

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

MLADEN ŠVERKO

RAZVOJ SCADA PODSUSTAVA ZA RANU DETEKCIJU SKRITIČNE SITUACIJE

Završni rad

Pula, srpanj, 2019.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

MLADEN ŠVERKO

RAZVOJ SCADA PODSUSTAVA ZA RANU DETEKCIJU SKRITIČNE SITUACIJE

Završni rad

JMBAG: 0016076929, izvanredni student

Studijski smjer: Informatika

Predmet: Programsko inženjerstvo

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc.dr.sc. Tihomir Orehovački

Pula, srpanj, 2019.



IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mogega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, _____, _____ godine



IZJAVA
o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom

_____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

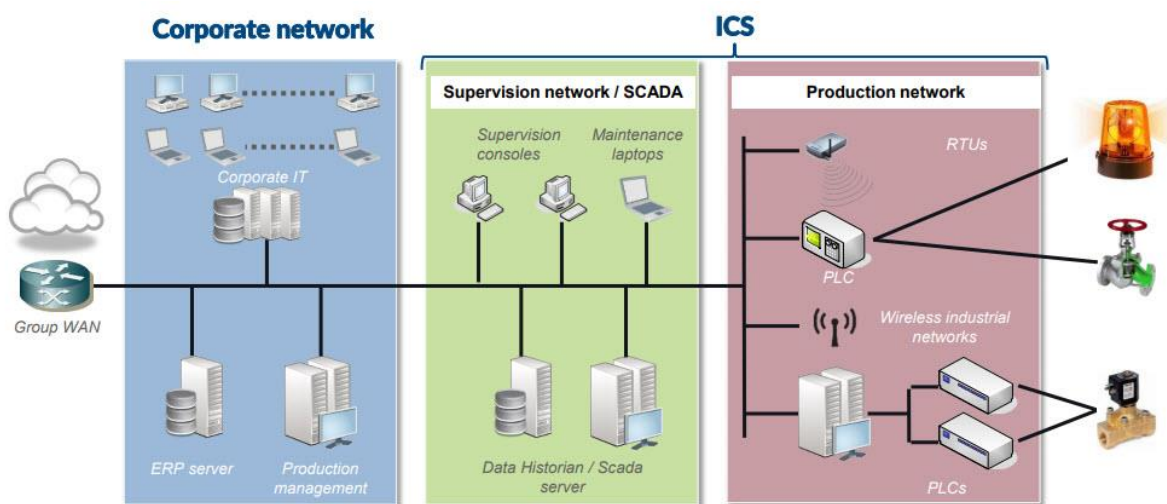
Sadržaj

Sadržaj	5
1. Uvod	6
2. Razrada funkcionalnosti	7
2.1 Osnovni zahtjevi funkcionalnosti	8
2.2 Use-case dijagram	9
2.3 Dijagram slijeda (<i>sequence diagram</i>)	10
2.4 Klasni dijagram	13
2.5 Dijagram tijeka podataka (<i>data flow</i>)	15
2.6 Model kritičnog događaja	16
3. Implementacija	17
3.1 Zahtjevi u odabiru tehnologije	17
3.2 Razvojne tehnologije – I/O server	18
3.3 Razvojne tehnologije – Web aplikacija	19
3.4 Arhitektura web aplikacije	20
3.5 Poziv metode za detekciju kritičnog događaja	22
3.6 Algoritam detekcije kritičnog događaja	22
4. Korisničke upute	25
4.1 Prijava (Logon)	25
4.2 Registriranje kritične situacije	25
4.3 Unos osnovnih podataka	27
4.4 Unos tagova (signala)	27
4.5 Spremanje modela kritične situacije	29
4.6 Pregled modela pohranjenih kritičnih situacija	29
4.7 Grafički prikaz aktualnih vrijednosti	30
4.8 Upozorenje na kritični događaj	31
5. Zaključak	32
6. Popis slika	33
7. Popis tablica	33
8. Literatura	33

