|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Ненад Зеленовић

**НАСЛОВ РАДА**

МАСТЕР РАД

- Мастер академске студије –

Нови Сад, 2021.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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**: | | | Мастер рад | |
| Аутор, **АУ**: | | | Ненад Зеленовић | |
| Ментор, **МН**: | | | др Бранислав Атлагић | |
| Наслов рада, **НР**: | | |  | |
| Језик публикације, **ЈП**: | | | Српски (латиница) | |
| Језик извода, **ЈИ**: | | | Српски/енглески | |
| Земља публиковања, **ЗП**: | | | Србија | |
| Уже географско подручје, **УГП**: | | | Војводина | |
| Година, **ГО**: | | | 2021. | |
| 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) | | |  | |
| Научна област, **НО**: | | | Електротехника и рачунарство | |
| Научна дисциплина, **НД**: | | | Примењене рачунарске науке и информатика | |
| Предметна одредница/Кључне речи, **ПО**: | | |  | |
| **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**: | | Члан: |  |
|  | | Члан: |  | Потпис ментора |
|  | | Члан, ментор: | Др. Бранислав Атлагић, доцент |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UNIVERSITY OF NOVI SAD •**FACULTY OF TECHNICAL SCIENCES**  21000 NOVI SAD, Dositej Obradović Square 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**: | | | Мастер thesis | |
| Author, **AU**: | | | Nenad Zelenović | |
| Mentor, **MN**: | | | Branislav Atlagić, Ph. D. | |
| Title, **TI**: | | |  | |
| Language of text, **LT**: | | | Serbian (latin script) | |
| Language of abstract, **LA**: | | | Serbian/English | |
| Country of publication, **CP**: | | | Serbia | |
| Locality of publication, **LP**: | | | Vojvodina | |
| Publication year, **PY**: | | | 2021. | |
| 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) | | |  | |
| Scientific field, **SF**: | | | Electrical and computer engineering | |
| Scientific discipline, **SD**: | | | Applied computer science and informatics | |
| Subject/Keywords, **S**/**KW**: | | |  | |
| **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**: | | Member: |  |
|  | | Member: |  | Menthor's sign |
|  | | Member, Mentor: | Ph. D. Branislav Atlagić |  |

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ •**ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, ТргДоситејаОбрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ MАСТЕР РАДА** | Лист/Листова: |
|  |

*(Податке уноси предметни наставник - ментор)*

| Студијски програм: | Примењено софтверско инжењерство |
| --- | --- |
| Руководилац студијског програма: | Др Драган Поповић, ред. проф. |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | Ненад Зеленовић | Број индекса: | **Е533/2019** |
| Област: | Електротехника и рачунарство | | |
| Ментор: | др Бранислав Атлагић | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ МАСТЕР РАДА:**

|  |
| --- |
|  |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
|  |

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

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

Образац**Q2.НА.15-07** - Издање 1

Sadržaj

# Uvod

Za vođenje uspešne kompanije potrebno je obezbediti odgovarajući sistem kojim je moguće deliti informacije kroz slojeve i delove kompanije, a da sistem bude pouzdan i sinhronizovan. Takođe te informacije moraju biti razumne i čitljive kroz sve delove kompanije, od pogona pa do kontrolnih soba i na kraju do soba gde se razvijaju biznis planovi. U prošlosti su zaposleni iz biznis slojeva kompanije komunicirali sa zaposlenima iz pogona i kontrolnih soba tako što se za iste pojmove koristili različiti nazivi. To je dovodilo do nesporazuma, nepotrebnog odlaganja i grešaka u toku poslovanja kompanije.

U poslednjih par decenija, industrijske kompanije su investirale u sisteme poslovne logistike (ERP - *Enterprise Resource Planning* sisteme). Takođe veliki deo novca i vremena je isto otišlo na investiranje u automatizaciju kontolnih sistema kao što je SCADA (*Supervisory Control And Data Acquisition*). Sve to sa ciljem približavanja ova dva sistema u jednu celinu. ERP sistemi su korisni tek kada ih snabdevaš sa trenutnim i validnim informacijama, dok se te informacije nalaze u kontrolnim sistemima [**1**]. Mnoge kompanije su još uvek u fazi gde zaposleni ručno razmenjuju i obrađuju potrebne informacije između EPR i kontrolnih sistema. Vremenom se polako počelo shvatati bitnost automatizacije sistema kao i automatizacija toka informacija, što bi dovodilo do smanjenja grešaka i sačuvalo bi dosta vremena. Uvideo se značaj dramatičnog skraćenja vremena između naručivanja proizvoda od strane klijenata i isporuke proizvoda tako što će prave informacije biti dostupne u pravo vreme na pravom mestu. Još više prednosti može se ostvariti kada automatizacija ne obrađuje samo razmenu informacija, ali takođe i omogućava kontrolisanje proizvodnih procesa.

U projektima koji su usmereni na integraciji između ERP i kontrolnih sistema, obično je uključen veliki broj odseka, kao što su proizvodnja, održavanje, laboratorija i kancelarija. Navedeno ukazuje da su u poslednjih par decenija preduzeća suočeni sa prazninom između ERP i kontrolnih sistema i kada se pokuša ta praznina smanjiti, pojavili su se problemi u komunikaciji između ljudi i sistema.

Bilo je potrebno da se razvije standard koji će uvesti odgovarajuće principe, terminologiju za uspešno poslovanje kompanije, kao i mogućnost integrisanja slojeva u jednu sinhronizovanu celinu. Kao rešenje tog problema uvodi se ISA-95 standard.



Slika . – ISA-95 Standard

# TEORIJSKE OSNOVE

## Istorija ISA-95 standarda

ISA-95 predstavlja internacionalni standard koji služi za integrisanje biznis i kontrolnih sistema u cilju smanjenja rizika, troškova i stvaranja grešaka koje idu paralelno sa implementacijom interfejsa između takvih sistema. ISA je globalna neprofitna organizacija. Prvobitno ISA je označavala *Instrument Society of America*, ali je ovaj naziv kasnije preimenovan u *Instrumentation, Systems and Automation Society*. Ovaj naziv je promenjen 2000 godine. ISA definiše svoje ključne aktivnosti kao što su: standardizacija, sertifikacija, obrazovanje i obuka, publikacije i izložbe iz oblasti industrijske automatizacije [**2**].

Iako je standard kreiran 1995 godine, njegovi principi i namena su do današnjeg dana ostali validni i relevanti. Većina razvojnog tima ISA-95 standarda je takođe razvijao stariju verziju standarda pod imenom ISA-88 koji se koristio za kontrolu skupa instrukcija (*batch control*), procesa i signala. Tako da nije čudno što se modeli i terminologija ova dva standarda približno podudaraju. Oba standarda pružaju koncepte koji omogućavaju definisanje kako se određeni proizvodi kreiraju.

## Svrha ISA-95 standarda

ISA-95 nije sistem automatizacije, već metoda, način rada, razmišljanja i komuniciranja. Ova metoda je opisana u nekoliko dokumenata, gde se svaki dokument sastoji ond stotinjak stranica. Sadrže modele i teminologiju koja može da se iskoristi za analizu pojedinačne proizvodnje određene kompanije. Svaki od modela fokusira se na specifične aspekte integracije.

Komuniciranje o sistemu može biti teško, jer različiti ljudi u isti razgovor često koriste različita imena opštim terminima. ISA-95 definiše reči koje se odnose na sisteme poslovne logistike (ERP) i na kontrolne sisteme. ISA-95 stavlja ovu terminologiju u modele koji jasno pokazuju vezu između različitih pojmova. Ovaj princip možemo uporediti sa nacrtima za kuću. Reči *prozor*, *vrata*, *zid* i *krov* su svima nama poznati i koristimo ih za međusobno razgovaranje kada se spominje kuća. Svaka kuća je drugačija, ali i dalje možemo opisati svaku kuću sa istim simbolima i rečima za vrata, krovove, zidove i prozore. Isto se odnosi i na ISA-95. Ne postoje dve slične proizvodne kompanije i ipak možemo koristiti ISA-95 modele i teminologiju za razgovor sa drugima o aktivnostima, funkcijama, tokovima informacija kompanije... Kao rezultat, postalo je lakše ne samo na nivou ljudske komunikacije, već i na tehničkom nivou, za integraciju različitih sistema.

Cilj ISA-95 standarda je da smanji troškove, rizike i greške povezane sa implementacijom interfejsa između ERP i kontrolnih sistema. Standard se može koristiti za pojednostavljenje implementacije novih softverskih proizvoda i da na kraju se stvori laka interoperatibilnost između ERP i kontrolnih sistema. ISA-95 definiše veliki broj potencijalnih prednosti. Omogućava kreirane raznih alata za lakšu integraciju ERP i kontrolnih sistema. Pruža se krajnim korisnicima da lakše kreiraju svoje zahteve. Dodatne pogodnosti tiču se integracije uopšteno, kao što su smanjenje troškova proizvodnih procesa i optimizacija lanca snabdevanja (*Supply Chain*).

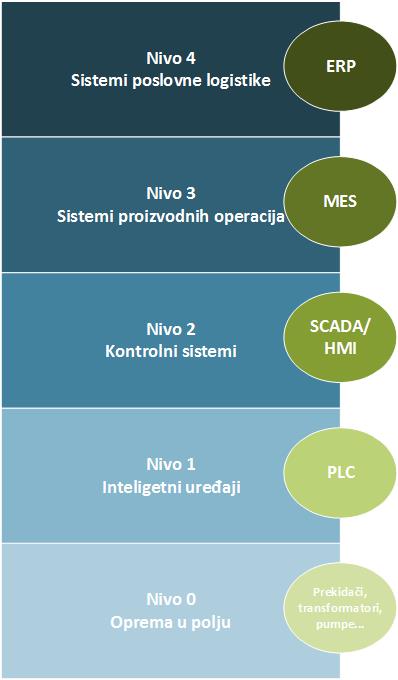
## Nivoi ISA-95 standarda

ISA-95 standard deli postojenja, opremu i imovinu na 5 nivoa [**3**]:

* **Nivo 0 (nulti)** – predstavlja samu opremu u polju (prekidači, transformatori, pumpe...)
* **Nivo 1** – predstavlja opremu koja očitava stanja sa polja tj iz nultog nivoa, kao i manipulaciju nad njima. Iako ovaj nivo vrši manipulaciju nad opremom u polju, on ne može samoinicijativno da pokrene te manipulacije. Za to dobija komande od narednog nivoa. PLC (*Programmable Logic Controller*) predstavlja kontroler koji zadovoljava potrebe ovog nivoa.
* **Nivo 2** – omogućava monitoring, nadgledanje, manuelnu i automatsku kontrolu nad opremom u polju. Ovaj nivo sadrži kompletan uvid u infrastrukturu celog sistema, kao i uvid u svaki vid promene vrednosti opreme u polju. Iz ovog nivoa se zadaju setovi instrukcija za promene vrednosti opreme u polju. Te instrukcije se šalju nivou 1, koji ih izvršava. Sistemi tipični za ovaj nivo predstavljaju SCADA (*Supervisory Control And Data Acquisition*) i HMI (*Human-Machine Interface*).
* **Nivo 3** – ovaj nivo definiše aktivnosti radnog procesa za proizvodnju željenih krajnjih proizvoda, sadrži uvid u detaljan raspored proizvodnje, kao i kompletne istorijske podatke. U ovom sloju se konstantno vrše moguće optimizacije u procesu proizvodnje. Neki od uslova u procesu proizvodnje koji se moraju ispoštovati su da proizvod bude kvalitetan, da se proizvodi bezbedno, pouzdano i efikasno. Najznačajni sistem za ovaj nivo predstavlja MES (*Manufacturing Execution System*)
* **Nivo 4** – u ovom nivou se obavljaju funkcije kao što su operacioni menadžment, planiranje proizvodnje i logistika. Određuje se koji će se materijali koristiti, pregled stanja na zalihama, zaposlenje radnika kao i menadžment njihovih radnih smena. Utvrđuju se dugoročni, srednjoročni ili kratkoročni planovi koji će doneti profit. Sistem koji se koriste za ovakve svrhe je ERP (*Enterprise Resource Planning*)

Drugi nazivi za ove nivoe su:

* **Nivo 1** – Inteligentni uređaji
* **Nivo 2** – Kontrolni sistemi
* **Nivo 3** – Sistemi proizvodnih operacija
* **Nivo 4** – Sistemi poslovne logistike



Slika . - Nivoi ISA-95 standarda

Jedna od bitnih karakteristika nivoa jeste njihova komunikacija. Jedna nivo može da komunicira samo sa susednim nivoima tj sa nivom ispred i iza sebe uz korišćenje odgovarajućih interfejsa.

