WP4-A5. Producția unui RockChain Tool interactiv.

"Finanțat de Uniunea Europeană. Punctele de vedere și opiniile exprimate aparțin, însă, exclusiv autorului (autorilor) și nu reflectă neapărat punctele de vedere și opiniile Uniunii Europene sau ale Agenției Executive Europene pentru Educație și Cultură (EACEA). Nici Uniunea Europeană și nici EACEA nu pot fi considerate răspunzătoare pentru acestea."

Conținut

1. INTRODUCERE

Acest document prezintă rezultatele activității WP4. A5 – Producția instrumentului interactiv RockChain. În timp ce sarcinile anterioare din WP4 se concentrau pe stratul de date (WP4. A1), rafinarea instrumentului de e-learning (WP4. A2) și specificațiile funcționale (WP4. A3), WP4. A5 descrie modul în care versiunea interactivă finală a RockChain a fost produsă, ambalată și pregătită pentru distribuție și utilizare în contexte reale de antrenament.

Obiectivul WP4. A5 este dublu. Pe de o parte, oferă o perspectivă consolidată asupra arhitecturii tehnice a instrumentului RockChain, arătând cum clientul mobil, serviciile backend și orchestrarea în timp real colaborează pentru a susține sesiuni educaționale multiplayer. Pe de altă parte, documentează fluxul de lucru de producție și implementare care conduce la versiuni gata de utilizare pentru Android și iOS, o implementare stabilă în backend și proceduri de bază pentru găzduire, acces și mentenanță.

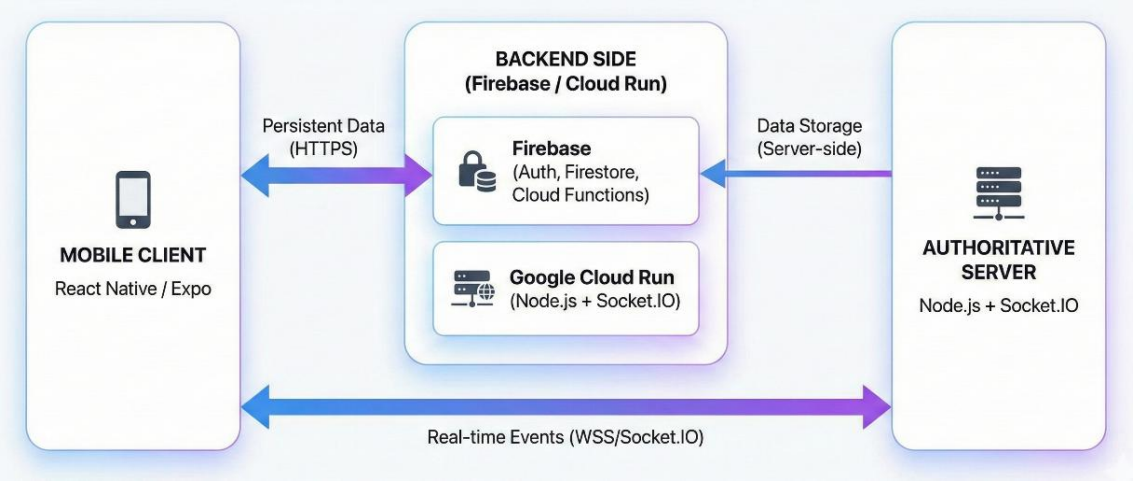
Rezultatul este o descriere "cum a fost construită" care poate fi folosită de personalul tehnic care trebuie să întrețină sau extinde instrumentul, precum și de partenerii de proiect care au nevoie de o privire de ansamblu clară asupra modului în care instrumentul interactiv RockChain a fost transformat într-o resursă gata de producție pentru VET și educația adulților.

2. ARHITECTURA SISTEMULUI INSTRUMENTULUI INTERACTIV ROCKCHAIN

Din punct de vedere tehnic, instrumentul interactiv RockChain este construit ca un sistem cu trei straturi:

- Un client mobil dezvoltat împreună cu Expo și React Native.
- Un backend Firebase care oferă autentificare, stocare persistentă și funcții serverless.
- Un server autoritar Node.js + Socket.IO care coordonează rundele în timp real și asigură că toți jucătorii văd o stare de joc consecventă.

Aceste componente sunt completate de un strat de internaționalizare, utilități de configurare a mediului și mecanisme de observabilitate de bază.



Figură 1: Arhitectură la nivel înalt.

2.1. Client mobil: framework-uri, punct de intrare și navigație

Clientul mobil RockChain este construit ca o aplicație React care rulează pe React Native, ambalată și orchestrată de Expo. În practică, asta înseamnă:

- Toate ecranele și elementele UI sunt scrise ca componente de funcție React folosind JSX.
- React 19 oferă modelul componentelor, hook-urile (*useState, useEffect, useContext*, hook-uri personalizate) și API-ul Context pe care aplicația îl folosește pentru a partaja starea jocului pe mai multe ecrane.
- React Native 0.79 redă aceste componente ca vizualizări native pe Android și iOS, astfel încât aplicația se simte și se comportă ca o aplicație nativă, împărtășind același cod JavaScript.

- Expo acționează ca wrapper și lanț de unelte: oferă serverul de dezvoltare, pachetul, sistemul de compilare EAS și acces la servicii native (rețea, stocare, informații despre dispozitiv) fără a menține proiecte separate Xcode/Android Studio.

La rulare, totul începe în *App.js*, care este punctul de intrare al clientului. Acest fișier conectează serviciile de tăiere transversală de bază de care fiecare ecran are nevoie:

- Injectează internaționalizarea prin înfășurarea întregului arbore de componente în
  I18nextProvider i18n={i18n}. Acest lucru permite oricărui ecran să folosească chei de traducere în loc de șiruri codificate fix și să aleagă automat limba utilizatorului.
- Acesta înfășoară interfața într-un *NavigationContainer* (din React Navigation), care definește un singur arbore de navigare pentru întreaga aplicație (stack-uri, tab-uri și fluxuri imbricate).
- Acesta include navigarea în interiorul a doi furnizori globali, *GameProviderHybridRobust* și *SimpleMiningProvider,* și afișează un *NetworkStatusBanner* la nivelul superior:
  - *GameProviderHybridRobust* este un strat React Context + hook care expune starea legată de joc (jocul curent și runda, jucători, inventare, cronometre, clasamente, gărzi de navigație).
  - *SimpleMiningProvider* oferă un mini-context dedicat pentru problemele de minerit, facilitând promovarea și rezolvarea provocărilor proof-of-work fără a le conecta strâns cu restul interfeței.
  - *NetworkStatusBanner* ascultă modificările de conectivitate și afișează avertismente atunci când dispozitivul este offline sau are conectivitate slabă.

Pentru că acești furnizori se află deasupra containerului de navigație, orice ecran, indiferent cât de adânc este în stivă, poate accesa starea jocului, starea de minerit și starea rețelei prin hook-uri, în loc să reimplementeze acea logică local.

Pe această bază tehnică, aplicația urmează un flux simplu, liniar, de navigare care reflectă ciclul de viață al unui joc. React Navigation este folosit cu un stack principal care îi ghidează pe utilizatori prin următoarele etape:

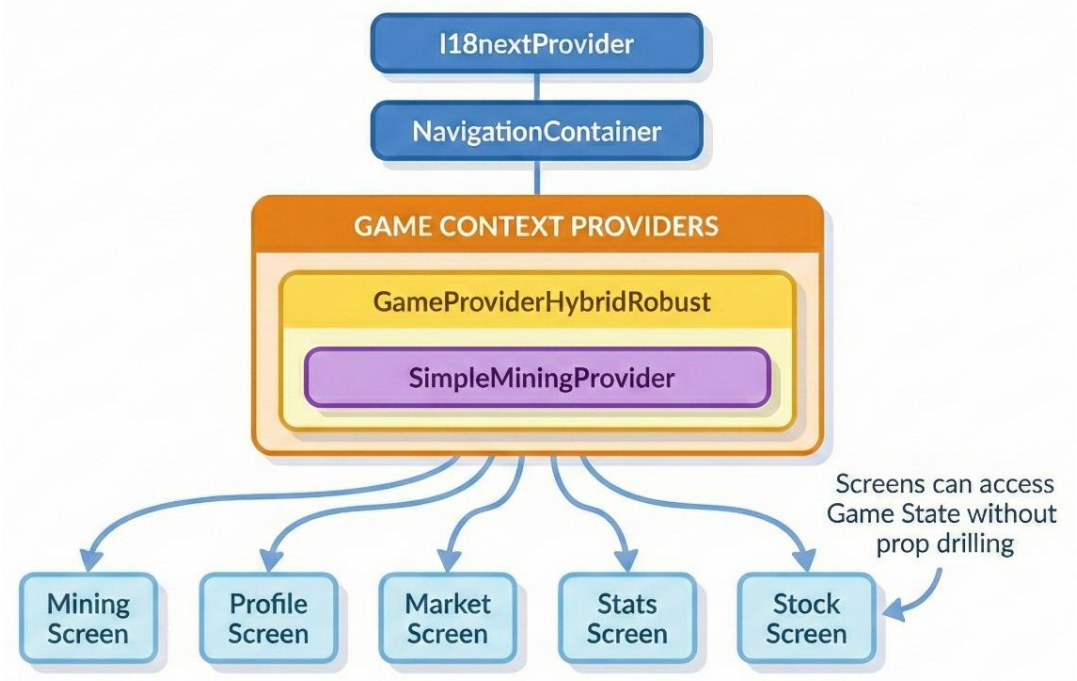
- *Login*: unde un jucător se înregistrează sau se autentifică folosind Firebase Auth.
- *Game:* un ecran wrapper care stabilește contextul pentru gameId-ul și userId-ul selectat .
- *WaitingRoom:* unde jucătorii se adună și văd cine s-a alăturat jocului înainte de începerea rundelor.
- *GameTabs:* mediul principal din joc folosit în timpul rundelor.

