Sustav za kupnju karata i njihovo spremanje u blockchain koristeći pametne ugovore

Cmrk, Antonio

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

Sveučilišni diplomski studij Računarstvo

SUSTAV ZA KUPNJU KARATA I NJIHOVO SPREMANJE U BLOCKCHAIN KORISTEĆI PAMETNE UGOVORE

Diplomski rad

Antonio Cmrk

Osijek, 2024. godina.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Antonio Cmrk
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1273R, 07.10.2022.
JMBAG:	0165081373
Mentor:	izv. prof. dr. sc. Mirko Köhler
Sumentor:	Miljenko Švarcmajer, univ. mag. ing. comp.
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 1:	Miljenko Švarcmajer, univ. mag. ing. comp.
Član Povjerenstva 2:	izv. prof. dr. sc. Zdravko Krpić
Naslov diplomskog rada:	Sustav za kupnju karata i njihovo spremanje u blockchain koristeći pametne ugovore
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Rezervirao Antonio Cmrk. Potrebno je napisati React komponente za pregled karata, odabir sjedala i postupak kupnje. Nakon uspješne kupnje, potrebno je zapisati informacije o vlasniku karte u pametni ugovor na blockchainu. Također treba spremiti informacije o vlasnicima karata, poput imena, ugovornih adresa i drugih relevantnih podataka, u pametnom ugovoru. Koristiti Solidity jezik za pisanje pametnog ugovora. Zatim prevesti i implementirati ugovor na Ethereum testnet mreži pomoću alata poput Remix IDE ili truffle framework-a.
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	16.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	4.10.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	17.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 17.10.2024.

Ime i prezime Pristupnika:	Antonio Cmrk
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1273R, 07.10.2022.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Sustav za kupnju karata i njihovo spremanje u blockchain koristeći pametne ugovore**

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

i sumentora Miljenko Švarcmajer, univ. mag. ing. comp.

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 pristupnika:

SADRŽAJ

1. UVOD

Tehnologija ulančanih blokova (eng. blockchain) se koristi sve više jer pruža sigurnost, transparentnost i decentralizaciju. Omogućava izgradnju sustava u kojem se podaci dijele među brojnim sudionicima, podaci se decentraliziraju kroz mrežu računala, te svako računalo predstavlja čvor i tako eliminiraju potrebu za centralnim autoritetom. Svaki čvor u blockchain mreži može usporediti hash vrijednosti, jer se hash svakog bloka temelji na prethodnom, sve do bloka 0, što omogućuje provjeru ispravnosti i otkrivanje neovlaštenih promjena podataka. Također, sve više se koriste i pametni ugovori. To su ugovori koji se sami izvršavaju kada su predefinirani uvjeti ispunjeni. Implementiraju se na blockchain mreži te su sigurni i transparentni. To omogućava automatizaciju, smanjuje troškove i vrijeme potrebno za transakcije zbog toga što nema potrebe za posrednicima ili centraliziranim autoritetom.

Cilj ovoga diplomskog rada je napraviti web aplikaciju u kojoj će korisnik moći intuitivno i jednostavno kupiti kartu za gledanje filma u kinu. Korisnik će moći odabrati kino u koje želi ići gledati film, zatim će se prikazati dostupni filmovi za koje može u tom trenutku kupiti kartu. Nakon što odabere film koji želi gledati, odabire sjedala koja još nisu zauzeta. Na kraju, kada je korisnik odradio sve prijašnje korake, podaci o vlasniku karata će biti pohranjeni pametnim ugovorom na blockchain. Time se osigurava da se podaci ne mogu krivotvoriti i da su autentični.

Rad se sastoji od šest poglavlja. U drugom poglavlju pregledana su rješenja slična temi ovog rada. Prikazano je kako se kupuju karte za kino preko web stranice u Hrvatskoj te u stranim zemljama i prikazan je jedan rad sličan ovom koji također koristi React i pametne ugovore za kupnju karata za kino. U trećem poglavlju objašnjene su tehnologije korištene za izradu web aplikacije ovog diplomskog rada. Za poslužiteljsku stranu aplikacije je korišten ASP.NET Core razvojni okvir u kombinaciji s programskim jezikom C# i SQL server express bazom podataka. Za korisničko sučelje korišten je React pisan u TypeScript programskom jeziku, uz Tailwind CSS za stiliziranje. Pametni ugovori su napisani u Solidity programskom jeziku i bit će implementirani na Sepolia Testnet mreži. Nakon toga, u četvrtom poglavlju objašnjeno je kreiranje programskog dijela diplomskog rada. Detaljno je opisano kako je kreiran svaki dio aplikacije te kako međusobno komuniciraju. U petom poglavlju objašnjen je cijeli tok aplikacije te kako radi i kako se koristi. Na kraju je dan zaključak i osvrt na cjelokupni diplomski rad.

1.1. Zadatak diplomskog rada

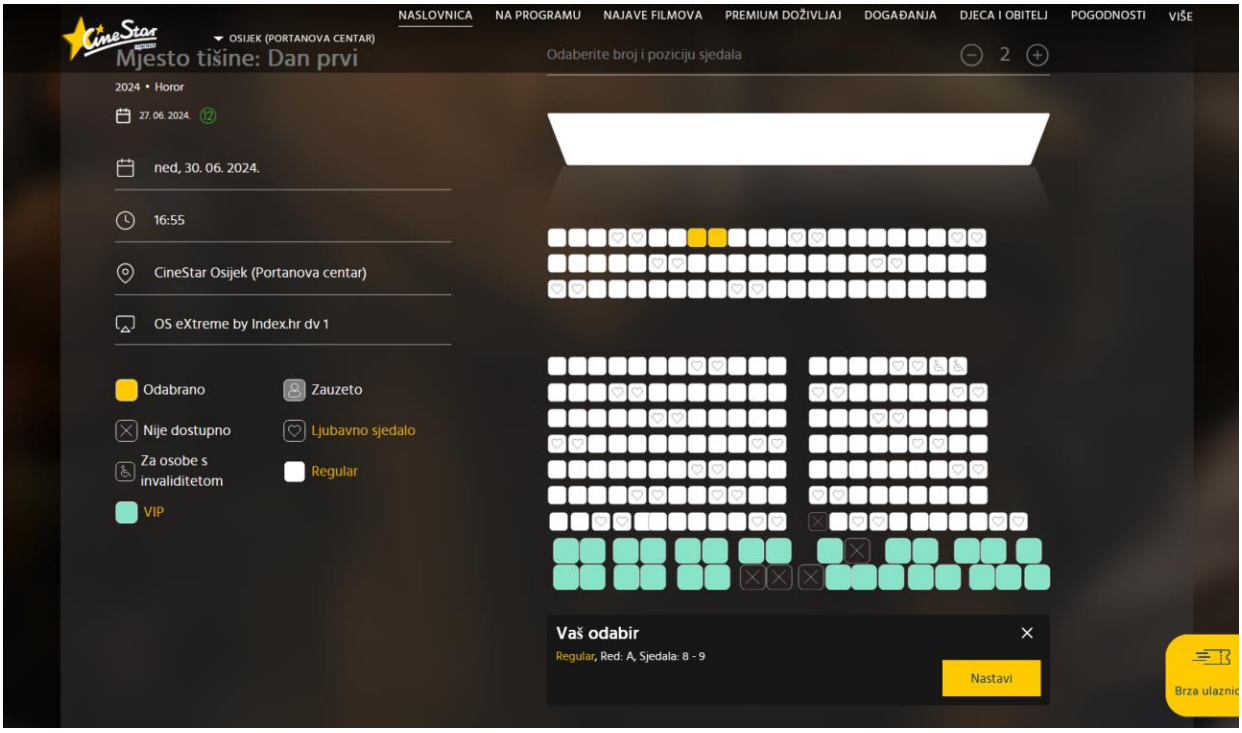
Potrebno je napisati React komponente za pregled karata, odabir sjedala i postupak kupnje. Nakon uspješne kupnje, potrebno je zapisati informacije o vlasniku karte u pametni ugovor na blockchainu. Također treba spremiti informacije o vlasnicima karata, poput imena, ugovornih adresa i drugih relevantnih podataka, u pametnom ugovoru. Koristiti Solidity jezik za pisanje pametnog ugovora. Zatim prevesti i implementirati ugovor na Ethereum Testnet mreži pomoću alata poput Remix IDE ili Truffle framework-a.

2. PREGLED PODRUČJA TEME

U ovom poglavlju bit će opisana slična postojeća rješenja koja su dostupna korisnicima. Kako je zadatak ovog diplomskog rada izraditi web aplikaciju za kupovanje karata, odabir sjedala i zapisivanje podataka o kupnji u pametni ugovor, pregledat će se neke web aplikacije za kupnju karata. Opisat će se web portal za kupovanje karata u Hrvatskoj i inozemstvu. Za Hrvatsku je opisana kupnja karata na web aplikaciji CineStar kina, a za inozemstvo kupnja karata na web aplikaciji Kino Rotterdam. Opisano je još i slično rješenje „CinemaTicket-Blockchain“. To je aplikacija koja također sprema podatke o kupnji karata za kino na Sepolia Testnet mrežu. Sepolia Testnet je testna Ethereum mreža koja služi za testiranje pametnih ugovora prije nego što se implementiraju na glavnu (eng. Mainnet) mrežu. Za transakcije na Sepolia Testnet mreži koristi se kriptovaluta SepoliaETH koja nema stvarnu vrijednost, nego služi samo za testiranje.

2.1. CineStar Osijek

CineStar web aplikacija je portal za kupnju karata za projekcije filmova. Kao primjer, korištena je web aplikacija CineStar Osijek. Na početnoj stranici prikazani su istaknuti i aktualni filmovi. Sadrži stranice „Na programu“ gdje se nalaze svi filmovi koji će se prikazivati u sljedećih nekoliko tjedana, „Najave filmova“ gdje su prikazani filmovi koji će se prikazivati u kinu s kratkim opisom i trailer videom, te „Premium doživljaj“, „Događanja“, „Djeca i obitelj“ i „Pogodnosti“. Prilikom odabira željenog filma, korisnik ima uvid u ime, kratak opis, trajanje, ograničenje godina te žanr filma. Nakon što je na stranici odabran film koji se želi pogledati, odabire se dan i vrijeme kad će korisnik ići u kino pogledati film. Nakon što se odabere vrijeme i datum gledanja filma, odabiru se mjesta, koja nisu već zauzeta od strane drugih korisnika, gdje korisnik želi sjediti u dvorani tijekom projekcije filma, što je prikazano slikom 2.1.

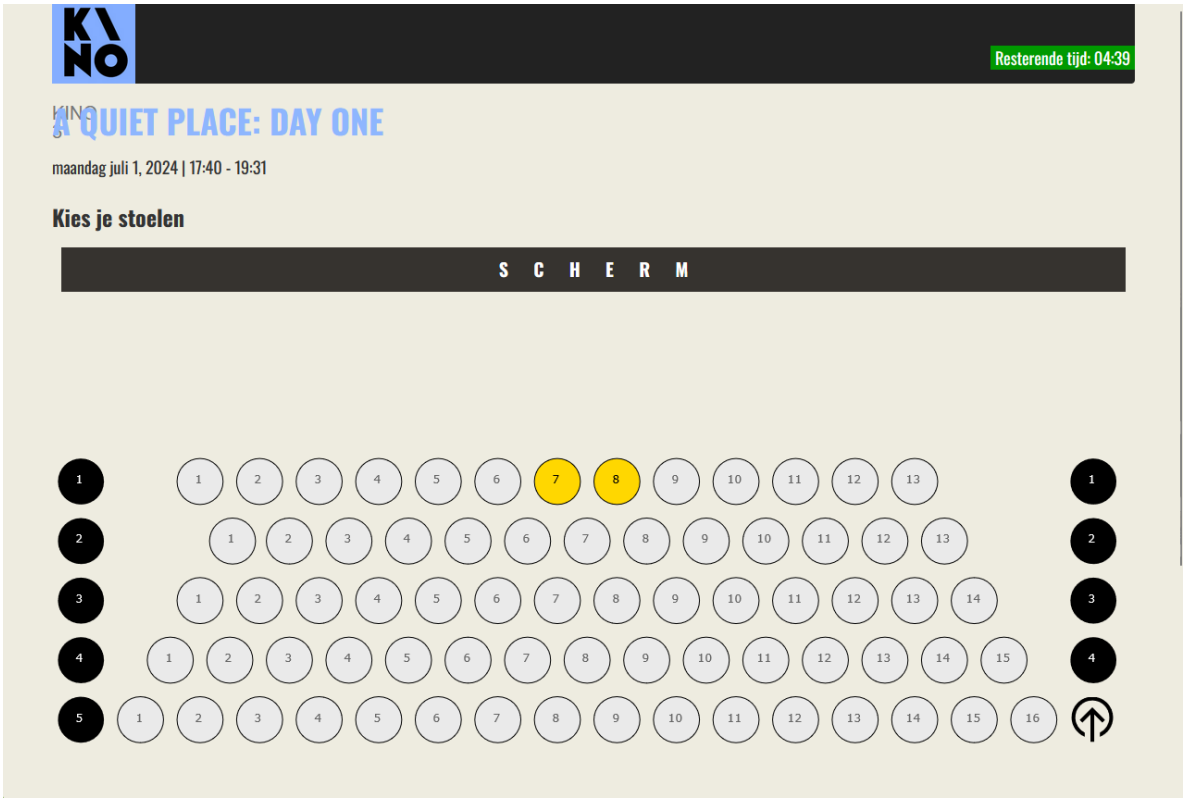


Sl. 2.1. Odabir mjesta sjedenja u dvorani za projekciju filma [1]

Na kraju se prikažu podaci o kupnji, vremenu, mjestu, dvorani, cijeni i slično, te korisnik potvrđuje odabir i plaća karte. Cijene karata za 2D projekciju filma iznose od 5 € do 10 €, dok za 3D od 7 € do 15 €. Web aplikacija pruža i opciju brze ulaznice. Korisnik stisne na tipku za brzu ulaznicu, odabere film, datum i vrijeme te tako brže dođe do koraka odabira mjesta u dvorani za projekciju filma [1].

2.2. Kino Rotterdam

Kino Rotterdam je web aplikacija za kino u Nizozemskoj u gradu Rotterdam. Radi na sličan način kao i već opisano kino. Početna stranica prikazuje istaknute filmove. Kao i kod web aplikacije CineStar, u ovoj aplikaciji kada se odabere određeni film otvara se stranica za odabir datuma i vremena projekcija filma te se uz to nalaze trailer i kratak opis filma. Nakon što se odabere film, potrebno je unijeti email adresu i nakon toga odabrati sjedala za koje se želi kupiti karte. Na slici 2.2. prikazan je odabir sjedala u web aplikaciji za ovo kino.

