WP4-A5. Proizvodnja interaktivnog RockChain alata.

Sadržaj

1. UVOD

Ovaj dokument prikazuje rezultate aktivnosti WP4. A5 – Proizvodnja interaktivnog alata RockChain. Dok su prethodni zadaci u WP4 bili usmjereni na podatkovni sloj (WP4. A1), usavršavanje alata za e-učenje (WP4. A2) i funkcionalnim specifikacijama (WP4. A3), WP4. A5 opisuje kako je konačna interaktivna verzija RockChaina proizvedena, pakirana i pripremljena za distribuciju i korištenje u stvarnim treninzima.

Cilj WP4. A5 je dvostruk. S jedne strane, pruža konsolidirani pregled tehničke arhitekture RockChain alata, pokazujući kako mobilni klijent, backend usluge i orkestracija u stvarnom vremenu surađuju kako bi podržali edukativne sesije za više igrača. S druge strane, dokumentira produkcijski i implementacijski tijek rada koji vodi do spremnih Android i iOS verzija, stabilne backend implementacije te osnovnih procedura za hosting, pristup i održavanje.

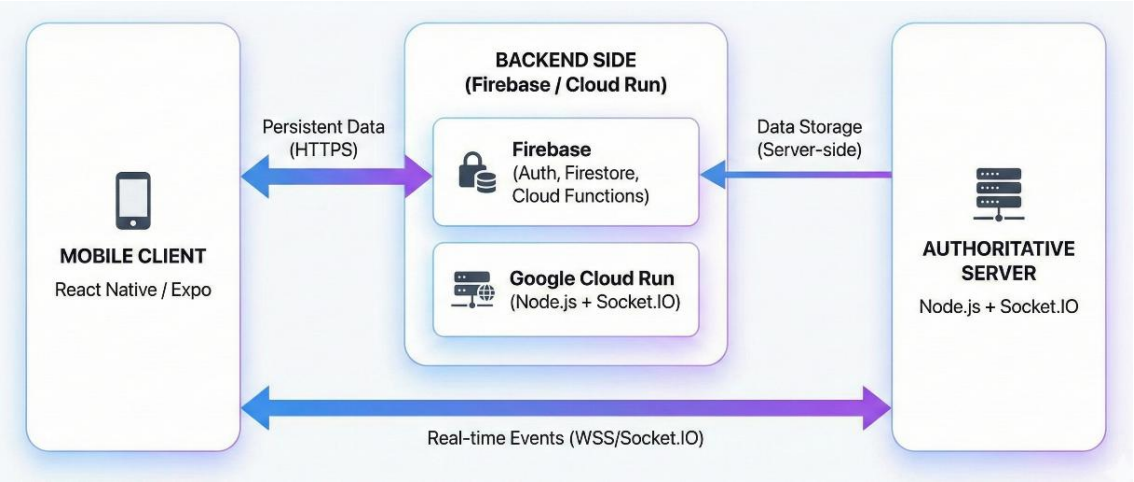
Rezultat je opis "kako je izrađen" koji mogu koristiti tehničko osoblje koje treba održavati ili proširiti alat, kao i projektni partneri kojima je potreban jasan pregled kako je interaktivni RockChain alat pretvoren u resurs spreman za proizvodnju za stručno i stručno obrazovanje odraslih.

2. ARHITEKTURA SUSTAVA INTERAKTIVNOG ROCKCHAIN ALATA

S tehničke strane, interaktivni RockChain alat izgrađen je kao troslojni sustav:

- Mobilni klijent razvijen s Expo-om i React Native-om.
- Firebase backend koji pruža autentifikaciju, trajnu pohranu i serverless funkcije.
- Autoritativni Node.js + Socket.IO server koji koordinira runde u stvarnom vremenu i osigurava da svi igrači vide dosljedno stanje igre.

Ove komponente nadopunjuju sloj internacionalizacije, alati za konfiguraciju okoliša i osnovni mehanizmi za promatranost.



Slika 1: Visoka razina arhitekture.

2.1. Mobilni klijent: okviri, ulazna točka i navigacija

RockChain mobilni klijent izgrađen je kao React aplikacija koja radi na React Nativeu, pakirana i orkestrirana od strane Expo-a. U praksi to znači:

- Svi ekrani i elementi sučelja napisani su kao React funkcije koristeći JSX.
- React 19 pruža model komponente, hookove (*useState, useEffect, useContext*, custom hooks) i Context API koji aplikacija koristi za dijeljenje stanja igre preko ekrana.
- React Native 0.79 prikazuje ove komponente kao nativne prikaze na Androidu i iOS-u, pa aplikacija djeluje i ponaša se kao nativna aplikacija dok dijeli jednu JavaScript bazu koda.
- Expo djeluje kao omotač i alatni lanac: pruža razvojni server, pakete, EAS build sustav i pristup izvornim uslugama (mreža, pohrana, informacije o uređaju) bez održavanja zasebnih Xcode/Android Studio projekata.

Tijekom izvođenja sve počinje u *App.js*, što je ulazna točka klijenta. Ova datoteka povezuje osnovne cross-cut servise koje svaki ekran treba:

- Uvodi internacionalizaciju omotavanjem cijelog stabla komponenti u *I18nextProvider i18n={i18n}.* To omogućuje bilo kojem ekranu korištenje prijevodnih ključeva umjesto tvrdo kodiranih nizova i automatsko odabir jezika korisnika.
- Sučelje obavija u *NavigationContainer* (iz React Navigation), koji definira jedinstveno navigacijsko stablo za cijelu aplikaciju (stogove, kartice i ugniježđene tokove).
- Navigaciju obuhvaća unutar dva globalna pružatelja, *GameProviderHybridRobust* i *SimpleMiningProvider,* te prikazuje *NetworkStatusBanner* na najvišoj razini:
    - *GameProviderHybridRobust* je React Context + hook sloj koji otkriva stanje povezano s igrom (trenutna igra i runda, igrači, inventari, tajmeri, rangiranje, navigacijske zaštite).
    - *SimpleMiningProvider* nudi posvećeni mini-kontekst za probleme rudarenja, olakšavajući pokretanje i rješavanje izazova dokazivanja rada bez njihovog čvrstog povezivanja s ostatkom korisničkog sučelja.
    - *NetworkStatusBanner* sluša promjene u povezivanju i prikazuje upozorenja kada je uređaj offline ili ima lošu povezanost.

Budući da se ti pružatelji nalaze iznad navigacijskog kontejnera, bilo koji ekran, bez obzira koliko duboko u snopu, može pristupiti stanju igre, stanju rudarenja i statusu mreže putem hookova, umjesto da tu logiku implementira lokalno.

Na temelju ove tehničke osnove, aplikacija slijedi jednostavan, linearan tijek navigacije koji odražava životni ciklus igre. React Navigation se koristi s glavnim slojem koji korisnike vodi kroz ove faze:

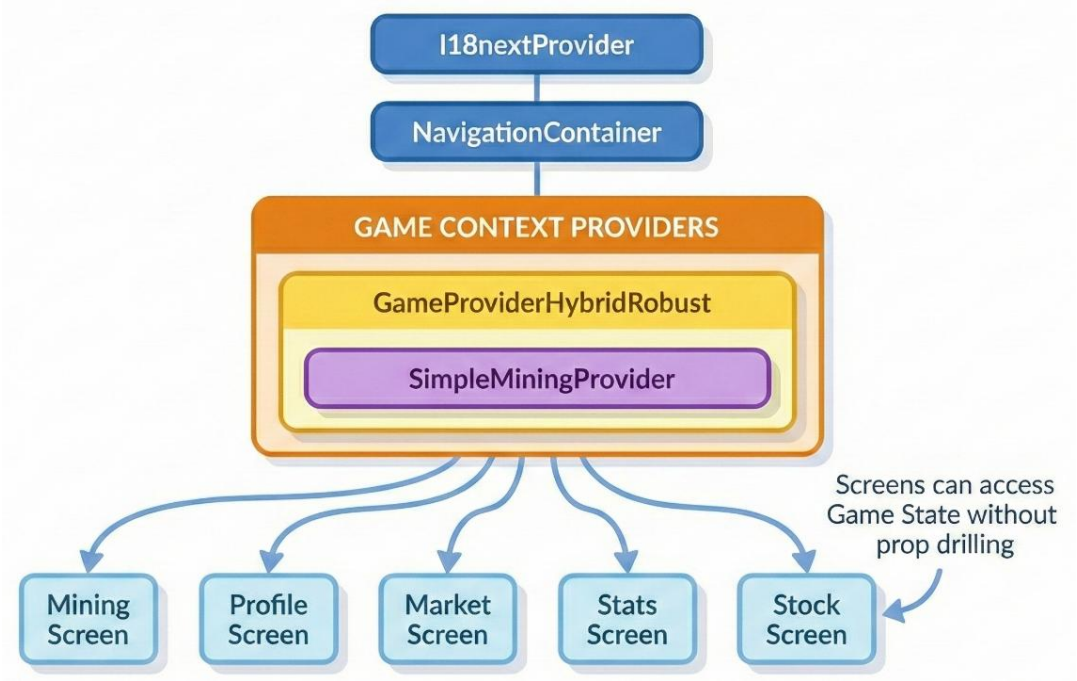
- *Prijava*: gdje se igrač registrira ili prijavi koristeći Firebase Auth.
- *Game:* omotni ekran koji postavlja kontekst za odabrani *gameId* i *userId*.
- *WaitingRoom:* mjesto gdje se igrači okupljaju i vide tko se pridružio igri prije početka rundi.
- *GameTabs:* glavno okruženje u igri koje se koristi tijekom rundi.
- *Ekrani na kraju runde i rezultati*: gdje se prikazuju sažeci, rangiranja i konačni ishodi.