## Razlike između sistema proizvodnih operacija i poslovne logistike

Sistem proizvodnih operacija i poslovne logistike dele neke uobičajne svrhe u proizvodnom okruženju. Ova dva sistema su dovoljno slična da se njihove jedinstvene karakteristike lako mogu predvideti [**4**]. U ovom poglavlju objasnićemo osnovne razlike između ova dva složena i najveća nivoa ISA-95 standarda. Glavnu razliku između ova dva nivoa možemo videti na tabeli 2.1.

|  |  |
| --- | --- |
| **Sistem proizvodnih operacija (MES)** | **Sistem poslovne logistike (ERP)** |
| * Gledanje u relanom vremenu * Fizičko kretanje proizvoda i odgovornost * Pogled iz radnih centara * Kako se proizvode proizvodi i gde se nalaze? | * Dugoročno gledanje * Troškovi proizvodnje i ukupna zarada * Pogled iz sala za sastanke * Koliko vrede moji proizvodi? |

Tabela . - Razlika između sistema proizvodnih operacija i poslovne logistike

### Fokus svrhe

ERP funkncioniše kao sredstvo za razmenu informacija unutar organizacije. Stistem povezuje svaki deo posla i omogućava nesmetan protok informacija. Delujući kao sveobuhvatni put podataka čini ERP dragocenim alatom za upravljanjem, jer donosiocima odluka daje mogućnost da duboko zarone u svaki deo poslovanja i na smislene načine povežu ranije udaljene podake.

Razlog za dodavanje MES-a je, s druge strane, pružanje sredstava za preciznu kontrolu proizvodnog procesa. MES preduzima korake ili generiše izveštaje na osnovu onoga što se trenutno događa kako bi nadziralo i ispravilo promenljive koje utiču na efikasnost proizvodnje. MES sinhronizuje brojne aspekte izrade kako bi organizovao najbolje moguće rešenje za postizanje manje rasipnog i profitabilnijeg procesa.

### Sredstva za prikupljanje podataka

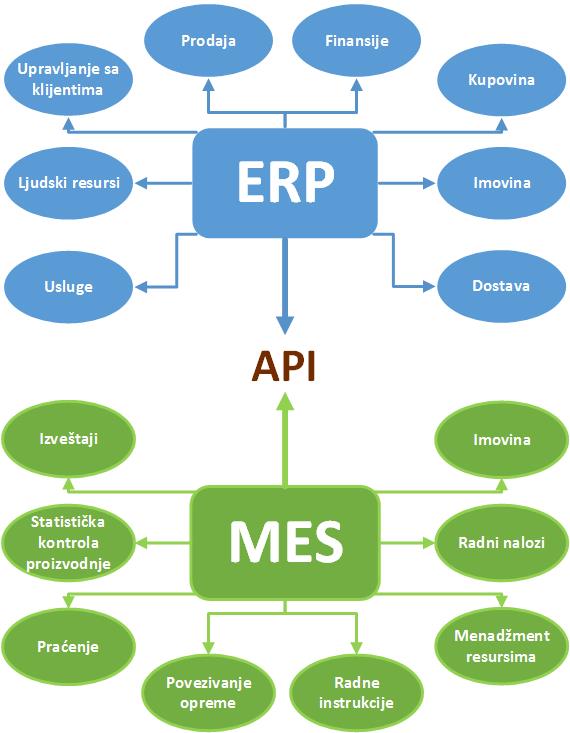
Ljudi obično većinu informacija daju u ERP. Ovo ručno prikupljanje podataka dobro funkcioniše, jer se sistem prvenstveno bavi prikupljanjem, organizovanjem i razmenjivanjem informacija širom organizacije radi planiranja i vođenja. Savremeni ERP sistemi imaju jednu bazu podataka za celu organizaciju, tako da se suvišni unosi smanjuju ili uklanjaju. Jedno sladište podatka takođe čini nesmetan proces razmene informacija u različitim funkcionalnim oblastima. Fokusira se na rad u vremenskim okvirima kao što su meseci, nedelje i dani.

MES se koristi za pokretanje proizvodne operacije, upravljanjem i izveštavanjem o aktivnostima postrojenja u slučaju događaja, u realnom vremenu. Održava se evidencija istorije podataka i ističu se izuzetci. Tipične funkcije su elektronski zapisi skupa instrukcija, merenje i doziranje, upravljanje opremom, podešavanje i čišćenje. Ove informacije pružaju tačne i pravovremene informacije ERP-u, omogućavajući kompaniji da reaguje dovoljno brzo kako bi mogla pratiti korake koji se brzo menjaju. Smanjivanje latencije na ovaj način povećava šanse za ostvarivanje većeg novčanog prinosa. MES radi u vremenskim okvirima kao što su dani, smene, sati i minute.

### Šta pokreće akcije

ERP izvršava instrukcije na osnovu finansijskih transakcija. Kada kupci izvrše narudžbine, dobavljači šalju te narudžbine. ERP je multifunkcionalan, ali ga programeri grade oko prvenstveno ekonomske podstrukture.

S druge strane, MES je vođenjem dešavanjem događaja. Ovaj sistem je dizajniran za nagledanje događaja specifičnih za proizodno okruženje. Nagledanje podataka u realnom vremenu omogućava MES-u da obavlja zadatke kao što su obezbeđivanje usaglašenosti za proizvodnim procesom, praćenje potrošnje zaliha, zakazivanje održavanja mašina na osnovu performansi i preuređivanje postupaka radi efikasnijeg korišćenja raspoloživih resursa.



Slika . - Grafički prikaz ERP i MES nivoa

## Prednosti korišćenja ISA-95 standarda

ISA-95 standard se može koristiti kao metoda za definisanje interfejsa između sistema poslovne logistike i kontrolnih sistema. Primenom ovog standarda troškovi se mogu drastično smanjiti. Integracija rešenja različitih dobavljača postaće manje složena kada svi koriste ovaj standard.

Pomaže pri unapređenju komunikacije između različitih kompanija. Svaka proizvodna kompanija koristi sopstvenu teminologiju za opisivanje funkcija, aktivnosti i odeljenja u kompaniji. Kada morate da radite sa spoljnim konsultantima, komunikacija će biti teška. Velika je šansa da ćete te pričati o različitim stvarima kada koristite iste izraze ili obrnutno. Problem se uvećava svaki put kada se započne novi projekat ili svaki put kada se razgovara sa različitim klijentima. Dakle, kada se razgovara o interfejsima, bilo bi dobro da se diskusija zasniva na standardnoj teminologiji tako da obe strane ostvare zajednički jezik.

Integracija između sistema poslovne logistike i kontrolnih sistema pre ISA-95 za vreme pravljenja nekog projekta je trajala između 1-2 godine sa procentom od 50% i manje uspešnosti. Nakon korišćenja ISA-95 standarda vremenski period se smanjio na 2-4 meseca sa procentom od 90% i više uspešnosti. Takođe definisanje specifikacija između ova dva sistema pre ISA-95 standarda je trajalo preko godinu dana, dok uz korišćenje ovog standarda taj vremenski period se prepolovio i iznosio je 6 meseci u nekim slučajevima i manje. Na osnovu ove statistike možemo da utvrdimo da je ISA-95 standard bio veliki uspeh.

ISA-95 standard nije ograničen na stvaranje interfejsa između sistema poslovne logistike i kontrolnih sistema, kako sugeriše osnovna namena standarda. To je takođe dobra smernica za sastavlajnje korisničkih zahteva, opisivanje funkcionalnih zahteva, razvijanje MES aplikacija i baza podataka, analiza i upoređivanje kapaciteta na različitim proizvodnim mestima i sticanje uvida u optimizaciju proizvodnih procesa. Modeli i terminologija ISA-95 standarda osigurava da svi ljudi koji koriste ovaj standard razmišljaju i razgovaraju na isti način o proizvodnim aktivnostima, sistemima i informacijama [**5**].

# TEHNOLOGIJA I ALATI

U ovom poglavlju pričaćemo o svim tehnologija i alatima koji su korišćeni za implementaciju rada.

## .NET Framework

Okruženje za razvoj sofvera, razvijano od strane Microsoft-a za Windows platforme. Uključuje veliku biblioteku klasa (*Framework Class Library*). Microsoft je sa razvojem .NET-a počeo ranih 1990-tih, pod nazivom *Next Generation Windows Services*. Početkom 2000-tih prva beta verzija .NET 1.0 je objavljena, a u avgustu 2000. u saradnji sa Intel-om i HP-om, Microsoft je počeo sa standardizacijom *CLI-ja*, koja će omogućiti izvršavanje različitih programskih jezika na različitim arhitekturama-platformama.

Programi se izvršavaju kroz softversko okruženje *CLR*, virtualnu mašinu koja sadrži: *memory managment*, *exception handling*, *garbage collector*... Omogućeno je korišćenje 25 programskih jezika od kojih su najpolularniji C#, C++ i VisualBasic. Jezici se, svaki preko svog kompajlera, kompajliraju u *CIL* među-jezik. Zatim, u zavisnosti od toga na kojoj se platformi izvršava, *CLR* kompajlira *CIL* u mašinski kod. Glavni razvojni alat je Visual Studio [**6**].

## Microsoft Visual Studio

Predstavlja integrirano razvojno okruženje. Koristi se za razvoj računarskih programa za Windows, veb-stranica, aplikacija i usluga. Koristi Microsoft-ove platforme za razvoj raznih API-ja za Windows, Windows Forms, WPF. Program takođe sadrži alate poput dizajnera oblika koji se koristi za pravljenje aplikacija s grafičkim korisničkim interfejsom, veb-dizajnera, dizajnera klasa i dizajnera shema baza podataka. Visual Studio podržava različite programske jezike i dozvoljava uređivaču koda i debugger-u da podržava gotovo bilo koji programski jezik. Ugrađeni jezici su C, C++, VB.NET, C# i F#. Također podržava XML, HTML, JavaScript i CSS [**7**].

## C#

Objektno orijentisan programski jezik koji je razvio Microsoft početkom 21-og veka. Reč je o jeziku opšte namene koji služi za pravljenje aplikacija u okviru .NET okruženja. Iako ne postoji dugo kao neki drugi programski jezici, C# je jedan od najpopularnijih jezika. C# se odlikuje velikim mogućnostima, jednostavnošću upotrebe i lakoćom usvajanja, zbog čega je danas jedan od najpopularnijih programskih jezika koji svoju primenu nalazi u velikim i malim kompanijama i u različitim oblastima [**8**].