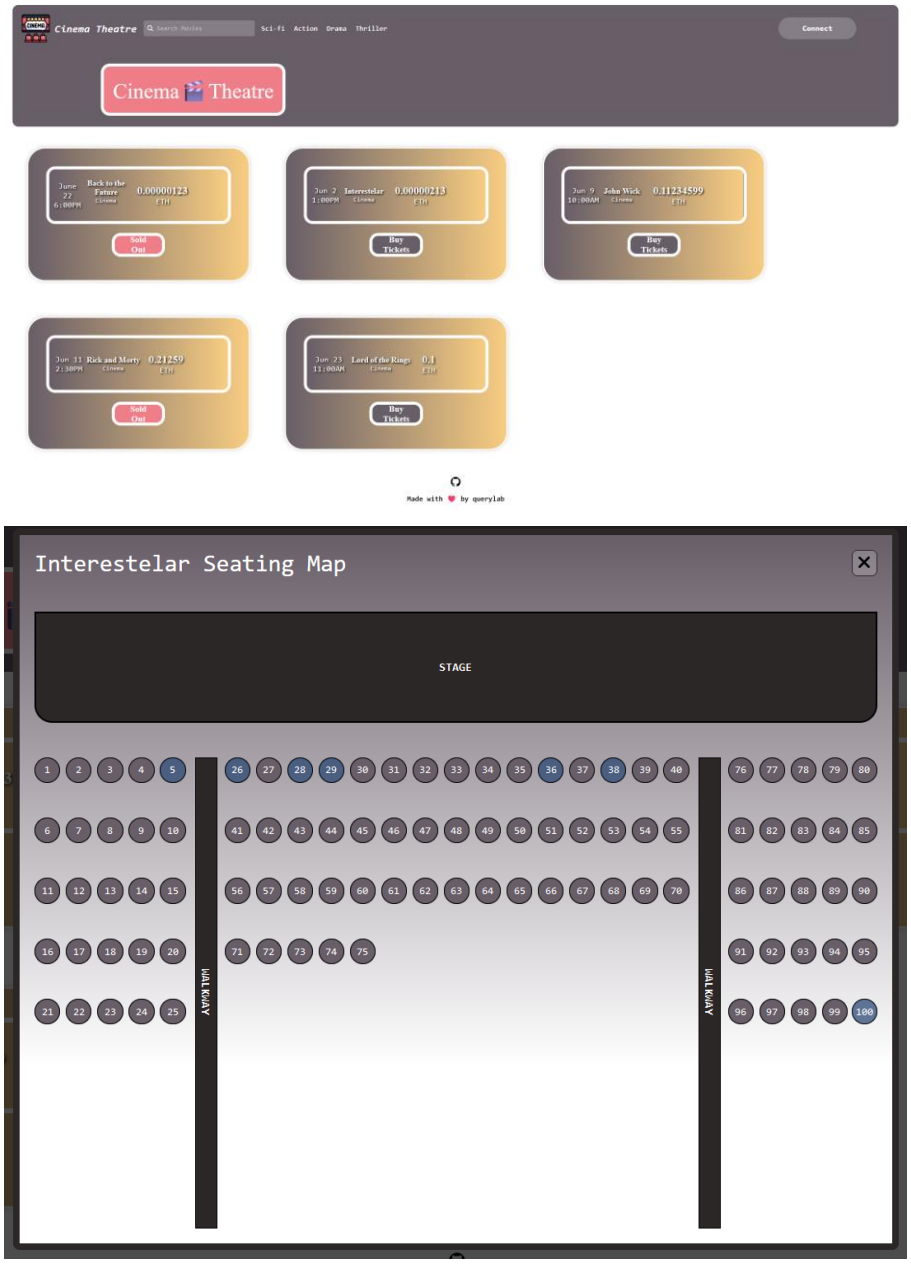


Sl. 2.2. Prikaz odabiranja sjedala za projekciju filma [2]

Nakon što su sjedala odabrana, sljedeći korak je odabir tipa karte. Ponuđeno je nekoliko različitih tipova karata kao što su regularna, studentska, za djecu itd. Cijene 2D projekcija su između 10 i 20 eura, ovisno o tipu karte [2].

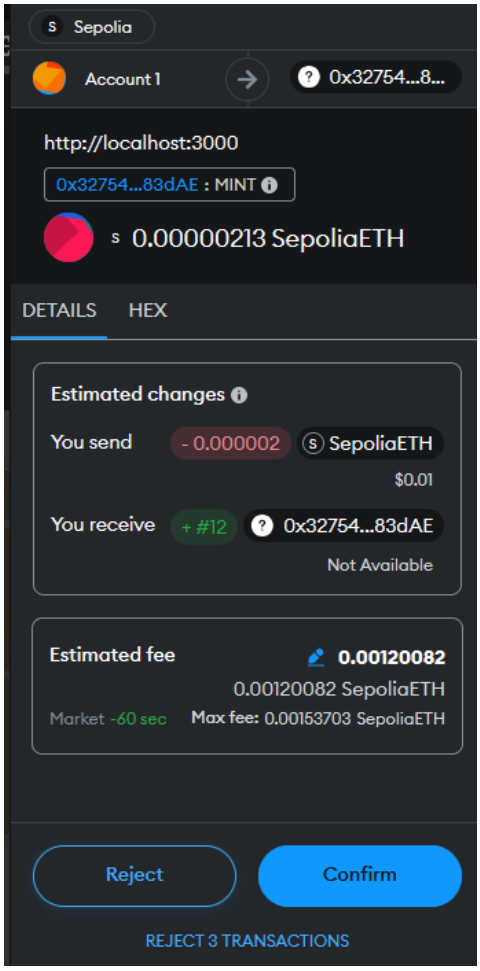
2.3. CinemaTicket-Blockchain

CinemaTicket-Blockchain je aplikacija čiji je izvorni kod na GitHub-u. To je aplikacija za prodaju karata koja koristi blockchain tehnologiju. Aplikacija pruža mogućnost odabiranja filma, mjesta i kupnje ulaznica za kino. Kada se odabere film prikazuje se stranica gdje se odabire mjesto u kinu. Odabir filma i sjedala prikazan je na slici 2.3.



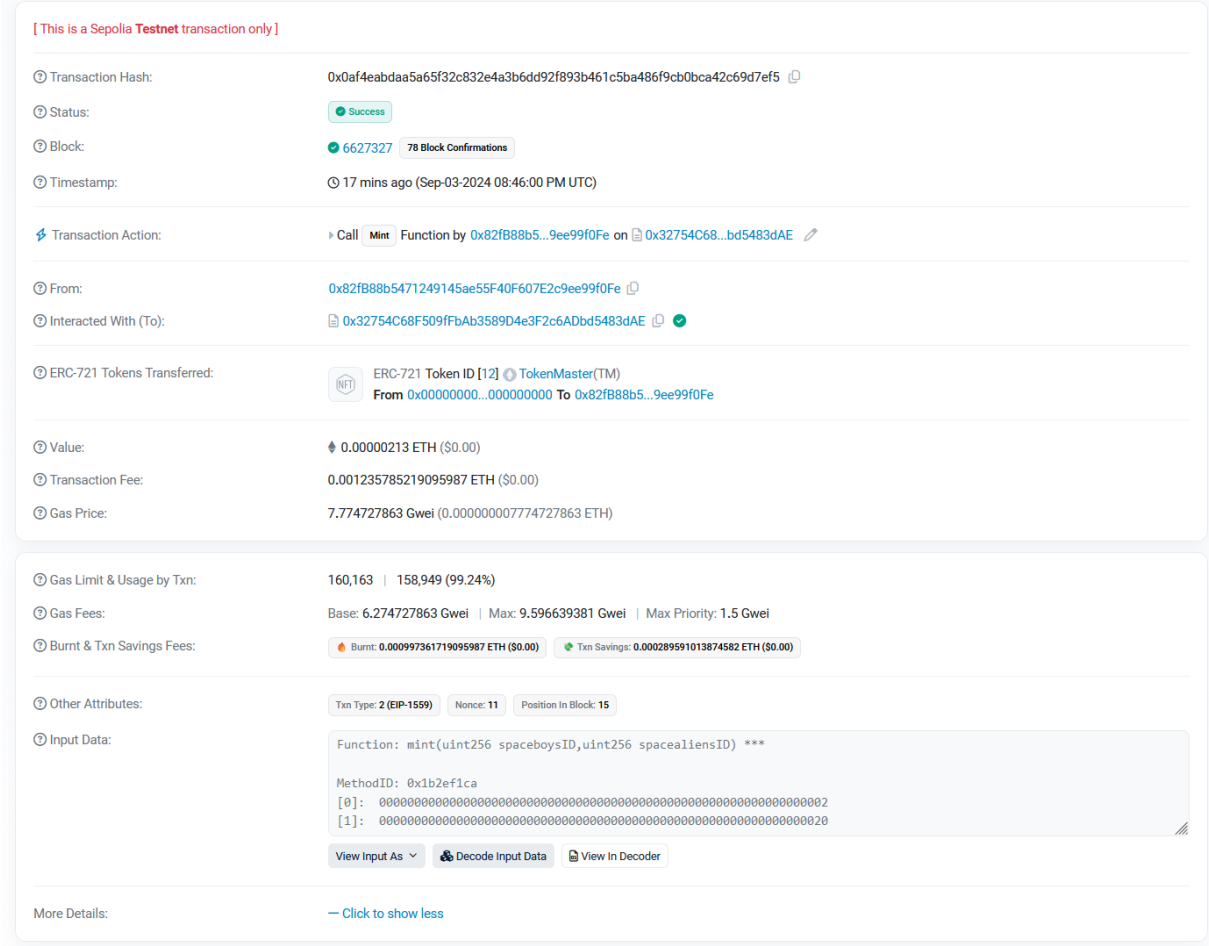
Sl. 2.3. Odabir filma i sjedala [3]

Ako se spojio MetaMask novčanik na dugme „Connect“ na početnoj stranici, moguće je kupiti kartu. Na slici 2.4. je prikazan upit na transakciju u MetaMask proširenju za internet preglednik.



Sl. 2.4. Prikaz uputa za transakciju

Ako transakcija bude uspješna, može se vidjeti na Sepolia Testnet mreži. Uspješna transakcija prikazana je slikom 2.5.



Sl. 2.5. Prikaz uspješne transakcije na Sepolia Testnet mreži [22]

CinemaTicket-Blockchain aplikacija je izuzetno slična aplikaciji ovog diplomskog rada. Za korisničko sučelje koristi ReactJS, pametni ugovor je napisan u Solidity programskom jeziku i implementiran na Sepolia Testnet. Za kompajliranje, implementaciju, testiranje i otklanjanje pogrešaka u tom projektu korišten je Hardhat. Hardhat je razvojno okruženje za Ethereum pametne ugovore. Ima lokalno testiranje, Solidity kompilaciju i jednostavnu implementaciju ugovora. Za sigurnosne transakcije korišten je MetaMask novčanik [3, 4].

3. KORIŠTENI ALATI I TEHNOLOGIJE

Ovaj diplomski rad, odnosno web aplikacija, rađena je u raznim alatima i tehnologijama kako bi aplikacija bila što bolja, optimiziranija i lakša za izradu i korištenje. Svaki dio rađen je u posebnoj tehnologiji te je kasnije sve povezano. Za bazu podataka korištena je SQL server express baza podataka. Za poslužiteljsku stranu korišten je ASP.NET Core u C# programskom jeziku za kreiranje web API-ja koji je implementiran u Visual Studio razvojnom okruženju. Za korisničko sučelje korištena je React biblioteka pisana u TypeScript programskom jeziku u Visual studio code razvojnom području. Solidity jezik je korišten za implementiranje pametnih ugovora na Ethereum blockchain platformu pisan unutar Remix IDE razvojnog područja.

3.1. Programski jezik C#

C# je objektno orijentiran programski jezik razvijen od strane Microsofta kao dio .NET platforme. Što znači da omogućava korištenje klasa, objekata, polimorfizma, enkapsulacije i nasljeđivanja. Prvi put je predstavljen 2000. godine. Koristi se za izradu raznih aplikacija kao što su desktop aplikacije, web aplikacije, video igre, cloud aplikacije, mikroservisi, mobilne aplikacije itd. Vrlo je svestran te je trenutno jedan on najkorištenijih jezika. Jedna od glavnih i najboljih značajki C#-a je ta što je jednostavan i čitljiv što rezultira u lakšem učenju, pisanju koda te održavanju. Sadrži standardnu biblioteku s mnogo funkcija koje olakšavaju pisanje koda, smanjuju broj linija cjelokupnog koda i grešaka. Ima ugrađen mehanizam za automatsko upravljanje memorije koji sam zauzima memoriju te kada ona više nije potrebna oslobađa ju. Podržava asinkrono programiranje što pomaže pri obradi mnogo zadataka istovremeno. Trenutna zadnja stabilna verzija C#-a je 12.0 koja je izašla u 2023[5, 6].

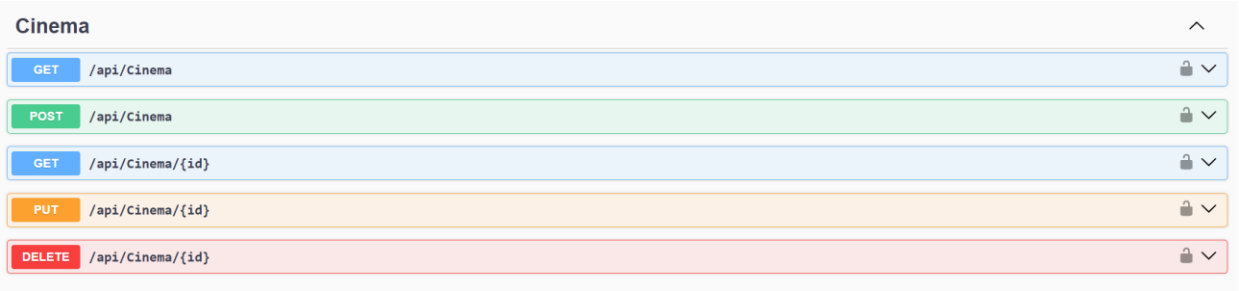
3.2. ASP.NET Core

ASP.NET Core je razvojni okvir za razvijanje programske podrške koji radi na više platforma. Koristi se za kreiranje modernih aplikacija, web aplikacija, servisa i mobilnih poslužitelja. Može raditi na Linuxu, Windowsima i macOS operativnim sustavima. Pomoću NuGet paketa moguće je uključiti samo one dijelove koji su potrebni te to čini sustav modularnim, smanjuje veličinu aplikacije i povećava performanse. Podržava razne razvojne mogućnosti kao što su MVC arhitektura kako bi se olakšalo razdvajanje logike i korisničkog

sučelja, Razor Pages kako bi se lagano mogla izraditi web stranica C# kodom, Blazor kako bi se C# programskim jezikom izrađivalo korisničko sučelje umjesto JavaScriptom, Web API kako bi se lagano ostvarila komunikacija između različitih sustava i aplikacija i ostalo. Otvorenog koda je te je izvorni kod dostupan na GitHub-u. Zbog svoje popularnosti postoji mnoštvo dokumentacije i uputa dostupnih online, uključujući i službenu Microsoft-ovu dokumentaciju. Najviše se koristi za izradu web aplikacija, API servisa, mikroservisa te velikih „enterprise“ aplikacija [7, 8].

3.3. Swagger UI

Swagger je skup pravila, specifikacija i alata koji pomažu pri dokumentiranju API-ja. Swagger UI služi kako bi dokumentacija bila što lakša, organiziranija i pristupačnija za pregledavanje i pretraživanje te se tako može značajno uštedjeti vrijeme. Korištenjem Swager UI dokumentaciju je lako razumjeti, čak i bez mnogo znanja o razvoju. Korištenjem postojećeg JSON ili YAML dokumenta kreira platformu koja poslaže metode get, post, put i delete. Kategorizira posebno metode za svaki kontroler te ih oboja određenim bojama: plava za get, žuta za put, zelena za post i crvena za delete. Primjer toga se može vidjeti na slici 3.1.



Sl. 3.1. Primjer dokumentacije u Swagger UI

Swagger pruža i mogućnost interakcije s poslužiteljskom stranom. Zbog toga nema potrebe za aplikacijama za testiranje API-ja kao što su Postman i slično. Vrlo je jednostavno za korištenje, klikne se na metodu koju se želi testirati, zatim dugme „Try it out“, te nakon toga dugme „Execute“ ako metoda ne zahtijeva nikakve parametre. Ako metoda zahtijeva parametre prvo je potrebno upisati podatke koje želimo testirati. Na primjer, ako se želi kreirati novi objekt, potrebno je unijeti atribute koje će taj objekt imati, ako nisu predefinirani [9].

