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

Mihajlo Džever

**RAZVOJ KLIJENTSKE WEB APLIKACIJE ZA AKVIZICIONO UPRAVLJAČKE SISTEME**

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**: | | | Михајло Џевер | |
| 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**: | | | СКАДА, Aквизиција, Командовање, Историјски подаци, Комуникација, Клијент, Сервер, ВЦФ | |
| **УДК** | | |  | |
| Чува се, **ЧУ**: | | | Библиотека ФТН, Д. Обрадовића 6, 21000 Нови Сад | |
| Важна напомена, **ВН**: | | |  | |
| Извод, **ИЗ**: | | |  | |
| Датум прихватања теме, **ДП**: | | |  | |
| Датум одбране, **ДО**: | | |  | |
| Чланови комисије, **КО**: | | Председник: |  |
|  | | Члан: |  | 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**: | | | Mihajlo Džever | |
| Mentor, **MN**: | | | dr Branislav Atlagić, assistant professor | |
| Title, **TI**: | | | Client Web application development for acquisition management systems | |
| 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**: | | | SCADA, Acquisition, Command, History data, Communication, Client, Server, WCF | |
| **UC** | | |  | |
| Holding data, **HD**: | | | Library of Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad | |
| Note, **N**: | | |  | |
| Abstract, **AB**: | | |  | |
| 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. Драган Поповић |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | Михајло Џевер | Број индекса: | **ПР 49/2016** |
| Област: | Софтвер са критичним одзивом | | |
| 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:**

|  |
| --- |
| Развој клијентске Веб апликације за аквизиционо управљачке системе |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
|  |

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

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

Образац **Q2.НА.15-04** - Издање 1

**SPISAK SKRAĆENICA**

|  |  |
| --- | --- |
| ***API*** | ***Application Programming Interface*** |
| ***AUS*** | ***Akviziciono upravljački sistem*** |
| ***LAN*** | ***Local Area Network*** |
| ***PLC*** | ***Programmable Logic Controller*** |
| ***RTU*** | ***Remote Telemetry Unit*** |
| ***SCADA*** | ***Supervisory Control And Data Acquisition*** |
| ***WEB*** | ***World Wide Web*** |
| ***TCP*** | ***Transmission Control Protocol*** |
| ***HTTP*** | ***HyperText Transfer Protocol*** |
| ***IIS*** | ***Internet Information Services*** |
| ***HTML*** | ***HyperText Markup Language*** |
| ***IP*** | ***Internet Protocol*** |
| ***REST*** | ***RRpresentational State transfer*** |
| ***JSON*** | ***JavaScript Object Notation*** |
| ***XML*** | ***Extensible Markup Language*** |
| ***WAN*** | ***Wide Area Network*** |
| ***CSS*** | ***Cascade Style Sheet*** |
| ***WCF*** | ***Windows Communication Foundation*** |

**SPISAK KORIŠĆENIH SLIKA**

|  |  |
| --- | --- |
| ***Slika 1.1*** | ***Prikaz rješavanog problema*** |
| ***Slika 2.1*** | ***Uprošćen prikaz SCADA arhitekture*** |
| ***Slika 3.1.1*** | ***Klijent-server arhitektura*** |
| ***Slika 5.1*** | ***Arhitektura rješenja*** |
| ***Slika 5.1.1*** | ***Komponente frontend-a*** |
| ***Slika 5.1.2*** | ***Main-table komponenta*** |
| ***Slika 5.1.3*** | ***Control-window komponenta*** |
| ***Slika 5.1.4*** | ***Log-window komponenta*** |
| ***Slika 5.1.5*** | ***Http-client-service*** |
| ***Slika 5.2.1*** | ***Arhitektura backend-a*** |
| ***Slika 6.1*** | ***Fizička organizacija sistema*** |

**SPISAK LISTINGA**

|  |  |
| --- | --- |
| ***Listing 5.1*** | ***Definicija grupe registara*** |
| ***Listing 5.2.1*** | ***WCF interfejs*** |
| ***Listing 5.2.2*** | ***RegisterController*** |