1. Uvod

SCADA (eng. *Supervisory Control And Data Acquisition System*) je relativno širok pojam koji dolazi iz industrijskog okruženja i dio je industrijskih kontrolnih sustava (eng. *ICS – Industrial Control System*). Premda su same granice SCADA sustava donekle nedorečene, te često dolazi do preklapanja sa sustavima izravne kontrole industrijskih procesa na jednoj, te informacijskim sustavima korporativne mreže na drugoj strani, može se reći da u tom smislu SCADA sustav postoji kao svojevrsni sloj koji povezuje ta dva sustava, te nekim svojim komponentama ulazi u oba. Kako će SCADA sustav biti definiran često ovisi i od pojedine situacije, odnosno pojedinačnog slučaja implementacije komponenti sustava. No, bez obzira na konkretan slučaj primjene, primarni zadatak SCADA sustava je kontrola procesa i prikupljanje podataka (arhiviranje i prosljeđivanje). S obzirom na današnju praksu, to se uglavnom postiže uz visok stupanj interoperabilnosti, te mogućnošću integracije prema podsustavima u okruženju.

Na donjoj slici prikazan je jedan primjer SCADA sustava u jednoj od mogućih izvedbi. U danom primjeru, SCADA sustav je prikazan kao dio ICS-a, te sadrži komponente za arhiviranje podataka i nadzor procesa u server-klijent strukturi sa komunikacijom prema korporativnoj mreži.



Slika 1: Industrijski kontrolni sustav

(Izvor: www.blackhat.com/docs/eu-14/materials/eu-14-Soullie-Industrial-Control-Systems-Pentesting-PLCs-101.)

Tako razgranat sustav vrlo je kompleksan u svakom svom segmentu i djelo je vodećih svjetskih proizvođača industrijske te informatičke opreme i softvera kao što su Siemens, Schneider, General Electric, Fujicu i sl. Navedene kompanije razvile su kompleksna razvojna sučelja za razvoj SCADA aplikacija i brojne komunikacijske servere posebno prilagođene industrijskim protokolima.

Međutim, takvi sustavi ipak imaju i nedostataka. Jedan od njih je tromost, bilo da je riječ o implementaciji novih tehnologija, fleksibilnosti u odnosu prema korisniku ili otklanjanju grešaka i sl. Još jedan česti kamen spoticanja je pretjerana razgranatost sustava što dovodi do teškog snalaženja u odabiru rješenja, i naravno, cijena.

Premda su globalne kompanije svakako prvi izbor za razvoj kompleksnog i robusnog industrijskog informacijskog sustava kao što je prikazan na slici, također postoji potreba za manjim i fleksibilnijim sustavom, ili samo nekom od komponenti SCADA sustava koja će biti posebno prilagođena određenim zahtjevima klijenta te cjenovno dostupnija, a istovremeno da takav sustav bude razvijen sa minimalnim problemima kompatibilnosti u odnosu na operacijske sustave. Namjera predmetnog projekta je razviti jedan takav podsustav kojemu će primarna zadaća biti pravovremena detekcija kritičnog događaja, te nastavno, generiranje upozorenja u trenutku ispunjenja zadanih parametara koji definiraju model kritičnog događaja. Slični podsustavi već postoje na tržištu, ali su to u principu vrlo skupa softverska rješenja koja se oslanjaju na razne algoritme strojnog učenja i analizu iznimno velike količine podataka, najčešće pohranjenih u oblaku. Za razliku od toga, ovo softversko rješenje ima za cilj postići zadovoljavajuću učinkovitost u ranom otkrivanju kritične situacije uz vrlo malu količinu obrađenih podataka te uz uključivanje korisnika (HMI operatera) u proces odabira ključnih parametara (varijabli) za definiranje kritičnog događaja. Takav podsustav biti će lako primjenjiv na različite industrijske procese, a uz odabir odgovarajućih mrežnih tehnologija, lako će koegzistirati uz postojeća SCADA rješenja.

2. Razrada funkcionalnosti

Razrada funkcionalnosti projekta biti će u nastavku teksta pojašnjena uz temeljne zahtjeve, te putem korištenih UML dijagrama kojima će se pobliže objasniti interakcija korisnika i aplikacije(eng. *use-case diagram*), funkcije aplikacije(eng. *sequence diagram*), objekti i veze među njima(eng. *class diagram*).