3.4. SQL server express baza podataka

SQL server express baza podataka je Microsoft-ov besplatan SQL server za jednostavnu upotrebu za manje i srednje velike aplikacije. Najviše se koristi u privatne svrhe, u obrazovnim institucijama i u malim tvrtkama jer je dostupna besplatno te se pomoću nje kreiraju razvojne i testne okoline, male poslovne aplikacije itd. Idealna je za aplikacije koje ne zahtijevaju veliku bazu podataka i/ili intenzivne performanse. Jednostavno se instalira i konfigurira, podržava različite programske jezike i alate, lako se integrira s drugim Microsoft-ovim alatima i tehnologijama. Negativna strana je što postoje ograničenja resursa. Maksimalna veličina pojedine baze je 10 GB, maksimalno koristi 1 GB RAM-a za SQL Server Database Engine. Podržava najviše četiri jezgara procesora [10].

3.5. React

React je jedna od najpopularnijih biblioteka JavaScript-a za izradu korisničkog sučelja. Razvio ju je Facebook 2013. godine. Omogućava programerima da kreiraju kompleksne web stranice koje učinkovito upravljaju dinamičkim podacima i promjenama prikaza korisničkog sučelja. React koristi komponentni model što znači da su svi elementi podijeljeni u manje dijelove koji se zovu komponente. Komponenta može sadržavati niti jednu, jednu ili više drugih komponenti te se iste komponente mogu koristiti na različitim mjestima. Koristi virtualni DOM (Document Object Model) za optimalizaciju ažuriranja korisničkog sučelja te tako uspijeva na najefikasniji način ažurirati stvarni DOM. Za pisanje komponenti koristi se JSX (JavaScript XML), sintaktički dodatak koji omogućuje pisanje koda nalik HTML-u unutar JavaScript koda. React ga pretvara u običan JavaScript tijekom kompilacije. Tok podataka je jednosmjeran što znači da se podaci prenose iz roditelj komponente u dijete komponentu. Nudi mnoštvo različitih biblioteka i alata. Jedni od najpopularnijih su React Router i React Redux. React ima veliku zajednicu što rezultira velikim brojem uputa, dokumentacije, biblioteka te podrške [11].

3.6. Vite

Vite je alat za izradu korisničkog sučelja aplikacije. Prvenstveno je razvijen za rad s Vue.js, ali podržava i druge biblioteke i okvire. Vite je izašao 2020, a kreirao ga je Evan You, kreator Vue.js-a. Stekao je popularnost zbog svoje brzine, jednostavnosti i efikasnosti. Pruža

optimiziranu gradnju, minimizira izlazne datoteke, jednostavno se konfigurira u datoteci vite.config.js ili vite.config.ts. Dolazi s podrškom za razne alate što znatno olakšava rad i kvalitetu koda, ima velik ekosistem plugin-ova s različitima alatima i bibliotekama, pruža podršku za typescript bez dodatnih konfiguracija [12].

3.7. TypeScript

TypeScript je programski jezik razvijen od strane Microsofta, proširuje JavaScript omogućavanjem statičnog pisanja. Dizajniran je da bi se pri pisanju koda pronašle pogreške, a ne tijekom izvršavanja koda. Koristi se tako da se nakon definiranja varijable, iza njezinog imena napiše dvotočka i tip podataka. Ako je tip podataka kojeg želimo statički pisati objekt, potrebno je kreirati sučelje (eng. Interface) ili tip (eng. Type), navesti ime novog korisničkog tipa, te unutar njega napisati imena varijabli koje će objekt sadržavati i uz njih dvotočka pa jednostavan tip tih podataka. Ako jedan objekt sadrži drugi objekt kao jedan od svojih atributa, potrebno je za drugi objekt napisati korisnički tip te njega iskoristiti kao tip unutar tijela prvog objekta. Iako se prvobitno čini da je TypeScript lošiji od JavaScript-a jer zahtijeva pisanje više koda, u praksi se isplati jer se lakše i brže mogu ispraviti pogreške tijekom pisanja koda, dok kod JavaScripta tek tijekom izvršavanja [13].

3.8. Tailwind

Tailwind CSS je popularan CSS okvir za stiliziranje korisničkog sučelja. Za razliku od ostalih razvojnih okvira za stiliziranje korisničkog sučelja, Tailwind CSS ne nudi gotove predefinirane komponente, već nudi korisničke klase za svako CSS svojstvo. To znači da umjesto pisanja cijelog CSS koda i odabiranja imena klase, komponenti se samo pridruži predefinirano ime klase te se automatski primjeni CSS koji je predefiniran za to ime klase. Na primjer, ako se želi postaviti pozadina komponente u plavu boju, umjesto da se stavlja neko ime za klasu komponente i piše kod u CSS datoteci, samo se napiše bg-blue-500 za ime klase i komponenta poprima pozadinu plave boje bez dodatnog pisanja koda. Korištenjem Tailwinda smanjuje se broj datoteka u projektu, što znači da se lakše održava projekt, omogućava veliku prilagodljivost i kreiranje jedinstvenih dizajna bez ograničenja na unaprijed predefinirane komponente. Nakon kompilacije, brišu se neiskorišteni stilovi te se tako poboljšavaju

performanse web stranice. Lagano je prilagodljiv, u datoteci tailwind.config.js se mogu predefinirati boje, veličine, margina i još mnogi drugi stilovi za korisničko sučelje [14].

3.9. Blockchain

Tehnologija ulančanih blokova (eng. blockchain) je distribuirana baza podataka koja održava kontinuirano rastući popis uređenih zapisa, koji se nazivaju blokovi. Ti su blokovi povezani pomoću kriptografije. Svaki blok sadrži kriptografski hash prethodnog bloka, vremensku oznaku i podatke o transakciji [15]. Blockchain postaje poznat pojavom bitcoin-a 2008. godine, omogućava izgradnju sustava u kojem se podaci dijele među brojnim sudionicima i tako eliminiraju potrebu za centralnim autoritetom. Tradicionalni sustavi koriste centralni poslužitelj za upravljanje podacima, u blockchainu, podaci se decentraliziraju kroz mrežu računala, gdje svako računalo predstavlja čvor. Svaki čvor sadrži kopiju cijelog blockchaina, čime se uklanja potreba za centralnim autoritetom. Svaka transakcija blockchaina vidljiva je svim sudionicima mreže. Tako svaki čvor može provjeriti ispravnost i povijest svih transakcija. Jednom kad su podaci zabilježeni na blockchainu oni se ne mogu mijenjati ili brisati. To se postiže kriptografskim vezanjem svakog bloka s prethodnim, stvarajući lanac (blockchain). Bilo koji pokušaj promjene podataka u jednom bloku zahtijevao bi izmjenu svih sljedećih blokova, što je praktički nemoguće. Blockchain koristi kriptografske metode za osiguranje podataka. Svaka transakcija je digitalno potpisana, a blokovi su kriptografski povezani. Ova sigurnosna svojstva čine blockchain otpornim na neovlaštene izmjene i napade. Najpoznatija primjerna blockchain tehnologije su kriptovalute, koje omogućuju sigurne i transparentne financijske transakcije bez potrebe posrednika poput banaka. Blockchain se koristi još i u pametnim ugovorima, opskrbnim lancima, zdravstvu, digitalnom identitetu itd. Blockchain se još uvijek razvija i prilagođava te se očekuje da će blockchain igrati sve značajniju ulogu u digitalnom svijetu [16].

3.10. Ethereum

Ethereum je blockchain platforma. Poznat je po tome što omogućava razvoj decentraliziranih financijskih aplikacija, izvršavanje pametnih ugovora i kriptovaluti Etheru. Kreirao ga je Vitak Buterin 2015. godine. Jedna od ključnih značajki Ethereum-a su pametni ugovori. Oni omogućuju automatizaciju transakcija i procesa bez potrebe za posrednicima. Pametni ugovori se pišu u različitim programskim jezicima, najpopularniji je Solidity, te se zatim izvršavaju unutar

sigurnog okruženja EVM-a (eng. Ethereum Virtual Machine). Digitalna valuta koja se koristi na Ethereum platformi je Ether. Ether se koristi za plaćanje transakcijskih naknada i kao nagrada kod rudarenja. Ethereum se najčešće koristi za platforme za zajmove, trgovanje, financijske derivativne instrumente i ostale financijske usluge bez potrebe za posrednicima, NFT tržišta, praćenje i upravljanje opskrbnim lancima kako bi se osigurala transparentnost i autentičnost proizvoda, sigurnosno pohranjivanje i upravljanje digitalnim identitetom itd. [17, 18].

3.11. Pametni ugovori

Pametni ugovori su samostalno izvršivi ugovori. Potpisani su i pohranjeni na blockchain mreži. Automatski se izvrše, dokumentiraju ili kontroliraju kada su ispunjeni određeni uvjeti ugovora. Pametni ugovori se programiraju tako da automatski izvrše određene radnje kada se unaprijed definirani uvjeti zadovolje. To omogućava automatizaciju. Svaka promjena i transakcija se bilježi na blockchain-u, koriste se kriptografske metode te to sprječava manipulacije i osigurava transparentnost. Ugovori se ne mogu promijeniti nakon što budu implementirani na blockchain-u bez suglasnosti svih sudionika. To osigurava da ugovori ostanu nepromijenjeni. Kako se pametni ugovori izvršavaju na blockchain-u, to znači da su smanjeni troškovi i vrijeme potrebno za transakcije jer nema potrebe za posrednicima ili centralnim autoritetom. Pametni ugovori najčešće se koriste za financijske transakcije, opskrbne lance, kupnju, prodaju i upravljanje nekretninama, glasanje i druge svrhe [19, 20].

3.12. Solidity

Solidity je programski jezik kreiran za pisanje pametnih ugovora koji se izvršavaju na Ethereum blockchain mreži. On je jedan od najpopularnijih programskih jezika za razvoj decentraliziranih aplikacija, omogućava kreiranje samostalnih, nepromjenjivih, transparentnih ugovora koji mogu automatizirati procese i transakcije u blockchain mreži. Solidity nalikuje JavaScript-u po sintaksi, što olakšava učenje njegove sintakse programerima koji su već upoznati s JavaScript programskim jezikom. Također, sadrži neke koncepte koji su slični konceptima drugih programskih jezika kao što su C++ i Python. Koristi statično pisanje, što znači da su tipovi varijabli definirani pri deklaraciji. Ima podršku za nasljeđivanje, eventi omogućuju emitiranje zapisa u blockchain logove, omogućuje korištenje biblioteka i sučelja, podržava EVM. Solidity se najčešće koristi za stvaranje tokena koji predstavljaju vlasništvo nad

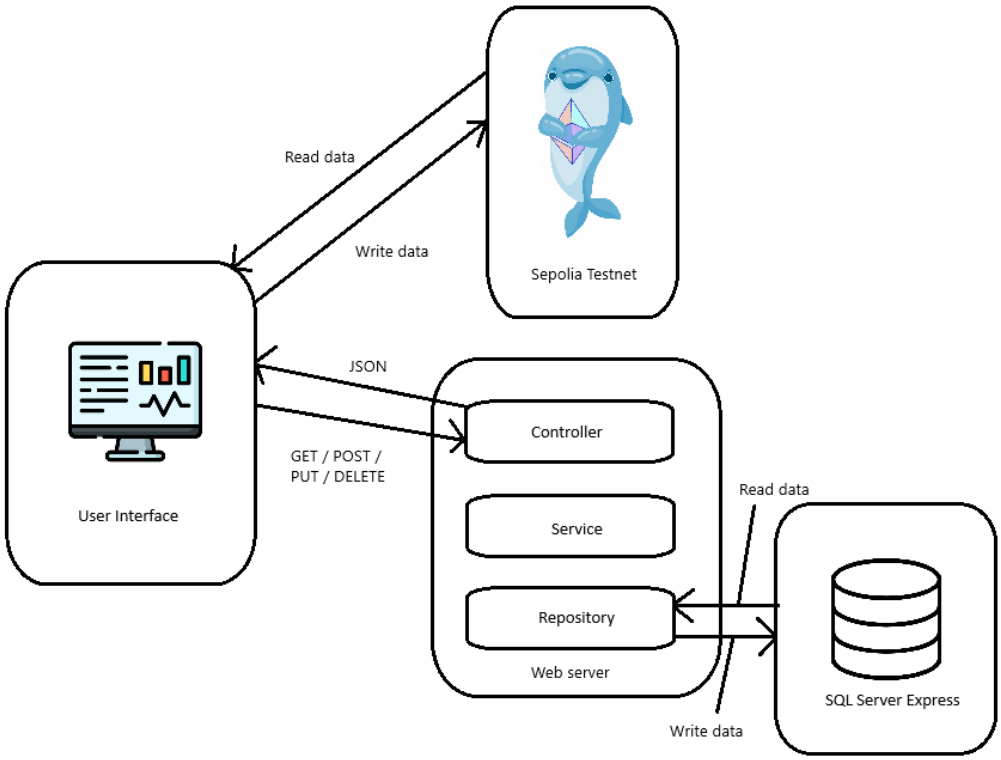
imovinom, pravilima ili uslugama, za kreiranje decentraliziranih aplikacija, financijskih aplikacija, pametnih ugovora itd. [21].

4. IZRADA WEB APLIKACIJE

U ovom poglavlju detaljno je opisana izrada web aplikacije. Opisano je kreiranje poslužiteljske strane aplikacije, kreiranje korisničke strane aplikacije i kreiranje pametnog ugovora.

4.1. Arhitektura aplikacije

Na slici 5.1. je prikazana arhitektura aplikacije. Svi podaci potrebni za prikaz podataka vezanih uz kino i korisnika se nalaze u bazi podataka koji se dohvaćaju preko „Repository“ klasa poslužiteljske strane aplikacije. Podaci se obrađuju unutar „Service“ sloja te se šalju preko api krajnjih točaka koje su napisane u „Controller“ klasama. Korisničko sučelje služi za prikaz sadržaja korisniku, šalje zahtjeve za dobivanje podataka s poslužiteljske strane i kada korisnik rezervira karte šalje podatke kako bi ih pametni ugovor mogao spremiti na blokchain te poziva funkciju za dobivanje podataka s blockchaina o kupljenim kartama.



