|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NOVOM SADU  **FAKULTET TEHNIČKIH NAUKA U NOVOM SADU** |  |

Nataša Spremo

**PRIMENA VIRTUALNE PRIVATNE MREŽE**

DIPLOMSKI RAD

- Osnovne akademske studije -

Novi Sad, 2021. godine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | | | |
| **КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА** | | | |
| Редни број, **РБР**: | | |  | |
| Идентификациони број, **ИБР**: | | |  | |
| Тип документације, **ТД**: | | | Монографска публикација | |
| Тип записа, **ТЗ**: | | | Текстуални штампани документ/ ЦД | |
| Врста рада, **ВР**: | | | Дипломски рад | |
| Аутор, **АУ**: | | | Наташа Спремо | |
| Ментор, **МН**: | | | др Бранислав Атлагић, доцент | |
| Наслов рада, **НР**: | | | Развој клијентске Веб апликације за аквизиционо управљачке системе | |
| Језик публикације, **ЈП**: | | | Српски ( латиница ) | |
| Језик извода, **ЈИ**: | | | Српски/енглески | |
| Земља публиковања, **ЗП**: | | | Србија | |
| Уже географско подручје, **УГП**: | | | Војводина | |
| Година, **ГО**: | | | 2020. | |
| Издавач, **ИЗ**: | | | Ауторски репринт | |
| Место и адреса, **МА**: | | | Факултет техничких наука (ФТН), Д. Обрадовића 6, 21000 Нови Сад | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | | 10/30/13/3/18/0/0 | |
| Научна област, **НО**: | | | Електротехничко и рачунарско инжењерство | |
| Научна дисциплина, **НД**: | | | Примењене рачунарске науке и информатика | |
| Предметна одредница/Кључне речи, **ПО**: | | | Мрежа, комуникација, TCP, IPsec, безбедност на интернету | |
| **УДК** | | |  | |
| Чува се, **ЧУ**: | | | Библиотека ФТН, Д. Обрадовића 6, 21000 Нови Сад | |
| Важна напомена, **ВН**: | | |  | |
| Извод, **ИЗ**: | | | Фали | |
| Датум прихватања теме, **ДП**: | | |  | |
| Датум одбране, **ДО**: | | |  | |
| Чланови комисије, **КО**: | | Председник: | др Ранко поповић, ред.проф |
|  | | Члан: | др Милана Бојанић, доцент | Потпис ментора |
|  | | Члан, ментор: | др Бранислав Атлагић, доцент |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**  21000 NOVI SAD, Trg Dositeja Obradovića 6 | | | |
| **KEY WORDS DOCUMENTATION** | | | |
| Accession number, **ANO**: | | |  | |
| Identification number, **INO**: | | |  | |
| Document type, **DT**: | | | Monographic publication | |
| Type of record, **TR**: | | | Textual material, printed/CD | |
| Contents code, **CC**: | | | Bachelor thesis | |
| Author, **AU**: | | | Nataša Spremo | |
| Mentor, **MN**: | | | dr Branislav Atlagić, assistant professor | |
| Title, **TI**: | | | Applying Virutal Private Network | |
| Language of text, **LT**: | | | Serbian | |
| Language of abstract, **LA**: | | | Serbian/English | |
| Country of publication, **CP**: | | | Serbia | |
| Locality of publication, **LP**: | | | Vojvodina | |
| Publication year, **PY**: | | | 2020. | |
| Publisher, **PB**: | | | Author’s reprint | |
| Publication place, **PP**: | | | Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | | 10/30/13/3/18/0/0 | |
| Scientific field, **SF**: | | | Electrical and computer engineering | |
| Scientific discipline, **SD**: | | | Applied computer science and informatics | |
| Subject/Key words, **S**/**KW**: | | | Network, communication, TCP, IPsec, security on internet | |
| **UC** | | |  | |
| Holding data, **HD**: | | | Library of Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad | |
| Note, **N**: | | |  | |
| Abstract, **AB**: | | | Фали | |
| Accepted by the Scientific Board on, **ASB**: | | |  | |
| Defended on, **DE**: | | |  | |
| Defended Board, **DB**: | | President: |  |
|  | | Member: |  | Menthor's sign |
|  | | Member, Mentor: | dr Branislav Atlagic, assistant professor |  |

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист/Листова: |
|  |

*(Податке уноси предметни наставник - ментор)*

|  |  |
| --- | --- |
| Врста студија: | ☒ Основне академске студије  ☐ Основне струковне студије |
| Студијски програм: | Примењено софтверско инжењерство |
| Руководилац студијског програма: | др Александар Селаков |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | Наташа Спремо | Број индекса: | **ПР 27/2016** |
| Област: | Примена рачунарских мрежа у инфраструктурним системима | | |
| Ментор: | др. Бранислав Атлагић | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА  ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:**

|  |
| --- |
| Примена виртуелне приватне мреже |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| Развити конзолну апликацију где клијент и сервер иницијално комуницирају преко TCP протокола на локалној мрежи. Омогућити повезивање клијента преко нове VPN адресе са сервером, уколико је виртуелна приватна мрежа активирана. Могућа даља комуникација преко VPNа или затварање везе. |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за: ☐ - Студента; ☐ - Ментора |

Образац **Q2.НА.15-04** - Издање 1

**SPISAK SKRAĆENICA**

|  |  |
| --- | --- |
| ***VPN*** | ***Virtual Private Network*** |
| ***IP*** | ***Internet Protocol*** |
| ***IPsec*** | ***Internet Protocol Security*** |
| ***TCP*** | ***Transmission Control Protocol*** |
| ***UDP*** | ***User Datagram Protocol*** |
|  |  |
| ***WEB*** | ***World Wide Web*** |
| ***TCP*** | ***Transmission Control Protocol*** |
| ***HTTP*** | ***HyperText Transfer Protocol*** |
| ***IIS*** | ***Internet Information Services*** |
| ***HTML*** | ***HyperText Markup Language*** |
| ***IP*** | ***Internet Protocol*** |
| ***REST*** | ***Rеpresentational State transfer*** |
| ***JSON*** | ***JavaScript Object Notation*** |
| ***XML*** | ***Extensible Markup Language*** |
| ***WAN*** | ***Wide Area Network*** |
| ***CSS*** | ***Cascade Style Sheet*** |
| ***WCF*** | ***Windows Communication Foundation*** |
| ***XSD*** | ***XML Schema Definition*** |

**SPISAK KORIŠĆENIH SLIKA**

|  |  |
| --- | --- |
| ***Slika 1.1*** | ***Prikaz rješavanog problema*** |
| ***Slika 2.1*** | ***Uprošćen prikaz SCADA arhitekture*** |
| ***Slika 3.1.1*** | ***Uspostavljanje veze koristeći TCP protokol*** |
| ***Slika 4.1.1*** | ***Klijent-server arhitektura*** |
| ***Slika 4.2.1*** | ***Web server*** |
| ***Slika 5.1*** | ***Krajnja tačka*** |
| ***Slika 5.2.1*** | ***Arhitektura WCF-a*** |
| ***Slika 7.1*** | ***Arhitektura rješenja*** |
| ***Slika 7.1.1*** | ***Two-Way Binding*** |
| ***Slika 7.1.2*** | ***Komponente frontend-a*** |
| ***Slika 7.1.3*** | ***Main-table komponenta*** |
| ***Slika 7.1.4*** | ***Control-window komponenta*** |
| ***Slika 7.1.5*** | ***Log-window komponenta*** |
| ***Slika 7.1.6*** | ***Http-client-service*** |
| ***Slika 7.2.1*** | ***Arhitektura backend-a*** |
| ***Slika 8.1*** | ***Fizička organizacija sistema*** |

**SPISAK LISTINGA**

|  |  |
| --- | --- |
| ***Listing 7.1*** | ***Definicija grupe registara*** |
| ***Listing 7.2*** | ***RegisterData klasa*** |
| ***Listing 7.2.1*** | ***WCF interfejs*** |
| ***Listing 7.2.2*** | ***RegisterController*** |

**SPISAK TABELA**

|  |  |
| --- | --- |
| ***Tabela 3.1*** | ***ISO/OSI referentni model*** |
| ***Tabela 7.1*** | ***Operacije nad registrima*** |