2.1 Osnovni zahtjevi funkcionalnosti

Kako je već iz uvoda razabire, cilj je ostvariti SCADA podsustav koji će ispuniti osnovne uvjete komunikacije (PLC kontroler - I/O server - sučelje) i grafičko sučelje koje će prikazivati promjene u realnom vremenu (generirana upozorenja i grafički prikaz kritične situacije). Sustav na strani korisnika mora biti što je moguće manje ovisan o nadogradnjama i novim verzijama operacijskog sustava. Inicijalno, arhiviranje podataka nije u prvom planu, kao ni propusnost, odnosno znatna količina podataka obrađena (dostupna krajnjem korisniku) u realnom vremenu. Konkretno, SCADA podsustav mora zadovoljavati slijedeće karakteristike:

- Prilagodljivost - univerzalno primjenjiv za opću industriju (eng. *general industry automation*).

Jedan od ključnih elemenata za postizanje funkcionalnosti sustava jeste definiranje parametara (modela) kritičnog događaja. Uključivanjem operatera (koji nadgleda industrijski proces, te je sa istim upoznat) u taj segment, postiže se fleksibilnost definiranja raznih kritičnih situacija koje mogu biti specifične za različite industrijske procese, ali uz univerzalan princip definiranja.

- Proširivost

Budući prethodno spomenuta univerzalna primjenjivost podrazumijeva komunikaciju sa različitim PLC kontrolerima na razini proizvodne mreže (eng. *production network*), time je neophodno ostvariti mogućnost proširenja na različite komunikacijske I/O servere. Inicijalno, cilj je ostvariti povezivost sa barem jednim standardnim kontrolerom koji je opće prisutan na tržištu, i u širokoj je primjeni (prim. Siemens S7-400, S7-300, S7-200).

- Komunikacija u realnom vremenu.

Podrazumijeva se da je za ranu detekciju kritične situacije, te generiranje pripadnog upozorenja nužno da se cijeli proces čitanja iz PLC-a, upisa u SQL bazu, te usporedba pohranjenih vrijednosti sa aktualnima odvija u vremenskim ciklusima koji omogućavaju pravovremenu detekciju i ostavljaju vremena za odgovarajuću reakciju operatera. Sukladno tome, cilj je postići da se navedeni ciklus odvija u vremenu do najviše dvije sekunde.