Sadržaj

[1. Uvod 8](#_Toc64561278)

[2. Akviziciono upravljački sistemi 9](#_Toc64561279)

[2.1 SCADA 9](#_Toc64561280)

[2.2 Integracija na Web-u 10](#_Toc64561281)

[3. Web aplikacije 11](#_Toc64561282)

[3.1 Klijent-server arhitektura 11](#_Toc64561283)

[3.2 Web server 12](#_Toc64561284)

[3.3 Web klijent 12](#_Toc64561285)

[3.4 HTTP 12](#_Toc64561286)

[4. Opis korišćenih tehnologija i alata 13](#_Toc64561287)

[5. Opis programskog rješenja 14](#_Toc64561288)

[5.1 Frontend 15](#_Toc64561289)

[5.2 Backend 18](#_Toc64561290)

[5.3 Oporavak od greški 20](#_Toc64561291)

[6. Testiranje rješenja 21](#_Toc64561292)

[7. Zaključak 22](#_Toc64561293)

[8. Literatura 23](#_Toc64561294)

# Uvod

Web je postao standard za udaljen pristup podacima i razmjenu informacija koristeći klijent-server arhitekturu. Razvoj Web tehnologija i standardizacija njihovih biblioteka omogućava jednostavan način za kreiranje i održavanje Web aplikacija. Sve ovo je uslovilo tradicionalne industrijske i SCADA sisteme da implementiraju Web tehnologije i da omoguće svojim korisnicima udaljen pristup podacima, razmjenu informacija i kontrolu nad procesima.

Međutim, prelaz na Web nije jednostavan, prije svega, zbog kompleksnosti SCADA sistema i njihove orijentisanosti ka radu u realnom vremenu, koje predviđa minimalna kašnjenja u kontinuiranoj komunikaciji podelemenata. S toga, nepredviđena kašnjenja na internetu predstavljaju nezaobilazan problem pri razvoju SCADA Web aplikacija. Aplikativni protokol za razmjenu podataka na Web-u jeste HTTP, te bi prelaz SCADA aplikacija na Web podrazumjevalo prilagođavanje struktura podataka HTTP protokolu. Sigurnost podataka je dodatni problem kojim se moraju baviti SCADA Web aplikacije. Svi ovi problemi su uticali da danas, na tržištu, postoji mali broj rješenja ovog tipa.

Akviziciono-upravljački sistem (AUS) je skup namjenskih, prostorno distribuiranih, međusobno povezanih računarskih modula, čiji je zajednički cilj ostvarivanje funkcija automatskog nadzora i/ili upravljanja fizičkim procesom u realnom vremenu. Strukturu AUS sistema odlikuje prostrona distribucija autonomnih računarskih komponenti, koje su komunikacionom mrežom spregnute u jednistven sistem. Razmjenom poruka između njih ostvaruje se međusobna kooperacija u ostvarenju zajedničkog cilja. Korisnici sve informacije o fizičkim procesima u jednom AUS-u dobijaju preko korisničkih interfejsa instaliranih na računarima u dispečerskim centrima (kontrolnim sobama). [1]

Zadatak je stvaranje web platforme koja bi omogućila korisniku udaljen pristup AUS-u, odnosno povezivanje preko interneta sa dispečerskim centrima, uz obezbjeđivanje adekvatnih performansi aplikacije tako da korisnik stekne utisak da se radi o gotovo trenutnoj reakciji na promjene u realnom vremenu. Web aplikacija treba imati iste funkcionalnosti koje su korisnicima ponuđene u dispečerskim centrima, očitavanje SCADA promjenljivih, upravljanje postrojenjem i sticanje uvida u istorijske podatke.



Slika 1.1 Prikaz rješavanog problema

# Akviziciono upravljački sistemi

Tradicionalna osnova industrijskih akviziciono-upravljačkih sistema su PLC (eng. Programmable Logic Controller) kontroleri, koji samostalno upravljaju delom postrojenja. Kako bi upravljanje celim postrojenjem bilo moguće njihov rad mora biti sinhronizovan [1].