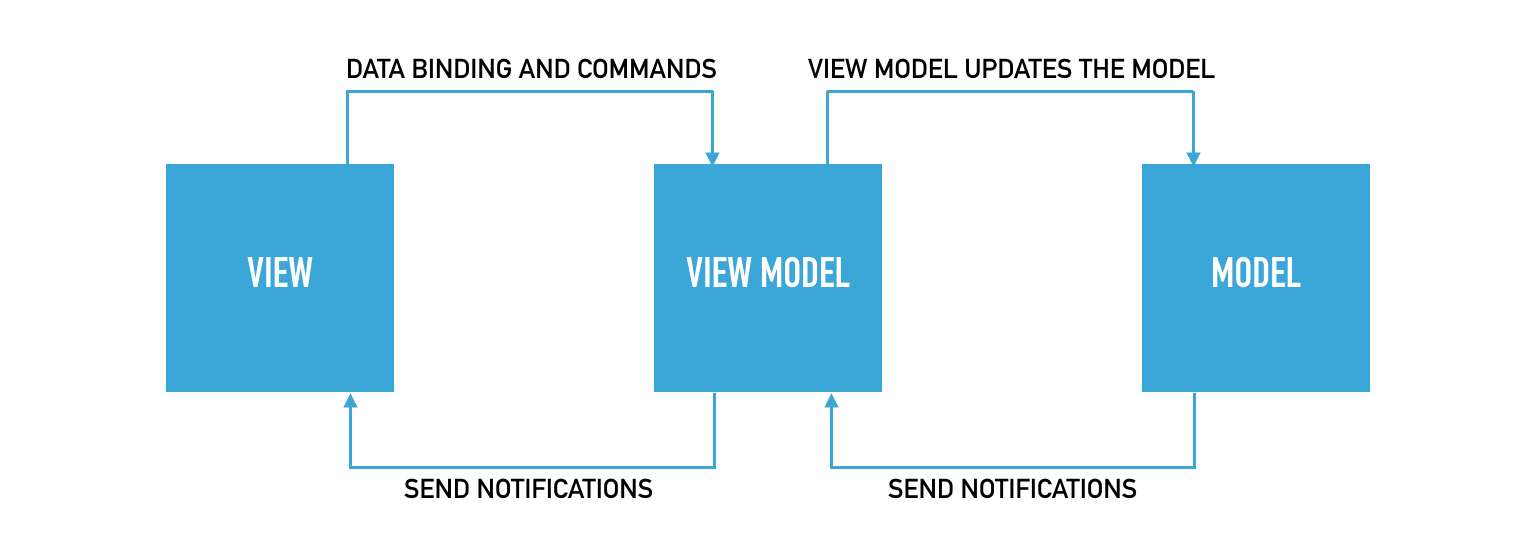
## Windows Presentation Form (WPF)

Grafički podsistem za renderovanje korisničkog interfejsa u aplikacijama zasnovanim na Windows operativnim sistemima [**9**]. Razvijen je od strane Microsoft-a. WPF koristi XAML, izveden od XML-a da definiše i poveže različute UI elemente. WPF aplikacije mogu biti razvijene kao samostalni desktop programi ili kao ugrađeni objekti u website stranicama. Ima za cilj da objedini niz zajedničkih interfejs elemenata, kao što su 2D/3D renderovanja, fiskirana i adaptivna dokumenta, tipografiju, vektorsku grafiku...

## Model-View-ViewModel (MVVM)

Patern koji razdvaja aplikaciju na više komponenti tako da svaka komponenta ima svoje specifične odgovornosti. MVVMarhitektura je preporučena od strane Google-a kao jedan od najboljih načina strukture koda Android aplikacija [**10**]. Pri korišćenju MVVMpaterna kod aplikacije je razdvojen na tri dela:

* *View* - Ova sekcija sadrži klase (Aktivnosti i Fragmenti) koje su zadužene za prikaz interfejsa i prihvatanje akcija korisnika, nakon čega o tome obaveštava *ViewModel*
* *ViewModel* - Ova sekcija sadrži klase koje su zadužene za pristup podacima (Repository) i da obaveste *View* ukoliko dolazi do promena.
* *Model* - Ova sekcija sadrži klase zadužene za pristup raznim vrstama podataka (baza, webservice...) i da izvrši abstrakciju takvih izvora podataka kroz jedan API.



Slika . - MVVM patern

## Windows Communication Fondation (WCF)

Servisno orijentisani model razmene poruka, koji omogućava programima da komuniciraju preko računarske mreže ili lokalno. WCFje alat koji u sebi uključuje set biblioteka razvijenih za distibuirano programiranje [**11**].

## Microsoft SQL Server

Predstavlja relacijsku bazu podataka kojoj je primarni jezik za upite *Transact SQL* (T-SQL), što znači da osim osnovnih i klasičnih (SELECT tipa) SQL upita dozvoljava i složenije stvari poput *if* naredni ili *while* petlji. *Transact SQL* nastao je kao plod suradnje Microsoft*-a* i Sybase-a. SQL server je baza podataka koja je namenjena manjim i srednjim bazama.

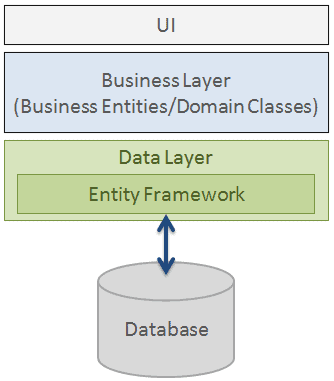
Prva verzija SQL Servera koji ima veze sa Microsoft*-*om izašla je 1989. godine pod imenom „SQL Server for OS/2 1.0“. Ta verzija bila je identična Sybase-ovom SQL serveru 3.0 koji je radio pod Unix sistemom. Microsoft SQL Server pod tim imenom se počeo prodavati 1992. godine, a puno ime mu je glasilo Microsoft SQL Server 4.2 koji se još uvek vrtio na OS/2 platformi. Prvi SQL Server za *Windows* operativne sisteme izašao je isto kada i sami Windows-i.

SQL Server je prva baza podataka na svetu koja je posedovala korisnički interfejs (*UI*). Sve baze su u tadašnje vreme radile pomoću „command-line“ sistema koji zna biti izrazito nezgodan [**12**].

## Entity Framework

Entity Framework predstavlja *ORM Framework* otvorenog koda za .NET koje podržava Microsoft. Omogućava programerima da rade sa podacima koristeći objekte klasa bez fokusiranja na osnovne tabele i kolone baze podataka u kojima se ti podaci čuvaju. Uz Entity Framework, programeri mogu raditi na višem nivou apstrakcije kada se bave podacima i mogu stvoriti i održavati aplikacije sa manje koda.

Na slici 3.2 možemo primetiti da se Entity Framework uklapa između poslovnih entiteta i baze podataka. Sprema podatke pohranjene u svojstvima poslovnih entiteta, a takođe preuzima podatke iz baze podataka i automatski ih pretvara u objekte poslovnih entiteta [**13**].



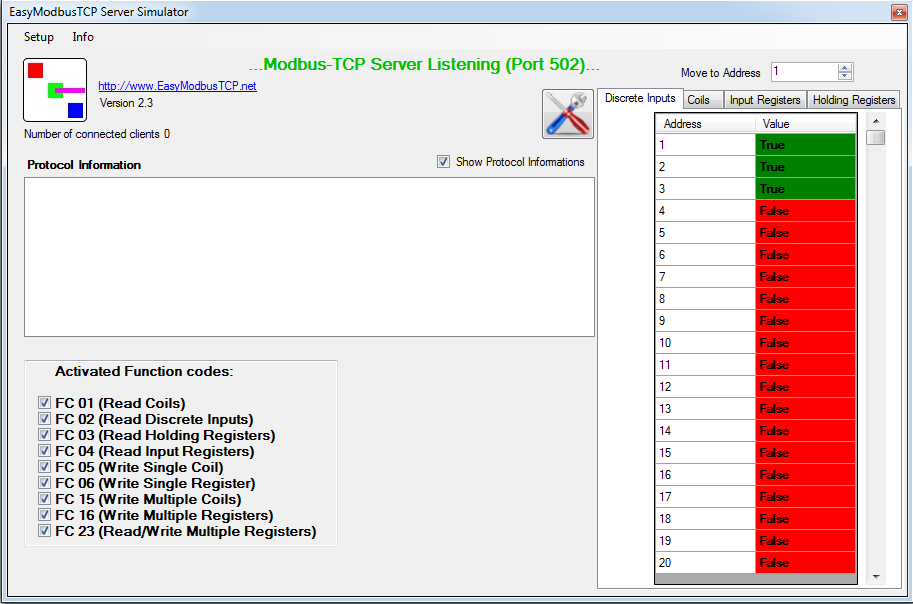
Slika . - Pozicija Entity Framework-a u arhitekturi sistema

## Easy Modbus Server Simulator

Modbus je protokol za prenos podataka koji se koristi zajedno sa *PLC*-om. Modbus je postao standard za komunikacijski protokol i sada je glavno sredstvo za povezivanje industrijskih elektronskih uređaja. Razvijen je za industrijske aplikacije, relativno je jednostavan za primenu i održavanje u poređenju sa drugim standardima [**14**].

Easy Modbus Server Simulator je dizajniran da pomogne programerima koji koriste Modbus protokol. Svi Modbus podaci su prikazani u Server Simulatoru i omogućava debagovanje grešaka klijentskih aplikacija. Podržava operacije čitanja i pisanja. Server Simulator podržava Modbus *TCP*, Modbus *UDP* i Modbus *RTU* u .NET verziji. Pomoćni kodovi funkcija mogu biti onemogućeni. Meni svojstva omogućava promenu protokola između Modbus TCP-a, UDP-a i RTU-a [**15**].

Vrednosti sa kojima Server Simulator radi su: *Coils, Discrete inputs, Input registers, Holding registers.* Prve dve vrednosti predstavljaju diskretne vrednosti, dok zadnje dve su analogne.



Slika . - Easy Modbus Server Simulator

## Dizajn paterni

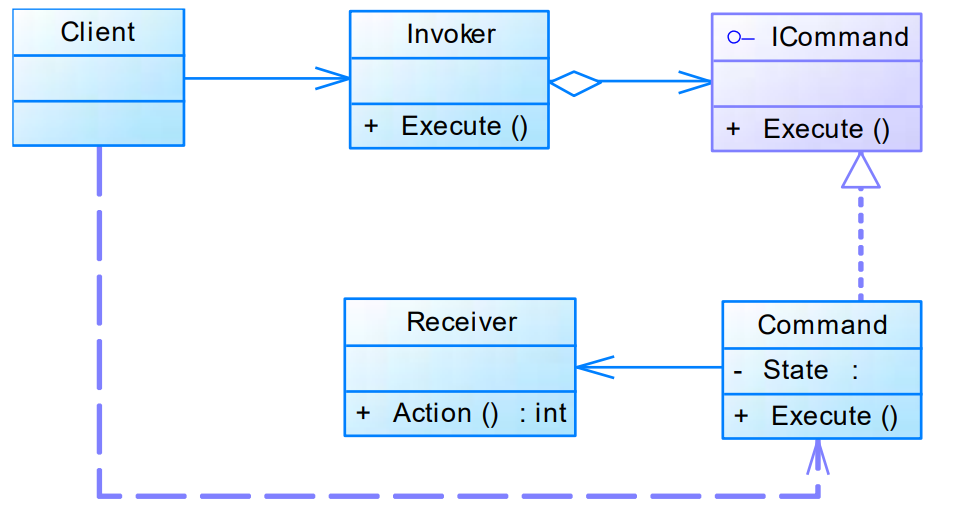
U softverskom inženjerstvu, *dizajn patern* ili *šablon* ili *obrasac* je opšte, ponovo upotrebljivo rešenje za česte probleme koji se sreću prilikom projektovanja softvera. *Dizajn patern* nije gotov dizajn koji se može direktno pretvoriti u izvorni kod. On služi samo kao opis ili šablon prilagođen da reši neki opširniji problem u posebnom kontekstu [**16**]. Postoje tri vrste *dizajn paterna*:

* *Strukturalni paterni* - bave se kompozicijom i obično predstavljaju različite načine za definisanje odnosa među objektima. Oni obezbeđuju da kada je neophodna promena u jednom delu sistema, ostatak sistema ne mora da se menja. Takođe pomažu da svaki deo sistema radi ono čemu je najbolje prilagođen. Neki od *strukturalnih paterna* su: Decorator, Facade, Flyweight, Adapter i Proxy.
* *Kreacioni paterni* - ovi paterni bave se kreacijom objekata, na način prilagođen određenoj primeni. Posebno su važni u situacijama u kojima bi uobičajen pristup kreiranju objekata doveo do povećanja kompleksnosti projekta. Neki od paterna koji spadaju u ovu grupu su: Constructor, Factory, Prototype, Singleton i Builder.
* *Bihevioralni paterni* - Ova grupa paterna tiče se poboljšanja komunikacije između različitih objekata u sistemu. Poznati primeri su: Strategy, Iterator, Mediator, Observer i Visitor.