**Sadržaj**

[***1.***](#_heading=h.3dy6vkm) ***Uvod 9***

[***2.***](#_heading=h.1t3h5sf) ***Osnove VPNa 10***

[**2.1**](#_heading=h.4d34og8) **VPN Tehnologije 10**

[**2.2**](#_heading=h.17dp8vu) **Kriterijumi odabira VPN provajdera 11**

[***3.***](#_heading=h.3rdcrjn) ***Sigurnosni mehanizmi 12***

[**3.1**](#_heading=h.26in1rg) **IPsec 12**

[**3.2**](#_heading=h.lnxbz9) **SSL/TLS 13**

[***4.***](#_heading=h.35nkun2) ***Web aplikacije 14***

[**4.1**](#_heading=h.1ksv4uv) **Klijent-server arhitektura 14**

[**4.2**](#_heading=h.44sinio) **Web server 15**

[**4.3**](#_heading=h.2jxsxqh) **Web klijent 15**

[***5.***](#_heading=h.z337ya) ***Windows Communication foundation (WCF) 16***

[**5.1**](#_heading=h.3j2qqm3) **Komunikacioni protokoli 16**

[**5.2**](#_heading=h.1y810tw) **Arhitektura 17**

[***6.***](#_heading=h.4i7ojhp) ***Opis korišćenih tehnologija i alata 18***

[***7.***](#_heading=h.2xcytpi) ***Opis programskog rješenja 19***

[**7.1**](#_heading=h.1ci93xb) **Frontend 21**

[**7.2**](#_heading=h.3whwml4) **Backend 24**

[**7.3**](#_heading=h.2bn6wsx) **Oporavak od greški 26**

[***8.***](#_heading=h.qsh70q) ***Testiranje rješenja 27***

[***9.***](#_heading=h.3as4poj) ***Zaključak 28***

[***10.***](#_heading=h.1pxezwc) ***Literatura 29***

# **Uvod**

U današnje vreme, u pogledu bezbednosti, kako u privatnom tako i u poslovnom okruženju, od najveće važnosti je zaštita podataka i informacija koje razmenjujemo svakodnevno putem interneta.

Ukoliko se korisnici nalaze van kuće, u nekom restoranu, hotelu, aerodromu ili drugom javnom mestu, najveći broj njih će koristiti bežične javne tačke za pristup internetu, odnosno WiFi. Posmatrano sa bezbednosnog aspekta, najveći izazov predstavlja sigurnost korišćenja ovakvih pristupnih tačaka, jer korisnici nemaju uvid ko sve koristi navedenu tačku za pristup i da li se nadgleda i snima razmena podataka putem te WiFi pristupne tačke.

Kada govorimo o potrebi korisnika za radom na daljinu, odnosno radom od kuće, koji podrazumeva pristupanje resursima software-as-a-service (SaaS), koji pripadaju određenoj organizaciji ili instituciji, neophodno je da se takav rad obavlja na bezbedan način, kako ne bi došlo do moguće kompromitacije podataka kojima korisnik pristupa.

Kako bi korisnici zaštitili svoje mrežne aktivnosti, privatnost i sprečili da dođe do kompromitacije podataka organizacije preporuka je korišćenje virtuelne privatne mreže, odnosno VPN (eng. Virtual Private Network) pristup. VPN tehnologija kreira privatni, šifrovani tunel za aktivnosti na mreži i znatno otežava bilo kome da prati ili nadgleda šta korisnik radi dok je na mreži. Takođe, VPN tehnologija omogućava kompanijama bolju zaštitu od mogućeg gubitka i kompromitacije podataka, tako što se između korisničke mreže, bilo da se koristi javni ili privatni WiFi, pravi bezbedan i šifrovan tunel, preko javnog interneta do mreže organizacije.

VPN tip konekcija počeo je da se razvija još u vreme dial up modema, međutim to se ne smatra pravom virtualnom privatnom mrežom. Poslednjih nekoliko godina VPN je postao vrlo popularan zbog mnoštva vrlina koje poseduje. Naročito od kad je sve više korporacija bilo primorano da organizuje i omogući bezbedan rad od kuće. S obzirom da je ova vrsta mreže pristupačna svima, koriste je, pored kompanija, i individualni korisnici. Sigurnost, skalabilnost, niska cena, lakoća implementacije, univerzalnost najčešće su osobine koje opisuju virtualne privatne mreže.

S obzirom da se pri svakom konektovanju, nakon autentifikacije korisnika, formira već pomenuti tunel između dva čvora, pruža potpunu privatnost od spoljašnjih činioca na internetu. Pošto se koriste javne mreže troškovi su dosta niski kada se uporede sa troškovima potrebnim za iznajmljivanje privatnih linija ili implementaciju privatnih Intranet mreža. Lakoća implementacije kao jedna od osobina znači da nema potrebe za promenom postojeće infrastrukture javnih mreža, pa se VPN implementira samo na strani korisnika. Zbog enkapsulacije podataka moguće je koristiti i podatke koji pripadaju nerutabilnim protokolima. Takođe se štedi i na broju globalnih IP adresa koje kompanija mora da poseduje, što opet smanjuje cenu implementacije virtuelnih privatnih mreža.

Aplikacije koje se obično koriste, bilo da su imejl, veb, poruke, društvene mreže i sl, zasnivaju se na IP (Internet Protocol) protokolu. Iako su razvijeni određeni standardi, nisu sve internet aplikacije bezbedne, određen broj aplikacija i dalje nije usaglašen sa važećim propisima o načinu deljenja i zaštite podataka. Neujednačen stav o načinu zaštite podataka ostavlja prostora mogućim zloupotrebama od strane napadača, odnosno hakera, koji zbog ovakvih propusta jednostavno mogu doći do ličnih podataka korisnika, kao što su broj tekućeg računa, kreditne kartice, kućne adrese i sl.

VPN kreira privatni tunel preko otvorenog interneta. Ideja je da sve što korisnik pošalje bude enkapsulirano i šifrirano u ovom privatnom komunikacijskom kanalu, pa čak i ako dođe do presretanja poslatih paketa, isti ne mogu biti dešifrovani. VPN predstavlja vrlo moćan i važan alat u zaštiti i bezbednosti korisnika i njihovih podataka, ali ima i svoja ograničenja. Ovde se postavlja pitanje ograničenja VPN-a i razumevanje gde se nalazi krajnja tačka VPN servera. Ukoliko se korisnik nalazi u Srbiji, a povezuje se na VPN server u drugoj državi, celokupan saobraćaj na internetu će biti prikazan kao da je korisnik pristupio iz mreže te druge države, odnosno neće biti vidljivo da je korisnik pristupio iz IP opsega Republike Srbije.

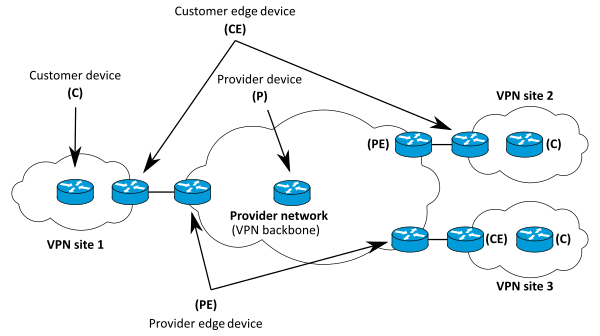
Dva najčešća tipa VPN-a su: korporativni ili poslovni VPN servis koji predstavlja poslovno orijentisano rešenje koje omogućava zaposlenima da se udaljeno povežu na korporativnu mrežu i korisnički VPN servis koji pojedincima omogućava da privatno surfuju od kuće.

# **OSNOVE VPN-a**

Virtuelna privatna mreža je mreža jedne institucije ili grupe korisnika realizovana preko javne ili deljene infrastrukture (Internet, provajderske mreže). Razlozi za uvođenje ove tehnologije su brojni, među kojima su i sigurnosni problemi, bolja organizacija saobraćaja i rutiranja, nedostatak podrške za QoS, problem broja IP adresa i migracija na Ipv6.

