Aplikacija za izdavanje i praćenje studentskih nagrada na blockchainu

Ištvan, Mihael

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:364837

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEK
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski studij programsko inženjerstvo

APLIKACIJA ZA IZDAVANJE I PRAĆENJE STUDENTSKIH NAGRADA NA LANCU BLOKOVA

Diplomski rad

Mihael Ištvan

Osijek, 2023

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 07.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Mihael Ištvan
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1205R, 07.10.2021.
OIB studenta:	94023014140
Mentor:	izv. prof. dr. sc. Mirko Köhler
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	izv. prof. dr. sc. Mirko Köhler
Član Povjerenstva 2:	Miljenko Švarcmajer, mag. ing. comp.
Naslov diplomskog rada:	Aplikacija za izdavanje i praćenje studentskih nagrada na blockchainu
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Zauzeto za studenta: Mihael Ištvan. Napraviti aplikaciju preko koje studenti mogu primati i dijeliti nagrade ostvarene tijekom studiranja. Studenti za svaki položeni ispit dobivaju bodove i skupljaju ih u svom novčaniku. Te bodove mogu slati drugim studentima ili zamijeniti ih za nagrade, kao što su kino ulaznice ili slično, tako što bodove pošalju na adresu te nagrade. Također bodovi se mogu dobiti i za druge uspjehe kao što je dekanova ili rektorova nagrada,, osvojena medalja na studentskom natjecanju i slično. Treba predložiti način bodovanja i način registracije studenata.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	07.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 03.10.2023.

Ime i prezime studenta:	Mihael Ištvan
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1205R, 07.10.2021.
Turnitin podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za izdavanje i praćenje studentskih nagrada na blockchainu**

izrađen pod vodstvom mentora izv. prof. dr. sc. Mirko Köhler

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1.UVOD *5*

1.1 Zadatak diplomskog rada **6**

2. PREGLED POSTOJEĆIH RJEŠENJA *7*

3. TEHNOLOGIJE ZA RAZVOJ APLIKACIJE *9*

3.1 Tehnologije klijentske aplikacije **9**

3.1.1 Typescript 9

3.1.3 Učitavanje na klijentu i na poslužitelju 11

3.2 Tehnologije pozadinske aplikacije **13**

3.2.1 Injekcija ovisnosti 14

3.2.2 Validator klase, transformator klase i DTO 15

3.2.3 Baza podataka 16

3.3 Tehnologije lanca blokova **16**

3.3.1 Polygon lanac blokova 16

3.3.3 Metamask 17

3.4 Tehnologije skaliranja **19**

3.4.1 Docker 19

3.4.2 Oracle poslužitelj 20

3.4.3 Nginx i Certbot 21

4. STUDOREWARD APLIKACIJA *22*

4.1 Uvod **22**

4.2 Klijentska aplikacija **22**

4.2.1 Prijava i autentikacija 22

4.2.2 Zaštite 23

4.2.3 Stranica pregleda nagrada 24

4.2.4 Stranica moj profil 27

4.2.5 Stranica nagrade 27

4.2.6 Preuzimanje nagrada 28

4.3 Pozadinska aplikacija **32**

4.3.1 Prisma schema 32

4.3.2 Programski kod 33

4.4 Oracle Cloud instanca **39**

4.4.1 Domena 41

4.5 Pametni ugovor **42**

5. ZAKLJUČAK *44*

LITERATURA *45*

POPIS I OPIS UPOTRJEBLJENIH KRATICA *46*

SAŽETAK *48*

ŽIVOTOPIS *50*

DODATAK *51*

1.UVOD

U današnjem obrazovnom okruženju koje se brzo razvija, institucije visokog obrazovanja suočavaju se sa stalnim izazovom njegovanja motivacije i angažmana studenata. Dok se tradicionalni sustavi ocjenjivanja već dugo koriste za ocjenjivanje akademskog uspjeha, postoji sve veća potreba za istraživanjem inovativnih pristupa koji mogu učinkovito potaknuti i priznati postignuća studenata. Ovaj diplomski rad istražuje koncept sustava nagrađivanja studenata, posebno se fokusirajući na implementaciju nezamjenjivih tokena, skraćeno NFT, kao nagrade za uspjeh studenta.

U mnogim obrazovnim ustanovama prevladava zabrinutost u pogledu motivacije i razine angažmana studenata. Studentima je često teško ostati motiviran tijekom svog akademskog putovanja, što rezultira smanjenim uspjehom, nezainteresiranošću i ograničenim osobnim rastom. Tradicionalni oblici priznanja, kao što su certifikati ili trofeji, izgubili su nešto od svoje privlačnosti i ne nailaze na odjek kod digitalne generacije. Stoga postoji hitna potreba za istraživanjem alternativnih pristupa za učinkovito poticanje i uključivanje studenata.

Predloženi sustav nagrađivanja studenata ima za cilj iskoristiti moć podataka za prepoznavanje i nagrađivanje iznimnih postignuća studenata na pravedan i transparentan način. Prikupljanjem i analizom pojedinačnih podataka studenata, kao što su akademska postignuća i preporuke fakulteta, sustav može pružiti objektivnu mjeru uspjeha studenata. Jedna potencijalna manifestacija ovog sustava je korištenje NFT-ova, jedinstvene digitalne imovine koja se može autentificirati i kojom se može trgovati na platformama lanca blokova. Ovi NFT-ovi mogu poslužiti kao dostižne i korisne nagrade za studente koji se ističu akademski ili primaju pohvale od cijenjenih članova fakulteta.

Implementacija sustava nagrađivanja temeljenog na podacima s NFT-ovima nudi brojne potencijalne prednosti za studente. Prvo, studentima pruža prikaz njihovih postignuća, stvarajući osjećaj ponosa i postignuća. Budući da su NFT-ovi digitalna imovina, studenti mogu izložiti svoje nagrade u online portfeljima ili na platformama društvenih medija, šireći svoje priznanje izvan granica obrazovne ustanove. Nadalje, ovaj sustav nagrađivanja potiče zdravo natjecanje među studentima, potičući njihovu motivaciju za akademskim uspjehom. Utvrđivanjem jasnih kriterija za primanje NFT nagrada, studente se potiče da postavljaju ambiciozne ciljeve, teže izvrsnosti i aktivno sudjeluju u svom obrazovnom putovanju. Štoviše, korištenje NFT-ova kao nagrada usklađeno je s digitalnom prirodom modernog svijeta, pojavljujući se među studentima koji su uronjeni u digitalna okruženja. Prihvaćanjem novih tehnologija, obrazovne ustanove mogu premostiti između tradicionalnih metoda priznavanja i

očekivanja digitalne generacije, čime se povećava angažman i zadovoljstvo studenata. Ovaj rad nastoji pružiti vrijedne uvide i preporuke za obrazovne institucije koje žele iskoristiti snagu podataka i digitalnih sredstava za učinkovito motiviranje. U poglavlju koje slijedi, predstavljena su postojeća rješenja ideje studentskih nagrada. Također su predstavljene i tehnologije korištene za rad, razlozi njihovog korištenja i njihove glavne značajke. Zatim u poglavlju nakon tehnologija objašnjen je rad aplikacije diplomskog rada, objašnjen je programski kod aplikacije i način rada glavnih značajki. Ime aplikacije diplomskog rada zove se StudoReward.

1.1 Zadatak diplomskog rada

Napraviti aplikaciju preko koje studenti mogu primati i dijeliti nagrade ostvarene tijekom studiranja. Studenti za svaki položeni ispit ili povećanjem srednje ocjene uspjeha povećavaju svoj spektar mogućih ostvarivih nagrada. Nagrade je nakon ostvarenja moguće preuzeti na svoj novčanik i iskoristiti ili poslati ih drugim korisnicima. Registracija studenta odvija se pomoću srce.hr preusmjeritelja a studentski uspjeh se generira nasumično.

2. PREGLED POSTOJEĆIH RJEŠENJA

Direktna konkurencija ideji da na temelju uspjeha studenta na akademskoj godini trenutno ne postoji, već se u praksi studentima izdaju nagrade obilježene vrlo visokim obrazovnim postignućem npr. Rektorova nagrada. Doduše, neke od svjetski poznatih platforma za učenje nude nagrade na ostvarene ciljeve na njihovim stranicama.

Duolingova implementacija pod nazivom Duolingo za škole prilagođena je platforma dizajnirana za poboljšanje iskustva učenja jezika u obrazovnim okruženjima. Omogućuje učiteljima da učinkovito prate i podržavaju napredak svojih učenika. Putem ove platforme edukatori mogu kreirati virtualne učionice, pratiti uspjeh učenika i dodijeliti određene lekcije ili vježbe koje će uskladiti s njihovim nastavnim planom i programom. Učenici zarađuju virtualnu valutu, poznatu kao Lingots, kao nagradu za dovršetak lekcija i postizanje prekretnica u poznavanju jezika. Ovi se Lingoti mogu koristiti za kupnju stavki unutar aplikacije, potičući osjećaj postignuća i motivacije među učenicima. Sve u svemu, Duolingo za škole kombinira učinkovitost Duolingovog zaigranog učenja jezika sa strukturom i odgovornošću koja je potrebna u obrazovnim okruženjima. [1]

Implementacija Khan Academy je revolucionarna obrazovna platforma koja je revolucionirala način na koji učenici uče. Sa svojom opsežnom bibliotekom besplatnih video lekcija i vježbi koje pokrivaju širok raspon tema, Khan Academy nudi personalizirano iskustvo učenja u vlastitom tempu. Učenici mogu pratiti svoj napredak, zaraditi značke i natjecati se s kolegama, motivirajući ih da se aktivno bave gradivom. Učitelji također mogu koristiti resurse Akademije Khan za praćenje uspješnosti učenika, dodjeljivanje lekcija i prilagođavanje nastave individualnim potrebama. Iskorištavanjem snage tehnologije i uvida temeljenih na podacima, Khan Academy je demokratizirala obrazovanje, čineći visokokvalitetno učenje dostupnim milijunima učenika diljem svijeta, bez obzira na njihovu pozadinu ili lokaciju. [2]

Implementacija Codecademyja dinamična je i interaktivna platforma koja učenicima omogućuje svladavanje vještina kodiranja i programiranja. Kroz praktične vježbe kodiranja, projekte i izazove iz stvarnog svijeta, Codecademy pruža praktično i impresivno iskustvo učenja. Platforma nudi tečajeve o raznim programskim jezicima i tehnologijama, omogućujući korisnicima da razviju vrijedne vještine koje su vrlo tražene u tehnološkoj industriji. Učenici mogu steći značke i certifikate dok napreduju, pružajući opipljiv dokaz svoje stručnosti. Codecademy potiče zajednicu koja pruža podršku u kojoj studenti mogu dijeliti svoja postignuća i tražiti pomoć od kolega. Sa svojim sučeljem prilagođenim korisniku i pristupom učenju prilagođenim igricama, Codecademy je postao glavni resurs za ambiciozne programere

i tehnološke entuzijaste, pomažući im da izgrade snažne temelje za uspješnu karijeru u tehnologiji. [3]

3. TEHNOLOGIJE ZA RAZVOJ APLIKACIJE

U nastavku će biti objašnjene tehnologije korištene za razvoj aplikacije.

3.1 Tehnologije klijentske aplikacije

Next.js, popularni okvir izgrađen na temelju Reacta, stekao je značajnu privlačnost posljednjih godina zbog svoje sposobnosti da pojednostavi razvoj dinamičnih i učinkovitih klijentskih aplikacija. Omogućuje prikazivanje na strani poslužitelja, skraćeno SSR, što omogućuje brže učitavanje stranica, poboljšanu optimizaciju za tražilice, skraćeno SEO, i poboljšana korisnička iskustva. Uz Next.js programeri mogu s manje napora stvarati moćne jednostranične aplikacije, skraćeno SPA, poslužiteljski prikazane aplikacije, skraćeno SSR i statične internetske stranice [4]. Next.js također se ističe svojim sustavom usmjeravanja (engl. *routing*), pojednostavljujući navigaciju na strani klijenta i dinamičku izradu rute. Okvir pruža intuitivno usmjeravanje temeljeno na datotekama, gdje struktura direktorija i datoteka projekta diktira ponašanje usmjeravanja, pojednostavljujući organizaciju aplikacije. Next.js omogućuje jednostavno dohvaćanje podataka, s metodama kao što su *getServerSideProps* i *getStaticProps* koje olakšavaju dohvaćanje podataka tijekom iscrtavanja na strani poslužitelja i generiranja statične stranice. Nadalje, okvir nudi pogodnost API ruta, omogućujući stvaranje funkcija bez poslužitelja za rukovanje pozadinskom logikom unutar same aplikacije.

3.1.1 Typescript

TypeScript je nadskup JavaScripta koji jeziku dodaje statične tipove. Programerima pruža mogućnost eksplicitnog definiranja tipova za varijable, parametre funkcije, povratne vrijednosti i više. Ove oznake tipa pomažu u otkrivanju pogrešaka tijekom prevođenja programskog koda aplikacije, prije nego što se kod izvrši, što dovodi do robusnijeg i predvidljivijeg koda. TypeScriptov sustav tipova omogućuje bolju dokumentaciju koda, poboljšano održavanje koda i poboljšanu suradnju u većim bazama koda.

