Web stranica za unos i provjeru porijekla uroda masline na blockchain

Smok, Ivan

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

WEB STRANICA ZA UNOS I PROVJERU PORIJEKLA URODA MASLINE NA BLOCKCHAIN

Završni rad br. 08/RAČ/2022

Ivan Smok

Bjelovar, listopad 2022.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Ivan Smok** JMBAG: **0314022043**

Naslov rada (tema): **Web stranica za unos i provjeru porijekla uroda masline na blockchain**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Ivan Sekovanić, mag.ing.inf.et comm.techn.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **dr.sc. Zoran Vrhovski, predsjednik**
2. **Ivan Sekovanić, mag.ing.inf.et comm.techn., mentor**
3. **Krunoslav Husak, dipl. ing. rač., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 08/RAČ/2022

U sklopu završnog rada potrebno je:

1. Opisati razvojni okvir za izradu web stranice potrebne za unos podataka i provjeru podataka
2. Analizirati i opisati mogućnost pohrane podataka na prvi sloj blockchaina
3. Predložiti i opisati pozadinsku (engl. backend) programsku podršku za komunikaciju s blockchainom
4. Izraditi i opisati web stranicu za unos i provjeru porijekla uroda masline
5. Izraditi pozadinsku (engl. backend) programsku podršku za pohranu i dohvaćanje podataka s blockchaina

Datum: 29.08.2022. godine

Mentor: **Ivan Sekovanić, mag.ing.inf.et comm.techn.**

REPUBLIKA HRVATSKA 2 VELEUČILIŠTE U BJELOVARU

Sadržaj

1. UVOD

Sigurnost podataka na mreži danas je jedna od najvećih sigurnosnih briga za poslužitelje usluga na internetu te njihove korisnike. Nažalost niti jedan sustav nije 100% siguran, ali zato svaki od njih ima svoje prednosti. Ovaj projekt prikazuje web aplikaciju koja koristi dva sasvim različita sustava za pohranu podataka te kako ih koristi s ciljem postizanja dodatne sigurnosti spremljenih podataka.

Prvi sustav, koji je također i najrasprostranjeniji u svijetu, je centralizirana baza podataka. Centralizirana baza podataka je središnje računalo na kojoj se nalaze svi podaci vezani za neku aplikaciju, na primjer svi članci neke web stranice. Mana takve baze podataka je da ako netko dođe do ovlasti korištenja računala s bazom podataka, ta osoba ima svu kontrolu nad korištenjem i manipuliranjem podataka u bazi. Zbog toga centralizirane baze podataka zahtijevaju veliku sigurnost od upada neželjenih korisnika.

S druge strane, postoje distribuirane baze podataka koje nemaju jedno centralno računalo sa svim podacima nego su podaci raspoređeni na više njih. To se može izvesti na više načina. Prvi je da su podaci raspodijeljeni na više dijelova te svako računalo ima samo jedan od tih dijelova podataka. A drugi način je da svako računalo koje je dio određene distribuirane baze podataka ima iste podatke što otežava krivotvorenje podataka u bazi. Takav primjer distribuirane baze podataka može se primijetiti u blockchain tehnologiji.

Rad je podijeljen na tri poglavlja. Prvo poglavlje govori o tehnologijama korištenim za izradu projekta. Drugo poglavlje detaljnije opisuje blockchain tehnologiju, govori o njenim značajkama te opisuje Syscoin mrežu, to jest njene značajke, prednosti i moguće opasnosti. Treće poglavlje pokazuje korištenje tih tehnologija na praktičnom projektu uz prikaz slika i programskog koda kao primjere. Te se nakon toga može pogledati zaključak koji ukratko opisuje svrhu izrade ovoga projekta.

2. KORIŠTENE TEHNOLOGIJE

Ovaj projekt je podijeljen na tri glavna dijela. To su baze podataka za spremanje podataka, aplikacijsko programsko sučelje za lakši rad s navedenim podacima te grafičko sučelje za ljepši prikaz podataka.

2.1. Baze podataka

2.1.1. Microsoft SQL Server

Microsoft SQL Server je sustav za upravljanje relacijskim bazama podataka. Pomoću njega je napravljena baza podataka koja sadrži tablicu sa entitetima koji se koriste u ovom projektu. Kao što mu je u imenu, Microsoft SQL server koristi SQL programski jezik za upravljanje podacima. SQL je jedan od starijih programskih jezika, nastao 1974. godine, koji se i dan danas koristi. Od tada je nastalo nekoliko novih dijalekata SQL-a i nastavaka za njega što se može vidjeti prema [1]. Prema [2], Microsoft SQL Server koristi Transact-SQL (T-SQL) nastavak. Stoga on nije programski jezik sam po sebi, već je samo nadogradnja na SQL.

2.1.2. Blockchain

Za razliku od centraliziranih sistema, svrha blockchain-a je postići skroz suprotno, a to je distribuiranost. Blockchain to postiže tako da se jedno administrativno tijelo koje upravlja podacima zamijeni mnoštvom računala povezanih preko mreže koja konstantno komuniciraju. Svako od tih računala mora biti sinkronizirano sa ostalim računalima kako bi moglo sudjelovati u blockchain mreži. Dok god računalo nije sinkronizirano sa ostatkom mreže, ono se gleda kao nevaljano.

2.2. Aplikacijsko programsko sučelje

2.2.1. Python

Python je jedan od najlakših programskih jezika za naučiti te ima široku primjenu koja može biti od izrada web stranice do učenja umjetne inteligencije. No, u ovom projektu Python je korišten za izradu aplikacijskog programskog sučelja s ulogom da se omogući lakše upravljanje podacima iz baza podataka.

2.2.2. Flask

Flask je Python aplikacijsko okruženje s namjenom za izradu web aplikacija čija se dokumentacija može naći u literaturi [3]. Ovaj projekt ne koristi Flask za kompletnu izradu web aplikacije nego se koriste neke njegove funkcionalnosti za omogućavanje komunikacije web aplikacije sa bazama podataka.

2.2.3. RPC

RPC (engl. *Remote procedure call*) je pojam koji opisuje pozivanje neke procedure na udaljenom mjestu. U ovom projektu ovaj tip izvršavanja koda se koristi za komunikaciju sa blockchain mrežom. Korištenjem Flask aplikacijskog sučelja uspostavlja se RPC konekcija sa Syscoin Core aplikacijom koja zatim izvršava procedure za spremanje i dohvaćanje podataka sa blockchain-a.

2.3. Grafičko sučelje

Za izradu grafičkog sučelja korištene su tri vrste tehnologija: HTML, SCSS i TypeScript. Te tri tehnologije su objedinjene Angular aplikacijskim okruženjem. Angular koristi, CSS/SCSS za vizualnu izmjenu navedenog dokumenta kao što je promjena veličine, boje i font teksta te još mnogo drugih opcija. Te na kraju koristi TypeScript, podvrstu JavaScript programskog jezika, koji se koristi za manipuliranje tokom aplikacije (promjena URL-a, dohvaćanje podataka i tako dalje).

2.3.1. HTML

HTML je jezik koji se koristi za pisanje hipertekst dokumenata. On sam po sebi nije programski jezik nego jedina namjena mu je strukturirano sadržavati tekst unutar hipertekst dokumentima. Zatim je zadaća web preglednika da to čita te hipertekst dokumente i prikaže ih korisniku na pregledniji način. Budući da postoji mnogo web preglednika, potrebno je da svaki od njih hipertekst datoteke prikazuje na isti način. HTML je neophodan za prikaz podataka na web preglednicima. I u slučaju otvaranja najobičnije .txt datoteke unutar web preglednika, on će te podatke smjestiti unutar HTML znakova.

2.3.2. SCSS

SCSS je naprednija verzija CSS-a, to jest on je predprocesorski skriptni jezik koji se koristi za dizajniranje web stranica.

Neke prednosti SCSS-a su što dopušta ugniježdenu sintaksu, ima više matematičkih funkcija, omogućuje izradu varijabli korištenje istih vrijednosti kroz cijeli projekt, također pruža @import i @export funkcije pomoću kojih se mogu koristiti klase iz drugih SCSS datoteka bile one lokalno unutar projekta ili na mreži. Kompatibilan je sa CSS-om što znači da se mogu koristiti bilo koje CSS biblioteke unutar SCSS datoteka.

No, ima i svoje nedostatke, kao što je sama potreba za prevođenjem u CSS, što može dovesti do povećanja same datoteke. SCSS datoteka može biti mnogo manja od CSS datoteke koja nastane prevođenjem.

2.3.3. TypeScript

TypeScript programski jest jezik koji je nastao od JavaScript-a. JavaScript je jedan od najkorištenijih programskih jezika te se najčešće koristi za izradu web aplikacija i web servera. Za razliku od JavaScript-a, TypeScript, kao što mu ime govori, pruža potporu za prepoznavanje tipa podatka u kodu.

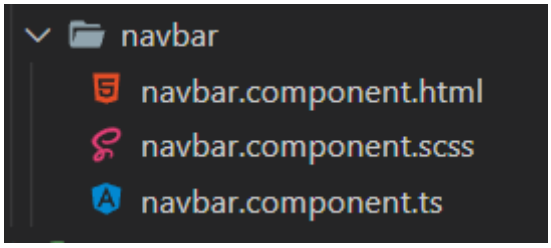
U JavaScriptu varijabla koja se inicijalizira kao broj 1, kasnije u kodu se može promijeniti u tekstualnu vrijednost. Znanje o podacima s kojima programer radi je jedan načina da izbjegne buduće greške koje se mogu dogoditi. Osim što TypeScript podržava tipove podataka, također uvodi tipove kao što su sučelja (engl. *interface*) i klase iz objektno-orijentiranog programiranja.

