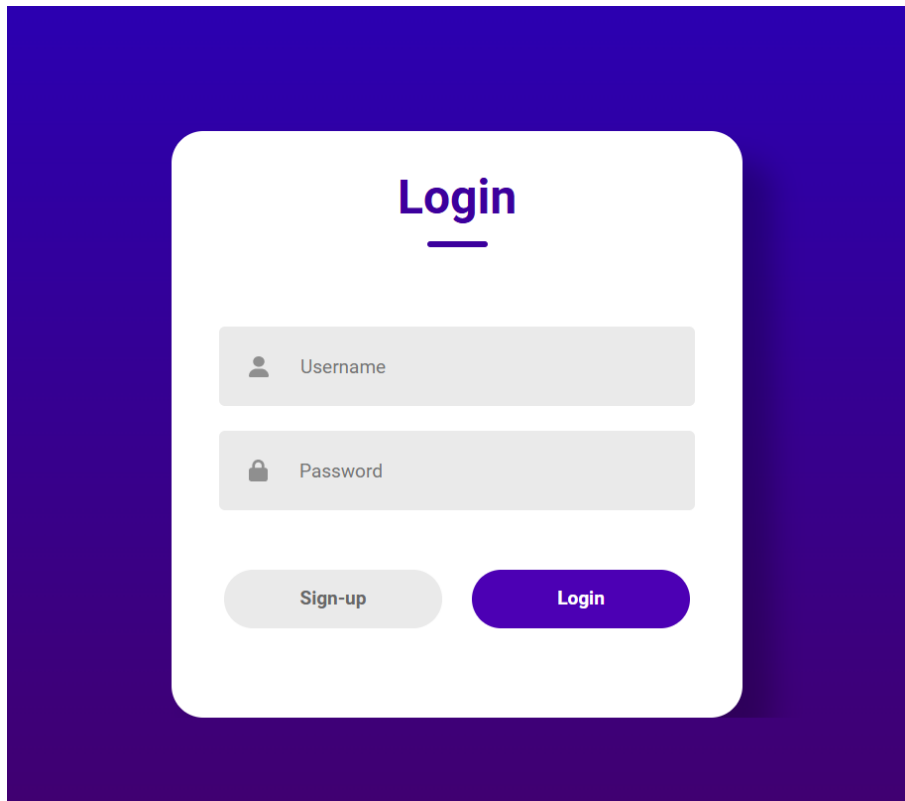


Figura 5.3: UI autentificare

2. Interfața principală

După autentificare, aplicația afișează o interfață intuitivă cu un navbar pentru navigare rapidă între module:

- **Dashboard**
- **Locations**
- **Partners**
- **Currency**
- **Rates**

- **Funcționalități cheie**

1. Dashboard

Prezintă grafice dinamice folosind React Google Charts pentru a vizualiza date statistice despre locațiile și cursurile valutare ale partenerului logat.

Datele sunt actualizate în timp real, oferind o privire de ansamblu asupra activităților din platformă.



Figura 5.4: UI dashboard

2. Locations

- Administrarea locațiilor asociate partenerului logat.
- Adăugare locații noi: Utilizatorul introduce o adresă, iar aplicația generează un preview *Google Maps*.
- Preluarea automată a coordonatelor GPS: Utilizând API-ul *Google Maps*, se obțin latitudinea și longitudinea corespunzătoare adresei introduse.
- Gestionarea locațiilor existente: ștergere, actualizare sau modificare informații adiționale.