3.1.2 Razlike Reacta i Next.js-a

Next.js i React povezane su tehnologije, ali služe različitim svrhama unutar ekosustava internetskog razvoja. React je JavaScript biblioteka koju je razvio Facebook za izradu korisničkih sučelja. Pruža deklarativni pristup razvoju korisničkog sučelja temeljen na komponentama, omogućujući razvojnim programerima stvaranje višekratno upotrebljivih i interaktivnih komponenti koje učinkovito upravljaju stanjem i prikazom podataka. React se fokusira na pristup učitavanja na strani klijenta, skraćeno CSR, gdje se početno učitavanje

stranice sastoji od minimalnog HTML dokumenta koji učitava JavaScript pakete odgovorne za učitavanje sadržaja u pregledniku klijenta. Iako se React ističe u stvaranju dinamičkih i interaktivnih korisničkih sučelja, on se sam po sebi ne bavi potrebama učitavanja na strani poslužitelja ili generiranja statičkih stranica, skraćeno SSG. Tablica 3.1 prikazuje usporedbu Nest.js-a i Reacta.

Tablica 3.1 usporedba Next.js-a i Reacta [5]

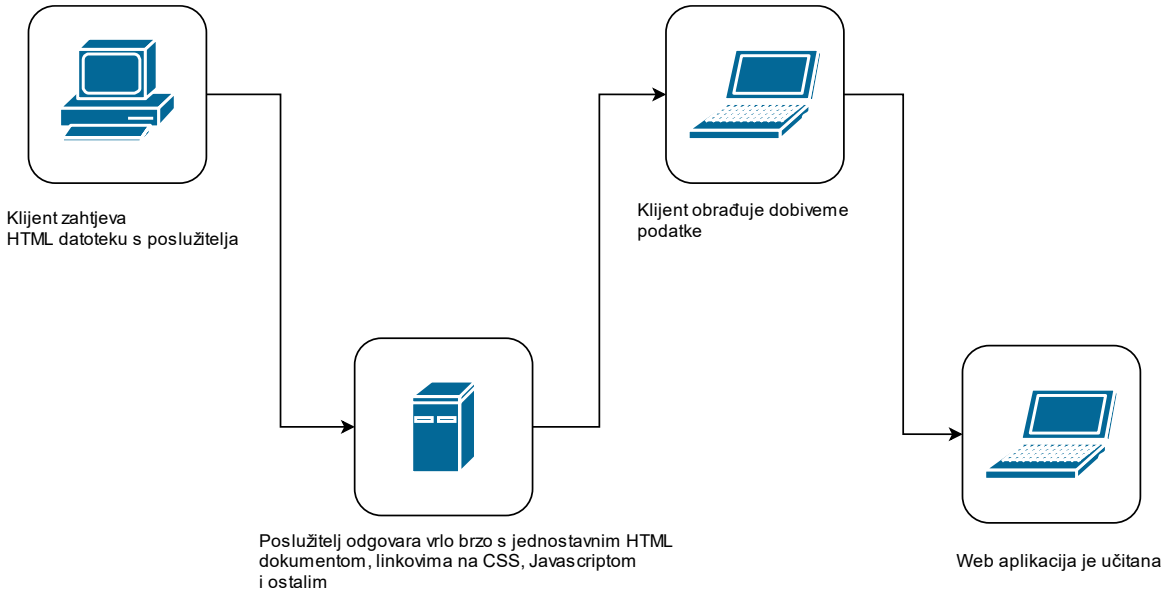
Kategorija	**Next.js**	**React**
Performanse	Internetske aplikacije razvijene sa Next.js-om su jako bazirane na platformi zahvaljujući SSR-u i SSG-u	Nema dijeljenja koda zbog loših performansi u React aplikacijama
Stanje obrazovanja	Teže učenje ako nema prethodnog znanja o Reactu	Lagano
Konfiguracija	Gotovo sve je konfigurabilno	Strogo
Fond talenata	Uzak	Širok
Zajednica	Mala s dobrom količinom resursa	Široka s mnogo resursa
Dokumentacija	Dobro napisana	Dobro napisana
Cijena razvoja	Mala	Mala
Značajke	Učitavanje na poslužiteljskoj strani, statičko generiranje stranica, automatsko usmjeravanje, optimizacija izvršnog koda aplikacije	React je dosta proširiv i neke od značajka se mogu omogućiti
SEO	Pogodan	Manje pogodan
Vanjska strana	Moguće je imati API prema vanjskoj strani koristeći API rute	React je uglavnom fokusiran na izgradnju UI-a
Multiplatformske aplikacije	Next.js je uglavnom iskoristiv za internetske aplikacije	React za internetske aplikacije, React Native za mobilne aplikacije

Typescript	Podržava	Podržava
Optimizacija slika	Ima ugrađenu podršku za podesive slike na manjim ekranima	Nije ugrađeno ali se može podesiti integracijom vanjskih paketa
Izvanmrežna podrška	Podržava	Podržava
Dinamične rute	Većima SSR internetskih okvira podržava dinamične rute	Obične rute

3.1.3 Učitavanje na klijentu i na poslužitelju

Prikazivanje na strani klijenta, skraćeno CSR, i prikazivanje na strani poslužitelja, skraćeno SSR dva su različita pristupa prikazivanju internetskog sadržaja i njegovoj isporuci korisnicima. Svaki pristup ima svoje prednosti i razmatranja, zadovoljavajući različite slučajeve upotrebe i ciljeve izvedbe.

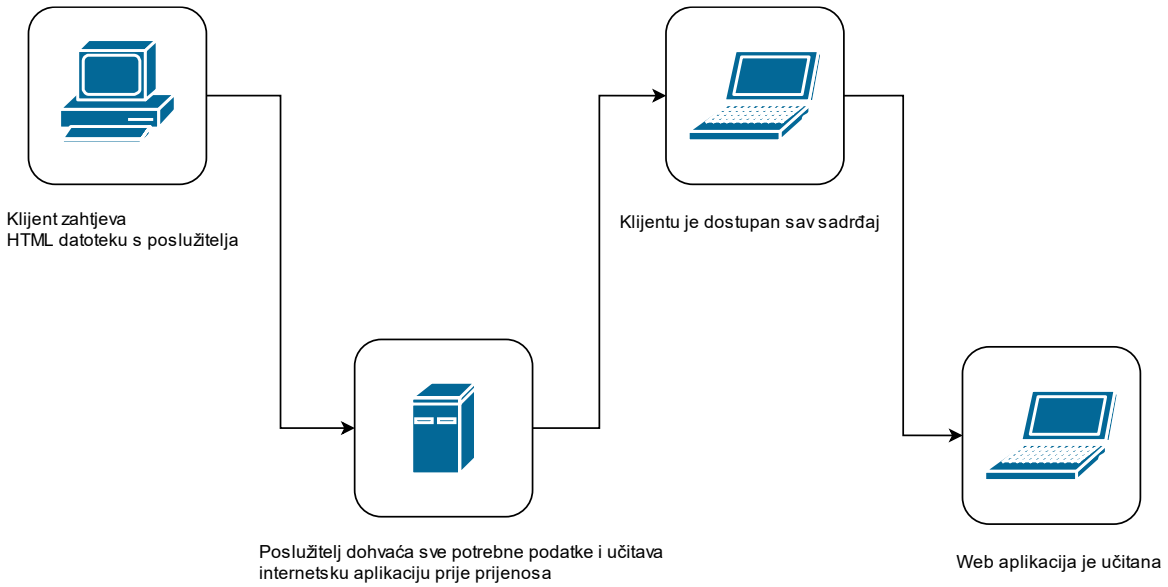
Učitavanje na strani klijenta početno učitavanje stranice sastoji se od laganog HTML dokumenta koji sadrži reference na JavaScript datoteke. Preglednik zatim preuzima i izvršava te skripte, koje generiraju sadržaj stranice na strani klijenta. Ovaj pristup pruža interaktivnije korisničko iskustvo jer interakcijama i ažuriranjima upravlja JavaScript bez potrebe za ponovnim učitavanjem cijele stranice. Aplikacije s jednom stranicom, skraćeno SPA, često koriste CSR, koristeći okvire kao što su React, Angular, Vue.js i zahtjevnije okvire tipa Next.js-a. Prednosti CSR-a uključuju brže prijelaze stranica nakon početnog učitavanja, budući da se s poslužitelja dohvaćaju samo ažuriranja podataka i korisničkog sučelja. Također omogućuje dinamičnija i interaktivnija korisnička sučelja. Međutim, početno vrijeme učitavanja CSR-a može biti sporije, što utječe na SEO i korisničko iskustvo, posebno na sporijim vezama ili manje moćnim uređajima. Budući da se stranica generira u pregledniku, tražilice bi mogle imati poteškoća s indeksiranjem sadržaja, osim ako se ne implementiraju tehnike specifične za SEO. Način učitavanja stranice na strani klijenta prikazano je na Slici 3.1.



Slika 3.1 *Dijagram učitavanja na strani klijenta*

Učitavanje na strani poslužitelja uključuje generiranje kompletnog HTML sadržaja na poslužitelju prije slanja klijentu. To znači da početno učitavanje stranice uključuje potpuno prikazani HTML, što poboljšava početnu brzinu učitavanja i vidljivost tražilice. Okviri poput Next.js napravljeni su za SSR, omogućujući razvojnim programerima postizanje ravnoteže između performansi i SEO-a. Glavna prednost SSR-a je brže početno učitavanje stranice, što dovodi do poboljšanog korisničkog iskustva i boljeg SEO indeksiranja. Osigurava da je sadržaj lako dostupan tražilicama i alatima za indeksiranje društvenih medija, što je ključno za vidljivost. Međutim, SSR može rezultirati manje interaktivnim stranicama, jer se izvršavanje JavaScripta odgađa nakon učitavanja stranice, što potencijalno dovodi do nešto sporijih naknadnih interakcija u usporedbi s CSR-om. Način učitavanja stranice na strani poslužitelja

prikazano je na Sl. 3.2. [6]



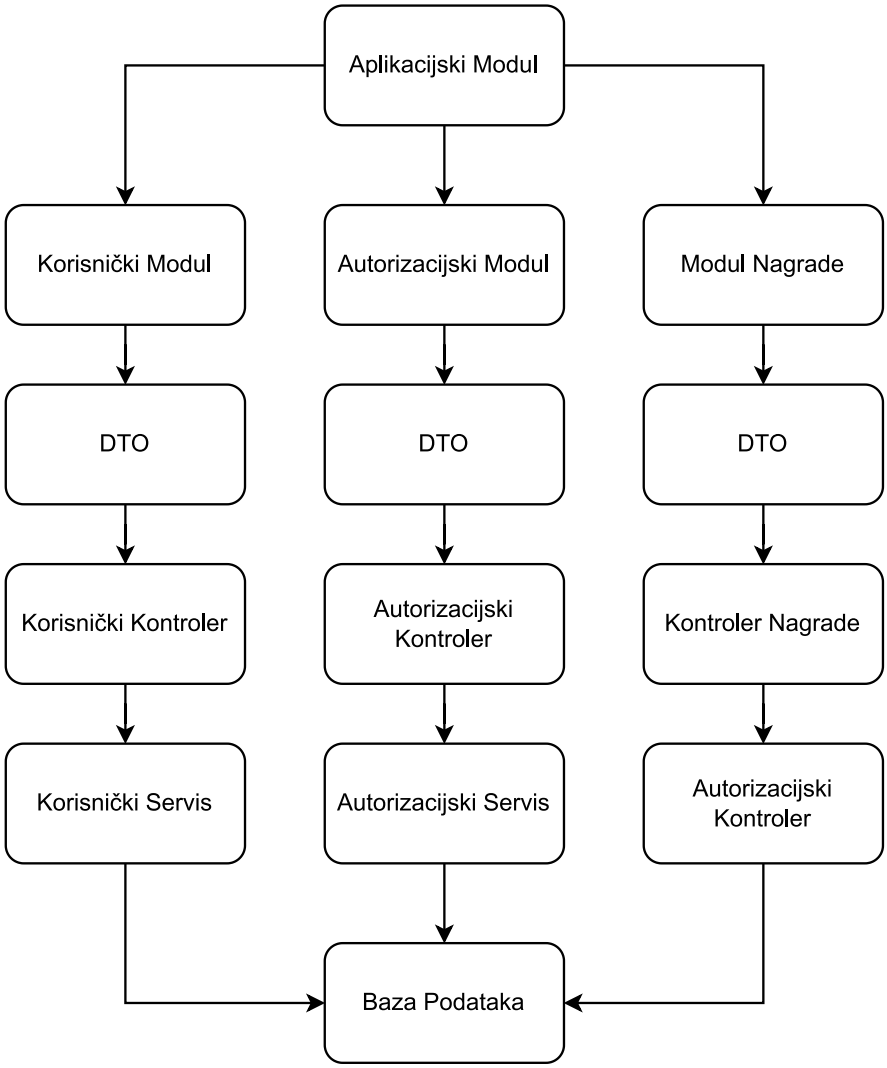
Slika 3.2 *Dijagram učitavanja na strani poslužitelja*

3.2 Tehnologije pozadinske aplikacije

Na pozadinskoj strani, Nest.js je pozadinski okvir za izgradnju skalabilnih i modularnih aplikacija na strani poslužitelja pomoću TypeScripta ili JavaScripta. Poznat je po svojoj modularnoj i organiziranoj arhitekturi, posuđujući koncepte iz Angulara i dovodeći ih u pozadinsko područje. Nekoliko je prednosti korištenja Nest.js-a za pozadinski razvoj. Nest.js potiče korištenje modularne arhitekture, omogućujući da aplikacija bude podijeljena na manje module koji se mogu ponovno koristiti i koje je lakše održavati i testirati. Ova arhitektura poboljšava organizaciju koda i pomaže u upravljanju složenošću većih projekata. Nest.js je napravljen s TypeScriptom na umu, koji dodaje tipove u JavaScript, čineći kod robusnijim i smanjujući pogreške tijekom izvođenja. To poboljšava kvalitetu koda, čitljivost i lakoću održavanja. Također jedna od ključnih značajki Nest.js-a je njegov ugrađeni sustav injekcije ovisnosti, skraćeno DI. To potiče labavu vezu (eng. *loose coupling*) između različitih dijelova aplikacije, čineći je fleksibilnijom i lakšom za testiranje. Nest.js isto tako koristi dekoratore za definiranje različitih aspekata aplikacije, kao što su usmjeravanje, međuprogramska oprema i provjera valjanosti podataka. To olakšava postavljanje krajnjih točaka i upravljanje objektima zahtjeva i odgovora.