2.3.4. Angular

Angular je aplikacijsko okruženje napravljeno za izradu web aplikacija čija se dokumentacija može pročitati na [4]. Nastao je na temelju AngularJS-a, starijeg aplikacijskog okruženja čiji je razvoj zaustavljen.

Angular koristi gore navedene tehnologije: HTML, CSS/SCSS i TypeScript. Projekti napravljeni pomoću Angular-a se temelje na izradi takozvanih komponenti. Komponente su Angular-ovi primarni dijelovi za izradu web stranice. Njihova svrha je raspodijeliti projekt na manje dijelove. To omogućuje lakši rad na velikim projektima jer je se lakše snaći u strukturi projekta budući da ga se može razdijeliti na dosta manjih dijelova. Rad sa puno većim brojem ljudi je također olakšan jer se tim može raspodijeliti da svatko radi na određenoj komponenti što također smanjuje broj pogrešaka pri spajanju grana dok se koristi Git tehnologija.

Pri kreiranju komponente ona se sastoji od tri dijela: HTML, CSS/SCSS (ovisno o tome što je odabrano da će se koristiti pri kreiranju projekta) i TypeScript datoteke (slika 2.1).



Slika 2.1: Primjer strukture Angular komponente

Prvi dio, koji definira HTML strukturu te komponente, je HTML datoteka. To na primjer može biti HTML koji definira samo prikaz navigacijske trake. Drugi dio je CSS/SCSS datoteka. Ta CSS/SCSS datoteka može utjecati na dizajn cijele stranice, ne samo te komponente. Stoga je dobro unutar te datoteke dati naznaku da se radi samo o toj komponenti pomoću id-a, klase ili Angular-ovih opcija koji definiraju HTML te komponente. I na kraju, TypeScript datoteka se ponaša kao kontroler te komponente, to jest ona upravlja njenim tokom kao što je upravljanje podacima koji se prikazuju unutar HTML datoteke, može mijenjati koji HTML elementi se prikazuju te mnogo drugih stvari. No, najbitnija stvar koju radi je da te tri datoteke spaja u jednu cjelinu što se može vidjeti u programskom kodu 2.1.

Programski kod 2.1: Spajanje HTML-a i SCSS-a s TypeScript dijelom komponente

```
@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.scss'],
})
```

3. BLOCKCHAIN

Blockchain je vrsta tehnologija koja se zasniva na principu distribuirane baze podataka. Ta tehnologija se naziva DLT (engl. *distributed ledger technology*). Ne postoji jedno administrativno tijelo koje upravlja svim podacima već je riječ o mnoštvu umreženih računala koja međusobno komuniciraju.

Na blockchain mrežu podaci se ne spremaju jedan po jedan kao u običnim bazama podataka. U blockchain tehnologiji podaci se slažu unutar blokova. Svaki blok ima maksimalnu veličinu podataka koju može sadržavati. Osim toga, svaki blok podataka spremljen na blockchain mrežu je povezan sa prethodnim i sljedećim blokom, od tuda je i došao naziv blockchain.

3.1. Čvorovi

Svako računalo koje je dio blockchain mreže naziva se čvor. Čvorovi se dijele u tri skupine: cjeloviti čvorovi (engl. *full nodes*), djelomični čvorovi (engl. *light nodes*) i čvorovi zaduženi za pronalazak ispravnih blokova podataka koji će se zapisati u mrežu (engl. *miners*).

3.1.1. Cjeloviti čvor

Srž blockchain tehnologije se zasniva na cjelovitim čvorovima. Oni su ti koji omogućuju distribuiranost podataka. Zbog toga ovi čvorovi su računala sa većim prostorom na tvrdom disku kao što su stolna te prijenosna računala ili poslužitelji. Oni trebaju tvrde diskove sa što više prostora jer svi podaci neke blockchain mreže se nalazi na njima. Svako računalo na sebi ima isti set podataka. U slučaju da neko od računala ima drugačije podatke od ostatka mreže ili da mu fali podataka, ono će se gledati kao nevaljano sve dok se ne sinkronizira sa ostatkom mreže. Takav način rada omogućuje veliku sigurnost podataka jer ako netko pokuša izmijeniti podatak na blockchain mreži, ta mreža to odbija. Osim ako preko 50% računala u mreži istovremeno promijeni taj podatak u istu vrijednost što je gotovo nemoguće da se dogodi zbog same količine računala koje sudjeluju u mreži.

Jedna od mana blockchain tehnologije koja se pojavljuje u tome području je veličina tvrdih diskova. Ako se ne izume puno veći tvrdi diskovi uz prihvatljive cijene koje svakodnevni čovjek može priuštiti, sve manje računala će moći biti dio blockchain mreže čime se smanjenje

distribuiranost. Osim toga, veličina blokchain mreže može postati i veća od standarda veličina tvrdih diskova što je još jedna od prijetnji razvoju blockchain tehnologije.

3.1.2. Djelomičan čvor

Djelomični čvorovi su manja računala koja nemaju jako velik prostor na tvrdom disku. To su inače mobiteli i slični uređaji. Oni imaju samo dio podataka zapisanih na blockchain mreži. Oni moraju dosta komunicirati sa cjelovitim čvorovima, na primjer oni dohvaćaju podatke s blockchain mreže. Također ako korisnik želi obaviti transakciju, upis tih podataka se obavlja tako da se obavijesti cjeloviti čvor o željenoj transakciji.

3.1.3. Čvor zadužen za pronalazak ispravnih blokova

Čvorovi zaduženi za pronalazak ispravnih blokova koriste se za pisanje podataka unutar blockchain mreže. Ovi čvorovi uzimaju popis transakcija, od cjelovitih čvorova, koje treba obaviti. Kada se transakcija spremi u blok i kada se on objavi tek tada se transakcija tretira obavljenom. Svaki blok ima određenu veličinu ovisno o kojoj blockchain mreži je riječ, ali generalno mogu biti do nekoliko megabajta. Kada ovaj čvor uspije generirati hash kod novog bloka, tek tada se ti podaci upisuju na mrežu. Kada je to sve uspješno odrađeno, čvor dobije novčanu naknadu za dodavanje novog bloka sa podacima. Ta novčana naknada se izdaje u kriptovaluti koja se koristi unutar te mreže.

3.1. Značajke blockchain tehnologije

Blockchain tehnologija ima sloj sigurnosti koji se zasniva na svojoj distribuiranosti. Podaci na blockchainu su praktički neizmjenjivi. Jednom kada se podaci spreme u blockchain mrežu, to jest izvrši se transakcija, ti podaci su teoretski neizmjenjivi zbog količine računala koja imaju isti podatak. U blockchain-u većina uvijek pobjeđuje.

3.1.1. Kriptografija

Kod razmatranja sigurnosti na blockchain-u, dolazi se do pojma zvanog kriptografija. Kriptografija je proces izmjene podataka, izmjena se može dogoditi na dva načina: šifriranje i

dešifriranje. Šifriranje je proces korištenja matematičkih algoritama kako bi od originalnog teksta dobili šifrirani tekst, a dešifriranje je suprotno, od šifriranog teksta se dobiva originalan tekst.

Postoje dvije glavne vrste šifriranja, a to su simetrično i asimetrično šifriranje. Oba načina šifriranja koriste ključeve za šifriranje i de šifriranje. Glavna razlika je što simetrično šifriranje koristi jedan tajni ključ koji se koristi za šifriranje i de šifriranje, a asimetrično šifriranje koristi dva ključa. Javni ključ koji je svima dostupan za šifriranje poruke i privatni ključ koji se koristi za dešifriranje poruke. Simetrično šifriranje je brže, no ima manju sigurnost od asimetričnog jer je potrebno primatelju poruke dati tajni ključ što može vidjeti treća strana koja prisluškuje promet. Dok s druge strane asimetrično šifriranje je sporije, no ima veću sigurnost budući da podaci šifrirani javnim ključem se ne mogu de šifrirati tim istim ključem već je potrebno imati privatni ključ kako bi poruku vratili u originalan tekst. To znači da vlasnik privatnog ključa taj ključ ne smije nikome davati.

Od ta dva načina šifriranja, blockchain koristi asimetrično šifriranje zbog veće sigurnosti. Budući da je javni ključ svima dostupan, bilo tko ga može koristiti. Zbog toga su uvedeni i digitalni potpisi. Digitalan potpis je način na koji se može dokazati integritet podataka i identitet pošiljatelja. On se generira pomoću hash-a nastalog od podataka na koji se primjenjuje privatni ključ korisnika. Primatelj zatim koristi javni ključ pošiljatelja kako bi dobio hash vrijednost podataka iz digitalnog potpisa i to uspoređuje sa hash vrijednosti podataka koji si pristigli transakcijom. Ako su te dvije hash vrijednosti iste, to znači da nitko nije mijenjao podatke koji se šalju te potvrđuje identitet pošiljatelja.

Korištenjem tih dviju tehnologija na blockchain se mogu spremiti podaci kao što su vlasništvo nad nečime, ugovori, certifikati, novčane transakcije i slične stvari. S velikom sigurnošću se može reći da ti podaci neće biti neželjeno promijenjeni od treće strane, a i vrlo se lako može dokazati izvor tih podataka, to jest tko je sudjelovao u transakciji.

Podaci šifrirani danas će doći u opasnost zbog velike razlike u tehnologiji koju donose superračunala. S današnjim računalima, dešifriranje neke hash vrijednosti bez matematičke funkcije i ključa za dešifriranje je gotovo nemoguće. Trajalo bi stotinama godina, no snagom superračunala to vrijeme se znatno smanjuje. Čak i da traje mjesec dana to može biti jako isplativo ako se dešifrira određene podatke.

