SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 26

Implementacija poslužiteljskog sustava za ispitivanje ispravnosti fiskalnih blagajni

Tomislav Prhat

Zagreb, svibanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 26

Implementacija poslužiteljskog sustava za ispitivanje ispravnosti fiskalnih blagajni

Tomislav Prhat

Zagreb, svibanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 26

Pristupnik:	**Tomislav Prhat (0036509157)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Stjepan Groš

Zadatak: **Implementacija poslužiteljskog sustava za ispitivanje ispravnosti fiskalnih blagajni**

Opis zadatka:

Kako bi se spriječilo crno tržište u RH uvedene su fiskalne blagajne čija zadaća je da svaki izdani račun prijave Poreznoj upravi. Međutim, fiskalne blagajne rade razni proizvođači programske podrške te se postavlja pitanje koliko su implementacije ispravne, odnosno, bez pogrešaka u kodu. Dodatno, zbog toga što je u proces prijave uključena kriptografija to znači da je vjerojatnost pogreške još veća budući da upotreba kriptografije, a posebno certifikata i digitalnog potpisa, nije toliko raširena. Zbog svega toga oportuno je razviti sustav koji bi omogućio ispitivanje ispravnosti implementacije fiskalnih blagajni. U sklopu ovog završnog rada potrebno je razviti poslužiteljski sustav koji prihvaća upite od fiskalnih blagajni te generira odgovore koji se potom vraćaju klijentu. Sustav mora omogućiti korištenje dodataka (engl. plugins) koji omogućavaju provjeru ispravnosti zahtjeva fiskalnih blagajni te manipulaciju odgovora radi ispitivanja korektnosti implementacije blagajni. Obratiti pozornost na vođenje evidencije o provedenim testovima te testovima koje je tek potrebno provesti. Potrebno je također implementirati klijentsku aplikaciju koja imitira pojednostavljenu fiskalnu blagajnu. Citirati korištenu literaturu i navesti dobivenu pomoć.

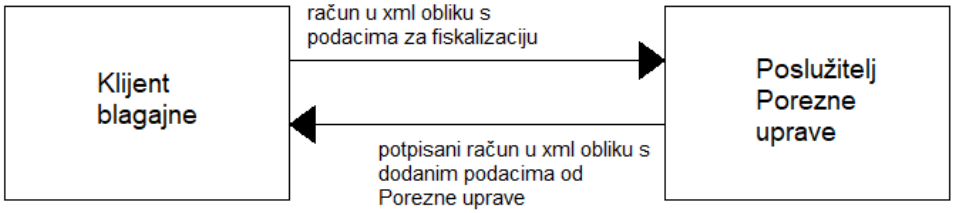
Rok za predaju rada: 11. lipnja 2021.

Sadržaj

1. Uvod 1
2. Sustav za ispitivanje fiskalnih blagajni 3
2.1. Poslužitelj za ispitivanje fiskalnih blagajni 4
2.2. Klijent za ispitivanje fiskalnih blagajni 4
2.3. Sustav dodataka 6
2.4. Web aplikacija 6
2.5. Baza podataka 7
3. Tehnologije za implementaciju sustava za ispitivanje fiskalnih blagajni 9
3.1. Kriptografija 9
3.2. Digitalni potpis 10
3.2.1. RSA algoritam 11
3.3. Digitalni certifikat 11
3.4. HTTP protokol 13
3.5. XML 14
4. Implementacija sustava za ispitivanje fiskalnih blagajni 15
4.1. Vanjske knjižnice 15
4.2. Odabir radnog okvira za implementaciju sustava 15
4.3. Implementacija poslužitelja za ispitivanje fiskalnih blagajni 16
4.4. Implementacija klijenta za ispitivanje fiskalnih blagajni 17
4.5. Implementacija baze podataka 18
5. Zaključak 20
6. Literatura 21

1. Uvod

Ovaj rad ima za cilj objasniti što je i na koji način je implementiran poslužiteljski sustav za ispitivanje ispravnosti fiskalnih blagajni. Fiskalne blagajne su programska rješenja koja omogućavaju fiskalizaciju preko interneta putem HTTP/HTTPS protokola. Fiskalizacija je skup mjera što ih od 1. siječnja 2013. godine provode obveznici fiskalizacije među koje spadaju kafići, restorani i slični objekti, a njezin je glavni cilj nadzor prometa u gotovini. Svaki račun se prije izdavanja korisniku šalje Poreznoj upravi pomoću internetske veze te time Porezna uprava ima uvid u svaki izdani račun. Time se suzbija utaja poreza i doprinosi porastu razine svijesti o potrebi plaćanja poreza, što ima za posljedicu uravnoteženo financiranje javnih troškova [1]. Na slici 1.1 se može vidjeti apstraktni prikaz arhitekture sustava. Klijent blagajne šalje zahtjeve poslužitelju Porezne uprave koji se zatim provjerava i šalje natrag klijentu fiskalne blagajne.



Slika 1.1: Arhitektura sustava klijent - poslužitelj

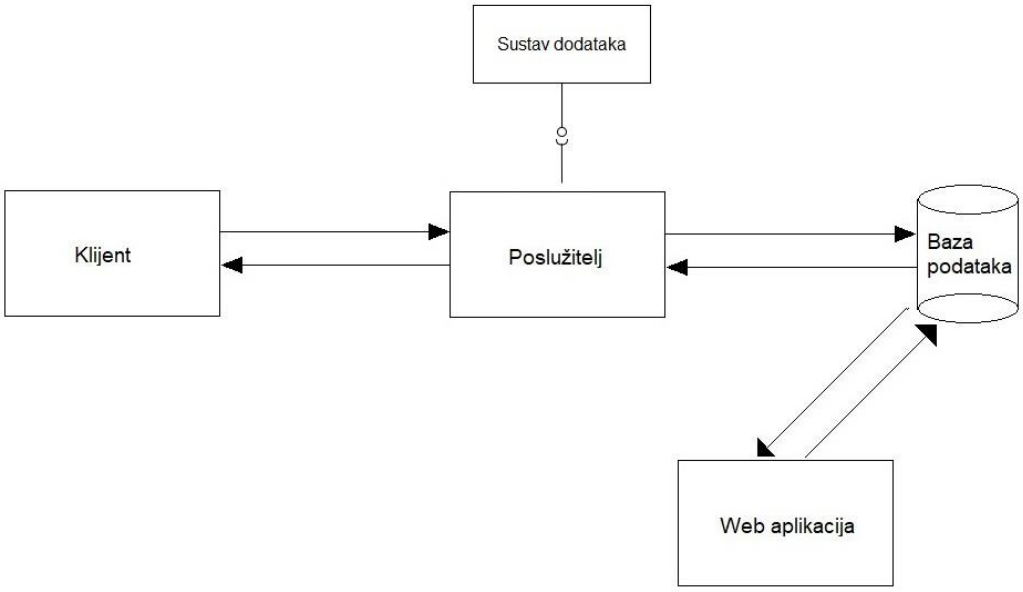
Najveći problem ovakvog sustava je što blagajne proizvode razni proizvođači programske potpore pa se postavlja pitanje koliko su implementacije točne, tj. bez pogrešaka. Ujedno u radu se koriste kriptografija, digitalni potpisi i digitalni certifikati čija uporaba nije toliko raširena. Upravo zbog ovih razloga potrebno je razviti sustav koji bi omogućio ispitivanje ispravnosti implementacije fiskalnih blagajni.