Kada korisnik uđe *u GameTabs*, raspored se mijenja u strukturu temeljenu na karticama. Donja traka s karticama dio je stabla navigacije (standardni React Navigation navigator za kartice), ali može se vizualno sakriti kako bi iskustvo više nalikovalo koherentnoj igri nego tipičnoj aplikaciji s vidljivim karticama. Interno, slušatelji vezani uz Socket.IO događaje i *GameContextHybridRobust* odlučuju kada će se prebacivati između modula — na primjer, automatski otvaraju Mining kad se pojavi novi problem ili prelaze na

sažetak na kraju runde kada timer dosegne nulu — kako učenici ne bi morali sami rješavati složene prijelaze.

Tijekom aktivne runde, unutar GameTabsa dostupno je šest glavnih modula:

- *Tržište*: središnji modul u kojem igrači kupuju i prodaju kamene blokove, otpad i reciklirane proizvode. Pretplaćuje se na događaje ažuriranja proizvoda i cijena te prikazuje dinamičke liste temeljene na trenutnom stanju tržišta.
- *Rudarstvo*: fokusirani prikaz koji se pojavljuje kada se pokrene izazov dokazivanja rada. Prikazuje vremenski ograničene aritmetičke zadatke, lokalno prikuplja odgovore i šalje ih autoritativnom poslužitelju na validaciju.
- *Recycle*: modul u kojem se dijelovi inventara igrača mogu pretvoriti u RockCoins ili nove resurse, operacionalizirajući logiku kružne ekonomije u konkretnim akcijama i transakcijama.
- *Statistika*: nadzorna ploča implementirana s React Native komponentama plus react-native-chart-kitom, koja prikazuje pokazatelje uspješnosti i rangiranja po rundi i kroz cijelu igru.
- *Profil*: osobno područje koje prikazuje avatar igrača, trenutni napredak i jezične postavke, a može otkriti i druge osnovne postavke u budućim ekstenzijama.
- *Dionice*: pogledajte sažetke količina i cijena ključnih proizvoda, djelujući kao brzi "tržišni prikaz". Igrači ga mogu otvoriti kako bi nakratko razumjeli koji su resursi obilni, a koji rijetki, prije nego što odluče hoće li kupiti, prodati ili reciklirati.



Slika 2: stablo komponenti i pružatelja.

Svi ti ekrani su standardne React komponente koje primaju podatke prvenstveno iz *GameProviderHybridRobust* i Socket.IO event streama, a ne iz ad-hoc lokalne države. Ova kombinacija, React komponenti i hookovi, React Native renderiranje, Expo build/runtime, React navigacija za kontrolu toka i pružatelji konteksta za dijeljeno stanje, pretvara mobilni klijent u koherentan, deklarativan front end koji je relativno jednostavan za razmišljanje i proširenje, a pritom i dalje djeluje kao izvorna multiplayer igra na uređajima početnika.

2.2. Upravljanje državom na strani klijenta

Na klijentu, RockChain se uvelike oslanja na React Context API i prilagođene hookove za upravljanje dijeljenim stanjem. Umjesto da svaki ekran čuva vlastitu kopiju igrača, tajmera ili rudarskih podataka, aplikacija većinu logike centralizira u nekoliko jasno definiranih pružatelja. Ekrani tada postaju uglavnom "prikazi": oni prihvaćaju to stanje i prikazuju ga, ali sami ne odlučuju o pravilima. To je ono što aplikaciju održava predvidljivom čak i kada mnogi događaji stignu u stvarnom vremenu.

U središtu ovog sloja nalazi *se GameContextHybridRobust*. Tehnički, to je React kontekst podržan komponentom pružatelja usluga i skupom hookova koji izlažu trenutačno stanje igre bilo kojem ekranu. Konceptualno, to je glavni izvor istine o klijentu. Drži se na okupu:

- Popis igrača u trenutnoj igri i njihovi inventari (proizvodi, otpad, RockCoins).
- Odbrojavanje i identifikatori trenutne runde, tako da svi ekrani znaju koliko je vremena ostalo i koja je runda aktivna.
- Stanje rudarenja i rangiranje, tako da rezultati rudarenja i ljestvice su dosljedni kroz Tržište, Statistiku i druge prikaze.
- Sinkronizirani prikaz serverskog sata, izgrađen od Socket.IO poruka, koji klijentu omogućuje da izvede preostalo vrijeme bez prevelikog udaljavanja od autoritativnog poslužitelja.
- Nekoliko "sigurnih navigacijskih" zaštitnih mehanizama koji sprječavaju uvjete utrke kada se navigacija mijenja istovremeno s novim događajima (na primjer, sprječavanje da ekran pročita zastarjele podatke baš kad runda završi).

Osim samog pohranjivanja podataka, *GameContextHybridRobust* također uključuje skup mehanizama idempotencije i resinkronizacije oko WebSocket veze. Svaki relevantni događaj nosi identifikatore poput *roundID-a* i verzije, a kontekst prati što je već primijenjeno. Ako se veza prekine i kasnije oporavi, kontekst može ponovno zatražiti ili uskladiti autoritativno stanje od poslužitelja umjesto da slijepo vjeruje onome što je posljednje renderirano. To omogućuje igraču da na trenutak zaključa telefon ili nakratko izgubi Wi-Fi bez potpunog kvarenja igre.

Logika rudarenja dodatno je enkapsulirana u namjenskom kontekstu, često nazvanom *SimpleMiningContext* i izložena putem *SimpleMiningProvidera*. Ideja je držati lagani red problema s rudarenjem i njihovo upravljanje korisničkim sučeljem odvojenim od ostatka stanja igre. Kada autoritativni poslužitelj emitira nove izazove rudarenja, oni se ubacuju u ovaj red; Kada korisnik odgovori ili istekne tajmer, oni se obrađuju i uklanjaju. Budući da se rudarenje odvija u ovom izoliranom kontekstu, ostatak korisničkog sučelja može ostati responzivan čak i ako se nekoliko problema stvori i riješi u brzom slijedu, a korisničko sučelje vezano uz rudarenje (poput modala ili odbrojavanja) ne mora biti ručno povezano sa svim ekranima.

Oko ova dva glavna konteksta, klijent koristi skup pomoćnih hookova za grupiranje specifičnih poslovnih pravila:

- *useLearningProgress* sluša događaje u igri i bilježi obrazovna postignuća (npr. iskopani blokovi, smanjenje otpada) u Firestore i može pokrenuti vizualne povratne informacije kada se dosegnu određene prekretnice.
- *useNetworkStatus* obavija NetInfo API i prenosi *NetworkStatusBanner*, tako da je svaki privremeni gubitak veze odmah vidljiv igraču.
- *useTutorial* kontrolira onboarding tokove za početnike, odlučujući kada prikazati objašnjenja ili savjete, a kada se povući i pustiti igrača da slobodno komunicira.

Zajedno, ti konteksti i udice implementiraju jasno načelo dizajna: ekrani trebaju biti što jednostavniji, dok se međusobno ponašanje (tajmeri, igrači, rudarenje, napredak u učenju, povezanost, tutorijali) odvija u zajedničkim, testabilnim modulima. To aplikaciju čini lakšom za razmišljanje, lakšom za proširenje (novi ekrani mogu ponovno koristiti iste hookove) i robusnijom u uvjetima stvarnog vremena, jer postoji jedno mjesto gdje se stanje igre izvodi iz backend događaja.

2.3. Firebase backend: autentifikacija i trajni podaci

U pozadini, RockChain se oslanja na Firebase kao upravljanu platformu koja integrira tri ključne usluge: autentifikaciju, Cloud Firestore i Cloud funkcije. Zajedno, ove usluge omogućuju sigurno upravljanje identitetom svakog korisnika, trajno pohranjivanje podataka o igri i napredak u učenju te kontrolirano izvršavanje osjetljivih operacija izravno na poslužitelju. Iako su svi tehnički detalji o modelu podataka, tokovima i sigurnosnim pravilima dokumentirani u isporučivom WP4-A1, ovdje je pregled kako se ova infrastruktura koristi unutar interaktivnog alata.

Srce sustava je Cloud Firestore, koji djeluje kao RockChainova "dugoročna memorija." Umjesto tablica kao u tradicionalnoj bazi podataka, Firestore organizira informacije u zbirke i dokumente. U produkcijskom okruženju, glavne zbirke su:

- *users*: sadrži jedan dokument po igraču, s njihovim osnovnim profilom i izvedbom u svakoj igri (rockCoins, otpad, proizvodi), grupiranim prema identifikatoru igre.
- *Igre*: Jedan dokument po sesiji igre, gdje se pohranjuju pristupni kod, domaćin, popis igrača, trenutni status, runda i određeni ključni indikatori koje koriste i klijent i poslužitelj u stvarnom vremenu.
- *Tržište*: Katalozi proizvoda i dinamičke cijene po igri, odvojeni od glavnog dokumenta kako ga ne bi preopteretili čestim ažuriranjima.
- *Blockchain*: Pojednostavljena povijest blokova iskopanih u svakoj igri, bilježeći tko je što i kada rudario, omogućujući da se mehanike inspirirane blockchainom odražavaju u podacima.