Okvir pruža učinkovit način za rukovanje pogreškama i iznimkama, poboljšavajući kvalitetu pozadinske aplikacije. Nest.js ima robustan međuprogramske funkcije (engl. *middleware*) koje omogućuju presretanje i manipuliranje zahtjevima i odgovorima, što ih čini

učinkovitim za implementaciju međuprogramske rješenja kao što su autentifikacija, bilježenje i više. Nest.js ima ugrađenu podršku za internetske sokete, što omogućuje stvaranje aplikacija u stvarnom vremenu. Također pruža modul mikroservisa za izgradnju distribuiranih i skalabilnih aplikacija pomoću tehnologija kao što su RabbitMQ i Kafka. Slika 3.3 prikazuje primjer arhitekture u Nest.js-u. [7].



Slika 3.3 *Dijagram arhitekture Nest.js-a*

3.2.1 Injekcija ovisnosti

Injekcija ovisnosti, skraćeno DI, je obrazac dizajna koji se koristi u softverskom inženjerstvu za postizanje principa Inverzije kontrole, skraćeno IoC. To uključuje prosljeđivanje potrebnih ovisnosti, uglavnom objekata ili usluga klasi umjesto da ta klasa sama stvara ili upravlja svojim ovisnostima. Ovo pomaže odvojiti komponente, poboljšati modularnost i poboljšati mogućnost testiranja i održavanja baze koda. U kontekstu programiranja, ovisnost je svaki vanjski resurs, objekt ili usluga na koju se komponenta oslanja

za obavljanje svojih zadataka. Primjeri ovisnosti mogu uključivati baze podataka, mrežne usluge, konfiguracijske postavke i druge klase ili module. Nest.js je popularan okvir Node.js za izgradnju skalabilnih i održivih aplikacija na strani poslužitelja. U velikoj mjeri koristi obrazac ubacivanja ovisnosti za upravljanje ovisnostima i izgradnju modularnih aplikacija. Način na koji Nest.js pruža injekciju ovisnosti:

- Pružatelji: u Nest.js-u pružatelj je klasa koja se može umetnuti u druge klase. Dobavljač je odgovoran za stvaranje instanci klasa koje ispunjavaju određene uloge u vašoj aplikaciji. Te uloge mogu varirati od rukovanja operacijama baze podataka do upravljanja provjerom autentičnosti. Davatelji su ukrašeni *@Injectable()* dekoratorom.
- Modul u Nest.js-u definira opseg u kojem funkcionira sustav ubrizgavanja ovisnosti. U modulu je definiran koji su pružatelji dostupni za ubrizgavanje. *@Module()* dekorator se koristi za definiranje modula.
- Ubacivanje ovisnosti: kada klasa zahtijeva ovisnost, možete koristiti ubacivanje konstruktora. U konstruktoru klase specificirate potrebne ovisnosti kao parametre, a Nest.js automatski daje instance tih ovisnosti kada se instanca klase stvori. Na ovaj način niste odgovorni za samostalno stvaranje ili upravljanje tim ovisnostima. [8]

3.2.2 Validator klase, transformator klase i DTO

U kontekstu razvoja Nest.js-a, kombinacija biblioteka validatora klase i transformatora klase nudi moćan način za rukovanje provjerom valjanosti i transformacijom unosa. Validator klase omogućuje definiranje validacijskih pravila unutar klasa pomoću dekoratora. To programerima omogućuje da osiguraju da se dolazni podaci pridržavaju određenih formata, ograničenja i poslovnih pravila. S druge strane, transformator klase pomaže u transformaciji i pretvaranju sirovih ulaznih podataka u željene instance klase. Kada se primjenjuju zajedno, ove biblioteke pojednostavljuju proces primanja, provjere i transformacije dolaznih podataka unutar Nest.js aplikacije. Ovo ne samo da poboljšava dosljednost i integritet podataka, već i pojednostavljuje rukovanje pogreškama, povećavajući ukupnu robusnost aplikacije.

DTO-ovi ili objekti prijenosa podataka igraju ključnu ulogu u okviru Nest.js-a. DTO su klase koje se koriste za definiranje i strukturiranje podataka koji se razmjenjuju između različitih dijelova aplikacije, često između klijenta i poslužitelja. Oni pružaju jasan i standardiziran način prijenosa podataka dok odvajaju interni model domene od vanjske komunikacije. Korištenjem DTO-a, programeri mogu osigurati da su podaci koje šalje i prima aplikacija ispravno provjereni, tipizirani i strukturirani. To promiče integritet podataka,

sigurnost i dosljednost, a istovremeno omogućuje jednostavnu prilagodbu promjenama u zahtjevima aplikacije ili vanjskim sučeljima. DTO-ovi se dobro usklađuju s modularnom arhitekturom Nest.js-a i mogu se neprimjetno integrirati s validatorom klase i transformatorom klase, poboljšavajući cjelokupno rukovanje podacima i komunikacijske procese unutar aplikacije. [9]

3.2.3 Baza podataka

Postgres je sustav za upravljanje relacijskim bazama podataka otvorenog koda, skraćeno RDBMS, pokazao se kao pouzdana opcija bogata značajkama za moderne aplikacije. Sa svojim transakcijama usklađenim s ACID-om, skraćeno za atomičnost, dosljednost, izolacija i trajnost, proširivošću i podrškom za napredne značajke kao što su JSONB, prostorni podaci i pretraživanje cijelog teksta, Postgres nudi programerima snažno i fleksibilno rješenje za pohranu podataka. Njegova kompatibilnost sa SQL standardom, zajedno s zajednicom koja ga održava, čini ga izvrsnim izborom za aplikacije koje zahtijevaju strukturiranu i relacijsku pohranu podataka. [10]

3.3 Tehnologije lanca blokova

Solidity, programski jezik za pisanje pametnih ugovora na Ethereum lancu blokova, omogućuje programerima stvaranje samoizvršnih ugovora koji provode unaprijed definirana pravila i uvjete. Ovi pametni ugovori omogućuju interakcije prema aplikacijama i pružaju siguran temelj za decentralizirane aplikacije. [11]

3.3.1 Polygon lanac blokova

Polygon je dinamička platforma lanca blokova koja rješava uska grla skalabilnosti Ethereuma, nudeći rješenje za skaliranje s više lanaca. Ovaj okvir omogućuje programerima da konstruiraju i implementiraju decentralizirane aplikacije s izuzetnom učinkovitošću, uz smanjene naknade i ubrzane brzine transakcija. U srcu Polygonovog ekosustava nalazi se Mumbai Testnet, ključna komponenta koja programerima pruža namjensko okruženje za usavršavanje njihovih projekata prije njihovog pokretanja na glavnoj mreži. Mumbai testnet funkcionira kao pješčanik (engl. *sandbox*), omogućujući eksperimentiranje s pametnim ugovorima i transakcijama bez financijskog opterećenja povezanog s glavnom mrežom Ethereum. Nasuprot tome, Ethereum je pionirski lanac blokova koji je uveo koncept pametnih ugovora i decentraliziranih aplikacija, ali se ipak bori s problemima skalabilnosti zbog svog

energetski intenzivnog mehanizma konsenzusa Proof of Work. Ova razlika pokazuje kako Polygonov inovativni pristup, uključujući njegov Mumbai testnet, ima za cilj ublažiti ove izazove i pružiti optimalno okruženje za izgradnju i implementaciju rješenja temeljenih na lancu blokova. Polygon nudi prikaz svoje mreže na internetskoj aplikaciji Polygonscan. [12]

3.3.2 Pametni ugovori

Pametni ugovori predstavljaju revolucionarni napredak u području tehnologije lanca blokova. To su samoizvršujući ugovori s unaprijed definiranim pravilima i uvjetima koji su izravno ugrađeni u njihov kod. Ovi ugovori automatiziraju i provode izvršenje ugovora bez potrebe za posrednicima, osiguravajući transparentnost, sigurnost i nepromjenjivost. Jedna značajna implementacija pametnih ugovora je standard ERC721, koji upravlja stvaranjem i upravljanjem nezamjenjivim tokenima, skraćeno NFT, na Ethereum lancu blokova. Za razliku od zamjenjivih tokena poput kriptovaluta koje su međusobno zamjenjive, ERC721 tokeni su jedinstveni i nedjeljivi, što ih čini idealnim za predstavljanje vlasništva nad digitalnom ili fizičkom imovinom poput umjetnina, kolekcionarskih predmeta i virtualnih nekretnina. Svaki ERC721 token je poseban i može se autenticirati, nudeći novu dimenziju vlasništva i porijekla. Ova je inovacija dovela do ekosustava digitalnih kreatora, umjetnika i kolekcionara koji iskorištavaju ugovore ERC721 kako bi revolucionirali koncept vlasništva i trgovine u digitalnom krajoliku. [13]

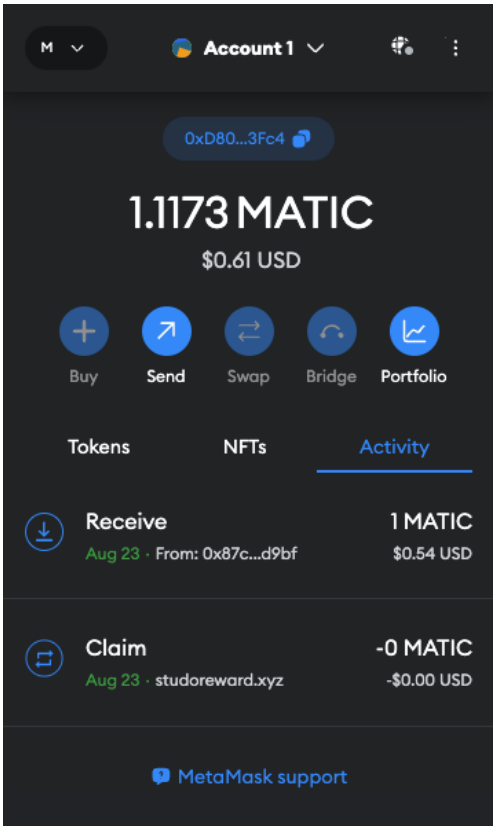
3.3.3 Metamask

Metamask je široko korišten novčanik za kriptovalute i proširenje za preglednik koji olakšava jednostavnu interakciju s decentraliziranim aplikacijama i Ethereum lancu blokova. Uz Metamask korisnici mogu sigurno pohranjivati, upravljati i prenositi svoju imovinu temeljenu na Ethereumu, kao i sudjelovati u raznim aktivnostima unutar decentraliziranog ekosustava. Jedan od najistaknutijih slučajeva upotrebe Metamaska je njegova integracija s nezamjenjivim tokenima. Metamask igra ključnu ulogu u omogućavanju korisnicima pristupa, kupnje, prodaje i interakcije s NFT-ovima. Evo kako se Metamask obično koristi s NFT-ovima:

- Novčanik i autentifikacija: Metamask se ponaša kao siguran novčanik u koji korisnici mogu pohraniti svoje Ethereum i druge ERC20 tokene. Kako bi koristili NFT tržišta ili platforme, korisnici povezuju svoj Metamask novčanik s tim platformama, omogućujući autentifikaciju i sigurne transakcije.