Rad je strukturiran na sljedeći način. U prvom poglavlju se opisuje sustav za ispitivanje fiskalnih blagajni, počevši od opisa poslužitelja za ispitivanje fiskalnih blagajni, zatim slijedi opis klijenta za ispitivanje. Nakon toga se opisuje sustav dodataka (*engl. plugin*) te na koji način s njim komunicira poslužitelj. Predzadnje potpoglavlje opisuje rad i način na koji je web aplikacija povezana s ostatkom sustava za ispitivanje. U zadnjem potpoglavlju se opisuje baza podataka, njene mogućnosti i zašto je potrebna u ovom sustavu. U drugom poglavlju se obrađuju definicije i način na koji funkcioniraju kriptografija, digitalni potpis i digitalni certifikat. U istome poglavlju se također opisuje protokol za prijenos podataka i informacija između klijent i poslužitelja te svi

njegovi elementi. Posljednji dio trećeg poglavlja opisuje tip podataka koji se koriste u fiskalizaciji, prijenosu podataka putem Interneta i također se opisuje vanjska knjižnica koja omogućuje pregled, provjeru i manipulaciju tog tipa podataka. U četvrtom poglavlju se opisuje implementacija sustava za ispitivanje fiskalnih blagajni. Poglavlje započinje uvodom u vanjske knjižnice (*engl. library*) te kako se one koriste u programskom rješenju. Nakon toga dolazi kako je izabran radni okvir za implementaciju sustava. Zatim slijedi implementacija poslužitelja za ispitivanje fiskalnih blagajni i detalji njegovog rada. Nakon toga dolazimo do implementacije klijenta za ispitivanje fiskalnih blagajni i načina na koji njegov rad utječe na sustav. U zadnjem potpoglavlju četvrtog poglavlja se opisuje implementacija baze podataka zajedno s njenim funkcionalnostima potrebnim za implementaciju sustava za testiranje fiskalnih blagajni. U petom poglavlju se nalazi zaključak za ovaj rad, vlastiti osvrt na cijeli sustav i planovi za budući rad. U šestom i posljednjem poglavlju se nalazi literatura koja je pomogla sastaviti ovaj sustav, kao i tekst ovog rada.

2. Sustav za ispitivanje fiskalnih blagajni

Sustav je zamišljen na način da korisnici svoje fiskalne blagajne mogu povezati direktno na poslužitelj i testirati implementaciju programskog rješenja njihove blagajne dok se preko aplikacije mogu registrirati korisnici, unositi testove koje žele pokrenuti i vidjeti uspješnosti testova koje su pokrenuli. Kao što je vidljivo na slici 2.1 klijent komunicira direktno s poslužiteljem na kojem se izvršavaju potrebni testovi koje dohvaća iz baze podataka. Poslužitelj također komunicira s sustavom dodataka (*engl. plugin*) u kojem su sadržane potrebne funkcionalnosti za provođenje testova. U bazi podataka su sačuvani svi relevantni podaci potrebni za provođenje i spremanje odrađenih testova. Web aplikacija je odvojena od samog sustava i ona komunicira jedino s bazom podataka kako bi mogla prikazati provedene testove te kako bi mogla zapisati testove koje je potrebno pokrenuti u bazu podataka.



Slika 2.1: Apstraktni prikaz sustava za testiranje fiskalnih blagajni

U sljedećih nekoliko poglavlja se opisuju svi dijelovi sustava te način na koji su oni povezani međusobno. U poglavlju 2.1 se opisuje poslužitelj za ispitivanje kao i način na koji obrađuje zahtjeve, u poglavlju 2.2 se opisuje klijent za ispitivanje i kako moraju zahtjevi izgledati, nadalje u poglavlju 2.3 se opisuje sustav dodataka koji služi za izmjenu, provjeru i potpisivanje zahtjeva. Opis web aplikacije koja je ulazna točna za upisivanje testova i provjeru prolaznosti testova je poglavlje 2.4. Konačno opis baze podataka u poglavlju 2.5 govori o vrsti baze podataka, kakvi sve podaci postoje i na koji način se ona uklapa u sustav.

2.1. Poslužitelj za ispitivanje fiskalnih blagajni

Poslužitelj je računalo ili mreža računa koje poslužuje informacije i podatke ostalim računalima. Ta računala se nazivaju klijenti i oni se povezuju na poslužitelj ili putem lokalne mreže ili pomoću širokopojasnih mreža poput Interneta. Poslužitelji mogu biti i obična računala koja dijele neke podatke s ostalim računalima.

Poslužiteljev zadatak je da prikuplja, šalje i po potrebi obrađuje podatke i informacije koje dolaze do njega. Od poslužitelja se u većini slučajeva očekuje da klijentu, koji mu je poslao neki zahtjev, pošalje nekakvu potvrdu o uspješnosti tog zahtjeva.

Poslužitelj implementiran u ovom sustavu koristi eXtensible Markup Language (XML) kao tip podataka koji se šalju mrežom pošto je to tip podataka kojim poslužitelj Porezne uprave manipulira i Hypertext Transfer Protocol (HTTP) kao protokol za slanje i primanje podataka jer je to standard za slanje podataka preko Interneta te ga isto koristi poslužitelj Porezne uprave.

Poslužiteljev glavni zadatak je primanje računa u XML obliku od klijenta, račun zatim pretraži kako bi našao identifikator poruke (ZKI), ZKI je jedinstveni broj duljine 32 znaka koji generira svaka fiskalna blagajna prije slanja računa prema poslužitelju Porezne uprave. Sljedeći korak poslužitelja je provjera koje dodatke je korisnik zatražio za provjeru s obzirom na jedinstveni identifikator fiskalne blagajne. Nakon toga se XML zajedno s traženim testovima šalje u sustav dodataka. Sustav dodataka odrađuje potrebne provjere i vraća poslužitelju odgovor s izmijenjenim dijelovima XML s obzirom na zatražene testove. Poslužitelj zatim pregledava XML i traži element „greške“, sve atribute iz tog elementa sprema na bazu podataka ukoliko on postoji, u suprotnome na bazu podataka ne sprema greške nego samo zapisuje „OK“ što označuje da su svi testovi prošli bez problema. U ovom koraku još obriše izvršene testove s baze podataka. Zadnji korak je slanje odgovora klijentu te u njemu vrati izmijenjeni XML. U slučaju bilo kakve greške pri obradi računa poslužitelj će klijentu poslati grešku i zatražiti ponovno slanje zahtjeva.

2.2. Klijent za ispitivanje fiskalnih blagajni

Klijent je bilo koji uređaj koji pristupa nekom poslužitelju, može biti stolno računalo, prijenosno računalo, mobilni uređaj i slično. Uređaji postaju klijent u trenutku kad zatraže resurs s nekog poslužitelja.

Klijent implementiran u ovom sustavu također koristi XML i HTTP kao i poslužitelj da bi se mogli povezati međusobno i imati dobru integraciju. Klijent u ovom slučaju glumi klijent fiskalne blagajne koji šalje zahtjeve za fiskalizaciju poslužitelju za ispitivanje fiskalnih blagajni. Klijent u sebi sadrži jedinstveni identifikator blagajne i račun koji želi fiskalizirati.

Glavni zadatak klijenta je slanje podataka koji uključuju identifikator blagajne i račun za fiskalizaciju. Nakon toga poslužitelj odrađuje svoj posao te vraća odgovor klijentu koji on sprema u svoju bazu podataka te taj račun može izdati svom korisniku ukoliko nije došlo do greške. Na slikama 2.2 i 2.3 su vidljivi primjeri računa koji se šalje Poreznoj upravi i koji se prima od Porezne uprave.

```
<tns:RacunZahtjev>
 <tns:Zaglavlje>
 <tns:IdPoruke>1011cff8-d8c6-4b7c-a3b0-c13c21d2311a</tns:IdPoruke>
 <tns:DatumVrijeme>24.01.2020T10:40:10</tns:DatumVrijeme>
 </tns:Zaglavlje>
 <tns:Racun>
 <tns:Oib>98765432198</tns:Oib>
 <tns:USustPdv>true</tns:USustPdv>
 <tns:DatVrijeme>24.01.2020T10:40:02</tns:DatVrijeme>
 <tns:OznSlijed>P</tns:OznSlijed>
 <tns:BrRac>
 <tns:BrOznRac>7</tns:BrOznRac>
 <tns:OznPosPr>P11</tns:OznPosPr>
 <tns:OznNapUr>123456789</tns:OznNapUr>
 </tns:BrRac>
 <tns:IznosUkupno>22.00</tns:IznosUkupno>
 <tns:NacinPlac>G</tns:NacinPlac>
 <tns:OibOper>01234567890</tns:OibOper>
 <tns:ZastKod>1fb0c204a133d041cf0bcdf9846c391b</tns:ZastKod>
 <tns:NakDost>false</tns:NakDost>
 </tns:Racun>
</tns:RacunZahtjev>
```