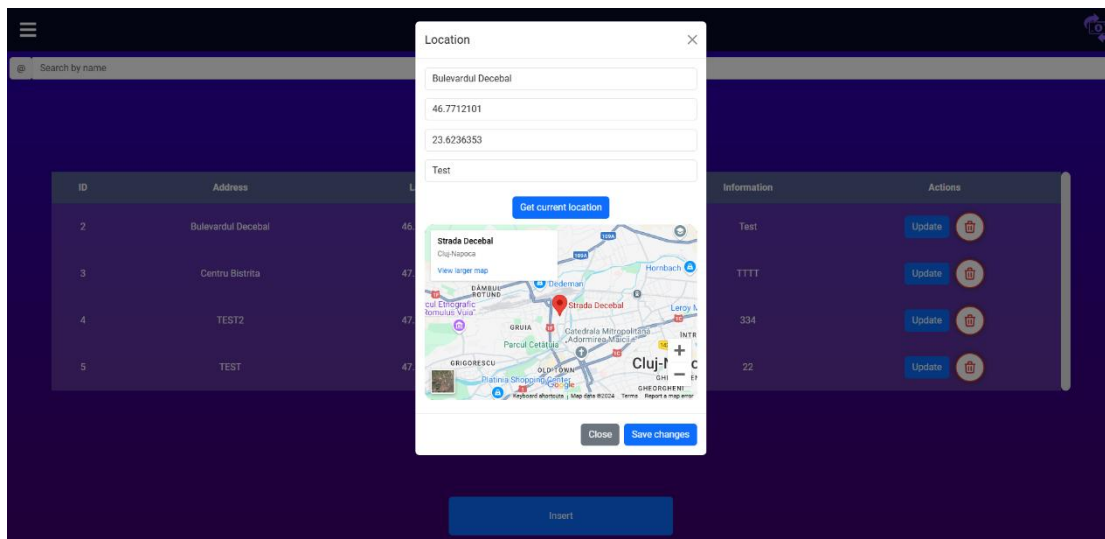
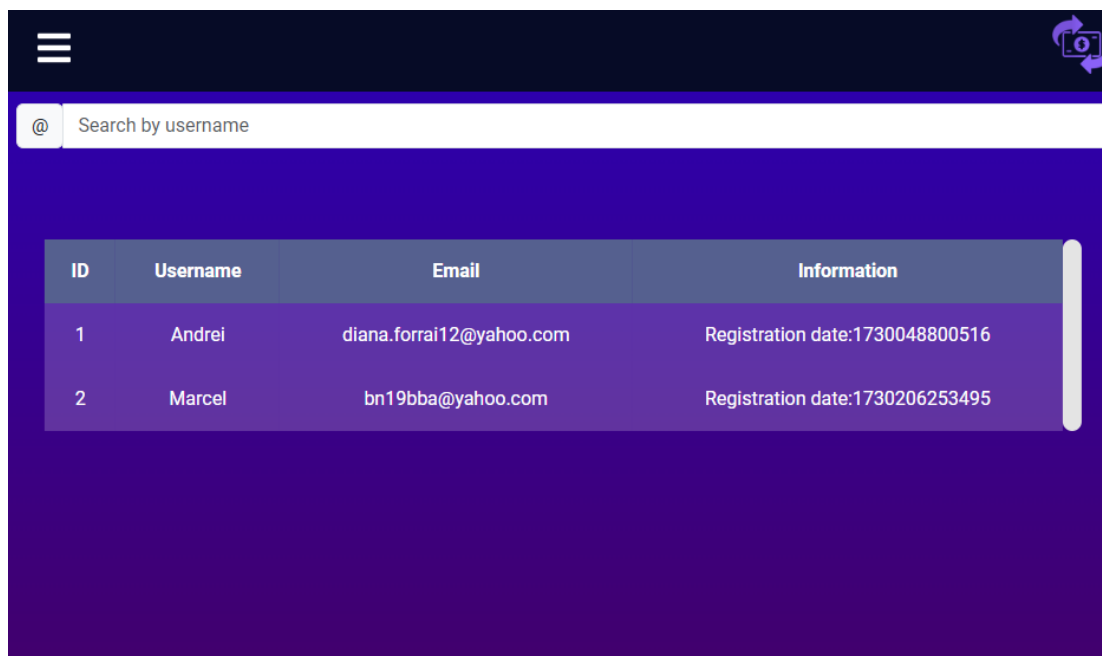


Figura 5.5: UI locations

3. Partners

Afișează o listă cu informațiile publice ale tuturor partenerilor înregistrați în platformă. Informațiile afișate cuprind numele partenerului, adresa de e-mail și detalii suplimentare pentru a ușura comunicarea și colaborarea.



The screenshot shows a web application interface for 'Partners'. At the top, there is a dark blue header with a hamburger menu icon on the left and a refresh icon on the right. Below the header is a search bar with a '@' icon and the text 'Search by username'. The main content area is a light blue background featuring a table with the following data:

ID	Username	Email	Information
1	Andrei	diana.forrai12@yahoo.com	Registration date:1730048800516
2	Marcel	bn19bba@yahoo.com	Registration date:1730206253495

Figura 5.6: UI partners

4. Currency

Gestionarea completă a monedelor:

- *Vizualizare*: Listarea tuturor monedelor existente în platformă.
- *Adăugare*: Introducerea de noi monede.
- *Modificare*: Actualizarea numelor sau altor detalii despre monede.
- *Ștergere*: Eliminarea monedelor care nu mai sunt utilizate.

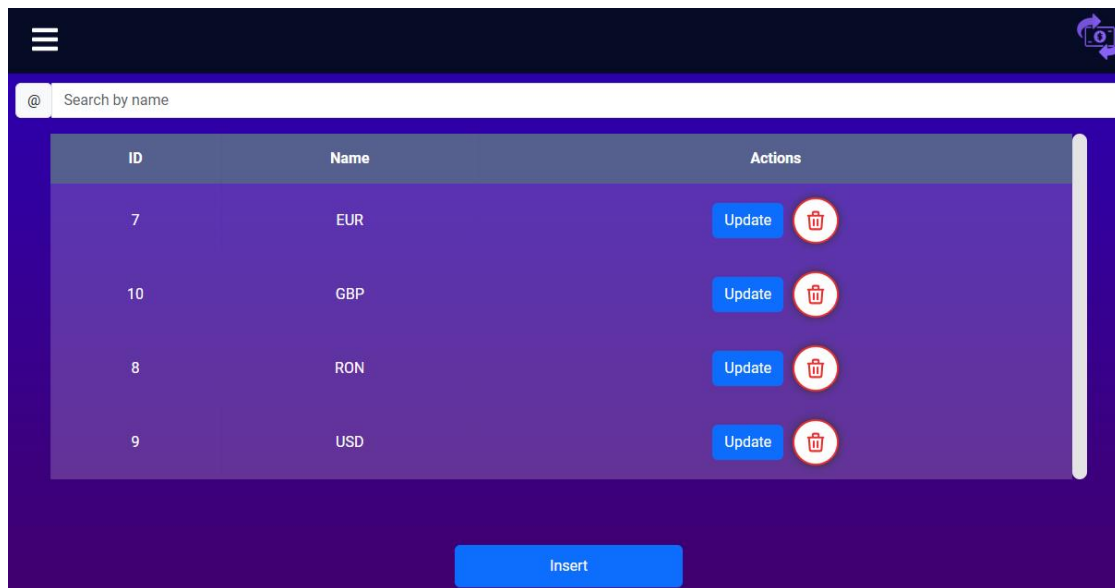


Figura 5.7: UI currency

5. Rates

Gestionarea cursurilor valutare pentru locațiile partenerului logat:

- *Adăugare*: Administratorii pot introduce noi cursuri valutare pentru locațiile proprii, asociate monedelor existente.
- *Vizualizare*: Listarea tuturor cursurilor valutare pentru locațiile configurate.
- *Modificare și ștergere*: Actualizarea sau eliminarea cursurilor existente.