Sl. 4.1. Prikaz arhitekture aplikacije

4.2. Kreiranje poslužiteljske strane aplikacije

Kako je za poslužiteljsku stranu web aplikacije korišten ASP.NET u C# programsku jeziku, za izradu poslužiteljske strane korišteno je Visual Studio razvojno okruženje. Kako bi poslužiteljska strana aplikacije radila ispravno potrebno je kreirati i strukturirati kod koji će komunicirati s bazom, obrađivati podatke, te ih slati na korisničku stanu i primati s korisničke strane aplikacije. Radi preglednosti, kod se razdvaja u različite mape. Kod ovog projekta podijeljen je na devet mapa. U datoteci „Controllers“ nalaze se kontroleri koji omogućuju komunikaciju s korisničkom stranom, u njima su metode kojima se može pristupiti URL-om. Na primjer, da se dobije popis svih kina u sustavu, na korisničkoj strani aplikacije poziva se https://localhost:7006/api/Cinema url. U mapi Dtos nalaze se „data transfer objects“. To su objekti koji služe za slanje i primanje podataka s korisničke strane. Na primjer, ako se želi poslati neki objekt na korisničku stranu, ali se ne želi poslati neki sigurnosni parametar koji se nalazi u modelu, taj podatak se makne pri kreiranju Dto-a, te se na korisničku stranu pošalju samo željeni podaci. Također, ako se dobivaju podaci s korisničke strane, a potrebno je dodati još neki podatak uz te podatke, primljeni Dto se pretvori u model te se nadoda još ono što je potrebno te se tako u cjelini spremaju podaci. Na primjer kada se korisnik registrira, unese email, korisničko ime i lozinku, prilikom registracije pridoda mu se i uloga „User“. U mapi „Data“ nalazi se datoteka „ApplicationDBContext.cs“ u kojoj su povezani modeli sustava kako bi se pomoću njih mogla kreirati i koristiti baza podataka pošto se u projektu koristi biblioteka Entity Famework. Entity Framework biblioteka omogućuje programerima da relacijski mapiraju podatke koristeći modele aplikacije te se tako izbjegava potreba za pisanjem koda za pristup podacima u bazi te eliminira potrebu za SQL jezikom. U mapi „Images“ se nalaze slike korištene za ovaj projekt. Mapa „Interfaces“ sadrži sučelja za servise korištene u projektu. U mapi „Mappers“ nalaze se datoteke koje služe kako bi se brzo i efikasno tipovi podataka mogli pretvarati iz jednog u drugi. Npr. iz modela u Dto ili suprotno. Mapa „Modles“ sadrži sve modele koji su korišteni u ovom projektu. Modeli su reprezentacija podataka u sustavu. Na primjer, model za kino sadrži id kina, ime kina, broj dvorana, broj sjedala u kinu, adresu kina i sliku. Zadnje dvije mape su „Repository“ i „Service“. U Repository mapi se nalaze datoteke koje sadrže metode koje komuniciraju s bazom podataka, preko njih se dobivaju i šalju podaci koji se koriste u aplikaciji. U Service mapi se nalaze datoteke koje odrađuju neki logički posao. Na primjer, kreiranje JWT tokena. Za registraciju i prijavu korištena je ASP.NET Core Identity biblioteka.

4.3. Kreiranje korisničke strane aplikacije

Aplikacija je rađena funkcijskim komponentama, što znači da je svaki dio aplikacije funkcija koja se nalazi u drugoj funkciji ili koja poziva drugu funkciju unutar sebe. Te se funkcije još nazivaju komponentama. Komponente su strukturirane hijerarhijski, što znači da postoje roditelj komponente i dijete komponente. Glavna struktura ovog projekta prikazana je slikom 4.2.

Linija ***Kod***

```
1:      import { ReactNode } from "react";
2:      import { Navbar } from "../components/Navbar";
3:      import { Footer } from "../components/Footer";
4:
5:      export const MainLayout = ({ children }: { children: ReactNode })=>{
6:        return (
            <div className="relative flex flex-col min-h-screen overflow-x-
7:      hidden center items-center bg-quaternary-light">
8:              <Navbar />
9:              <main>{children}</main>
10:             <Footer />
11:           </div>
12:       );};
```

Sl. 4.2. Glavna struktura korisničkog sučelja

Prve tri linije koda služe za dohvaćanje koda iz drugih datoteka, prva da se dohvati tip React komponenti jer je projekt pisan u TypeScryptu, a ostale dvije da se dohvate komponente Navbar i Footer. Zatim komponenta „MainLayout“ koja predstavlja glavnu strukturu. Ona prima komponentu koju treba prikazati kao prop te ju renderira između Navbar i Footer komponente. U ovom projektu se kao prop predaje komponenta NavigationRoutes. NavigationRoutes komponenta sadrži rute do svih stranica korisničke strane aplikacije. Za navigaciju među tim stranicama koristi se biblioteka React Router. Ona služi za kreiranje dinamičkih jednostranih stranica tako što kontrolira navigaciju i renderira stranice na temelju URL-a. Korisničko sučelje aplikacije sastoji se od osam različitih stranica. Jedna od njih je „Home“. To je stranica koja se otvori kada se pokrene projekt. Ondje korisnik može odabrati u koje kino želi ići gledati film. „Auth“ je stranica na kojoj se korisnik može prijaviti ili registrirati. „CinemaPage“ je stranica koja, nakon što se odabere kino s Home stranice, prikazuje sve projekcije koje su dostupne u tom kinu. Zatim stranica „MovieDetails“ prikazuje detalje o filmu, kao što su ime filma, sadržaj,

trajanje, originalni naslov, žanr, godina izlaska filma, zemlja podrijetla, vrijeme, datum i cijena projekcije.

Nakon što se pritisne na gumb „Buy Ticket“ u MovieDetails korisnik je preusmjeren na stranicu „SelectSeats“. Ondje odabire sjedala koja nisu već rezervirana. Stranica se sastoji od mreže gdje su potvrdni okviri u obliku stolica, svaki potvrdni okvir predstavlja jedno mjesto u dvorani kina. Kad se pritisne gumb „Reserve Selected Seats“, otvara se modal na kojemu piše za koji film kupuje kartu, koja je mjesta korisnik rezervirao, koliko košta pojedinačno sjedalo i ukupno sva sjedala za tu kupnju. Na dnu modala su tipke za prihvaćanje ili odbijanje kupnje. Ako korisnik prihvati kupnju, otvara se MetaMask novčanik i pita korisnika da prihvati i naplati transakciju za kupnju. Ako korisnik prihvati zapisuju se podaci na blockchain i korisnik je preusmjeren na stranicu „SuccessPurchase“. Ondje se prikazuju neki od podataka kupnje, korisničko ime, ime filma za koji se kupila karta, ime kina, oznaka dvorane, broj reda i stupca za sjedala koja su rezervirana, datum i vrijeme projekcije. „UserTickets“ je stranica gdje korisnik ima uvid u sve prijašnje karte koje je kupio. „Dashboard“ je stranica kojoj samo administrator ima pristup. Ondje na osnovu korisničkog imena može pretražiti sve transakcije koje je korisnik s tim korisničkim imenom obavio, te može dodati novo kino, dvorane i sjedala. Također, može dodati novi film u sustav te napraviti novu projekciju za odabrano kino i dvoranu. „Error“ je stranica koja se prikazuje ako korisnik pokuša upisati URL koji ne postoji. Na toj stranici postoji tekst na kojem piše error i gumb koji preusmjeri korisnika natrag na početnu stranicu.

Za upravljanje globalnim stanjem podataka korištena je biblioteka „React Redux Toolkit“. Ta biblioteka pruža mogućnost spremanja podataka iz bilo koje komponente u aplikaciji te čitanje istih iz bilo koje druge komponente. Svaki korak koji korisnik napravi sprema se u redux stanje, te kada korisnik odabere kino, prikažu se projekcije za to kino, a ako odabere jednu projekciju i odluči se vratiti tipkom nazad, ostat će odabrano kino i prikazat će se projekcije za to kino. Također, kada se korisnik logira, podaci se spremaju u redux stanje te se tako omogućava rukovanje podacima vezanima o korisniku unutar cijele aplikacije korisničkog sučelja. U slučaju ako se osvježi kartica preglednika, korisnikovi podaci o prijavi ostaju pohranjeni u lokalnom skladištu podataka te je on i dalje prijavljen kada se stranica ponovo učita.

Biblioteka korištena za ikone je „Material Icons - Material UI“. Ikone korištene u ovom projektu su ikone za prijavu i odjavu, za prikazivanje izbornika i zatvaranje izbornika te sjedala za prikaz sjedala u dvorani kina.

Pri korištenju poziva krajnjih točaka poslužiteljske strane, korištena je biblioteka axios. Ona služi za kreiranje HTTP poziva, preformira asinkrone radnje, omogućava lagano slanje GET, POST, PUT, DELETE zahtjeva. Slika 4.3. prikazuje primjer pozivanja POST zahtjeva metode prilikom prijave korisnika, ona šalje korisničko ime i lozinku te čeka odgovor sa server strane aplikacije. Ako je zahtjev uspješan u odgovoru se dobiva JWT token, u suprotnom u odgovoru se vrati greška.

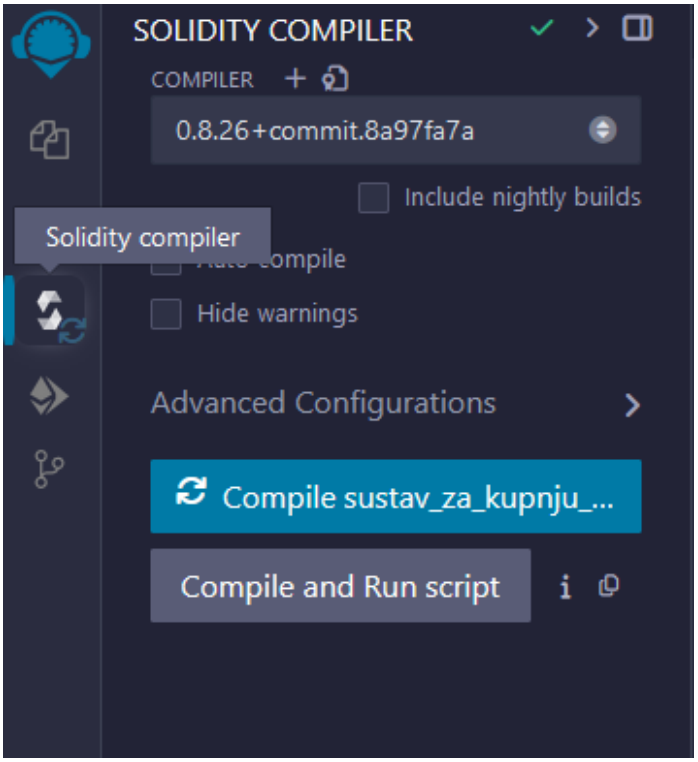
Linija ***Kod***

```
1:      import axios from "axios";
2:      import { API_URL } from "../constants";
3:
4:      export const login = (username: string, password: string) => {
5:        return axios.post(`${API_URL}/Account/login`, {
6:          username: username,
7:          password: password,
8:        });
9:      };
```

Sl. 4.3. Login metoda za prijavu korisnika

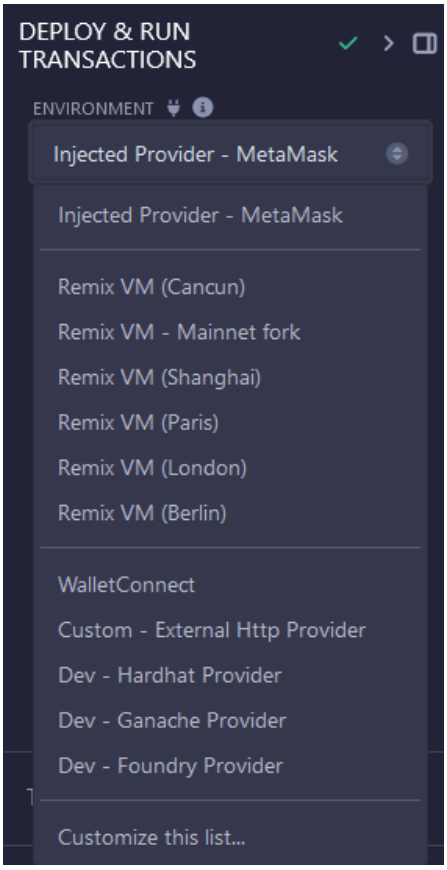
4.4. Kreiranje pametnog ugovora

Svaki pametni ugovor treba započeti licencom. U ovom projektu korištena je „// SPDX-License-Identifier: MIT“ licenca. Uz to, kod mora imati pragma smjernicu koja se koristi za određivanje verzije Solidity kompajlera koja je potrebna za kompajliranje i pokretanje ugovora. U ovom projektu korištena je „pragma solidity ^0.8.0;“ smjernica. To znači da ovaj pametni ugovor može biti kompajliran s verzijom Solidity 0.8.0 ili novijom, ali ne s verzijom 0.9.0 ili novijom od verzije 0.9.0, jer simbol ^ dopušta sve verzije koje ne unose promjene koje mogu uzrokovati nekompatibilnost. Kod ovog projekta se sastoji od struktura za kino i sjedalo, funkcije za kreiranje rezervacije i funkcije za čitanje kreirane rezervacije. Nakon što je kod napisan, potrebno je kompajlirati pametni ugovor. Navedeno se radi tako da se odabere kartica „Solidity compiler“, zatim se odabere željena verzija Solidity-ja koja je jednaka kao i verzija pametnog ugovora te se pritisne na gumb na kojem piše „Compile“ i ime datoteke pametnog ugovora. Solidity compiler kartica je prikazana na slici 4.4.



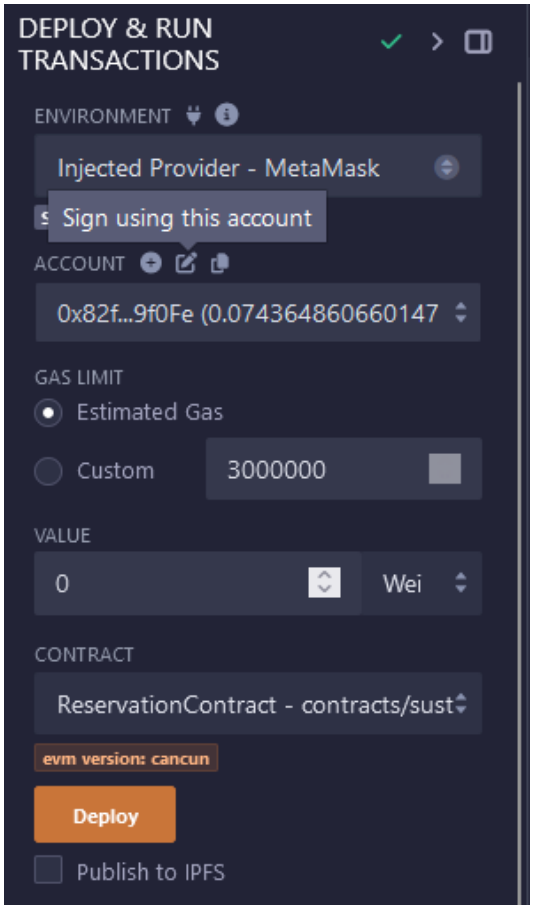
Sl. 4.4. Prikaz Solidity compiler kartice u Remix IDE

Kada je pametni ugovor kompajliran može se implementirati na blockchain mrežu. To se postiže tako da se odabere „Deploy & Run Transactions“ kartica. Nakon toga je potrebno narediti još nekoliko opcija. U padajućem izborniku „Environment“, ako se želi postaviti pametni ugovor na Sepolia Testnet mrežu, potrebno je odabrati „Injected Provider - MetaMask“. To je prikazano slikom 4.5.