- NFT tržišta: NFT tržišta kao što su OpenSea, Rarible i Foundation omogućuju korisnicima kupnju, prodaju i trgovanje NFT-ovima. Metamask se integrira s ovim platformama, omogućujući korisnicima da pregledavaju, licitiraju, kupuju i upravljaju svojim NFT kolekcijama izravno iz svojih novčanika.
- Vlasništvo i porijeklo: Metamask pruža transparentan i provjerljiv način dokazivanja vlasništva i autentičnosti NFT-ova. Integracija novčanika s Ethereum lancu blokova osigurava da su zapisi o vlasništvu sigurni i zaštićeni od neovlaštenih promjena.
- Interakcija s decentraliziranim aplikacijama: Mnoge decentralizirane aplikacije koriste NFT za stvaranje jedinstvenih stavki ili iskustava u aplikaciji. Metamask olakšava ovu interakciju dopuštajući korisnicima da povežu svoje novčanike s decentraliziranim aplikacijama, omogućujući aktivnosti poput igranja, virtualnih svjetova i više.
- Stvaranje i kovanje NFT-ova: Neke platforme dopuštaju korisnicima da kuje svoje vlastite NFT-ove, u biti stvarajući i tokenizirajući svoj digitalni sadržaj. Metamask se često koristi za olakšavanje procesa kovanja, osiguravajući da su novostvoreni NFT-ovi sigurno pohranjeni u korisničkom novčaniku.

Metamaskovo korisničko sučelje i kompatibilnost s nizom decentraliziranih aplikacija i NFT tržišta učinili su ga ključnim alatom za svakoga tko je zainteresiran za istraživanje, prikupljanje i sudjelovanje u rastućem svijetu NFT-ova. Pojednostavljuje složen proces upravljanja imovinom temeljenom na Ethereumu i interakciju s aplikacijama koje pokreće lanac blokova, otvarajući uzbudljive mogućnosti za kreatore, kolekcionare i entuzijaste. [14] Slika 3.5 prikazuje izgled Metamask modala u internetskom pregledniku.



Slika 3.5 *Metamask aplikacija u internetskom pregledniku*

3.4 Tehnologije skaliranja

3.4.1 Docker

Docker je najpopularnija platforma za kontejnerizaciju, pruža prikladan i skalabilan način pakiranja i postavljanja ovih komponenti aplikacije. Docker je redefinirao razvoj softvera, implementaciju i upravljanje kroz svoju revolucionarnu tehnologiju kontejnerizacije. U svojoj srži, Docker omogućuje stvaranje laganih, izoliranih okruženja poznatih kao kontejneri. Ovi kontejneri enkapsuliraju aplikaciju zajedno sa svim njezinim ovisnostima, uključujući biblioteke i konfiguracijske datoteke, osiguravajući dosljedno ponašanje u različitim računalnim okruženjima. Ovo eliminira probleme kompatibilnosti i olakšava glatke prijelaze između faza razvoja, testiranja i proizvodnje. Dockerova učinkovitost leži u dijeljenju kernela glavnog operativnog sustava između spremnika, što rezultira minimalnom potrošnjom resursa i optimalnom gustoćom poslužitelja. Kontejneri su dopunjeni slikama (images), koje služe kao nacrti za izradu spremnika. Slika je snimka određenog okruženja, uključujući kod aplikacije, vrijeme izvođenja, sistemske alate i biblioteke. Slike mogu se lako dijeliti i verzionirati, omogućujući programerima da reproduciraju identična okruženja bilo gdje. Za definiranje sadržaja i konfiguracije slike, programeri koriste Dockerfiles. Dockerfiles su

tekstualne datoteke koje sadrže skup uputa za izradu slike korak po korak. Ove upute mogu uključivati povlačenje osnovne slike, kopiranje datoteka, instaliranje softvera, konfiguriranje postavki i više. Dockerfiles pružaju reproduktibilan i automatiziran način stvaranja slika, osiguravajući dosljednost i smanjujući ljudske pogreške. Kombinacija spremnika, slika i Dockerfilesa čini temelj Dockerove moći. Ova kombinacija osigurava da se aplikacije mogu besprijekorno razvijati, testirati i implementirati u različitim okruženjima. [15] Slika 3.6 prikazuje preporučeni izgled Dockerfilea za produkcijsku aplikaciju.

```
FROM node:16-slim
RUN apt-get update
RUN apt-get install -y openssl
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm ci
COPY . .
RUN npx prisma generate
RUN npm run build

CMD ["npm", "run", "start:prod"]
```

Slika 3.6 *Dockerfile produkcijske aplikacije*

3.4.2 Oracle poslužitelj

Oracle poslužitelj nudi robusnu okolinu bogatu značajkama za objavu produkcijskih aplikacija, što ga čini održivim izborom za implementaciju aplikacija izgrađenih s frontendom, backendom i bazom podataka. Jedan značajni aspekt je njegova besplatna razina, inicijativa koja pruža pristupačnu ulaznu točku za korisnike da iskuse mogućnosti Oracleovih usluga u oblaku bez inicijalnih troškova. Oracle Cloud besplatna razina uključuje niz usluga, uključujući Oracle autonomnu bazu podataka, izvršnu instancu i više, što korisnicima omogućuje da s lakoćom razvijaju, testiraju i postavljaju aplikacije.

U okviru besplatne razine, Oracle je predstavio instancu A1, koja je izgrađena na arhitekturi temeljenoj na Armu. Ova arhitektura, poznata kao A1, dizajnirana je za optimizaciju performansi i troškovne učinkovitosti za različita radna opterećenja, što je čini posebno prikladnom za aplikacije koje zahtijevaju uravnotežene mogućnosti obrade. Instance A1 nude troškovno učinkovito rješenje uz zadržavanje pouzdanih razina performansi. Ova je arhitektura

prikladna za programere, startupove i mala poduzeća koja traže povoljnu opciju za izgradnju i implementaciju aplikacija u oblaku.

Oracleov besplatni sloj, zajedno s arhitekturom A1, omogućuje korisnicima da istražuju Oracleove usluge u oblaku, uključujući njegova renomirana rješenja za baze podataka, bez tereta početnih troškova. Ova pristupačnost promiče inovacije i eksperimentiranje, a istovremeno nudi pouzdanu osnovu za poduzeća koja se mogu povećavati kako se njihovi zahtjevi razvijaju. Prednosti korištenja Oracle poslužitelja s Dockerom uključuju:

- Skalabilnost: Oracle poslužitelj podržava horizontalnu skalabilnost, omogućujući aplikacijama da se nose s povećanim prometom i zahtjevima korisnika. Uz Docker, aplikacije se mogu jednostavno skalirati postavljanjem više spremnika na različitim poslužiteljima, osiguravajući optimalno korištenje resursa.
- Sigurnost: Oracle poslužitelj nudi robusne sigurnosne značajke, uključujući napredne mehanizme provjere autentičnosti i enkripcije, osiguravajući zaštitu osjetljivih podataka. Docker pruža izolaciju između spremnika, smanjujući rizik od sigurnosnih propusta i neovlaštenog pristupa.
- Jednostavna implementacija i prenosivost: Docker omogućuje programerima da upakiraju svoju aplikaciju, zajedno s njezinim ovisnostima i konfiguracijama, u prijenosni spremnik. Ovo pojednostavljuje proces implementacije, budući da se spremnici mogu lako premještati kroz različita okruženja, uključujući razvoj, testiranje i proizvodnju. [16]

3.4.3 Nginx i Certbot

Nginx je internetski poslužitelj visokih performansi i obrnuti proxy poslužitelj koji se može koristiti uz Oracle poslužitelj za poboljšanje performansi, sigurnosti i skalabilnosti aplikacije. Nginx djeluje kao pristupnik, usmjeravajući dolazne zahtjeve na odgovarajuće pozadinske usluge. Konfiguracija aplikacije u Nginxu radi se na način da se izrađuje datoteka pod nazivom domene aplikacije koja radi kao proxy koji omogućuje kada se korisnik spoji na HTTP port 80 da mu prikaže aplikaciju pokrenutu u našem proizvoljnom portu, konkretko u primjeru na portu 3000. Certbot, besplatni alat otvorenog koda, pojednostavljuje postupak dobivanja i upravljanja SSL/TLS certifikatima. Osiguravanjem aplikacije HTTPS-om pomoću Certbota, osjetljivi podaci koji se prenose između korisnika i aplikacije ostaju šifrirani i zaštićeni. [17]

4. STUDOREWARD APLIKACIJA

U nastavku će biti objašnjene glavne značajke aplikacije.

4.1 Uvod

Kako je već prije navedeno, StudoReward je aplikacija za dohvaćanje i preuzimanje studentskih nagrada. Aplikacija koristi testni srce.hr preusmjeritelj (engl. redirect) kao prijavu u aplikaciju. Nakon uspješne prijave, studentu je prikazana stranica dostupnih nagrada koje su prikazane ovisno o njegovom uspjehu. Zbog limitacije srce.hr sustava uspjeh studenta je generiran kod prve prijave. Svaka definirana nagrada uz parametre koje joj označavaju naslov i opis, ima i definirane kriterije za prikaz studentu. U ovom slučaju to su parametri koji student mora ostvariti kako bi mu nagrada postala vidljiva. Aplikacija također daje uvid u trenutni uspjeh studenta, prikaz preuzetih nagrada i detaljan prikaz nagrada. Glavna funkcionalnost aplikacije je spajanje korisničkog proizvoljnog novčanika u lancu blokova na kojeg može preuzeti nagrade. Studentske nagrade nemaju određen oblik a u primjeru testne aplikacije koristit će se ERC721 token kao primjer nagrade.

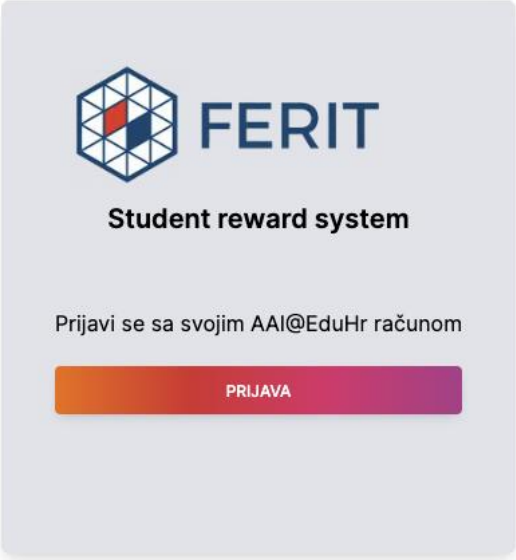
4.2 Klijentska aplikacija

4.2.1 Prijava i autentikacija

Prijava u sustav radi se pomoću srce.hr preusmjeritelja. Razlog tome bio je što svaki student već ima vlastiti login kojeg koristi da bi se prijavio u sustav Merlin i Studomat, time nema potreba za izradom dodatnog vanjskog logina. Klikom na prijavu studenta se preusmjeruje na srce.hr formu za prijavu koja nakon uspješne prijave vrati studenta na stranicu dostupnih nagrada. Autentikacija se bazira na JWT-u kojeg srce.hr preusmjeritelj vrati pozadinskom programu koji nakon primanja JWT-a i obavljanja svojih funkcija sprema JWT u kolačiće. Na slici 4.3 vidljiv je izgled dobivenog JWT-a od srce.hr. Tim kolačićima se može pristupiti na klijentskoj aplikaciji gdje se radi dodatno dohvaćanje objekta trenutno prijavljenog studenta kako bi se potvrdila prijava.

```
{
  aai: '',
  ime: 'Branka',
  prezime: 'Zovko-Cihlar',
  titula: '',
  email: '',
  ovlasti: null,
  iat: 1693769892,
  exp: 1693856292
}
```

Slika 4.3 *Primjer JWT-a vraćenog od preusmeritelja*



Slika 4.4 *Prijava u sustav*

JWT je često korišćen mehanizam za autentifikaciju i autorizaciju u internetski aplikacijama. To je kompaktno, samosadržajno i digitalno potpisano token koji se sastoji iz tri glavna dijela: zaglavlja (engl. *header*), glavnog dijela (engl. *payload*) i potpisa (engl. *signature*). Zaglavlje sadrži informacije o algoritmu koji se koristi za potpisivanje tokena. Tvrdnje često uključuju informacije poput identiteta korisnika, vremena isteka tokena i dozvoljenih akcija. Ove tvrdnje mogu biti dekodirane, ali ne i promenjene, jer se integritet tokena osigurava digitalnim potpisom koji se generiše na osnovu zaglavlja, tvrdnji i tajnog ključa poznatog samo poslužitelju.