*Dizajn paterni* koji su korišćeni u implementaciji rada biće nabrojavi i objašnjeni u tekstu ispod.

### Command

*Command* patern kreira distancu između klijenata koji zahtevaju operacije i objekata koji ih izvršavaju. Patern je izrazito višestran. On može da podrži: slanje zahteva ka različitim objektima, smeštanje zahteva u redove, logovanje i odbijanje zahteva [**17**].



Slika . - Command patern

U okviru UML dijagrama možemo videti više učesnika čije su uloge sledeće:

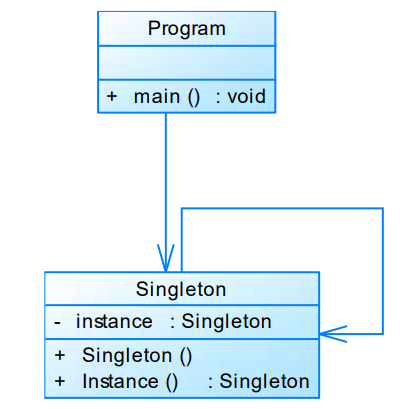
* *Client* - kreira i izvršava komande
* *ICommand* - interfejs koji navodi operacije Execute koje se mogu izvršiti
* *Invoker* - poziva klasu Command da izvrši određenu akciju
* *Command* - klasa koja implementira Execute operaciju tako što uključuje operacije iz klasa Receiver
* *Receiver* - klasa koja može da izvrši zahtevanu akciju
* *Action* - operaciju koju je potrebno izvršiti

*Command* patern na prvi pogled ima puno učesnika, ali se neki od njih odbacuju kada se koriste delegati.

Razlog uvođenja paterna: Ako želimo da implementiramo pravi *MVVM* patern, potrebno je odvojiti svu poslovnu logiku iz *View*-a i smestiti u *ViewModel*. Kada se doda neko dugme u *View*-u za to dugme je potrebno uraditi *binding* (vezivanje). Sa tim *binding*-om kažemo da će sva poslovna logika premestiti u *ViewModel* i tamo će se pozvati. Takođe mora postojati implementirana metoda na *ViewModel*-u, koja će biti pozvana preko kliktanja na dugme. Ta metoda predstavlja komandu. Sam taj *binding* prestavlja akciju na koju se trigeruje komanda.

### Singleton

*Singleton* pattern ograničava instanciranje klase i osigurava da samo jedna instanca date klase postoji i pruža globalnu tačku pristupa ka toj instanci. Patern osigurava da je klasa instancirana samo jednom i da su svi zahtevi upućeni ka tom jednom i samo jednom objektu [**18**].



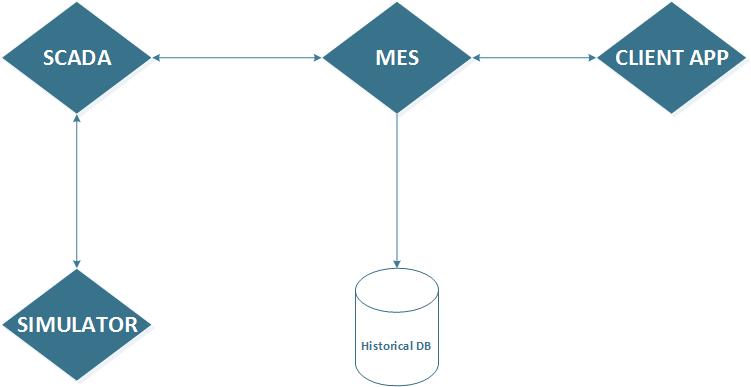
Slika . - Singleton patern

UML dijagram klasa koji predstavlja *singleton* patern je prikazan na slici 3.5. Sa slike se može videti da kreiramo samo klasu *Singleton* koja obezbeđuje da će biti kreirana samo jedna njena instanca.

Razlog uvođenja paterna: U radu *Singleton* i *MVVM* su na neki način usko povezani. Kada pravimo neki *View* u i ako znamo da će objekti unutar tog *View* biti dinamični, onda se dešava problem da tokom dizajniranja nemamo uvid u izgled *View* zbog dinamičnih objekata. Rešenje toga jeste pravljenje jednog dizajn *ViewModel*-a u kojem možemo definisati statičke objekte i tako imati uvid tokom dizajniranja. Povezivanje tog dizajn *ViewModel*-a i *View* pored korišćenja *MVVM* koristi se i *Singleton* patern.

# aRHITEKTURA SISTEMA

Aplikacija sistema je podeljena na nekoliko komponenti. Svaka komponenta ima neka svoja zaduženja i procese koje obavlja nezavisno od ostatka sistema i drugih komponenti. Arhitektura sistema i komunikacija između komponenti je prikazana na slici 4.1.



Slika . - Arhitektura sistema

Komponente sistema predstavljaju:

1. *Simulator -* Simulator predstavlja third-party aplikaciju koja služi za simuliranje vrednosti elemenata u polju. Ostatak teksta o Simulatoru možemo pronaći u poglavlju 3.9.
2. *SCADA* - je servis koji vrši periodičnu akviziciju stanja vrednosti na simulatoru koristeći MODBUS protokol. Kada primi trenutno stanje, šalje se razlika trenutnog i prethodnog stanja na MES servis na interpretaciju.
3. *MES* - servis koji prima podatke sa Scada-e i interpretira njihovo značenje. Dužan je da proveri da li se neki od uređaja nalazi u alarmnom stanju i da zapisuje podatke u SQL bazu podataka(na slici Historical DB). Za to vreme se pristigli podaci proslede na klijentskoj aplikaciji. Ako se utvrdi da je neki uređaj u alarmnom stanju, dodatne informacije se šalju klijentskoj aplikaciji. Na zahtev klijenta generiše izveštaje i šalje ih nazad.
4. *Historical DB* - predstavlja SQL bazu podataka u kojoj se čuva stanje sistema. Takođe čuvaju se svi događaji koji su se desili u sistemu tj. sva alarmatna stanja. Na zahtev MES komponente, Historical DB vraća tražene vrednosti.
5. *Client App -* korisnik za interakciju sa sistemom ima na raspolaganju korisnički interfejs. Funkcionalnosti mu omogućavaju manipulaciju uređajima koji se nalaze u sistemu, prikaz izveštaja, kao i pregled alarmatnih stanja u sistemu.

U ovom radu fokus je bio na komandovanju opreme u polju preko klijentske aplikacije ili preko simulatora, poštovaći ISA-95 standarde i koncepte. Na osnovu toga implementirani su svi nivoi ISA-95 standarda, osim sistema poslovne logistike. U sistemu postoji simulator koji simulira rad prekidača u polju. *SCADA* vrši periodičnu akviziciju stanja vrednosti na simulatoru i te vrednosti šalje ka *MES*-u. *MES* obavlja biznis logiku sistema, čuva istorijske događaje u bazi podataka, generiše izveštaje... Pored toga postoji i klijentska aplikacija preko koje možemo da komandujemo sa prekidačima, dobijamo povratne vrednosti u slučaju komandovanja preko simulatora u vidu alarma, kao i mogućnost ispisivanje izveštaja.

U narednom poglavlju objasnićemo detaljan rad aplikacije i njenih komponenti.

# IMPLEMENTACIJA APLIKACIJE

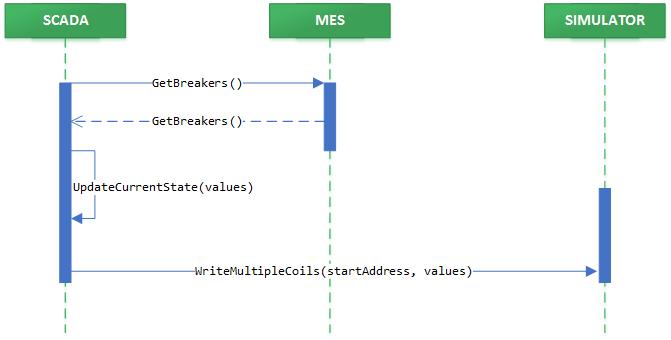
U ovom poglavlju biće objašnjena implementacija i rad sledećih komponenti aplikacije:

## SCADA

### Inicijalno pokretanje

Da bi se ostvarile visoke performanse *SCADA* komponente, potrebno je ovaj servis što više olakšati kad su u pitanju procesi koji se dešavaju u njemu. Želimo da *SCADA* servis ima što manje poslova koje treba da obavlja. Razlog ostvarivanja visokih performansi jeste to da želimo da *SCADA* što pre očita nove vrednosti sa simulatora i te vrednosti prosledi dalje u sistem da bi se krajni korisnik mogao obavestiti o novonastalim promenama. Zbog ovakvih uslova, *SCADA* ne ostvaruje pristup istorijskim podacima niti vrši neku biznis logiku sistema.

Nakon prvog pokretanja *SCADA* servisa potrebno je prethodno sačuvano stanje prekidača upisati u simulator. Pošto *SCADA* ne sadrži te vrednosti, ona mora da ih zatraži od servisa koji poseduje te vrednosti tj. od *MES*-a. Otvara se komunikacioni kanal prema *MES*-u i traži se trenutno stanje prekidača. Nakon prijema vrednosti, prvo se ažuriraju vrednosti u *SCADA* memoriji, a zatim se i ažurira stanje u simulatoru. Na slici 5.1 možemo da vidimo dijagram sekvenci ove opisane akcije. Kada se zavšri ažuriranje stanja, *SCADA* može da nastavi sa svojim primarnim zadatkom, a to je akvizicija podataka sa simulatora.