Za kritične operacije, klijenti ne pišu izravno u te kolekcije. Umjesto toga, pozivaju serverske funkcije (Cloud Functions) koje rade s posebnim privilegijama, primjenjuju validacije i poslovna pravila, a tek tada mijenjaju podatke u Firestoreu. Te su funkcije detaljno opisane u izvješću WP4-A1, ali unutar aplikacije imaju tri glavne uloge:

- Upravljanje životnim ciklusom igre: funkcije poput *createGame*, *joinGame* i *deleteGame* stvaraju i brišu dokumente igre, provjeravaju jedinstvenost koda i sprječavaju zastarjele sesije s istog hosta.
- Logika tržišta i rudarenja: Funkcije poput *assignProductsToGame*, *updateProductPrices*, *generateMiningProblem* i *submitMiningSolution* kontroliraju razvoj proizvoda, cijena i rudarskih izazova, osiguravajući dosljedna pravila kroz sve igre.
- Nagrade i profili: Na kraju svake runde ili igre, funkcije poput *assignRoundRewards* i *updateUserProfile* konsolidiraju rezultate te ažuriraju korisničke profile i snapshot dokumente, održavajući dosljednost između onoga što je prikazano na ekranu i onoga što je trajno spremljeno.

Osim toga, integracija s Firebaseom uključuje sustav autentifikacije i mali sloj lokalne pohrane na uređaju (*AsyncStorage*). Autentifikacija osigurava da je svaki zahtjev prema backendu povezan s jedinstvenim korisničkim ID-jem, koji se dosljedno koristi u Firestore i na WebSocket poslužitelju. Lokalna pohrana omogućuje aplikaciji pamćenje aktivnih sesija i spremanje bitnih podataka između lansiranja, izbjegavajući prekide u slučaju kratkih prekida veze ili slučajnog zatvaranja aplikacija.

Za detalje o potpunoj implementaciji, uključujući sigurnosna pravila, tokove podataka i aspekte povezane s GDPR-om, partneri mogu pogledati tehničko izvješće WP4-A1.

2.4. Autoritativni poslužitelj i usluge u stvarnom vremenu

Iako Firebase sigurno i uporno upravlja pohranom i logikom servera, RockChain također treba sloj koji gotovo trenutno reagira na akcije igrača, upravlja timerima i u stvarnom

vremenu odlučuje tko pobjeđuje u rudarskim izazovima. Kako bi zadovoljio tu potrebu, projekt koristi poslužitelj u stvarnom vremenu razvijen s Node.js i Socket.IO, zapakiran u Docker sliku i implementiran na Google Cloud Run. WP4-A1 detaljno opisuje ovu arhitekturu, ali ovdje je sažetak kako podržava interaktivni alat.

Jednostavno rečeno, ovaj server djeluje kao sudac za svaku utakmicu u tijeku. Za svaku utakmicu pamće:

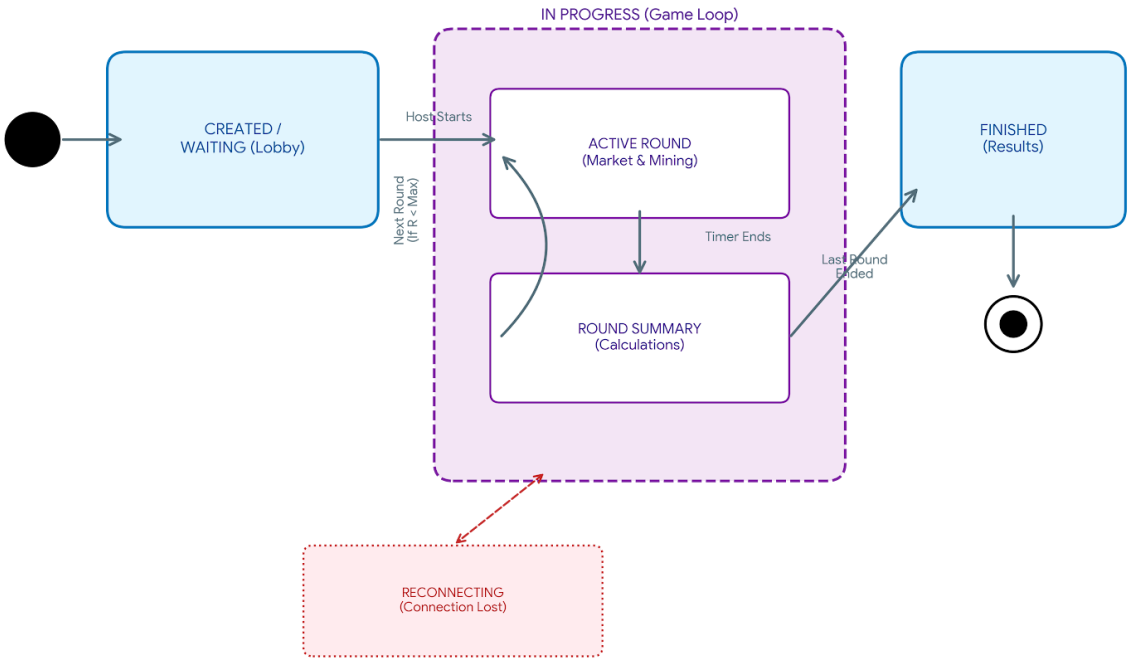
- Koji su igrači povezani i u kojoj sobi.
- Njihove trenutne zalihe i pozicije u rangiranju.
- Izazov aktivnog rudarenja (ako ga ima).
- Službeni status runde, uključujući točno vrijeme kada bi trebala završiti.

Na temelju tog statusa, poslužitelj generira identifikatore poput *roundId*, *verzije* i *roundEndsAtM-a*, koji se pridružuju poslanim događajima. To omogućuje i klijentu i backendu da u svakom trenutku znaju na koju rundu misle i izbjegava zabunu s dupliciranim ili zakašnjelim porukama.

Ponašanje igre strukturirano je kao jednostavan automat stanja s četiri glavne faze:

- IGRANJE: runda je aktivna; Igrači mogu kupovati, reciklirati ili rješavati probleme rudarenja dok vrijeme istječe.
- ROUND_CALCULATING: radnje se pauziraju, a poslužitelj izračunava rezultate.
- ROUND_END: konačne vrijednosti su sada dostupne; Igrači mogu vidjeti svoje nagrade i pozicije.
- WAITING_FOR_NEXT_ROUND: igra se pauzira dok voditelj ne odluči započeti novu rundu ili završiti igru.

Kako igra napreduje kroz te faze, server emitira različite događaje poput *start_round, ROUND_CALCULATING, round_ended, game_completed, inventory_data, player_rankings_data* ili *economy:state*. Mobilna aplikacija sluša te događaje i ažurira ekrane, tajmere i sažetke u stvarnom vremenu, osiguravajući da svi igrači imaju dosljedan pregled onoga što se događa.



Slika 3: Rockchain automat stanja.

Budući da mnoge akcije mogu stići gotovo istovremeno (posebno na kraju runde), server koristi mehanizme zaključavanja i idempotentne nadogradnje kako bi izbjegao sukobe. Kritične operacije, poput dodjele nagrada ili ažuriranja inventara, izvode se uredno, bez preklapanja, i označene su okruglim identifikatorima kako bi se zanemarila ponavljanja. Kada server može pristupiti Firebaseu, također sprema rezultate svake runde (inventare, nagrade, ekonomski status), osiguravajući da su snimka u memoriji i trajno pohranjeni podaci usklađeni.

Na klijentskoj strani, sva ta složenost je enkapsulirana u Socket.IO adapteru. Ovaj adapter automatski odabire ispravan poslužitelj (lokalni ili Cloud Run), otvara i održava WebSocket vezu, prosljeđuje informacije o autentifikaciji i igri nakon ponovnog povezivanja te prevodi backend događaje u format koji se očekuje u aplikaciji koju razvija React. Zahvaljujući ovom srednjem sloju, server se može razvijati tijekom vremena bez utjecaja na korisničko iskustvo ili pedagoško funkcioniranje alata

2.5. Međusektorska pitanja: internacionalizacija, konfiguracija i uočljivost

Osim glavnih klijentskih i backend komponenti, RockChain uključuje nekoliko mehanizama za presijecanje koji alat čine upotrebljivim u različitim zemljama, lakšim za implementaciju u različitim okruženjima i lakšim za otklanjanje grešaka tijekom pilot-

projekta. Tri od njih su posebno relevantna: internacionalizacija, konfiguracija okoliša i promatranost.

2.5.1. Internacionalizacija i dostupnost

Od samog početka, RockChain je dizajniran kao alat za više zemalja. To je značilo da jezična podrška ne može biti naknadna misao. Umjesto toga, projekt je integrirao sloj internacionalizacije temeljen na i18nextu.

Osnovna postavka nalazi se u *src/i18n/index.js*, što:

- Učitava jezične pakete za engleski (EN), španjolski (ES), njemački (DE), hrvatski (HR) i rumunjski (RO).
- Detektira jezik uređaja ili koristi eksplicitni izbor igrača da odluči koji paket aktivirati.
- Izlaže i18n instancu koju ostatak aplikacije koristi putem I18nextProvidera.