- *Ecrane de final de rundă și rezultate*: unde sunt afișate rezumate, clasamente și rezultate finale.

Când utilizatorul intră *în GameTabs*, structura trece la o structură bazată pe tab-uri. Bara de file de jos face parte din arborele de navigare (un navigator standard pentru filele React Navigation), dar poate fi ascunsă vizual astfel încât experiența să se simtă mai degrabă ca un joc coerent decât ca o aplicație obișnuită cu tab-uri vizibile. Intern, ascultătorii atașați evenimentelor Socket.IO și *GameContextHybridRobust* decid când să comute între module — de exemplu, deschiderea automată a Mining când apare o problemă nouă sau navigarea către rezumatul de final de rundă când cronometrul ajunge la zero — astfel încât cursanții să nu fie nevoiți să gestioneze singuri tranzițiile complexe.

În timpul unei runde active, șase module principale sunt disponibile în *GameTabs*:

- *Piață*: modulul central unde jucătorii cumpără și vând blocuri de piatră, deșeuri și produse reciclate. Se abonează la evenimente de actualizare a produselor și prețurilor și afișează liste dinamice în funcție de starea actuală a pieței.
- *Minerit*: o perspectivă concentrată care apare atunci când este declanșată o provocare proof-of-work. Afișează probleme aritmetice cronometrate, colectează răspunsuri local și le trimite serverului autoritar pentru validare.
- *Reciclare*: modulul în care părți din inventarul jucătorului pot fi convertite în RockCoins sau resurse noi, operaționalizând logica economiei circulare în acțiuni și tranzacții concrete.
- *Statistici*: un tablou de bord implementat cu componente React Native plus react-native-chart-kit, care arată indicatori de performanță și clasamente pe rundă și pe întregul joc.
- *Profil*: o zonă personală care arată avatarul jucătorului, progresul curent și setările limbii, și poate expune alte preferințe de bază în extensiile viitoare.
- *Acțiuni*: rezumă în vizualizare cantitățile și prețurile produselor cheie, acționând ca o scurtă "imagine de piață". Jucătorii îl pot deschide pentru a înțelege pe scurt care resurse sunt abundente sau rare înainte de a decide dacă să cumpere, să vândă sau să recicleze.



Figură 2: Arbore de componente și furnizori.

Toate aceste ecrane sunt componente standard React care își primesc datele în principal de la *GameProviderHybridRobust* și fluxul de evenimente Socket.IO, mai degrabă decât de la statul local ad-hoc. Această combinație, componentele și hook-urile React, randarea React Native, build/runtime-ul Expo, React Navigation pentru controlul fluxului și furnizori de context pentru starea partajată, transformă clientul mobil într-o interfață coerentă și declarativă, relativ ușor de discutat și extins, dar totodată se simte ca un joc multiplayer nativ pe dispozitivele cursanților.

2.2. Managementul stării pe partea clientului

Pe client, RockChain se bazează puternic pe API-ul React Context și pe hook-uri personalizate pentru a gestiona starea partajată. În loc să permită fiecărui ecran să păstreze propria copie a jucătorilor, cronometrelor sau a datelor de extragere, aplicația centralizează cea mai mare parte a logicii în câțiva furnizori bine definiți. Ecranele devin apoi în mare parte "vizualizări": ele aderă la acea stare și o redă, dar nu decid ele însele regulile. Acesta este motivul pentru care aplicația rămâne previzibilă chiar și atunci când multe evenimente apar în timp real.

În centrul acestui strat se află *GameContextHybridRobust*. Din punct de vedere tehnic, este un React Context susținut de o componentă provider și un set de hook-uri care

expun starea curentă a jocului pe orice ecran. Din punct de vedere conceptual, este principala sursă de adevăr pentru client. Se menține împreună:

- Lista jucătorilor din jocul curent și inventarele lor (produse, deșeuri, RockCoins).
- Cronometrele și identificatorii rundei curente, astfel încât toate ecranele să știe cât timp a mai rămas și care rundă este activă.
- Starea mineritului și clasamentele, astfel încât rezultatele și clasamentele de minerit să fie consistente în toate părțile de piață, statistici și alte perspective.
- O vedere sincronizată a ceasului serverului, construită din Socket.IO mesaje, care permite clientului să obțină timpul rămas fără a se îndepărta prea mult de serverul autoritar.
- Mai multe protecții de "navigație sigură", care previn condițiile de cursă atunci când navigația se schimbă în același timp cu procesarea noilor evenimente (de exemplu, evitând ca un ecran să citească date învechite chiar la finalul rundei).

Dincolo de simpla stocare a datelor, *GameContextHybridRobust* integrează și un set de mecanisme de idempotență și resincronizare în jurul conexiunii WebSocket. Fiecare eveniment relevant poartă identificatori precum *roundId* și version, iar contextul ține evidența a ceea ce a fost deja aplicat. Dacă conexiunea se întrerupe și ulterior se recuperează, contextul poate resolicita sau reconcilia starea autoritară de la server în loc să aibă încredere orbă în ceea ce a fost redat ultima dată. Acesta este motivul pentru care un jucător poate bloca telefonul pentru o clipă sau să piardă Wi-Fi-ul pentru scurt timp fără a-și strica complet jocul.

Logica de minerit este încapsulată în continuare într-un context dedicat, adesea denumit *SimpleMiningContext* și expus prin *SimpleMiningProvider*. Ideea aici este să păstrăm o coadă ușoară a problemelor de minerit și gestionarea interfeței lor separată de restul stării jocului. Când serverul autoritar emite noi provocări de minerit, acesta este împins în această coadă; Când utilizatorul răspunde sau cronometrul expiră, aceștia sunt procesați și eliminați. Deoarece mineritul este gestionat în acest context izolat, restul interfeței poate rămâne receptiv chiar dacă mai multe probleme sunt create și rezolvate rapid, iar interfața legată de minerit (precum modalele sau numărătoarele inverse) nu trebuie să fie conectată manual pe fiecare ecran.

În jurul acestor două contexte principale, clientul folosește un set de hook-uri auxiliare pentru a grupa reguli de business specifice:

- *useLearningProgress* ascultă evenimentele jocului și înregistrează realizările educaționale (de exemplu, blocuri extrase, deșeuri reduse) în Firestore și poate declanșa feedback vizual atunci când anumite repere sunt atinse.
- *useNetworkStatus* înfășoară API-ul NetInfo și transmite *NetworkStatusBanner*, astfel încât orice pierdere temporară de conectivitate să fie imediat vizibilă pentru jucător.

- *useTutorial* controlează fluxurile de onboarding pentru utilizatorii începători, hotărând când să arate explicații sau indicii și când să facă un pas înapoi pentru a lăsa jucătorul să interacționeze liber.

Împreună, aceste contexte și cârlige implementează un principiu clar de design: ecranele trebuie să fie cât mai simple posibil, în timp ce comportamentul transversal (cronometre, jucători, minerit, progres în învățare, conectivitate, tutoriale) se află în module partajate, testabile. Acest lucru face ca aplicația să fie mai ușor de raționat, mai ușor de extins (ecranele noi pot reutiliza aceleași hook-uri) și mai robustă în condiții de timp real, deoarece există un singur loc unde starea jocului este derivată din evenimentele din backend.

2.3. Firebase backend: autentificare și date persistente

În backend, RockChain se bazează pe Firebase ca platformă gestionată care integrează trei servicii cheie: Autentificare, Cloud Firestore și Cloud Functions. Împreună, aceste servicii permit gestionarea securizată a identității fiecărui utilizator, stocarea persistentă a datelor din joc și progresul învățării, precum și executarea controlată a operațiunilor sensibile direct pe server. Deși toate detaliile tehnice despre modelul de date, fluxurile și regulile de securitate sunt documentate în livrabilul WP4-A1, iată o prezentare generală a modului în care această infrastructură este folosită în cadrul instrumentului interactiv.

Inima sistemului este Cloud Firestore, care acționează ca "memoria pe termen lung" a RockChain. În loc de tabele ca într-o bază de date tradițională, Firestore organizează informațiile în colecții și documente. În mediul de producție, principalele colecții sunt:

- *utilizatori*: conține un document per jucător, cu profilul lor de bază și performanța în fiecare joc (rockCoins, deșeuri, produse), grupate după identificatorul jocului.
- *Jocuri*: Un document pe sesiune de joc, unde sunt stocate codul de acces, gazda, lista jucătorilor, starea curentă, runda și anumiți indicatori-cheie folosiți atât de client, cât și de server în timp real.
- *Piață*: Cataloage de produse și prețuri dinamice per joc, păstrate separate de documentul principal pentru a nu fi supraîncărcate cu actualizări frecvente.
- *Blockchain*: O istorie simplificată a blocurilor extrase în fiecare joc, înregistrând cine a minat ce și când, permițând ca mecanicile inspirate de blockchain să fie reflectate în date.