Slika 2.2: Račun koji se šalje Poreznoj upravi na fiskalizaciju

```
<tns:RacunOdgovor xmlns:tns="http://www.apis-it.hr/fin/2012/types/f73"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <tns:Zaglavlje>
 <tns:IdPoruke>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</tns:IdPoruke>
 <tns:DatumVrijeme>01.09.2012T21:10:34</tns:DatumVrijeme>
 </tns:Zaglavlje>
 <tns:Jir>2cf55235-9470-4b5c-a539-463f52b109d2</tns:Jir>
</tns:RacunOdgovor>
```

Slika 2.3: Odgovor za račun koji Porezna uprava šalje klijentu fiskalne blagajne

2.3. Sustav dodataka

Dodaci (*engl. plugins*) su dio programa koji dodaje nove funkcionalnosti programu bez da izmjenjuje sam početni program, tj. program poziva dodatke kako bi iskoristio njihove funkcionalnosti. Dodaci nude veliku fleksibilnost programu. Ukoliko programska potpora nema arhitekturu zasnovanu na dodacima može doći do potrebe da se izmjenjuje cijela programska podrška kako bi se implementirala nova funkcionalnost.

Dodaci u ovom sustavu služe kao pomoć poslužitelju da odradi sve zahtijevane testove koje je korisnik zatražio bez mijenjanja cijelog koda kada dođe do potrebe za novim testom koji još ne postoji u bazi testova. Dodaci u ovom sustavu se mogu podijeliti u 3 kategorije:

1. Modificirajući dodaci – služe za izmjenu dijelova XML-a koji su zatraženi u testu, mogu dodavati, mijenjati i brisati dijelove.
2. Dodaci za provjeru – služe za provjeru strukture, tipova podataka, jesu li prisutni svi potrebni elementi da bi XML bio valjan i slično. Ukoliko dođe do neke pogreške u XML pod element „greške“ će zapisati grešku i nastaviti dalje s provjerom ukoliko nije došlo do nepopravljive pogreške. Postoji 6 pogrešaka do kojih može doći i one su:
   - s001: "Poruka nije u skladu s XML shemom : #element ili lista elemenata koji nisu ispravni po shemi#"
   - s002: "Certifikat nije izdan od strane FINA RDC CA ili je istekao ili je ukinut."
   - s003: "Certifikat ne sadrži naziv 'Fiskal' ."
   - s004: "Neispravan digitalni potpis."
   - s005: "OIB iz poruke zahtjeva nije jednak OIB-u iz certifikata"
   - s006: "Sistemska pogreška prilikom obrade zahtjeva."
3. Potpisujući dodaci – služe za potpisivanje XML dokumenta kao što bi to bilo od strane poslužitelja Porezne uprave s obzirom na umetnuti ključ i certifikat

2.4. Web aplikacija

Web aplikacija je programsko rješenje koje je pokrenuto na nekom web poslužitelju, tj. nije potrebno imati ju instalirano lokalno na računalo da bi radila. Dovoljno je pristupiti web adresi na kojoj se nalazi i imati sve funkcionalnosti koje ona nudi. Ukoliko aplikacija ima više vrsta ovlasti, različiti korisnici mogu imati manje ili više funkcionalnosti s obzirom na njihova prava.

Web aplikacija u ovom sustavu služi za registraciju korisnika, nakon registracije korisnik može dodavati fiskalne blagajne, generirati certifikate, ključeve i sve potrebne stvari za rad fiskalne blagajne. Također korisnik putem web aplikacije unosi testove koje želi pokrenuti na svojoj blagajni koji se potom spremaju na bazu podataka. Nakon pokretanja testova pomoću fiskalne blagajne korisnik na svojoj nadzornoj ploči unutar web aplikacije može vidjet testove koji su završeni te detalje o samom testu, kada su izvršeni testovi, koji testovi su pokrenuti te uspješnost provedenih testova. U suštini, web aplikaciju ovog sustava je glavni dio za praćenje testiranja.

2.5. Baza podataka

Baza podataka je kolekcija informacija ili podataka organizirana tako da je lako pretraživati, dodavati i osvježavati podatke. Postoji više vrsta baza podataka od kojih su najpopularnije relacijske baze podataka i semantičke baze podataka. Relacijske baze podataka se temelje na tablicama s stupcima i redcima koji su indeksirani za lakše pretraživanje dok se semantičke baze podatak temelje na grafovima koji sadrže čvorove i listove.

Baza podataka u ovom sustavu pripada relacijskim bazama podataka. U nju se spremaju podaci o korisnicima, podaci o svim dodacima (*engl. plugins*) koji postoje u sustavu, dio te tablice je vidljiv u tablici 2.1. U tablici 2.2 su podaci o testovima koje je potrebno pokrenuti, dok su u tablici 2.3 podaci o završenim testovima. Baza podataka povezuje sve dijelove sustava na način da svi uzimaju iste podatke s baze i osvježavaju podatke po potrebi. Web aplikacija dodaje testove, a poslužitelj za testiranje pokreće one testove koji su zapisani u bazi podataka. Nakon završetka testa poslužitelj zapisuje na bazu završene testove.

ID	Name	Type	Description
1	ModifyPlugin	modifyPlugins	Changes the date of the given file to the current date and time
2	SignPlugin	signPlugin	Signs the given file using the newcert and newkey certificate and key respectively
3	CheckPlugin	checkPlugins	Verifies the given file using XMLVeirifier from the sign xml library

Tablica 2.1: Dio podataka iz tablice *tests*

ID	Tests	cashRegister_ID
1	ModifyPlugin ModifyPlugin2 AlwaysInvalidPlugin AlwaysValidPlugin CheckPlugin SignPlugin	3
2	AlwaysInvalidPlugin AlwaysValidPlugin CheckPlugin	5
3	ModifyPlugin AlwaysValidPlugin CheckPlugin	6

Tablica 2.2: Podaci iz tablice tests_to_run

ID	ZKI	JIR	Plugins_used	Conclusion	cashRegister_ID
1	f81d4fae-7dec-11d0-a765-00a0c91e6bf6	8e4ef809-a01c-4b71-9054-8e4da522db7e	ModifyPlugin CheckPlugin	s004: Neispravan digitalni potpis. s002: Certifikat nije izdan od strane FINA RDC CA ili je istekao ili je ukinut.	3
2	f81d4fae-7dec-11d0-a765-00a0c91e6bf6	63019fdb-35c6-482d-a34f-8575cb838c01	AlwaysInvalidPlugin CheckPlugin SignPlugin	s005: OIB iz poruke zahtjeva nije jednak OIB-u iz certifikata s003: Certifikat ne sadrži naziv Fiskal .	5
3	f81d4fae-7dec-11d0-a765-00a0c91e6bf6	ffa7fc5f-1730-4ec3-861e-0a98b5ddd034	ModifyPlugin AlwaysValidPlugin CheckPlugin	s006: Sistemska pogreška prilikom obrade zahtjeva.	6

Tablica 2.3: Podaci o završenim testovima iz tablice finished_tests

3. Tehnologije za implementaciju sustava za ispitivanje fiskalnih blagajni

U sljedećih nekoliko poglavlja će biti riječi tehnologijama koje omogućuju slanje i primanje računa između fiskalnih blagajni i poslužitelja Porezne uprave na siguran i učinkovit način. U poglavlju 3.1. se opisuje što je kriptografija i kako se ona koristi, zatim u poglavlju 3.2. slijedi digitalni potpisi kao postupak potpisivanja podataka ili informacija od strane pošiljatelja kako bi se utvrdio njegov identitet. Poglavlje 3.3. objašnjava čemu služe digitalni certifikati te koja je njihova svrha. Ova 3 poglavlja opisuju sigurnosni dio prijenosa podataka, dok se u poglavljima nakon, poglavlje 3.4. i 3.5., objašnjava protokol kojim se podaci prenose s klijenta fiskalne blagajne na poslužitelj Porezne uprave i obratno te vrsta podataka koji se tim protokolom šalju.