4.2.2 Zaštite

Autentikacijske zaštite (engl. *guards*) služe kako bi zabranili korisnicima pristup stranici kojoj ne bi trebali imati pristup u trenutnom stanju. Aplikacija koristi dvije autentikacijske zaštite: zaštita početne stranice i zaštita svih ostalih stranica. Zaštita početne stranice štiti stranicu da joj ne mogu pristupiti korisnici koji su već prijavljeni a zaštita svih ostalih stranica postoji kako joj ne bi mogli pristupiti korisnici koji nisu prijavljeni u sustav. Ovim zaštitama omogućujemo da pristup aplikaciji imaju samo studenti i da svako od studenata ima svoj jedinstveni prikaz nagrada. Također štitimo nagrade od potencijalnih neželjenih preuzimanja nagrada i zloupotrjebljavanja aplikacije. Zaštite rade na način da učitavaju JWT iz kolačića te ako postoji ili ne postoji poduzimaju definirane mjere koje su preusmjeravanje

na početnu stranicu ako student nije prijavljen ili na stranicu dostupnih nagrada u slučaju kada je student prijavljen. Slike 4.5 i 4.6 prikazuju programski kod navedenih zaštita.

```
"use client";

import { useEffect } from "react";

const requireAuth = () => {
  const loggedIn = document.cookie.includes("accessToken");

  if (loggedIn) {
    window.location.href = "/items";
  }
};

export const LandingAuthPageInvisible = () => {
  useEffect(() => {
    requireAuth();
  }, []);

  return <></>;
};      You, last month • implemented ferit auth
```

Slika 4.5 *Zaštita početne stranice*

```
"use client";

import { useEffect } from "react";

const requireAuth = () => {
  const loggedIn = document.cookie.includes("accessToken");

  if (!loggedIn) {
    window.location.href = "/";
  }
};

export const AuthPageInvisible = () => {
  useEffect(() => {
    requireAuth();
  }, []);

  return <></>;
};      You, last month • implemented ferit auth …
```

Slika 4.6 *Zaštita svih stranica*

4.2.3 Stranica pregleda nagrada

Stranica dostupnih nagrada studentu omogućuje prikaz dostupnih nagrada. Nagrade koje mu nisu vidljive na ovoj stranici su ili nedostupne ili već preuzete. Stranica svoju jednu funkcionalnost radi na način da nakon potpunog učitavanja stranice koristeći *useEffect* pokrene

funkciju *fetchRewards* koja pomoću NPM paketa *Axios* dohvati dostupne nagrade sa pozadinske aplikacije i nakon što su nagrade dohvaćenje ih spremi u varijablu stanja pomoću kuke (engl. *hook*) setRewards i prikazuje ih u tabličnom formatu u obliku ponavljajućih komponenata. Slika 4.7 prikazuje programski kod za dohvaćanje nagrada, Slika 4.8 prikazuje programski kod za prikazivanje nagrada.

```
export default function Items() {
  const [isLoading, setLoading] = useState(true);
  const [rewards, setRewards] = useState([]);
      You, 4 weeks ago • progress commit …
  useEffect(() => {
    fetchRewards();
  }, []);

  const fetchRewards = async () => {
    try {
      const response = await axios.get(
        `${process.env.NEXT_PUBLIC_API_URL}/reward`,
        {
          headers: { Authorization: `Bearer ${getCookie('accessToken')}` },
        }
      );
      setRewards(response.data);
      setLoading(false);
    } catch (error) {
      console.error('Error fetching user data:', error);
      setLoading(false);
    }
  };
```

Slika 4.7 *Programski kod dohvaćanja nagrada*

```
<>
  {rewards.length > 0 ? (
    rewards.map((reward: any) => (
      <Link
        key={reward.id}
        href={`reward/${reward.id.toString()}`}
        className="grid md:grid-cols-2 gap-4 border-2 border-gray-300
        max-w-lg group rounded-lg border border-transparent px-5 py-4
        transition-colors hover:border-gray-300 hover:bg-gray-100 hover:dark:border-neutral-700      You, 1 second ago • Un
        hover:dark:bg-neutral-800/30"
      >
        <div className="flex items-center justify-center">
          <Image
            className="relative dark:drop-shadow-[0_0_0.3rem_#ffffff70]"
            src={reward.imageUrl}
            alt="Next.js Logo"
            width={200}
            height={200}
            priority
          />
        </div>

        <div className="flex flex-col justify-center">
          <h2 className={`mb-3 text-2xl font-semibold`}>
            {reward.name}{' '}
            <span className="inline-block transition-transform group-hover:translate-x-1 motion-reduce:transform-none">
              -&gt;
            </span>
          </h2>
          {reward.shortDescription ? (
            <p className={`m-0 max-w-[30ch] text-sm opacity-50`}>
              {reward.shortDescription}
            </p>
          ) : null}
        </div>
      </Link>
    ))
  ) : (
    <>Trenutno nema dostupnih nagrada :(</>
  )}
</>
```

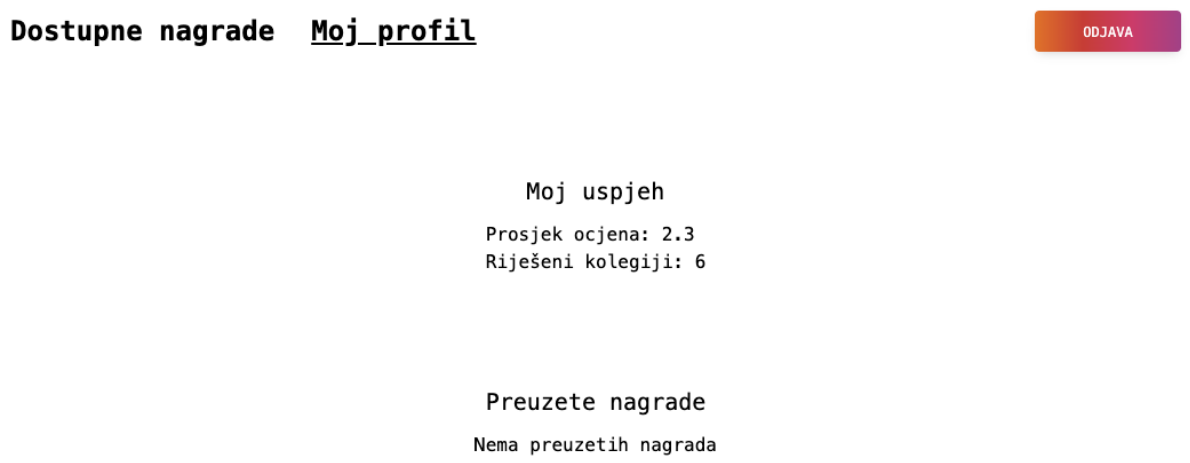
Slika 4.8 *Programski kod prikazivanja nagrada*

Osim svoje osnovne funkcionalnosti stranica omogućuje gumbe koji služe kao navigacija po aplikaciji: Dostupne nagrade i Moj profil. Navigacija se sastoji od dva gumba od kojih barem jedan uvijek je pritisnut i uz to daje studentu mogućnost za odjavom, koja ga preusmjeri na prijavu. Pomakom na nagradu prikazuje se efekt lebdenja koji daje korisniku osjećaj interakcije sa stranicom i klikom na nagradu se otvara detaljan prikaz nagrade. Stranica je također zaštićena sa *ProtectPage zaštitom*. Slika 4.9 prikazuje stranicu dostupnih nagrada.

Slika 4.9 *Stranica dostupnih nagrada*

4.2.4 Stranica moj profil

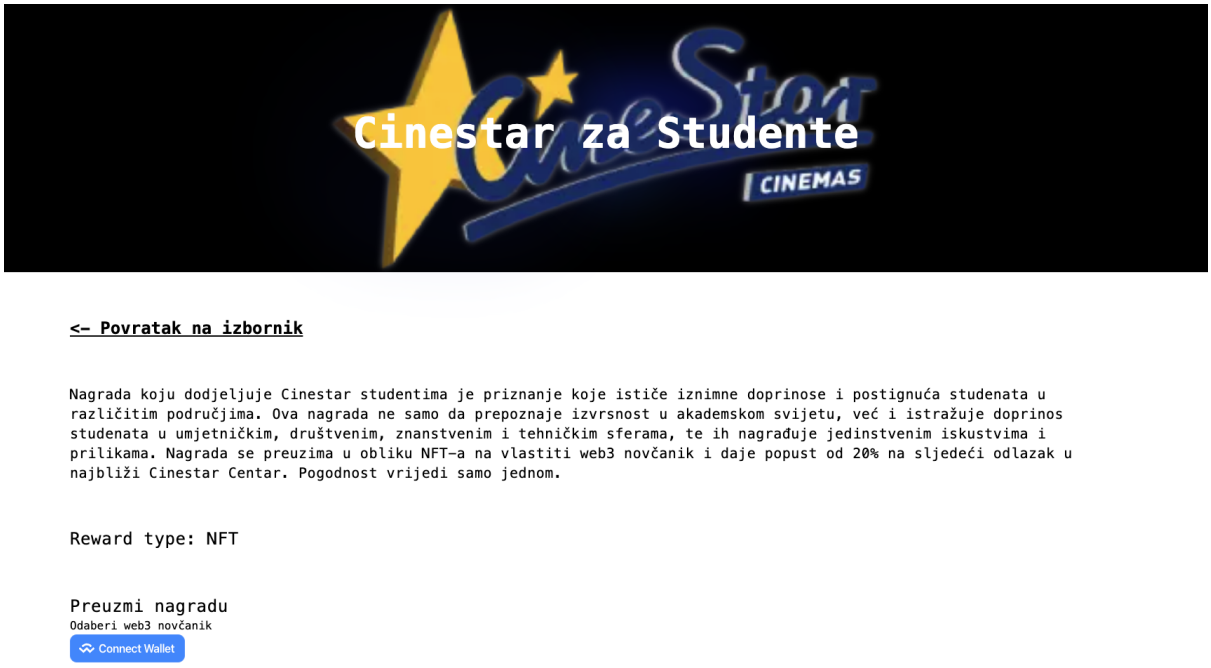
Stranica moj profil daje uvid studentu u njegov uspjeh i u preuzete nagrade. Preuzete nagrade su prikaz isti kao i na stranici dostupnih nagrada u obliku tablice i ponavljajućih komponenata. Stranica se nalazi na odvojenoj ruti ali izgledom kao dio cjelokupnog menija. Stranica moj profil svoje glavne funkcionalnosti izvršava na način da na inicijalnom učitavanju stranice radi poziv prema pozadinskoj aplikaciji, koja vraća detaljan objekt trenutnog studenta koji sadržava ugniježdena polja s informacijama o uspjehu studenta i o preuzetim nagradama. Ako student nije preuzeo niti jednu nagradu onda mu je prikazana odgovarajuća poruka. Slika 4.10 prikazuje stranicu moj profil.



Slika 4.10 *Stranica moj profil*

4.2.5 Stranica nagrade

Stranica detaljnog pregleda nagrada omogućuje detaljan prikaz nagrade s naslovom i potpunim opisom. Ako se nagrada prikaže studentu na stranici dostupnih nagrada znači da je može preuzeti. Glavna funkcionalnost stranice detaljnog pregleda nagrada je preuzimanje nagrada koja se radi pomoću paketa WalletConnect. WalletConnect je primijenjen u obliku poveznice klijentske aplikacije sa pametnim ugovor. Pomoću WalletConnect-ovog Web3Modala koji se integrira u klijentsku aplikaciju studenti mogu trenutno povezati svoj novčanik u lancu blokova s aplikacijom i time na siguran način dokazati i provesti preuzimanje nagrade na svoj novčanik. Podatke o nagradama stranica dobiva preko *propsa* od stranice dostupnih nagrada koja prosljeđuje objekt nagrade. Slika 4.11 prikazuje stranicu detaljnog pregleda nagrade.



Slika 4.11 *Stranica detaljnog pregleda nagrade*

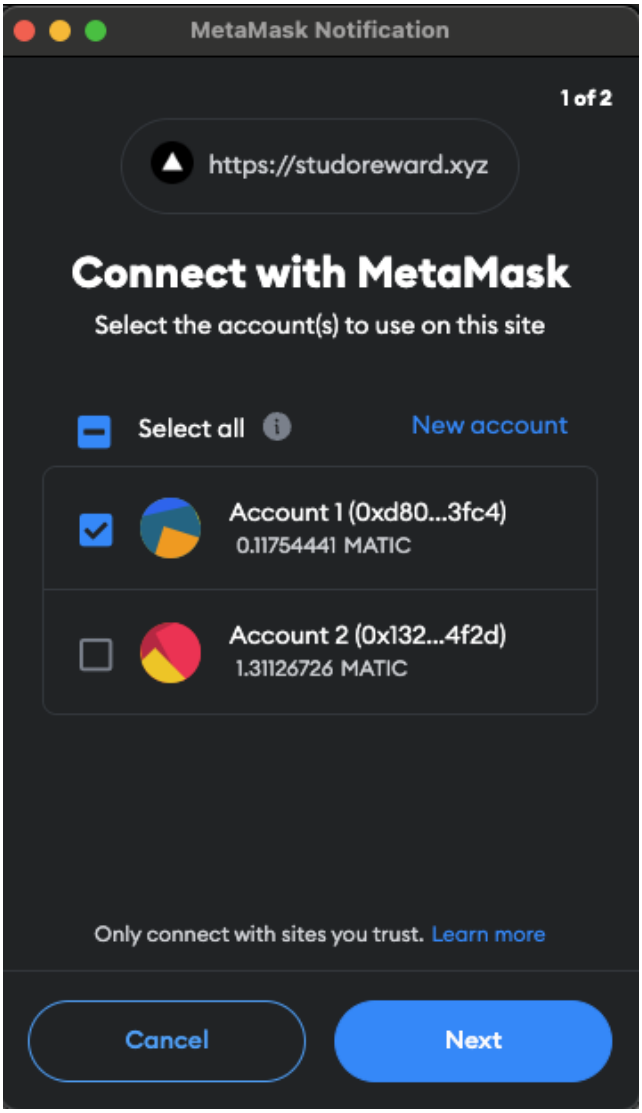
4.2.6 Preuzimanje nagrada

Da bi student preuzeo nagradu potrebno je otići na stranicu detaljnog prikaza nagrade. Tamo klikom na „Connect Wallet“ gumb otvara se *Web3Modal* gdje odabire novčanik na kojeg hoće preuzeti nagradu. Novčanik može spojiti odabirom na aplikaciju ili skeniranjem QR koda. Slike 4.12 i 4.13 prikazuju „Connect Wallet“ gumb i Metamask modal koji se otvara klikom na gumb.