Vremenom se javlja potreba za upravljanjem uređajima koji se nalaze na različitim fizički udaljenim lokacijama. Za tu svrhu se koriste različiti komunikacioni protokoli kao sredstva komunikacije između udaljenih uređaja. U sistemima gde je potrebno precizno upravljanje, a koje može biti narušeno usled komunikacionih problema ili mernih grešaka, najčešće se glavne odluke u vezi upravljanja prepuštaju čoveku [1].

Razlikujemo dvije grupe akviziciono upravljačkih sistema, na osnovu geografske topologije i algoritama upravljanja koji se primjenjuju.

Prvu grupu AUS sistema čine sistemi orijentisani ka upravljanju prostorno ograničenim industrijskim postrojenjima. Ova klasa odlikuje se manjom dislokacijom pojedinih elemenata sistema, pouzdanijim odvijanjem komunikacije i visokim stepenom automatizacije upravljačkih aktivnosti. Upravljački algoritam koji se ugrađuje u ove sisteme je vrlo složen i uslovljen radom u realnom vremenu. Određenim dijelovima proizvodnog postrojenja upravljaju PLC kontroleri, koji i predstavljaju osnovu ovih sistema [1].

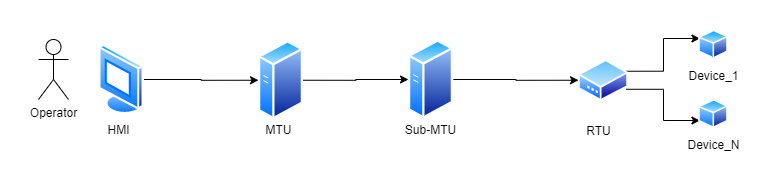
Drugu grupu AUS sistema čine sistemi orijenitisani ka upravljanju geografski dislociranim sistemima, kakvi su, na primjer, gasovodi, naftovodi, elektroenergetski sistemi i slično. Složenost tehnološkog procesa koji se nadgleda je visoka, pa su zadaci automatskog vođenja procesa prilično ograničeni. Na to dodatno utiču i problemi u komunikaciji, kao i greške u mjerenju, pa se kod ovih sistema akcenat stavlja na kvalitetan nadzor tehnološkog procesa, uz formiranje sveobuhvatnog istorijata događaja u sistemu. Kod ovih sistema, odluke o promjenama radnog režima uglavnom donosi operater u dispečerskoj radnoj stanici [1]

Rad ovakvih sistema odvija se pod kontrolom složenih programskih paketa pod nazivom SCADA (eng. Supervisory Control And Data Acquisition) [1].

## SCADA

SCADApredstavlja sistem za nadzor i upravljanje realnim sistemima. Spada u klasu računarskih sistema za rad u realnom vremenu – sistemi koji treba da obezbede adekvatan (zagarantovan) odziv na promene stanja kontrolisanog objekta [1]. SCADAprogramski paket (Slika 2.1) se sastoji od:

* Akviziciono-upravljačkog sistema,
* Realnog (industrijskog) postrojenja,
* Klijentske radne stanice sa grafičkim interfejsom



Slika 2.1 – Uprošćen prikaz SCADA arhitekture

Upravljačku stranu u manjim sistemima najčešće predstavlja samo jedan računar, dok u složenim sistemima predstavlja veliki broj hardverskih i softverskih resursa. Softver obrađuje podatke i prikazuje ih operateru u grafičkom okruženju. Grafički interfejs treba da bude što intuitivniji, tako da se operateri mogu lako obučavati.

Zadatak udaljene stanice (RTU – eng. Remote Telemetry Unit) je da komunicira sa senzorima i aktuatorima u postrojenju.