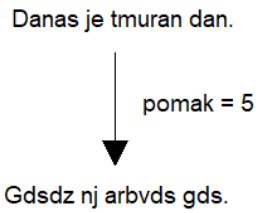
3.1. Kriptografija

Kriptografija je proučavanje tehnika sigurne komunikacije koje omogućuju samo pošiljatelju i primatelju da mogu pročitati i razumjeti sadržaj poruke. Također omogućuje potpisivanje poruka i dokumenata kako bi se mogao dokazati identitet pošiljatelja. Usko je povezana sa enkripcijom, a to je postupak pretvaranja običnog teksta u šifrirani tekst te prilikom dolaska do primatelja ponovno pretvaranje u obični tekst koji može pročitati.

Postoje 4 osnovna sigurnosna zahtjeva svake komunikacije preko Interneta:

1. Autentikacija – proces u kojem se provjerava identitet korisnika, odnosno proces kojim korisnik dokazuje da je zaista onaj za kojeg se predstavlja.
2. Integritet – predstavlja sigurnost da poruka ili podaci u prijenosu nisu mijenjani ili uništeni od strane treće strane koja ne bi smjela vidjeti podatke ili poruku. Digitalnim potpisom osigurava se cjelovitost i izvornost podataka koji se šalju mrežom. Jednostavnom provjerom može se utvrditi jesu li onda ti podaci nelegalno naknadno mijenjani.
3. Tajnost – podaci koji se šalju moraju biti dostupni jedino pošiljatelju i njegovom sugovorniku.
4. Neporecivost – onemogućavanje poricanja akcije koju je osoba poduzela, pošiljatelj ne može reći da on nije poslao poruku.

Jedan od najjednostavnijih kriptografskih algoritam je Cezarova šifra koja uzme proizvoljan tekst te zatim svako slovo pomiče za fiksan broj mjesta unaprijed ili unazad. Npr. ako imamo slovo 'G' i naš fiksni pomak je 5 tada je 'G' + 5 = 'L'. I takav proces će se odviti za sva slova kao što je prikazano na slici 3.1. Ovim postupkom ćemo dobiti šifriranu riječ koja je ne razumljiva čovjeku na prvi pogled, ali se jednostavnom analizom može razbiti ova šifra. Postupak razbijanja ovakvih šifri se naziva frekvencijska analiza [2].

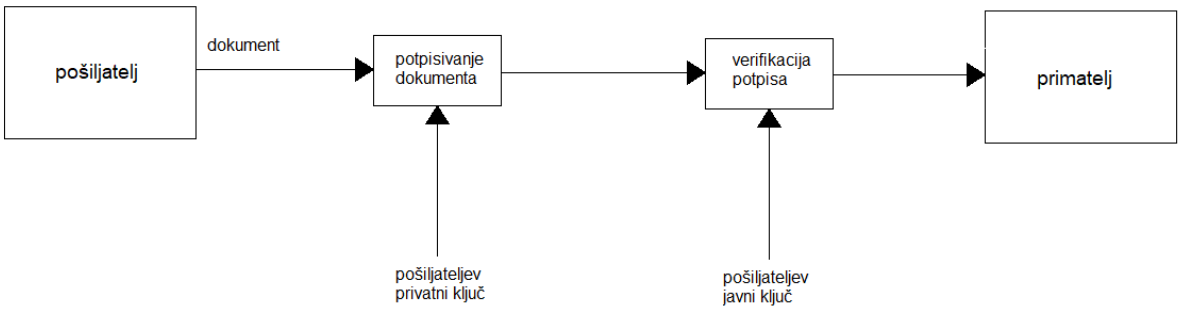


Slika 3.1: Cezarova šifra

3.2. Digitalni potpis

Digitalni potpis je matematička tehnika korištena za validaciju autentičnosti i integriteta poruke ili digitalnog dokumenta. To je digitalni ekvivalent vlastoručnog potpisa na papiru kojim se potvrđuje identitet pošiljatelja, s time da digitalni potpis nudi puno veću sigurnosti. Cilj digitalnih potpisa je da riješe problem petljanja i lažnog predstavljanja u komunikaciji. Digitalni potpisi nude dokaz o porijeklu i identitetu poruke ili dokumenta. U mnogim državama su digitalni potpisi pravno obvezujući isto kao i tradicionalni vlastoručni potpisi [3].

Digitalni potpisi su bazirani na kriptografiji javnog ključa, tzv. asimetričnom kriptografijom. Korištenjem algoritma javnog ključa kao što je RSA, generiraju se dva ključa, jedan privatni i jedan javni. Privatni ključ se nikad ne smije dijeliti s drugim jer on predstavlja potpis te ukoliko postane dostupan još nekome osim pošiljatelju dolazi do gubitka autentičnosti i integriteta, a javni ključ se dijeli s sugovornicima koji će s tim ključem dekriptirati podatke vezane s potpisom. Ukoliko primatelj ne može dekriptirati poruku javnim ključem znači da je došlo do pogreške ili je pogrešan ključ. Digitalno potpisivanje dokumenta je prikazano na slici 3.2 gdje pošiljatelj potpisuje svoj dokument svojim privatnim ključem i šalje ga primatelju. Zatim primatelj provjerava potpis pomoću pošiljateljevim javnim ključem. Korištenje digitalnih potpisa zahtijeva povjerenje svih strana da je osoba koja je kreirala digitalni potpis održala privatni ključ privatnim.



Slika 3.2: Digitalno potpisivanje dokumenta pomoću digitalnog popisa

3.2.1. RSA algoritam

RSA (Rivest–Shamir–Adleman) algoritam je asimetrični kriptografski algoritam koji se najviše koristi u sigurnom prijenosu podataka. Asimetrični kriptografski sustav znači da ima dva ključa, jedan javni i jedan privatni. Također je jedan od najstarijih, javno objavljen 1978. godine. Nazvan je po njegovim izumiteljima: Ron Rivest, Adi Shamir, and Leonard Adleman. RSA se temelji na činjenici da je teško faktorizirati velike brojeve. Javni ključ se sastoji od 2 broja gdje je jedan umnožak dvaju primarnih brojeva. Privatni ključ je također deriviran od ta ista 2 broja. Snaga enkripcije u potpunosti ovisi o veličini ključa te ukoliko se on poveća 2 ili 3 puta snaga enkripcije se poveća eksponencijalno. Ključevi su tipično dugi između 1024 i 2048 bitova.

3.3. Digitalni certifikat

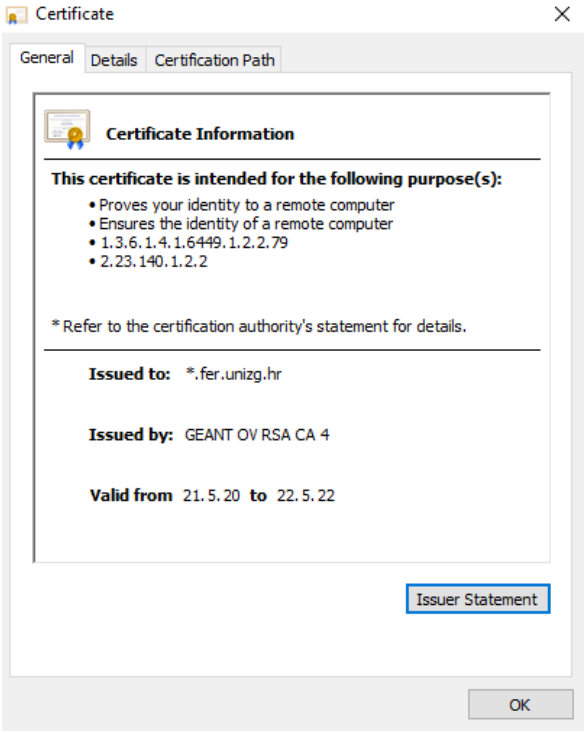
Digitalni certifikat je potvrda u elektroničnom obliku koja predstavlja digitalni identitet u digitalnim transakcijama te omogućava sigurnu i povjerljivu komunikaciju Internetom. Pošiljatelj dokazuje svojim poslovnim partnerima, prijateljima i elektroničnim servisima da je poslana informacija autentična.