Pentru operațiuni critice, clienții nu scriu direct în aceste colecții. În schimb, ele invocă funcții server (Funcții Cloud) care rulează cu privilegii speciale, aplică validări și reguli de

business și modifică datele în Firestore. Aceste funcții sunt detaliate în raportul WP4-A1, dar în cadrul aplicației au trei roluri principale:

- Managementul ciclului de viață al jocului: funcții precum createGame, *joinGame* și *deleteGame* creează și șterge documente de joc, verifică unicitatea codului și previne sesiunile învechite de la aceeași gazdă.
- Logica pieței și mineritului: Funcții precum *assignProductsToGame*, *updateProductPrices*, *generateMiningProblem* și *submitMiningSolution* controlează modul în care evoluează produsele, prețurile și provocările miniere, asigurând reguli consistente în toate jocurile.
- Recompense și profiluri: La finalul fiecărei runde sau joc, funcții precum *assignRoundRewards* și *updateUserProfile* consolidează rezultatele și actualizează profilurile utilizatorilor și documentele instantanee, menținând consistența între ceea ce a fost văzut pe ecran și ceea ce este salvat permanent.

În plus, integrarea cu Firebase include sistemul de autentificare și un mic strat de stocare locală pe dispozitiv (*AsyncStorage*). Autentificarea asigură că fiecare cerere către backend este legată de un ID de utilizator unic, folosit consecvent în Firestore și pe serverul WebSocket. Stocarea locală permite aplicației să își amintească sesiunile active și să salveze date esențiale între lansări, evitând întreruperile în cazul unor întreruperi scurte ale conexiunii sau închiderii accidentale a aplicației.

Pentru detalii despre implementarea completă, inclusiv regulile de securitate, fluxurile de date și aspectele legate de GDPR, partenerii pot consulta raportul tehnic WP4-A1.

2.4. Server autoritar și servicii în timp real

Deși Firebase gestionează stocarea și logica serverului în mod sigur și persistent, RockChain are nevoie și de un strat care să răspundă aproape instantaneu la acțiunile jucătorului, să gestioneze cronometrele și să decidă în timp real cine câștigă provocările de minerit. Pentru a răspunde acestei nevoi, proiectul folosește un server în timp real dezvoltat cu Node.js și Socket.IO, ambalat într-o imagine Docker și implementat pe Google Cloud Run. WP4-A1 detaliază această arhitectură, dar iată un rezumat al modului în care suportă instrumentul interactiv.

Pe scurt, acest server acționează ca arbitru pentru fiecare meci aflat în desfășurare. Pentru fiecare joc, păstrează în memorie:

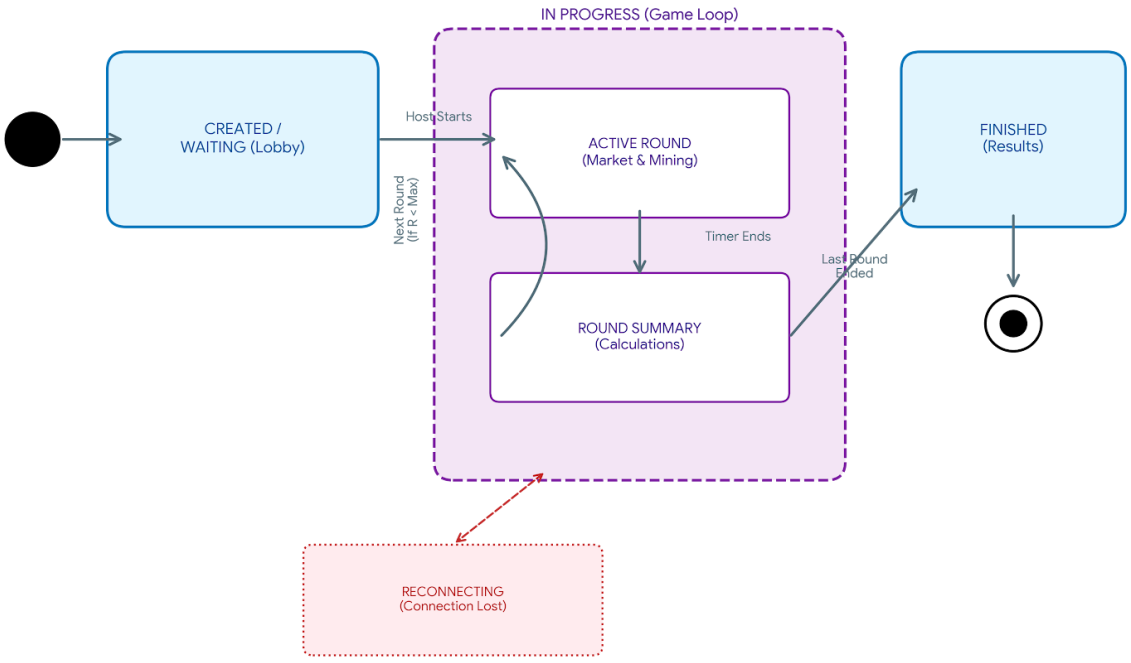
- Care jucători sunt conectați și în ce cameră.
- Stocurile lor actuale și pozițiile lor în clasament.
- Provocarea mineritului activ (dacă există).
- Starea oficială a rundei, inclusiv ora exactă la care ar trebui să se încheie.

Pe baza acestei statusuri, serverul generează identificatori precum *roundId*, *version* și *roundEndsAtMs*, care sunt atașate evenimentelor trimise. Acest lucru permite atât clientului, cât și backend-ului să știe la ce rundă se referă în orice moment și evită confuzia cu mesajele duplicate sau întârziate.

Comportamentul jocului este structurat ca o mașină de stări simplă, cu patru faze principale:

- JOC: runda este activă; Jucătorii pot cumpăra, recicla sau rezolva probleme de minerit în timp ce cronometrul funcționează.
- ROUND_CALCULATING: acțiunile sunt puse pe pauză, iar serverul calculează rezultatele.
- ROUND_END: valorile finale sunt acum disponibile; Jucătorii pot vedea recompensele și pozițiile lor.
- WAITING_FOR_NEXT_ROUND: jocul este pus pe pauză până când gazda decide să înceapă o nouă rundă sau să încheie jocul.

Pe măsură ce jocul avansează prin aceste faze, serverul emite diferite evenimente precum *start_round, ROUND_CALCULATING, round_ended, game_completed, inventory_data, player_rankings_data* sau *economy:*state. Aplicația mobilă ascultă aceste evenimente și actualizează ecranele, cronometrele și rezumatele în timp real, asigurând că toți jucătorii au o perspectivă consecventă asupra a ceea ce se întâmplă.



Figură 3: Mașină de stări Rockchain.

Deoarece multe acțiuni pot apărea aproape simultan (mai ales la finalul unei runde), serverul folosește mecanisme de blocare și actualizări idempotente pentru a evita

conflictele. Operațiunile critice, cum ar fi atribuirea recompenselor sau actualizarea inventarelor, sunt executate într-un mod ordonat, fără suprapuneri, și sunt marcate cu identificatori de rundă pentru a ignora repetițiile. Când serverul poate accesa Firebase, salvează și rezultatele fiecărei runde (inventare, recompense, stare economică), asigurând că snapshot-ul din memorie și datele stocate permanent sunt aliniate.

Pe partea clientului, toată această complexitate este încapsulată într-un adaptor Socket.IO. Acest adaptor selectează automat serverul corect (local sau Cloud Run), deschide și menține conexiunea WebSocket, transmite autentificarea și informațiile jocului după reconectare și traduce evenimentele backend în formatul așteptat de aplicația dezvoltată de React. Datorită acestui strat intermediar, serverul poate evolua în timp fără a afecta experiența utilizatorului sau funcționarea pedagogică a instrumentului

2.5. Preocupări transversale: internaționalizare, configurare și observabilitate

Dincolo de componentele principale pentru client și backend, RockChain include mai multe mecanisme de tăiere transversală care fac unealta utilizabilă în diverse țări, mai ușor de implementat în medii diferite și mai ușor de depanat în timpul piloților. Trei dintre ele sunt deosebit de relevante: internaționalizarea, configurarea mediului și observabilitatea.

2.5.1. Internaționalizare și accesibilitate

Încă de la început, RockChain a fost conceput ca un instrument multi-țară. Aceasta însemna că sprijinul lingvistic nu putea fi o idee de ultim moment. În schimb, proiectul a integrat un strat de internaționalizare bazat pe i18next.

Configurația de bază se află în *src/i18n/index.js*, care:

- Încarcă pachete de limbă pentru engleză (EN), spaniolă (ES), germană (DE), croată (HR) și română (RO).
- Detectează limba dispozitivului sau folosește alegerea explicită a jucătorului pentru a decide ce pachet să activeze.
- Expune o instanță i18n pe care restul aplicației o folosește prin I18nextProvider.