Tok podataka od senzora ka glavnoj upravljačkoj stanici naziva se akvizicija. Akvizicija podataka je proces pomoću koga se fizičke pojave iz realnog sveta transformišu u digitalni format podataka pogodan za računarsku obradu. Senzori predstavljaju spregu između realnog i digitalnog sveta. Sama akvizicija se izvršava posredstvom polling-a tj. periodičnog slanja zahteva za dobavljanje podatak. Period akvizicije zavisi od same dinamike promena u sistemu koji se nadzire [1].

Tok podataka od glavne upravljačke stanice ka aktuatorima predstavlja upravljanje. Upravljanje može biti manuelno – od strane operatera ili automatsko – posredstvom logike za automatizaciju u skladu sa zahtevima samog postrojenja kojim se upravlja [1].

Istorijski podaci nam služe za sticanje uvida u rad sistema u dužem vremenskom intervalu. Najčešće se koriste za post-mortem analizu rada sistema u situacijama kada dođe do otkaza celokupnog sistema [1].

## Integracija na Web-u

Za klijentesku radnu stanicu, u SCADA sistemima, dominantan izbor predstavljaju klasične desktop aplikacije. Glavni razlog je to što, za razliku od Web aplikacija, desktop aplikacije mogu da funkcionišu normalno i bez internet veze, pa u slučaju da dođe do otkaza, stanice ne ostaju bez nadzora i upravljanja. U prilog tome ide i činjenica da su u prošlosti internet mreže bile spore i nepouzdane i da Web sistemi nisu bili namjenjeni da osplužuju kritičnu infrastrukturu u realnom vremenu.

Pojavom brzih i pouzdanih internet konekcija, ubrzanog razvoja Web tehnologija i sve većoj popularnosti Web aplikacija stvaraju se preduslovi za upotrebu Web aplikacija i u SCADA sistemima, pogotovo u SoftSCADA sistemima koji tolerišu određena – minimalna kašnjenja zbog nepredvidivog kašnjenja na internetu. Dodatno, na Web platformi ublaženi su problemi minimalnih tehničkih zahteva i puštanja u rad novih verzija aplikacije.

# Web aplikacije

Web aplikacija je aplikativni softver koji se nalazi na, udaljenom, web serveru, za razliku od aplikacija koje se pokreću lokalno na operativnom sistemu računara. Korisnik pristupa web aplikacijama preko interneta, pomoću web pretraživača (npr. Google Chrome, Mozilla Firefox i drugi). Ovakve aplikacije koriste klijent-server arhitekturu.

Većina Web aplikacija, koje komunikaciju vrše preko interneta, za razmjenu informacija koriste HTTP protokol.

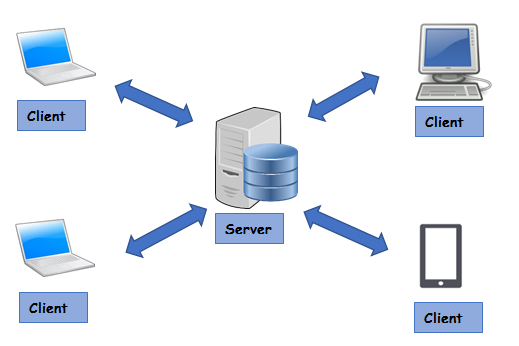
Web aplikacije imaju mnogo različitih upotreba, a sa tim dolazi i mnogo potencijalnih prednosti. Neke uobičajene prednosti Web aplikacija uključuju:

* Veb aplikacije se mogu pokrenuti na više platformi, bez obzira na operativni sistem ili uređaj, pod uslovom da je pregledač kompatabilan,
* Svi korisnici pristupaju istoj verziji aplikacije,
* Nisu instalirane na hard disku, čime se eliminišu prostorna ograničenja,
* Smanjuju troškove kako za preduzeća koja ih razvijaju, tako i za krajnje korisnike, jer preduzeće troši manje resursa na održavanje softvera i korisničke podrške i manji su zahtjevi za resursima računara krajnjeg korisnika.

Povećana upotreba interneta među kompanijama i pojedincima uticala je na način na koji se vode preduzeća. To je dovelo do širokog usvajanja Web aplikacija, jer se kompanije prebacuju sa tradicionalnih modela na mrežne modele. Web aplikacije daju mogućnost preduzećima da pojednostavne svoje poslovanje, povećaju efikasnost i smanje troškove.