Postoji više vrsta i podela VPN uređaja:

* Podela prema tome kome pripadaju uređaji i gde su u VPN (Provider Provisioned VPN):
  1. C – customer device – uređaj koji se nalazi unutar korisničke mreže i nije direktno povezan sa mrežom provajdera. C uređaji nisu svesni VPN mreže.
  2. CE – customer edge device – uređaj koji se nalazi na granici korisničke mreže koji omogućuje pristup PPVPN.
  3. PE – provider edge device – uređaj ili set uređaja na granici provajder mreže koji povezuje korisničku mrežu uz pomoć CE uređaja. PE uređaji su svesni VPN mreže i održavaju stanje VPN-a.
  4. P – provider device – uređaj koji funkcioniše unutar provajder mreže i ne povezuje se direktno ni sa jednim korisničkim endpoint-om. Može pružiti rutiranje za mnoge provajder-operabilne tunele koji pripadaju različitim korisnicima PPVPN-a. Iako je P uređaj glavni činilac u implementaciji PPVPN-a, nije VPN-aware i ne održava VPN stanje, kao provider edge device.



Slika 1.1 Prikaz VPN terminologije

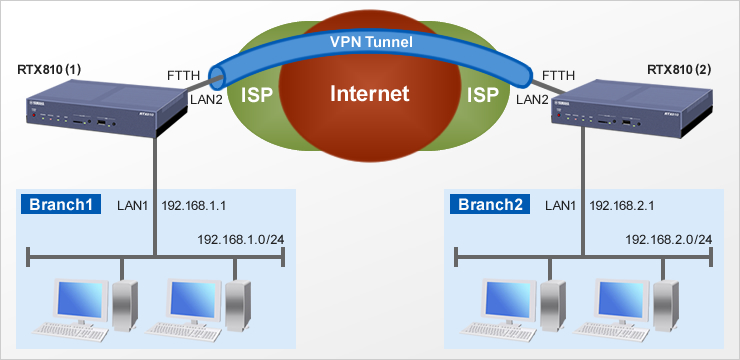
* Podela prema tome ko ih realizuje

1. Provider provisioned
2. Customer enabled

* Podela prema vrsti servisa
  1. Site-to-site (LAN-to-LAN)
     + Intranet (lokacije jedne institucije)
     + Extranet (povezivanje različitih institucija)
  2. Remote access
     + Compulsory (access server inicira VPN vezu)
     + Voluntary (klijent inicira VPN vezu)
* Podela prema sloju rada
  1. Layer 1
  2. Layer 2
  3. Layer 3
* Podela prema poverljivosti podataka
  1. Trusted VPN
  2. Secure VPN

Glavna funkcionalnost VPN mreže jeste sigurna komunikacija između autentifikovanih korisnika u čiji sadržaj nemaju uvid spoljni posmatrači. Korisnici mogu biti na geografski udaljenim lokacijama i nesmetano koristiti virtualnu privatnu mrežu, ukoliko imaju ispravne kredencijale. Prednosti VPN-a su brojne, neke od kojih su brzina, fleksibilnost, privatnost, skalabilnost i finansijske prednosti.

Sam princip rada VPN-a uvodi pojam tunelovanja (eng. *Tunneling*). Tunelovanje je sastavni deo ove mreže i predstavlja prenos paketa podataka namenjenih privatnoj mreži preko javne mreže. Ruteri javne mreže nemaju informaciju da prenose pakete koji pripadaju privatnoj mreži i VPN pakete tretiraju kao normalan saobraćaj. Pojam tunel uvodi se zbog toga što su podaci koji putuju tunelom razumljivi samo onima koji se nalaze na njegovom izvorištu i odredištu. Početak i kraj tunela se nalaze u VPN mrežama. Kada enkapsulirani paket stigne na odredišnu adresu vrši se deenkapsulacija i prosleđivanje podataka na konačno odredište. Ceo proces enkapsulacije, transporta i deenkapsulacije naziva se tunelovanje (Slika 1.2).



Slika 1.2 Prikaz rada VPN mreže

Jedna od glavnih funkcionalnosti VPN mreža je maskiranje ili promena prave IP adrese korisnika. Iako VPN provajderi preduzimaju jake mere da zaštite informacije korisnika, dešavaju se greške poput tzv. IP leak-a ili greške pri konekciji na server.

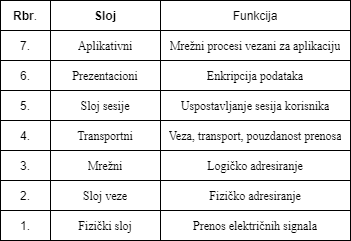
Trenutno postoji široka ponuda provajdera virtualnih privatnih mreža na internetu. Prilikom odabira treba temeljno ispitati sigurnost i bezbednost samog provajdera i usluga koje pruža.

## **VPN Tehnologije**

# Komunikacioni Protokoli

Komunikacionim protokolom se označava skup pravila i procedura koji kontroliše tok komunikacije između serijski spregnutih uređaja u okviru računarske mreže. Posredstvom komunikacionih protokola se obezbeđuje uspešna interakcija između udaljenih procesa [1].

OSI (eng. Open Systems Interconnection Basic Reference Model) model je model koji deli arhitekturu mreže u sedam logičkih nivoa (Tabela 3.1), daje spisak funkcija, servisa i protokola koji funkcionišu na svakom od nivoa. Jedan od najvećih razloga zašto je uveden referentni model je da se izvrši standardizacija samih protokola.

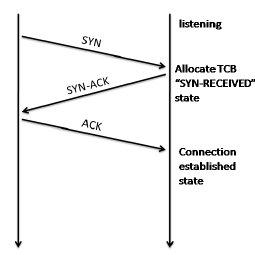


*Tabela 3.1 ISO/OSI referentni model*

## TCP

TCP (eng. Transfer Control Protocol) je jedan od glavnih protokola za razmjenu podataka na internetu. Nastao je u početnoj implementaciji mreže u kojoj je dopunio internet protokol (IP). S toga se cijeli paket obično naziva TCP/IP. TCP pripada četvrtom nivou OSI referentnog modela. Pruža pouzdanu i uređenu isporuku toka okteta (bajtova) između aplikacija koje se izvode na hostovima i komuniciraju putem IP mreže [2].

TCP je connection-oriented protokol, zahtjeva uspostavljanje veze između klijenta i servera prije slanja podataka. Veza se uspostvlja tako što klijent i server urade „rukovanje“, odnosno razmjene tri poruke sa podešenim odgovarajućim kontrolnim bitima (Slika 3.1.1).



*Slika 3.1.1 Uspostavljanje veze koristeći TCP protokol*

## HTTP

HTTP (eng. Hyper Text Transfer Protocol) je aplikativni protokol, namjenjen za razmjenu HTML (eng. Hyper Text Markup Language) dokumenata putem Web aplikacija. Na transportnom nivou koristi TCP (eng. Transmission Control Protocol) protokol. Funkcioniše kao zahtjev-odgovor protokol u klijent-server arhitekturi. Klijent šalje HTTP zahtjev serveru. Server, koji sadrži HTML datoteke i drugi sadržaj, klijentu vraća odgovor. Odgovor sadrži informacije o statusu dovršenosti zahtjeva i takođe može sadržati traženi sadržaj u svom tijelu poruke [3].

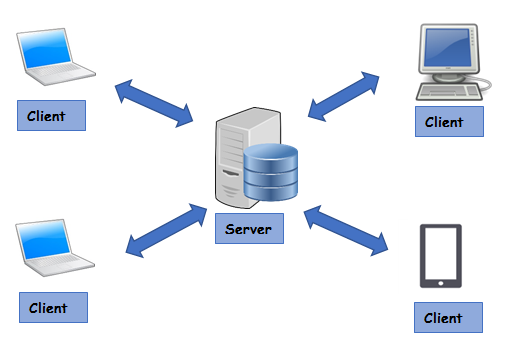
Protokol definiše metode koje ukazuju na radnju koju treba izvršiti na indetifikovanom resursu. Najznačajnije metode su [3]:

* GET – zahtjeva da se klijentu dostavi navedeni resurs. Zahtjevi koji koriste GET treba da samo preuzimaju podatke i ne smiju imati drugog efekta.
* POST – zahtjeva da Web server prihvati podatke koji se nalaze u tijelu zahtjeva, najčešće za njihovo skladištenje i čuvanje. Često se koristi pri otpremanju datoteka ili pri slanju popunjenog Web obrasca.
* PUT – koristi se za ažuriranje postojećih resursa na serveru. Ako željeni resurs ne postoji na serveru, kreira ga.
* DELETE – zahtjeva brisanje navedenog resursa sa servera.