Svi tekstovi namijenjeni korisnicima pohranjuju se kao prijevodni ključevi u *src/i18n/locales/*.json*. Za svaki podržani jezik postoji odgovarajuća JSON datoteka u kojoj su ti ključevi mapirani na stvarne stringove. Komponente tada traže tekstove po tipkama, a ne po tvrdom kodiranju engleskog ili bilo kojeg drugog jezika.

Kako bi iskustvo bilo trajno, odabrani jezik se sprema u AsyncStorage. To znači da:

- Prvi put kad se aplikacija pokrene, može automatski koristiti jezik uređaja.
- Ako korisnik promijeni jezik putem sučelja, ta se preferencija sprema lokalno.
- Pri sljedećim lansiranjima, aplikacija vraća isti jezik bez ponovnog traženja.

Dodavanje novog jezika je jednostavno i ne zahtijeva diranje osnovnog koda:

1. Kreirajte novu JSON datoteku pod *src/i18n/locales/* s istim ključevima kao postojeći jezici.
2. Registrirajte novi jezični kod u *supportedLngs* u *src/i18n/index.js*.
3. Osigurajte prijevode za sve relevantne tonove.

Ovaj dizajn podržava jedan od temeljnih ciljeva RockChaina: alat se može prenijeti u dodatne zemlje i kontekste radom na prijevodnim datotekama umjesto mijenjanja logike aplikacije.

2.5.2. Konfiguracija okruženja i detekcija mreže

Budući da RockChain mora raditi u nekoliko konteksta izvršavanja (lokalni razvoj, staging i produkcija), projekt izbjegava hardkodiranje URL-ova ili pretpostavke o okruženju. Umjesto toga, koristi mali, ali eksplicitan konfiguracijski sloj.

Dva modula su ovdje u središtu:

src/config/environment.js

- Izračunava vrijednosti poput SOCKET_URL, NODE_ENV, timeouta i debug zastavica.
- Čita iz postavki vremena izgradnje i varijabli okruženja kako bi, na primjer, odlučio treba li aplikacija komunicirati s lokalnim Socket.IO serverom, instancom za pripremu ili produkcijskom Cloud Run uslugom.

src/utils/networkUtils.js

- Detektira radi li aplikacija u simulatoru/emulatoru ili na fizičkom uređaju.
- Na temelju tih informacija, odabire odgovarajuću adresu poslužitelja:
    - Localhost ili određeni LAN IP tijekom razvoja.
    - Cloud Run URL u staging ili produkcijskim verzijama.

Osim toga, Expo varijable okruženja (poput EXPO_PUBLIC_WEBSOCKET_URL) omogućuju projektu da nadjača krajnje točke u određenim build profilima. Na primjer, produkcijska EAS verzija može žično povezati WebSocket endpoint na službeni Cloud Run URL, dok preview verzija može ukazivati na testnu instancu.

Konačni učinak je da se ista baza koda može ponovno izgraditi za različita okruženja jednostavnim odabirom pravog profila izgradnje i konfiguracije, bez uređivanja izvornih datoteka. To izravno podržava ponovljivost i olakšava partnerima kloniranje ili ažuriranje implementacije.

2.5.3. Uočljivost i otpornost

Na kraju, arhitektura uključuje minimalni skup alata za razumijevanje što se događa u stvarnom vremenu i za preživljavanje privremenih mrežnih problema. To je posebno važno u učionici, gdje kvaliteta Wi-Fi-ja može varirati.

Na strani klijenta:

- *GameContextHybridRobust* bilježi informacije o pomaku sata, dupliciranim događajima i pokušajima ponovne sinkronizacije na konzolu. Tijekom razvoja i pilot testiranja, ti zapisi pomažu identificirati situacije u kojima se klijent i autoritativni poslužitelj privremeno razilaze i koliko često su potrebne resinkronizacije.
- Kombinacija automatskog ponovnog povezivanja socketa i React Contexta osigurava da, kada se veza ponovno uspostavi, klijent može ponovno zatražiti trenutno autoritativno stanje umjesto da se oslanja na zastarjele podatke.

Na strani servera:

- Alat logMetric *emitira strukturirane JSON događaje za ključne trenutke poput veze,* round_start, round_end, resinkronizacije i pogreške. Kada server radi na

Cloud Runu, ti se događaji mogu zabilježiti Google Cloud Logging ili sličnim alatima, pružajući vremensku crtu događaja u svakoj sesiji igre.

- Server provodi obradu idempotentnih događaja koristeći identifikatore poput *roundId*, verzija i, gdje je relevantno, *operationId*. To sprječava duplicirane nagrade ili nedosljedno stanje kada se poruke ponovno pokušavaju ili stignu kasno.
- Zaključavanja po rundi (*roundLocks*, *gameStates*) osiguravaju da se kritični dijelovi — poput izračuna na kraju runde — ne izvršavaju istovremeno za istu igru i rundu.

Zajedno, ovi mehanizmi znače da je arhitektura ne samo funkcionalna za pilote, već i pratljiva i robusna. Programeri i tehnički potkovani partneri mogu pregledavati zapise kako bi razumjeli kako su se sesije razvijale, dok igrači doživljavaju igru koja nastavlja raditi čak i kada se pojave kratkotrajni problemi s povezivanjem.

3. KLJUČNI TOKOVI IZVOĐENJA U INTERAKTIVNOM ROCKCHAIN ALATU

Iako prethodni odjeljak objašnjava kako je izgrađena arhitektura RockChaina, ovdje se fokusiramo na ono što osoba doživljava prilikom korištenja alata: kako se prijavljuje, kako se pridružuje ili stvara igru, kako napreduju runde i kakve odgovore sustav nudi. Ukratko, promatramo proces korak po korak iz perspektive korisnika.

3.1. Autentifikacija i onboarding

Sve počinje na ekranu za prijavu (*LoginScreen*), gdje korisnici mogu učiniti dvije stvari:

- Kreirajte novi račun unosom minimalnih potrebnih podataka (kao što su email adresa, lozinka i ime koje će biti vidljivo u igri).
- Prijavite se s postojećim računom, ponovno koristeći njihove spremljene podatke za prijavu.

Ovaj zaslon je izravno povezan s Firebase sustavom autentifikacije. Kada netko unese svoje podatke i klikne "Enter", aplikacija radi sljedeće:

1. On šalje vjerodajnice Firebaseu na provjeru.
2. Ako je sve točno, Firebase vraća autentificirani korisnički ID i pristupni token.
3. Ako dođe do pogreške (npr. pogrešna lozinka ili već registrirana e-pošta), aplikacija prikazuje jasnu poruku i traži ispravak podataka.

Što se događa nakon uspješne prijave?

Tri se stvari događaju automatski i gotovo istovremeno:

Profil se stvara (ili dohvaća) u bazi podataka.

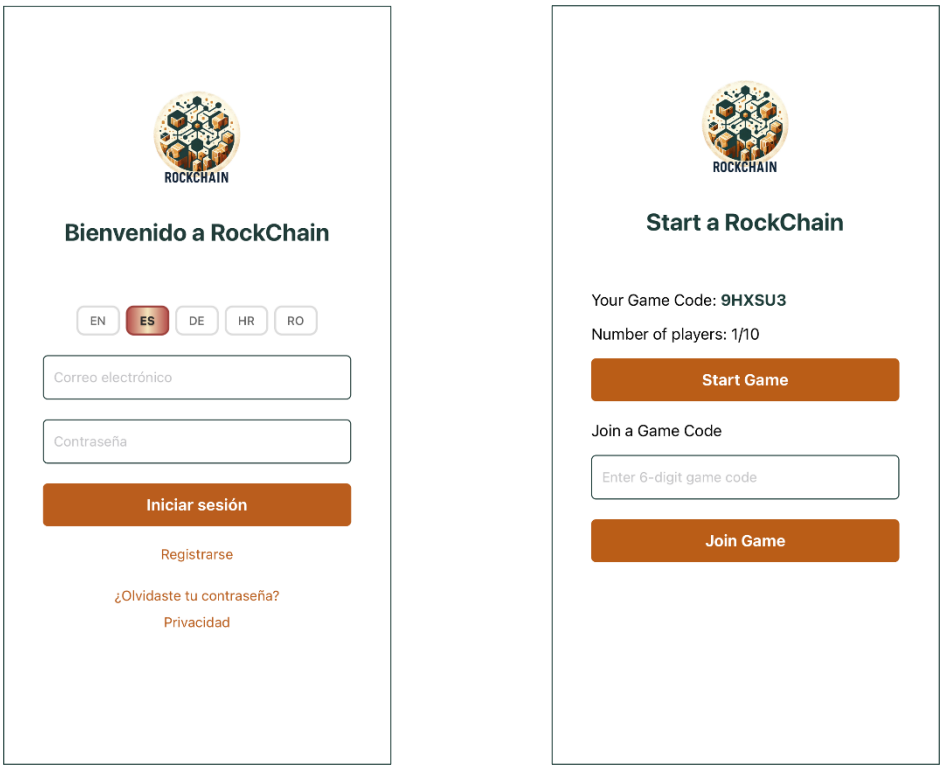
- Ako je račun nov, aplikacija ili funkcija na poslužitelju stvara dokument u bazi podataka s osnovnim informacijama: vidljivim imenom, e-mailom, datumom kreiranja, zadanim jezikom itd.
- Ako je profil već postojao, ponovno se koristi, zajedno s poviješću prethodnih igara, ako ih ima.