## 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 3.1.1).



Slika 3.1.1 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.

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

Neke od mana ove arhitekture jesu:

* 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.

## Web server

Web server je serverski softver ili sistem jednog ili više računara posvećenih pokretanju serverskog softvera, koji može da zadovolji klijentske HTTP zahtjeve na javom Webu ili takođe na privatni LAN i WAN mrežama. Web server može upravljati HTTP zahtjevima klijenata, za Web resurse, koji se odnose na jednu ili više serverskih konfigurisanih Web stranica.

Primarna funkcija Web servera jeste čuvanje, obrada i isporuka Web resursa klijentima. Resursi mogu biti HTML datoteke, slike, skripte, video i dr.

## Web klijent

Web klijent je program sposoban da komunicira sa Web serverima, zahtjeva i prima informacije od njih i obrađuje ih za prikaz korisniku ili druge svrhe. Jedna od najpoznatijih vrsta Web klijenta jesu Web pretraživači.

## 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.

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

* 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.

# 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 [2].
* **.NET –** besplatna, open-source platforma za razvoj mnogih vrsta aplikacija. Dijeli funkcionalnosti između različitih aplikacija i tipova aplikacija pomoću klasnih biblioteka [3].
* **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 [4].
* **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 [5].
* **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 [6].
* **TypeScript** – programski jezik proizved od strane Microsoft-a koji proširuje JavaScriptstatičkom tipizaciom i konceptima objekto-orjentisanog programiranja [7].
* **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 [8].
* **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 [9].
* **HTML** (***eng. HyperText Markup Language***) – opisni jezik specijalno namjenjen opisu strukture Web stranica [10].
* **CSS** (***eng. Cascade Style Sheet***) – jezik koji opisuje kako će se HTMLelementi prikazivati na ekranu. Može kontrolisati izgleda više stranica od jednom [11].
* **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 [12].
* **WCF** (***eng. Windows Communication Foundation***) – 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 [13].

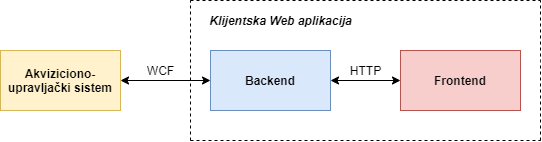
# Opis programskog rješenja

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 veb aplikacija za akviziono-upravljačke sisteme je podjeljena na dvije logičke cjeline (Slika 5.1):

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



Slika 5.1 Arhitektura rješenja

Aplikacija koristi podatke dobijene od strane akviziciono upravljačkog sistema. Podaci su predstavljeni 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:

* Analogni ulazi – nivo kontinualnog električnog signala koji je u svakom trenutku proporcionalan trenutnoj vrijednosti mjerene veličine (pritisak, masa, napon i sl.) [1],
* 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%)[1],
* 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) [1],
* Digitalni izlazi – posredstvom njih se ostvaruje upravljanje tipa uključi/isključi ili otvori/zatvori, za izvršne elemente u postrojenju (npr, sklopke i ventili) [1],
* 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 [1].

Aplikacija podržava i korištenje četiri vrste alarma, u slučaju da za njima postoji potreba:

* 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 zadržavanje trenutne vrijednosti registra 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 5.1). Pretpostavlja se da se registri, definisani istom grupom, nalaze na sukcesivnim memorijskim lokacijama, od početne adrese (startAddress) nadalje. Postojanje grupa omogućava automatsku akviziciju za sve registre, unutar iste grupe, u istom vremenskom periodu, definisanom akvizicionim intervalom (acquisitionInterval).



Listing 5.1 Definicija grupe registara

## Frontend

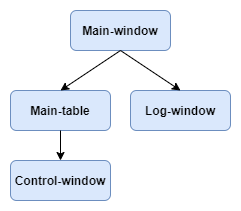
Frontend je prezentacioni dio Web aplikacije, koji omogućava interakciju korisnika sa željenim resursima. Frontend koristi usluge servera, te je ujedno i Web klijent.