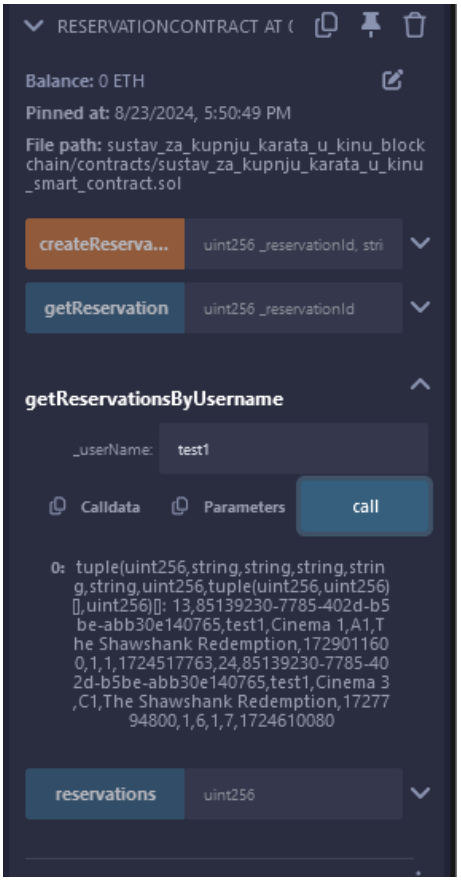
Sl. 4.5. Prikaz odabira okoline za implementaciju u Remix IDE

To spaja Remix i MetaMask. Ako korisnik nema instaliranu ekstenziju MetaMask u internet pregledniku, potrebno ju je instalirati i napraviti račun. MetaMask je novčanik za kriptovalute koji korisnicima omogućuje pohranjivanje, slanje i primanje imovine temeljene na Ethereum-u i interakciju s decentraliziranim aplikacijama. Nakon što korisnik odabere opciju „Injected Provider - MetaMask“, otvori se iskočni prozor MetaMask ekstenzije te se dobije upit za dopuštenje da se MetaMask spoji s Remix-om. Pritiskom na „Connect“, MetaMask i Remix se povežu. Potrebno je postaviti MetaMask na željenu test mrežu. Postavljene postavke prije implementacije se mogu vidjeti na slici 4.6.



Sl. 4.6. Prikaz postavki za implementaciju pametnog ugovora na mrežu

Nakon što je sve postavljeno, pritisne se dugme „Deploy“. MetaMask će pitati za dopuštenje i kada se odobri bit će naplaćena transakcija i pametni ugovor implementiran na Sepolia Testnet mrežu. Nakon što je pametni ugovor implementiran na Testnet mreži, moguće je komunicirati s njim u sekciji „Deployed Contracts“. Ondje se nalaze sve funkcije koje se mogu koristiti. Ako se radi o funkciji koja čita podatke s ugovora, nju se može koristiti bez dodatne naplate, no ako se zapisuju podaci u pametni ugovor na blockchain mrežu, potrebno je platiti transakciju kriptovalutom. Slika 4.7. prikazuje funkcije kojima se može komunicirati s pametnim ugovorom.

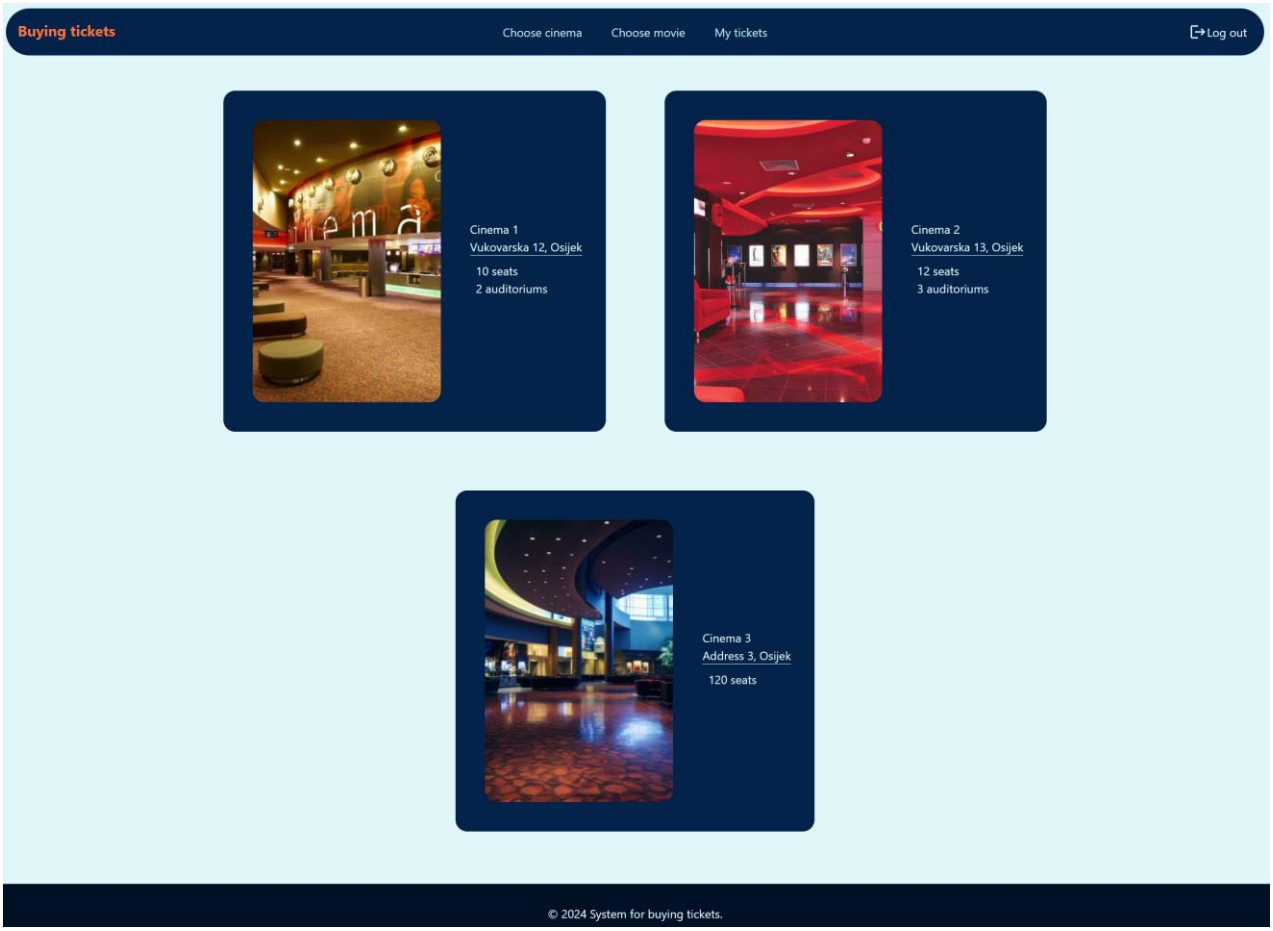


Sl. 4.7. Prikaz metoda kojima se može komunicirati s pametnim ugovorom

5. PREGLED ZAVRŠENE APLIKACIJE

U ovom poglavlju detaljno je opisan tok korištenja aplikacije te je svaki dio popraćen slikama. Za potrebe testiranja korištena je SepoliaETH kriptovaluta. To je kriptovaluta koja se koristi za testiranje blockchain transakcija na Sepolia Testnet mreži.

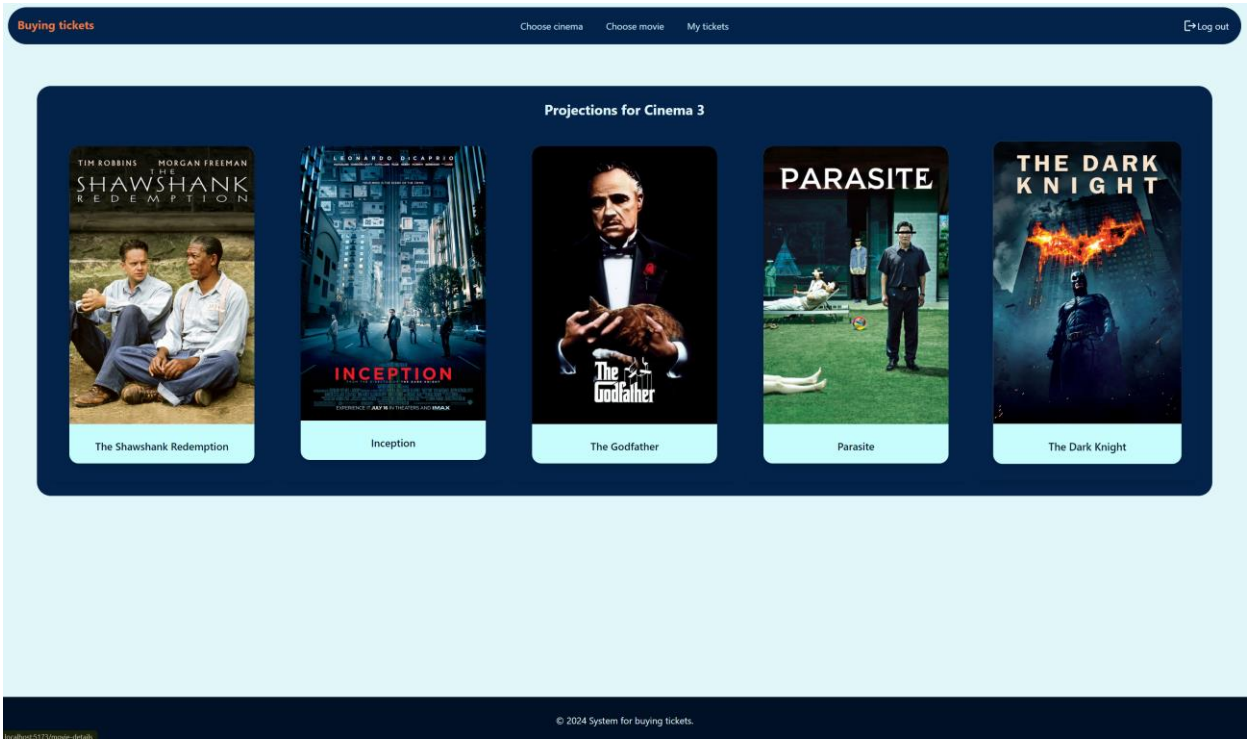
Kada se pokrene aplikacija, korisniku se prikazuje „Home“ stranica. Ondje može vidjeti navigacijsku traku na kojoj se nalazi ime aplikacije, link na Home stranicu, link na stranicu za odabir filmova, link do stranice s korisnikovim kupljenim kartama, link do stranice nadzorne ploče ako je prijavljeni korisnik administrator te gumb koji vodi do stranice za prijavu i registraciju ili, ako je korisnik prijavljen, onda je prikazan gumb za odjavu. Na dnu stranice prikazan je footer. Glavni sadržaj Home stranice je mogućnost odabira kina. Korisnik odabire kino u koje želi ići gledati film. Home stranica je prikazana na slici 5.1.



Sl. 5.1. Prikaz „Home“ stranice aplikacije

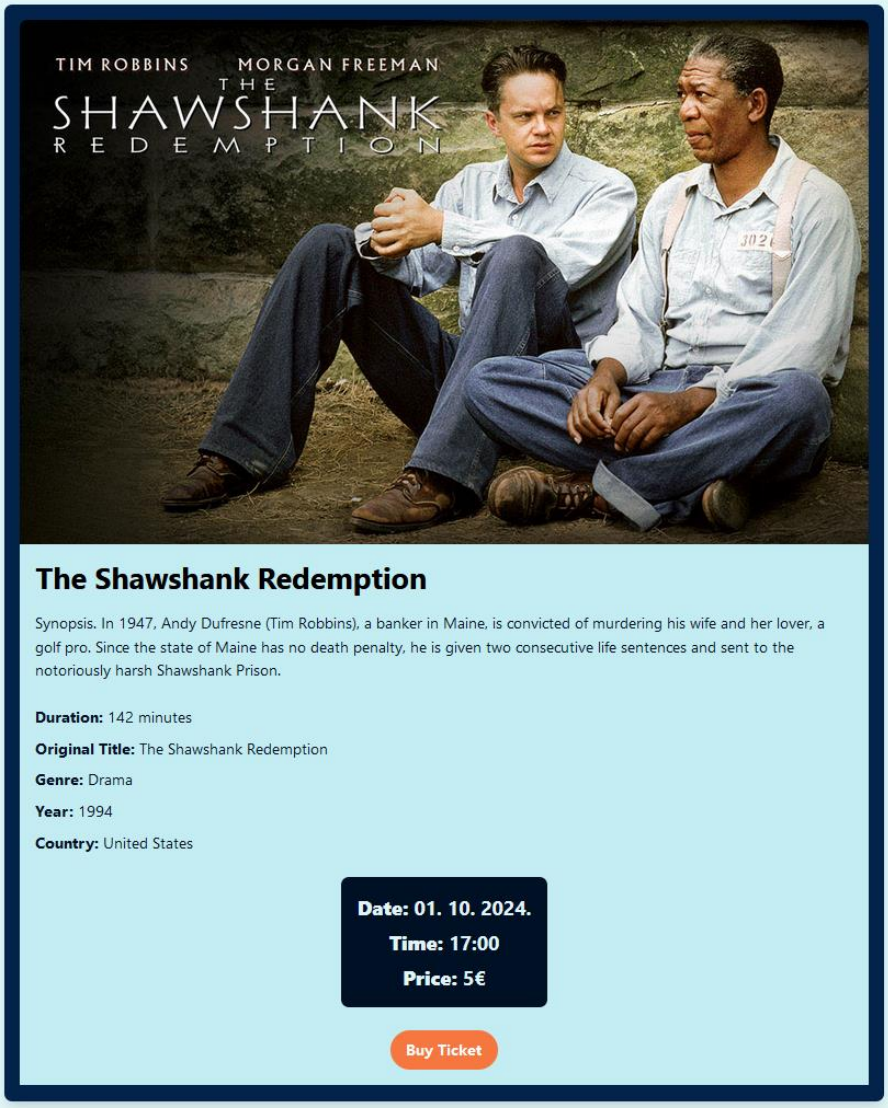
Nakon što korisnik odabere kino u koje želi ići gledati film, preusmjeren je na stranicu „CinemaPage“. Ondje se nalaze svi filmovi koje je moguće gledati u odabranom kinu. Na

stranici se nalazi ime odabranog kina i svi filmovi u obliku kartica na kojima se nalazi ime filma i naslovna slika filma. CinemaPage je prikazana slikom 5.2.



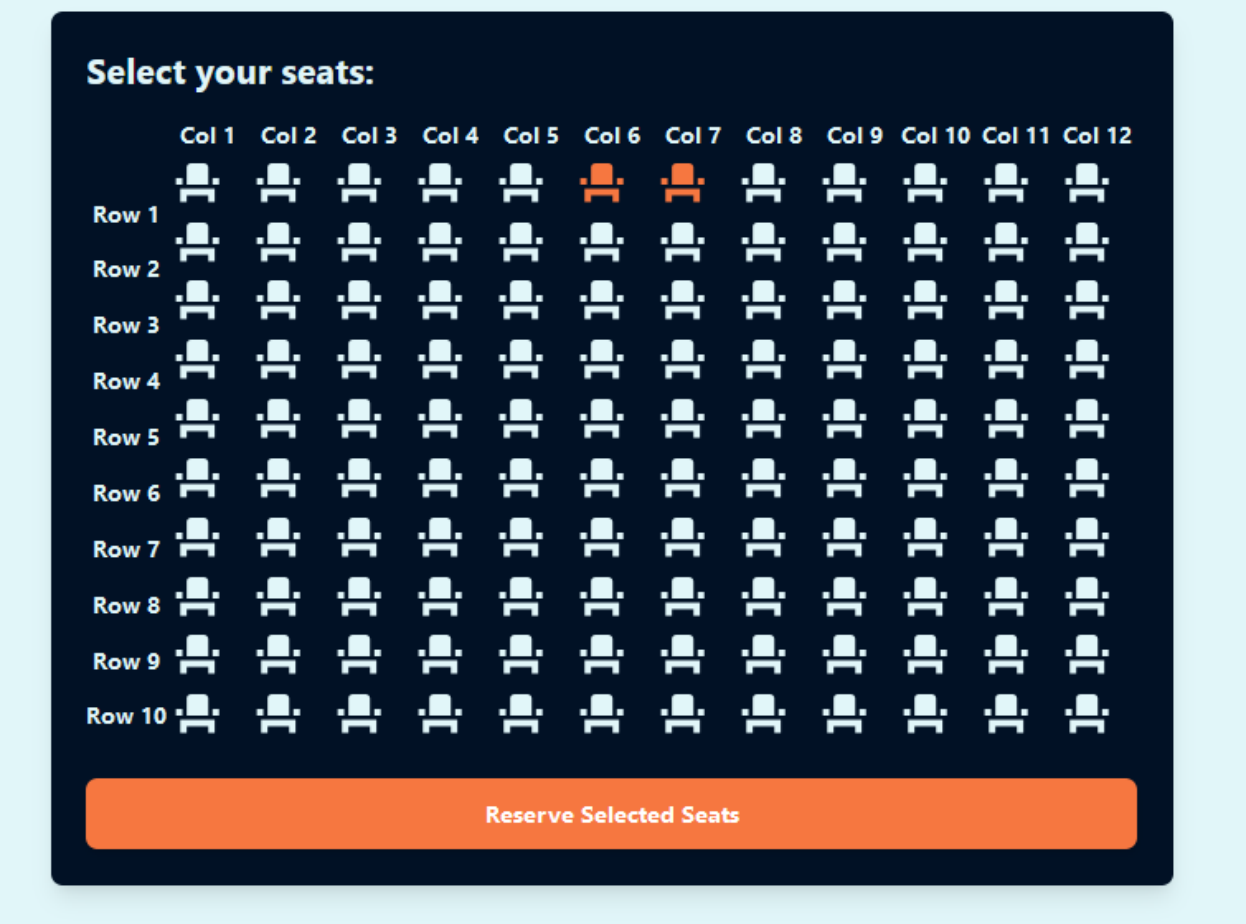
Sl. 5.2. Prikaz „CinemaPage“ stranice aplikacije

Nakon što korisnik odabere film koji želi gledati, preusmjeren je na stranicu „MovieDetails“. Ondje pišu detalji o filmu i projekciji. Za film piše ime, opis filma, koliko film traje, koji je originalni naslov filma, koji je žanr filma, koje godine je film izašao te iz koje je zemlje, a za projekciju piše datum, vrijeme i cijena. Na stranici se nalazi još i dugme „Buy Ticket“. Prikaz MovieDetails stranice je prikazan slikom 5.3.