Toate textele orientate către utilizator sunt stocate ca chei de traducere în *src/i18n/settings/*.json*. Pentru fiecare limbaj suportat, există un fișier JSON corespunzător unde acele chei sunt mapate în șiruri reale. Componentele solicită apoi texte după cheie, în loc să codeze direct engleza sau orice altă limbă.

Pentru a face experiența persistentă, limbajul selectat este salvat în AsyncStorage. Aceasta înseamnă că:

- Prima dată când aplicația rulează, poate folosi implicit limba dispozitivului.
- Dacă utilizatorul schimbă limba prin interfață, acea preferință este salvată local.
- La lansările ulterioare, aplicația restabilește aceeași limbă fără să mai întrebe.

Adăugarea unui limbaj nou este simplă și nu necesită atingerea codului de bază:

1. Creează un fișier JSON nou sub *src/i18n/localități/* cu aceleași chei ca limbajele existente.
2. Înregistrează noul cod de limbă în *suportedLngs* în *src/i18n/index.js*.
3. Oferă traduceri pentru toate cheile relevante.

Acest design susține unul dintre obiectivele de bază ale RockChain: instrumentul poate fi transferat către alte țări și contexte lucrând la fișiere de traducere, în loc să modifice logica aplicației.

2.5.2. Configurarea mediului și detectarea rețelei

Deoarece RockChain trebuie să ruleze în mai multe contexte de execuție (dezvoltare locală, staging și producție), proiectul evită codarea directă a URL-urilor sau a presupunerilor de mediu. În schimb, folosește un strat de configurare mic, dar explicit.

Două module sunt centrale aici:

src/config/environment.js

- Calculează valori precum SOCKET_URL, NODE_ENV, timeout-uri și steaguri de depanare.
- Citește setările de build-time și variabilele de mediu pentru a decide, de exemplu, dacă aplicația ar trebui să comunice cu un server local Socket.IO, o instanță staging sau serviciul de producție Cloud Run.

src/utils/networkUtils.js

- Detectează dacă aplicația rulează într-un simulator/emulator sau pe un dispozitiv fizic.
- Pe baza acestor informații, alege adresa serverului potrivită:
    - Localhost sau un anumit IP LAN în timpul dezvoltării.
    - URL-ul Cloud Run în staging sau build-uri de producție.

Pe lângă asta, variabilele de mediu Expo (cum ar fi EXPO_PUBLIC_WEBSOCKET_URL) permit proiectului să suprascrie endpoint-urile din profiluri de build specifice. De exemplu, o compilație EAS de producție poate conecta prin cablu punctul final WebSocket la URL-ul oficial Cloud Run, în timp ce o compilație de previzualizare poate indica o instanță de testare.

Efectul net este că aceeași bază de cod poate fi reconstruită pentru medii diferite pur și simplu alegând profilul de build și configurația potrivită, fără a edita fișierele sursă. Aceasta susține direct reproducibilitatea și facilitează clonarea sau actualizarea implementării pentru parteneri.

2.5.3. Observabilitate și reziliență

În final, arhitectura include un set minim de instrumente pentru a înțelege ce se întâmplă în timp real și pentru a supraviețui problemelor temporare de rețea. Acest lucru este deosebit de important în contextul unei săli de clasă, unde calitatea Wi-Fi-ului poate varia.

Pe partea clientului:

- *GameContextHybridRobust* înregistrează informații despre deplasarea ceasului, evenimente duplicate și încercări de resincronizare către consolă. În timpul dezvoltării și testelor pilot, aceste jurnale ajută la identificarea situațiilor în care clientul și serverul autoritar diverg temporar și cât de des sunt necesare resincronizările.
- Combinația dintre reconectarea automată a socket-ului și React Context asigură că, atunci când conexiunea este restabilită, clientul poate solicita din nou starea autoritară curentă, în loc să se bazeze pe date învechite.

Pe partea de server:

- Un utilitar logMetric *emite evenimente JSON structurate pentru momente cheie precum conexiunea,* round_start*,* round_end, resincronizarea și eroarea. Când serverul rulează pe Cloud Run, aceste evenimente pot fi capturate de Google Cloud Logging sau instrumente similare, oferind o cronologie a ceea ce s-a întâmplat în fiecare sesiune de joc.
- Serverul impune gestionarea evenimentelor idempotente folosind identificatori precum *roundId*, version și, acolo unde este cazul, *operationId*. Acest lucru previne recompensele duplicate sau starea inconsistentă atunci când mesajele sunt reîncercate sau ajung cu întârziere.
- Blocajele pe rundă (*roundLocks*, *gameStates*) asigură că secțiunile critice — cum ar fi calculele de la sfârșitul rundei — nu sunt executate simultan pentru același joc și rundă.

Luate împreună, aceste mecanisme înseamnă că arhitectura nu este doar funcțională pentru piloți, ci și trasabilă și robustă. Dezvoltatorii și partenerii tehnici pot inspecta jurnalele pentru a înțelege cum au evoluat sesiunile, în timp ce jucătorii experimentează un joc care continuă să funcționeze chiar și atunci când apar probleme de conectivitate de scurtă durată.

3. FLUXURI CHEIE ÎN TIMP DE RULARE ÎN INSTRUMENTUL INTERACTIV ROCKCHAIN

Deși secțiunea anterioară explică cum este construită arhitectura RockChain, aici ne concentrăm pe ceea ce experimentează o persoană când folosește instrumentul: cum se conectează, cum se alătură sau creează un joc, cum progresează rundele și ce fel de răspunsuri oferă sistemul. Pe scurt, analizăm procesul pas cu pas din perspectiva utilizatorului.

3.1. Autentificare și integrare

Totul începe pe ecranul de autentificare (*LoginScreen*), unde utilizatorii pot face două lucruri:

- Creează un cont nou introducând informațiile minime necesare (cum ar fi o adresă de email, parolă și un nume vizibil în joc).
- Conectează-te cu un cont existent, reutilizând datele salvate.

Acest ecran este conectat direct la sistemul de autentificare Firebase. Când cineva introduce datele și apasă pe "Enter", aplicația face următoarele:

1. Trimite acreditările către Firebase pentru verificare.
2. Dacă totul este corect, Firebase returnează un ID de utilizator autentificat și un token de acces.
3. Dacă există o eroare (de exemplu, parolă incorectă sau email deja înregistrat), aplicația afișează un mesaj clar și cere corectarea datelor.

Ce se întâmplă după ce te-ai conectat cu succes?

Trei lucruri se întâmplă automat și aproape simultan:

Profilul este creat (sau recuperat) în baza de date.

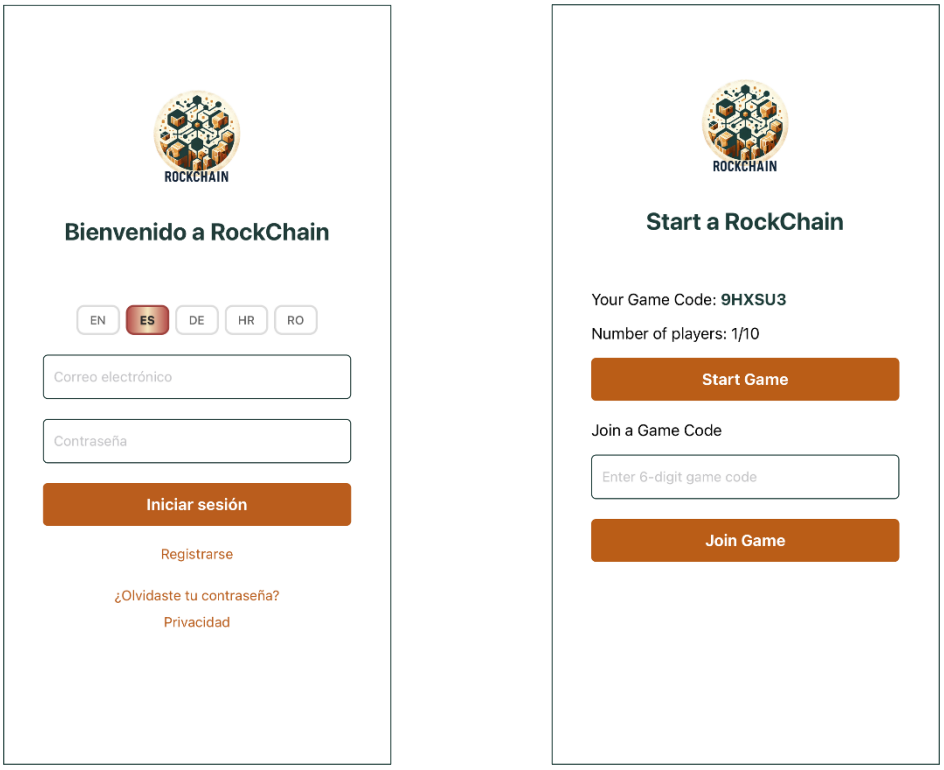
- Dacă este un cont nou, aplicația sau o funcție de pe server creează un document în baza de date cu informații de bază: nume vizibil, email, data creării, limba implicită etc.
- Dacă a existat deja, profilul este refolosit, împreună cu istoria jocurilor anterioare, dacă există.