Osnovni cilj frontend-a jeste da vjerno i na jednostavan način prikaže podatke dobijene od strane AUS-a, omogući slanje komandi za čitanje ili pisanje podataka i prikaže istorijat izmjena korisniku.

Za razvoj frontend 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 [14].

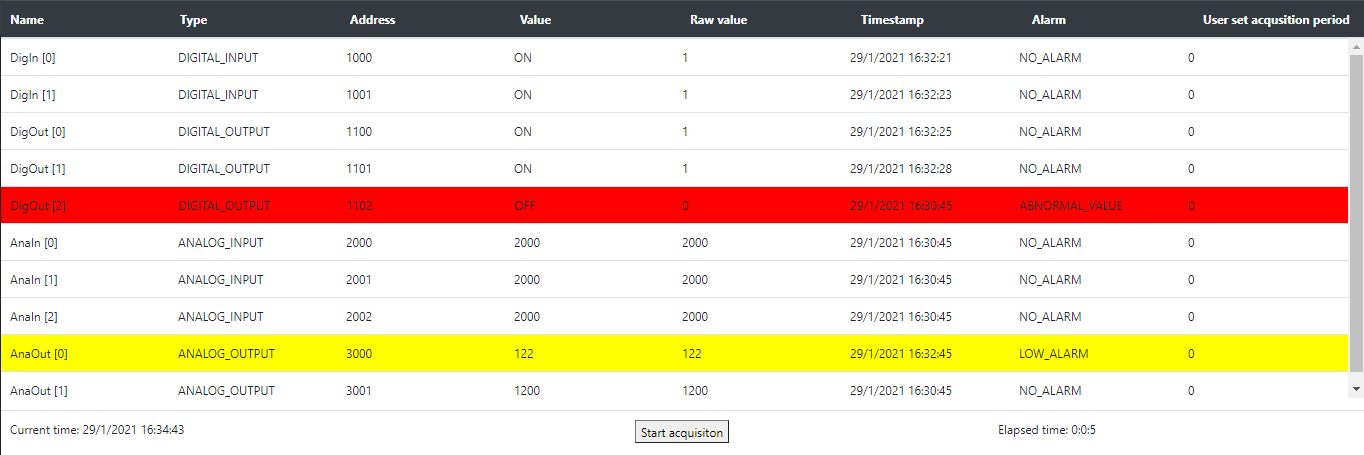
Pored glavne komponente, frontend klijentske Web aplikacije za akviziciono upravljačke sisteme se sastoji od još četiri dodatne komponente:

1. Main-window – za navigaciju izmedju Main-table i Log-window komponenti i njihova roditeljska komponenta (Slika 5.1.1),
2. Main-table – za prikaz podataka iz AUS-a i roditeljska za Control-window (Slika 5.1.1),
3. Log-window – za prikaz istorijata izmjena,
4. Control-window – omogućava slanje komandi za čitanje i pisanje AUS-u.



Slika 5.1.1 Komponente frontend-a

Main-table komponenta u sebi sadrži tabelu u kojoj svaki red predstavlja jedan registar iz AUS-a. 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 3.1.2). 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.

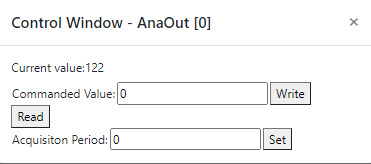
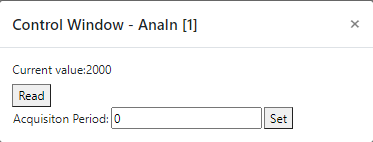


Slika 5.1.2 Main-table komponenta

U zavisnosti od vrste alarma, pozadina reda će da promjeni 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 (Slika 3.1.2).

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:

* Za izlazne registe – dozvoljene su komande promjena vrijednost, čitanje vrijednosti i podešavanje perioda akvizicije (Slika 3.1.3),
* Za ulazne registe – dozvoljene su komande čitanje vrijednosti i podešavanje perioda akvizicije (Slika 3.1.3).