# Web aplikacije

## Klijent-server arhitektura

Klijent-server arhitektura je računarski model u kojem server hostuje, isporučuje i upravlja većinom resursa i usluga koje klijent koristi. Ova vrsta arhitekture ima jedan ili više klijentskih računara povezanih sa centralnim serverom preko mreže (Slika 4.1.1) [6].



*Slika 4.1.1 Klijent-server arhitektura*

Klijentski računari pružaju interfejs koji omogućava korisniku računara da zatraži usluge servera i da prikaže rezultate koje server vraća. Serveri čekaju zahtjeve od klijenata, a zatim odgovaraju na njih. U idealnom slučaju, server klijentima nudi standardizovani, transparentni interfejs tako da klijenti ne moraju biti svjesni specifičnosti sistema koji pruža usluge [6].

Primjena klijent-server arhitekture u organizaciji će povećati produktivnost korišćenjem isplativih korisničkih interfejsa, poboljšanog skladištenja podataka, velike povezanosti i pouzdanosti usluga aplikacija [6].

Neke od mana ove arhitekture jesu [6]:

* Kada su česti istovremeni zahtjevi klijenata, serveri se mogu ozbiljno preopteretiti, stvarajući zagušenje prometa i ogromno kašnjenje ili potpuni nedostatak odgovara klijentu,
* Budući da je arhitektura centralizovana, ako kritični server otkaže, zahtjevi klijenata se ne ispunjavaju.

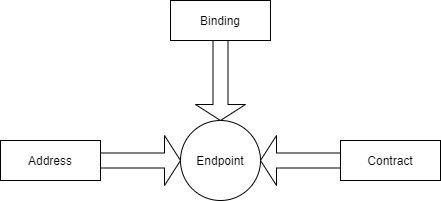
# Windows Communication foundation (WCF)

WCF je framework za izgradnju aplikacija orijentisanih na servise. Korišćenjem WCF-a, podaci se mogu slati kao asinhrone poruke sa krajnje tačke jednog servisa na drugi. Poruke mogu biti jednostavne, kao jedan znak ili riječ poslati kao XML, ili kompleksne, kao tok binarnih podataka [8].

Zasniva se na pojmu komunikacije zasnovane na porukama, a sve što se može modelovati kao poruka (na primer, HTTP zahtev ili poruka u redu čekanja) može biti predstavljeno na jedinstven način u programskom modelu. Ovo omogućava jedinstveni API u različitim transportnim mehanizmima [8].

Model razlikuje klijente koji su aplikacije koje započinju komunikaciju i servise koji su aplikacije koje čekaju da klijenti komuniciraju s njima i odgovore na njihove zahtjeve. Jedna aplikacija može djelovati i kao klijent i kao servis [8].

Poruke se šalju između krajnjih tačaka (eng. endpoint). Krajnje tačke su mesta na kojima se poruke šalju ili primaju (ili oboje) i definišu sve informacije potrebne za razmenu poruka. Servis izlaže jednu ili više krajnjih tačaka aplikacije, a klijent generiše tačku koja je kompatibilna sa jednom od krajnjih tačaka servisa. Krajnja tačka na standardni način opisuje gdje poruke treba slati, kako ih treba slati i kako poruke treba da izgledaju [8].



*Slika 5.1 Krajnja tačka*

Krajnja tačka se sastoji od lokacije (adrese) koja definiše gdje se poruka može poslati, komunikacionog mehanizma (binding) koji difiniše kako se poruke šalju i denificije za set poruka (contract) koje se mogu poslati ili primiti na datoj lokacije (Slika 5.1).

## Komunikacioni protokoli

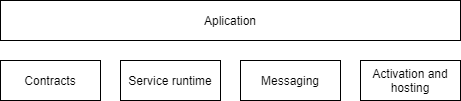
Jedan potreban element komunikacionog steka je transportni protokol. Poruke se mogu slati preko intraneta i Interneta koristeći uobičajene transportne protokole, kao što je TCP. Uključeni su i drugi protokoli koji podržavaju komunikaciju sa aplikacijama za redosled poruka i čvorovima na mreži ravnopravne mreže. Više ugrađenih transportnih mehanizama može se dodati pomoću ugrađenih produžnih tačaka WCF-a [8].

Još jedan potreban element u steku komunikacija je kodiranje koje specificira kako je data poruka formatirana. WCF pruža sledeća kodiranja [8]:

* Kodiranje teksta, interoperabilno kodiranje,
* Kodiranje mehanizma za optimizaciju prenosa poruka,
* Binarno kodiranje za efikasan prenos.

## Arhitektura

Sledeća slika prikazuje slojeve arhitekture WCF-a (Slika 4.2.1).

****

*Slika 5.2.1 Arhitektura WCF-a*

Ugovori (eng. Contracts) definišu različite aspekte sistema poruka. Ugovor o podacima opisuje svaki parametar koji čini svaku poruku koju usluga može stvoriti ili potrošiti. Parametri poruke su definisani XML Schema definition jezikom (XSD), omogućavajući bilo kojem sistemu koji razumije XML da obrađuje dokumente [9].

Nivo izvršavanja usluge (eng. Service runtime) definiše ponašanja koja se događaju samo tokom rada servisa. Opisuje prigušivanje, koje kontroliše koliko se poruka obrađuje, što se može promjeniti ako potražnja za servisom naraste do unaprijed postavljene granice [9].

Sloj za razmenu poruka (eng. Messaging) sastoji se od kanala. Kanal je komponenta koja obrađuje poruku na neki način, na primer, autentifikacija poruke. Kanali rade na porukama i zaglavljima poruka. Ovo se razlikuje od nivoa izvršavanja usluge, koji se prvenstveno brine za obradu sadržaja tela poruka. Postoje dve vrste kanala: transportni i protokolski [9].

Sloj za aktivaciju i hostovanje (eng. Activation and hosting) definiše kako i gdje krajnji program može biti pokrenut (samostalna izvršna datoteka ili hostovanje na drugoj aplikaciji).

# Opis korišćenih tehnologija i alata

* **C# -** programski jezik opšte namjene. Podržava imperativnu, objektno-orijentisnu, funkcionalnu i deklarativnu paradigmu. Spada u grupu novijih programskih jezika i razvijen je od strane kompanije Microsoft. Sintaksno veoma podseća na C i C++ jezike, ali je njegova sintaksa znatno jednostavnija. Takođe, u C# -u obezbeđeno je i automatsko rukovanje memorijom što ga čini sigurnijim za programiranje [10].
* **.NET –** besplatna, open-source platforma za razvoj mnogih vrsta aplikacija. Dijeli funkcionalnosti između različitih aplikacija i tipova aplikacija pomoću klasnih biblioteka [11].
* **ISS *(eng. Internet Information Services)*** – softverski paket dizajniran za Windows Server. Koristi se za hostovanje veb stranica i drugog sadržaja na internetu. Može da hostuje klasične HTML stranice i dinamičke veb stranice [12]..
* **Visual Studio** – integrisano razvojno okruženje (*eng*. *Integrated development environment IDE*) razvijeno od strane kompanije Microsoft. Posjeduje podrušku za 36 programskih jezika, među kojima je vodeći C#. Koristi Microsoft-ove alate za razvoj različitih vrsta aplikacija poput Windows Forms, Windows Presentation Foundation, Windows Communication Foundation, Microsoft Silverlight, WebApi i Azure-a namenjenog za Cloud okruženja [13].
* **Visual Studio Code** – minimalan ali veoma napredan editor programskog koda. Dolazi sa ugrađenom podrškom za JavaScript i TypeScript što ga čini savršenim okruženjem za razvoj Web aplikacija [14].
* **TypeScript** – programski jezik proizved od strane Microsoft-a koji proširuje JavaScriptstatičkom tipizaciom i konceptima objekto-orjentisanog programiranja [15].
* **Angular 9** – okruženje za razvoj klijentskih Single-Page Web aplikacija baziranih na TypeScript-u, razvijen od strane Google-a. U sebi sadrži i Karma okruženje za testiranje aplikacija [16].
* **REST** (***eng. REpresentational State Transfer***) – stil arhitekture softvera koji se koristi prilikom razvoja Web servisa. U osnovi stila je da sve što se nalazi na serveru posmatra kao resurs. Resursi se dobavljaju sa servera posredstvom osnovnih metoda HTTP protokola u JSON ili XML formatu [17].
* **HTML** (***eng. HyperText Markup Language***) – opisni jezik specijalno namjenjen opisu strukture Web stranica [18].
* **CSS** (***eng. Cascade Style Sheet***) – jezik koji opisuje kako će se HTMLelementi prikazivati na ekranu. Može kontrolisati izgleda više stranica od jednom [19].
* **Bootstrap 4** – besplatna biblioteka bazirana na HTML-u, CSS-u i JavaScript-u koja obezbeđuje kolekcije predefinisanih stilova, kontrola i načina organizacije elementa u HTML dokumentima. Takođe podržava i responsivni Web dizajn [20].