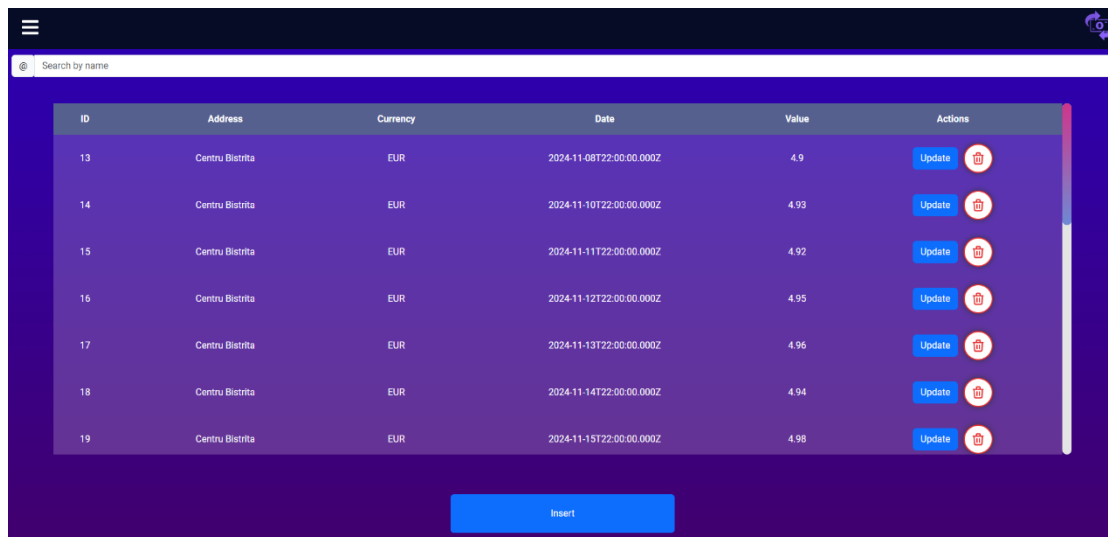


Figura 5.8: UI rates

- **Implementare modulară**

Aplicația este organizată în componente React reutilizabile, fiecare componentă fiind responsabilă pentru o funcționalitate specifică. Navigarea între modulele aplicației este realizată prin *React Router*, ceea ce permite afișarea dinamică a conținutului fără a reîncărca pagina.

Exemple de componente:

- *Dashboard.js*: Responsabilă pentru afișarea graficelor și statisticilor.
- *Location.js*: Forme și tabele pentru gestionarea locațiilor.
- *Rates.js*: Gestionarea și afișarea cursurilor valutare.

- **Securitate și UX**

1. *Autentificare și sesiuni securizate*

Utilizarea *JWT* pentru autentificare și autorizare. Token-ul este stocat în *localStorage* și utilizat pentru verificarea accesului la API-uri.

2. *Notificări*

Aplicația utilizează un context global pentru gestionarea notificărilor (de tip *toast*) pentru a afișa mesaje de succes sau eroare în urma operațiunilor efectuate.

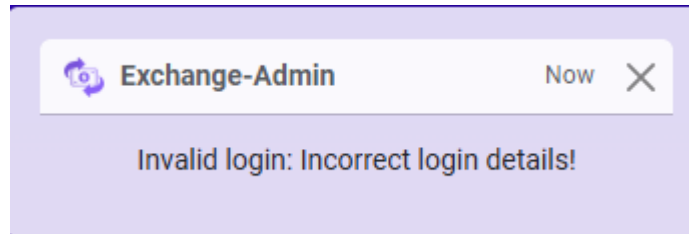


Figura 5.9: Toast notification

3. *Design responsive*

Stilizarea aplicației este realizată cu *Bootstrap*, oferind compatibilitate cu diverse dimensiuni de ecran și dispozitive.

5.4. Aplicația Client

- **Descriere generală**

Aplicația Client este o *Single Page Application (SPA)* realizată cu *React.js*, destinată utilizatorilor finali care doresc să vizualizeze cursurile valutare și locațiile disponibile ale partenerilor din platformă. Aplicația oferă o interfață simplă și intuitivă, care permite utilizatorilor să găsească rapid ratele de schimb și să vizualizeze locațiile pe hartă. În plus, aplicația oferă opțiuni de filtrare avansate, bazate pe geolocație sau pe cele mai bune rate de schimb, pentru a facilita experiența utilizatorului.