- Dostupnost

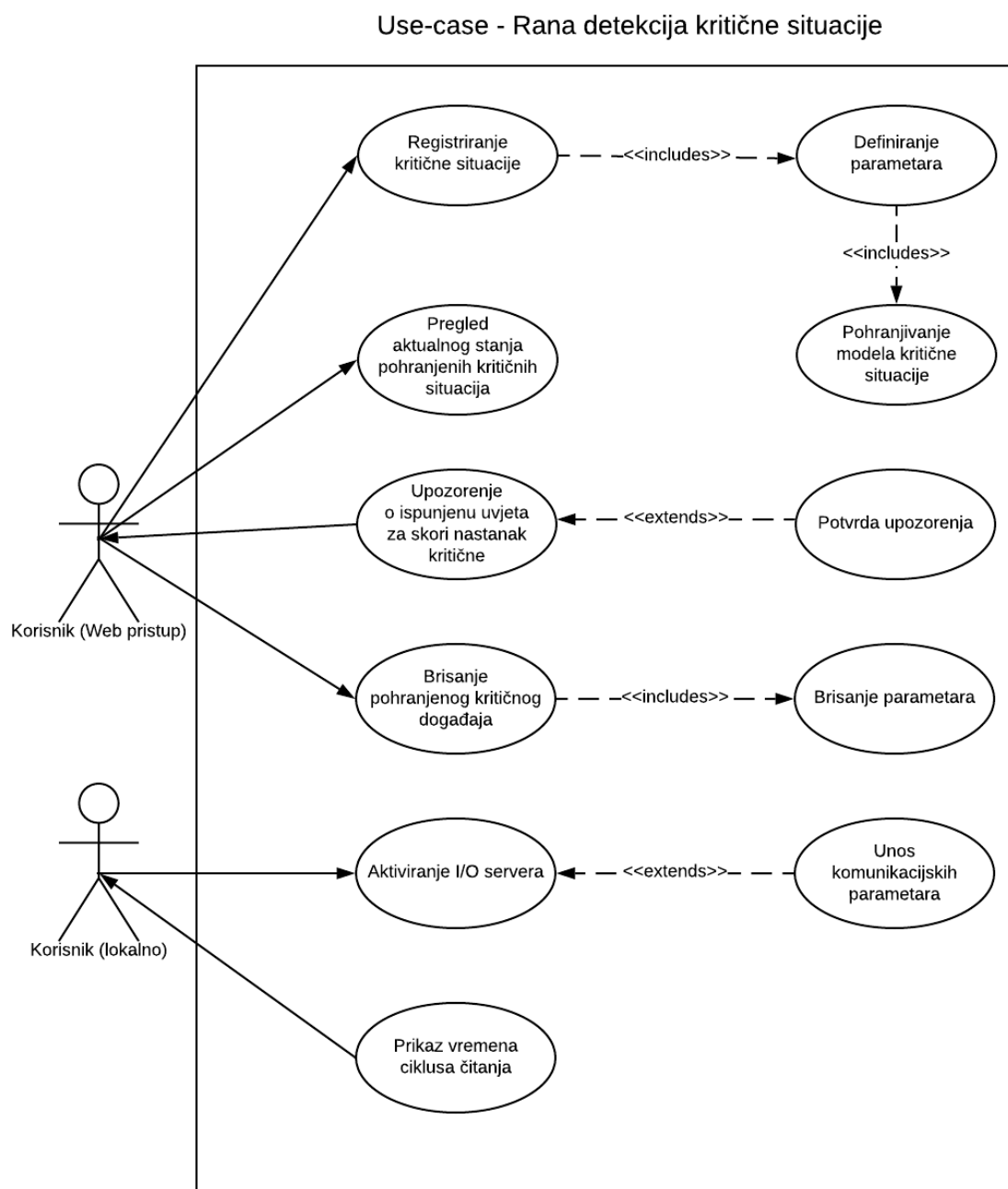
Sustav ne smije biti isključivo dostupan unutar kontrolne sobe, već mora biti upotrebljiv na razini industrijskog kontrolnog sustava (ICS).

- Nadogradnja i modularnost

Budući će predmetni podsustav, sukladno dosad postavljenim zahtjevima, samostalno komunicirati, tj. čitati podatke sa PLC-a, time se otvaraju mogućnosti za dodatne funkcionalnosti poput generiranja raznih izvješća o proizvodnji (eng. *report generator*) koja također mogu biti dostupna na razini ICS-a. Kako slična rješenja mogu funkcionirati kao nadogradnja (podsustavi) na postojeće SCADA sustave, time je poželjno ostaviti mogućnost nadogradnje istih u obliku zasebnih modula.