Sl. 5.3. Prikaz „MovieDetails“ stranice aplikacije

Kada se pritisne na dugme „Buy Ticket“ korisnik je preusmjeren na stanicu „SelectSeats“. Na toj stranici se nalazi reprezentacija sjedala. Ondje korisnik odabire koje sjedalo želi rezervirati za projekciju. Prikaz te stranice je na slici 5.4.

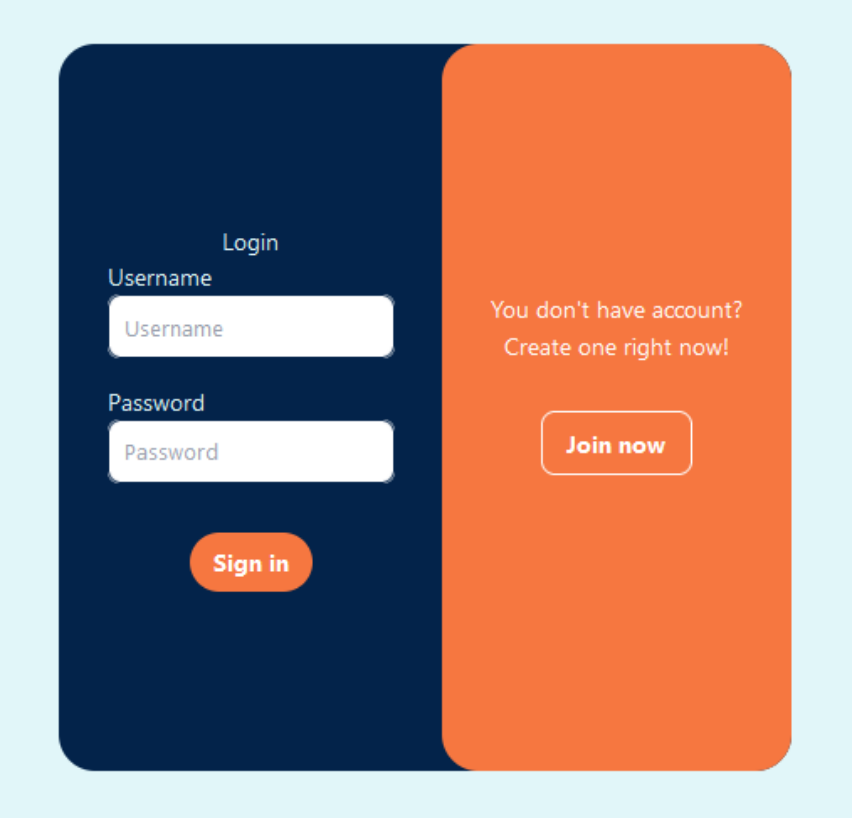


Sl. 5.4. Prikaz „SelectSeats“ stranice aplikacije

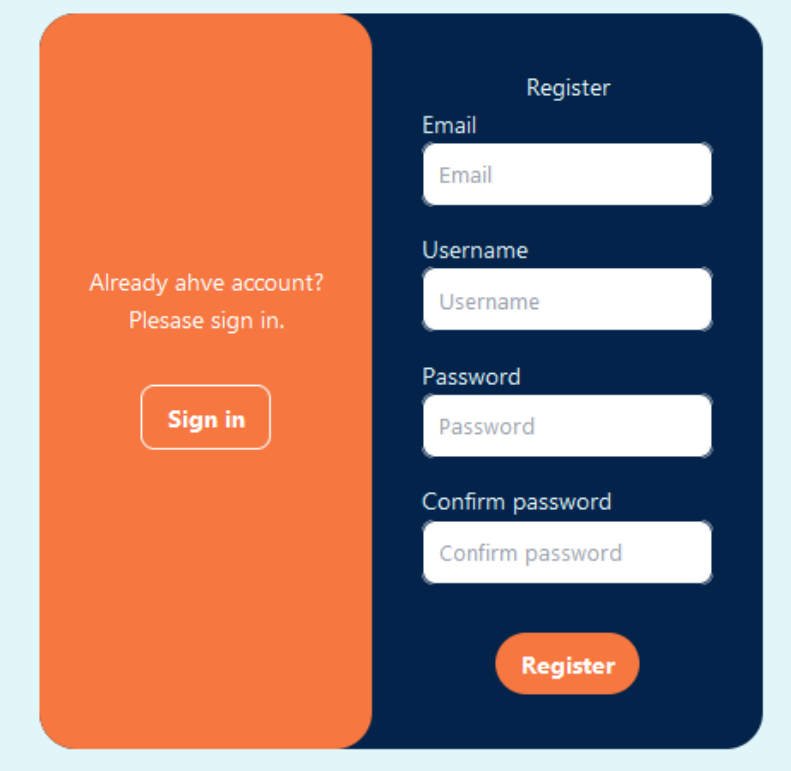
Sjedala koja su svijetlo plave boje su slobodna sjedala. Kada korisnik odabere (klikne) sjedalo, ako je slobodno, mijenja boju u narančastu te postaje odabrano i sjedalo koje je tamno narančaste boje je već rezervirano te se ne može odabrati. Slika 5.5. prikazuje sva tri stanja sjedala.

Sl. 5.5. Prikaz svih mogućih stanja sjedala

Nakon što korisnik odabere sjedala koja želi rezervirati, u slučaju da nije prijavljen, aplikacija ga upozori na to te ga preusmjeri do Auth stranice. Na Auth stranici korisnik se može prijaviti i/ili registrirati, ukoliko nema račun. Za prijavu je potrebno unijeti korisničko ime i lozinku, a za registraciju korisničko ime, email, lozinku, i ponovno lozinku zbog provjere. Slike 5.6. i 5.7. prikazuju komponente za prijavu i registraciju na Auth stranici.



Sl. 5.6. Prikaz forme za prijavu

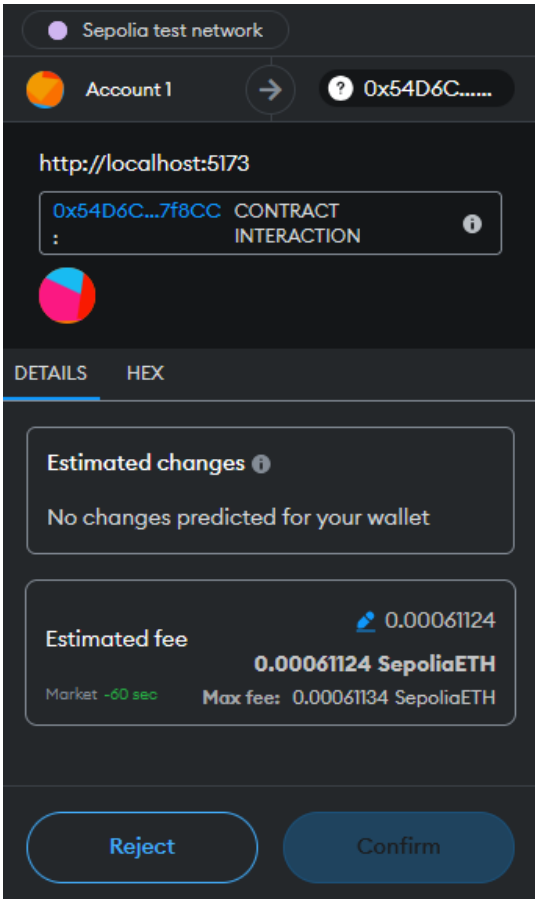


Sl. 5.7. Prikaz forme za registraciju

Nakon što se korisnik prijavio, odabrao mjesta koja želi registrirati i pritisnuo gumb „Reserve Selected Seats“, otvara se modal koji prikazuje film za koji se kupuje karta, sjedala koja je korisnik odabrao za rezervaciju, cijena jedne ulaznice i ukupna cijena za sva sjedala koja je korisnik odabrao. Prikaz tog modala je na slici 5.8.

Sl. 5.8. Prikaz modala za prihvaćanje rezervacije

Ako korisnik odbije rezervaciju u modalu, preusmjeren je na ponovo odabiranje filma i rezervacija je poništena, u suprotnom, ako prihvati rezervaciju otvara se MetaMask novčanik te korisnik dobiva upit želi li prihvatiti transakciju. Primjer iskočnog prozora upita za transakciju pri kupnji karte se može vidjeti na slici 5.9.

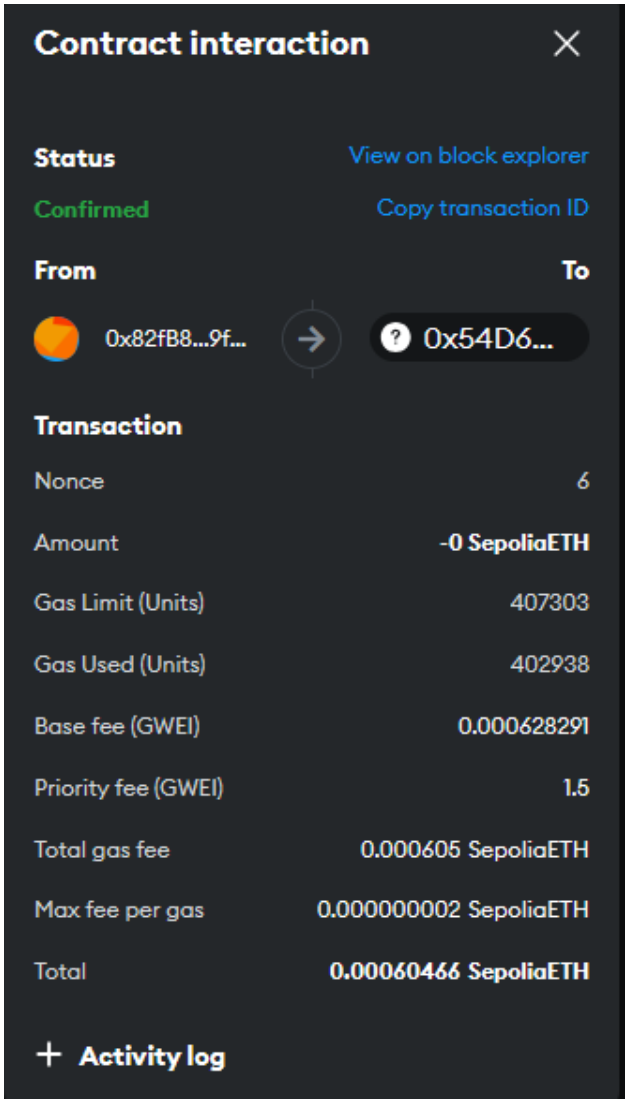


Sl. 5.9. Prikaz primjera iskočnog prozora upita za transakciju pri kupnji karte

Nakon toga, ako je kupnja uspješna, korisnik je preusmjeren na stranicu „SuccessPurchase“ gdje može vidjeti podatke o kupnji. Podaci koji se nalaze su korisničko ime, ime filma, ime kina, dvorana kina, datum i vrijeme projekcije te sjedala koja je korisnik rezervirao. Stranica SucessPurchase je prikazana slikom 5.10.

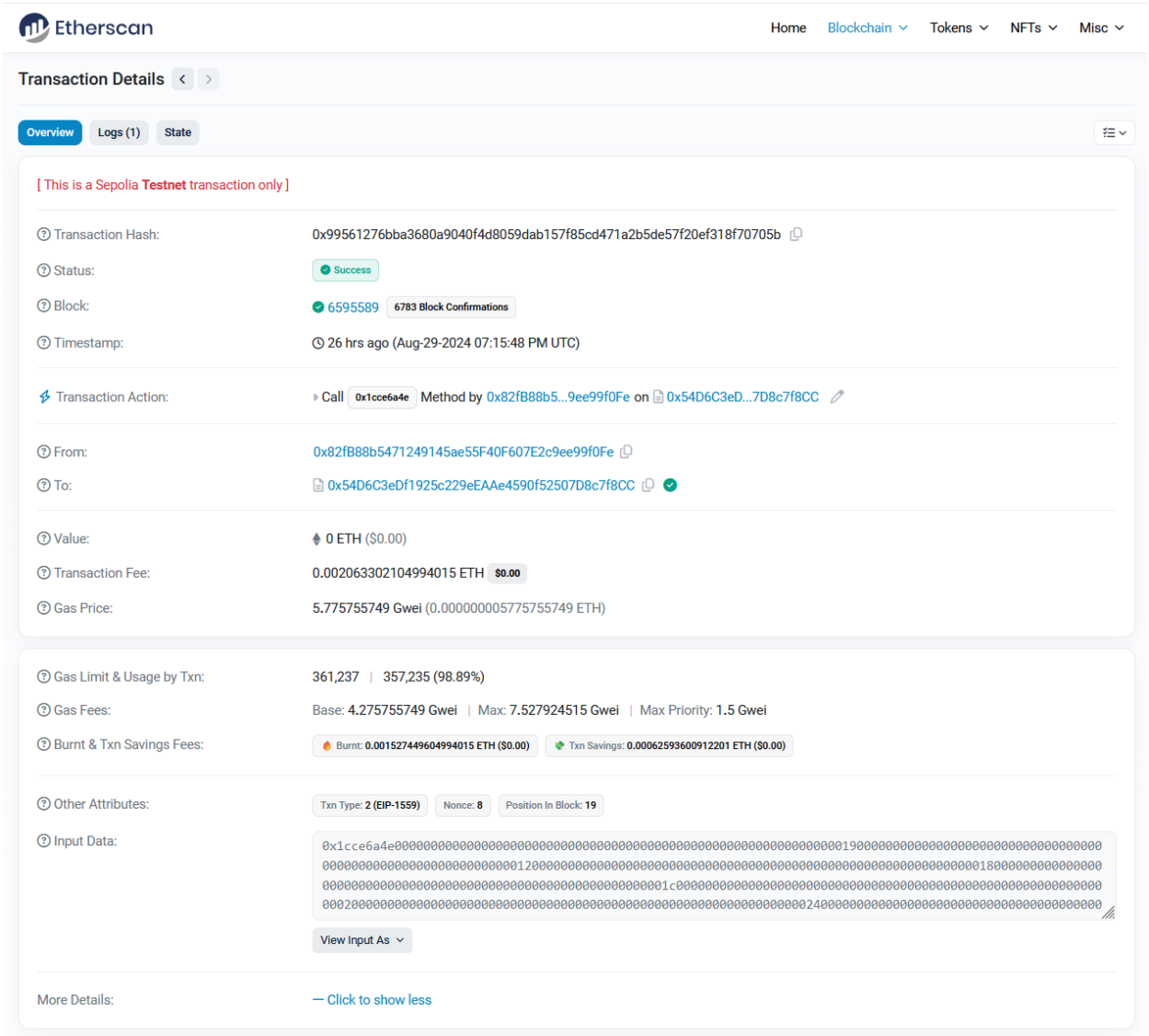
Sl. 5.10. Prikaz primjera podataka na stranici SuccessPurchase

Nakon uspješne kupnje moguće je vidjeti transakciju u novčaniku MetaMask. Primjer jedne transakcije pomoću MetaMask novčanika je prikazana slikom 5.11.