Certifikat predstavlja elektroničku identifikacijsku iskaznicu koja sadrži ključ, informacije o vlasniku, vijeku trajanja certifikata, izdavatelju, te ovjeru od strane izdavatelja da je taj certifikat valjan. Također, povezuje vlasnika certifikata s njegovim javnim ključem. Primjer certifikata je prikazan na slici 3.3. na kojoj se vidi kome je izdan certifikat, od koga je izdan, njegov vijek trajanja te što sve on dokazuje.

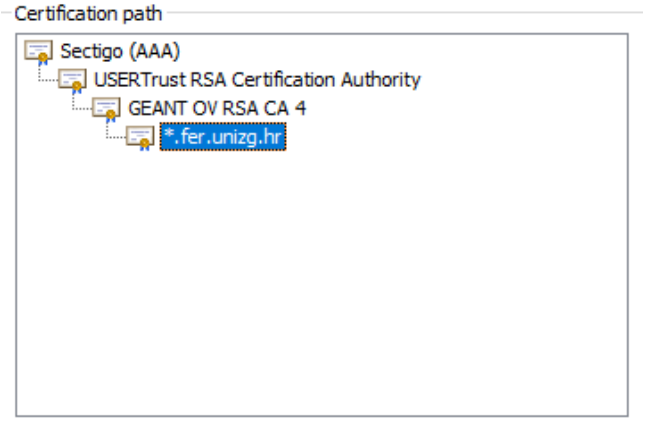
Upravljanjem certifikatima se brine tijelo za ovjere (*engl. certificate authority (CA)*). Svako tijelo za ovjere ima u svojoj bazi popis svih certifikata koje je izdalo te sve potrebne informacije koje

idu uz certifikat. Tijela za ovjeru se smatraju pouzdanim trećim stranama za izdavanje certifikata. Svako tijelo za ovjeru ima nadređeno tijelo za ovjeru osim korijenskog tijela te se na taj način uspostavlja povjerenje među njima.

Primjer hijerarhije certifikata je prikazano na slici 3.4. gdje je prikazana hijerarhija od korijenskog certifikata tj. tijela za ovjeru do najnižeg certifikata u hijerarhiji.



Slika 3.3: Primjer valjanog certifikata www.fer.unizg.hr



Slika 3.4. Primjer hijerarhije certifikata za www.fer.unizg.hr

Digitalni certifikati se koriste u komunikaciji fiskalnih blagajni gdje se certifikat šalje zajedno s podacima za fiskalizaciju. Na slici 3.5. se vidi kako bi izgledao jedan certifikat koji se nalazi u fiskalnoj blagajni i koji služi za potvrdu identiteta vlasnika blagajne, tj. obrta koji tu blagajnu koristi.

```
 <X509Certificate>

MIIEyDCCA7CgAwIBAgIEPssQ2TANBgkqhkiG9w0BAQUFADArMQswCQYDVQQGEwJIUjE NMAsGA1UEChMERklOQTENMAsGA1UE
CxMEREVNTzAeFw0xMjA5MjcxMDQ5MThaFw0xNDA5MjcxMTE5MT haMFkxCzAJBgNVBAYTAkhSMSYwJAYDVQQKEx1BUElTIElU
ICBELk8uTy4gSFIwMjk5NDY1MDE5OTEPMA0G A1UEBxMGWkFHUkVCMREwDwYDVQQDEwhGSVNLQUwgMjCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCA QoCggEBAOEPIO7AMxdO5o6jw5BUy2UAKXdtdYavlEtVPVRwoBxA4YxPzDFjsnJGapAH4DW7YKqgLLLT
6EjrK kCvjOpgr5LsPUjZk7/gZrVNpcBIzm6ECiywBGjB623/kdWqYdd9Al+1KbOVa+PRp9D7JzjLQTZaXTc4FR5wGo 4w9jC
22ixqb+F6aDzF6TLsICJVKFWV4g6glMLlKgsrZxk+O55/1f7o0442dCGuZ7IQtp1UK7TH9a7KilvLNv+1 OKyZDZfzYGyPVkB
/Tt7N5NO0R1ICbeATvDcDyl2BQfECYC+z+3dfDybcOFF+QmBsntIKO4p7+LxR/Yob5/Gx PrIi6KVyDdECAwEAAaOCAcQwggH
AMAsGA1UdDwQEAwIFoDBCBgNVHSAEOzA5MDcGCSt8iFAFHwUDATAq MCgGCCsGAQUFBwIBFhxodHRwOi8vZGVtby1wa2kuZml
uYS5oci9jcHMvMCQGA1UdEQQdMBuBGXNhbmRy YS5wb3BvdmljQGFwaXMtaXQuaHIwgc4GA1UdHwSBxjCBwzBCoECgPqQ8MDo
xCzAJBgNVBAYTAkhSMQ0w CwYDVQQKEwRGSU5BMQ0wCwYDVQQLEwRERU1PMQ0wCwYDVQQDEwRDUkw3MH2ge6B5hk9sZGFwOi8
 vZGVtby1sZGFwLmZpbmEuaHIvb3U9REVNTyxvPUZJTkEsYz1IUj9jZXJ0aWZpY2F0ZVJldm9jYXRpb25MaXN0JTNCYmluYXJ
5hiZodHRwOi8vZGVtby1wa2kuZmluYS5oci9jcmwvZGVtb2NhLmNybDArBgNVHRAEJDAig A8yMDEyMDkyNzEwNDkxOFqBDzI
wMTQwOTI3MTExOTE4WjAfBgNVHSMEGDAWgBR6YCOOSJ0ya6TlLd24 WbSU/EJinjAdBgNVHQ4EFgQUK/IWmnKEKdD/PLV06Cd
dkYi7WrAwCQYDVR0TBAIwADANBgkqhkiG9w0BA QUFAAOCAQEAABb3RnPlwEBNKEWy3YsFRwJSZoMTBGwv7Q0sWf9nTHMS3vg
FqNUSx8iqP99tnoppav7Td WoQ5zBtW935Nkev7rhqGADT4R/7pYeYc2R+Yjsbn6rxo6zDv/oIrd8LtFHccx3iUYgt/n1J5Qi
GIQ601fXwYftua VcaMnwsVBCuFy8vut4PiSPVsfML/R/oLQ3IqKJmNMXy4Rm8qjeOdgLyYwTuz7oKZ+N4+kq4tpuuJn86EUk
3q sQo+FOVfwFi7cew7qtDYHapqAC4zTb6isqB8/Wcagucb4zNU3Vgeixz5Nsd 8g28hcC/OzhHHGZpQvB0WVF5YlF1Od7AYB
L+DDI9Lg==

</X509Certificate>
```

Slika 3.5. Primjer certifikata unutar XML dokumenta

3.4. HTTP protokol

Hypertext Transfer Protocol (HTTP) je protokol preko aplikacijskog sloja TCP/IP modela. HTTP je vrsta zahtjev-odgovor sustava u arhitekturi klijent-poslužitelj. Njegova najraširenija primjena je World Wide Web (WWW). Preko njega je moguće slati sve vrste podataka pomoću raznih metoda kao što su:

1. GET: zahtijeva od poslužitelja prikaz specificiranog resursa. Prema specifikaciji, ova metoda bi trebalo samo dohvaćati resurse [4].
2. POST: šalje zahtjev u kojem traži od poslužitelja da izmijeni ili kreira novu instancu resursa na bazi podataka.
3. PUT: zamjenjuje sve tražene resurse s novim koji se šalju u zahtjevu.
4. DELETE: briše specificirani resurs.