3.2. Syscoin

Syscoin je jedna od mnogih blockchain mreža. Ona koristi aspekte iz Bitcoin i Ethereum mreže. Uputstva za korištenje Syscoin mreže mogu se pronaći u Syscoin dokumentaciji [5].

Syscoin iz Bitcoin mreže se uzima PoW (engl. Proof of Work) protokol. Uloga PoW-a je osigurati podatke tako da sprječava njihovo lažiranje i izmjenu. To se postiže pomoću čvorova zaduženih za pronalazak ispravnih blokova. Ti čvorovi trebaju generirati hash vrijednost za svaki novi blok sa određenim uvjetima, kao što je uvjet da hash vrijednost mora počinjati sa određenim brojem nula. Taj broj nula ovisi o broju čvorova koji pokušavaju generirati hash vrijednost bloka, što je veći broj čvorova to je i veći broj nula koje trebaju predvoditi hash vrijednost. Osim toga, generalno se uzima još jedna sigurnosna mjera protiv lažiranja podataka na blockchain-u. A to je da se blok računa kao validan nakon što se generira još nekoliko blokova nakon toga bloka. Time ako netko pokuša lažirati podatke na blockchain mreži treba uspješno izračunati hash trenutnog bloka i nekoliko sljedećih blokova. No, to je gotovo nemoguće zbog količine drugih računala koje također računaju hash vrijednosti novih blokova.

Syscoin iz Ethereum mreže koristi princip korištenja virtualnog stroja. Syscoin mreža koristi mrežno pojačani virtualni stroj. Razlike između Syscoin-ovog i Etherum-ovog virtualnog stroja je što Syscoin pruža PoW protokol za dodatnu sigurnost dok i dalje pruža skalabilnost pametnih ugovora.

Syscoin, kao i druge blockchain mreže, dijeli se na dva dijela. Jedna mreža gdje se izvršavaju prave transakcije i jedna mreža koja se koristi za testiranje.

3.2.1. Mainnet

Mainnet je glavna mreža Syscoin blockchain-a. Na mainnet-u se koristi prava kriptovaluta korištene blockchain mreže. Kod Syscoin mreže je to SYS žeton. Ti žetoni se mogu izmijeniti za fizički novac na tržištu. Vrijednost tih žetona može se drastično mijenjati zbog čega se danas dosta koriste kao ulaganje. Prema [6] vrijednost SYS žetona 2. siječnja 2022. $1.24 USD, a 9. listopada 2022. bila je oko $0.16 USD. To je drastičan pad od oko 87% u devet mjeseci što se također može vidjeti na slici 3.1.

Slika 3.1: Crypto.com graf sa vrijednošću SYS žetona

Svrha mainnet-a je korisnicima pružiti stabilno i sigurno okruženje za izvršavanje transakcija čime mreža dokazuje svoju kredibilnost. Što više korisnika vjeruje u neku blockchain mrežu, time njena vrijednost što više raste zbog veće sklonosti takvih korisnika da ulažu u žetone. No, to ponekad ne znači da je ta mreža kredibilna. Blockchain mreža se ne može smatrati kredibilnom samo zbog velikog broja korisnika, drugi jako bitan parametar je njen životni vijek.

Čak i ako se blockchain mreža smatra kredibilnom, to ne znači da ne postoje opasnosti na njoj. Također se treba razmatrati kredibilnost kriptovalute, odnosno žetona. Danas su Bitcoin (BTC) i Ethereum (ETH) žetoni najkorištenije kriptovalute. S velikom sigurnošću se može reći da su to kredibilni žetoni. No zbog prirode blockchain mreže, ne može se reći da su stabilne financijske opcije. Tako da je preporučeno dobro se informirati o žetonima koji se kupuju. Pogotovo ako je žeton poprilično nov, na primjer mlađi od godinu dana ili pak malo više. Ima dosta primjera gdje se pojavi novi žeton na kredibilnoj blockchain mreži, na primjer Ethereum, no vrijednost žetona vrlo brzo opadne nakon što je pušten u mrežu. Ima više razloga zašto se to može dogoditi. Jedan od njih je da ljudi jednostavno nisu zainteresirani stoga vrijednost žetona značajno padne naspram početne vrijednosti na koju je žeton postavljen. A zatim postoji pojam zvani povlačenje tepiha (engl. *Rug Pull*).

Rug Pull je pojam koji se koristi u blockchain svijetu kada je riječ o prevari gdje netko „povuče tepih“ i uzme sve što su imali korisnici koji su „stajali na tepihu“. Kada se kreira novi žeton na blockchain mreži, on ima neku početnu vrijednost te se pusti određen broj žetona u cirkulaciju koje korisnici mogu kupiti i prodavati. U *Rug Pull* tehnici osnivaču se također dodijeli

popriličnо velik broj žetona. Jedan od načina prevare je da osnivač zatim koristeći društveni inženjering (na primjer uvjeri ljude da će to biti uspješan žeton) ili neki proizvod (na primjer video igra) stvara uzbuđenje oko žetona i privlači masu ljudi koji zatim kupuju žeton. Time vrijednost žetona raste i u određenom trenutku osnivač prodaje sve svoje žetone čime zarađuje veliku svotu novca, a vrijednost žetona opada. Osim toga, još jedan od načina izvođenja *Rug Pull* prevare je da programer/i koji su napravili žeton u kodu naprave promijene koje omogućuju da samo oni mogu prodati žetone.

Postoji zaštita protiv toga zvana mehanizam protiv kitova (engl. *Anti-Whale Mechanism*). Na kripto tržištima kitovi su osobe koje posjeduju veliku količinu žetona, dovoljno veliku da mogu utjecati na vrijednost kriptovalute koje posjeduju. U *Rug Pull*-u je to osnivač žetona i/ili programer koji ga je napravio. *Anti-Whale Mechanism* se može implementirati na više načina. Jedan od njih je da korisnik ne može prodati žetone neko vrijeme nakon što ih je kupio, na primjer par mjeseci. Drugi način na koji se može implementirati je da korisnik ne može prodati sve svoje žetone od jednom nego je limitiran da može prodati određen postotak žetona u određenom vremenskom periodu. Time se sprječava da jedno ili nekoliko centralnih tijela profitira prodajom svojih žetona i narušavaju vrijednost žetona.

Zbog toga je dobro istražiti žeton prije nego ga se kupuje, a i blockchain mrežu na kojoj se on nalazi.

3.2.2. Testnet

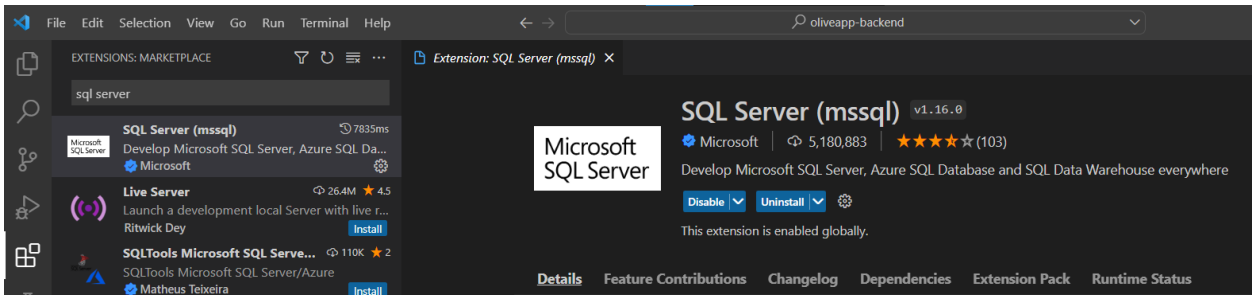
Testnet je testna mreža Syscoin blockchain-a. Njena glavna svrha je testiranje napravljenih promjena i dodataka na mreži prije nego se puste u produkciju, odnosno na mainnet. Zbog toga žetoni korišteni na testnet mreži nemaju pravu novčanu vrijednost jer nije riječ o pravom tržištu.

Testnet je također jako koristan za programere jer ga mogu koristiti za kreiranje svojih aplikacija koje koriste blockchain tehnologiju. Programeri blockchain mreže time imaju korisnike koji isprobavaju promjene koje se planiraju uvesti u glavnu mrežu, dok s druge strane programeri imaju besplatan način da testiraju svoju aplikaciju kako će raditi sa blockchain mrežom. Zbog toga je ovaj projekt je rađen korištenjem testnet mreže.