Sl. 5.11. Prikaz jedne transakcije pomoću MetaMask novčanika

MetaMask nudi mogućnost pregleda transakcije unutar sebe te pruža mogućnost prikaza detalja transakcija pomoću hiperveze na Etherscan stranici. Etherscan omogućava pregled i analizu za transakcije za pametne ugovore. Kada se klikne na „View on block explorer“, otvara se Etherscan stranica u kojoj se prikazuju detalji o odabranoj transakciji. Primjer Eterscan detalja o transakciji je prikazan slikom 5.12.

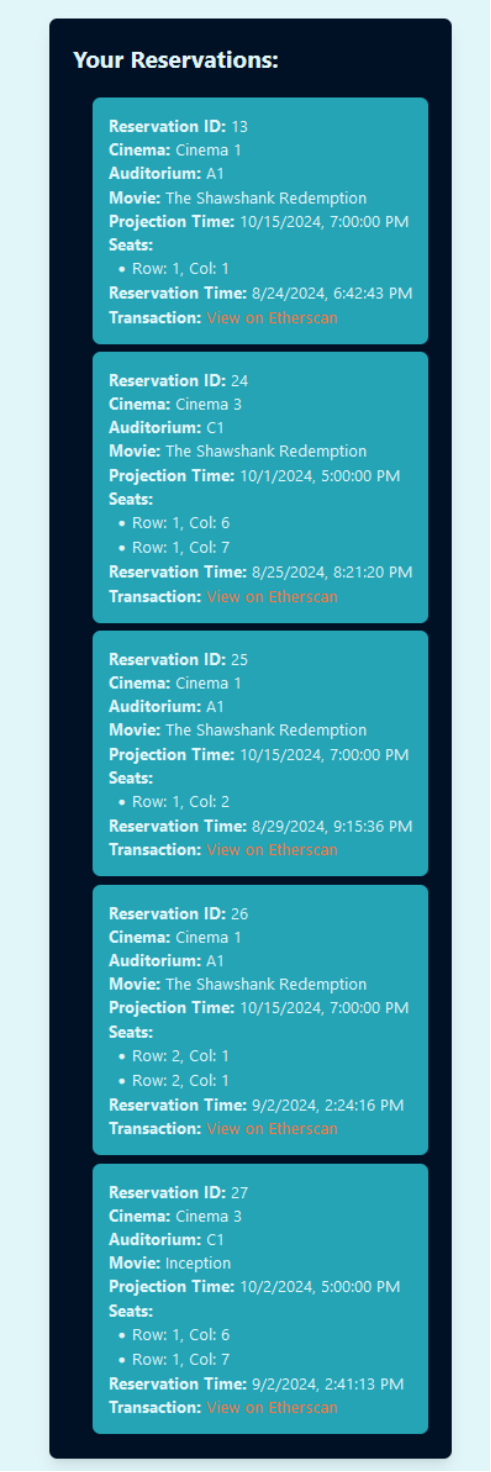


Sl. 5.12. . Primjer Eterscan detaljia o transakciji [22]

U Etherscan detaljima o transakciji Input data je kriptiran i moguće ga je dekriptirati uz pomoć stranice kao što je na primjer Ethereum input data decoder. Ondje je potrebno unijeti ABI (eng. Application Binary Interface) i „Input Data“ iz Eterscan transakcije te je tako moguće dobiti podatke koji su poslani u toj transakciji. Prikaz toga je na slici 5.13.

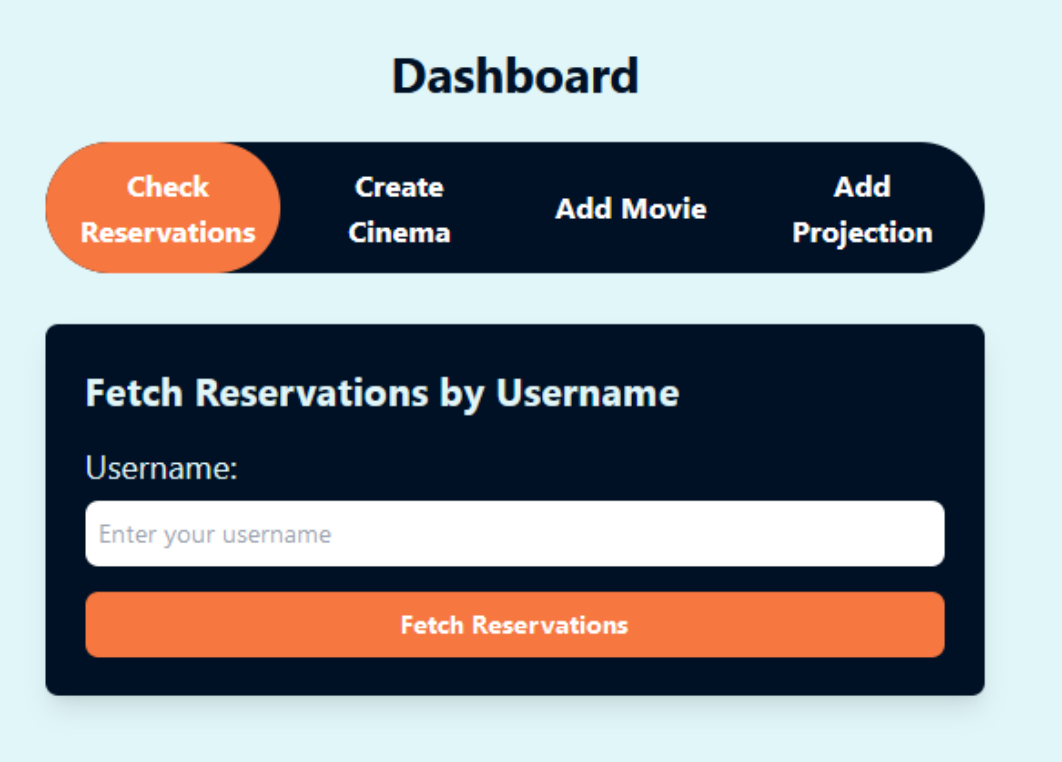
Sl. 5.13. Primjer Ethereum input data dekodera [23]

Kada korisnik ima kupljenu kartu, može ju vidjeti na stranici „UserTickets“. Ondje su prikazane sve karte koje je korisnik kupio. Također, može kliknuti na „View on Etherscan“ i otvorit će se detalji o toj transakciji na Etarscan stranici. Ta stranica je prikazana slikom 5.14.



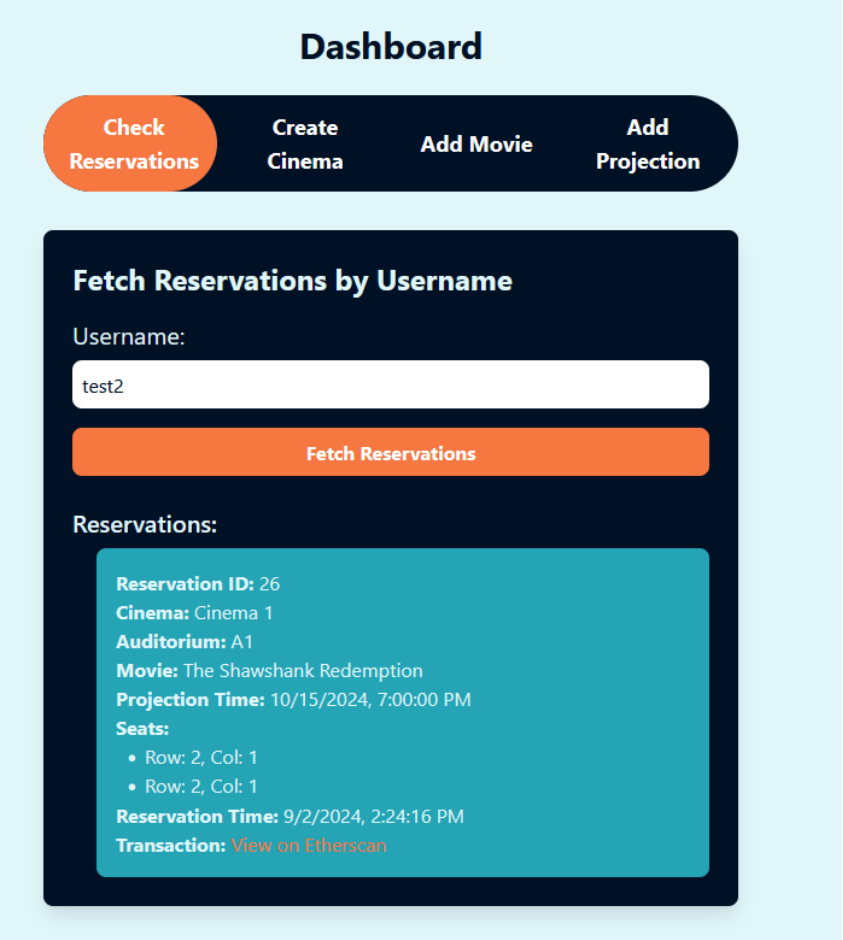
Sl. 5.14. Primjer stranice UserTickets

Ako je korisnik administrator, on ima pristup Dashboard stranici. Na toj stranici se mogu vidjeti sve kupljene karte od strane određenog korisnika, kreirati novo kino sa željenim brojem dvorana i sjedala, kreirati novi film i kreirati nova projekcija. Prikaz Dashboard stranice je na slici 5.15.



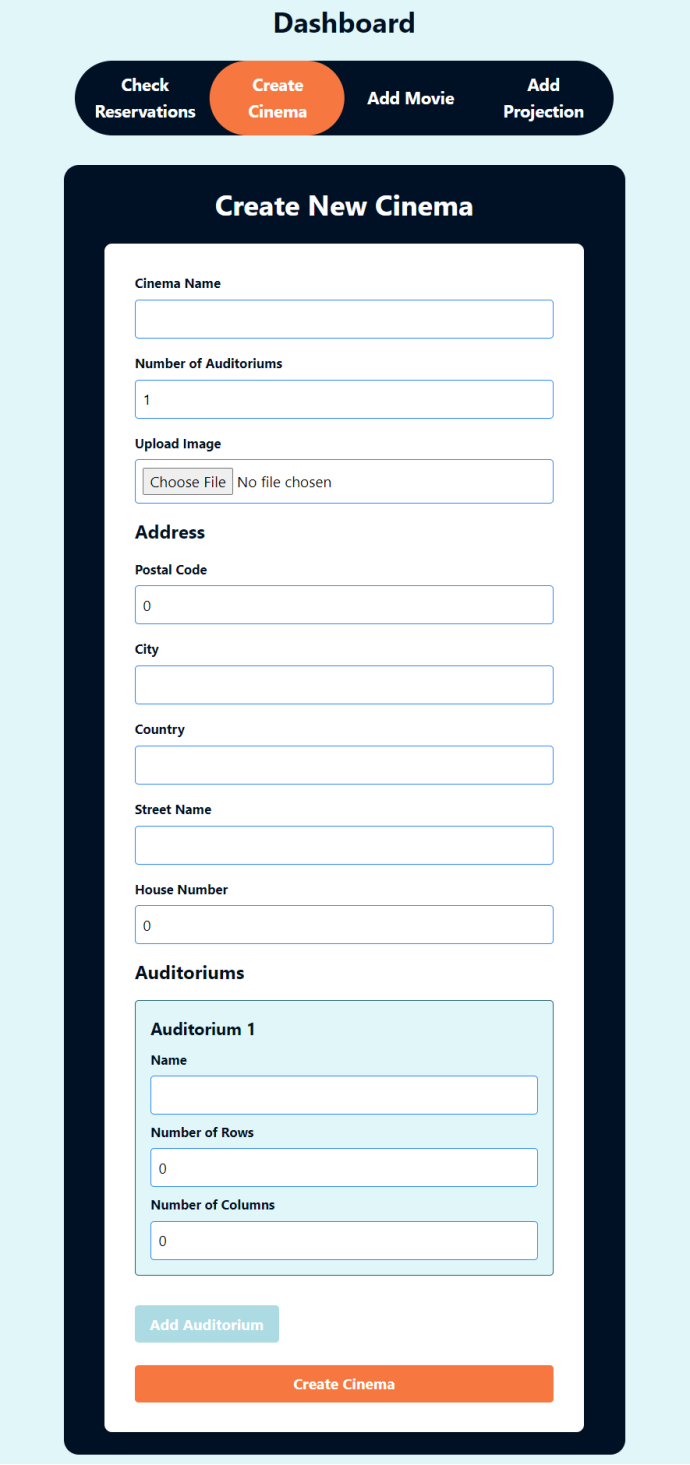
Sl. 5.15. Prikaz „Dashboard“ stranice

Kada je administrator na „Check Reservations“ kartici, unese ime korisnika čije rezervacije želi vidjeti, pritisne dugme „Fetch Reservations“ te dobije s blockchain mreže podatke o rezervacijama tog korisnika. Prikaz kako to izgleda je na slici 5.16.