Un token securizat este obținut pentru acțiunile jocului.

- Acest token permite aplicației să comunice cu serverul și să acceseze funcții protejate, cum ar fi crearea sau aderarea la jocuri.
- Este folosită și pentru a se conecta la server în timp real, astfel încât sistemul să știe cine se află în spatele fiecărei acțiuni.

Accesează zona principală de joc.

- Aplicația încetează să mai afișeze ecranul de autentificare și merge direct în zona de joc, unde persoana poate:
- Începe un joc nou.
- Alătură-te unui joc existent introducând un cod.
- Vezi informații de bază despre RockChain.
- Sau să vezi realizările lor anterioare.



Figură 4: Ecrane de AUTENTIFICARE și JOC.

RockChain nu face diferența între "conturi normale" și "conturi gazdă". Oricine este autentificat poate crea un joc și devine gazdă dacă își împărtășește codul și alți jucători se alătură sesiunii. Acest lucru face ca procesul să fie același pentru toată lumea: fiecare persoană accesează spațiul de joc, iar apoi grupurile decid natural pe care să le folosească ca joc comun.

3.2. Crearea jocului și aderarea la jucători

Ciclul de viață al unui joc pe RockChain este conceput să fie cât mai simplu și fluid posibil. Odată ce o persoană se conectează, aplicația o tratează ca pe toți ceilalți: i se atribuie

automat propriul joc, pe care îl găzduiește, iar de acolo poate decide dacă îl folosește ca sesiune pentru grupul său sau dacă se alătură altcuiva folosind un cod.

Nu este nevoie să alegi între "creare" sau "găzduire": dacă împărtășești codul tău și începi jocul, tu ești gazda. E atât de simplu.

3.2.1. Joc gazdă automat pe *GameScreen*

Imediat după ce se conectează cu succes, aplicația duce persoana la ecranul principal al jocului. În acel moment, aplicația creează automat un joc personal în fundal: nu este nevoie să apeși niciun buton "create".

În backend, acest proces se ocupă de:

- Verificarea dacă acea persoană avea deja jocuri active anterioare și, dacă da, închiderea lor pentru a evita duplicatele.
- Asigurarea faptului că catalogul global de produse și datele de bază ale pieței sunt disponibile.
- Crearea unui nou document de joc cu:
  - Un cod unic de partajat (cum ar fi *ABC123*)
  - ID-ul utilizatorului ca gazdă
  - Stare inițială: "*în așteptare*"
  - Contorul rundului setat la zero
  - Și orice setări implicite care ar putea fi necesare

Când crearea este finalizată, aplicația are tot ce îi trebuie pentru a începe. Pe ecran, persoana va vedea ceva de genul.

- "*Codul tău de joc: ABC123*" în partea de sus.
- "*Jucători conectați: 1*".
- Un buton "*Start game*".
- Un câmp pentru a introduce un cod dacă vrea să se alăture unei alte sesiuni.

Dacă decizi să folosești jocul ca o sesiune de grup, pur și simplu:

- Împărtășește-ți codul cu coechipierii tăi (cu voce tare, pe ecran, prin chat etc.).
- Așteaptă ca toată lumea să se conecteze (contorul jucătorilor arată asta).
- Apasă "Start joc."
- Nu este necesară nicio altă acțiune: acel joc personal devine jocul oficial de grup.

3.2.2. Să te alături jocului altcuiva

Se poate întâmpla opusul când ajungi la acest ecran, poate că ai deja propriul tău joc creat, dar vrei să te alături unui alt joc pe care un prieten l-a început deja.

Pentru asta există zona "Alătură-te unui joc". Tot ce trebuie să faci este:

- Introduceți codul de sesiune partajat de gazdă.

- Apasă butonul "*Alătură-te jocului*".

Aplicația va:

- Caută acel joc folosind codul.
- Verifică dacă este încă deschis pentru jucători noi.
- Adaugă utilizatorul pe lista participanților.
- Creează sau actualizează datele lor personale de joc (*monede, inventar, deșeuri etc*.).
- Schimbă contextul jocului lor: din acel moment, devin parte a sesiunii gazdei.

Din acel moment, persoana vede codul gazdei și numărul corect de jucători conectați pe ecran. Când grupul este complet, gazda poate apăsa "*Start game*", ceea ce îi va duce pe toți în sala de așteptare și va începe jocul.

3.3. Sala de așteptare și sincronizarea

Odată ce o persoană decide să folosească propriul joc ca gazdă sau se alătură altcuiva, toți participanții sunt duși de pe ecranul principal al jocului în sala de așteptare. Acest ecran are un rol foarte specific: să adune grupul într-un punct de pornire stabil și să se asigure că, atunci când începe numărătoarea inversă, toate dispozitivele sunt perfect sincronizate.

Lobby-ul nu își gestionează separat propriul stat. În schimb, ascultă două surse cheie de informații:

- *GameContextHybridRobust*, care colectează:
  - Actualizări ale documentului jocului (*games*/{*gameId*}), cum ar fi sosirea de jucători noi sau modificări de stare
  - Lista actuală a jucătorilor și starea generală a jocului (în așteptare, în joc, terminat etc.)
- Evenimente servere în timp real (*Socket.IO),* care indică:
  - Începutul numărătoarei inverse,
  - Schimbări de fază a jocului (de exemplu, trecerea de la "*așteptare*" la "*jucare*")

Cu aceste informații, holul afișează doar elementele esențiale:

- Lista jucătorilor conectați la acel joc,
- Starea curentă a jocului (mesaje precum "*Așteptăm jucătorii*", "*Gata de început*", "*Începem în 3...2...1...*"),
- Indicatori opționali că jucătorii sunt pregătiți, dacă această funcție a fost activată

Ce se întâmplă când gazda apasă "*Start game*"?

Când gazda consideră că grupul este pregătit și apasă "*Start game*":

1. Dispozitivul tău apelează *funcția startGame* din backend, trimițând ID-ul jocului.
2. Serverul verifică că:
    - Jocul este într-o stare validă (de exemplu, nu este deja în desfășurare și sunt suficienți jucători)
    - Coordonează cu serverul în timp real pentru a pregăti începutul primei ture:
        - Este generat un identificator unic de rundă (*roundId*)
        - Se calculează timpul exact la care se va încheia runda (*roundEndsAtMs*)
        - Jocul este marcat ca pregătit pentru a trece la faza de joc după numărătoare inversă

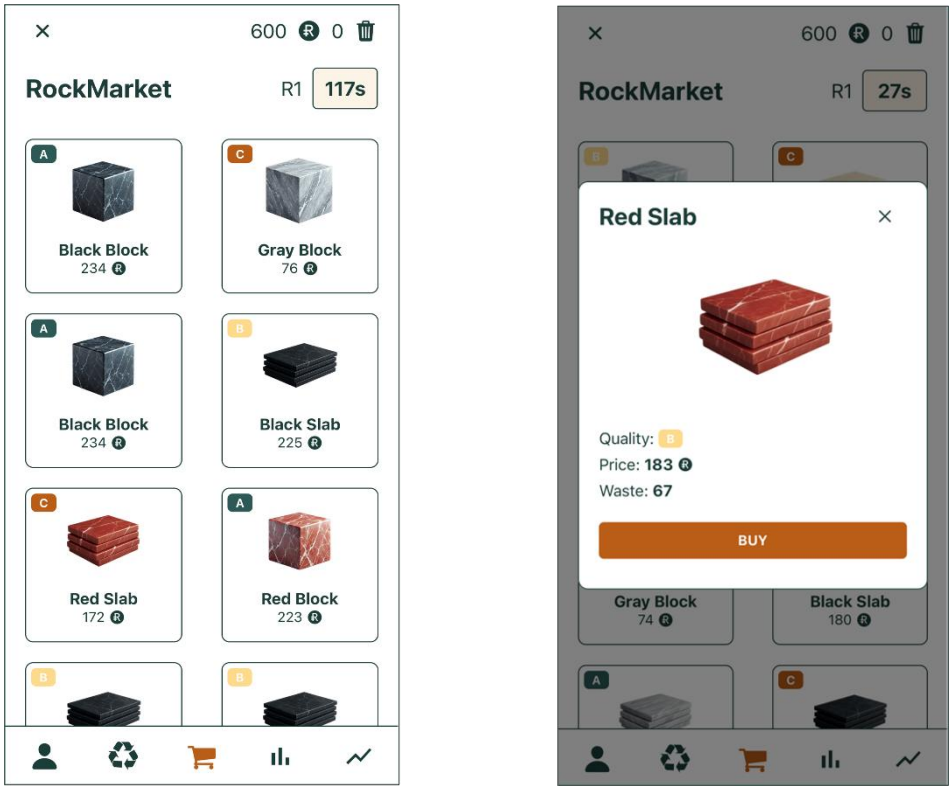
Serverul emite apoi o serie de evenimente prin *Socket.IO*:

- Un eveniment de start cu numărătoarea inversă ("*runda va începe în câteva secunde*")
- Evenimente opționale care marchează progresul numărătoarei inverse ("*3... 2... 1...*")
- Un eveniment care indică faptul că runda a început oficial

Fiecare sală de așteptare primește aceste evenimente, iar *GameContextHybridRobust* folosește timestamp-ul (*roundEndsAtMs*) pentru a calcula timpul rămas pe fiecare dispozitiv. Chiar dacă există mici diferențe în conexiunea Wi-Fi, toți jucătorii părăsesc sala de așteptare și intră în rundă cu același timp limită de timp

Sala de așteptare funcționează ca o cameră de sincronizare. Aceasta asigură că:

- Toată lumea este conectată și vizibilă în joc.
- Gazda poate decide momentul exact de început.
- Întregul grup începe runda în același timp, datorită unei combinații de funcții cloud și comunicare în timp real.