4. PRIKAZ PROJEKTA

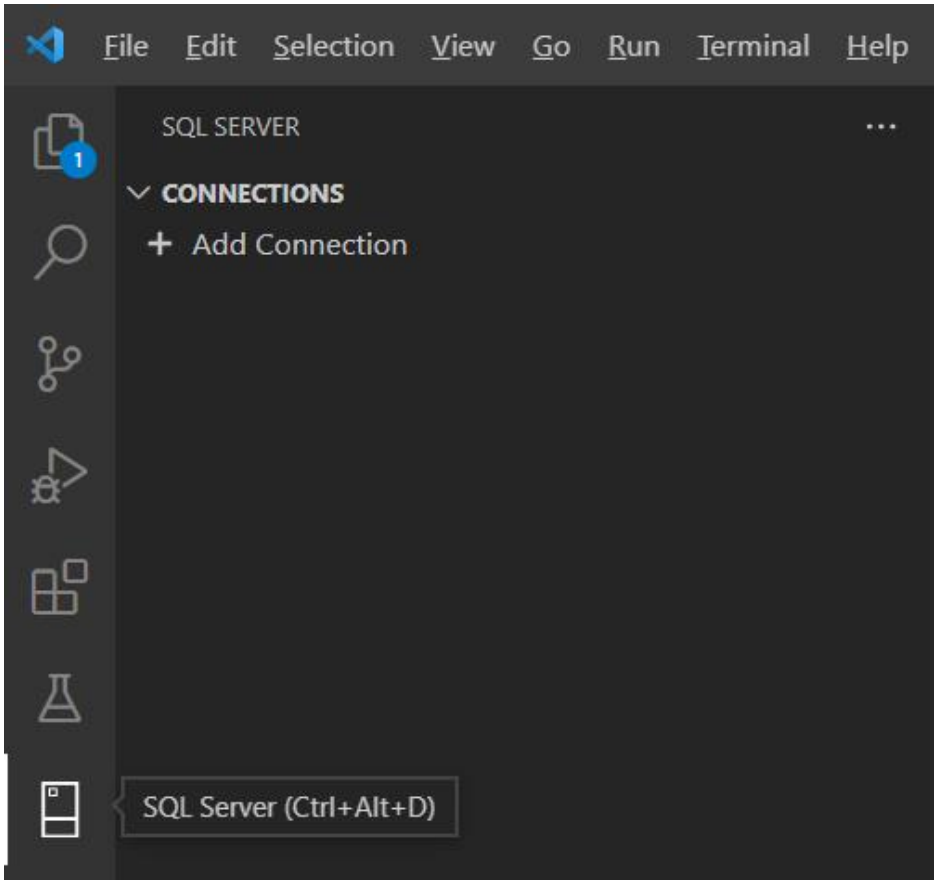
4.1. Izrada baze podataka

Izrada baze podataka i tablice ovog projekta napravljena je korištenjem SQL Server nastavka unutar Visual Studio Code programa. To se može vidjeti na slici 4.1.



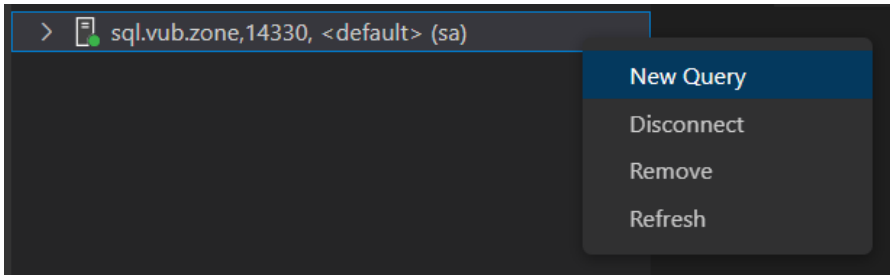
Slika 4.1: Prikaz mjesta instalacije SQL Server nastavka

Instalacijom tog nastavka u navigacijskoj traci sa lijeve strane se pojavljuje nova opcija koja se može vidjeti na slici 4.2.



Slika 4.2: Mogućnost upravljanja SQL serverom

Tim nastavkom se zatim spaja na Microsoft SQL Server. Nakon uspješne konekcije na server, dobiva se mogućnost izvršavanja SQL upita prema njemu (slika 4.3). U ovom slučaju je riječ o serveru kojega koristi više ljudi stoga je potrebno paziti koji upiti se upisuju kako se ne bi oštetio tuđi rad.



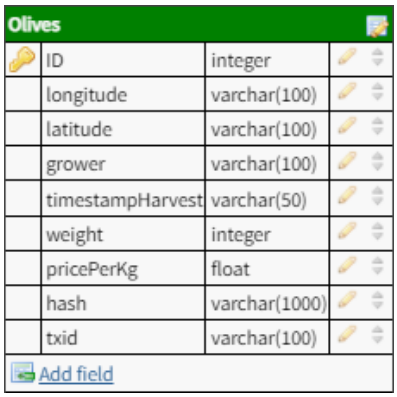
Slika 4.3: Stvaranje novog upita prema serveru

Prvo je napisan upit za izradu baze podataka što se može vidjeti u programskom kodu 4.1.

Programski kod 4.1: Upit za kreiranje baze podataka na serveru

```
CREATE DATABASE ismok_final;
```

Nakon toga je osmišljen model tablice koja će se kreirati unutar baze podataka. Za to je korištena DB DESIGNER web stranica, a kreirani model se može vidjet na slici 4.4.



Slika 4.4: Model tablice kreiran pomoću DB DESIGNER web aplikacije

Nakon toga je napisan upit koji kreira tablicu prema prethodno navedenom modelu unutar ismok_final baze podataka što se može vidjeti u programskom kodu 4.2.

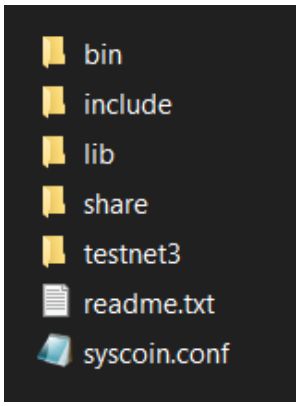
Programski kod 4.2: Upit za kreiranje Olives tablice na serveru

```
CREATE TABLE [ismok_final].[dbo].[Olives] (
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [longitude] [varchar](100) NOT NULL,
    [latitude] [varchar](100) NOT NULL,
    [grower] [varchar](100) NOT NULL,
    [timestampHarvest] [varchar](20) NOT NULL,
    [weight] [int] NOT NULL,
    [pricePerKg] [float] NOT NULL,
    [hash] [varchar](64) NOT NULL,
    [txid] [varchar](64) NOT NULL
)
```

4.2. Korištenje Syscoin-a

4.2.1. Konfiguracija

Kako bi se omogućila komunikacija sa Syscoin mrežom prvo je potrebno konfigurirati Syscoin Core aplikaciju, za ovog izradu projekta je korištena 4.4.0rc9 inačica aplikacije. Na slici 4.5 je prikazan sastav syscoin-core4.4.0rc9 mape.



Slika 4.5: Sastav mape Syscoin Core 4.4.0rc9 aplikacije

Kao što se može vidjeti, zadnja datoteka unutar te mape je syscoin.conf datoteka. To je konfiguracijska datoteka u kojoj se trebaju namjestiti postavke kako će se aplikacija ponašati. Sadržaj te datoteke je prikazan na slici 4.6.

```
testnet=1
[test]
zmqpubnevm=
rpcuser=user
rpcpassword=password
listen=1
daemon=1
server=1
assetindex=1
port=18369
rpcport=18370
rpcallowip=127.0.0.1
gettestnet=1
addnode=54.203.169.179
addnode=54.190.239.153
```

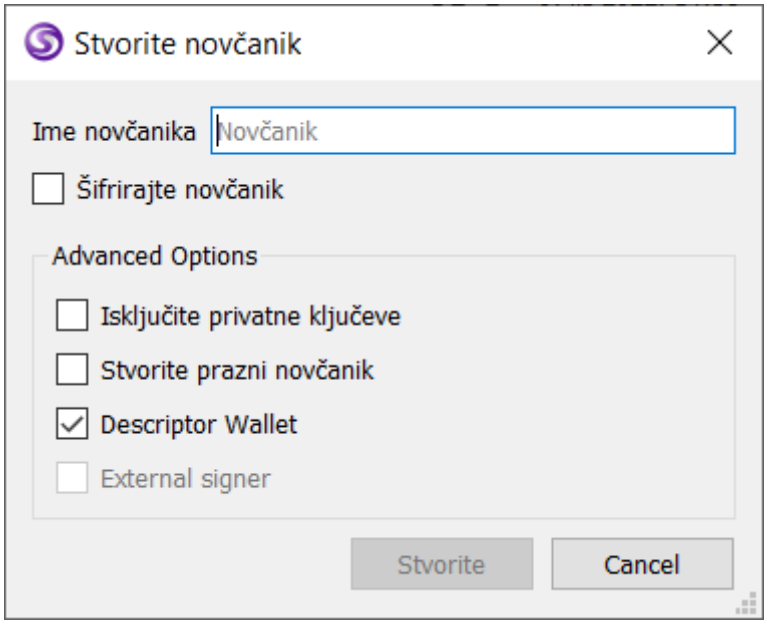
Slika 4.6: Sadržaj syscoin.conf datoteke

Od svih tih opcija potrebno je obratiti pažnju na testnet=1 koja Syscoin Core aplikaciji naznačuje da umjesto mainnet mreže koristi se testnet mreža. I druge opcije na koje se treba obratiti pažnja su: rpcuser, rpcpasswrod, rpcport i rpcallowip. Navedene postavke će se koristiti pri izvršavaju RPC-a.