Slika . - Inicijalni start SCADA servisa

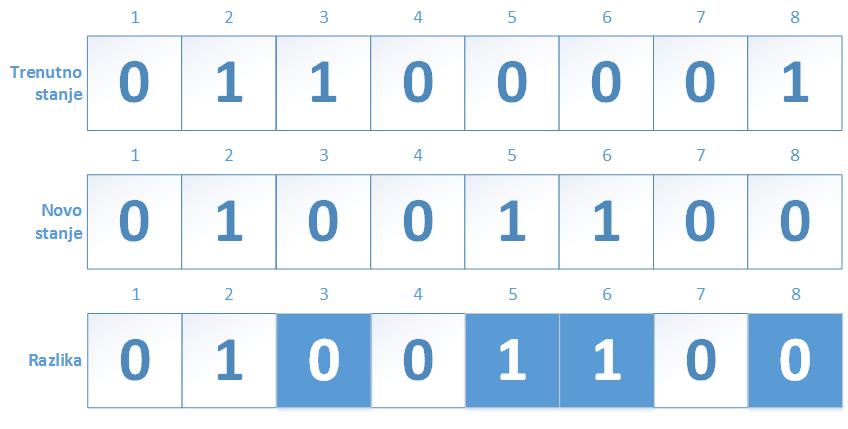
*SCADA* poseduje svoju lokalnu bazu podataka, ali u toj bazi postoji samo jedna tabela pod nazivom **CoilsAddress**. Značenje te tabele predstavljeno je na tabeli 5.1. Razlog postojanja ove baze jeste da bi se optimizovala akvizicija podataka sa simulatora. Iz baze možemo dobiti informaciju o minimalnoj i maksimalnoj vrednosti korišćene adrese. Na osnovu te informacije možemo da smanjimo krug prekidača za koje je potrebno vršiti akviziciju. Bez ove informacije morali bi vršiti akviziciju svih adresa sa simulatora, bez obzira da li se dešava promena ili ne.

|  |  |
| --- | --- |
| **Address *(int)*** | **Označava adresu na simulatoru** |
| **Id *(int)*** | **Označava jedinstveni identifikacioni broj prekidača u sistemu** |
| **Used *(bool)*** | **Označava da li je neki prekidač zauzeo mesto na datoj adresi u simulatoru** |

Tabela . - Prikaz redova tabele CoilsAddress

### Akvizicija podataka sa simulatora

Pošto se stanje prekidača u polju tj. simulatoru može konstantno menjati, mora se periodično proveravati njihovo stanje. *SCADA* periodično, koristeći MODBUS protokol, proziva simulator i „pita“ kakvo je trenutno stanje u polju. Od simulatora dobije informaciju o novom stanju prekidačke opreme. Po prijemu novog stanja primljena je informacija o stanju svakog prekidača u polju. Međutim, tu se nalaze i prekidači kojima se nije promenilo stanje, a čak može da se desi i da se nijednom nije promenilo stanje. Slika 5.2 pokazuje kako se utvrđuje gde je došlo do promena. Ovom metodom se smanjuje opterećenje ostalih komponenti kojima se prosleđuje stanje.



Slika . - Ilustracija iznačunavanja razlike stanja prekidača

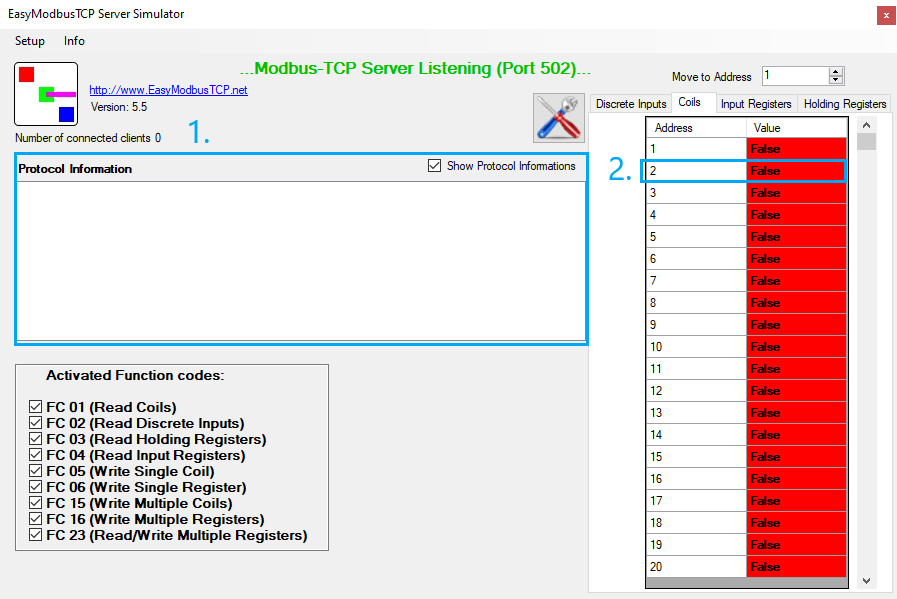
Po prijemu novog stanja, trenutno i novo stanje se proslede u funkciju koja utvrđuje razliku između ta dva stanja. Utvrđivanje se vrši tako što se izvrši logička operacija **ekskluzivno ili** između njih da bi se dobili indeksi izmenjenih vrednosti. Izlaz funkcije je novo stanje sa označenim promenjenim vrednostima. Nakon toga se novo stanje sačuva kao trenutno stanje i promenjene vrednosti pošalju na *MES* komponentu.

### Komandovanje

Kada je reč o komandovanju *SCADA* radi samo ono što joj je zadano. Tu ne postoji nikakva logika. Od *MES* dobija instrukcije i te instrukcije prosleđuje ka simulatoru. Te instrukcije predstavljaju otvaranje/zatvaranje prekidača. Pošto u sistemu postoje prekidači, svaki prekidač u sebi sadrži jedinstveni identifikacioni broj. *MES* pošalje na *SCADA* sistem instrukciju otvori/zatvori prekidač i identifikacioni broj prekidača. Pošto *SCADA* u sebi sadrži listu **CoilsAddress** na osnovu toga možemo da odredimo koji prekidač na kojoj adresi u simulatoru treba da otvorimo/zatvorimo.

## Simulator

Za simulaciju sistema koristimo Easy Modbus Server Simulator, sa kojim vršimo komandovanje opreme u polju. Sve izmene nastale u simulatoru, šalju se na SCADA servis preko Modbus TCP protokola. Simulator iza sebe nema nikakvu logiku, njegov posao je da “simulira” tj. samo vrši otvaranje/zatvaranje opreme u polju. On nema uvid o kakvoj je to opremi reč, kakve su specifikacije opreme, naziv, primena... Zbog ovakvog načina rada, simulator vrši komandovanje u veoma kratkom vremenskom roku, u milisekundama.

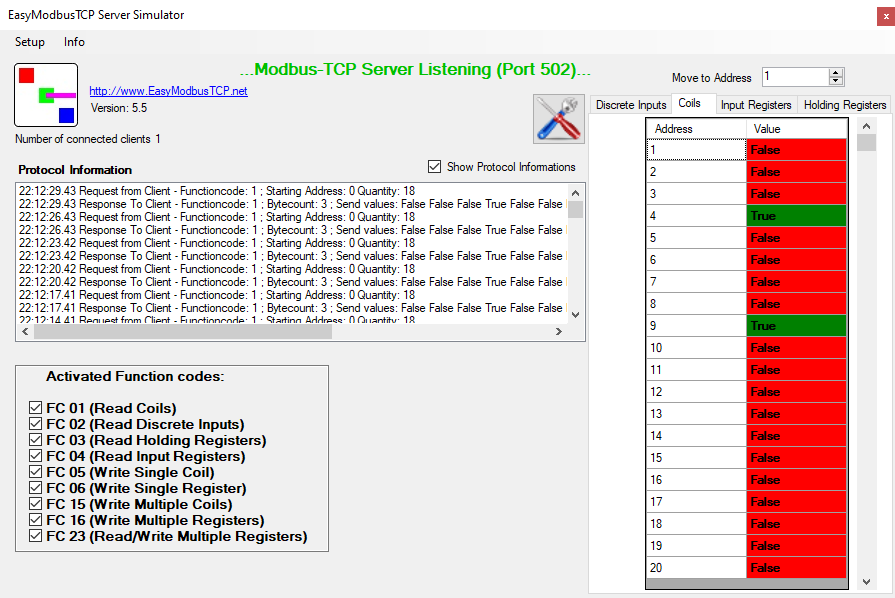


Slika . - Simulator pre starta SCADA servisa

Na slici 5.3 vidimo inicijalno stanje simulatora nakon prvog paljenja. Možemo da primetimo par bitnih prozora unutar simulatora:

1. Informaciona tabla – ovde informacije o vremenu kada je simulator primio podatke od *SCADA* servisa i kada je poslao vrednosti na *SCADA* servis
2. Tab Coils sa mogućim adresama za komandovanje. Vrednosti koje su moguće su otvoreno/zatvoreno tj. **true/false**.

Nakon startovanja *SCADA* servisa i kad se proslede vrednosti na simulator, izgled možemo da vidimo na slici 5.4. Informaciona tabla se popunila sa raznim informacija, dok neke adrese na tab Coils-u su promenile svoju vrednosti. Komandovanje se vrši tako što kursor miša postavimo na željenu adresu i dvoklikom potvrdimo tu akciju.



Slika . - Simulator nakon starta SCADA servisa

## MES

### Inicijalno pokretanje

Nakon inicijalnog pokretanja *MES* servisa, pokreće se inicijalizacija modela tako što se čitaju podaci iz baze podataka i na osnovu podataka se inicijalizuju objekti koji se smeštaju u model. U sistemu objekti koji se mogu inicijalizovati su prekidači i alarmi. Pošto alarme koristimo samo kada generišemo izveštaje i kada se vrši obrada akvizicije i u oba ta slučaja vršimo upis i čitanje iz baze podataka. Tako da njih nije potrebno smeštati u model, dok kod prekidača je druga stvar. Da ne bi na svaku obrade akvizicije vadili podatke iz baze podataka i tako vršili proveru koji prekidač je promenio stanje, brže i lakše je da ti podaci nam se nalaze u modelu.

Znači nakon inicijalnog pokretanja izvlačimo podatke o prekidačima iz baze pomoću *Entity Framework*-a i čuvamo ih u listama koje se nam se nalaze u modelu. Vrednosti i polja koja karakterišu prekidač u sistemu možemo videti na tabeli 5.2.

|  |  |
| --- | --- |
| **Id *(int)*** | **Označava jedinstveni identifikacioni broj prekidača u sistemu** |
| **Name (nvarchar(max))** | **Označava naziv prekidača** |
| **Current state *(bool)*** | **Označava trenutno stanje prekidača** |
| **Last state (bool)** | **Označava prethodno stanje prekidača** |

Tabela . – Prikaz redova tabele prekidača

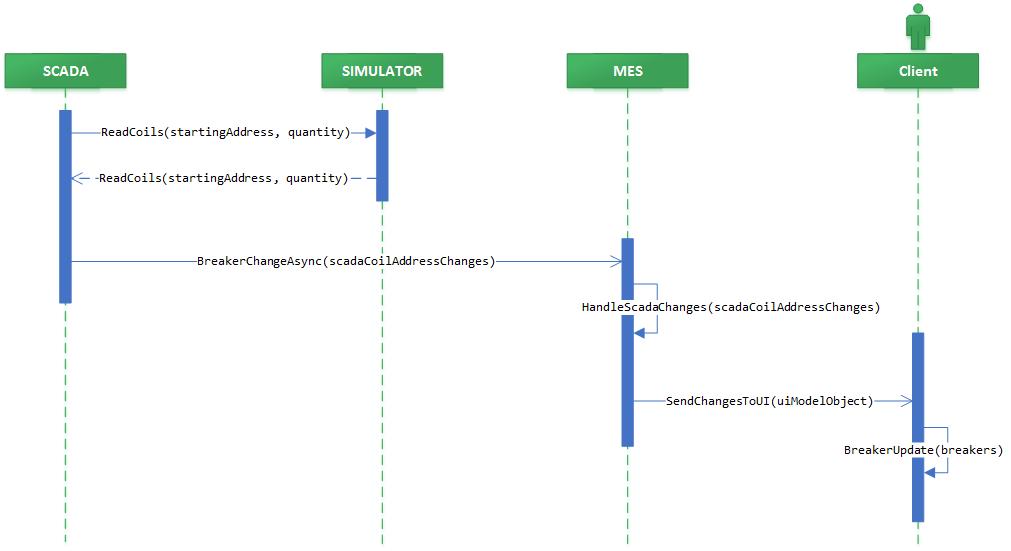
### Obrade akvizicije

Po prijemu podataka obrađenih u poglavlju 5.1.2 potrebno je da se ti podaci interpretiraju i da im se da neka semantika. Pošto je *SCADA* dizajnirana da nema nikakvu biznis logiku, odgovornost za tu logiku je prebačena na *MES*.

