|  |  |
| --- | --- |
|  | Zbornik radova Fakulteta tehničkih nauka, Novi Sad |

**UDK: 004.9**

**DOI:** [**https://doi.org/10.24867/28BE25Ljubovic**](https://doi.org/10.24867/28BE25Ljubovic)

**RAZVOJ BLOKČEJN PODSISTEMA ZA VEB 3.0 PLATFORMU ZA UPRAVLJANJE KURSEVIMA**

**DEVELOPMENT OF THE BLOCKCHAIN SUBSYSTEM FOR A WEB 3.0 PLATFORM FOR COURSE MANAGEMENT**

Stefan Ljubović, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

**Kratak sadržaj –** *Rad se bavi razvojem veb 3.0 aplikacije za upravljanje kursevima primenom računarstva u oblaku. Sadrži opis primena blokčejna za plaćanje i kreiranje NFT sertifikata pomoću Solidity-a, IPFS-a kao i AWS CDK serverless- infrastrukture. Rad uključuje i opis implementacije ovih tehnika izvršene u programskom jeziku Go za backend i komunikacuju sa IPFS veb 3 memorijom kao i korišćenje Solidity-a za pametne ugovore.*

**Ključne reči:** *blokčejn, pametni ugovori, solidity, AWS CDK, serverless*

**Abstract** – *The work deals with the development of a course management application using cloud computing. It contains a description of blockchain applications for payment and creation of NFT certificates using Solidity, IPFS as well as AWS CDK serverless-infrastructure. The paper includes a description of the implementation of these techniques in the Go programming language for the backend and communication with IPFS web3 storage, as well as the use of Solidity for smart contracts.*

**Keywords:** *blockchain, smart contracts, Solidity, AWS CDK, serverless*

**1. UVOD**

Tehnologija blokčejna je donela revoluciju u različite industrije, nudeći inovativna rešenja za dugotrajne izazove. U ovom radu dublje istražujemo praktične primene tehnologije blokčejna, fokusirajući se posebno na platne sisteme i kreiranje Non-Fungible Tokena (NFT). Razmatramo kako se Solidity, programski jezik koji se često koristi za razvoj pametnih ugovora na blokčejn platformama, može iskoristiti za ove svrhe. Takođe, koristimo snagu AWS CDK (Cloud Development Kit) i serverless infrastrukture kako bismo stvorili skalabilno okruženje za naša rešenja bazirana na blokčejnu.

Dok blokčejn napreduje, važno je razumeti kako se može primeniti za unapređenje sistema za plaćanja i uvođenje novih koncepta poput NFT-ova. U ovom radu ćemo ne samo razgovarati o teoriji već i detaljno opisati kako ove tehnike možemo primeniti koristeći programski jezik Go.

Ova kombinacija tehnologija pruža uzbudljive moguć­nosti za inovacije.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NAPOMENA:**

**Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, vanr. prof.**

Na kraju rada, čitaoci će bolje razumeti praktične primene blokčejn tehnologije i kako je mogu iskoristiti u različitim oblastima.

**2. POJAM BLOKČEJN TEHNOLOGIJE**

Blokčejn predstavlja poseban oblik distribuiranih sistema, tačnije, jednu od mogućih implementacija tehnologije poznate kao Distribuirana glavna knjiga (DLT). Pametni ugovori su karakterističan deo blokčejn tehnologije i značajno unapređuju njenu upotrebu. U nastavku ćemo pružiti pregled ključnih pojmova koji su od suštinskog značaja za razumevanje blokčejna.

Pored ovih osnovnih koncepata, važno je napomenuti da AWS CDK (Cloud Development Kit) može biti koristan alat u kontekstu razvoja blokčejna, pojednostavljujući kreiranje i upravljanje beskrajnim infrastrukturnim resur­sima za blokčejn aplikacije. Takođe, pomenućemo i kre­iranje Truffle projekta i proces dizanja pametnih ugovora pomoću ovog popularnog razvojnog okvira.