- **Structura aplicației**

După selecția unui buton, utilizatorii vor fi redirecționați către componentele corespunzătoare care afișează informațiile dorite. *React Router* este utilizat pentru a naviga între componentele aplicației. Modulele aplicației:

- **Nearest Rates**
- **Best Rates**
- **All Rates**

Grupare pe locație: Cursurile sunt grupate după locație, astfel încât utilizatorul poate accesa informațiile necesare într-un format ușor de navigat.

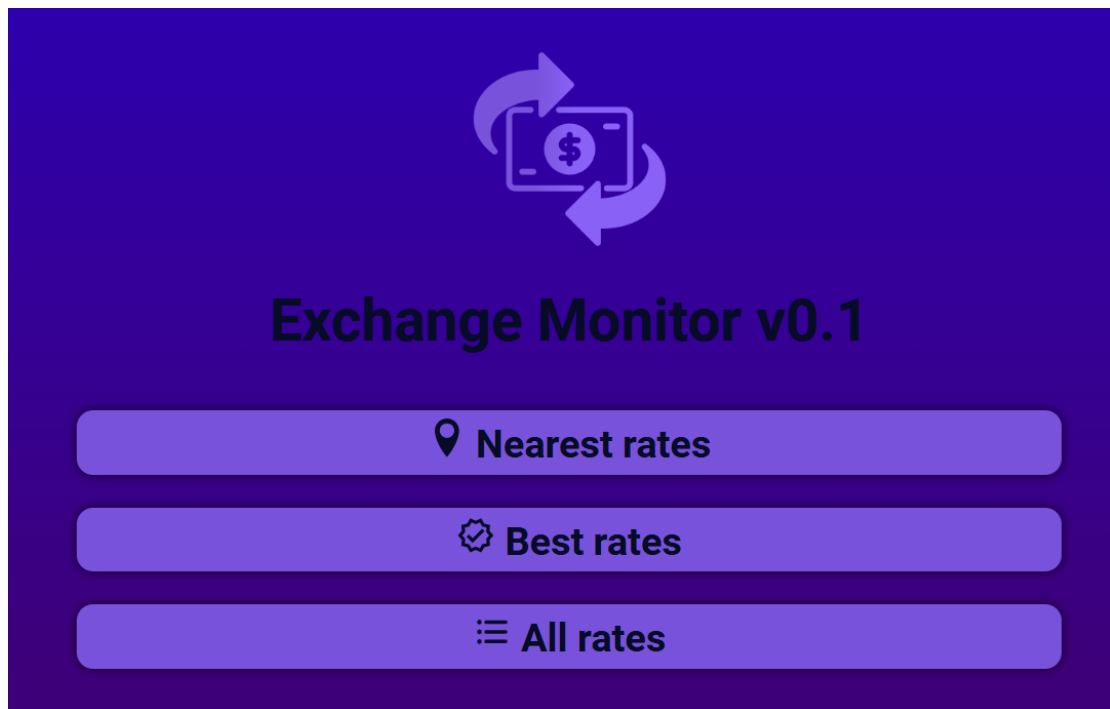


Figura 5.10: UI client

- **Funcționalități cheie**

1. *NearestRates*

Componenta *NearestRates* este dedicată utilizatorilor care doresc să vadă cursurile valutare ale celor mai apropiate locații. Folosind *geolocația*, aplicația va solicita permisiunea utilizatorului pentru a obține coordonatele GPS și a calcula distanțele față de locațiile de schimb valutar.

```
{
  "result": [
    {
      "idRates": 1,
      "idLocation": 5,
      "address": "Strada Exemplu, Nr. 10, București",
      "idCurrency": 2,
      "name": "EUR",
      "date": "2024-12-15T14:30:00Z",
      "value": 4.8795,
      "distance": 1.34
    },
    {
      "idRates": 2,
      "idLocation": 7,
      "address": "Strada Test, Nr. 20, Cluj-Napoca",
      "idCurrency": 2,
      "name": "EUR",
      "date": "2024-12-15T14:15:00Z",
      "value": 4.8800,
      "distance": 5.67
    }
  ],
  "page": 1,
  "success": true
}
```

Figura 5.11: Raspunsul serverului

- *Harta și coordonate:* În această componentă, fiecare locație va fi asociată cu coordonatele GPS (latitudine și longitudine) și va fi vizualizată pe *Google Maps* (acest lucru este recurent și în celelalte componente, deci nu se va relua).