Obrada se sastoji iz četiri segmenta:

1. Čuvanje stanja pristiglih vrednosti
2. Upisivanje stanja u bazu podataka
3. Obaveštavanje klijenta o promenama

Ova akcija je opisana preko dijagrama sekvenci na slici 5.5.



Slika . - Obrada akvizicije

#### Čuvanje stanja pristiglih vrednosti

Od *SCADA* servisa stiže informacija na kom ID-u prekidača je koje novo stanje. Za dobijeni ID se iz *MES* modela uzima konkretan objekat prekidača u kom se nalaze podaci potrebni za obradu. U slučaju da prekidač ne postoji u modelu, baca se izuzetak i tako se zna da *SCADA* i *MES* nisu u konzistentnom stanju. Nakon dobijanja konkretnog prekidača njegovo stanje se ažurira. Treba primetiti da je ovo samo u radnom modelu i da još nije završilo u istorijskoj bazi podataka.

#### Upisivanje stanja u bazu podataka

Obrađeni podaci se moraju čuvati u istorijskoj bazi iz razloga što se istorija koristi za:

* Inicijalizaciju *MES* servisa
* Kreiranje izveštaja

Zbog performansi podaci ne završavaju odmah u bazi podataka, nego se upisuju u liste. Količina podataka u jednoj listi zavisi od pristiglih promena. Kad se obrade sve pristigle promene, vrši se upis u istorijsku bazu podataka i liste se nakon toga očiste da bi se nove promene mogle primenjivati.

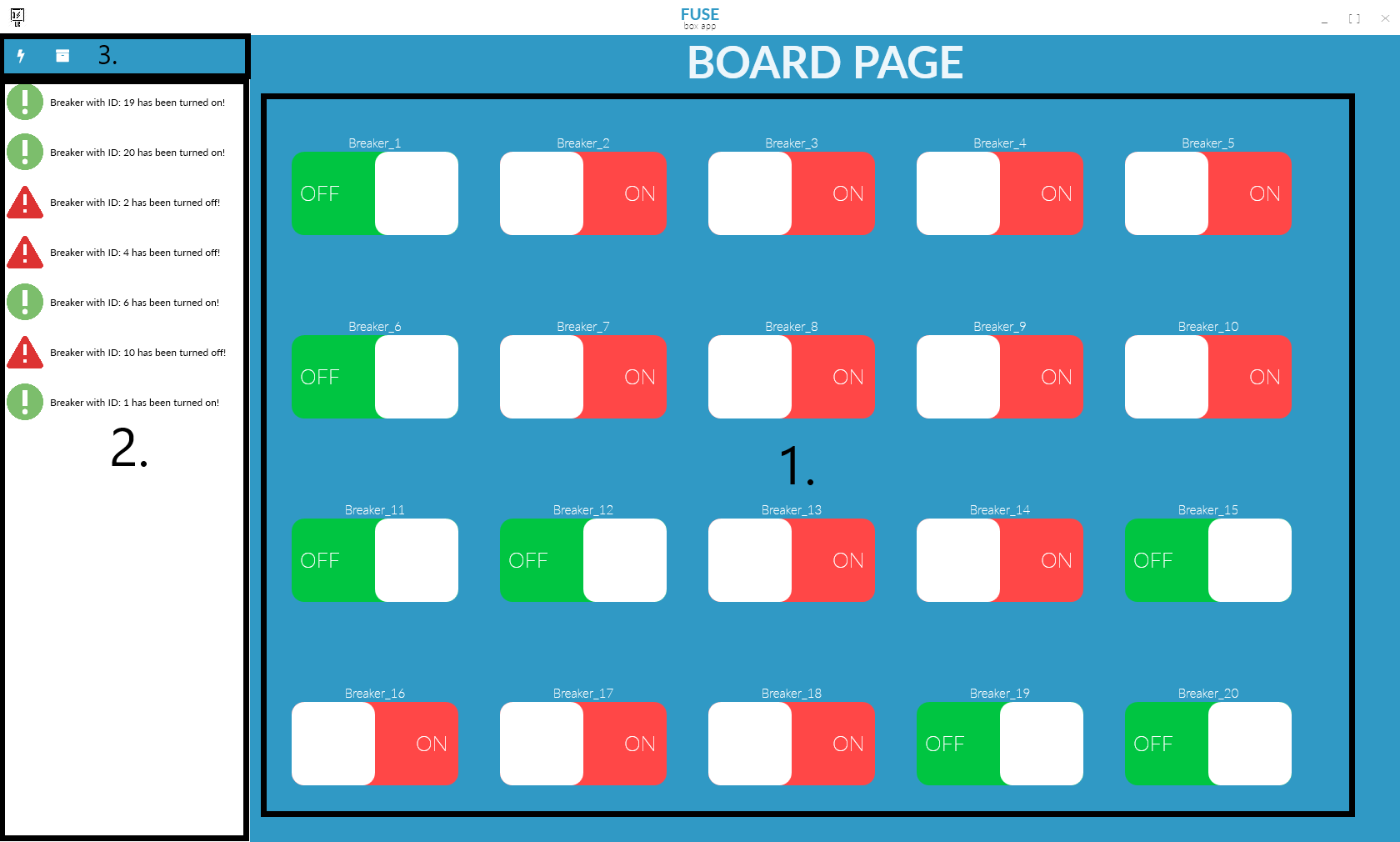
#### Obaveštavanje klijenta o promenama

Odmah nakon obrade podataka, promene se „pakuju“ u format podataka poznat klijentu tj u objekat **UIModelObject**. U taj objekat se pakuju svi afektovani prekidači, kao i generisani alarmi vezani za njih. Slanje je asinhrono.

## Klijentska aplikacija

Preko klijentske aplikacije, korisniku se omogućava nekoliko akcija kao što su: komandovanje prekidača, uvid u alarme i generisanje izveštaja. Na slici 5.6 možemo da vidimo izgled aplikacije, takođe možemo da primetimo nekoliko različitih prozora:

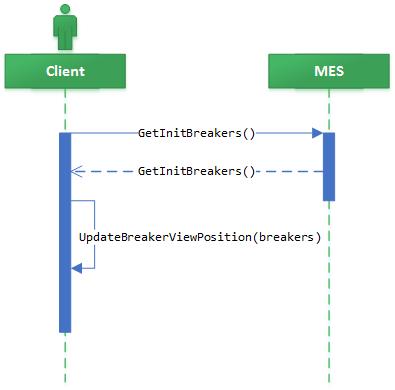
1. Glavni prozor
2. Lista sa alarmima
3. Tab pomoću kojeg menjamo glavni prozor. Dozvoljeni prozori su tabela sa prekidačima i prozor za generisanje izveštaja



Slika . - Klijentska aplikacija

### Inicijalno pokretanje

Nakon prvog pokretanja, klijentska aplikacija traži od *MES*-a svu postojeću opremu u sistemu sa njegovom trenutnom vrednošću. Što bi značilo da trebaju prekidači sa vrednostima da li je otvoren/zatvoren dati prekidač. Nakon prijema podataka, klijentska aplikacija vrši ažuriranje glavnog prozora. Prave se objekti u obliku dugmeta za svaki prekidač, gleda se koja mu je trenutna vrednost. Ako je prekidač zatvorenog kola, beli kvadratić unutar dugmeta će prekiti ispis **ON** i pozadina dugmeta će preći u zelenu. U slučaju da je prekidač otvorenog kola, beli kvadratić će ovaj put prekriti ispis **OFF** i pozadina dugmeta biće crvene boje. Na slici 5.7 možemo da vidimo dijagram sekvenci ove opisane akcije. Nakon ažuriranja glavnog prozora, može se nastaviti sa daljim akcijama koje klijentska aplikacija podržava.

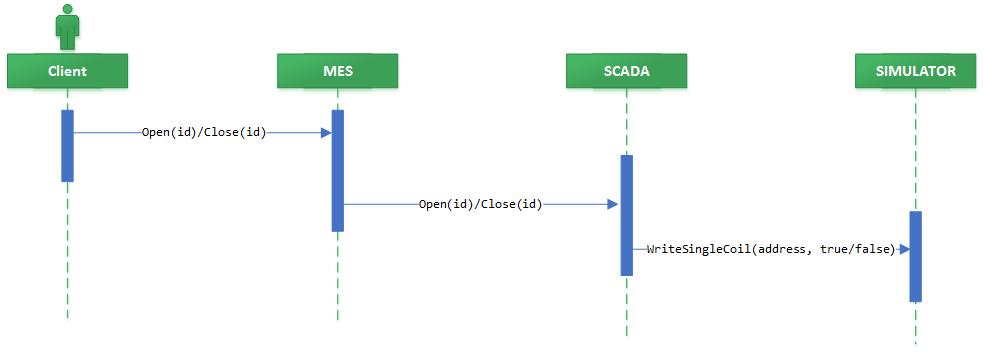


Slika . - Inicijalni start klijentske aplikacije

### Komandovanje

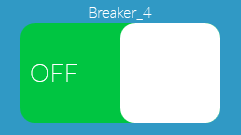
Komandovanje je akcija koja prolazi kroz sve komponente sistema. Akcija počinje na klijentskoj aplikaciji kliktanje miša na prekidač. Klikom na željeni prekidač šaljemo na *MES*, id prekidača koji želimo da komandujemo. Na osnovu njegovog trenutnog stanja tj. da li je otvoren ili zatvoren pozivamo odgovarajuće funkcije. Zatim *MES* informacije prosleđuje na *SCADA* servis. *SCADA* zatim na osnovu id-a iz modela traži adresu prekidača i tu adresu zajedno sa komandom šalje na simulator.

Ova akcija je opisana dijagramom sekvenci koja se nalazi na slici 5.8.

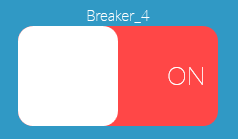


Slika . - Komandovanje preko klijentske aplikacije

Komandovanje prekidača na klijentskoj aplikaciji se vrši tako što se kursor miša postavi na dugme prekidača i levim klikom miša potvrdimo akciju. Da bi se akcija izvršila miš je potrebno postaviti ili na dugme ili na beli kvadratić unutar dugmeta. Ako je prekidač pre izvršenja akcije bio u zatvorenom kolu, on sada prelazi u otvoreno kolo, dugme prelazi iz zelene boje u crvenu, a beli kvadratić sa desne strane prelazi na levu. Ako je prekidač pre izvršenja akcije bio u otvorenom kolu, postupak se ponavlja ali u obratnom smeru. Na slici 5.9 i slici 5.10 možemo da vidimo izgled dugmati sa klijentske aplikacije kada su ona u zatvorenom i otvorenom kolu.