Slika 4.12 *Connect Wallet gumb*

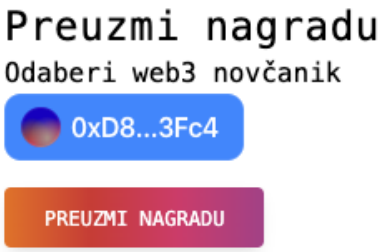
Slika 4.13 *Modal za odabir novčanika*

Za ovaj primjer odabrana je aplikacija Metamask koja kod odabira pita korisnika za adresu na koju se hoćemo spojiti. Slika 4.14 prikazije modal za odabir računa na Metamasku.



Slika 4.14 *Modal za odabir računa u Metamasku*

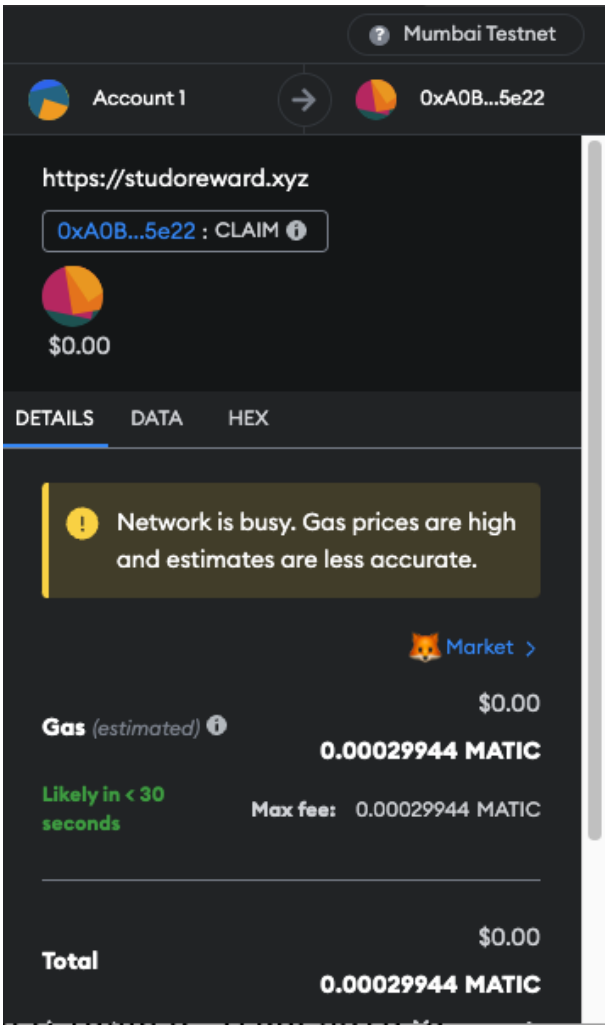
Nakon što odaberemo račun, klijentska aplikacija radi provjeru ako je korisnik već preuzeo trenutnu nagradu pozivom na pozadinsku aplikaciju. Ako nije preuzeo, onda mu prikazuje gumb „Preuzmi nagradu“. Slika 4.15 prikazuje gumb za preuzimanje nagrade.



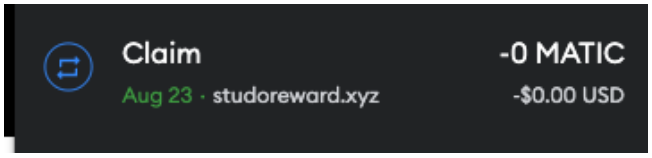
Slika 4.15 *gumb za preuzimanje nagrade*

Klikom na gumb za preuzimanje nagrade otvara se modal od odabrane povezane aplikacije, u ovom slučaju to je Metamask. Ovaj modal pita za potpis (engl. *signature*) od korisnika da

potvrdi transakciju na lanac blokova. Razlog zašto pita za dopuštenje transakcije je zato što za preuzimanje nagrade potrebno je platiti naknadu za plin. Svakodnevno se naplata za plin mijenja i kroz razdoblja u danu može biti skuplji a nekada jeftiniji. Nakon što je prihvaćena naplata za plin i transakcija je time potpisana, klijentska aplikacija radi POST API poziv na pozadinsku aplikaciju kako bi se napravio zapis da je korisnik preuzeo nagradu. Slike 4.16 i 4.17 prikazuju Metamask modal za prihvaćanje transakcije i prikaz nakon uspješne transakcije.

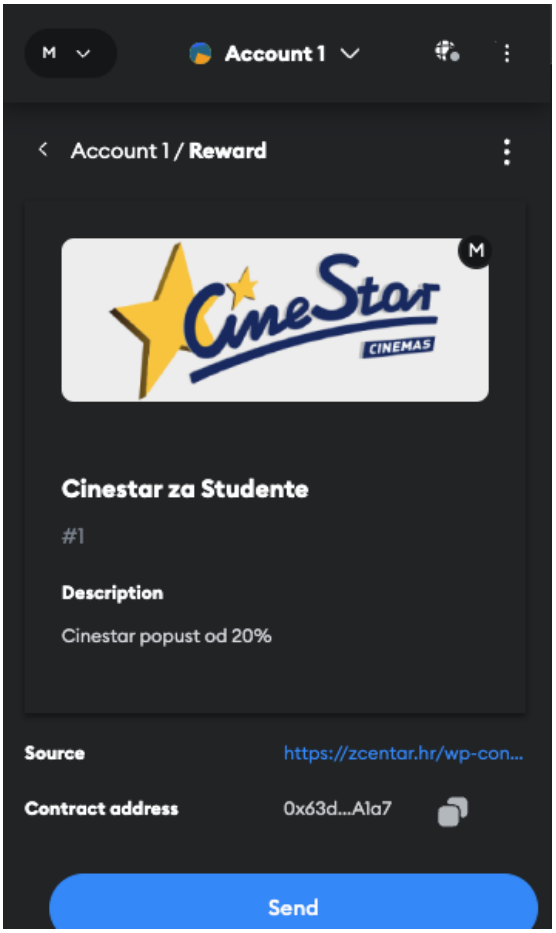


Slika 4.16 *Modal za prihvaćanje transakcije*



Slika 4.17 *Prikaz uspješne transakcije*

Nakon preuzete nagrade detalji transakcije se mogu pogledati na Polygonscanu klikom na gumb „Pogledaj transakciju“. Slika 4.18 prikazuje Metamask modal nakon uvezene nagrade.



Slika 4.18 *Prikaz u Metamasku nakon uvezene nagrade*

4.3 Pozadinska aplikacija

4.3.1 Prisma schema

Schema Prisme definira strukturu baze, također služi za interakciju s bazom i omogućuje *Intellisense* kod pisanja koda za neki zadatak baze podataka. Za interakciju s bazom potrebno je u .env datoteci definirati URL koji služi za spajanje na bazu podataka i izraditi Prisma modul s prije objašnjenom predefiniranom Nest.js naredbom. Slika 4.20 Prisma konfiguraciju i User model u Prisma schemi.

```
generator client {
  provider = "prisma-client-js"
}

You, last month | 1 author (You)
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

You, 4 weeks ago | 1 author (You)
model User {
  aai       String  @id
  firstName String
  lastName  String
  title     String?
  email     String?
  iat       Int
  exp       Int

  statistics UserStatistics?
  claims     Claim[]
}
```

Slika 4.20 *Konfiguracija i User model u Prisma schemi*

4.3.2 Programski kod

Kao što je već prije navedeno, Nest.js preporučuje u svojoj dokumentaciji striktno praćenje njihove definirane strukture koda, također i daje predefinirane CLI naredbe za izradu takve strukture.

4.3.2.1 Autorizacija

Za autorizaciju korisnika koji se prijavljuje putem srce.hr preusmjeritelja koristi se GET ruta definirana u *Auth* kontroleru. Slika 4.22 prikazuje programski kod funckije za autentikaciju studenta.

```
@Controller('auth')
export class AuthController {
  constructor(private authService: AuthService) {}

  @Get('/handle-redirect')
  @ApiOperation({
    tags: [`Auth`],
    summary: 'Handle redirect',
  })
  @Redirect(`${process.env.APP_URL}/items`, 302)
  async handleRedirect(
    @Query() dto: HandleRedirectDto,
    @Res({ passthrough: true }) response: Response,
  ) {
    if (dto.jwt) {
      try {
        const { token } = await this.authService.handleRedirect(dto);

        response.cookie('accessToken', token.access_token);
      } catch (err) {
        return new ResponseError('AUTH.REDIRECT', err);
      }
    }
  }
}
```

Slika 4.22 *Programski kod funkcije za autentikaciju studenta*

Navedena funkcija očekuje da joj srce.hr nakon uspješne autentikacije studenta preusmjeri pomoću URL-a pozadinske aplikacije JWT studenta. Nakon što zaprimi studentov JWT provjerava se koristeći Prismu postoji li student s takvim imenom u bazi, ako ne postoji izradi ga i generira novi JWT kojeg sprema u kolačiće i preusmjeruje korisnika na stranici dostupnih nagrada.

4.3.2.2 Zaštita

Zaštita ruta pozadinske aplikacije radi se pomoću JWT strategije koja kao posredna funkcija (engl. *middleware*) provjerava JWT kojeg klijentska aplikacija šalje kako bi potvrdila identitet korisnika. Provjera se radi tako da iz glave (engl. *headers*) HTTP poziva izvuče autorizacijsko polje koje sadržava JWT, dekodira ga i provjerava integritet dekodiranog objekta. Provjeru radi na način da pretražuje bazu ako postoji korisnik sa danim *aai* identifikacijskim ključem, koji je zbog limitacije srce.hr uvijek „test“. Ako ne postoji korisnik u bazi s danim identifikacijskim ključem onda pozadinska aplikacija vrati grešku sa statusom 403. Za implementaciju JWT strategije korišten je NPM paket passport koji nudi predefinirane strategija autentikacije i omogućuje funkcije za rad s JWT-om. Slika 4.23 prikazuje programski kod JWT strategije.

```
@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy, 'jwt') {
  constructor(config: ConfigService, private prisma: PrismaService) {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      secretOrKey: config.get('JWT_SECRET'),
    });
  }

  async validate(payload: { aai: string }) {
    const user = await this.prisma.user.findUnique({
      where: { aai: payload.aai },
    });
    return user;
  }
}
```

Slika 4.23 *Programski kod JWT strategije*

Također za korištenje strategije potrebno je definirati zaštitu koja koristi strategiju. Zaštita se definira pomoću *AuthGuard* funkcije implementirane od nestjs/passport NPM paketa. Takva zaštita može se koristiti kao dekorator pomoću predefiniranog Nest.js dekoratora *UseGuards* u definiranim rutama u kontrolerima. Slika 4.24 prikazuje programski kod JWT zaštite.

```
export class JwtGuard extends AuthGuard('jwt') {
  constructor() {
    super();
  }
}
```

Slika 4.24 *Programski kod JWT zaštite*

4.3.2.3 DTO

Kao što je prije objašnjeno, DTO definira strukturu podataka koju može primiti. Korištenjem DTO-a na svakoj ruti omogućuje validaciju i verifikaciju podataka koji stižu na pozadinsku aplikaciju. Slika 4.25 prikazuje programski kod CreateRewardDto DTO-a.

```
export class CreateRewardDto {
  /**
   * Reward name
   */
  @IsNotEmpty()
  @IsString()
  name: string;

  /**
   * Reward short description
   */
  @IsNotEmpty()
  @IsString()
  shortDescription: string;

  /**
   * Reward description
   */
  @IsNotEmpty()
  @IsString()
  description: string;

  /**
   * Reward contract address
   */
  @IsNotEmpty()
  @IsString()
  contractAddress: string;

  /**
   * Reward image url
   */
  @IsNotEmpty()
  @IsString()
  imageUrl: string;

  /**
   * Reward type
   */
  @IsNotEmpty()
  @IsString()
  type: string;
}
```

Slika 4.25 *Programski kod DTO-a na ruti upisa novog preuzimanja nagrade*

U slučaju da DTO nije važeći, tj. da je klijentska aplikacija poslala objekt koji ne spada u pravila definiranog DTO-a te rute, aplikacija će baciti grešku pod imenom *Bad Response*.