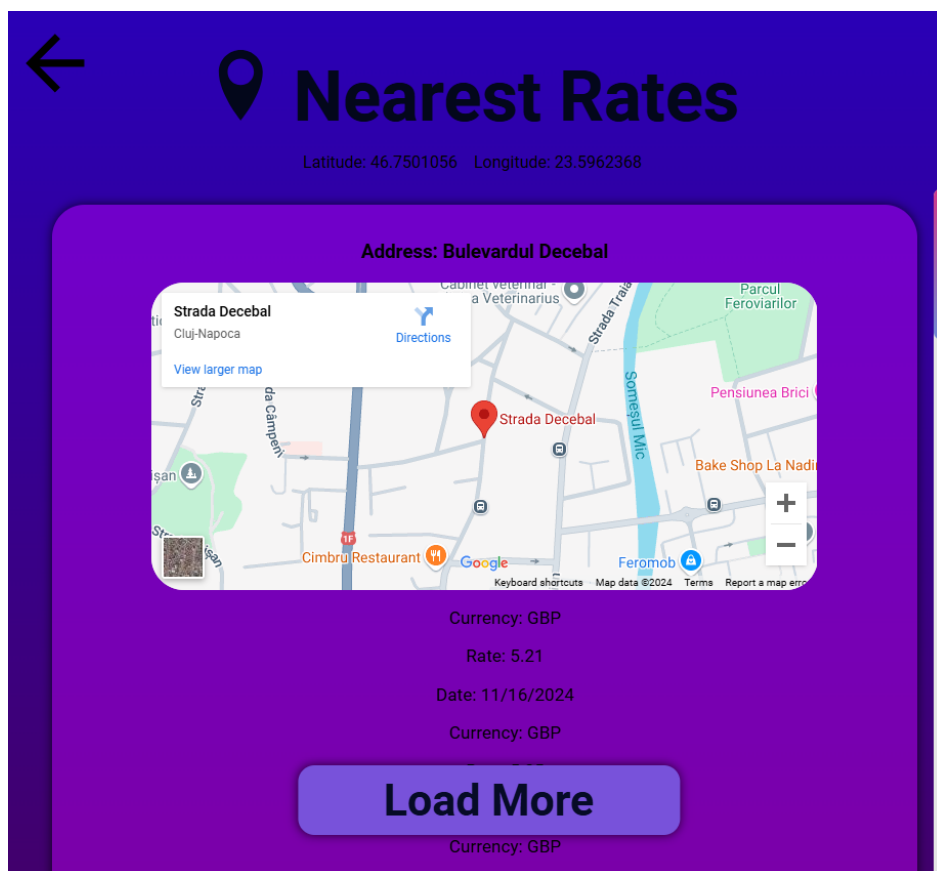


Figura 5.12: UI nearest rates

2. *BestRates*

Această componentă afișează cele mai bune rate de schimb pentru fiecare locație disponibilă în platformă. Cursurile valutare sunt ordonate în funcție de valoarea lor, iar utilizatorul poate vizualiza locațiile care oferă cele mai bune rate.

Ratele de schimb sunt filtrate și afișate în funcție de cea mai bună valoare disponibilă, oferind utilizatorului o selecție rapidă și eficientă a celor mai avantajoase opțiuni. Componenta *BestRates* utilizează sistemul de paginare configurat la nivelul API-ului.

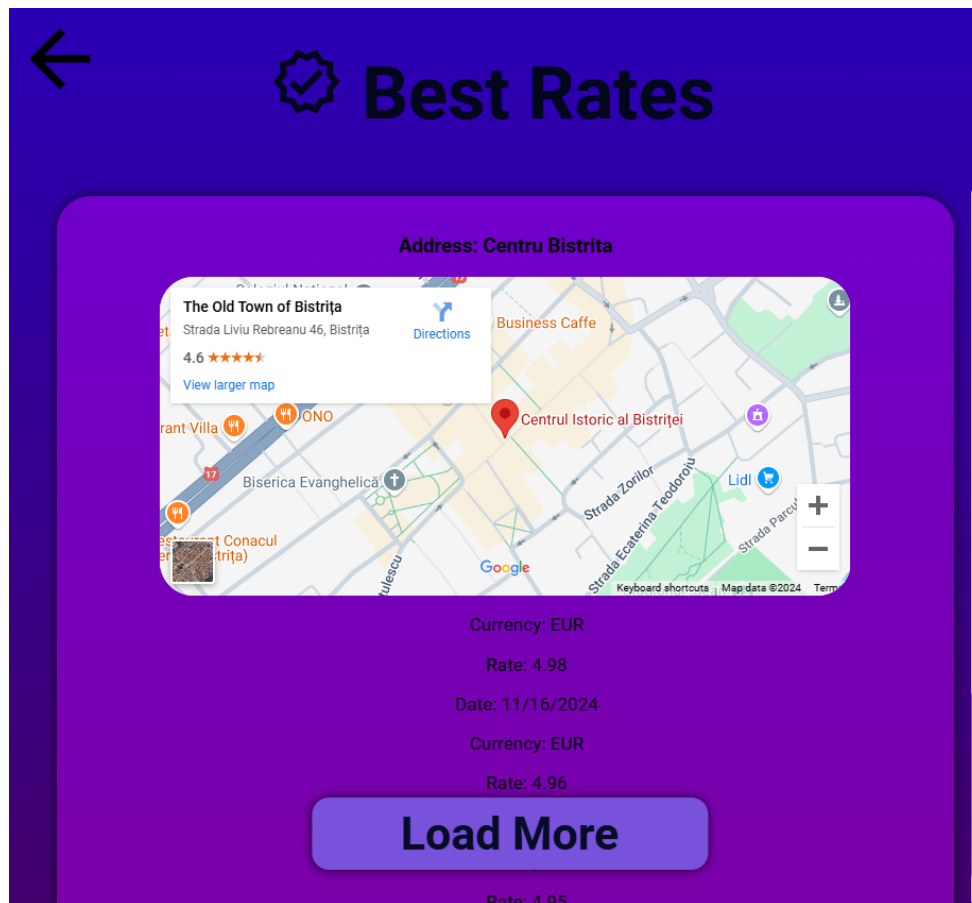


Figura 5.13: UI best rates

3. AllRates

Această componentă listează *toate ratele de schimb disponibile* pentru toate locațiile înregistrate în sistem. Cursurile sunt afișate fără a fi filtrate în funcție de criterii precum distanța sau valoarea, oferind o imagine de ansamblu completă asupra disponibilității și valorii cursurilor valutare în platformă, dar sunt filtrate după data înregistrării.