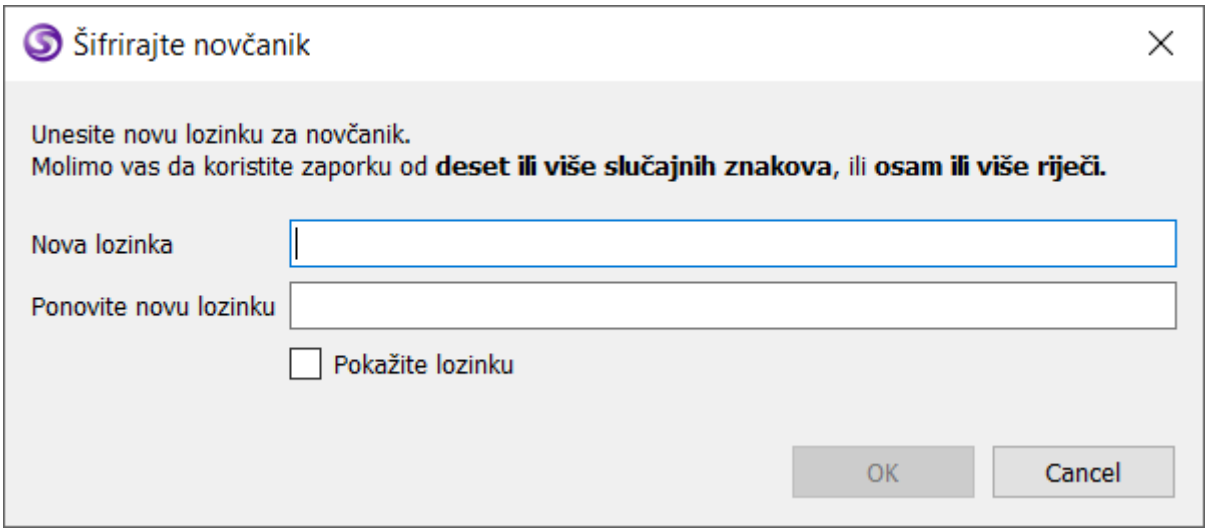
4.2.2. Novčanik

Kako bi se omogućilo spremanje podataka na mrežu potrebno je kreirati novčanik što se može vidjeti na slici 4.7. Novčanik korišten za ovaj projekt je također šifriran (slika 4.8) što omogućuje da ne može bilo tko koristiti taj novčanik već je potrebno korištenje zaporke. Prije nego se izvršava transakcija prema blockchain-u potrebno je otključati novčanik sa odabranom zaporkom. Pri otključavanju novčanika treba se odrediti na koliko dugo taj novčanik bude otključan što je vrlo korisna opcija kada se radi o ovako automatiziranoj radnji koja se obavlja kroz kod kao što je u projektu prikazano. Na taj način se novčanik može otključati na određen broj sekundi da se može izvršiti transakcija nakon čega se novčanik vraća u zaključano stanje.

Nakon toga se stvara primateljska adresa na koju je dodano Syscoin testnih žetona. Ti žetoni će se koristiti za plaćanje transakcija odnosno spremanje podataka na testnet mrežu.



Slika 4.7: Forma za kreiranje novčanika u Syscoin Core aplikaciji



Slika 4.8: Postavljanje lozinke pri šifriranju novčanika

4.3. Funkcije za upravljanje bazama podataka

Nakon izrade baze podataka i tablice u njoj te konfiguracije Syscoin Core aplikacije napravljena je aplikacija u Python programskom jeziku koja omogućuje web aplikaciji da komunicira sa Syscoin Core aplikacijom.

Najprije se u aplikaciji definira nekoliko globalnih varijabli koje će se koristiti kroz kod. To se može vidjeti u programskom kodu 4.3, no neke od vrijednosti su sakrivene.

Programski kod 4.3: Postavljanje globalnih varijabli

```
# rpcuser and rpcpassword from syscoin.conf file
rpc_user, rpc_password = "user","password"

# wallet_passphrase
wallet_passphrase = "***"

# Database connection string
conn_str ="Driver={SQL Server};SERVER=***;DATABASE=ismok_final;UID=***;PWD=***"
```

Kao što se može vidjeti u programskom kodu 4.3 se pojavljuju rpcuser i rpcpassword vrijednosti koje su postavljene unutar syscoin.conf datoteke.

4.3.1. Komunikacija s blockchain-om

Komunikacija s blockchain-om odvija se pomoću dvije funkcije. Jedna pomoću koje se podaci zapisuju na blockchain i druga pomoću koje se podaci dohvaćaju sa blockchain-a. Te funkcije zahtijevaju RPC konekciju što se omogućava programskim kodom 4.4.

Programski kod 4.4: Uspostavljanje RPC konekcije

```
rpc_conn =
AuthServiceProxy("http://%s:%s@127.0.0.1:18370/wallet/Smok"%(rpc_user,rpc_passwor
d))
```

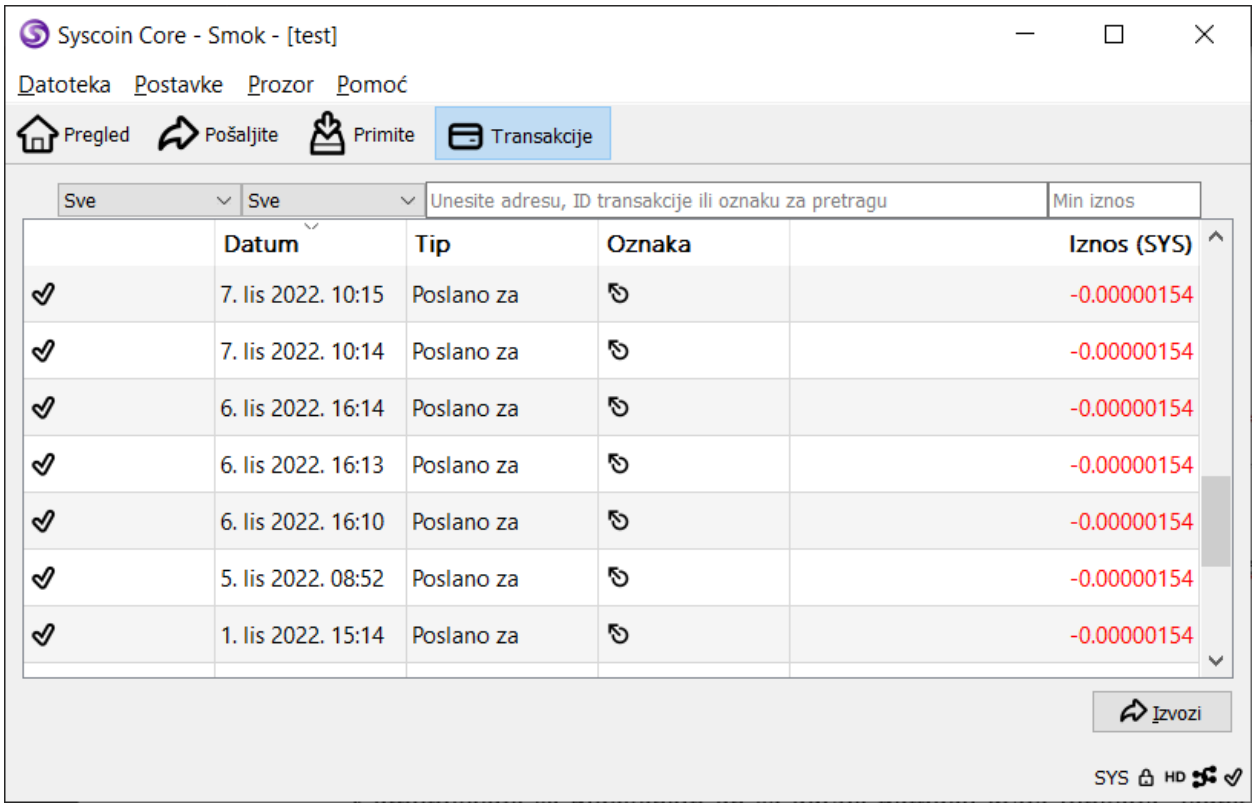
U programskom kodu 4.4 se također može vidjeti par konfiguracija iz syscoin.conf datoteke kao što je rpcallowip i rpcport. Osim toga također se može vidjeti da se konekcija uspostavlja preko kreiranog novčanika.

Programski kod 4.5: Funkcija za upis podataka na blockchain

```
rpc_conn.walletpassphrase(wallet_passphrase, 10)
craw = rpc_conn.createrawtransaction([], [{ 'data': hash }])
fraw = rpc_conn.fundrawtransaction(craw)
sraw = rpc_conn.signrawtransactionwithwallet(fraw['hex'])
sendraw = rpc_conn.sendrawtransaction(sraw['hex'])

del rpc_conn
return sendraw
```

Programski kod 4.5 prikazuje sadržaj funkcije koja zapisuje podatak na blockchain. Najprije se izvršava programski kod 4.4 koji kreira RPC konekciju. Zatim se pomoću te RPC konekcije otključava novčanik na deset sekundi pomoću zaporke spremljene u globalnoj varijabli. Zatim se najprije kreira transakcija unutar koje spremamo hash vrijednost koju funkcija prima kao ulazni parametar. Nakon čega se ta transakcija financira sa žetonima koji se nalaze u novčaniku, u ovome projektu to je iznosilo 0.00000154 SYS žetona što se može vidjeti na slici 4.9. Zatim se digitalno potpisuje transakciju pomoću novčanika i nakon toga se transakcija šalje čvorovima zaduženima za zapis podataka na blockchain.



Slika 4.9: Prikaz kreiranih transakcija

Programski kod 4.6: Funkcija za dohvaćanje podataka sa blockchain-a

```
transaction = rpc_conn.getrawtransaction(txid, 1)
data = str(transaction['vout']).split('OP_RETURN ')[1].split("',")[0]

del rpc_conn
return data
```

Programski kod 4.6 prikazuje sadržaj funkcije koja dohvaća hash vrijednost spremljenu na blockchain. Ta funkcija kao ulazni parametar prima ID transakcije koja se dobije izvršavanjem funkcije iz programskog koda 4.5. Podatak spremljen unutar blockchain transakcije se uvijek nalazi uz OP_RETURN tekst tako da u ovoj funkciji JSON objekt transakcije se pretvori u string vrijednost, a zatim se korištenjem funkcija split dohvati spremljenu hash vrijednost.

4.3.2. Komunikacija s bazom podataka

Komunikacija s bazom podataka odvija se pomoću četiri funkcije i jednom testnom funkcijom za izmjenu podataka kako bi se prikazalo korištenje provjere podataka uz blockchain tehnologiju.