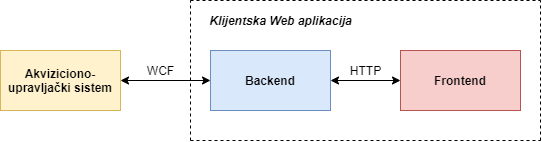
# Opis programskog rješenja

Kao što je spomenuto u prvom poglavlju ovoga rada, zadatak je razviti aplikaciju koja će omogućiti korisnicima udaljen pristup akviziciono upravljačkim sistemima, radi nadzora i upravljanja istim.

Aplikacija je nezavisna od akviziciono upravljačkog sistema sa kojim je povezana, odnosno može se smatrati potpuno različitom aplikacijom. Zasnovana je na klijent-server arhitekturi, gdje klijent i server komuniciraju korišćenjem HTTP protokola i REST principa. Veza sa akviziciono upravljačkim sistemom se ostvaruje korišćenjem WCF tehnologije.

Klijentska Web aplikacija za akviziono-upravljačke sisteme je podjeljena na dvije cjeline (Slika 7.1):

* Frontend – Web klijent,
* Backend – Web server.



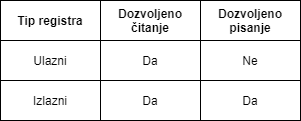
*Slika 7.1 Arhitektura rješenja*

Aplikacija koristi podatke dobijene od strane akviziciono upravljačkog sistema. Podaci su u vidu registara koji predstavljaju realne, fizičke veličine unutar idustrijskih postrojenja povezanih sa akviziciono upravljačkim sistemom.

Svaki registar ima svoj tip koji opisuje vrijednost sadržanu u samom registru, kao i da li ta vrijednost može biti izmjenjena od strane korisnika. Podržani su pet tipova registra [1]:

* Analogni ulazi – nivo kontinualnog električnog signala koji je u svakom trenutku proporcionalan trenutnoj vrijednosti mjerene veličine (pritisak, masa, napon i sl.),
* Analogni izlazi – generisanjem kontinualnog pobudnog signala zadaje radnu tačku izvršnog elementa u postrojenju. Tipična primjena je kontrola pozicije regulacionog ventila, odnosno stepena njegove otvorenosti/zatvorenosti (0-100%),
* Digitalni ulazi – predstavljaju fizičke veličine koje su diskretne po svojoj prirodi i generički odgovaraju prekidaču u polju. Najčešće se preko njih prati stanje izvršnih elemenata u postrojenju (npr, sklopka, ventil),
* Digitalni izlazi – posredstvom njih se ostvaruje upravljanje tipa uključi/isključi ili otvori/zatvori, za izvršne elemente u postrojenju (npr, sklopke i ventili),
* Brojački ulazi – za njih je karakteristično da učestanost električnih impulsa koji se prihvataju predstavlja mjeru trenutne vrijednosti fizičke veličine (tipično protok), a njihova akumulirana vrijednost ukupnu količinu u nekom periodu.

U zavisnosti od toga da li je registar ulazni ili izlazni nad njim su dozvoljene funkcije čitanja i pisanja (Tabela 7.1).



*Tabela 7.1 Operacije nad registrima*

Aplikacija podržava i korištenje alarma. Alarmi upozoravaju korisnika da se vrijednost registra nalazi van njegovih predviđenih, nominalnih vrijednosti. Definisana su četiri tipa alarma:

* Visoka vrijednost – označava da je vrijednost registra iznad maksimalne, predviđene optimalne vrijednosti, ali u predviđenim granicama odstupanja,
* Niska vrijednost – označava da je vrijednost registra ispod minimalne, predviđene optimalne vrijednosti, ali u predviđenim granicama odstupanja,
* Abnormalna vrijednost – označava da se vrijednost registra nalazi van predviđenih granica odstupanja i da zadržavanje trenutne vrijednosti može dovesti do realnih problema u industrijskom postrojenju,
* Bez alarma – vrijednost registra se nalazi u optimalnom stanju.

U zavisnosti od implementacije akviziciono upravljačkog sistema, registri mogu biti podjeljeni u grupe. Svaka grupa registara je definisana tipom registra, početnom adresom, brojem registara i akvizicionim intervalom (Listing 7.1). Pretpostavlja se da se registri, definisani istom grupom, nalaze na sukcesivnim memorijskim lokacijama, od početne adrese (startAddress) nadalje.

Postojanje grupa nije nužno za funkcionisanje aplikacije, ali omogućava automatsku akviziciju za registre, unutar iste grupe, u istom vremenskom periodu, definisanom akvizicionim intervalom (acquisitionInterval).



*Listing 7.1 Definicija grupe registara*

Radi lakše komunikacije između komponenti aplikacije (frontend-a i backend-a) umjesto 5 različitih klasa, kojim bi bili opisani svaki tip registra posebno, koristi se jedna klasa, pod imenom RegisterData (Listing 7.2), koja sadrži sve neophodne informacije o registru, nebitno kojem tipu pripadao. Klasa u sebi sadrži sve informacije od značaja korisniku.

**

*Listing 7.2 RegisterData klasa*

## Frontend

Frontend je prezentacioni dio Web aplikacije, koji omogućava interakciju korisnika sa željenim resursima. Frontend koristi usluge servera posredstvom HTTP protokola, te je ujedno i Web klijent.

Osnovni cilj frontend-a jeste da vjerno i na jednostavan način prikaže podatke dobijene od strane akviziciono upravljačkog sistema, omogući slanje komandi za čitanje ili pisanje podataka i prikaže istorijat izmjena korisniku. Treba da bude intuitivan i da podatke predstavlja na sličan način kao i softver u dispečerskim centrima akviziciono upravljačkog sistema.

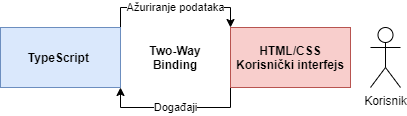
Za razvoj frontend-a je korišten Angular, tehnologija koja omogućava kreiranje jednostraničnih veb aplikacija koje se sastoje od više komponenti koje međusobno komuniciraju.

Komponente su podskup direktiva i one se nalaze u stablu komponenti aplikacije. Svaka komponenta je zadužena za prikaz određenog dijela stranice. Zamjenom jedne komponente drugom, mjenjamo izgled stranice bez ponovnog učitavanja svih podataka od strane pretraživača. Komponente u sebi mogu da sadrže druge komponente. Takve komponente se zovu roditeljske. Svaka Angular aplikacija sadrži u sebi jednu glavnu, roditeljsku komponentu, koja predstavlja čitav ekran. Ona je roditeljska komponenta za sve ostale [21].