**2.1. Distribuirani sistemi**

Distribuirani sistemi predstavljaju arhitektonski model u kojem računarski resursi i funkcionalnosti nisu centralizo­vani na jednom mestu, već su raspoređeni i povezani preko mreže [1]. Ova decentralizovana struktura omogu­ćava veću otpornost na kvarove, skalabilnost i efikasniju upotrebu resursa. Distribuirani sistemi se često koriste u različitim oblastima, uključujući cloud računarstvo, mre­že, bankarstvo i naravno, blokčejn tehnologiji. Ovaj kon­cept igra ključnu ulogu u razumevanju i implemen­taciji tehnoloških rešenja današnjeg doba.

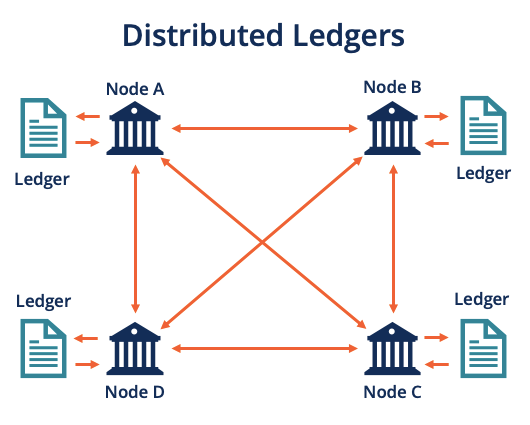
**2.2. Distribuirana glavna knjiga**

Tehnologija distribuirane knjige (DLT) omogućava viso­ko dostupnu bazu podataka održavanu fizički razmešte­nim uređajima, tzv. čvorovima, u nepouzdanom okruže­nju. DLT obećava povećanje efikasnosti i transparentnosti u saradnji između pojedinaca i organizacija, nudeći otpornost na neovlašćeno korišćenje podataka, sigurnost, nepromenjivost i anonimnost [2].

Primene DLT-a uključuju lanac snabdevanja, finansije i zdravstvenu zaštitu. Na primer, u lancu snabdevanja, DLT se koristi za stvaranje sigurne skladišne platforme, koja se replicira preko više čvorova, omogućavajući sigurno skladištenje podataka i automatizaciju poslovnih procesa.

Međutim, implementacija aplikacija na DLT-u suočava se s kompromisima između različitih karakteristika, poput dostupnosti i doslednosti. Izbor odgovarajućeg DLT dizajna postaje ključan, s obzirom na specifične zahteve aplikacije, ali i uzimajući u obzir održivost aplikacije tokom vremena.

U suštini, DLT služi kao zajednička digitalna infrastruktura za visoko dostupne, distribuirane baze podataka u nepouzdanom okruženju, gde se podaci prenose i dodaju knjizi u obliku transakcija. Validacija transakcija se vrši putem digitalnih potpisa i kriptografije javnog ključa.



*Slika 1. Distribuirana glvana knjiga*

**2.3. Blokčejn kao implementacija DLT-a**

Bloklčejn funkcioniše kao distribuirana knjiga, sastav­ljena od niza šifrovanih blokova koji sadrže informacije o validnoj mrežnoj aktivnosti. Svaki blok je vezan za prethodni, stvarajući nepromenljiv i transparentan lanac podataka. Ova tehnologija omogućava bezbednu razmenu vrednosti između entiteta putem interneta, sa čim elimi­niše potrebu za posrednicima. Decentralizovana priroda blokčejna omogućava postizanje saglasnosti i dogovora u mreži bez centralnog upravljanja, što doprinosi transpa­rentnosti i poverenju. Rudarenje, proces dodavanja bloko­va u lanac putem validacije od strane čvorova u mreži, igra ključnu ulogu u održavanju integriteta podataka. Rudari su nagrađeni za dodavanje validnih blokova, što ih motiviše da održavaju konzistentnost podataka i obezbede sigurnost mreže [3].