2.2 Use-case dijagram

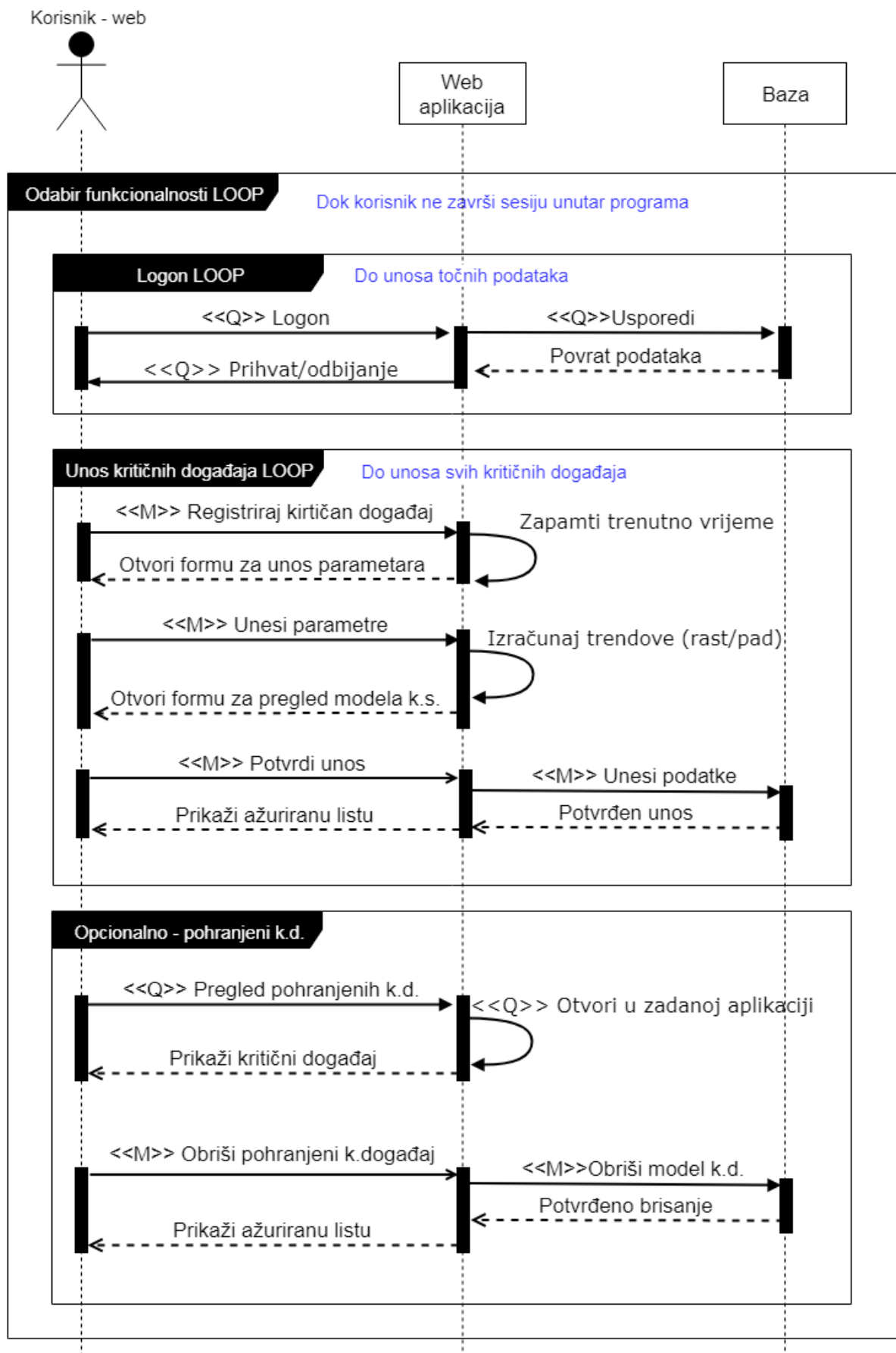
Use-case dijagram je ponašajni UML dijagram koji ima za cilj prikazati ponašanje sustava iz perspektive korisnika, odnosno, interakciju korisnika i sustava. Budući se predmetni SCADA podsustav kao cjelokupno softversko rješenje sastoji od dva zasebna modula (web aplikacija i I/O server), time i dolje prikazan dijagram prikazuje interakciju sa sustavom iz perspektive korisnika koji koristi web-aplikaciju, te lokalno, I/O server. Korisnik (putem web aplikacije) evidentira kritični događaj koji se dogodio, parametre istog, te posljedične aktivnosti unutar sustava, kao i dogovor sustava na moguću (evidentiranu i pohranjenu) kritičnu situaciju. Korisnik koji djelu sustava (I/O server) pristupa lokalno, pokreće komunikaciju prema PLC-u, odnosno omogućava čitanje podataka sa istog. Navedenu aktivnost (lokalni korisnik) potrebno je učiniti samo inicijalno, pri pokretanju računala koje u konkretnom slučaju jeste, ali ne mora biti ono na kojem se nalazi i Web server.