4.3.2.4 Rute pozadinske aplikacije

Pozadinska aplikacije sadrži par modula koji imaju neovisnu zadaću:

- Claim: definira POST rutu zaštićenu JWT-om za izradu zapisa u bazu kada student preuzme nagradu

- Reward: definira GET rute zaštićene JWT-om za dohvaćanje podataka o pojednim nagradama i za dohvaćanjem korisničkih dostupnih nagrada, također sadrži nezaštićenu GET rutu za dohvaćanje metapodataka za pojedinu nagradu
- User: definira JWT zaštićenu GET rutu za dohvaćanje trenutnog prijavljenog korisnika

Uz predefinirane rute sadrži i predefiniranu strukturu Success i Error klasa koje definiraju strukturu podataka koja se vraća na klijentsku aplikaciju. Slika 4.26 prikazuje programski kod klasa koje definiraju strukturu odgovora prema klijentskoj aplikaciji.

```
import { IResponse } from '../interfaces/response.interface';

You, last month | 1 author (You)
export class ResponseError implements IResponse {
  constructor(infoMessage: string, data?: any) {
    this.success = false;
    this.message = infoMessage;
    this.data = {
      code: data?.code ?? 500
      message: data?.messa var undefined
      meta: data?.meta ?? undefined,
      clientVersion: data?.clientVersion ?? undefined,
    };
  }
  message: string;
  data: any;
  errorMessage: any;
  error: any;
  success: boolean;
}

You, last month | 1 author (You)
export class ResponseSuccess implements IResponse {
  constructor(infoMessage: string, data?: any, notLog?: boolean) {
    this.success = true;
    this.message = infoMessage;
    this.data = data;
    if (!notLog) {
      try {
        const offuscateRequest = JSON.parse(JSON.stringify(data));
        if (offuscateRequest && offuscateRequest.token)
          offuscateRequest.token = '*******';
      } catch (error) {}
    }
  }
  message: string;
  data: any[];
  errorMessage: any;
  error: any;
  success: boolean;
}
```

Slika 4.26 *Programski kod ResponseError i ResponseSuccess klase*

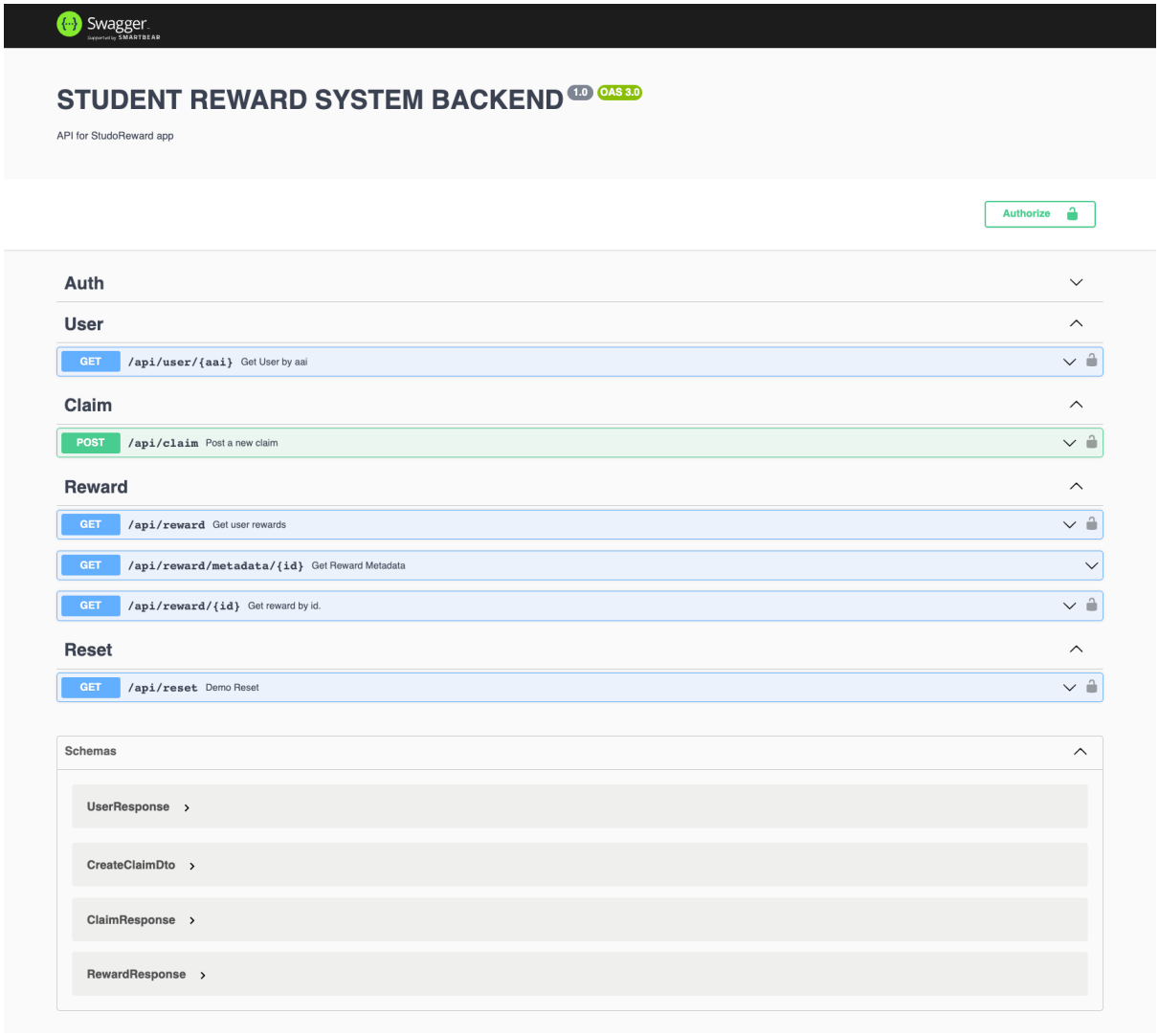
4.3.2.5 Swagger dokumentacija

Kao što je već prije navedeno, Swagger je predefinirana struktura dokumentacije koja izrađuje u konkretnom slučaju izradi stranicu na URL-u pozadinske aplikacije koja služi kao dokumentacija i pregled svih dostupnih ruta. Swagger se integrira u aplikaciju na način da se instalira Swagger NPM paket i pomoću predefiniranih Swagger funkcija izrađuje konfiguracija Swaggera. Slika 4.27 prikazuje programski kod konfiguracije Swaggera.

```
const swaggerConfig = new DocumentBuilder()
  .setTitle('STUDENT REWARD SYSTEM BACKEND')
  .setDescription('API for StudoReward app')
  .setVersion('1.0')
  .addBearerAuth(
    {
      type: 'http',
      scheme: 'bearer',
      bearerFormat: 'JWT',
      name: 'JWT',
      description: 'Enter JWT token',
      in: 'header',
    },
    'JWT-auth',
  )
  .build();

const document = SwaggerModule.createDocument(app, swaggerConfig);
SwaggerModule.setup('api', app, document);
```

Slika 4.27 *Programski kod konfiguracije Swaggera*

Nakon što se pokrene aplikacija, Swagger dokumentacija je vidljiva na URL-u pozadinske aplikacije sa sufiksom /api. Slika 4.28 prikazuje Swagger dokumentaciju dostupnu na poveznici pozadinske aplikacije.

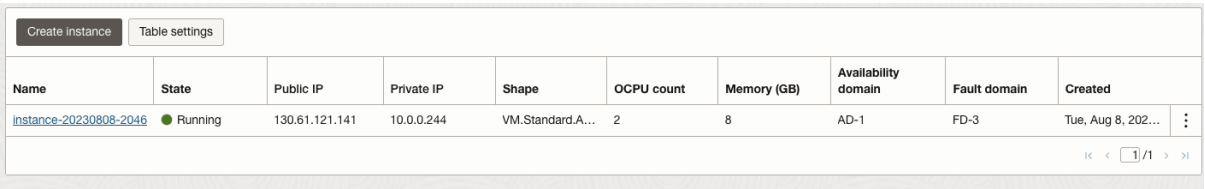


Slika 4.28 *Swagger dokumentacija pozadinske aplikacije*

Swagger također uz sam prikaz postojećih ruta prikazuje i uvid u rute koje su zaključane, parametre koje rute primaju i odgovore koje rute vraćaju.

4.4 Oracle Cloud instanca

Kao što je već prije navedeno, za objavu produkcijske verzije aplikacije koristit ćemo Oracle Cloud. Sve komponente aplikacije pokrenute su i vanjski dostupne pomoću instance. Slika 4.29 prikazuje pokrenutu instancu na Oracle Cloudu.



Slika 4.29 *pokrenuta instanca*

Zbog sigurnosti, direktni pristup instanci kao računalu trenutno ima samo jedno računalo sa definiranim SSH ključem. Aplikacije su nakon prevađanja izvršnog koga pokrenute pomoću PM2 procesnog menadžera koji omogućuje aplikacijama da ostanu pokrenute u pozadini sustava. Slika 4.30 prikazuje pokrenute aplikacije na Oracle instanci.

```
ubuntu@instance-20230808-2046:/var/www$ pm2 status
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
1	be	default	0.0.1	fork	204261	27h	0	online	0%	104.0mb	ubuntu	disabled
2	fe	default	0.39.4	fork	7548	14D	0	online	0%	65.6mb	ubuntu	disabled

Slika 4.30 *Pokrenute aplikacije*

Baza podataka je zbog jednostavnosti pristupa i instalacije pokrenuta u Docker kontejneru. Slika 4.31 prikazuje pokrenutu Postgres bazu u kontejneru.

```
ubuntu@instance-20230808-2046:/var/www$ docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS       PORTS                                          NAMES
b46cf2acbefd   postgres:13    "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   0.0.0.0:5434->5432/tcp, :::5434->5432/tcp      student-reward-system-be-dev-db-1
```

Slika 4.31 *Pokrenuta baza podatka u kontejneru*

Kako bi aplikacija bila pristupačna iz vanjskih izvora, instanca koristi već prije objašnjeni Nginx. Nakon instalacije Nginxa i otvaranja potrebnih HTTP portova, izrađena je konfiguracija aplikacije u *sites-enabled* i *sites-available* direktorijima. Slika 4.32 prikazuje Nginx konfiguraciju aplikacije.

```
#The Nginx server instance
server{
    server_name studoreward.xyz;
    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        # location /overview {
        #     proxy_pass http://127.0.0.1:3000$request_uri;
        #     proxy_redirect off;
        # }
    }

    location /api {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_pass http://127.0.0.1:3001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        # location /overview {
        #     proxy_pass http://127.0.0.1:3001$request_uri;
        #     proxy_redirect off;
        # }
    }
}

server{
    if ($host = studoreward.xyz) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name studoreward.xyz;
    return 404; # managed by Certbot
}
```

Slika 4.32 *studoreward.xyz Nginx konfiguracija*

4.4.1 Domena

Domena aplikacije kupljena je na platformi Namecheap. Kako bi domena bila vidljiva na IP adresu Oracle Cloud instance potrebno je navesti u opcijama kupljene domene IP adresu instance. Opcija koja se treba konfigurirati zove se „A Record“ koji označava IP adresu domene. Slika 4.33 prikazuje konfiguraciju domene na Namecheapu.

Type	Host	Value	TTL
A Record	@	130.61.121.141	Automatic

ADD NEW RECORD

Slika 4.33 *konfiguracija domene na Namecheapu*

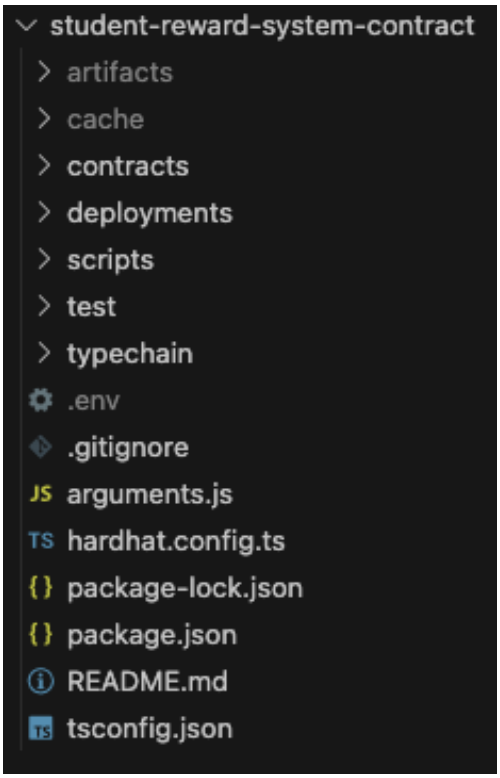
4.5 Pametni ugovor

Kao što je već objašnjeno, pametni ugovor je program na lancu blokova koji rade unaprijed određenu funkciju. U kontekstu StudoReward aplikacije pametni ugovor služi kao kolekcija nagrada, u konkretnom slučaju kao kolekcija NFT-a. Slika 4.34 prikazuje programski kod pametnog ugovara Reward.