**2.4. Pametni ugovori**

Pametni ugovori, predloženi od strane Nika Szaboa u 1990-ima, koriste tehnologiju blokčejna za automatsko izvršavanje ugovornih klauzula kada se ispune unapred definisani uslovi. Ovi ugovori smanjujući rizik, administrativne troškove i povećavajući efikasnost poslovnih procesa. Njihova nepromenjivost u blokčejnu pruža transparentnost i smanjuje zlonamerne radnje. Ipak, postoje izazovi poput pitanja privatnosti izvršenja ugovora i potrebe za obezbeđivanjem ispravnosti ugovora zbog mogućih grešaka u računarskim programima. [4]

**2.5 Ethereum virtuelna mašina**

Analogija "distributed ledger" često se koristi za opisivanje blokčejna, poput Bitcoina, koji koristi osnovne alatke kriptografije za decentralizovanu valutu. Registar vodi evidenciju aktivnosti uz pravila koja određuju šta se može ili ne može uraditi da bi se izmenio registar. Ethereum, sa svojom kriptovalutom Ether, funkcioniše kao raspodeljena mašina stanja koja omogućava izvršavanje pametnih ugovora. Ethereum-ova virtuelna mašina (EVM) definiše pravila za promenu stanja iz bloka u blok. [5]

EVM se opisuje kao funkcija prelaska stanja Y(S, T) = S', gde S predstavlja staro stanje, T su transakcije, a S' je novo stanje. Ethereum koristi veliku strukturu podataka, modifikovani Merkle Patricia Trie, za čuvanje računa povezanih heševima.

Transakcije su kriptografski potpisane instrukcije od strane računa, a postoje dva tipa: one koje rezultuju pozivima poruka i one koje rezultuju kreiranjem ugovora. Ugovori sadrže Merkle Patricia skladišni trie i izvršavaju se kao niz EVM operacija koje obavljaju standardne operacije steka, poput XOR, AND, SABIRANJE, ODUZIMANJE. EVM takođe implementira blokčejm-specifične operacije steka, kao što su ADRESA, SALDO, BLOKHEŠ.

Implementacije EVM-a uključuju Py-EVM (Python), evmone (C++), ethereumjs-vm (JavaScript), eEVM (C++), revm (Rust), a sve moraju poštovati specifikaciju iz Ethereum Yellowpaper-a. Tokom devetogodišnje istorije Ethereuma, EVM je prošao kroz revizije, a različite implementacije postoje u različitim programskim jezicima.

**2.6 IPFS**

IPFS (InterPlanetary File System) je modularni set protokola za organizaciju i prenos podataka, osmišljen sa principima adresiranja sadržaja i peer-to-peer mrežnog povezivanja. IPFS ima više implementacija, ali njegova glavna svrha je decentralizovano objavljivanje podataka, uključujući datoteke, direktorijume i web lokacije. [6]

IPFS se sastoji od implementacije protokola, decentralizovane mreže čvorova i modularnog skupa protokola za organizaciju i prenos podataka. Ipak, važno je napomenuti da IPFS nije sam provajder za čuvanje podataka, niti provajder cloud usluga.

IPFS funkcioniše tako što podatke predstavlja blokovima sa jedinstvenim identifikatorom, poznatim kao Content Identifier (CID), koji se generiše kombinovanjem heša podataka sa njihovim kodikom. Za rad sa CID-ovima i adresiranim sadržajem, IPFS koristi InterPlanetary Linked Data (IPLD) i Merkle DAG (usmeren aciklički graf). Za adresiranje fajlova, direktorijuma i simboličkih veza, IPFS koristi UnixFS. [4]

IPFS koristi Kademliju, distribuiranu tabelu sa heširanim ključem (DHT), kako bi pronašao čvorove u mreži koji čuvaju tražene podatke. Kademlija je efikasan sistem otporan na promene čvorova. IPFS čvorovi koriste Bitswap, porukama baziran, peer-to-peer mrežni protokol za razmenu podataka.

Ukratko, IPFS omogućava decentralizovano objavljivanje podataka kroz adresiranje sadržaja, peer-to-peer mrežno povezivanje i efikasne protokole za organizaciju i prenos podataka.

**3.** **ARHITEKTURA SISTEMA**

Platforma je koncipirana kao decentralizovana aplikacija za veb 3.0, s jasno odvojenim klijentskim i serverskim segmentima koji ostvaruju međusobnu komunikaciju putem HTTPS protokola.

Obe aplikacije, kako klijentska tako i serverska, smeštene su na AWS oblaku – S3 servis koristi se za hostovanje statičkog veb-sajta, dok je backend razvijen kao serverless aplikacija (vidi Sliku 2).