Slanje HTTP zahtjeva:

```
Request URL: http://localhost:8888/
Request Method: GET
Referrer Policy: strict-origin-when-cross-origin
```

Također postoje odgovori koji se šalju klijentu s obzirom na uspješnost zahtjeva:

1. 1xx: Informacijski odgovori –poslužitelj mora procesirati zahtjev duže vremena.
2. 2xx: Uspješni odgovori – zahtjev je uspješno izvršen.
3. 3xx: Preusmjeravanja – resurs premješten na neku drugu lokaciju i slično.

4. 4xx: Klijentska greška – zahtjev nije dobro definiran, klijent nema dovoljna prava za tu akciju itd.
5. 5xx: Poslužiteljska greška – poslužitelj ne zna kako bi postupio s greškom, zatražena metoda nije implementirana itd.

Primljeni HTTP odgovor:

```
Status Code: 200 OK
Content-Length: 11
Content-Type: text/html; charset=UTF-8
Date: Wed, 19 May 2021 15:49:21 GMT
Etag: "4a89adc661e3a6477184883028eccf2dd7c7ca64"
Server: TornadoServer/6.1
```

3.5. XML

eXtensible Markup Language (XML) je jednostavan i proširiv format teksta. XML služi za čuvanje i prijenos podataka. Dizajniran je da bude razumljiv čovjeku i računalu. Napravljen je prema Standard Generalized Markup Language (SGML; ISO 8879) jeziku. XML ima stroga pravila pisanja. Upravo zbog toga ima mogućnost validacije podataka, može se strojno provjeriti struktura i valjanost podataka. Također je prenosiv jer je obična tekstualna datoteka. Vrlo je rasprostranjen i dobro je podržan u većini programskih jezika [5].

U kontekstu ovog testiranja XML dokumenti služe za spremanje i slanje računa između blagajne i poslužitelja Porezne uprave. Na slici 2.6. su vidljivi parametri koji se šalju iz blagajne poslužitelju.

Lxml je vanjska knjižica programskog jezika Python [6] koja omogućuje jednostavno rukovanje XML dokumentima. Ova knjižnica nudi sljedeće beneficije: jednostavnost pri korištenju, odlična brzina raščlanjivanja dokumenata. Dokumentacija je opširna i razumljiva, kao i primjeri za korištenje unutar nje. Također nudi jednostavnu konverziju u tipove podataka korištene u programskom jeziku Python, što za posljedicu ima jednostavno i brzo rukovanje s dokumentima.

4. Implementacija sustava za ispitivanje fiskalnih blagajni

Za implementaciju sustava je korišten programski jezik Python [7] zbog svoje jednostavnosti i sintakse kako bi kod ostao relativno kratak i jednostavan za čitanje te zbog raznolikih vanjskih knjižnica. Razvojno okruženje korišteno za ostvarivanje sustava je PyCharm [8]. Ovo razvojno okruženje je odabrano zbog svoje integracije s Python-om. PyCharm je najpopularnije razvojno okruženja za Python projekte [9].

U sljedećih nekoliko poglavlja se opisuje kako je sustav implementiran, poglavlja započinju objašnjenjem što su to vanjske knjižnice. U poglavlju 4.2 će biti riječi o odabiru radnog okvira za implementaciju sustava za ispitivanje. Način na koji je implementiran poslužitelj za ispitivanje te knjižnice korištenje za implementaciju su opisani u poglavlju 4.3. U poglavlju 4.4 se opisuje implementacija klijenta za ispitivanje te njegove pripadne vanjske knjižnice. Za kraj se opisuje implementacija baze podataka te potrebne vanjske knjižice.

4.1. Vanjske knjižnice

Knjižnica (*engl. library*) u smislu programskih jezika je dio koda koji se može uvesti u program kako bi dobili potrebne funkcionalnosti bez pisanja vlastite implementacije. Programski jezici imaju standardne knjižice uvezene pri samom pokretanju programa. Knjižnice mogu, osim dodavanja novih funkcionalnosti, implementirati nove tipove podataka, kao što su liste, polja i stogovi. Također mogu dodavati dodatna upozorenja i greške koje standardna knjižnica ne vidi ili nije podešena na taj način. Kod knjižnica je poslagan tako da ga mogu koristiti razni programi koji nisu međusobno povezani.

Ovaj sustav je implementiran u Python-u te se u njemu u vanjskim knjižnica nalaze moduli. Moduli se dio koda koji dodaje funkcionalnost i uvoze se u program pomoću `import` funkcije. Najveća prednost modula je što daju veću preglednost koda i laku implementaciju. Python-ova standardna knjižnica se brine o točnoj sintaksi, semantici i može se pobrinuti za osnovne naredbe poput I/O funkcionalnosti.

4.2. Odabir radnog okvira za implementaciju sustava

Za potrebe ovog projekta razmatrala su se tri radna okvira: Tornado [10], Twisted + Flask [11] [12] i Django [13]. Sva tri imaju mogućnosti implementacije klijentske i poslužiteljske strane sustava sa svojim specifičnim funkcionalnostima. Django otpada odmah na početku jer je previše opsežan za potrebe ovog sustava. Temelji se na predlošcima koji se zatim slažu u jednu cjelinu.

Nadalje Flask je jednostavan za rad i implementaciju potrebnih funkcija ali zbog njegovih loših performansi pod velikim teretom i ne mogućnosti obrade više klijenata odjednom mu je potrebno dodati Twisted radni okvir. Twisted nudi poboljšanje svih nedostataka Flask-a ali ima negativnu stranu gdje implementacija i povezivanje nije trivijalno. Konačno, ostaje Tornado radni okvir koji nudi sve potrebno te dobru implementaciju nedostataka ostalih radnih okvira. Tornado nudi obradu više korisnika odjednom, ima potpunu integraciju za klijentsku i poslužiteljsku stranu sustava te nudi jednostavno implementiranje potrebnih funkcija.

4.3. Implementacija poslužitelja za ispitivanje fiskalnih blagajni

Poslužitelj za ispitivanje fiskalne blagajne je ostvaren pomoću Tornado knjižnice. Tornado je radni okvir za web i asinkrona mrežna knjižnica. Koristeći ne blokirajuće mrežne I/O, Tornado je pogodan za skaliranje na desetke tisuća istovremenih otvorenih konekcija što je u ovom sustavu vrlo bitno ukoliko se više korisnika želi povezati u isto vrijeme. Tornado radni okvir zajedno s HTTP poslužiteljem nudi ravnopravnu alternativu WSGI-u [10]. Asinkronost i ne blokirajuće funkcije su se često poistovjećuju, ali nisu baš iste. Funkcija je blokirajuća kada čeka nešto da se dogodi prije vraćanja rezultata. Funkcija može blokirati iz mnogih razloga: mrežni I/O, diskovni I/O itd. Svaka funkcija je zapravo blokirajuća, barem na kratko, dok koristi resurse procesora. Što se tiče Tornada najbitnija su blokiranja s obzirom na mrežni I/O. Asinkrona funkcija vraća rezultat prije nego završi s izvođenjem i uzrokuje da se dodatni rad izvrši u pozadini koji će vratiti objekt najčešće putem `await` ili `yield` funkcije. Funkcije u sustavu su pisane na način da budu asinkrone i ne blokirajuće jer Tornado koristi jedno-dretvene petlje što znači da u jednom trenutku može biti aktivna samo jedna operacija.

`tornado.web` je jednostavni web radni okvir unutar Tornado knjižnice sa asinkronim značajkama. Poslužitelj je dio jedne klase MainHandler u kojoj su sve HTTP metode koje želimo omogućiti. U ovom sustavu je to HTTP metoda POST koja u tijelu zahtjeva prima identifikator blagajne i XML koji je potrebno testirati.