Acest design modular și structura unitară a interfeței garantează o experiență fluidă pentru utilizatori, fie că sunt interesați de cele mai bune rate de schimb, locațiile cele mai apropiate sau o viziune generală asupra tuturor ratelor disponibile.

Capitolul 6. Manual de instalare și utilizare

Platforma trebuie instalată pe orice stație care poate funcționa ca server, unde vor fi compilate și executate aplicațiile individuale pentru server, client și administrator. Serverul trebuie să fie configurat pentru a fi accesibil din exterior, astfel încât orice utilizator să poată accesa oricare dintre cele două aplicații frontend.

Cerințe preliminare: *Node.js* și o bază de date *MySQL* configurată.

Pentru instalarea *Node.js*, se va naviga la URL-ul următor de unde se poate descărca software-ul: <https://nodejs.org/en/download/package-manager/current>. *MySQL* se poate descărca și instala în funcție de device de la următorul URL: <https://www.mysql.com/downloads/>.

Proiectul se va deschide cu un interpretor de cod (exemplu Visual Studio Code), și se va naviga la path-ul folder-ului care conține aplicația server cu comanda: `cd locația-locală/exchange/server`, unde *locația-locală* este path-ul unde a fost stocată aplicația. Se va deschide fișierul `app.js`, iar datele de conectare la serviciul *MySQL* se vor modifica cu cele configurate de fiecare utilizator la instalare. La rularea aplicației server, vor fi create automat toate tabelele și relațiile dintre ele, cu condiția ca datele de conectare furnizate să fie corecte. Pentru a lansa aplicația server, se va deschide un terminal, iar apoi se va rula comanda `npm start`.

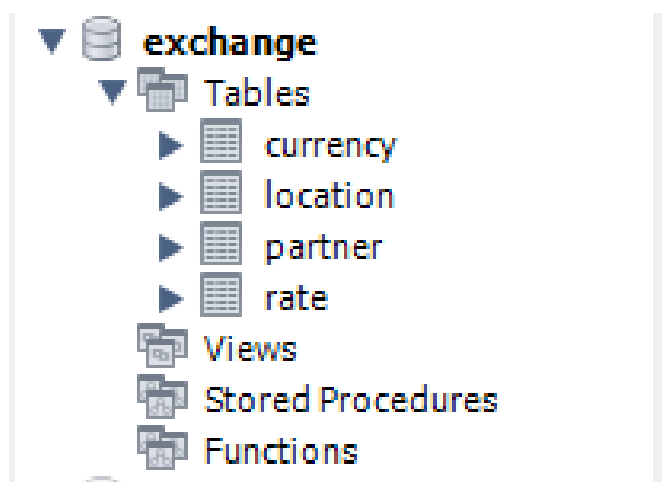


Figura 6.1: Structura bazei de date

După lansarea cu succes a aplicației server, documentația API-ului poate fi accesată la adresa *localhost:port/api*, unde portul este indicat în Visual Studio Code în momentul rulării aplicației, de obicei fiind setat la 3000.

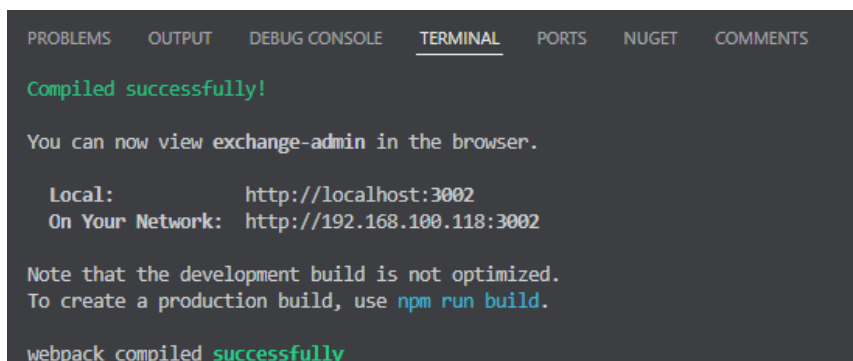
Următorul pas constă în rularea aplicațiilor de front-end, proces similar cu cel al aplicației server. Se va deschide un terminal separat pentru fiecare aplicație, navigând la calea corespunzătoare a fiecărei aplicații: *locație-locală/exchange/exchange-client* pentru aplicația client și *locație-locală/exchange/exchange-admin* pentru aplicația de administrare, unde *locația-locală* este path-ul unde a fost stocată aplicația. Aplicațiile vor fi lansate prin executarea comenzii *npm start* în terminalele respective.

```
PS E:\exchange> cd exchange-client
PS E:\exchange\exchange-client> npm start

> exchange-client@0.1.0 start
> react-scripts start
```

Figura 6.2: Exemplu rulare

După rularea cu succes a acestora, aplicațiile vor fi accesibile la adresa *localhost:port*, unde portul este indicat de Visual Studio Code pentru fiecare aplicație în parte.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  NUGET  COMMENTS

Compiled successfully!

You can now view exchange-admin in the browser.

Local:      http://localhost:3002
On Your Network: http://192.168.100.118:3002

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Figura 6.3: Mesajul obținut la compilare cu succes

Notă: Comanda *npm start* lansează aplicația într-o variantă de debug, care poate avea performanțe inferioare. Pentru a rula sau a crea un build de producție, se va utiliza comanda *npm run build* în locul acesteia.