Prva funkcija ima jedan ulazni parametar, a to je string podataka koji se spremaju u bazu od kojih radi hash vrijednost i to vraća. To se može vidjeti u progamskom kodu 4.7.

Programski kod 4.7: Funkcija za kreiranje hash vrijednost podataka

```
hashed_string = hashlib.sha256(string.encode('utf-8')).hexdigest()
return hashed_string
```

Funkcija za spremanje podataka u baze može se vidjeti u programskom kodu 4.8. Na početku funkcije se dohvaćaju ulazni parametri koji se šalju unutar zahtjeva. Nakon čega se poziva funkcija za stvaranje hash vrijednosti iz programskog koda 4.7. Zatim se ta hash vrijednost sprema na blockchain korištenjem funkcije iz programskog koda 4.5 te se odmah dohvaća ID te transakcije. I na kraju se otvara konekcija prema bazi pomoću konekcijskog stringa koji je definiran u jednoj od globalnih varijabli te se izvršava SQL naredba koja sprema sve te podatke u Olives tablicu.

Programski kod 4.8: Funkcija za spremanje podataka u baze

```
# Get data from request
longitude = str(request.json['longitude'])
latitude = str(request.json['latitude'])
grower = str(request.json['grower'])
timestampHarvest = str(request.json['timestampHarvest'])
weight = str(request.json['weight'])
pricePerKg = str(float(request.json['pricePerKg']))

# Create a hash from all the parameters
hash = hash_string(longitude+latitude+grower+timestampHarvest+weight+pricePerKg)

# Write to blockchain and get txid
txid = write_to_blockchain(hash)

# Insert into database
conn = pyodbc.connect(conn_str)
cursor = conn.cursor()
      cursor.execute(f""“““NSERT INTO Olives(
            longitude, latitude,
            grower, timestampHarvest,
            weight, pricePerKg,
            hash, txid)
            VALUES(
            '{longitude}', '{latitude}',
            '{grower}', '{timestampHarvest}',
            '{weight}', '{pricePerKg}',
            '{hash}', '{txid}'
)""“““
```

Nakon funkcije za upis podataka u baze, napravljena je funkcija koja ih dohvaća. No, potrebno je formatirati te podatke kako bi ih se lakše koristilo unutar web aplikacije. Zbog toga se podaci dohvaćeni iz baze promijenjeni u niz JSON objekata. Tijek te funkcije može se vidjeti u programskom kodu 4.9.

Programski kod 4.9: Dohvaćanje podataka iz baze podataka

```
conn = pyodbc.connect(conn_str)
cursor = conn.cursor()
cursor.execute("SELECT * FROM Olives")

# Create a list of Olives dictionaries from the cursor
Olives = collections.namedtuple("Olives",[
      "ID", "longitude", "latitude",
      "grower", "timestampHarvest",
      "weight", "pricePerKg",
      "hash", "txid"
])
result = cursor.fetchall()
data = [Olives(*record)._asdict() for record in result]

cursor.close()
conn.close()

return jsonify({'status': 'success', 'message': 'Olives fetched successfully',
"data": data})
```

Funkcija koja provjerava ispravnost podataka spremljenih u bazi se može vidjeti u programskom kodu 4.10. Na početku funkcije dohvaća se ID transakcije iz zahtjeva. Nakon čega se iz baze podataka dohvaća redak sa podacima čiji je ID transakcije jedan ulaznom parametru. Zatim se ponovno računa hash vrijednost tih podataka dohvaćenih iz baze podataka, također se dohvaća hash vrijednost spremljena na blockchain pod tim ID-em transakcije. Ta dvije hash vrijednosti i hash vrijednost koja je spremljena u bazi moraju sve tri biti jednake kako bi se taj podatak smatrao ispravnim. U suprotnom ako je bilo koja od tih hash vrijednosti drugačija, to znači da je netko neovlašteno mijenjao podatke unutar baze te se taj podatak smatra neispravnim. Nakon provjere hash vrijednosti, funkcija vraća odgovor je li podatak i dalje ispravan ili ne.

Programski kod 4.10: Funkcija za provjeru ispravnosti podataka iz baze podataka

```
# Get data from request
txid = request.json['txid']

# Get Olives from database for the given txid
conn = pyodbc.connect(conn_str)
cursor = conn.cursor()
cursor.execute(f"""
      SELECT * FROM Olives WHERE txid = '{txid}'
""")
row = cursor.fetchone()
cursor.close()
conn.close()

# Get saved, current and blockchain hashes
s_hash = row.hash
c_hash  = hash_string(
      row.longitude + row.latitude +
      row.grower + row.timestampHarvest +
      str(row.weight) + str(row.pricePerKg)
)
b_hash = get_transaction_data(txid)

if s_hash == c_hash and s_hash == b_hash:
      return jsonify({'status': 'success', 'message': 'Olives are valid'})
else:
      return jsonify({'status': 'error', 'message': 'Olive are invalid'})
```

U projektu se također može naći funkcija za brzu promjenu podataka unutar baze kako bi se testirala funkcionalnost iz programskog koda 4.10. Dio te funkcije se nalazi u programskom kodu 4.11.

Programski kod 4.11: Dio funkcije za izmjenu podataka u bazi

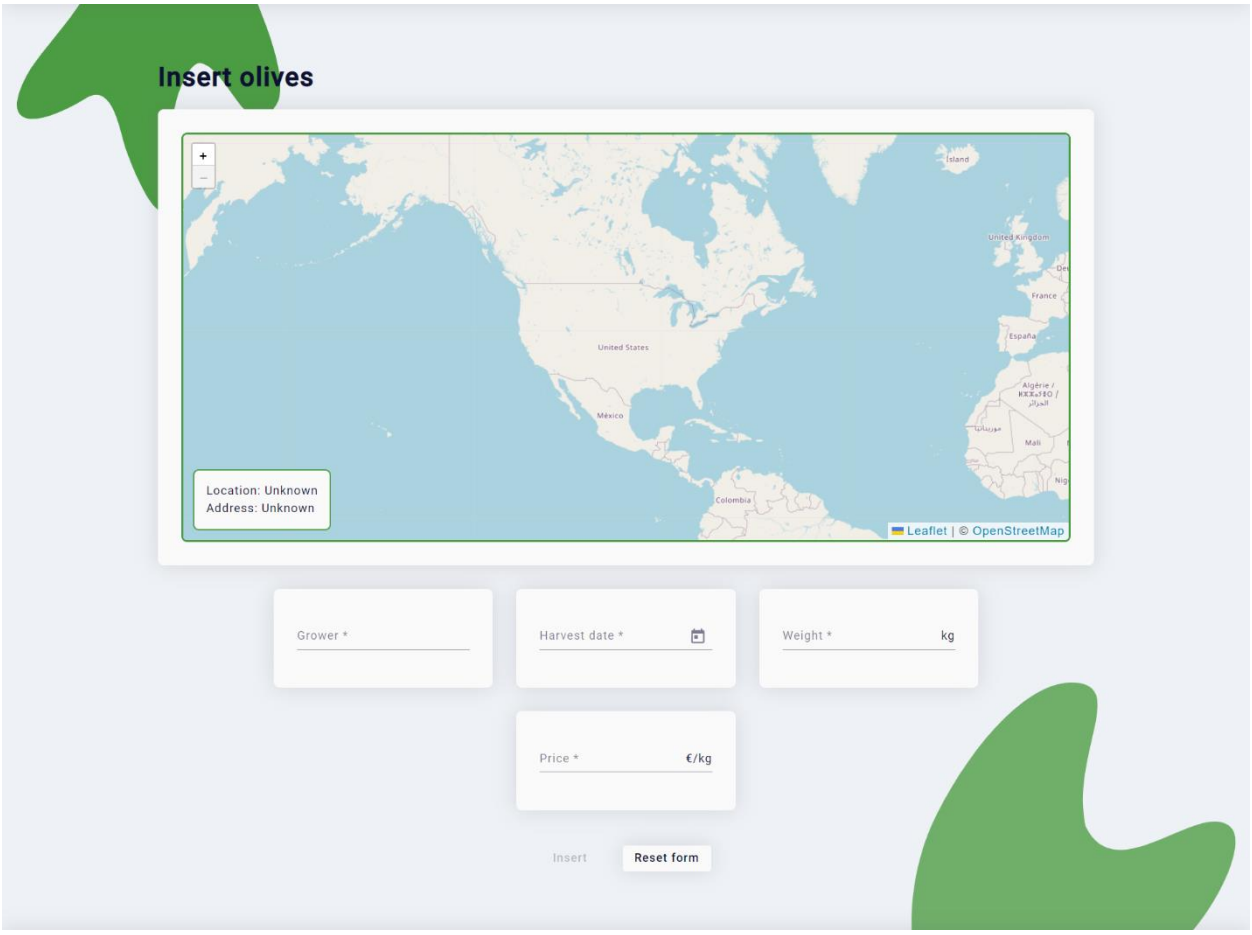
```
cursor.execute(f"""
  UPDATE Olives SET grower = 'Scam Olives d.o.o. {datetime.now}' WHERE ID % 2 = 0
""")
```