Slika . - Zatvoren prekidač



Slika . - Otvoren prekidač

### Alarmi

Alarmi u implementaciji aplikacije predstavljaju odziv na događaje u sistemu. Pod tim podrazumevamo na svaku akciju komandovanja prekidača, bilo to sa klijentske aplikacije ili sa simulatora, se generiše novi alarm. Time želimo da obavestimo krajnjeg korisnika o promenama u sistemu. Ovo ima posebnu važnost kada je reč o komandovanju sa simulatora, jer preko alarma možemo korisnika obavestiti o promenama u sistemu.

Objekat alarm sadrži nekoliko polja i atributa koja ga opisuju, te vrednosti su opisane u tabeli 5.3.

|  |  |
| --- | --- |
| **Id *(int)*** | **Označava jedinstveni identifikacioni broj alarma u sistemu** |
| **BreakerId (int)** | **Označava jedinstveni identifikacioni broj prekidača za koji je dati alarm vezan** |
| **Timestamp (datetime2)** | **Označava vreme kada se desila akcija koja je pokrenula generisanje alarma** |
| **Message (nvarchar(max))** | **Označava dodatni opis alarma** |

Tabela . - Prikaz redova tabele alarma

Na klijentskoj aplikaciji postoji poseban tab u kome su alarmi smešteni i vidljivi korisniku. Taj odsek je objašnjen u poglavlju 5.4. Izgled alarma može imati dva slučaja, kada je prekidač prešao u otvoreno kolo tj. kada je isključen () i kada je u zatvorenom kolu tj. kada je uključen (). Izgled alarma se sastoji od ikonice koja zavisi od statusa prekidača. Ako je prekidač isključen ikonica je u obliku trougla, crvene boje je i u sredi ima znak upitnika, u slučaju da je prekidač uključen ikonica je u tom slučaju kružnog oblika, zelene boje i sa znakom upitnika u sredi. Pored ikonice se nalaz i dodatni opis alarma. Poruka u kojoj piše u kom je trenutnom statusu prekidač.

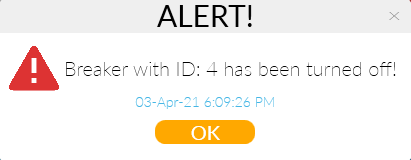


Slika . - Izgled alarma kada je prekidač isključen

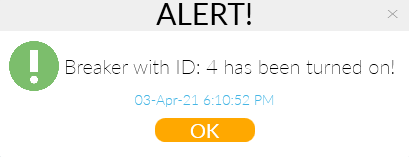


Slika . - Izgled alarma kada je prekidač uključen

Ako želimo neke dodatne informacije koje su vezane za alarme, onda moramo postati kursor miša na alarm i akciju potvrditi levim klikom miša. Odgovor te akcije javlja se u obliku novog mini prozora koji zamrzava glavni prozor sve dok se taj mini prozor ne ugasi. Mini prozor prati neki šablonski izgled glavnog prozora i u sebi sadrži sve informacije kao i alarm sa glavnog prozora. Pored toga kao dodatna informacija je vreme generisanja datog alarma. Vreme je opisano u formatu: *Dan – Mesec – Godina (Sati:Minute:Sekunde)*. Izlaz iz ovog prozora i povratak na glavni prozor je moguć levim klikom miša na dugme **OK** ili na dugme **X**. Izgled ovog prozora, možemo da vidimo na



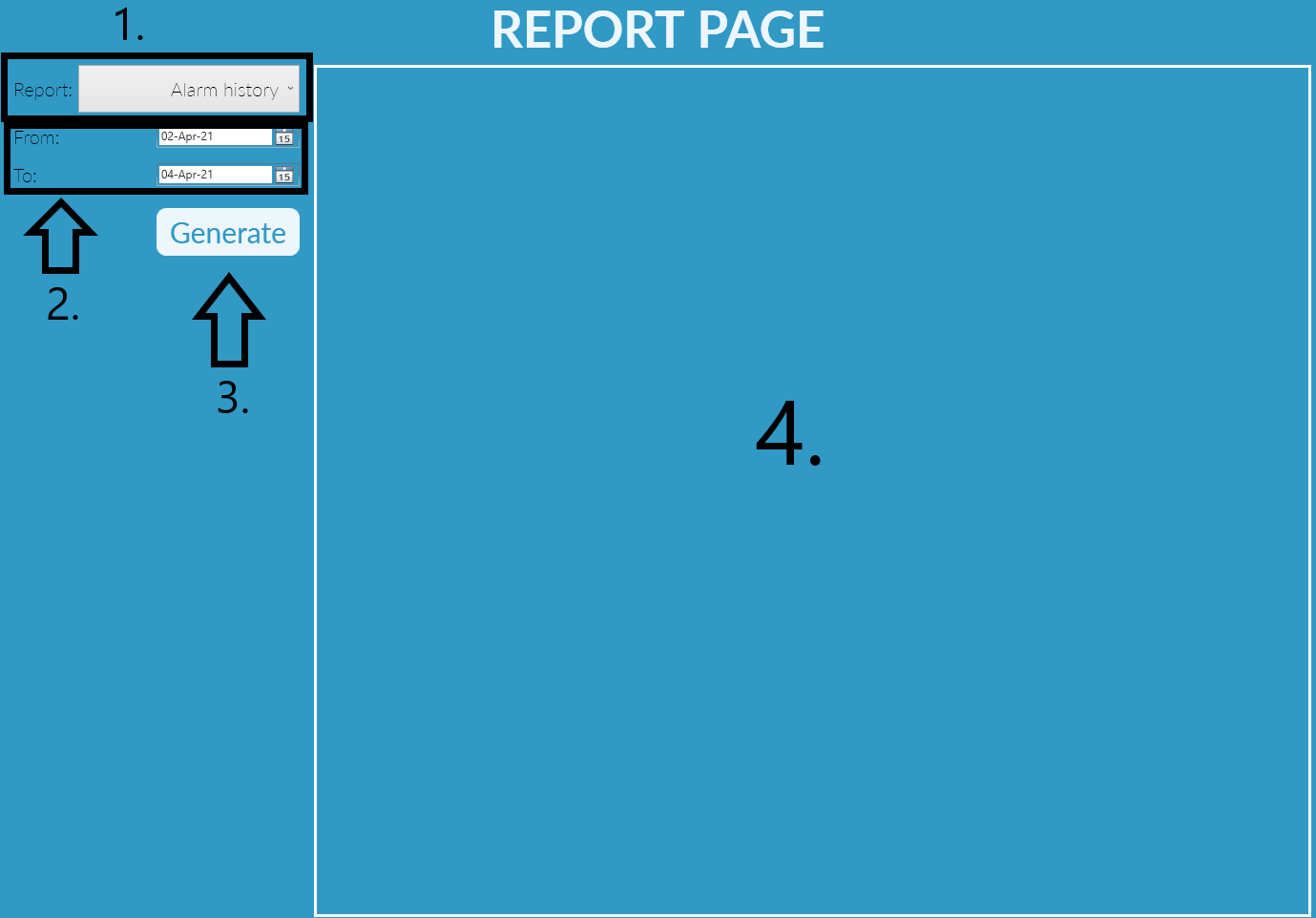
Slika . - Izgled prozora alarma sa dodatnim informacijama kada je prekidač isključen



Slika . - Izgled prozora alarma sa dodatnim informacijama kada je prekidač uključen

### Izveštaji

Report ili izveštaj je dokument koji predstavlja informacije koje su organizovane u formatu za određenu svrhu. Iako se sažeci izveštaja mogu dostavljati usmeno, potpuni izveštaji su gotovo uvek u obliku pisanih dokumenata. Do prozora sa izveštajima se dolazi levim klikom miša na ikonicu kutije koja se nalazi u levom gornjem uglu glavne stranice. Izgled glavne stranice i ikonice kutije za izveštaje je objašnjeno u poglavlju 5.4. Klikom se prozor sa prekidačima nestaje i pojavljuje se prozor za izveštaje. Izgled ovog prozora je opisan na slici 5.15.



Slika . - Izgled prozora sa izveštajima

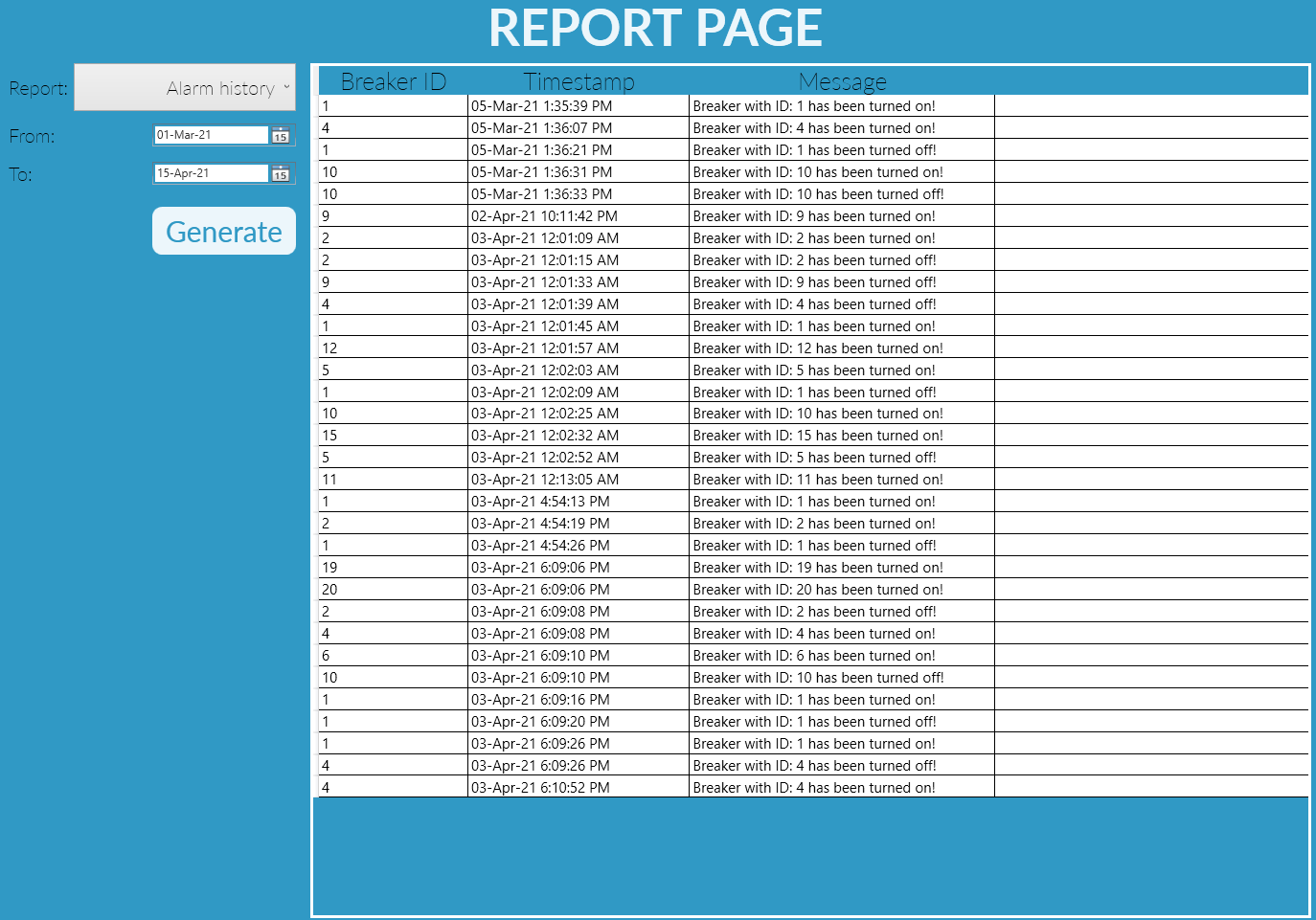
Na slici vidimo nekoliko bitnih elemenata koji pomažu da lakše i što preciznije generišemo željeni izveštaj:

1. Combox sa nazivima izveštaja koje sistem podržava
   1. Alarm History – istorija alarma
   2. Current Equipment – svi trenutni prekidači u sistemu
2. Birači datuma za pomoću kojih se određuje vremenski opseg generisanja alarma
3. Dugme pomoću kojeg potvrđujemo opcije
4. Glavni prozor sa podacima izveštaja

Izveštaj istorija alarma (*Alarm History*) ispisuje sledeće kolone:

* ID prekidača za koji je alarm vezan
* Vreme generisanja alarma
* Dodatni opis

Na slici 5.16 se nalazi izgled glavnog prozora sa datim izveštajem.