Komponenta se sastoji od tri fajla koja opisuju njen izgled i funkcionisanje:

* HTML fajl – opisuje strukturu Web stranice,
* CSS fajl – opisuje stilski izgled stranice definisane HTML fajlom,
* TypeScript fajl – sadrži konstruktor za inicijalizaciju komponente i sve funkcionalnosti koje komponenta omogućava.

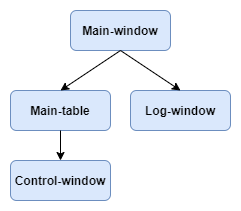
Podaci u TypeScript fajlu i prikaz iz HTML/CSS povezani su putem Two-Way Binding mehanizma (Slika 7.1.1) koji obezbjeđuje da se promjene nad podacima propagiraju na korisnički prikaz.



*Slika 7.1.1 Two-Way Binding*

Pored glavne komponente, frontend klijentske Web aplikacije za akviziciono upravljačke sisteme se sastoji od još četiri dodatne komponente (Slika 7.1.2):

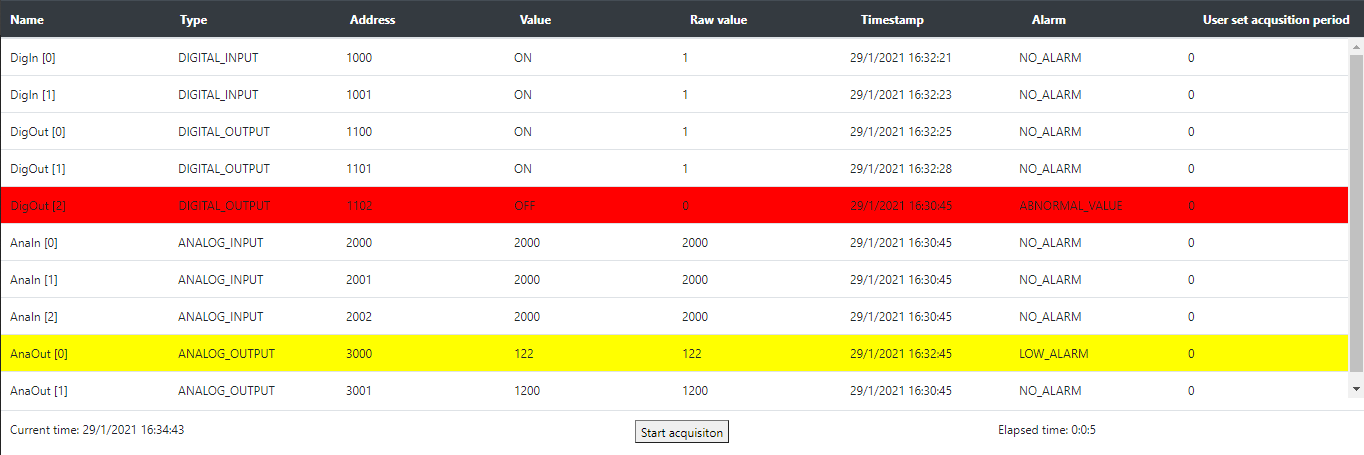
1. Main-window – za navigaciju izmedju Main-table i Log-window komponenti i njihova roditeljska komponenta,
2. Main-table – za prikaz podataka iz AUS-a i roditeljska za Control-window,
3. Log-window – za prikaz istorijata izmjena,
4. Control-window – omogućava slanje komandi za čitanje i pisanje AUS-u.



*Slika 7.1.2 Komponente frontend-a*

Main-table komponenta u sebi sadrži tabelu u kojoj svaki red predstavlja jedan registar iz akviziciono upravljačkog sistema. Informacije koje sadrži tabela su naziv registra, tip registra, njegova adresa, vrijednost izražena u inženjerskim jedinicima, sirova vrijednost, datum zadnje izmjene, vrsta alarma i period akvizicije koju korisnik podešava (Slika 7.1.3).

Pored tabele, komponenta sadrži i informaciju o trenutnom vremenu, koliko vremena je prošlo od pokretanja stranice i dugme „Start acquisition“, pritiskom na koje se pokreće automatska akvizicija vrijednosti registara.

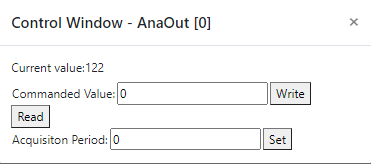
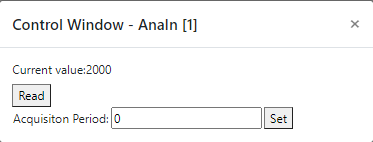


*Slika 7.1.3 Main-table komponenta*

U zavisnosti od vrste alarma, pozadina reda mijenja boju. Ako je registar u režimu visoke ili niske vrijednosti pozadina će biti žute boje, dok je za abnormalnu vrijednost, pozadina crvene boje. U svim ostalim slučajevima pozadina je bijele boje. Klikom na neki od registara, pozadina postaje osjenčena, odnosno siva, kako bi se korisniku naznačilo koji je registar pritisnuo.

U koliko korisnik želi da izdaje komande, koje će se izvršiti nad registrom, duplim klikom na neki od registara otvara se pop-up, odnosno Control-window komponenta. Komande koje su dozvoljene korisniku su definisane na osnovu tipa registra kojim se rukuje (Slika 7.1.5):

* Za izlazne registe – dozvoljene su komande promjena vrijednost, čitanje vrijednosti i podešavanje perioda akvizicije,
* Za ulazne registe – dozvoljene su komande čitanje vrijednosti i podešavanje perioda akvizicije.

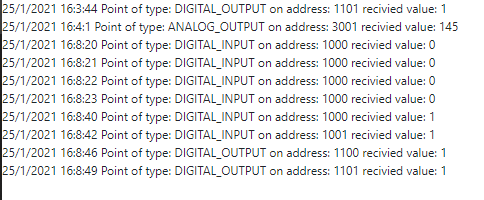
*Slika 7.1.5 Control-window komponenta*

Pritiskom na dugme „Read“ fronted formira zahtjev za čitanje, koji posredstvom HTTP protokola proslijeđuje dalje backend-u na obradu. Na osnovu dobijenog odgovora od backend-a, frontend, ako je zahtjev uspješno obavljen, mijenja vrijednost registra na dobijenu ili, ako zahtjev nije uspješno obavljen, obavještava korisnika o grešci (Slika 7.1.5).

Isto se dešava pritiskom na dugme „Write“, sa tim da se u HTTP zahtjev, koji se šalje backend-u, dodaje i vrijednost definisana poljem „CommandedValue“ (Slika 7.1.5).

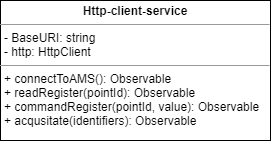
Pritiskom na dugme „Set“ se podešava akvizicija registra na osnovu polja „Acquisiton Period“. Vrijednost 0 znači da akvizija neće biti rađena, dok se za sve ostale vrijednosti, veće od 0, na zadati broj sekundi formira zahtjev za čitanje koji se dalje proslijeđuje backend-u (Slika 7.1.5).

Za vođenje istorijata izmjena vrijednosti registara zadužena je komponenta Log-window, koja za svaki uspješno ispunjen zahtjev formiran od strane Main-table i Control-window komponenti, vrši upis promjene vrijednosti registra. Informacije koje se upisuju jesu datum i vrijeme kada je izvršen zahtjev, tip, adresa i nova vrijednost registra nad kojim je izvršen željeni zahtjev (slika 7.1.6).



*Slika 7.1.6 Log-window komponenta*

Pored komponenti, frontend sadrži i servis za komunikaciju sa backend-om, pod imenom http-client-service (Slika 7.1.7). Servis je opisan TypeScript programskim jezikom i označen je Injectable dekoratorom, što obezbjeđuje da ga Angular okruženje može injektovati u bilo koju komponentu koja ima potrebu za njim putem DependancyInjection mehanizma.