4.4. Angular

4.4.1. Grafički prikaz

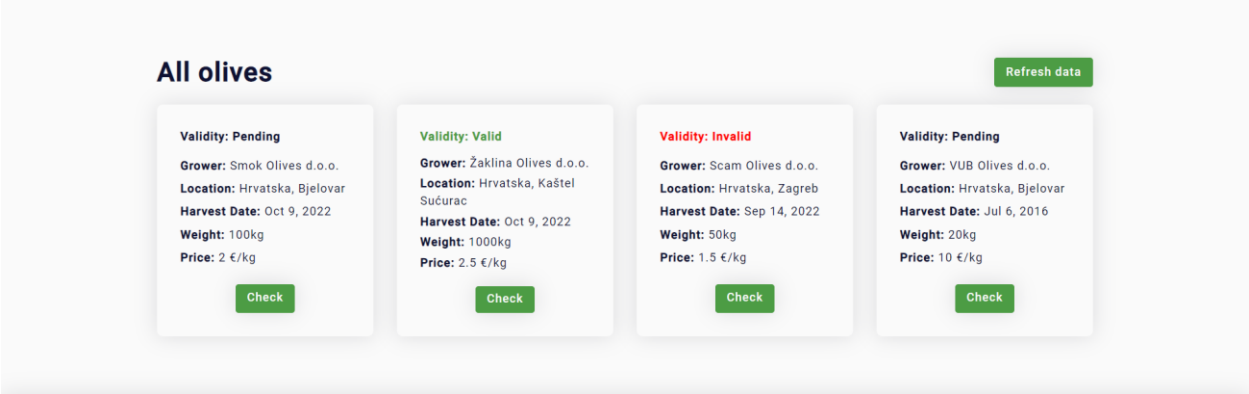
Grafički dio web aplikacije je podijeljen na dvije stranice. Početna stranica koja je se ponaša kao stranica dobrodošlice i ukratko opisuje aplikaciju. No fokus se nalazi na drugoj stranici koja se ponaša kao upravljačka ploča. Na njoj korisnik unosi podatke u formu koji se zatim spremaju u baze, te također može vidjeti sve spremljene podatke i provjeriti njihovu ispravnost.

Druga stranica sastoji se od jedne forme koja se može vidjeti na slici 4.10. Na slici se također može vidjeti kako korisnik ne može podnijeti formu sve dok ne popuni formu.



Slika 4.10: Prikaz grafičkog sučelja forme za unos podataka

Drugi dio upravljačke ploče je odjeljak gdje su prikazani podaci iz baze podataka te postoji opcija da se provjeri ispravnost određenog podatka kojeg korisnik odabere što se može vidjeti na slici 4.11. Ta slika već prikazuje rezultat provjere nekih podataka



Slika 4.11: Prikaz grafičkog sučelja za dohvat i provjeru ispravnosti podataka

4.4.2. Lijeno učitavanje i navigacija putanjama

Lijeno učitavanje (engl. *lazy loading*) je opcija unutar Angular programskog sučelja da se ne preuzima sav kod unutar web preglednika korisnika od jednom. Svrha lijenog učitavanja je da se preuzima samo onaj kod koji će se koristiti. Time se ubrzava inicijalno učitavanje stranice. Lijeno učitavanje se izvršava preko navigacijskih putanja kojima korisnika ide kroz stranicu. Postavljanje lijenog učitavanja može se vidjeti u programskom kodu 4.11.

Programski kod 4.11: Lijeno učitavanje

```
const routes: Routes = [
  {
    path: '',
    data: { pageTitle: 'landing.pageTitle' },
    loadChildren: () => import(
      './landing/landing.module').then((m) => m.LandingModule
    ),
  },
  {
    path: 'dashboard',
    data: { pageTitle: 'dashboard.pageTitle' },
    loadChildren: () => import(
      './dashboard/dashboard.module').then((m) => m.DashboardModule
    ),
  },
  { path: '**', redirectTo: '/', pathMatch: 'full' },
];
```

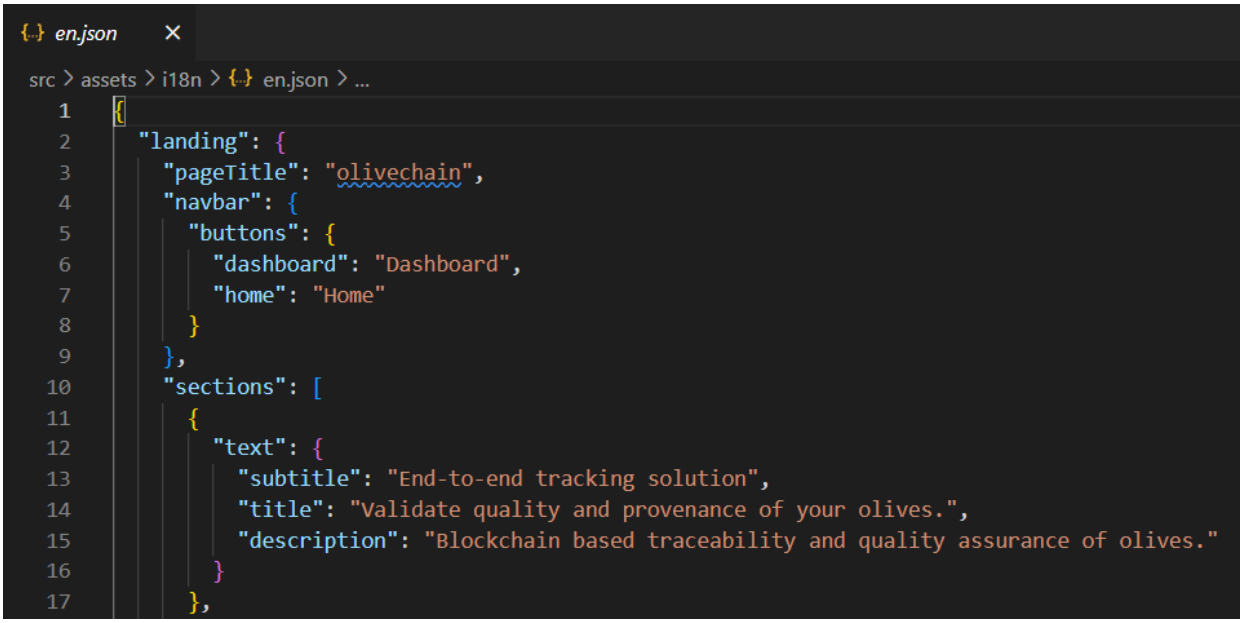
Lijeno učitavanje se odvija pomoću lambda funkcija čiji se rezultat sprema u loadChildren svojstvo odabranog objekta klase Routes. Kako bi navigacija pomoću tih putanja bila potpuno funkcionalna, potrebno je koristiti Angular sistem za promjene putanja. Umjesto korištenja href svojstva unutar HTML-ovog <a> elementa koristi se routerLink svojstvo čiji se primjer može vidjeti u programskom kodu 4.12.

Programski kod 4.12: Prikaz Angular-ovog sistema za promjenu putanje

```
<button mat-button routerLink="/dashboard">
```

4.4.3. Postavljanje internacionaliziranosti

Engleski jezik danas je veoma proširen stoga velik broj web aplikacija ima opciju promjene jezika između lokalnog jezika kao što bi u ovom primjeru bio hrvatski i engleskom jezika zbog njegove široke primjene. Ovaj projekt je također napravljen sa mogućnosti dodavanja drugih jezika unutar aplikacije. To je omogućeno pomoću dva paketa: @ngx-translate/core i @ngx-translate/http-loader. Kako bi se omogućilo korištenje različitih jezika unutar web aplikacije, potrebno je sav tekst aplikacije pisati u odvojene datoteke, a ne unutar HTML-a. Prikaz takve datoteke može se vidjeti na slici 4.12.



Slika 4.12: Prikaz en.json datoteke sa tekstom aplikacije

Kako bi se omogućilo dohvaćanje teksta iz te datoteke, u aplikaciji je napravljena funkcija koja stvara prilagođen TranslateLoader objekt što se može vidjeti u programskom kodu 4.13 te se u programskom kodu 4.14 može vidjet postavljanje @ngx-translate paketa unutar aplikacije kako bi se omogućilo njihovo korištenje kroz cijelu aplikaciju.

Programski kod 4.13: Izrada prilagođen TranslateLoader-a

```
export function createTranslateLoader(http: HttpClient) {
  return new TranslateHttpLoader(
      http, './assets/i18n/', `.json?_=${new Date().toJSON()}`
  );
}
```

Programski kod 4.14: Postavljanje @ngx-translate paketa

```
TranslateModule.forRoot({
   defaultLanguage: 'en',
   loader: {
      provide: TranslateLoader,
      useFactory: createTranslateLoader,
      deps: [HttpClient],
   },
}),
```

Primjer čitanja teksta iz en.json datoteke može se vidjeti u programskom kodu 4.15 čiji se rezultat može vidjeti na slikama 4.13 i 4.14.

Programski kod 4.15: Korištenje translate cijevi

```
<h1 class="mb-4 font-bold">{{ 'landing.sections.0.text.title' | translate }}</h1>
<p class="mb-4">{{ 'landing.sections.0.text.description' | translate }}</p>
```

Slika 4.13: Prikaz rezultata korištenja @ngx-translate paketa

```
<h1 _ngcontent-wgj-c127 class="mb-4 font-bold">Validate quality and
provenance of your olives.</h1>
<p _ngcontent-wgj-c127 class="mb-4">Blockchain based traceability and
quality assurance of olives.</p>
```

Slika 4.14: Prikaz HTML-a unutar web preglednika

4.4.4. Forme i vezanje podataka

Angular pruža niz opcija za povezivanje vrijednosti varijable sa određenim HTML elementom. Kao što je spremanje vrijednosti koja se upisuje unutar <input/> elementa ili pak običan <p> element u kojem se prikazuju podaci. Korištenjem običnog JavaScript koda potrebno je ručno raditi funkciju koja će dohvaćati vrijednost upisanu u <input> element i sprema u željenu varijablu. Korištenjem Angular-ovih to nije potrebno, dovoljno je napraviti varijablu tipa AbstractControl ili FormControl zatim se tu varijablu poveže sa željenim HTML elementom. Izradu instance FormGroup klase koja sadrži niz AbstractControl objekata može se vidjeti u programskom kodu 4.16.