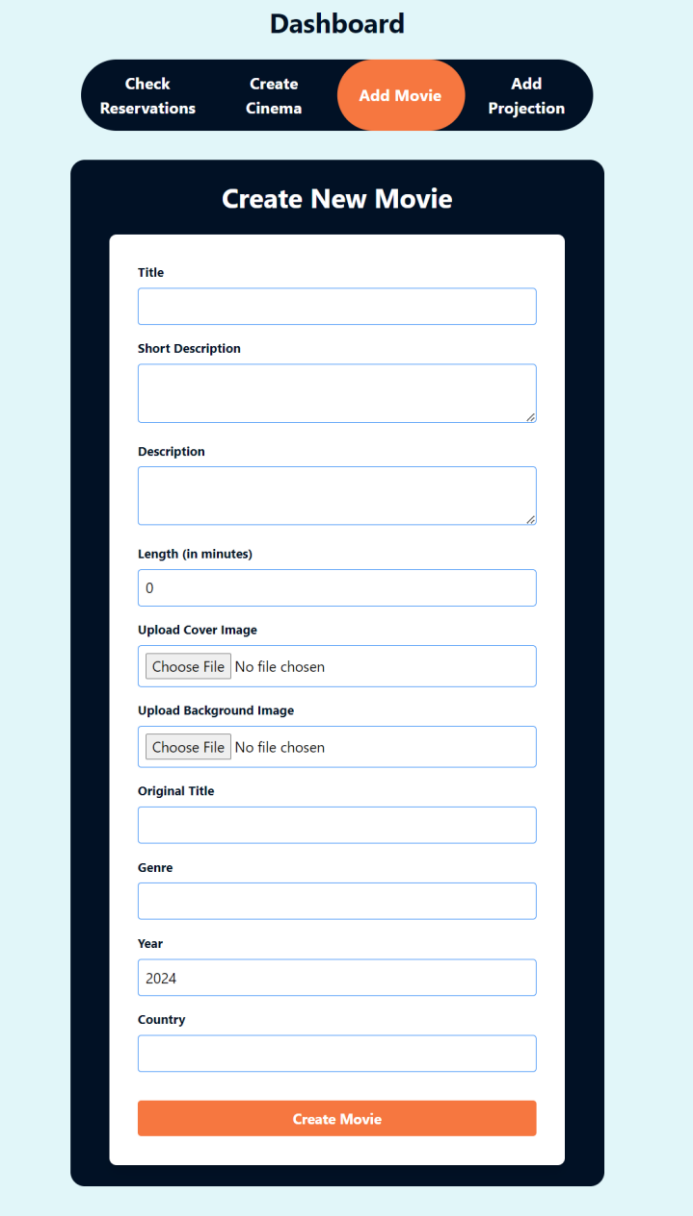
Sl. 5.16. Prikaz „Check Reservation“ kartice kada je odabran korisnik

Ako administrator odabere karticu „Create Cinema“, onda može kreirati novo kino u sustavu. Potrebno je unijeti sve podatke o kinu, adresi i dvorani. Za kino je potrebno ime, broj dvorana i slika. Za adresu poštanski broj, grad, državu, ime ulice i kućni broj. Zatim za dvoranu je potrebno unijeti ime i broj redova i stupaca sjedala. Ta kartica je prikazana slikom 5.17.



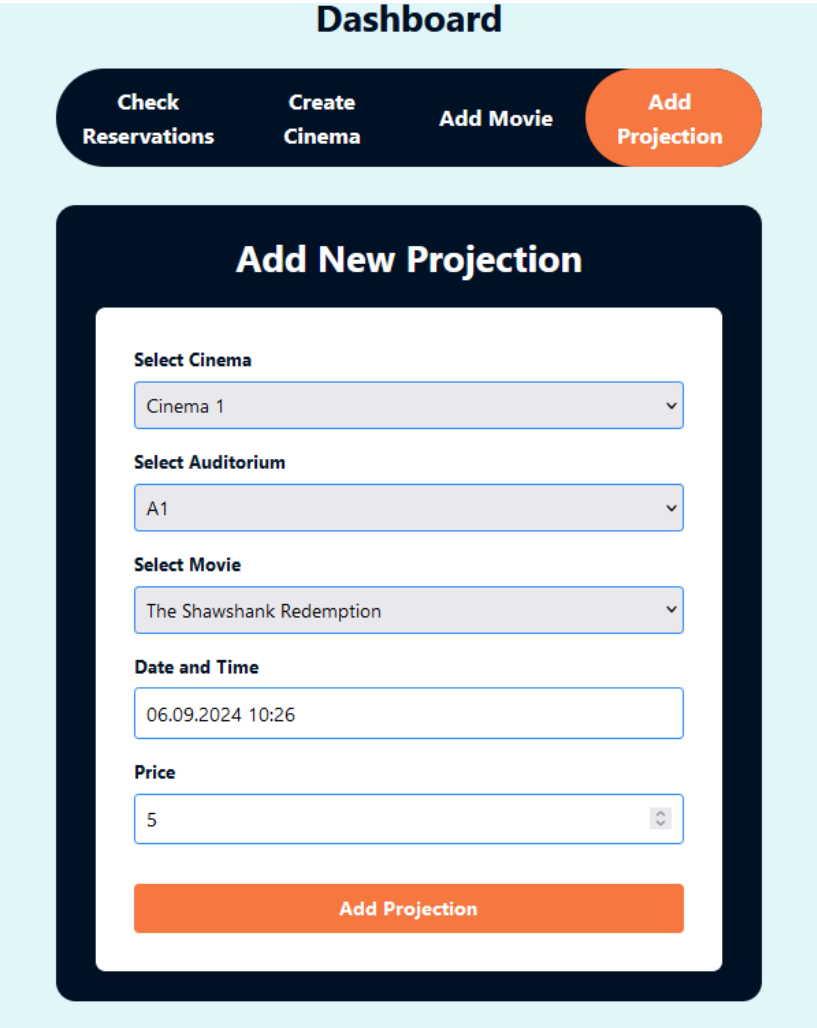
Sl. 5.17. Prikaz „Create Cinema“ kartice

Zatim, na „Add Movie“ kartici administrator može dodati novi film u sustav. Potrebno je upisati ime filma, kratki opis, detaljniji opis, trajanje filma, original naziv, žanr, godinu proizvodnje, zemlju filma, naslovnu sliku i sliku pozadine. Prikaz te kartice je na slici 5.18.



Sl. 5.18. . Prikaz „Add Movie“ kartice

Nakon što je film kreiran može se koristiti u projekcijama kina. Ako je potrebno kreirati novu projekciju, administrator to može učiniti unutar kartice „Add projection“. Ondje odabere kino za koje želi kreirati projekciju, zatim mu se pokažu dvorane tog kina, nakon čega odabere u kojoj dvorani će biti projekcija, a nakon toga bira film iz baze. Na kraju odabere datum, vrijeme i cijenu projekcije te pritisne gumb „Add projection“. Prikaz te kartice je na slici 5.19.



Sl. 5.19. Prikaz „Add Projection“ kartice

6. ZAKLJUČAK

Cilj ovog diplomskog rada je bio izraditi web aplikaciju za kupovanje karata za kino te nakon odabira željenog filma i sjedala zapisati podatke o korisniku i karti (kao što su broj sjedala, oznaka dvorane, vrijeme i datum prikazivanja filma) na blockchain mrežu pomoću pametnog ugovora. Aplikacija je izrađena u nekoliko različitih tehnologija: za bazu podataka korištena je lokalna baza SQL express server, za poslužiteljsku stranu aplikacije korišten je Microsoftov ASP.NET Core razvojni okvir u C# programskom jeziku kako bi se omogućilo upravljanje podacima baze podataka i pozivima API-ja s korisničke strane, za korisničku stranu korištena je React biblioteka pisana u TypeScript-u kako bi se korisniku omogućio intuitivan i lagan rad aplikacijom te za pametni ugovor korišten je Solidity jezik i Sepolia Testnet mreža za implementaciju pametnog ugovora.

Aplikacija se može i proširiti u nekoliko različitih smjerova. Jedan od njih je da se implementira i preuredi za stvarno kino ili lanac kina koji još nemaju mogućnost kupovanja karata preko internet preglednika ili imaju, ali je loš. To bi se ostvarilo tako da se postave stvarni detalji o kinu, kao što su adresa, ime, broj sjedala, raspored sjedala i slični podaci. Također, trebao bi se dodati sustav plaćanja da se može platiti karticom. Uz to, moguće je kreirati i mobilnu aplikaciju koja će se spajati na istu poslužiteljsku stranu aplikacije te će korisnici tako moći brzo i jednostavno kupiti karte za film koji žele ići gledati. Aplikacija bi se mogla i nadograditi u drugom smjeru tako da se nastavi koristiti blockchain, da se plaća kriptovalutama, ali većina korisnika nije upoznata s takvim tehnologijama te njihova integracija ne bi bila uspješna ako kriptovalute ne postanu svakodnevna valuta među normalnim korisnicima.

LITERATURA

[1] CineStar Cinemas, CineStar Cinemas Osijek, [online], Osijek. Dostupno na: https://osijek.cinestarcinemas.hr/. [Pristupljeno: 1.7.2024.].

[2] KINO Bioscoop in Rotterdam [online], Rotterdam. Dostupno na: https://kinorotterdam.nl/. [Pristupljeno: 1.7.2024.].

[3] Emy, querylab/CinemaTicket-Blockchain [online], 2024. Dostupno na: https://github.com/querylab/CinemaTicket-Blockchain. [Pristupljeno: 3.9.2024.].

[4] What is Hardhat? [online], Alchemy, 2023. Dostupno na: https://docs.alchemy.com/docs/what-is-hardhat. [Pristupljeno: 3.9.2024.].

[5] C# Programming: What It Is, How It's Used + How to Learn It [online], Coursera, 2024. Dostupno na: https://www.coursera.org/articles/c-sharp. [Pristupljeno: 1.7.2024.].

[6] C# Language Specification , ECMA , Geneva, 2006".

[7] What is ASP.NET Core? [online], Umbraco. Dostupno na: https://umbraco.com/knowledge-base/asp-dot-net-core/. [Pristupljeno: 1.7.2024.].

[8] Assis Zang , ASP.NET Core Basics: ASP.NET Core Overview [online], Telerik, 2023. Dostupno na: https://www.telerik.com/blogs/aspnet-core-basics-aspnet-core-overview. [Pristupljeno: 1.7.2024.].

[9] Swagger & Swagger UI: What is it and Why is it a must for your APIs? [online], Chakray. Dostupno na: https://www.chakray.com/swagger-and-swagger-ui-what-is-it-and-why-is-it-a-must-for-your-apis/. [Pristupljeno: 26.8.2024.].

[10] What is SQL Server Express used for? [online], Your Office Anywhere. Dostupno na: https://www.yourofficeanywhere.co.uk/info-hub/what-is-sql-server-express-used-for/. [Pristupljeno: 1.7.2024.].

[11] Chinmayee Deshpande, What Is React? [Easily Explained] [online], Simplilearn, 2024. Dostupno na: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs. [Pristupljeno: 1.7.2024.].

[12] Eric Simons, What is Vite (and why is it so popular)? [online], StackBlitz, 2024. Dostupno na: https://blog.stackblitz.com/posts/what-is-vite-introduction/. [Pristupljeno: 1.7.2024.].

[13] P., Williams, WHAT IS TYPESCRIPT? [online], Medium, 2023. Dostupno na: https://medium.com/@pryncwill819/what-is-typescript-35f4e6f37bd5. [Pristupljeno: 1.7.2024.].

[14] Soham De Roy, What is Tailwind CSS? A Beginner's Guide [online], freeCodeCamp, 2022. Dostupno na: https://www.freecodecamp.org/news/what-is-tailwind-css-a-beginners-guide/. [Pristupljeno: 1.7.2024.].

[15] What Is Blockchain and How Does It Work? [online], BlackDuck. Dostupno na: https://www.synopsys.com/glossary/what-is-blockchain.html. [Pristupljeno: 1.7.2024.].

[16] What is Blockchain Technology? [online], AWS. Dostupno na: https://aws.amazon.com/what-is/blockchain/. [Pristupljeno: 1.7.2024.].

[17] What is Ethereum? [online], Ethereum. Dostupno na: https://ethereum.org/en/what-is-ethereum/. [Pristupljeno: 1.7.2024.].

[18] The Investopedia Team, What Is Ethereum and How Does It Work? [online], Investopedia, 2024. Dostupno na: https://www.investopedia.com/terms/e/ethereum.asp. [Pristupljeno: 1.7.2024.].

[19] What Are Smart Contracts in Blockchain? [online], Chainlink, 2023. Dostupno na: https://chain.link/education/smart-contracts. [Pristupljeno: 1.7.2024.].

[20] Chiradeep BasuMallick, Smart Contracts: Types, Benefits, and Tools [online], Spiceworks, 2023. Dostupno na: https://www.spiceworks.com/tech/innovation/articles/what-are-smart-contracts/. [Pristupljeno: 1.7.2024.].

[21] S. C., Team, What is Solidity : A Practical Guide to Blockchain Programming [online], Shardeum, 2023. Dostupno na: https://shardeum.org/blog/solidity/. [Pristupljeno: 1.7.2024.].
[22] TESTNET Sepolia (ETH) Blockchain Explorer [online], Etherscan. Dostupno na: https://sepolia.etherscan.io/. [Pristupljeno: 9.9.2024.].
[23] Ethereum input data decoder [online]. Dostupno na: https://lab.miguelmota.com/ethereum-input-data-decoder/example/. [Pristupljeno: 9.9.2024.].

SAŽETAK

Sustav za kupnju karata i njihovo spremanje u blockchain koristeći pametne ugovore pregledava, analizira i daje rješenje koje pomoću blockchain pametnih ugovora korisnik može kupiti karte za projekciju filma u kinu. U radu su analizirana slična postojeća rješenja za kupovanje karata u kinu i spremanje na blokchain mrežu.

Cilj je konstruirati aplikaciju tako da korisnicima bude omogućena kupnja karte za kino na lagan i intuitivan način. Takav sustav implementiran je uz pomoć razvojnih okvira ASP.NET Core za poslužiteljsku stranu aplikacije, SQL Server Express baze podataka, React-a za korisničko sučelje i RemixIDE za kreiranje pametnog ugovora pisanog u Solidity programskom jeziku. Aplikacija omogućuje korisniku pregled i odabir kina u koje želi ići gledati film, zatim prikazuje filmove koji se prikazuju u tom kinu i pruža mogućnost odabira filma, prikazuje i daje mogućnost odabira sjedala koje nije zauzeto, i zapisuje podatke o karti i korisniku na blokchain pametnim ugovorom.

Zaključno, implementacija aplikacije je ispunila glavne zahtjeve što su kupnja karte i spremanje podataka na blockchain mrežu, ali postoje značajke koje se mogu nadograditi i poboljšati za još bolji rad aplikacije.

Ključne riječi: blockchain, Ethereum, kino, kupnja karata, pametni ugovor

ABSTRACT

A system for buying tickets and storing them in the blockchain using smart contracts

The system for buying tickets and storing them in the blockchain using smart contracts reviews, analyzes and provides a solution that, using the blockchain of smart contracts, the user can buy a ticket for a movie screening in a cinema. The paper analyzed similar existing solutions for buying tickets in the cinema and storing them on the blockchain network.

The goal is to construct the application so that users are able to purchase a movie ticket in an easy and intuitive way. Such a system was implemented with the help of ASP.NET Core development frameworks for the server side of the application, SQL Server Express database, React for the user interface and RemixIDE for creating a smart contract written in the Solidity programming language. The application allows the user to view and select the theater where he wants to go to watch a movie, then displays the movies that are shown in that theater and provides the option to select the movie, displays and provides the option to select the seat that is not occupied, and records the ticket and user information on the blockchain with a smart contract.

In conclusion, the implementation of the application has fulfilled the main requirements of buying a ticket and saving data on the blockchain network, but there are features that can be upgraded and improved to make the applications work even better.

Keywords: blockchain, cinema, Ethereum, smart contract, ticket purchase

ŽIVOTOPIS

Autor ovog diplomskog rada, Antonio Cmrk, rođen je 30. 4. 2000. u Našicama. Preselio se iz Čađavačkog Luga u Slatinu 2002. godine. Ondje 2007. godine kreće u osnovnu školu Eugena Kumičića u Slatini koju završava odličnim uspjehom. Već tada dobiva interes za fiziku, matematiku i informatiku. Zbog toga 2015. godine upisuje Srednju školu Marka Marulića u Slatini, smjer elektrotehničar. Svake godine sudjeluje na školskim i županijskim natjecanjima iz matematike i fizike. 2019. završava srednju školu i upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, sveučilišni preddiplomski studij, smjer Računarstvo. Godine 2022. završava preddiplomski studij te upisuje diplomski studij, smjer Računarstvo, izborni blok Programsko inženjerstvo. Trenutno je student 2. godine diplomskog studija Računarstvo.

_____

Potpis autora