Za akcije u igri dobiva se siguran žeton.

- Ovaj token omogućuje aplikaciji komunikaciju sa serverom i pristup zaštićenim značajkama, poput stvaranja ili pridruživanja igrama.
- Također se koristi za povezivanje sa serverom u stvarnom vremenu, tako da sustav zna tko stoji iza svake radnje.

Pristupite glavnom području igre.

- Aplikacija prestaje prikazivati zaslon za prijavu i ide izravno u područje igre, gdje osoba može:
- Pokreni novu igru.
- Pridružite se postojećoj igri unosom koda.
- Pogledajte osnovne informacije o RockChainu.
- Ili pogledajte njihova prethodna postignuća.



Slika 4: LOGIN i ekrani IGRE.

RockChain ne razlikuje "normalne račune" od "host računa". Svatko tko je prijavljen može kreirati igru i postati domaćin ako podijeli svoj kod i drugi igrači se pridruže njihovoj sesiji. To čini proces istim za sve: svaka osoba pristupa svom prostoru za igru, a zatim grupe prirodno odlučuju koji će koristiti kao zajedničku igru.

3.2. Kreiranje igre i pridruživanje od strane igrača

Životni ciklus igre na RockChainu osmišljen je da bude što jednostavniji i besprijekorniji. Nakon što se osoba prijavi, aplikacija je tretira kao i sve ostale: automatski joj se dodjeljuje vlastita igra koju hosta, a zatim može odlučiti hoće li je koristiti kao sesiju za svoju grupu ili se pridružiti nečijoj drugoj koristeći kod.

Nema potrebe odlučivati između "kreiranja" ili "hostinga": ako dijelite svoj kod i pokrenete igru, vi ste domaćin. Tako je jednostavno.

3.2.1. Automatski host igra na *GameScreenu*

Odmah nakon uspješne prijave, aplikacija vodi osobu na glavni ekran igre. U tom trenutku aplikacija automatski stvara osobnu igru u pozadini: nema potrebe pritiskati gumb "create".

U pozadini, ovaj proces se brine o:

- Provjeravam ima li ta osoba već aktivne igre i, ako da, zatvaram ih kako bi se izbjegle duplikate.
- Osiguravajući dostupnost globalnog kataloga proizvoda i osnovnih tržišnih podataka.
- Kreiranje novog dokumenta igre s:
  - Jedinstveni kod za dijeljenje (kao što je *ABC123*)
  - Korisnički ID kao domaćin
  - Početni status "*čeka"*
  - Brojač rundi postavljen na nulu
  - I sve zadane postavke koje bi mogle biti potrebne

Kada je kreiranje završeno, aplikacija ima sve što joj treba za početak. Na ekranu će osoba vidjeti nešto poput.

- "*Vaš kod igre: ABC123*" na vrhu.
- "*Povezani igrači: 1*".
- Gumb "*Start game*".
- Polje za unos koda ako želi pristupiti nekoj drugoj sesiji.

Ako odlučite koristiti svoju igru kao grupnu sesiju, jednostavno:

- Podijelite svoj kod sa suigračima (naglas, na ekranu, putem chata itd.).
- Pričekaj da se svi povežu (brojač igrača to pokazuje).
- Pritisni "Start game."
- Nije potrebna daljnja radnja: ta osobna igra postaje službena grupna utakmica.

3.2.2. Pridruživanje tuđoj igri

Suprotno se također može dogoditi kad dođete do tog ekrana, možda već imate svoju igru napravljenu, ali želite se pridružiti drugoj igri koju je prijatelj već započeo.

Zato služi područje "Pridruži se igri". Sve što trebate učiniti je:

- Unesite kod sesije koji dijeli host.
- Pritisni gumb "*Pridruži se igri*".

Aplikacija će:

- Potraži tu igru koristeći kod.
- Provjerite je li još uvijek otvoren za nove igrače.
- Dodajte korisnika na popis sudionika.
- Kreirajte ili ažurirajte njihove osobne podatke o igri (*novčići, inventar, otpad itd*.).
- Promijenite kontekst njihove igre: od tog trenutka postaju dio domaćinove sesije.

Od tog trenutka osoba vidi kod domaćina i točan broj povezanih igrača na ekranu. Kada je grupa kompletna, voditelj može pritisnuti "*Start game*", što će sve odvesti u čekaonicu i započeti igru.

3.3. Čekaonica i sinkronizacija

Kada osoba odluči koristiti vlastitu igru kao domaćina ili se pridruži tuđoj, svi sudionici se prebacuju s glavnog ekrana igre u čekaonicu. Ovaj ekran ima vrlo specifičnu ulogu: okupiti grupu na stabilnoj početnoj točki i osigurati da, kada odbrojavanje počne, svi uređaji budu savršeno sinkronizirani.

Lobi ne upravlja svojom državom zasebno. Umjesto toga, sluša dva ključna izvora informacija:

- *GameContextHybridRobust*, koji prikuplja:
  - Ažuriranja u dokumentu igre (*games*/{*gameId*}), poput dolaska novih igrača ili promjena statusa
  - Trenutni popis igrača i opći status igre (čekanje, u igri, završeno itd.)
- Događaji servera u stvarnom vremenu (*Socket.IO*), koji označavaju:
  - Početak odbrojavanja,
  - Promjene faze igre (npr. prelazak s "*čekanja*" na "*igranje*")

S tim informacijama, predvorje prikazuje samo osnovne stvari:

- Popis igrača povezanih s tom igrom,
- Trenutni status igre (poruke poput "*Čekamo igrače*", "*Spremni za početak*", "*Počinje za 3...2...1...*"),
- Opcionalni indikatori da su igrači spremni, ako je ta značajka uključena

Što se događa kada domaćin pritisne "*Start game*"?

Kad voditelj procijeni da je grupa spremna i pritisne "*Start igre*":

1. Vaš uređaj poziva funkciju *startGame* u pozadini i šalje ID igre.
2. Server provjerava da:
   - Igra je u valjanom stanju (npr. nije već u tijeku i ima dovoljno igrača)
   - Koordinirajte s real-time serverom za pripremu za početak prvog poteza:

- Generira se jedinstveni identifikator runde (*roundId*)
- Izračunava se točno vrijeme kada će runda završiti (*roundEndsAtMs*)
- Igra se označava kao spremna za prelazak u fazu igre nakon odbrojavanja

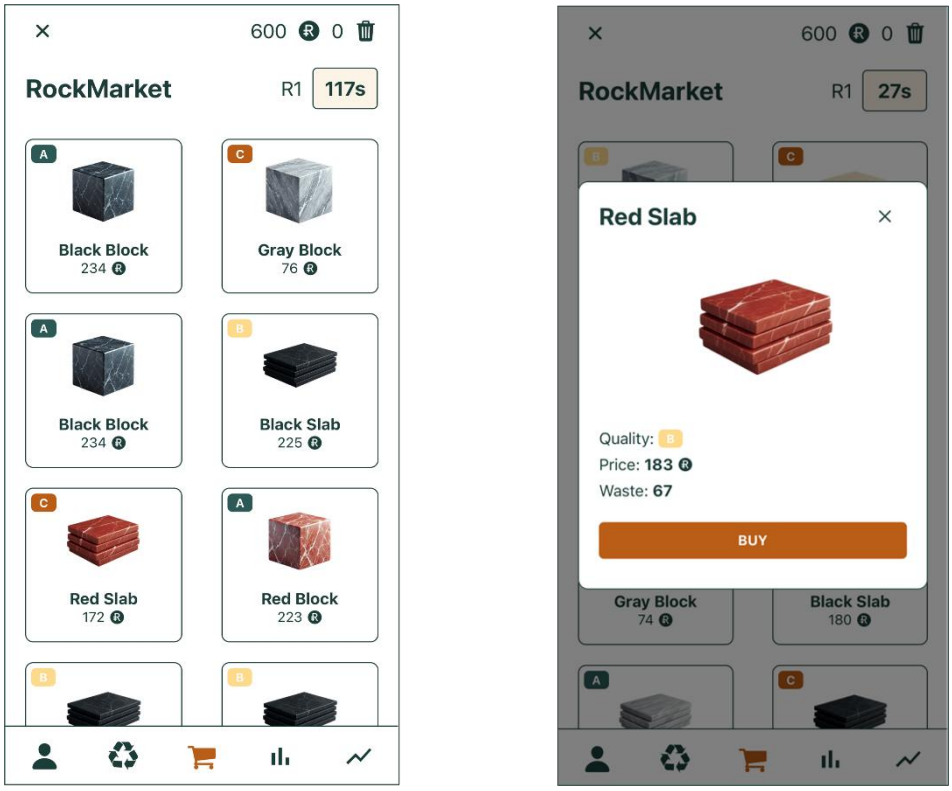
Poslužitelj zatim šalje niz događaja putem *Socketa.IO*:

- Događaj s odbrojavanjem ("*runda počinje za nekoliko sekundi*")
- Opcionalni događaji označavaju napredak odbrojavanja ("*3... 2... 1...*")
- Događaj koji označava da je runda službeno započela

Svaka čekaonica prima te događaje, a *GameContextHybridRobust* koristi vremensku oznaku (*roundEndsAtM)* za izračun preostalog vremena na svakom uređaju. Čak i ako postoje male razlike u Wi-Fi vezi, svi igrači napuštaju čekaonicu i ulaze u rundu s istim vremenskim ograničenjem

Čekaonica djeluje kao komora za sinkronizaciju. Osigurava da:

- Svi su povezani i vidljivi u igri.
- Voditelj može odlučiti točan trenutak za početak.
- Cijela grupa započinje rundu u isto vrijeme, zahvaljujući kombinaciji cloud značajki i komunikacije u stvarnom vremenu.