Figură 5: Ecranul MARKET.

3.4. Gameplay în timpul rundei: navigarea între ecrane

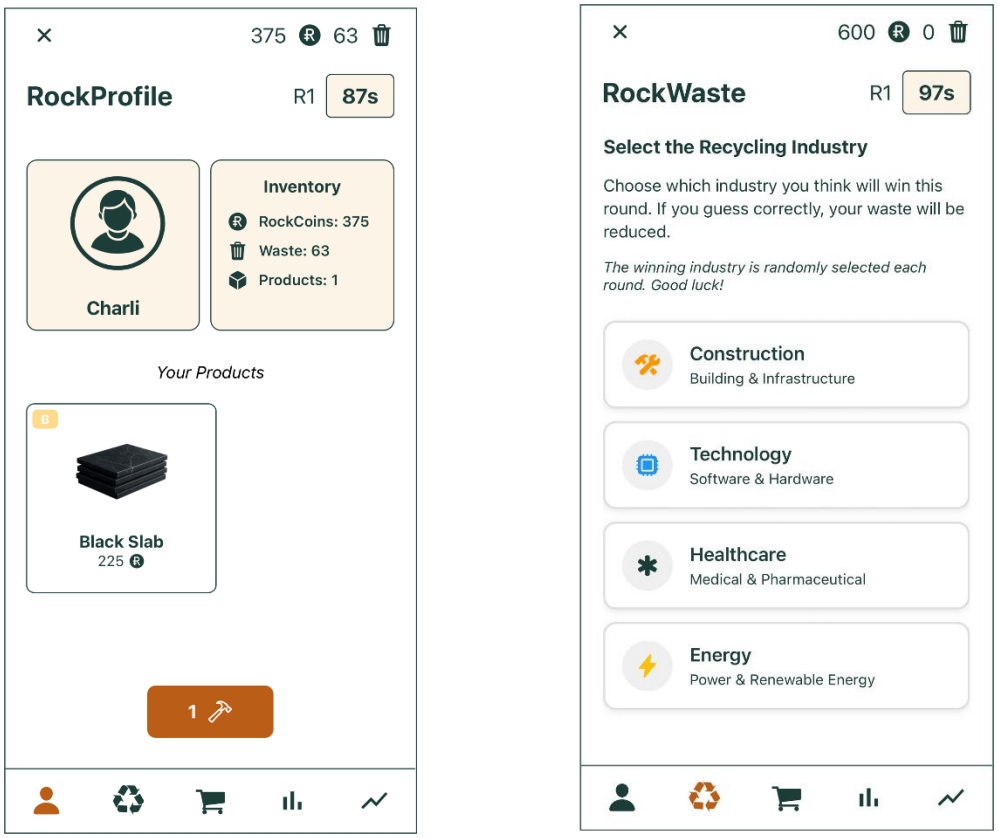
Când începe o rundă, aplicația mută automat toți jucătorii din lobby către interfața principală a jocului, numită ***GameTabs***. De atunci, fiecare persoană se poate deplasa liber între mai multe ecrane-cheie pe durata rundei.

Timpul de joc este controlat de o valoare centrală numită *roundEndsAtMs*, care definește exact când ar trebui să se încheie runda. Această valoare este furnizată de server în timp real, iar contextul jocului (*GameContextHybridRobust*) o convertește într-un numărătoare inversă locală, astfel încât toate dispozitivele să aibă aceeași limită de timp, indiferent de întârzieri minore de conexiune.

În timpul unei runde tipice de 120 de secunde, jucătorii pot interacționa cu următoarele module:

- *MarketScreen*: permite jucătorilor să cumpere și să vândă produse (*blocuri, plăci, materiale reciclate*). Lista de produse este actualizată automat dacă serverul schimbă prețurile sau aprovizionarea.
- *RecycleScreen*: aici poți transforma o parte din inventarul tău în RockCoins sau materiale noi. Fiecare acțiune este validată în backend, care verifică dacă regulile sunt respectate înainte de a o aplica.

- *MiningScreen*: oferă provocări de tip "dovadă de muncă". Aplicația afișează probleme matematice cu o limită de timp, jucătorul își trimite răspunsul, iar serverul decide cine a avut dreptate și cine a fost cel mai rapid.
- *StatsScreen*: afișează indicatori de performanță și clasamente, cu grafice actualizate în funcție de progres.
- *ProfileScreen*: oferă un rezumat al profilului utilizatorului, resurselor curente și RockCoin-urilor. De asemenea, îți permite să schimbi limba. Informațiile provin direct de la Firestore.



Figură 6: ECRANE PROFILE și RECYCLE.

- *StockScreen*: oferă o prezentare generală a pieței: ce produse sunt disponibile, cum variază prețurile și ce resurse sunt rare sau abundente. Este util pentru planificare înainte de cumpărare sau reciclare.

Un principiu de bază rămâne constant pe toate aceste ecrane: aplicația nu ia niciodată deciziile finale de una singură. Fiecare acțiune efectuată de jucător este interpretată ca o intenție trimisă serverului. De exemplu:

- *"Vreau să cumpăr produsul X la prețul actual"*
- *"Vreau să reciclez aceste deșeuri în această industrie"*
- *"Acesta este răspunsul meu la provocarea mineritului"*

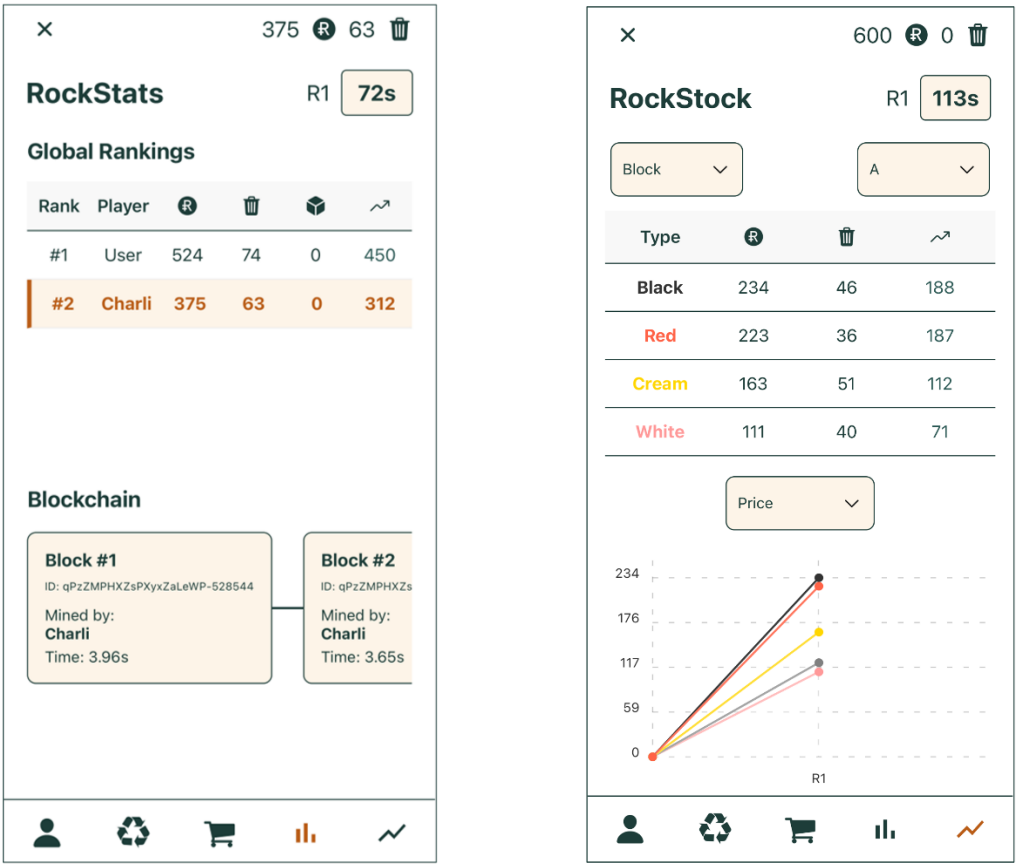
Serverul în timp real primește fiecare acțiune, verifică dacă este validă în funcție de starea curentă a jocului (inventar, prețuri, cronometru etc.) și:

- Îl aplică dacă totul este în regulă, actualizând datele în memorie (și în Firestore dacă este necesar).
- Îl respinge dacă nu respectă regulile sau sosește prea târziu.

Serverul trimite apoi actualizările tuturor jucătorilor: inventaruri noi, clasamente, modificări de preț sau rezultate de minerit.