Capitolul 7. Concluzii

Platforma reprezintă o soluție completă pentru gestionarea și vizualizarea cursurilor valutare, destinată atât utilizatorilor finali, cât și administratorilor care trebuie să actualizeze și să gestioneze datele legate de cursurile de schimb și locațiile asociate acestora. Implementarea acestei platforme se bazează pe un set de tehnologii moderne și arhitecturi robuste, care asigură un sistem scalabil și sigur pentru procesarea și afisarea datelor în timp real.

1. Implementarea funcționalităților

Un aspect esențial al acestei platforme este *configurabilitatea* și *dinamismul*. Utilizatorii și administratorii pot configura locațiile și monedele, iar sistemul permite asocierea dinamică a cursurilor de schimb cu diverse locații geografice. De asemenea, administratorii pot gestiona ușor cursurile de schimb, având posibilitatea de a vizualiza și actualiza datele direct din aplicația admin. Această abordare asigură flexibilitate și permite adaptarea rapidă la schimbările pieței valutare.

În ceea ce privește *rapoartele și vizualizarea datelor*, platforma include grafice interactive și hărți pentru a oferi o înțelegere mai clară a evoluției cursurilor de schimb. Aceste vizualizări sunt realizate cu ajutorul *React Google Charts* și *Google Maps*, oferind utilizatorilor și administratorilor un mod eficient de a analiza și compara datele financiare.

2. Securitate și performanță

Un alt punct forte al platformei este atenția acordată securității. API-urile sunt protejate prin *JWT*, iar parolele utilizatorilor sunt criptate folosind *MD5*, ceea ce garantează un nivel ridicat de protecție a datelor sensibile. De asemenea, performanța este optimizată atât pe partea de server, cât și pe partea de frontend. Utilizarea *Node.js* pe server asigură un timp de răspuns rapid pentru procesarea cererilor, iar aplicațiile frontend sunt construite cu *React.js*, un framework care permite randarea rapidă și interactivă a paginilor.

3. Instalare și utilizare

Platforma este ușor de instalat și de utilizat. După configurarea corectă a serverului și a bazei de date *MySQL*, aplicațiile server, admin și client pot fi lansate local, fiecare pe porturi separate, iar interacțiunile între ele sunt gestionate eficient. Utilizatorii pot accesa aplicațiile frontend de pe orice dispozitiv conectat la rețea, iar administratorii pot modifica rapid datele din backend. Pentru rularea în producție, este disponibilă comanda *npm run build*, care creează o versiune optimizată a aplicațiilor.

4. Provocări și oportunități de îmbunătățire

În ciuda robusteții soluției, implementarea acestei platforme poate întâmpina provocări legate de scalabilitatea bazei de date și de gestionarea unui număr mare de utilizatori simultani. În plus, integrarea cu surse externe de date pentru actualizarea cursurilor de schimb poate reprezenta o oportunitate de îmbunătățire. De asemenea, ar putea fi introdusă o funcționalitate de alertă pentru utilizatori, care să îi notifice atunci când o rată de schimb atinge un prag specificat.

În concluzie, platforma prezentată reprezintă o soluție completă și scalabilă pentru gestionarea și vizualizarea cursurilor de schimb, fiind construită pe tehnologii moderne care garantează performanță, securitate și o experiență de utilizator interactivă. Platforma are un mare potențial de extindere și poate fi îmbunătățită continuu, adaptându-se la cerințele pieței financiare în schimbare.

Bibliografie

- [1] **Goodhart, C., & Tsang, J. (2020).** *Financial Applications in the Digital Economy: Opportunities and Challenges*. Journal of Financial Studies, 45(3), 123-139.
- [2] **Zhou, H., Lin, Q., & Wang, Y. (2019).** *API-Driven Financial Systems: A Study on Data Integration and Security Challenges*. Proceedings of the International Conference on Data and Application Security, 12(5), 78-90.
- [3] **Kumar, S., & Sharma, R. (2021).** *Modern Web Technologies for Financial Applications: A Review of React.js and Its Use Cases*. International Journal of Computer Applications, 63(2), 45-56.
- [4] **Lee, J., & Park, H. (2022).** *Location-Based Financial Applications: Enhancing User Experience through Geolocation Features*. International Journal of Technology and Innovation, 29(1), 34-49.
- [5] **MySQL Documentation.** (Accessed 2023). *Relational Databases in Large-Scale Applications*. Oracle Corporation.

Anexe

- Secvențe de cod relevante:

```
router.get('/nearest', function (req, res, next) {
  const { latitude, longitude, page = 1 } = req.query;
  const limit = 5;
  const offset = (page - 1) * limit;

  if (!latitude || !longitude) {
    res.status(400).json({ error: 'The request has missing information!', success: false });
    return;
  }

  const query = `
    SELECT rate.idRates, rate.idLocation, location.address, rate.idCurrency, currency.name, rate.date, rate.value,
    (6371 * acos(cos(radians(${latitude}))) * cos(radians(location.latitude)) * cos(radians(location.longitude) - radians(${longitude})) + sin(radians(${latitude}))
    FROM rate
    INNER JOIN location ON rate.idLocation = location.idLocation
    INNER JOIN currency ON rate.idCurrency = currency.idCurrency
    ORDER BY distance, rate.date DESC
    LIMIT ${limit} OFFSET ${offset}`;

  req.db.query(query, (err, result) => {
    if (err) {
      res.status(500).json({ error: err.message, success: false });
      return;
    }
    res.json({ result, page: page, success: true });
  });
});
```