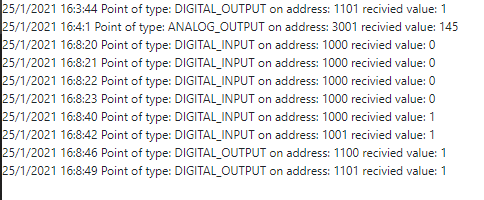
Slika 5.1.3 Control-window komponenta

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 3.1.3).

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 3.1.3).

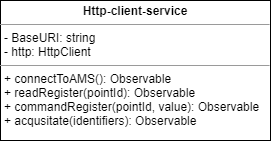
Pritiskom na dugme „Set“ se podešava akvizicija registra na osnovu polja „Acquisiton Period“. Vrijednost 0 znači da akvizija neće biti rađena, dok se za sve ostale vrijednosti, veće od 0, na zadati broj sekundi formira zahtjev za čitanje koji se dalje proslijeđuje backend-u (Slika 3.1.3).

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 3.1.4).



Slika 5.1.4 Log-window komponenta

Pored komponenti, frontend sadrži i servis za komunikaciju sa backend-om, pod imenom http-client-service (Slika 3.1.5). 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 5.1.5 Http-client-service

U servisu su definisana sledeća polja polja:

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

Definase metode servisa su:

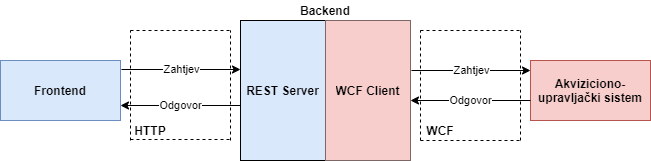
* connectToAMS – omogućava povezivanje klijenta sa akviziciono upravljačkim sistemom, poziva se prilikom pokretanja frontend-a na pretraživaču,
* readRegister – metoda za čitanje najsvježije vrijednosti željenog registra,
* commandRegister – metoda za komandovanje, odnosno promjenu vrijednosti, registra,
* acqusitate – omogućava akviziciju grupe registara definisanih registarskom grupom (Listing 5.1).

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 komnikaciju 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 3.2.1).



Slika 5.2.1 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 5.2.1):

* 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 5.1).
* 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 5.2.1 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 5.2.2).

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 5.2.1),
* Registers – rječnik u kojem se nalaze svi registri dobijeni metodom GetRegisters (Listing 5.2.1) gdje je ključ jedinstveni identifikator registra,
* RegisterGroups – rječnik u kojem se nalaze sve grupe registara dobijene metodom GetRegisterGroups (Listing 5.2.1) gdje je ključ jedinstveni identifikator grupe.



Listing 5.2.2 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 greški

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 optimalno 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

.

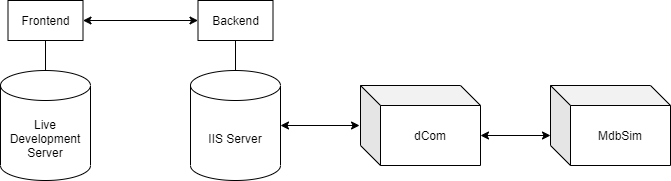
# Testiranje rješenja

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 6.1 Fizička organizacija sistema

# Zaključak

Dobre karakteristike web aplikacije se ogledaju u adekvatnom rukovanju izuzecima, gde su prepoznati 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 interneta 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 pad sistema u AUS-u ne dovodi do pada weba aplikacije i obrnuto.

Međutim, postoji određen broj ograničenja aplikacije. IP adresa AUS-a je programerski 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 ukuca IP adresu svakog pojedinačnog AUS-a, 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://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>, **C#**

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

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

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

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

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

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

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

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

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

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

[13] <https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf>, **WCF**

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

[15] <https://angular.io/api/common/http/HttpClientModule>, **HttpClient modul**