Datorită acestei structuri, fiecare rundă este:

- Interactiv: jucătorii pot trece de la un ecran la altul și pot lua decizii liber.
- Corect și consecvent: aceleași reguli se aplică tuturor, iar starea jocului este menținută sincronizată pentru întregul grup.
- Controlat de timp: valoarea *roundEndsAtMs* asigură că toți jucătorii termină runda în același timp, indiferent de diferențele minore dintre dispozitivele sau rețelele lor.



Figură 7: STATURI și ECRANE STOCK

3.5. Calcule de final de rundă și rezultate finale

Când timpul rundei expiră (*roundEndsAtMs*), sistemul încetează să mai accepte acțiuni noi de la jucători. Din acel moment, controlul trece complet către backend, care este responsabil pentru înghețarea stării jocului, calcularea rezultatelor și afișarea informațiilor într-un mod clar și consecvent.

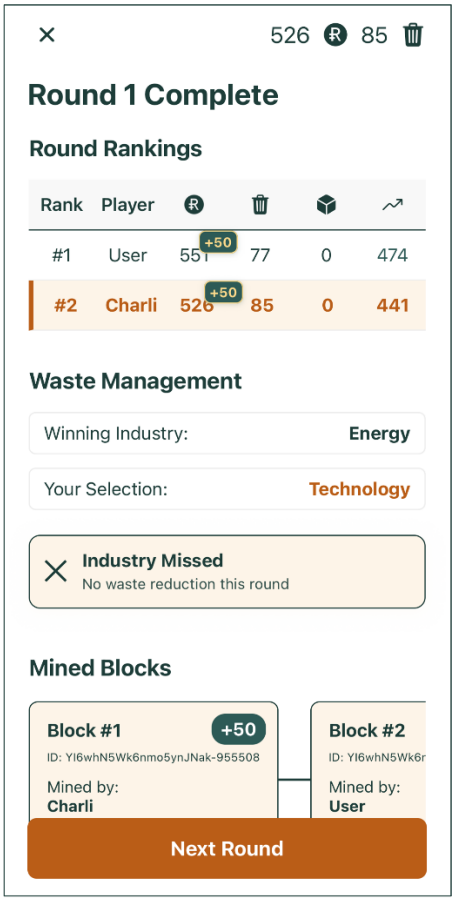
Pe serverul în timp real (Socket.IO), procesul de închidere a rundei urmează următorii pași:

1. **Acțiuni de înghețare:**
   Serverul schimbă starea jocului la *ROUND_CALCULATING* și încetează să mai accepte mișcări noi. Orice mesaj care ajunge târziu este ignorat sau marcat ca invalid.
2. **Calculează recompensele în siguranță:**
   Folosind blocaje la nivel de rundă și verificări pentru a preveni repetarea, serverul:
   - Procesează fiecare acțiune o singură dată
   - Evită duplicarea recompenselor sau numărarea aceluiași eveniment de două ori
   - Asigură că acțiunile simultane nu generează erori sau date inconsistente
3. **Alegerea industriei câștigătoare:**
   Logica configurată este aplicată pentru a determina care industrie a câștigat runda, o informație cheie pentru feedback-ul pe care jucătorii îl vor primi.
4. **Salvarea rezultatelor în Firestore**
   Actualizările serverului:
   - Documentul jocului în *games*/{*gameId*} (status, numărul rundei, scorurile)
   - Statisticile individuale din *users*/{*userId*}.*jocuri*[*gameId*]

Dacă este disponibil, Firebase Admin gestionează aceste operațiuni în mod sigur și eficient.

5. **Notificați dispozitivele**
   Odată ce calculele au fost făcute și datele salvate, serverul trimite o serie de evenimente precum:
   - *ROUND_CALCULATING*
   - *ROUND_ENDED*
   - *REWARDS_ASSIGNED*

   Aceste mesaje permit clienților să închidă faza activă a jocului, să arate că runda s-a încheiat și apoi să afișeze rezultatele detaliate.



Figură 8: Sfârșitul ecranului rotund.

În aplicație, contextul jocului (*GameContextHybridRobust*) ascultă aceste evenimente și actualizează interfața. Trece de la vizualizarea activă a jocului la ecrane de rezumat, precum:

- *EndOfRoundScreen* arată industria câștigătoare, cine a rezolvat provocarea mineritului (dacă a existat una), cum s-au schimbat inventarul fiecărui jucător și RockCoin-urile.
- *GameResultsScreen* (sau un ecran final similar) arată clasamentul general al jucătorilor, statisticile acumulate din rundele jucate și alți indicatori utili pentru discuție sau reflecție.

Aceasta închide runda pentru întregul grup. Deoarece toate datele importante sunt stocate în Firestore, jucătorii pot:

- Să revină pe ecranul principal și să încep un joc nou ca gazde cu același grup sau cu altul, sau
- Ieși din joc și continuă cu celelalte activități ale traseului.

Jocurile finalizate și datele lor rămân disponibile pentru utilizare ulterioară în WP5 (evaluări, analize sau recenzii în sesiuni viitoare), indiferent ce fac jucătorii în continuare.

Împreună, acest flux completează ciclul care a început la autentificare: trece de la autentificare la configurarea jocului, la experiența activă și, în final, la o concluzie cu rezultate clare. Datorită combinației unui server în timp real, a unei structuri solide în Firestore și a unor ecrane de rezumat bine proiectate, arhitectura tehnică devine o experiență practică, repetabilă și utilă în diferite contexte educaționale.

4. GĂZDUIRE, ACCES ȘI DISTRIBUȚIE

Din punct de vedere operațional, instrumentul interactiv RockChain este livrat ca un serviciu cloud găzduit plus o aplicație mobilă. Centrele de instruire nu trebuie să instaleze sau să întrețină servere locale; Au nevoie doar de acces la internet și sunt potrivite pe cel puțin una dintre platformele principale.

Această secțiune oferă o perspectivă concisă și tehnică asupra modului în care instrumentul este găzduit și accesat. Pentru instrucțiuni pas cu pas, orientate către instructori, despre pregătirea și rularea unei sesiuni, cititorii pot consulta "Notele ghidului" WP4-A4, astfel încât utilizatorii să poată folosi instrumentul cu ușurință.

4.1. Găzduire backend

Backend-ul este găzduit pe ***Google Cloud*** în cadrul proiectului RockChain și este compus din elementele descrise în Secțiunea 2:

- *Cloud Firestore*: stochează jocuri, utilizatori, date de piață și înregistrări legate de învățare.
- *Firebase Cloud Functions*: implementează operațiunile autentificate care creează și gestionează jocuri, generează probleme de minerit, atribuie recompense și actualizează profilurile utilizatorilor.
- *Node.js + Socket.IO server pe Cloud Run*: coordonează rundele în timp real și acționează ca cronometror și arbitru autoritar.

Toate datele de bază sunt stocate într-o regiune europeană, aliniind implementarea cu cerințele UE privind protecția datelor și simplificând conformitatea pentru partenerii care operează în diferite țări. Deoarece aceste servicii sunt complet gestionate, nu este nevoie de servere locale de baze de date sau infrastructură personalizată în centrele de educație: mentenanța și scalarea sunt gestionate la nivel de cloud.

4.2. Distribuția aplicațiilor mobile

Clientul RockChain este distribuit ca **aplicație mobilă** cel puțin pentru una dintre cele două platforme principale:

- O versiune Android (APK/AAB), potrivită pentru instalare prin listare de magazin sau distribuție directă către dispozitive gestionate.
- O versiune iOS (IPA prin TestFlight sau App Store), în funcție de strategia de distribuție convenită cu fiecare partener.

De asemenea, puteți găsi linkurile și/sau codurile QR publicate în zona web RockChain, aici: https://rockchain.eu/tool/

În practică, instructorii care pregătesc un curs pot vizita spațiul web RockChain, pot scana codul QR sau pot urma linkul pentru platforma lor și pot instala versiunea actuală de producție a aplicației pe dispozitivele lor sau pe tabletele deținute de instituție.

4.3. Accesează fluxul de lucru pentru formatori și cursanți

Pentru utilizatorii finali, accesul la RockChain urmează o secvență simplă, repetabilă:

1. **Instalează aplicația**
    - Cursanții și instructorii instalează aplicația RockChain pe dispozitivele lor Android sau iOS, fie pe telefoanele personale/tabletele, fie pe dispozitive deținute de instituție, unde aplicația poate fi preinstalată.
2. **Creează sau folosește un cont**
    - La prima utilizare, se înregistrează sau se autentifică prin Firebase Auth de pe *LoginScreen*, așa cum este descris în Secțiunea 3.1.
    - Conturile pot fi reutilizate pe parcursul sesiunilor și cursurilor, astfel încât utilizatorii să nu fie nevoie să se înregistreze din nou de fiecare dată.
3. **Alătură-te sau găzduiește sesiuni de joc**
    - Odată autentificat, fiecare jucător ajunge pe *GameScreen*, unde un cod personal de joc este pregătit pentru ei în fundal.
    - Grupuri mici decid pe care cod să folosească: un jucător acționează ca gazdă, împărtășind codul și începând jocul, iar ceilalți se alătură folosind inputul "Join game".
    - Antrenorii pot permite fie cursanților să-și găzduiască propriile jocuri (câte unul pe grup), fie să acționeze ca gazde dacă vor să ruleze un demo sau să controleze mai strict sincronizarea.