Slika 5: TRŽIŠNI ekran.

3.4. Igranje tijekom runde: navigacija između ekrana

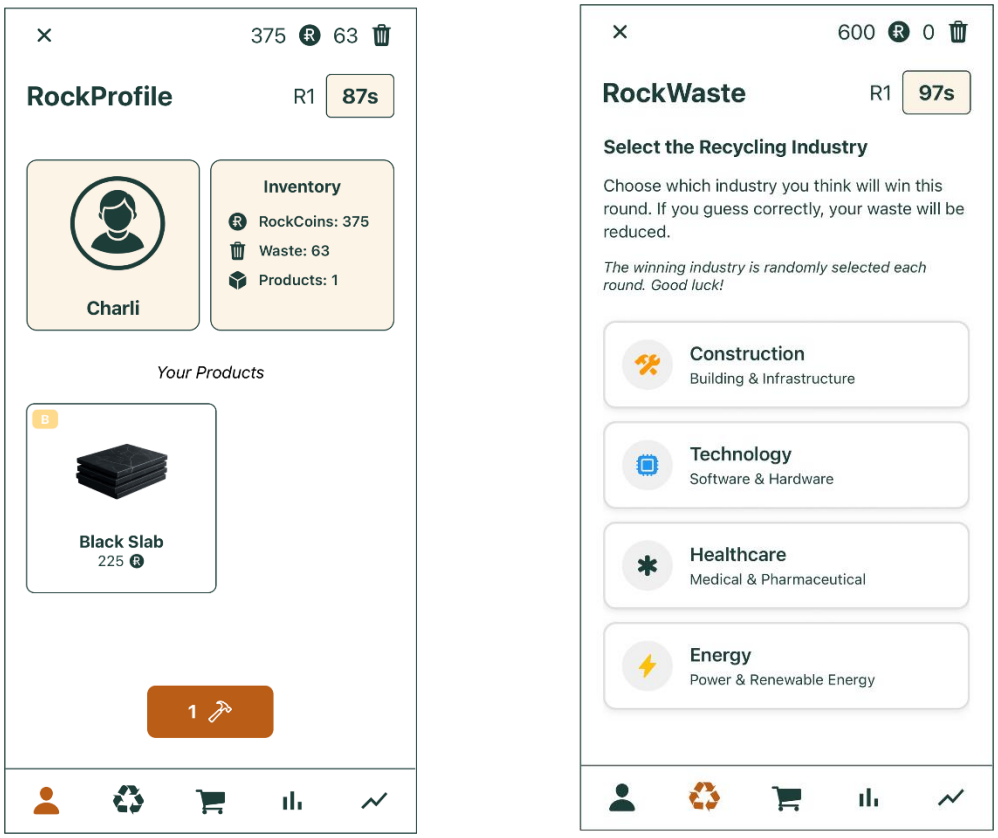
Kada runda započne, aplikacija automatski premješta sve igrače iz lobija u glavno sučelje igre, nazvano ***GameTabs***. Od tog trenutka svaka osoba može slobodno prelaziti između nekoliko ključnih ekrana tijekom cijele runde.

Vrijeme igre kontrolira središnja vrijednost zvana *roundEndsAtMs*, koja točno definira kada runda treba završiti. Ovu vrijednost server daje u stvarnom vremenu, a kontekst igre (*GameContextHybridRobust*) pretvara je u lokalno odbrojavanje tako da svi uređaji imaju isto vremensko ograničenje, bez obzira na manja kašnjenja u vezi.

Tijekom tipične runde od 120 sekundi, igrači mogu komunicirati sa sljedećim modulima:

- *MarketScreen*: omogućuje igračima kupnju i prodaju proizvoda (*blokove, ploče, reciklirane materijale*). Popis proizvoda se automatski ažurira ako server promijeni cijene ili ponudu.
- *RecycleScreen*: ovdje možete dio svog inventara pretvoriti u RockCoins ili nove materijale. Svaka radnja se potvrđuje u backendu, koji provjerava poštuju li se pravila prije primjene.

- *MiningScreen*: nudi izazove "proof-of-work". Aplikacija prikazuje matematičke zadatke s vremenskim ograničenjem, igrač predaje svoj odgovor, a server odlučuje tko je bio točan, a tko najbrži.
- *StatsScreen*: prikazuje pokazatelje uspješnosti i rangiranja s grafikonima koji se ažuriraju prema napretku.
- *ProfileScreen*: pruža sažetak korisničkog profila, trenutnih resursa i RockCoinsa. Također vam omogućuje da promijenite jezik. Informacije dolaze izravno iz Firestorea.



Slika 6: PROFILNI i RECIKLIRAJ ekrani.

- *StockScreen*: pruža pregled tržišta: koji su proizvodi dostupni, kako se cijene razlikuju i koji su resursi rijetki ili obilni. Koristan je za planiranje prije kupnje ili recikliranja.

Jedno osnovno načelo ostaje dosljedno na svim ovim ekranima: aplikacija nikada sama ne donosi konačne odluke. Svaka radnja koju igrač izvrši tumači se kao namjera poslana serveru. Na primjer:

- *"Želim kupiti X proizvod po trenutnoj cijeni"*
- *"Želim reciklirati ovaj otpad u ovoj industriji"*
- *"Ovo je moj odgovor na izazov rudarenja"*

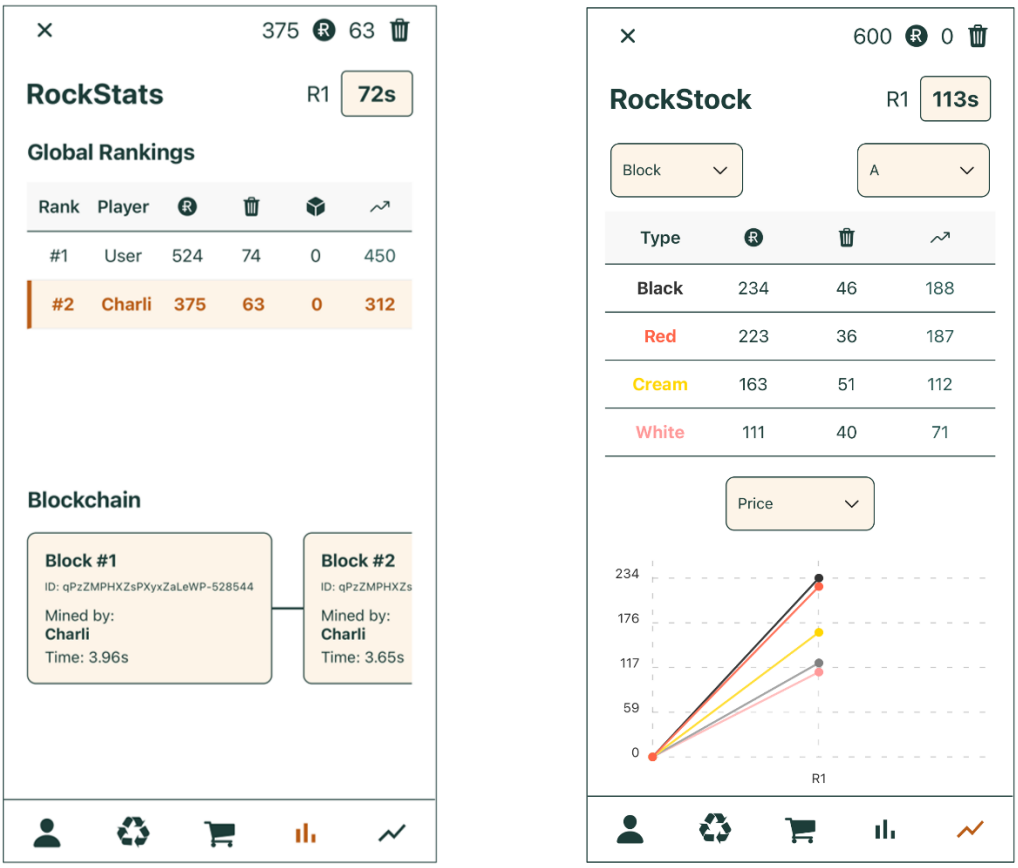
Poslužitelj u stvarnom vremenu prima svaku radnju, provjerava je li valjana prema trenutnom stanju igre (inventar, cijene, tajmer itd.) i:

- Primjenjuje se ako je sve u redu, ažurirajući podatke u memoriji (i u Firestore ako je potrebno).
- Odbija ga ako ne poštuje pravila ili stigne prekasno.

Poslužitelj zatim šalje ažuriranja svim igračima: nove inventare, rangiranje, promjene cijena ili rezultate rudarenja.

Zahvaljujući ovoj strukturi, svaka runda je:

- Interaktivno: igrači se mogu kretati između različitih ekrana i slobodno donositi odluke.
- Pošteno i dosljedno: ista pravila vrijede za sve, a stanje igre je usklađeno za cijelu grupu.
- Vremenski kontrolirano: *vrijednost roundEndsAtMs* osigurava da svi igrači završe rundu u isto vrijeme, bez obzira na manje razlike u njihovim uređajima ili mrežama.