A screenshot of a computer

Description automatically generated

*Slika 2. Dijagram arhitekture rešenja*

Na klijentskoj strani, aplikacija je implementirana pomoću React.js biblioteke, koja se izvršava u veb pregledaču. Takođe, unutar nje koristi se web3.js biblioteka za integraciju s digitalnim novčanicima i veb 3.0 provajderima (u ovom slučaju, MetaMask). Aplikacija takođe ostvaruje komunikaciju s IPFS (InterPlanetary File System) mrežom za skladištenje korisničkih datoteka i s čvorovima blokčejn mreže za pozivanje funkcija pametnih ugovora.

Poslovna logika serverskog dela ostvarena je Lambda funkcijama, koje komuniciraju s NoSQL bazom podataka DynamoDB. DynamoDB sadrži podatke o entitetima sistema koji nisu deo decentralizovane mreže. Kad se zahtev uputi serveru, prvo prolazi kroz API Gateway, koji na osnovu detalja iz zahteva okida odgovarajuću Lambda funkciju. Autorizacija i autentifikacija korisnika implementirane su putem Cognito servisa, koristeći Cognito User Pool koji sadrži sve informacije o registrovanim korisnicima. Za praćenje Lambda funkcija, u rešenje je integrisan AWS CloudWatch servis, koji prati i prikuplja metrike korišćenih resursa.

Centralna komponenta koja čini aplikaciju decentrali­zovanom jeste Ethereum blokčejn mreža, nad kojom se izvršavaju pametni ugovori. U ovom kontekstu, klijentska strana je odgovorna za komunikaciju s čvorovima Ethereum mreže. Pametni ugovori su kreirani i deployovani na Ethereum platformu pomoću Truffle radnog okvira.

**4.** **IMPLEMENTACIJA REŠENJA**

**4.1 Klijentska strana**

Za implementaciju klijentske strane korišten je Reakt.js (engl. React.js) radni okvir, zajedno sa veb3.js (engl. web3.js) skupom biblioteka koje omogućavaju integraciju sa digitalnim novčanicima. Na primer, kao što je MetaMask (engl. MetaMask) novčanik ugrađen u veb pretraživač, a omogućava da korisnici povežu svoj Eterijum nalog sa nalogom na sistemu i na taj omogući da korisnici interaguju sa pametnim ugovorima. MetaMask je Eterijum provajder (engl. provider).

**4.2 Truffle**

Truffle je razvojno okruženje i radni okvir za testiranje blokčejn sistema na Ethereum virtuelnoj mašini (EVM). Instalacija Truffle paketa, pisanje pametnih ugovora u Solidity jeziku, i konfiguracija Truffle projekta ključni su koraci u procesu postavljanja pametnih ugovora na Ethereum mrežu.

Pametni ugovori se pišu u Solidity jeziku, a zatim kompajliraju naredbom truffle compile. Skripte za postavku ugovora smeštene su u migrations/ direktorijumu, numerisane su i izvršavaju se kako bi ugovori bili postavljeni na mrežu. Konfiguracija se vrši u truffle-config.js datoteci, gde se koristi hdwallet-provider za potpisivanje transakcija.

Konačno, postavka pametnih ugovora na Sepolia test mrežu izvršava se komandom truffle migrate --network sepolia. Sve ove korake olakšava Truffle okruženje, omogućavajući brži razvoj i postavljanje pametnih ugovora na Ethereum mrežu. U nastavku sledi listing pametnog ugovora, kao i izvšavanje migracije:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

import "@openzeppelin/contracts/access/Ownable.sol";

import "@openzeppelin/contracts/utils/Counters.sol";

contract CryptoCert is ERC721URIStorage, Ownable {

    constructor() ERC721("Crypto Cert", "CRC") {}

    using Counters for Counters.Counter;

     Counters.Counter private \_tokenIds;

    function mintNFT(address recipient, string memory tokenURI)

        public onlyOwner

        returns (uint256)

    {

        \_tokenIds.increment();

        uint256 newItemId = \_tokenIds.current();

        \_mint(recipient, newItemId);

        \_setTokenURI(newItemId, tokenURI);

        return newItemId;

    }

}

*Listing 1. primer pametnog ugovora*

**4.3 Serverska strana**

Serverska strana sistema koristi Amazonove serverless servise, a ključni deo je čuvanje podataka na decentra­lizovanom skladištu IPFS (InterPlanetary File System). IPFS je odabran kao rešenje za čuvanje datoteka, uklju­čujući metapodatke NFT-ova (Non-Fungible Token) zbog efikasnosti i ograničenja kapaciteta skladištenja blok­čejna.