*Slika 7.1.7 Http-client-service*

U servisu su definisana sledeća polja polja:

* BaseURI – predefinisana adresa backend-a, ondosno Web servera,
* http – objekat klase HttpClient koji nudi mogućnost slanja zahtjeva ka serveru

Definisane metode servisa su:

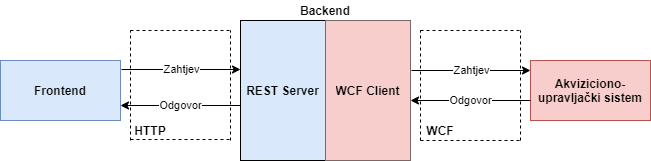
* connectToAMS – omogućava povezivanje klijenta sa akviziciono upravljačkim sistemom, poziva se prilikom pokretanja frontend-a na pretraživaču,
* readRegister – metoda za čitanje najsvježije vrijednosti željenog registra,
* commandRegister – metoda za komandovanje, odnosno promjenu vrijednosti, registra,
* acqusitate – omogućava akviziciju grupe registara definisanih registarskom grupom (Listing 6.1).

Sve metode vraćaju objekat klase Observable, koji omogućava asinhrono čekanje odgovora od servera, odnosno backend-a. Nakon prijema odgovora, isti se obrađuje i vrše se ažuriranje vrijednosti tabele u Main-table komponenti i obavještava se korisnik.

## Backend

Cilj backend-a je da komunikaciju između korisnika i AUS-a učini vremenski učinkovitom, odnosno, da korisnik stekne utisak da se sve promjene i zahtjevi dešavaju trenutno. Njegov zadatak jeste da prima zahtjeve od korisnika sa frontend-a, odradi provjeru validnosti zahtjeva, dalje ga proslijedi AUS-u, sačeka odgovor i proslijedi ga nazad klijentu i održavanje ažurirane liste registara radi utvrđivanja validnosti zahtjeva sa frontend-a.

Backend je odrađen u .NET Core-u, korišćenjem REST tehnologije za komunikaciju sa klijentom, u svojstvu REST servera, i WCF tehnologije za komunikaciju sa AUS-om, u svojstvu WCF klijenta (Slika 7.2.1).



*Slika 7.2.1 Arhitektura backend-a*

U komunikaciji sa akviziciono upravljačkim sistemom, backend koristi WCF interfejs pod imenom IWCFContract u kojem je definisan minimalan broj metoda koje omogućavaju razmjenu svih potrebnih informacija (Listing 7.2.1):

* GetRegisters – metoda od akviziciono upravljačkog sistema traži listu svih registara koji se nalaze u industrijskim postrojenjima sa kojima isti komunicira.
* GetRegisterGroups - metoda vraća informacije o grupama registara definisanim klasom RegisterGroup (Listing 7.1).
* ReadCommand – akviziciono upravljačkom sistemu se šalje zahtjev za čitanje jednog registra koji je identifikovan parametrom pointId,
* WriteCommand – metoda koja akviziciono upravljačkom sistemu šalje zahtjev za izmjenu vrijednosti registra idenfikovanog sa parametrom pointId. Nova vrijednost registra je definisana parametrom value.
* DoAcquisition – šalje zahtjev za akviziciju grupe registara. Kao parametar se šalje lista objekata klase RegisterGroup. „DoAcquisition“ se može koristiti samo ako metoda GetConfigItems vrati listu koja sadrži jedan ili više objekata u sebi. U suprotnom, nema informacije o grupama sličnih registara te se validan zahtjev ne može poslati.



*Listing 7.2.1 WCF interfejs*

Akviziciono-upravljački sistemi implementiraju serversku stranu WCF interfejsa i opslužuju WCF klijenta, backend web aplikacije. Interfejs se nalazi u dll fajlu sa svim klasama i enumeracija potrebnim za normalno funkcionisanje klijent-server komunikacije i ne zahtjeva reference na dodatne biblioteke.

Za prihvatanje i obrađivanje zahtjeva od klijenta, odnosno frontend-a, zadužen je REST API kontroler pod imenom RegisterController (Listing 7.2.2).

RegisterController definiše 3 polja neophodna za optimalno funkcionisanje:

* Client – objekat klase WCFClient, zadužen za otvaranje konekcije ka akviziciono upravljačkom sistemu. Preko objekta ove klase se pozivaju sve metode definisane WCF interfejsom IWCFContract (Listing 7.2.1),
* Registers – rječnik u kojem se nalaze svi registri dobijeni metodom GetRegisters (Listing 7.2.1) gdje je ključ jedinstveni identifikator registra,
* RegisterGroups – rječnik u kojem se nalaze sve grupe registara dobijene metodom GetRegisterGroups (Listing 7.2.1) gdje je ključ jedinstveni identifikator grupe.



*Listing 7.2.2 RegisterController*

Kada klijent želi da dobije informacije od backenda, šalje zahtjev na jednu od 4 metode kontrolera, zajedno sa zahtjevanim parametrima:

* Connect – prima zahtjev za povezivanje sa akviziciono upravljačkim sistemom. Po prijemu zahtjeva vrši se provjera da li je konekcija ka željenom akviziciono upravljačkom sistemu otvorena, ako nije, kreira se novi objekat klase WCFClient koji otvara konekciju. Ako je konekcija već otvorena ili nakon uspješno uspostavljene konekcije, backend od akviziciono upravljačkog sistema traži podatke o postojećim registrima i grupe registara, koje potom smješta u svoja privremena skladišta memorije i proslijeđuje kao odgovor klijentu,
* ReadSingleRegister – metoda kao parametar prima jedinstveni identifikator registra, vrši provjeru validnosti zahtjeva, odnosno, provjerava da li se registar sa datim identifikatorom nalazi u privremenoj memoriji. Ako se nalazi, šalje se zahtjev akviziciono upravljačkom sistemu za čitanjem nove vrijednosti tog registra. Odgovor na zahtjev se proslijeđuje klijentu i nove informacije o registru se pamte u memoriju.
* CommandSingleRegister – prima zahtjev za promjenu vrijednosti registra. Kao parametre prima identifikator registra i vrijednot koju treba upisati. Vrši se provjera validnosti zahtjeva, ista kao kod ReadSingleRegister metode. Ako je zahtjev validan, proslijeđuje se akviziciono upravljačkom sistemu i čeka se na potvrdu uspješnosti zahtjeva.
* DoTheAcqusition – prima zahtjev za akviziciju grupe registara. Svaka grupa, čije vrijednosti registara treba isčitati, je definisana jednim brojem proslijeđenim u listi identifiers. Vrši se provjera validnosti zahtjeva, provjerom da li se grupe definisane datim brojem nalaze u promjenljivoj RegisterGroups. Ako je zahtjev validan, on se proslijeđuje dalje akviziciono upravljačkom sistemu.

## Oporavak od grešaka

Budući da se sva komunikacija odvija preko internet mreže i da se radi o sistemima kritične infrastukture gdje je detektovanje i oporavak od grešaka neophodno, u implementaciju rješenja su uključene i situacije koje dovode do grešaka i način za oporavak od istih.

Definisane su četiri tipične situacije u kojima dolazi do greške:

* Gubitak internet konekcije – ukoliko dođe do prekida veze između frontend-a i backend-a ili web aplikacije i akviziciono upravljačkog sistema pretpostavlja se da je došlo do problema sa internet mrežom te se automatski pokreće mehanizam za ponovno uspostvaljanje veze, koji inicijalizuje frontend slanjem zahtjeva za povezivanje sa akviziciono upravljačkim sistemom. Zahtjev se šalje sve dok se ponovna konekcija ne uspostavi,
* Korupcija WCF komunikacionog kanala između Web aplikacije i akviziciono upravljačkog sistema – do ove greške dolazi ako je komunikacioni kanal prezatrpan, ako se desila greška na WCF serveru ili prekidom internet mreže kojeg Web aplikacija nije svjesna. Oporavak od ove greške podrazumjeva slanje zhatjeva akviziciono upravljačkom sistemu za uspostavljanje novog kanala sve dok isti ne bude uspostavljen,
* Nevalidni zahtjevi – namjerne ili nenamjerne greške koje učini korisnik prilikom slanja zahtjeva sa frontend-a. Korisnik greške može da učini iz korisničkog interfejsa ili pomoću alata koji mu omogućavaju da šalje zahtjeve koji nisu dozvoljeni interfejsom. Definisana su tri tipa ovih grešaka:
  + Nepostojeći identifikator registra tokom čitanja ili komandovanja istog – pretpostavlja se da je došlo do korupcije podataka na frontend-u, korisnik se obavještava o nastaloj grešci i pokreće se mehanizam za ponovno preuzimanje podataka sa servera,
  + Korisnik pokuša da komanduje registar koji nije odgovarajućeg tipa – ova greška nastaje kada korisnik pokuša da komanduje ulaznim registrima nad kojima je omogućena samo operacija čitanja. Kao i u prvom slučaju, pretpostavlja se da je došlo do korupcije podataka na frontendu te se pokreće isti mehanizam za oporavak,
  + Nepostojeći identifikator grupe registara tokom akvizicije – isto kao u prvom slučaju.
* Uspješno poslat zahtjev bez odgovora akviziciono upravljačkog sistema – pretpostavlja se da je došlo do greške na strani akviziciono upravljačkog sistema i da je sa njegove strane pokrenut oporavak od greške koji će vratiti sistem u normalno stanje. Ipak, ukoliko dođe do iste greške 5 puta za redom pokreće se mehanizam za ponovno uspostavljanje konekcije sa akviziciono upravljačkim sistemom od strane backend-a.