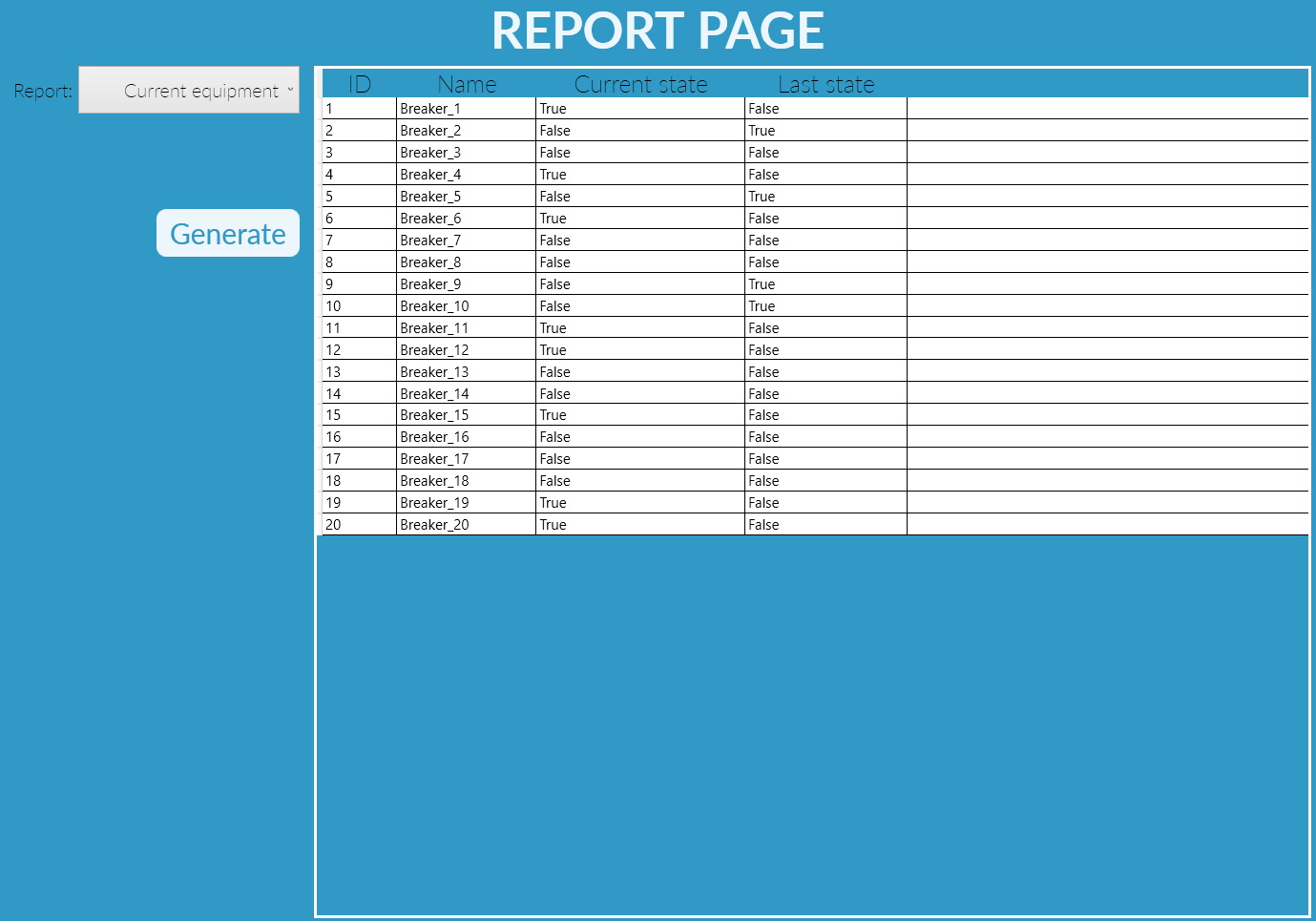


Slika . - Izveštaj istorije alarma

Izveštaj svih trenutnih prekidača u sistemu (*Current Equipment*) ispisuje sledeće kolone:

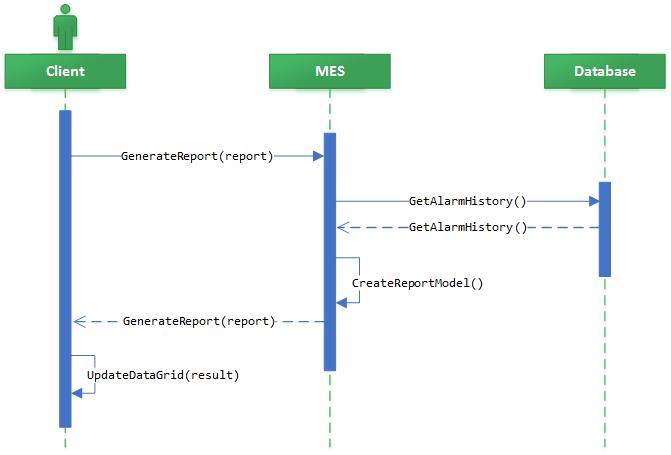
* ID prekidača
* Naziv prekidača
* Trenutno stanje prekidača
* Prethodno stanje prekidača

Na slici 5.17 se nalazi izgled glavnog prozora sa datim izveštajem.



Slika . - Izveštaj svih trenutnih prekidača u sistemu

Način na koji se izveštaji generišu jeste taj da korisnik preko klijentske aplikacije unese odgovarajuće uslove za generisanje izveštaja. Zatim ti uslovi se šalju na *MES* servis koji zatim na osnovu željenog izveštaja, izvlači iz baze podataka odgovarajuće podatke. Zatim te podatke “pakuje” u **ReportModel** objekat i vraća ih nazad klijentskoj aplikaciji. Kad ti podaci pristignu na klijentsku aplikaciju vrši se ispis na glavni prozor. Ilustraciju opisane akcije možemo da vidimo na slici 5.18.



Slika . - Generisanje izveštaja

# ZAKLJUČAK

# LITERATURA

[1] Bianca Scholten, *Road to Integration*, ISBN-13: 978-0979234385

[2] TechTarget, *Definition, ANSI/ISA-95*, datum pristupa: 25/03/2021 <<https://searcherp.techtarget.com/definition/ANSI-ISA-95>>

[3] Siemens, *ISA 95 Framework & Layers*, datum pristupa: 26/03/2021 <<https://www.plm.automation.siemens.com/global/en/our-story/glossary/isa-95-framework-and-layers/53244>>

[4] Intraratio, *3 Key Differences Between ERP and MES Systems*datum pristupa: 26/03/2021 <<https://intraratio.com/post/key-differences-between-erp-and-mes>>

[5] ISA-95, *Overview of advantages*,datum pristupa: 27/03/2021 <<https://isa-95.com/advantages>>

[6] Guru99, *What is Microsoft .Net Framework?*, datum pristupa: 27/03/2021 <<https://www.guru99.com/net-framework.html>>

[7] GeeksforGeeks, *Introduction to Visual Studio*, datum pristupa: 27/03/2021 <<https://www.geeksforgeeks.org/introduction-to-visual-studio>>

[8] Jon Skeet, *C# in Depth, 3rd Edition*, ISBN-13: 978-1617291340

[9] WPF Tutorial, *What is WPF?*, datum pristupa: 27/03/2021 <<https://www.wpf-tutorial.com/aboutwpf/what-is-wpf>>

[10] Wintellect, *Model-View-ViewModel (MVVM) Explained*, datum pristupa: 27/03/2021 <<https://www.wintellect.com/model-view-viewmodel-mvvm-explained>>

[11] Microsoft, *What Is Windows Communication Foundation*, datum pristupa: 28/03/2021 <<https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf>>

[12] Wikipedia, *Microsoft SQL Server*, datum pristupa: 28/03/2021 <<https://wikipedia.org/wiki/Microsoft_SQL_Server>>

[13] Entity Framework Tutorial, *What is Entity Framework?*, datum pristupa: 28/03/2021 <<https://www.entityframeworktutorial.net/what-is-entityframework.aspx>>

[14] Wikipedia, *Modbus*, datum pristupa: 28/03/2021 <<https://wikipedia.org/wiki/Modbus>>

[15] Easy Modbus TCP, *Modbus Server Simulator*, datum pristupa: 28/03/2021 <<http://easymodbustcp.net/modbus-server-simulator>>

[16] Source Making, *Design Patterns*, datum pristupa: 28/03/2021 <<https://sourcemaking.com/design_patterns>>

[17] Source Making, *Command Design Pattern*, datum pristupa: 28/03/2021 <<https://sourcemaking.com/design_patterns/command>>

[18] Source Making, *Singleton Design Pattern*, datum pristupa: 28/03/2021 <<https://sourcemaking.com/design_patterns/singleton>>

# LISTA KORIŠĆENIH SKRAĆENICA

|  |  |
| --- | --- |
| Skraćenica | Pun naziv |
| ISA  ERP  SCADA  PLC  HMI  MES  ORM  WPF  WCF  SQL  HP  CLI  CLR  CIL  API  XML  HTML  CSS  XAML  UI  MVVM  UML  TCP  UDP  RTU | ***Instrumentation, Systems and Automation Society***  ***Enterprise Resource Planning***  ***Supervisory Control And Data Acquisition***  ***Programmable Logic Controller***  ***Human-Machine Interface***  ***Manufacturing Execution System***  ***Object–relational mapping***  ***Windows Presentation Foundation***  ***Windows Communication Foundation***  ***Structured Query Language***  ***Hewlett-Packard***  ***Common Language Infrastructure***  ***Common Language Runtime***  ***Common Intermediate Language***  ***Application Programming Interface***  ***Extensible Markup Language***  ***Hypertext Markup Language***  ***Cascading Style Sheets***  ***Extensible Application Markup Language***  ***User Interface***  ***Model-View-ViewModel***  ***Unified Modeling Language***  ***Transmission Control Protocol***  ***User Datagram Protocol***  ***Remote Terminal Unit*** |

# Podaci o kandidatu

Kandidat Nenad Zelenović je rođen 01.04.1995. godine u Somboru. Osnovnu školu “Nikola Tesla” u Bačkom Brestovcu završio je 2010. godine. Završio je srednju ekonomsku školu u Somboru 2014. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisao 2014. godine, smer Elektroenergetski Softverski Inženjering. 2019. godine završava osnovne studije i upisuje master akademske studije na Fakultetu Tehničkih Nauka, smer Primenjeno Softversko Inženjerstvo. Ispunio je sve obaveze i položio sve ispite predviđene studijskim programom.