Slika 7: STATS i STOCK ekrani

3.5. Izračuni na kraju runde i konačni rezultati

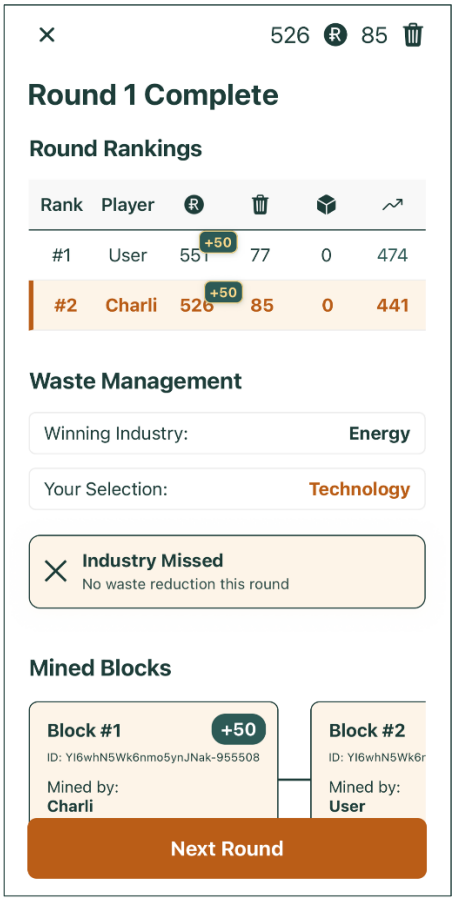
Kada istekne vrijeme runde (*roundEndsAtMs*), sustav prestaje prihvaćati nove radnje od igrača. Od tog trenutka, kontrola u potpunosti prelazi na backend, koji je odgovoran za zamrzavanje stanja igre, izračunavanje rezultata i jasno i dosljedno prikazivanje informacija.

Na serveru u stvarnom vremenu (Socket.IO), proces zatvaranja runde slijedi sljedeće korake:

1. **Akcije zamrzavanja:**
   Server mijenja stanje igre u *ROUND_CALCULATING* i prestaje prihvaćati nove poteze. Sve poruke koje stignu kasno se ignoriraju ili označavaju kao nevaljane.
2. **Izračunajte nagrade sigurno:**
   Korištenjem zaključavanja i provjera na razini rundi za sprječavanje ponavljanja, server:
   - Svaku radnju obrađuje samo jednom
   - Izbjegava dupliciranje nagrada ili brojanje istog događaja dvaput
   - Osigurava da istovremene radnje ne generiraju pogreške ili nedosljedne podatke
3. **Odabir pobjedničke industrije:**
   Konfigurirana logika primjenjuje se kako bi se odredilo koja je industrija pobijedila u rundi, što je ključna informacija za povratne informacije koje igrači dobivaju.
4. **Spremanje rezultata u Firestore**
   Ažuriranja servera:
   - Dokument igre u *games*/{*gameId*} (status, broj runde, rezultati)
   - Pojedinačne statistike u *users*/{*userId*}.*games*[*gameId*]

   Ako je dostupno, Firebase Admin upravlja tim operacijama sigurno i učinkovito.
5. **Obavijestite uređaje**
   Nakon što su izračuni napravljeni i podaci spremljeni, poslužitelj šalje niz događaja kao što su:
   - *ROUND_CALCULATING*
   - *ROUND_ENDED*
   - *REWARDS_ASSIGNED*

   Te poruke omogućuju klijentima da zatvore aktivnu fazu igre, pokažu da je runda gotova i zatim prikažu detaljne rezultate.



Slika 8: Ekran na kraju runde.

U aplikaciji, kontekst igre (*GameContextHybridRobust*) sluša te događaje i ažurira sučelje. Prelazi iz aktivnog prikaza igre na sažete ekrane, kao što su:

- *EndOfRoundScreen* prikazuje pobjedničku industriju, tko je riješio izazov rudarenja (ako ga je uopće bilo), kako su se inventar i RockCoini svakog igrača promijenili.
- *GameResultsScreen* (ili sličan završni ekran) prikazuje ukupni poredak igrača, prikupljene statistike iz odigranih rundi i druge korisne pokazatelje za raspravu ili razmišljanje.

Time se krug zatvara za cijelu grupu. Budući da su svi važni podaci pohranjeni u Firestoreu, igrači mogu:

- vratite se na glavni ekran i započnete novu igru kao domaćini s istom ili drugom grupom, ili
- Izađite iz igre i nastavite s drugim aktivnostima na tečaju.

Dovršene partije i njihovi podaci ostaju dostupni za kasniju upotrebu u WP5 (evaluacije, analize ili pregledi u budućim sesijama), bez obzira na to što igrači rade sljedeće.

Zajedno, ovaj tijek zaokružuje ciklus koji je započeo prilikom prijave: od autentifikacije, preko postavljanja igre, do aktivnog iskustva i na kraju do zaključka s jasnim rezultatima. Zahvaljujući kombinaciji servera u stvarnom vremenu, čvrste strukture u Firestoreu i dobro dizajniranih sažetaka, tehnička arhitektura postaje praktično, ponovljivo i korisno iskustvo u različitim obrazovnim kontekstima.

4. HOSTING, PRISTUP I DISTRIBUCIJA

S operativnog stajališta, interaktivni RockChain alat isporučuje se kao hostirana usluga u oblaku plus mobilna aplikacija. Centri za obuku ne moraju instalirati ili održavati lokalne servere; potrebni su im samo pristup internetu i prikladni su barem na jednoj od glavnih platformi.

Ovaj odjeljak nudi sažet, tehnički pregled načina na koji se alat hosta i pristupa. Za korak-po-korak, upute usmjerene na trenera o pripremi i vođenju sesije, čitatelji mogu pogledati WP4-A4 "Bilješke smjernica", kako bi korisnici mogli lako koristiti alat.

4.1. Backend hosting

Backend je hostan na ***Google Cloudu*** pod projektom RockChain i sastoji se od elemenata opisanih u Odjeljku 2:

- *Cloud Firestore*: pohranjuje igre, korisnike, tržišne podatke i zapise vezane uz učenje.
- *Firebase Cloud Functions*: implementirajte autentificirane operacije koje stvaraju i upravljaju igrama, generiraju probleme s rudarenjem, dodjeljuju nagrade i ažuriraju korisničke profile.
- *Node.js + Socket.IO server na Cloud Runu*: koordinira runde u stvarnom vremenu i djeluje kao autoritativni mjerač vremena i sudac.

Svi ključni podaci pohranjuju se u europskoj regiji, usklađujući implementaciju sa zahtjevima EU-a za zaštitu podataka i pojednostavljujući usklađenost za partnere koji djeluju u različitim zemljama. Budući da su ove usluge u potpunosti upravljane, nema potrebe za lokalnim poslužiteljima baza podataka ili prilagođenom infrastrukturom u obrazovnim centrima: održavanje i skaliranje obavljaju se na razini oblaka.

4.2. Distribucija mobilnih aplikacija

RockChain klijent distribuira se kao **mobilna aplikacija** barem za jednu od dvije glavne platforme:

- Android verzija (APK/AAB), prikladna za instalaciju putem popisa u trgovini ili izravnu distribuciju na upravljane uređaje.
- iOS verzija (IPA putem TestFlighta ili App Store listinga), ovisno o strategiji distribucije dogovorenoj s partnerima.

Također možete pronaći poveznice i/ili QR kodove objavljene na RockChain web stranici, ovdje: https://rockchain.eu/tool/

U praksi, treneri koji pripremaju tečaj mogu posjetiti RockChain web prostor, skenirati QR kod ili slijediti poveznicu za svoju platformu te instalirati trenutnu produkcijsku verziju aplikacije na svoje uređaje ili na tablete u vlasništvu institucije.

4.3. Pristupite tijeku rada za trenere i polaznike

Za krajnje korisnike, pristup RockChainu slijedi jednostavan, ponovljiv slijed:

1. **Instaliraj aplikaciju**
    - Polaznici i treneri instaliraju RockChain aplikaciju na svoje Android ili iOS uređaje, bilo na osobne telefone/tablete ili na uređaje u vlasništvu institucije gdje je aplikacija možda unaprijed instalirana.
2. **Kreirajte ili koristite račun**
    - Pri prvom korištenju registriraju se ili prijavljuju putem Firebase Autha s *LoginScreena*, kako je opisano u odjeljku 3.1.
    - Računi se mogu ponovno koristiti kroz sesije i tečajeve, tako da korisnici ne moraju svaki put ponovno registrirati se.
3. **Pridružite se ili organizirajte sesije igara**
    - Nakon prijave, svaki igrač završava na *GameScreenu*, gdje mu se u pozadini priprema osobni kod igre.
    - Male grupe odlučuju čiji će kod koristiti: jedan igrač djeluje kao domaćin dijeleći svoj kod i pokrećući igru, a ostali se pridružuju koristeći unos "Pridruži se igri".
    - Treneri mogu dopustiti učenicima da sami organiziraju svoje igre (po jednu po grupi) ili sami biti domaćini ako žele voditi demonstraciju ili preciznije kontrolirati vrijeme.