```
contract Reward is ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIdCounter;

    string private baseTokenURI;
    uint256 private maxTokens;

    constructor(
        string memory _baseTokenURI,
        uint256 _maxTokens
    ) ERC721("Reward", "RWRD") {
        baseTokenURI = _baseTokenURI;
        maxTokens = _maxTokens;

        _tokenIdCounter.increment();
    }

    function setBaseTokenURI(
        string memory _newBaseTokenURI
    ) external onlyOwner {
        baseTokenURI = _newBaseTokenURI;
    }

    function claim() external payable {
        require(_tokenIdCounter.current() < maxTokens, "All tokens claimed");

        uint256 tokenId = _tokenIdCounter.current();
        _mint(msg.sender, tokenId);
        _setTokenURI(
            tokenId,
            string(abi.encodePacked(baseTokenURI, Strings.toString(tokenId)))
        );

        _tokenIdCounter.increment();
    }
}
```

Slika 4.34 *Programski kod pametnog ugovora nagrade*

Reward, pametni ugovor nagrade StudoReward aplikacije, nasljeđuje Openzeppelinove defnirane pametne ugovore i funkcionalnosti poput Ownable, koji omogućuje vlasništvo nad ugovorom u slučaju promjene meta podataka nagrade ili ručnog dodjeljivanja nagrada. Navedeni pametni ugovor iskorištava se za jedan tip nagrade na aplikaciji i u testnoj aplikaciji je objavljen na Polygon Mumbai testnetu uz pomoć već prije navedenog Hardhat razvojnog okruženja. Slika 4.35 prikazuje Hardhat repozitorij pametnog ugovora.



Slika 4.35 *Hardhat repozitoj pametnog ugovora*

5. ZAKLJUČAK

Korištenje Next.js-a, Nest.js-a i pametnih ugovora za stvaranje internetske stranice koja nagrađuje postignuća učenika odluka je utemeljena na inovativnosti i praktičnosti. Next.js sa svojim stranicama koje prikazuje poslužitelj i jednostavnim usmjeravanjem, osigurava brzo i privlačno korisničko iskustvo, ključno za obrazovnu platformu usmjerenu na angažman studenata. Njegova kompatibilnost s TypeScriptom dodaje sloj sigurnosti tipa, olakšavajući robustan razvoj koda i učinkovitu suradnju. U međuvremenu, Nest.js sa svojom modularnom arhitekturom i uvođenjem ovisnosti, pruža čvrstu osnovu za pozadinski razvoj. Njegova integracija s TypeScriptom i ugrađena podrška za dekoratere čine razvijanje API krajnjih točaka, rukovanje autentifikacijom i upravljanje protokom podataka lakim. Pametni ugovori, omogućeni tehnologijom lanca blokova, donose transparentnost i sigurnost u sustav nagrađivanja. Oni jamče da su postignuća učenika provjerljiva, zaštićena od neovlaštenih promjena i da automatski pokreću nagrade bez posredničke intervencije. Ovaj paket tehnologija usklađen je kako bi izradio rješenje kao internetsku aplikaciju koje ne samo da slavi i potiče uspjeh učenika kroz interakcije i glatka korisnička iskustva, već također osigurava vjerodostojnost i integritet mehanizma nagrađivanja. Koristeći Next.js, Nest.js i pametne ugovore, obrazovna internetska stranica usklađuje tehnološku snagu s dizajnom usmjerenim na korisnika, redefinirajući način na koji se postignuća učenika prepoznaju i nagrađuju u digitalnom dobu.

LITERATURA

[1] „Duolingo for schools“, 04.09.2023. [Online]. Dostupno na: https://schools.duolingo.com/

[2] „Khan Academy“, 04.09.2023. [Online]. Dostupno na: https://www.khanacademy.org/,

[3] „Codecademy. 04.09.2023. [Online]. Dostupno na: https://www.codecademy.com/,

[4] „Next.js“, 04.09.2023. [Online]. Dostupno na: https://nextjs.org/,

[5] „Next.js vs React: The Difference and Which Framework to Choose “, 04.09.2023. [Online]. Dostupno na: https://ninetailed.io/blog/next-js-vs-react/,

[6] „From Single-Page Application to Server-Side Rendering “, 04.09.2023. [Online]. Dostupno na: https://medium.com/zattoo_tech/from-single-page-application-to-server-side-rendering-82173d42e2f8

[7] „Nest.js“, 04.09.2023. [Online]. Dostupno na: https://docs.nestjs.com/,

[8] „Design Patterns Explained – Dependency Injection“, 04.09.2023. [Online]. Dostupno na: https://stackify.com/dependency-injection/

[9] „Validation“, 04.09.2023. [Online]. Dostupno na: https://docs.nestjs.com/techniques/validation

[10] „PostgreSQL Documentation“, 04.09.2023. [Online]. Dostupno na: https://www.postgresql.org/docs/current/,

[11] C. Dannen, "Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners", 2022.

[12] „Polygon University“, 04.09.2023. [Online]. Dostupno na: https://university.polygon.technology/,

[13] „Introduction to smart contracts“, 04.09.2023. [Online]. Dostupno na: https://ethereum.org/en/smart-contracts/#introduction-to-smart-contracts,

[14] „Learn Metamask“, 04.09.2023. [Online]. Dostupno na: https://learn.metamask.io/overview,

[15] „Docker Documentation “, 2023. [Online]. Dostupno na: https://docs.docker.com/,

[16] „Oracle Cloud Infrastructure“, 04.09.2023. [Online]. Dostupno na: https://www.oracle.com/cloud/

[17] „Certbot Instructions “, 04.09.2023. [Online]. Dostupno na: https://certbot.eff.org/instructions

POPIS I OPIS UPOTRJEBLJENIH KRATICA

- HTML (engl. HTML – Hypertext Markup Language) - Standardni jezik za označavanje dokumenata dizajniranih za prikaz u internetskih pregledniku
- API (engl. Application programming interface) – Aplikacijsko programsko sučelje. Skup potprograma, gotovih funkcija i protokola koje programer može koristiti za razvijanje vlastitih programa.
- HTTP (engl. Hypertext Transfer Protocol) – Mrežni protokol za prijenos podataka kojeg koriste izvorišno i odredišno računalo za uspostavu komunikacije.
- HTTPS (engl. Hypertext Transfer Protocol Secure) – Mrežni protokol za prijenos podataka kojeg koriste izvorišno i odredišno računalo za uspostavu komunikacije sa enkripcijom podataka.
- CSS (engl. CSS - Cascading Style Sheets) - Jezik stilskog lista koji se koristi za opisivanje prezentacije dokumenta napisanog na označnom jeziku, kao što je HTML
- NPM (engl. NPM – Node Package Manager) - Upravitelj paketa za programski jezik JavaScript
- SSH (engl. SSH - Secure Shell Protocol) - Kriptografski mrežni protokol za sigurno upravljanje mrežnim uslugama preko nezaštićene mreže
- SQL (engl. SQL - Structured Query Language) - Jezik specifičan za domenu koji se koristi u programiranju i dizajniran je za upravljanje podacima koji se drže u sustavu za upravljanje relacijskom bazom podataka
- JWT (engl. JWT – Jason Web Token) - Predloženi internetski standard za stvaranje podataka s potpisom
- SEO (engl. Search Engine Optimization) - proces poboljšanja kvalitete i količine prometa na internetsku stranicu ili internetsku stranicu s tražilica
- NFT (engl. Non-Fungible Token) - su jedinstveni kriptografski tokeni koji postoje na lancu blokova i ne mogu se replicirati
- SSR (engl. Server-Side Rendering) – način učitavanja sadržaja stranice gdje se generira sadržaj na poslužitelju, a zatim šalje pregledniku
- SPA (engl. Single-Page Application) – vrsta internetske aplikacije koja je u interakciji s korisnikom dinamičko učitava trenutne stranice umjesto učitavanja cijelih novih stranica s poslužitelja
- SSG (engl. Static Site Generation) – tehnika za učitavanje internetskih stranica unaprijed, tijekom procesa izrade

- IDE (engl. Integrated Development Environment) – aplikacija koja programerima pomaže u učinkovitom pisanju programskom koda
- CSR (engl. Client-Side Rendering) – tehnika prikazivanje stranica izravno u pregledniku s JavaScriptom
- IoC (engl. Inversion of Control) – princip dizajna koji omogućuje manje povezanih klasa i stoga ih je lakše testirati i održavati
- DTO (engl. Dana Transfer Object) – objekt koji prenosi podatke između procesa
- ORM (engl. Object Relational Mapping) – tehnika koja se koristi za stvaranje "mosta" između objektno orijentiranih programa i, u većini slučajeva, relacijskih baza podataka
- ACID (engl. atomicity, consistency, isolation, and durability) – akronim koji se odnosi na skup od 4 ključna svojstva koja definiraju transakciju: atomičnost, dosljednost, izolaciju i trajnost
- RDBMS (engl. relational database management system) – program koji se koristi za stvaranje, ažuriranje i upravljanje relacijskim bazama podataka
- ERC721 – standardi za nezamjenjive tokene, NFT-eve
- ERC20 – standard za stvaranje tokena koji se koristi u Ethereum lancu blokova i virtualnim strojevima
- A1 – Oraclovo rješenje za instance u oblaku koje pružaju determinističku izvedbu, linearnu skalabilnost i sigurnu arhitekturu s najboljom cijenom i učinkom na tržištu.
- SSL (engl. Secure Sockets Layer) – standardna tehnologija za osiguravanje internetske veze šifriranjem podataka koji se šalju između internetskih stranica i preglednika
- TTL (engl. Transport Layer Security) – kriptografski protokol dizajniran za pružanje sigurnosti komunikacije preko računalne mreže
- POST, GET – HTTP metode za slanje i dohvaćanje podataka sa poslužitelja
- ENV (skraćeno od engl. environment) – datoteka koja sadrži lokalne varijable aplikacije
- URL (engl. Uniform Resource Locator) – jedinstveni identifikator koji se koristi za lociranje izvora na Internetu
- CLI (engl. Command-Line Interface) – interakcija s uređajem ili računalnim programom s naredbama preko terminala
- PM2 – upravitelj procesa za Node.js aplikacije s ugrađenim balanserom opterećenja, omogućuje da zauvijek održi aplikacije na životu u pozadini.

SAŽETAK

Diplomski rad objašnjava razloge i proces razvoja aplikacije za izdavanje i studentskih nagrada na blockchainu. Navodi tehnologije korištene za izradu aplikacije. Kratko objašnjava princip rada tehnologija i razloga korištenja navedenih tehnologija. Detaljno je objašnjen princip strukture programskog koda kao i načine korištenja aplikacije u oblaku. Posebna pozornost posvećena je na sigurnost i moderne metode korištene za izradu aplikacije. Prikazani su najvažniji elementi klijentskog dijela aplikacije. Detaljno je objašnjena struktura programskog koda klijentske i pozadinske aplikacije. Konačno, objašnjen je način rada pametnog ugovora na lancu blokova.

Ključne riječi: internetska aplikacija, klijent i poslužitelj, lanac blokova, NFT, sigurnost

ABSTRACT

An application for issuing and tracking student awards on the blockchain

The thesis explains the rationale and process of developing applications for issuing and student awards on the blockchain. Lists the technologies used to create the applications. It briefly explains the working principle of the technology and the reasons for using the mentioned technologies. The principle of the structure of the program code as well as the way of using applications in the cloud is explained in detail. Special attention is paid to security and modern methods used to create applications. The most important elements of the client application are shown. The structure of the client and background application code is explained in detail. Finally, the working method of the smart contract on the block chain is explained.

Keywords: blockchain, client and server, NFT, security, web application

ŽIVOTOPIS

Mihael Ištvan, rođen 5. studenog 1999 u Koprivnici. Godine 2018. upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek pri Sveučilištu Josipa Jurja Strossmayera u Osijeku. Za vrijeme druge i treće godine studija obavlja poslove vanjskog suradnika kao internetski programer u poduzetništvima u inozemstvu. Nakon upisa diplomskog studija modeliranja i dizajna programskih sustava 2021. godine nastavlja karijeru kao internetski programer i programer lanca blokova u Hrvatskim poduzećima.

Mihael Ištvan: _____

DODATAK

Poveznica za pregled aplikacije dostupna je na studoreward.xyz, a izvorni kodovi dostupni su na sljedećim poveznicama:

- Klijentska aplikacija: https://github.com/blekso/student-reward-system-fe
- Pozadinska aplikacija: https://github.com/blekso/student-reward-system-be
- Pametni ugovor: https://github.com/blekso/student-reward-system-contract
- Adresa pametnog ugovora Cinestar nagrade: 0x63de1209f9470866Da4eC357c78F22eC69aAA1a7
- Adresa pametnog ugovora Eurospin nagrade: 0xE0f001e2E5778B62C18020F9278697D47F3366d5