Poslužitelj se pokreće na sljedeći način:

```
app = make_app()
app.listen(8888)
tornado.ioloop.IOLoop.current().start()
```

`make_app` je funkcija unutar poslužitelja koja vraća aplikaciju tip tornado.web.Application koji određuje na kojoj lokaciji će se aplikacija slušati i koju klasu će koristiti, u ovom slučaju to su „/“ i MainHandler:

```
def make_app():
      return tornado.web.Application([
            (r"/", MainHandler),
      ])
```

`app.listen` je funkcija kojom se aplikaciji govori na kojem pristupu da sluša. U ovom slučaju to je pristup 8888.

Posljednja linija radi nekoliko stvari:

- `tornado.ioloop` je modul u kojem se dio na pokretanje nalazi.
- `IOLoop` je klasa koja sadrži I/O petlju.
- `current` je funkcija unutar IOLoop klase koja dohvaća trenutni I/O petlju dretve.
- `start` započinje I/O petlju sve dok ga se ne zaustavi pomoću funkcije za zaustavljanje ili gašenjem poslužitelja.

Poslužitelj obrađuje dobivene podatke pomoću funkcija koje su nalaze u plugin_controller.py datoteci. Ta datoteka sadrži nekoliko funkcija od kojih `check_output` traži potrebne elemente, pokreće test, pretražuje rezultat kako bi našao greške ukoliko ih ima te u posljednjem koraku pokreće `updatedb` funkciju koja sprema rezultate testa u bazu podataka. Također postoji `getplugins` funkcija koja s obzirom na identifikator blagajne poslan u zahtjevu klijenta nalazi potrebne dodatke koje treba pokrenuti.

4.4. Implementacija klijenta za ispitivanje fiskalnih blagajni

Klijent za ispitivanje fiskalnih blagajni je pojednostavljena verzija fiskalne blagajne koja može slati vlastiti identifikator zajedno sa zahtjevom za fiskalizaciju u XML obliku na poslužitelj. Klijent je također implementiran pomoću Tornado knjižnice [10]. Tornado nudi i implementaciju za klijentski dio aplikacije. Za ovaj sustav koristimo asinkroni HTTP klijent. Klijent prvo čita XML datoteku čiju lokaciju čita iz config.ini datoteke. Config.ini je konfiguracijska datoteka kojoj se podešavaju parametri nekog programa. U ispisu 4.1 se vidi prikaz jedne takve datoteke. Za čitanje te datoteke se koristi ConfigParser knjižnica [14]. Ta knjižnica nudi veliku kolekciju metoda koje obrađuju konfiguracijske datoteke kao što je u našem slučaju config.ini. Takve datoteke su podijeljene u sekcije kao što su u našem primjeru *database i file***.** Unutar tih sekcija se nalaze atributi u obliku ključ – vrijednost. ConfigParser knjižnica čita te datoteke tako da se najprije odabere sekcija i zatim odabere ključ čiju vrijednost želimo pročitati.

Prikaz načina na koji se piše konfiguracijska datoteka:

```
[database]
host = ec2-54-74-156-137.eu-west-1.compute.amazonaws.com
database_name = d4rqq6mn2b2d3j
user = jepygaecuqjkfy
port = 5432
password = 9bfa293a8594509b144397277e45ec4ff84a0e1d0ebaa097afeb0696d3e71a4d

[file]
input = .\Data\RequestExamples\9.2_Poruka_zahtjeva_za_racun_certifikat.xml
```

Ispis 4.1: Konfiguracija spremljena u datoteku config.ini

Inicijalizacija ConfigParsera kako bi dobili potrebne funkcionalnosti:

```
configur = ConfigParser()
```

Čitanje datoteke iz koje želimo pročitati konfiguraciju:

```
configur.read('config.ini')
```

Odabir sekcije unutar koje se nalazi potrebni ključ i odabir ključa:

```
configur.get('file', 'input')
```

Nakon čitanja konfiguracije iz datoteke, klijent ima sve potrebne argumente za slanje HTTP zahtjeva na poslužitelj za testiranje fiskalne blagajne. Šalje se HTTP zahtjev POST metodom u kojoj se u tijelu šalju identifikator blagajne i XML zahtjeva za fiskalizaciju. Unutar Tornado knjižnice se koristi `tornado.httpclient.AsyncHTTPClient.fetch` funkcija koja objedinjuje HTTP zahtjev, POST metodu i tijelo zahtjeva koje želimo poslati. Nakon slanja zahtjeva, klijent čeka na odgovor poslužitelja u kojem dobiva ili odgovor u XML obliku ili grešku ukoliko je došlo do greške u radu poslužitelja.

4.5. Implementacija baze podataka

U ovom sustavu se koristi relacijska baza podataka preko sustava za upravljanje bazom podataka PostgreSQL [15]. Relacijske baze podataka se sastoje od relacija, odnosno tablica koje su definirane svojim imenom i skupom atributa, te omogućavaju brzu i jednostavnu pohranu velike količine podataka. PostgreSQL je sustav za upravljanje bazama podataka (SUBP) otvorenog koda koji se velikom većinom drži SQL:2011 standarda.

Baza podataka je podešena na Internetu pomoću Heroku-a [16]. Heroku je platforma u Internetskom oblaku koja omogućuje programerima da izgrade, pokrenu i održavaju aplikacije potpuno u oblaku. Ujedno nudi rješenja za razne programske jezike, pa tako i za baze podataka. Preko Heroku-a je pokrenuta instanca poslužitelja PostgreSQL baze podataka.

Za pristup bazi podataka Heroku stvara potrebne parametre za spajanje na poslužitelj baze, najkorišteniji takav poslužitelj je PgAdmin4 [17]. Preko PgAdmin4-a je moguće kreirati tablice, korisnike i sve potrebne elemente za sustav. Kreacija tablica se obavlja *CREATE* naredbom unutar PgAdmin4-a.

Kreiranje tablice *tests*:

```
CREATE TABLE tests (
      id INTEGER PRIMARY KEY,
      name VARCHAR(50),
      type VARCHAR(50),
      description VARCHAR(500)
);
```

Kreiranje tablice *teststorun*:

```
CREATE TABLE teststorun (
      id INTEGER PRIMARY KEY,
      tests VARCHAR(200),
      cashRegister_id INTEGER
);
```

Kreiranje tablice *finishedtests*:

```
CREATE TABLE finishedtests (
      id INTEGER PRIMARY KEY,
      zki VARCHAR(50),
      jir VARCHAR(50),
      plugin_used VARCHAR(500),
      datetime timestamp with time zone,
      conclusion VARCHAR(500),
      cashRegister_id INTEGER
);
```

Psycopg je najpopularnija vanjska knjižnica koja spaja Python programe s PostgrSQL bazom podataka te se koristi u ovom sustavu. Glavna značajka ove knjižice je potpuna integracija s Python-ovom specifikacijom za baze podataka [18] i očuvanje sigurnosti dretve što omogućuje da više dretvi koristi istu konekciju. Dizajniran je za kompleksne višedretvne aplikacije koje stvaraju i uništavaju puno kursora i stvaraju velik broj istovremenih konekcija [19]. Kursori su privremeni podaci u memoriji nakon pokretanja SQL naredbe. Iz njih je moguće čitati redove, mijenjati ili brisati dijelove redaka.

Konekcija se stvara pomoću `connect` funkcije koja prima parametre za povezivanje s bazom podataka, oni su također spremljeni u konfiguracijsku datoteku. U ispisu 4.1 pod sekcijom *database* se mogu vidjeti parametri potrebni za povezivanje s bazom podataka u ovom sustavu. Nakon povezivanja se stvaraju kursori pomoću kojih se izvršavaju SQL naredbe na bazi podataka i spremaju se u kursor. Konekcija se zatvara naredbom `close`.