Iz perspektive trenera, to znači da vođenje RockChain sesije uglavnom sastoji:

- Osiguravajući da svi sudionici imaju instaliranu aplikaciju i mogu li se prijaviti.
- Organiziranje učenika u male grupe (svaka s domaćinom koji dijeli kod igre).
- Praćenje napretka grupa kroz runde i korištenje zaslona s rezultatima za debrifing.

4.4. Zahtjevi za centre za obuku

Budući da se RockChain oslanja na cloud hosting i mobilnu distribuciju, infrastrukturni zahtjevi za centre za obuku su minimalni:

- Mreža: Wi-Fi mreža u učionici s pristupom internetu, sposobna podržavati istovremene veze ovisno o broju uređaja korištenih u jednoj sesiji.
- Uređaji: Android ili iOS pametni telefoni/tableti koji zadovoljavaju minimalne OS zahtjeve za implementiranu verziju (kako je navedeno u RockChain dokumentaciji).
- Nema lokalne instalacije poslužitelja: centri ne moraju postavljati dodatne poslužitelje; Sva koordinacija igre, pohrana i autentifikacija odvijaju se u oblaku.

Ova kombinacija, Google Cloud backend, mobilnih aplikacija za Android i iOS te pristupa putem RockChain web područja, omogućuje korištenje interaktivnog RockChain alata u različitim zemljama i institucijama bez složenih lokalnih postavki, uz zadržavanje jedinstvene, centralizirane implementacije koju projektni partneri mogu održavati i ažurirati.

5. NADZOR, OTPORNOST I ODRŽAVANJE

Završni aspekt ovog dokumenta je osigurati da se interaktivni RockChain alat može nadzirati, otklanjati pogreške i održavati tijekom vremena, osobito tijekom pilot projekata u stvarnim učionicama. Cilj nije izgraditi težak sloj promatranosti, već osigurati dovoljnu vidljivost i robusnost kako bi partneri mogli razumjeti što se događa i održati sustav stabilnim.

5.1. Promatranost i logiranje

I klijent i poslužitelj uključuju lagane mehanizme za promatranje koji pomažu tijekom razvoja, pilot implementacije i analize incidenata.

Na klijentskoj strani, *GameContextHybridRobust* bilježi ključne tehničke signale konzoli, kao što su:

- Pomak sata između lokalnog uređaja i autoritativnog poslužitelja.
- Pokušaji ponovne sinkronizacije (na primjer, nakon ponovnog spajanja).
- Otkrivanje duplikata ili događaja izvan redoslijeda.

Ti se zapisi uglavnom koriste tijekom razvoja i pilot testiranja, kada programeri ili tehnički partneri pokreću aplikaciju s priloženim alatima za otklanjanje grešaka. Olakšavaju reprodukciju i dijagnosticiranje problema vezanih uz vreme, navigaciju i dosljednost stanja.

Na serverskoj strani, *logMetric* alat zapisuje strukturirane JSON događaje u Cloud Logging. Tipični događaji uključuju:

- Nove veze i prekidi veze.
- Runda počinje i završava.
- Operacije ponovne sinkronizacije.
- Pogreške ili neočekivani uvjeti.

S tim zapisima, tehničko osoblje može:

- Pogledajte koliko je utakmica odigrano u svakom periodu.
- Identificirajte obrasce pogrešaka ili uska grla u performansama (na primjer, ponovljene sinkronizacije za određene igre).
- Podržite otklanjanje pogrešaka kada partneri prijavljuju incidente, povezivanjem korisničkih prijava s konkretnim događajima u zapisima.

Cjelokupni pristup je namjerno lagan: nema složene nadzorne ploče, ali ima dovoljno strukturiranih informacija za osnovno održavanje, optimizaciju i rješavanje problema.

5.2. Otpornost i upravljanje kvarovima

Budući da je RockChain alat za multiplayer u stvarnom vremenu koji se koristi preko Wi-Fi-ja u učionici, mora se nositi s povremenom povezanošću i privremenim kvarovima bez narušavanja iskustva. Stoga je u arhitekturu ugrađeno nekoliko strategija otpornosti:

- Automatsko ponovno povezivanje socketa: Socket.IO adapter klijenta pokušava se automatski ponovno povezati kada je mreža privremeno izgubljena. Nakon uspješnog ponovnog povezivanja, ponovno šalje autentifikacijski token i *join_game* informacije kako bi server mogao ponovno spojiti socket na ispravnu igru i korisnika.
- Idempotentne operacije na poslužitelju: Operacije na strani poslužitelja oslanjaju se na identifikatore poput *roundID-a*, brojeva verzija i, gdje je potrebno, ID-jeva operacija. To omogućuje prepoznavanje ponovljenih ili kasnih poruka i njihovo ignoriranje umjesto da se primjenjuju dvaput, čime se izbjegavaju duplicirane nagrade ili nedosljedno stanje prilikom ponovnog pokušaja poruka.
- Zaključavanja na razini runde za kritične faze: Tijekom osjetljivih faza poput izračuna na kraju runde, server koristi zaključavanja na razini runde tako da se za svaku igru i rundu odvija samo jedan tijek izračuna u isto vrijeme. To sprječava uvjete utrke kada se pri kraju odbrojavanja pojavi mnogo radnji.
- Graciozno upravljanje desinkronizacijom: Ako je klijent bio offline, u pozadini ili doživi kratki prekid veze, kombinacija *GameContextHybridRobust* i autoritativnog servera omogućuje uređaju da se ponovno sinkronizira s trenutnim stanjem kada se vrati. Umjesto da se oslanja na ono što je prethodno prikazano, klijent može osvježiti svoj prikaz aktivne runde, inventara i rangiranja.

U praksi, ti mehanizmi znače da kratkoročni mrežni problemi ili prolazne pogreške ne bi trebali poništiti sesiju igre. Treneri obično mogu nastaviti s aktivnošću bez potrebe za dubokom tehničkom intervencijom, a učenici koji nakratko izgube vezu mogu se ponovno pridružiti i nastaviti igrati u dosljednom stanju.

5.3. Održavanje i budući razvoj

S aspekta održavanja, WP4. A5 iznosi nekoliko jednostavnih, ali važnih principa za daljnju evoluciju alata:

- Držite real-time logiku centraliziranom na autoritativnom serveru: Sve vremenski kritične odluke (tko je prvi rudario, kada runda završava, kako se primjenjuju nagrade) ostaju na serveru. Klijent konzumira *autoritativno stanje* umjesto da pokušava implementirati vlastita alternativna pravila. To smanjuje rizik od odstupanja između različitih klijentskih verzija.

- Tretirajte Cloud Functions kao jedinstvenu ulaznu točku za backend operacije: Svaka nova operacija koja mijenja trajne podatke treba biti implementirana kao Cloud Function, registrirana u Functions/index.js i uvijek pozivana kroz autentificirane omotače na klijentu. To održava sigurnosne provjere i poslovnu logiku na serveru te olakšava razmišljanje o protoku podataka.
- Promjene i migracije shema dokumenata: Kada se sheme dokumenata u Firestore razvijaju (na primjer, dodavanjem novih polja u *games/{gameId}* ili *users/{userId}.games[gameId]*), te promjene trebaju biti eksplicitno dokumentirane. Ako je potrebno prilagoditi postojeće podatke, skripte za migraciju ili alati mogu se smjestiti pod *funkcije/src* kako bi partneri imali jasan put za ažuriranje produkcijskih podataka.

Prateći ove prakse, tehnički partneri mogu dodavati nove značajke — poput dodatnih ekrana, dodatnih indikatora, produženog zapisivanja ili novih načina igre — bez ugrožavanja stabilnosti postojeće igre. RockChain ostaje održiv i proširiv i nakon početnog životnog vijeka projekta, uz očuvanje osnovnog ponašanja koje je potvrđeno tijekom pilot projekata u WP5.

6. ZAKLJUČCI

WP4. A5 je isporučio produkcijsku verziju interaktivnog RockChain alata, konsolidirajući dizajn, implementaciju i implementaciju iz prethodnih zadataka WP4. Dobiveni sustav objedinjuje moderan, višejezični mobilni klijent, backend temeljen na Firebaseu i autoritativni server u stvarnom vremenu na Cloud Runu, sve implementirano na skalabilnoj cloud infrastrukturi i pakirano u Android i iOS verzije koje se mogu instalirati na standardne uređaje koji se koriste u VET i obrazovanju odraslih.

Eksplicitnim dokumentiranjem arhitekture sustava, tokova izvođenja, produkcijskog i implementacijskog tijeka rada te procedura hostinga, pristupa i održavanja, ova aktivnost osigurava da RockChain nije jednokratni prototip, već ponovljiv i održiv resurs. Tehničko osoblje ima jasnu referencu o tome kako je alat strukturiran i kako ga ažurirati, dok treneri imaju stabilnu, instalacivu aplikaciju koja se može integrirati u stvarne tečajeve uz minimalno lokalno postavljanje.

Na taj način, interaktivni RockChain alat postaje praktična srž e-learning ponude projekta: on operacionalizira pedagoški i kurikularni rad iz prethodnih radnih radnih programa u konkretno, igrivo iskustvo koje se može koristiti u učionicama u zemljama partnerima. Alat je sada spreman podržati pilot aktivnosti u WP5 i pruža čvrstu osnovu za potencijalnu buduću eksploataciju i proširenje nakon trajanja projekta.