Programski kod 4.16: Kreiranje FormGroup varijable sa nizom AbstracControl objekata

```
this.olivesForm = this.formBuilder.group({
   longitude: [null, Validators.required],
   latitude: [null, Validators.required],
   grower: [null, Validators.required],
   timestampHarvest: [new Date(), Validators.required],
   weight: [null, Validators.required],
   pricePerKg: [null, Validators.required],
});
```

Povezivanje HTML elemenata sa FormGroup i AbstractControl objektima može se vidjeti u programskom kodu 4.17 gdje se povezuje FormGroup objekt sa HTML <form> elementom, a zatim u programskom kodu 4.18 prikazano je povezivanje jednog AbstractControl objekta sa <input> elementom.

Programski kod 4.17: Povezivanje FormGroup objekta sa <form> elementom

```
<form [formGroup]="olivesForm" (submit)="insert()">
```

Programski kod 4.18: Povezivanje AbstractControl objekta sa <input> elementom

```
<input
    matInput formControlName="grower" type="text"
    placeholder="{{
        'dashboard.roles.harvester.shop.form.form-fields.grower.placeholder'
        | translate
    }}"
/>
```

4.4.5. Servisi

Kako bi se omogućilo lagano upravljanje bazama napravljeni su servisi. Servisi su klase koje se najčešće rade korištenjem singleton uzorka. Time se izbjegava nepotrebno kreiranje novih instanci servisa. Jedan od načina naznačivanja da je riječ o singleton klasi u Angular-u može se vidjeti u programskom kodu 4.19. Taj kod se piše neposredno ispred definiranja klase.

Programski kod 4.19: Naznaka single klase u Angular-u

```
@Injectable({
  providedIn: 'root',
})
export class OlivesService {
```

A dohvaćanje servisa se može se izvršiti u samom konstruktoru komponente gdje se servis koristi. Primjer dohvaćanja OlivesService servisa može se vidjeti u programskom kodu 4.20

Programski kod 4.20: Dohvaćanje servisa

```
constructor(
      private formBuilder: FormBuilder,
      private mapService: MapService,
      private olivesService: OlivesService
)
```

Struktura OlivesService servisa može se vidjet u programskom kodu 4.21. Servis se sastoji od jedne varijable koja definira URL putanju aplikacije koja komunicira s bazom podataka i Syscoin Core aplikacijom. Također ima tri funkcije od kojih prva zvana insertOlives ima jedan ulazni parametar koji je djelomičan objekt tipa IOlives. Taj ulazni parametar se sprema u bazu. Izvedbom druge funkcije checkOlivesValidity dobiva se ispravnost maslina iz baze čiji je txid jednak ulaznom parametru funkcije. I zadnja funkcija, getOlives, dohvaća niz objekata tipa IOlives iz baze.

Programski kod 4.21: Struktura OlivesService servisa

```
export class OlivesService {
  private apiUrl = 'http://127.0.0.1:5000/olives';

  constructor(private http: HttpClient) {}

  insertOlives(olives: Partial<IOlives>): Observable<IDBResponse> {
    return this.http.post<IDBResponse>(`${this.apiUrl}/insert`, olives);
  }

  checkOlivesValidity(txid: string): Observable<IDBResponse> {
    return this.http.post<IDBResponse>(`${this.apiUrl}/validity`, { txid });
  }

  getOlives(): Observable<IDBResponse> {
    return this.http.get<IDBResponse>(`${this.apiUrl}`);
  }
}
```

Primjer pozivanja jedne funkcije OlivesService servisa te sortiranje niza podataka kojeg ona vraća može se vidjeti u programskom kodu 4.22.

Programski kod 4.22: Pozivanje getOlives funkcije iz OlivesService servisa

```
getOlives() {
  this.olivesService.getOlives().subscribe((res) => {
    if (res.data) {
      this.olives = res.data.sort(
        (a, b) =>
          new Date(b.timestampHarvest).getTime() -
          new Date(a.timestampHarvest).getTime()
      );
    }
  });
}
```

5. ZAKLJUČAK

Rasprostranjenost i prihvaćenost blockchain tehnologije se znatno povećalo zbog popularnosti kriptovaluta. To proširenje je pokazalo da trenutna tehnologija, kao što su tvrdi diskovi, onemogućuje pun potencijal blockchain tehnologije. No, usprkos tome i dalje ima svoju svrhu. Tehnologija možda još nije spremna za korištenje blockchain-a u svakodnevici, no dosegla je dovoljnu razinu da omogući spremanje podataka na sigurno mjesto sa velikom sigurnošću da se ti podaci neće izmijeniti. To je velik napredak kada je riječ o podacima čiji integritet i ispravnost su od velike važnosti.

Angular je tehnologija koja se redovito proširuje i popravlja novim ažuriranjima. Korisniku omogućuje lakšu izradu web stranica uz niz paketa i nadogradnji koje nudi. Također je veoma koristan pri izradi većih projekata ili projekata u timu sa većim brojem ljudi zbog načina na koji se piše kod. No, treba se imati na umu da makar je Angular vrlo poznata tehnologija, nije i najkorištenija. Manje korisnika koji koriste neku tehnologiju znači manje šanse da se naiđe na rješenje problema na kojeg se naišlo.

Prije izrade web aplikacije bitno je istražiti koje su tehnologije dostupne kako bi se među njima odabrale one koje najbolje odgovaraju za izradu te aplikacije. Veličina projekta i tima koji radi na aplikaciji su također jedan od faktora koji utječu na to. Potrebno je i paziti koliko veliku sigurnost aplikacija zahtjeva pogotovo ako je riječ o aplikaciji koja radi sa osjetljivim i osobnim podacima korisnika aplikacije. Većina tehnologija za izradu aplikacija se redovito mijenja kroz ažuriranja, stoga je vrlo važno pratiti promjene u tehnologijama koje se koriste. Tako se može pružiti što sigurnije i bolje rješenje.

6. LITERATURA

[1] Agnieszka Kozubek-Krycuń. What Is a SQL Dialect, and Which one Should You Learn? [Online]. 2020. Dostupno na: https://learnsql.com/blog/what-sql-dialect-to-learn/#:~:text=SQL%20Is%20the%20Language%20for%20Talking%20to%20Databases&text=PostgreSQL%2C%20MySQL%2C%20Oracle%2C%20and,call%20these%20variants%20SQL%20dialects. (12.10.2022.)

[2] Microsoft. SQL Server technical documentation [Online]. 2022. Dostupno na: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16. (12.10.2022.)

[3] Pallets. Flask [Online]. 2022. Dostupno na: https://flask.palletsprojects.com/en/2.2.x/. (20.9.2022.)

[4] Google. Angular [Online]. 2022. Dostupno na: https://angular.io/docs. (10.7.2022.)

[5] Syscoin. Syscoin [Online]. 2022. Dostupno na: https://docs.syscoin.org/docs/intro/syscoin-what/. (20.9.2022.)

[6] Crypto.com. Crypto.com [Online]. 2022. Dostupno na: https://crypto.com/price/syscoin (9.10.2022.)

7. OZNAKE I KRATICE

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

RPC – Remote Procedure Call

SCSS – Sassy Cascading Style Sheets

SQL – Structured Query Language

SYS – Syscoin žeton

URL – Uniform Resource Locator

8. SAŽETAK

Naslov: Web stranica za unos i provjeru porijekla uroda masline na blockchain

U ovom radu opisane su tehnologije koje se koriste. A to su HTML, SCSS i TypeScript korištenjem Angular aplikacijskog okruženja, Python i Flask aplikacijskog okruženja i SQL programskog jezika. Opisana je i blockchain tehnologija te mogućnosti koje ona pruža. Također su prikazani primjeri korištenja navedenih tehnologija počevši od izrade centralizirane baze podataka nakon čega se pokazuje korištenje blockchain tehnologije pomoću Syscoin Core aplikacije. Blockchain je omogućio provjeru integriteta spremljenih podataka. Nakon toga se pokazuje kod koji omogućuje lakšu komunikaciju s bazom podataka i blockchain-om. Na kraju je prikazano korištenje Angular-a, to jest primjeri kako koristiti neke od tehnologija koje pruža kao što su izrada singleton servisa, promjena web putanja aplikacije, povezivanje HTML elemenata sa JavaScript varijablama i tako dalje.

Ključne riječi: blockchain, Syscoin, baza podataka, Angular, Python, Flask.

9. ABSTRACT

Title: A website for entering and verifying the provenance of the olive crop on the blockchain

In this final thesis are described used technologies. Those are HTML, SCSS and TypeScript used by Angular framework, Python and Flask framework and SQL programming language. Blockchain technology and the possibilites it provides are also described. Examples of using mentioned technologies is shown aswell starting with the making centralized database after which the use of blockchain technology using the Syscoin Core application is demonstrated. Blockchain enabled verifying the integrity of the saved data. Afer that the code which enables easier communication with database and blockchain is shown. At the end the usage of Angular is shown, that is the examples of how to use some of the technologies which it provides such as making of singleton service, changing web route of the application, binding HTML element with JavaScript variables and so on.

Keywords: blockchain, Syscoin, baza podataka, Angular, Python, Flask.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 20.10.2022.	Ivan Smok	Ivan Smok

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

Ivan Smok

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 20.10.2022.

Ivan Smok

potpis studenta/ice