Većina decentralizovanih aplikacija na Ethereum platfor­mi kombinuje blokčejn i offchain skladište podataka poput IPFS-a. Datoteke i metapodaci o NFT-ovima čuvaju se na IPFS-u, dok se jedinstveni identifikatori (ID) tokena čuvaju na blokčejnu-u. Ovo poboljšava efikasnost mreže.

U radu se koristi kombinacija Golang i JavaScript klijentskih biblioteka za web3.storage servis. Kroz implementaciju funkcije ReceiveCertificate, sertifikati se obrađuju nakon završetka kursa. Slika sertifikata se preuzima sa S3 bucket-a, a zatim se kreira CID za sliku na Web3 Storage-u. Generišu se metapodaci za sertifikat, skladište na IPFS-u, i zatim se koristi za kreiranje NFT-a na frontend delu sistema. Na kraju, funkcija vraća URL gde se može pronaći finalni NFT sertifikat.

**5. ZAKLJUČAK**

Zaključak ovog rada sumira sveobuhvatno istraživanje i implementaciju blokčejn tehnologije, pametnih ugovora i decentralizovanih aplikacija. Kroz projekat Truffle, koji je poslužio kao osnova za razvoj Ethereum pametnih ugovora, istražena je serverless arhitektura koristeći Golang i AWS CDK, što naglašava prilagodljivost modernih tehnologija oblaka u podršci blokčejn aplikacijama.

Centralni deo istraživanja bio je fokus na korišćenju IPFS-a za skladištenje metapodataka i slika povezanih sa NFT sertifikatima u Web3 skladištu. Ovaj inovativni pristup obezbeđuje decentralizovanu i nepromenljivu prirodu podataka, integrišući efikasno IPFS i Web3 skladište. Kroz implementaciju frontend-a koristeći React i MetaMask, pružen je korisnički orijentisan pristup interakciji sa Ethereum blokčejnom.

U radu je naglašen značaj blokčejn tehnologije u stvaranju decentralizovanih i transparentnih sistema. Integracija Ethereum-a, IPFS-a, Web3 skladišta i AWS CDK-a u projektu predstavlja primer pristupa razvoju blokčejn aplikacija, kombinujući infrastrukturu blok lanca, logiku pametnog ugovora i interakciju frontend-a sa Ethereum platformom.

Ovaj rad ima za cilj da doprinese razumevanju i primeni aplikacija zasnovanih na blokčejnu, nudeći opipljiv plan za buduće projekte u brzo-razvijajućem okruženju decentralizovanih aplikacija.

# 6. LITERATURA

|  |  |
| --- | --- |
| [1] | N. Barney, “Distributed ledger technology (DLT) TechTarget”. |
| [2] | “Distributed Systems,” March 2018. [Online]. Available: http://www.dgma.donetsk.ua/docs/kafedry/avp/metod/van%20Steen%20-%20Distributed%20Systems.pdf. |
| [3] | S. K. и. H. G. H. Natarajan, “Distributed Ledger Technology and Blockchain,” FinTech Note, 2017. |
| [4] | N. Szabo, “Smart Contracts,” [Online]. Available: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html. [Accessed February 2024]. |
| [5] | ethereum.org, “EVM,” [Online]. Available: https://ethereum.org/en/developers/docs/evm/. |
| [6] | “How IPFS works,” [Online]. Available: https://ipfs.io/. [Accessed 2 2024]. |

**Kratka biografija:**

|  |  |
| --- | --- |
| A person with short hair wearing a black shirt  Description automatically generated | **Stefan Ljubović** rođen je u Subotici 1999. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2022.god.  kontakt: [ljubovicstefan@gmail.com](mailto:ljubovicstefan@gmail.com) |