Anexa 1: Endpoint API pentru client, pentru cursuri în funcție de distanță

```
router.post('/login', function (req, res, next) {
  const { username, password } = req.body;
  const query = 'SELECT * FROM partner WHERE username = ?';

  // Hash the password using MD5
  const hashedPassword = crypto.createHash('md5').update(password).digest('hex');

  req.db.query(query, [username], (err, results) => {
    if (err) {
      res.status(500).json({ success: false, error: err.message });
      return;
    }
    if (results.length > 0) {
      const partner = results[0];
      if (partner.password === hashedPassword) {
        const token = jwt.sign({ id: partner.idPartner, username: partner.username }, secretKey, { expiresIn: '24h' });
        res.json({ success: true, token: token });
      } else {
        res.status(401).json({ success: false, error: 'Incorrect login details!' });
      }
    } else {
      res.status(401).json({ success: false, error: 'Incorrect login details!' });
    }
  });
});
```

Anexa 2: Endpoint securizat din API, pentru logarea administratorilor


```
const ToastContext = createContext();

export const ToastProvider = ({ children }) => {
  const [showToast, setShowToast] = useState(false);
  const [toastMessage, setToastMessage] = useState('');

  const showToastMessage = (message) => {
    setToastMessage(message);
    setShowToast(true);
  };

  return (
    <ToastContext.Provider value={{ showToastMessage }}>
      {children}
      <Toast position="top-end" className="p-3 toast-message" style={{ position: 'absolute', top: 5, right: 5 }} onClose={() => setShowToast(false)}
        show={showToast} delay={3000} autohide>
        <Toast.Header>
          <img src={logo} className="rounded me-2" alt="" height="20px" width="20px" />
          <strong className="me-auto">Exchange-Admin</strong>
          <small>Now</small>
        </Toast.Header>
        <Toast.Body>{toastMessage}</Toast.Body>
      </Toast>
    </ToastContext.Provider>
  );
};

export const useToast = () => useContext(ToastContext);
```

Anexa 3: Toast Context

```
function App() {
  return (
    <div className="App">
      <ToastProvider>
        <Router>
          <AppContent />
        </Router>
      </ToastProvider>
    </div>
  );
}

function AppContent() {
  const location = window.location;

  return (
    <>
      {location.pathname !== '/login' && (
        <>
          <Navbar />
          <img className="logo" src={logo} alt="" />
        </>
      )}
      <Routes>
        <Route path="/login" element={<LoginPage />} />
        <Route path="/" element={<PrivateRoute component={Dashboard} />} />
        <Route path="/dashboard" element={<PrivateRoute component={Dashboard} />} />
        <Route path="/partners" element={<PrivateRoute component={Partners} />} />
        <Route path="/locations" element={<PrivateRoute component={Locations} />} />
        <Route path="/currency" element={<PrivateRoute component={Currency} />} />
        <Route path="/rates" element={<PrivateRoute component={Rates} />} />
      </Routes>
    </>
  );
}
```

Anexa 4: Navigare cu React Router pentru aplicația administrator

```

<div id="currency-table" className="table-container">
  {isLoading ? (<h1>Loading currencies...</h1>) : (<table {...getTableProps()}>
    <thead>
      {headerGroups.map((headerGroup) => (
        <tr {...headerGroup.getHeaderGroupProps()}>
          {headerGroup.headers.map((column) => (
            <th {...column.getHeaderProps()}>
              {column.render("Header")}
            </th>
          ))}
        </tr>
      ))}
    </thead>
    <tbody {...getTableBodyProps()}>
      {rows.map((row) => {
        prepareRow(row);
        return (
          <tr {...row.getRowProps()}>
            {row.cells.map((cell) => (
              <td {...cell.getCellProps()}> {cell.render("Cell")} </td>
            ))}
          </tr>
        );
      })}
    </tbody>
  </table>
)}
</div>

```

Anexa 5: Implementarea React Table

```

const generateMapUrlFromAddress = (address) => {
  if (address) {
    return `https://www.google.com/maps/embed/v1/q=${encodeURIComponent(address)}&zoom=16&maptype=roadmap`;
  }
  return '';
};

```

Anexa 6: Generare URL pentru Google Maps

```

<div className='data-container'>
  {Object.keys(groupedRates).length > 0 ? (
    <ul>
      {Object.entries(groupedRates).map(([address, rates], index) => (
        <div className='container-rate' key={index}>
          <h3>Address: {address}</h3>
          <iframe
            width="80%"
            height="300px"
            loading="lazy"
            allowFullScreen
            src={generateMapUrlFromAddress(address)}
          ></iframe>
          <ul>
            {rates.map((rate, idx) => (
              <li key={idx}>
                <p>Currency: {rate.name}</p>
                <p>Rate: {rate.value}</p>
                <p>Date: {new Date(rate.date).toLocaleDateString()}</p>
              </li>
            ))}
          </ul>
        </div>
      ))}
    </ul>
  ) : (
    <p>No rates found.</p>
  )}
</div>
<button className='load-more' onClick={loadNewData}>Load More</button>

```

Anexa 7: Cod UI Client