5. Zaključak

U ovom radu su opisani bitni dijelovi sustava za fiskalizaciju zajedno s sustavom za testiranje fiskalnih blagajni u koje spadaju klijent, poslužitelj, baza podataka i sustav dodataka. Svi su oni ključni za neometani rad sustava. Nadalje, opisani su i procesi koji se odvijaju u procesu fiskalizacije između klijenta fiskalne blagajne i poslužitelja Porezne uprave zbog kojih je moguća sigurna i točna razmjerna poruka i podataka te fiskalizacija. Prikazani su i dijelovi enkripcije koji ulaze u ovaj proces kako nitko drugi osim fiskalne blagajne i poslužitelja Porezne uprave ne bi mogao utjecati na poruku i informacije koje se razmjenjuju među njima.

Ovaj sustav je napravljen kako bi se došlo do univerzalnog i automatiziranog rješenja za testiranje programskih rješenja fiskalnih blagajni pošto postoje razne implementacije od raznih proizvođača. Opisana je implementacija programskog rješenja za poslužitelj Porezne uprave, klijent fiskalne blagajne i baze podataka te sve potrebne vanjske knjižnice korištene u izradi sustava.

Kroz zadnjih 3 mjeseca rada na ovom radu sam naučio mnogo informacija o procesu fiskalizacije i o procesu testiranja. Mislim da ovo rješenje može služiti kao osnova za kompleksnije sustave za ispitivanje fiskalnih blagajni. Također mislim da je ovaj rad koristan svima koji žele bolje razumjeti kako radi fiskalizacija i komunikacija klijenta i poslužitelja općenito.

Rezultati ovog rada su zadovoljavajući, s obzirom na dostupno vrijeme i s obzirom da nismo imali pravu fiskalnu blagajnu za testiranje. Ima nekoliko stvari koje bi bilo potrebno doraditi i napraviti. Prvi korak bi bio omogućavanje provjera na strani klijenta koji bi radio dodatne provjere nakon što primi odgovor od poslužitelja Porezne uprave te bi nakon svoje provjere spremao rezultate u bazu podataka te bi se na kraju rezultati poslužitelja i klijenta uspoređivali i tražile bi se razlike među njima. Drugi korak bi bila potpuna automatizacija testiranja, gdje bi se za sve kombinacije testova koji su implementirani u sustavu radile permutacije. Ne bi bilo potrebno posebno označavati testove koje želimo pokrenuti iako bi to bila opcija ukoliko korisnik tako želi. Treći korak bi bio omogućavanje dodavanja novih dodataka bez ručnog dodavanja u direktorij, automatizacija tog procesa, primjerice putem GitHub-a [20].

6. Literatura

[1] »Fina,« [Mrežno]. Available: https://www.fina.hr/fiskalizacija#info. [Pokušaj pristupa 20 Svibanj 2021].

[2] A. Đerek, S. Groš, M. Mikuc i M. Vuković, Ožujak 2021. [Mrežno]. Available: https://www.fer.unizg.hr/_download/repository/02._Osnove_kriptografije_i_kriptoanalize_--_prvi_dio.pdf. [Pokušaj pristupa 19 Svibanj 2021].

[3] J. Yau, 20 Kolovoz 2020. [Mrežno]. Available: https://www.hellosign.com/blog/electronic-signature-laws-around-the-world-a-look-at-esignature-laws-by-country. [Pokušaj pristupa 18 Svibanj 2021].

[4] R. Fielding i J. Reschke, »IETF,« Lipanj 2014. [Mrežno]. Available: https://datatracker.ietf.org/doc/html/rfc7231#section-4. [Pokušaj pristupa 16 Svibanj 2021].

[5] »W3C,« 11 Listopad 2016. [Mrežno]. Available: https://www.w3.org/XML/. [Pokušaj pristupa Svibanj 2021].

[6] »lxml,« 21 Ožujak 2021. [Mrežno]. Available: https://lxml.de/. [Pokušaj pristupa 21 Svibanj 2021].

[7] »Python,« [Mrežno]. Available: https://www.python.org/. [Pokušaj pristupa 19 Svibanj 2021].

[8] »JetBrains PyCharm,« 2021. [Mrežno]. Available: https://www.jetbrains.com/pycharm/. [Pokušaj pristupa Svibanj 2021].

[9] »Geeks for Geeks,« 7 Veljača 2020. [Mrežno]. Available: https://www.geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/. [Pokušaj pristupa 20 Svibanj 2021].

[10] »Tornado Documentation,« [Mrežno]. Available: https://www.tornadoweb.org/en/stable/index.html. [Pokušaj pristupa 17 Svibanj 2021].

[11] »Twisted,« 2021. [Mrežno]. Available: https://twistedmatrix.com/trac/. [Pokušaj pristupa Ožujak 2021].

[12] »Flask,« 2010. [Mrežno]. Available: https://flask.palletsprojects.com/en/2.0.x/. [Pokušaj pristupa Ožujak 2021].

[13] »Django,« 2021. [Mrežno]. Available: https://www.djangoproject.com/. [Pokušaj pristupa Ožujak 2021].

[14] »Python Documentation,« 2021. [Mrežno]. Available: https://docs.python.org/3/library/configparser.html. [Pokušaj pristupa Svibanj 2021].

[15] »PostgreSQL,« [Mrežno]. Available: https://www.postgresql.org/. [Pokušaj pristupa 20 Svibanj 2021].

[16] »Heroku,« [Mrežno]. Available: https://www.heroku.com/home. [Pokušaj pristupa 17 Svibanj 2021].

[17] »pgAdmin,« [Mrežno]. Available: https://www.pgadmin.org/. [Pokušaj pristupa 18 Svibanj 2021].

[18] »Python,« 29 Ožujak 2001. [Mrežno]. Available: https://www.python.org/dev/peps/pep-0249/. [Pokušaj pristupa 25 Svibanj 2021].

[19] »Psycopg – PostgreSQL database adapter for Python,« 2020. [Mrežno]. Available: https://www.psycopg.org/docs/index.html. [Pokušaj pristupa 5 Svibanj 2021].

[20] »GitHub,« 2021. [Mrežno]. Available: https://github.com/. [Pokušaj pristupa 2021].

[21] »Porezna uprava,« [Mrežno]. Available: https://www.porezna-uprava.hr/HR_Fiskalizacija/Stranice/FiskalizacijaNovo.aspx. [Pokušaj pristupa 16 Svibanj 2021].

Implementacija poslužiteljskog sustava za ispitivanje ispravnosti fiskalnih blagajni

Sažetak

Fiskalna blagajna je programsko rješenje koje služi u procesu fiskalizacije. Fiskalizacija je proces u kojem obrti i slični poslovi šalju Poreznoj upravi podatke o računu prije izdavanja računa stranci. Time se omogućuje Poreznoj upravi efikasniji nadzor nad gotovinskim prometom. Programska rješenja fiskalnih blagajni su rađena od strane raznih proizvođača. Samim time dolazi do pitanja koliko su te implementacije ispravne, odnosno bez pogrešaka u kodu. Zbog toga oportuno je razviti sustav za ispitivanje ispravnosti fiskalnih blagajni. U ovom radu se opisuju elementi koju ulaze u proces fiskalizacije i komunikacije između klijenta fiskalne blagajne i poslužitelja Porezne uprave. Također se opisuje sustav koji je razvijen za navedeni problem i svi njegovi dijelovi.

Implementation of server system for testing correctness of fiscal cash registers

Abstract

The fiscal register is a software solution that serves in the process of fiscalization. Fiscalization is a process in which trades and similar businesses send invoice data to the Tax Administration before issuing invoices to customers. This enables the Tax Administration to control cash transactions more efficiently. Software solutions for fiscal registers were made by various manufacturers. This raises the question of how correct these implementations are, and without error in the code. Therefore, it is opportune to develop a system for examining the correctness of fiscal registers implementations. This paper describes the elements used in the process of fiscalization and communication between the fiscal register client and the Tax Administration server. It also describes the system developed for the said problem and all its parts.