Din perspectiva unui antrenor, asta înseamnă că organizarea unei sesiuni RockChain constă în principal din:

- Asigurându-se că toți participanții au aplicația instalată și se pot autentifica.

- Organizarea cursanților în grupuri mici (fiecare cu un gazdă care împărtășește un cod de joc).
- Monitorizarea progresului grupurilor prin runde și utilizarea ecranelor de rezultate pentru debriefing.

4.4. Cerințe pentru centrele de formare

Deoarece RockChain se bazează pe găzduire în cloud și distribuție mobilă, cerințele de infrastructură pentru centrele de instruire sunt minime:

- Rețea: o rețea Wi-Fi de clasă cu acces la internet, capabilă să suporte conexiuni simultane de la numărul de dispozitive folosite într-o sesiune.
- Dispozitive: smartphone-uri/tablete Android sau iOS care îndeplinesc cerințele minime de sistem de operare pentru versiunea implementată (așa cum este specificat în documentația RockChain).
- Fără instalare locală de servere: centrele nu trebuie să implementeze servere suplimentare; Toată coordonarea, stocarea și autentificarea jocurilor sunt gestionate în cloud.

Această combinație, backend Google Cloud, aplicații mobile pentru Android și iOS și acces prin zona web RockChain, permite utilizarea instrumentului interactiv RockChain în diferite țări și instituții fără o configurație locală complexă, păstrând totodată o implementare centralizată unică care poate fi menținută și actualizată de partenerii proiectului.

5. MONITORIZARE, REZILIENȚĂ ȘI ÎNTREȚINERE

Un ultim aspect al acestui document este asigurarea faptului că instrumentul interactiv RockChain poate fi monitorizat, depanat și întreținut în timp, mai ales în timpul piloților în săli de clasă reale. Scopul nu este construirea unui stack puternic de observabilitate, ci de a oferi suficientă vizibilitate și robustețe astfel încât partenerii să poată înțelege ce se întâmplă și să mențină sistemul stabil.

5.1. Observabilitate și înregistrare

Atât clientul, cât și serverul includ mecanisme ușoare de observabilitate care ajută în timpul dezvoltării, implementării piloților și analizei incidentelor.

Pe partea clientului, *GameContextHybridRobust* înregistrează semnale tehnice cheie către consolă, cum ar fi:

- Derivă de ceas între dispozitivul local și serverul autoritar.
- Încercări de resincronizare (de exemplu, după reconectare).
- Detectarea evenimentelor duplicate sau în afara ordinii.

Aceste jurnale sunt folosite în principal în timpul dezvoltării și testelor pilot, când dezvoltatorii sau partenerii tehnici rulează aplicația cu instrumente de depanare atașate. Acestea facilitează reproducerea și diagnosticarea problemelor legate de sincronizare, navigație și consistența stării.

Pe partea de server, un *utilitar logMetric* scrie evenimente JSON structurate în Cloud Logging. Evenimentele tipice includ:

- Conexiuni și deconectări noi.
- Runda începe și se termină.
- Resincronizarea operațiunilor.
- Erori sau condiții neașteptate.

Cu aceste jurnale, personalul tehnic poate:

- Vezi câte meciuri s-au jucat în fiecare perioadă.
- Identifică tipare de eroare sau blocaje de performanță (de exemplu, resincronizări repetate pentru anumite jocuri).
- Susține depanarea atunci când partenerii raportează incidente, corelând rapoartele utilizatorilor cu evenimente concrete din jurnale.

Abordarea generală este intenționat ușoară: nu există un tablou de bord complex de monitorizare, dar există suficiente informații structurate pentru a susține mentenanța de bază, optimizarea și depanarea.

5.2. Reziliența și gestionarea defecțiunilor

Deoarece RockChain este un instrument multiplayer în timp real folosit prin Wi-Fi în clasă, trebuie să facă față conectivității intermitente și a defecțiunilor temporare fără a strica experiența. Prin urmare, mai multe strategii de reziliență sunt integrate în arhitectură:

- Reconectarea automată a soclului: Adaptorul Socket.IO al clientului încearcă să se reconecteze automat atunci când rețeaua este pierdută temporar. După o reconectare reușită, retrimite tokenul de autentificare și *informațiile join_game* astfel încât serverul să poată reconecta socket-ul la jocul și utilizatorul corect.
- Operațiuni idempotente pe server*:* Operațiile pe partea serverului se bazează pe identificatori precum *roundId*, numerele de versiune și, acolo unde este necesar, ID-urile operațiunilor. Acest lucru face posibilă recunoașterea mesajelor repetate sau întârziate și ignorarea lor, în loc să le aplici de două ori, evitând recompensele duplicate sau starea inconsistentă atunci când mesajele sunt reîncercate.
- Blocaje la nivel de rundă pentru fazele critice: În fazele sensibile, cum ar fi calculele de la sfârșitul rundei, serverul folosește blocaje la nivel de rundă, astfel încât să ruleze un singur flux de calcul simultan pentru fiecare joc și rundă. Acest lucru previne condițiile de cursă atunci când multe acțiuni ajung aproape de finalul numărătoarei inverse.
- Gestionare grațioasă a desincronizării*:* Dacă un client a fost offline, în fundal sau suferă o scurtă deconectare, combinația dintre *GameContextHybridRobust* și serverul autoritar permite dispozitivului să se resincronizeze cu starea curentă când revine. În loc să se bazeze pe ceea ce a fost redat anterior, clientul își poate reîmprospăta vizualizarea rundei active, inventarelor și clasamentelor.

În practică, aceste mecanisme înseamnă că problemele de rețea pe termen scurt sau erorile tranzitorii nu ar trebui să invalideze o sesiune de joc. Antrenorii pot continua de obicei activitatea fără a necesita intervenție tehnică profundă, iar cursanții care își pierd temporar conexiunea pot reveni și pot continua să joace într-o stare constantă.

5.3. Întreținere și evoluție viitoare

Din perspectiva întreținerii, WP4. A5 stabilește câteva principii simple, dar importante, pentru a ghida evoluția viitoare a instrumentului:

- Păstrează logica în timp real centralizată pe serverul *autoritar:* Toate deciziile critice de timp (cine a minat primul, când se termină runda, cum sunt aplicate

recompensele) rămân pe server. Clientul consumă *authoritativeState* în loc să încerce să implementeze propriile reguli alternative. Acest lucru reduce riscul de divergență între diferitele versiuni ale clientului.

- Tratați Funcțiile Cloud ca punct de intrare unică pentru operațiunile backend: Orice operație nouă care modifică datele persistente ar trebui implementată ca o Funcție Cloud, înregistrată în funcții/index.js și întotdeauna apelată prin wrappers autentificați pe client. Acest lucru menține verificările de securitate și logica de business pe server și facilitează raționamentul despre fluxurile de date.
- Modificări și migrări ale schemelor documentelor: Când schemele documentelor Firestore evoluează (de exemplu, adăugarea unor câmpuri noi la *games/{gameId}* sau *users/{userId}.games[gameId]*), aceste modificări trebuie documentate explicit. Dacă datele existente trebuie ajustate, scripturile de migrare sau utilitarele pot fi plasate sub *funcții/src* astfel încât partenerii să aibă o cale clară de a actualiza datele de producție.

Urmând aceste practici, partenerii tehnici pot adăuga funcții noi — cum ar fi ecrane suplimentare, indicatori suplimentari, logare extinsă sau moduri noi de joc — fără a compromite stabilitatea jocului existent. RockChain rămâne întreținibil și extensibil după durata inițială de viață a proiectului, păstrând totodată comportamentul de bază validat în timpul piloților în WP5.

6. CONCLUZII

WP4. A5 a livrat o versiune pregătită pentru producție a instrumentului interactiv RockChain, consolidând munca de proiectare, implementare și implementare realizată în sarcinile anterioare ale WP4. Sistemul rezultat reunește un client mobil modern, multilingv, un backend bazat pe Firebase și un server autoritar în timp real pe Cloud Run, toate implementate pe infrastructură cloud scalabilă și ambalate în versiuni Android și iOS care pot fi instalate pe dispozitive standard folosite în VET și educația pentru adulți.

Prin documentarea explicită a arhitecturii sistemului, a fluxurilor de execuție, a fluxului de lucru de producție și implementare, precum și a procedurilor de găzduire, acces și mentenanță, această activitate asigură că RockChain nu este un prototip izolat, ci un activ reproductibil și ușor de întreținut. Personalul tehnic are o referință clară despre cum este structurat instrumentul și cum să-l actualizeze, în timp ce trainerii au o aplicație stabilă, instalabilă, care poate fi integrată în cursuri reale cu o configurare locală minimă.

Astfel, instrumentul interactiv RockChain devine nucleul practic al ofertei de e-learning a proiectului: el operaționalizează munca pedagogică și curriculară a WP-urilor anterioare într-o experiență concretă și jucabilă care poate fi folosită în sălile de clasă din țările partenere. Instrumentul este acum pregătit să susțină activitățile pilot în WP5 și oferă o bază solidă pentru potențiale exploatări și extinderi viitoare dincolo de durata proiectului.