Greške tipa nevalidni zahtjevi su najopasnije po normalno funkcionisanje akviziciono upravljačkog sistema, jer njih najčešće stvaraju zlonamjerni pojedinci sa ciljem rušenja cjelokupnog sistema ili dobijanja povjerljivih informacija, s toga je ovim greškama dato najviše pažnje

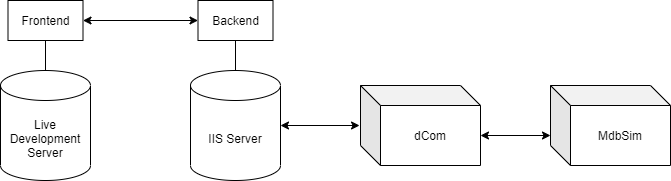
# Testiranje rješenja

Testiranje aplikacije je obavljeno korištenjem dCom aplikacije, školski akviziciono upravljački sistem koji je namenjen za prvi i najdirektniji kontakt studenata sa SCADA funkcionalnostima, i MdbSim aplikacije, simulator jednog ili više procesnih kontrolera instaliranih u hipotetičkom industrijskom postrojenju, koji sa centralnim SCADA serverom (u ovom slučaju sa dCom-om) komunicira posredstvom Modbus protokola.

Prilikom pokretanja, dCom uspostavlja konekciju sa MdbSim posredstvom TCP implementacije Modbus protokola i pokreće instancu WCF servera koja implementira interfejs IWCFContract (Listing 5.2.1). WCF server je implementiran sa minimalnim izmjenama originalnog koda dCom aplikacije.

Backend Web aplikacije se izvršava na IIS (eng. Internet Information Services) serveru. Pored izvršavanja backend-a, IIS se mogu zadati i dodatni zadaci kao što su autentifikacija, autorizacija.

Frontend se izvršava na Live Development Serveru, koncept NodeJS okruženja za razvoj i izvršavanje JavaScript aplikacija.



*Slika 8.1 Fizička organizacija sistema*

# Zaključak

Dobre karakteristike Web aplikacije se ogledaju u adekvatnom rukovanju izuzecima, kao uslovu za pouzdan rad. Tokom razvoja sprežne komponente prepoznati su najčešće mogući problemi i samim tim neće biti naglih ispada iz programa. Korisnički interfejs, iako jednostavan, je potpuno intuitivan. Algoritam je brz i pouzdan, jedino od čega zavisi brzina rada aplikacije jeste od brzine protoka informacija na internetu i načina implementacije WCF Servera u akviziciono upravljačkom sistemu.

Takođe, dobra strana aplikacije je razdvojenost od samog akviziciono upravljačkog sistema sa kojim obavlja komunikaciju. Funkcionisanje aplikacije nije zavisno od funkionisanje akviziciono upravljačkog sistma, što znači da otkaz u AUS-u ne dovodi do pada weba aplikacije i obrnuto.

Međutim, postoji određen broj ograničenja razvijene aplikacije. IP adresa AUS-a je programski zadata te bi eventualne izmjene adrese zahtjevale da se ulazi u kod web aplikacije i iz njega adresa mijenja. Takođe, budući da se sva komunikacija odvija preko interneta, rad aplikacije u mnogome zavisi od brzine protoka interneta i podložna je kašnjenima i prekidom rada uslijed zagušenja komunikacionih kanala ili nedostaka internet usluge.

Ne postoji nikakva vrsta enkripcije podataka koji se razmjenjuju te je komunikacija između web aplikacije i AUS-a podložna napadima, kao što su presretanje i fabrikacija zahtjeva, što bi moglo štetno uticati na funkcionisanje cjelokupnog sistema. Nije vođeno računa ni o autentifikaciji i autorizaciji korisnika jer je aplikacija namjenjena u obrazovne svrhe i za rad sa školskim simulatorima.

Mogući pravac daljeg usavršavanja jeste nadogradnja web aplikacije tako da omogućava povezivanje na više WCF Servera, odnosno akviziciono-upravljačkih sistema. Da bi ovo postalo stvarnost, bilo bi neophodno da se omogući korisniku da unese IP adresu svakog pojedinačnog AUS-a ili da, iz predefinisane liste AUS-a, izabere željene, kao i da se napravi baza podataka koja bi adrese i dodatne informacije o sistemima pamtila.

Sledeći korak bi mogao biti uvođenje autentifikacije i autorizacije, gdje bi se mogle podjeliti uloge među korisnicima, svaka uloga sa posebnim ovlašćenjima i mogućnostima pristupa. Bilo bi neophodno uvesti neku vrstu tokena, odnosno kukija, prilikom komunikacije, radi raspoznavanja korisnika na backend-u. Srećom, postoje brojne biblioteke na internetu koje omogućavaju laku, brzu i sigurnu autentifikaciju i autorizaciju.

Treba povesti računa i o enkripciji samih podataka koji se razmjenjuju između aplikacije i AUS-a. Postavlja se pitanje šta i na koji način zaštiti tako da brzina komunikacije između web aplikacije i AUS-a bude na zadovoljavajućem nivou, odnosno, da se ne izgubi privid da se svi zahtjevi izvršavaju „momentalno“.

# Literatura

[1] Atlagić, Branislav. Softver sa kritičnim odzivom - Projektovanje SCADA sistema. Novi Sad : Fakultet tehničkih nauka u Novom Sadu, 2015.

[2] [https://tools.ietf.org/html/rfc793](https://tools.ietf.org/html/rfc793%20), **TCP**

[3] <https://tools.ietf.org/html/rfc2616>, **HTTP**

[4] <https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>, **Web aplikacije**

[5] <https://blog.stackpath.com/web-application/>, **Prednosti Web aplikacija**

[6] <https://www.techopedia.com/definition/438/clientserver-architecture>, **Klijent-server arhitektura**

[7] Nancy J. Yeager, Robert E. McGrath – Web Server Technology, 1996.

[8] <https://docs.microsoft.com/en-us/dotnet/framework/wcf/fundamental-concepts>

[9] <https://docs.microsoft.com/en-us/dotnet/framework/wcf/architecture>

[10] <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>, **C#**

[11] [https://docs.microsoft.com/en-us/dotnet/core/introduction#support](https://docs.microsoft.com/en-us/dotnet/core/introduction%23support), **.NET**

[12] <https://techterms.com/definition/iis>, **ISS**

[13] <https://visualstudio.microsoft.com/vs/>, **Visual Studio**

[14] <https://code.visualstudio.com/docs>, **Visual Studio Code**

[15] <https://www.typescriptlang.org/>, **TypeScript**

[16] <https://angular.io/guide/architecture>, **Angular**

[17] <https://www.codecademy.com/articles/what-is-rest>, **REST**

[18] <https://www.w3schools.com/html/html_intro.asp>, **HTML**

[19] <https://www.w3schools.com/css/css_intro.asp>, **CSS**

[20] <https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp>, **Bootstrap**

[21] [https://angular.io/api/core/Component#description](https://angular.io/api/core/Component%23description), **Komponenta**