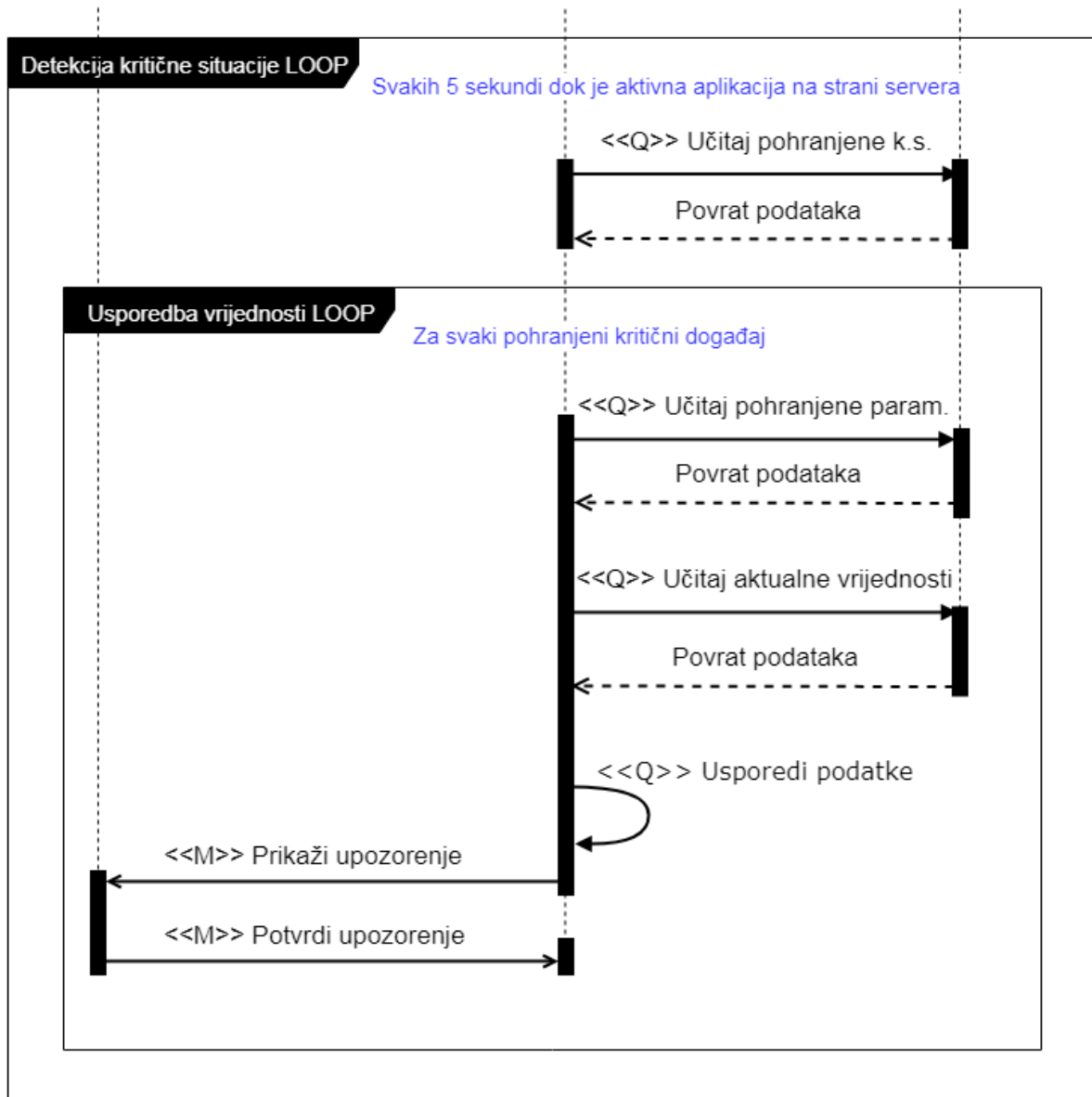
Slika 2: Use-case dijagram

2.3 Dijagram slijeda (*sequence diagram*)

Na donjim dijagramima slijeda, prikazano je međudjelovanje sudionika (elemenata sustava) u vremenskom redoslijedu aktivnosti, odnosno, vremenski slijed slanja poruka između životnih tokova (eng. *lifecycle*). Budući se predmetni SCADA podsustav, kako je već rečeno, sastoji od dva odvojena programa koja međusobno nemaju izravnu interakciju, time postoje i dva odvojena dijagrama slijeda.

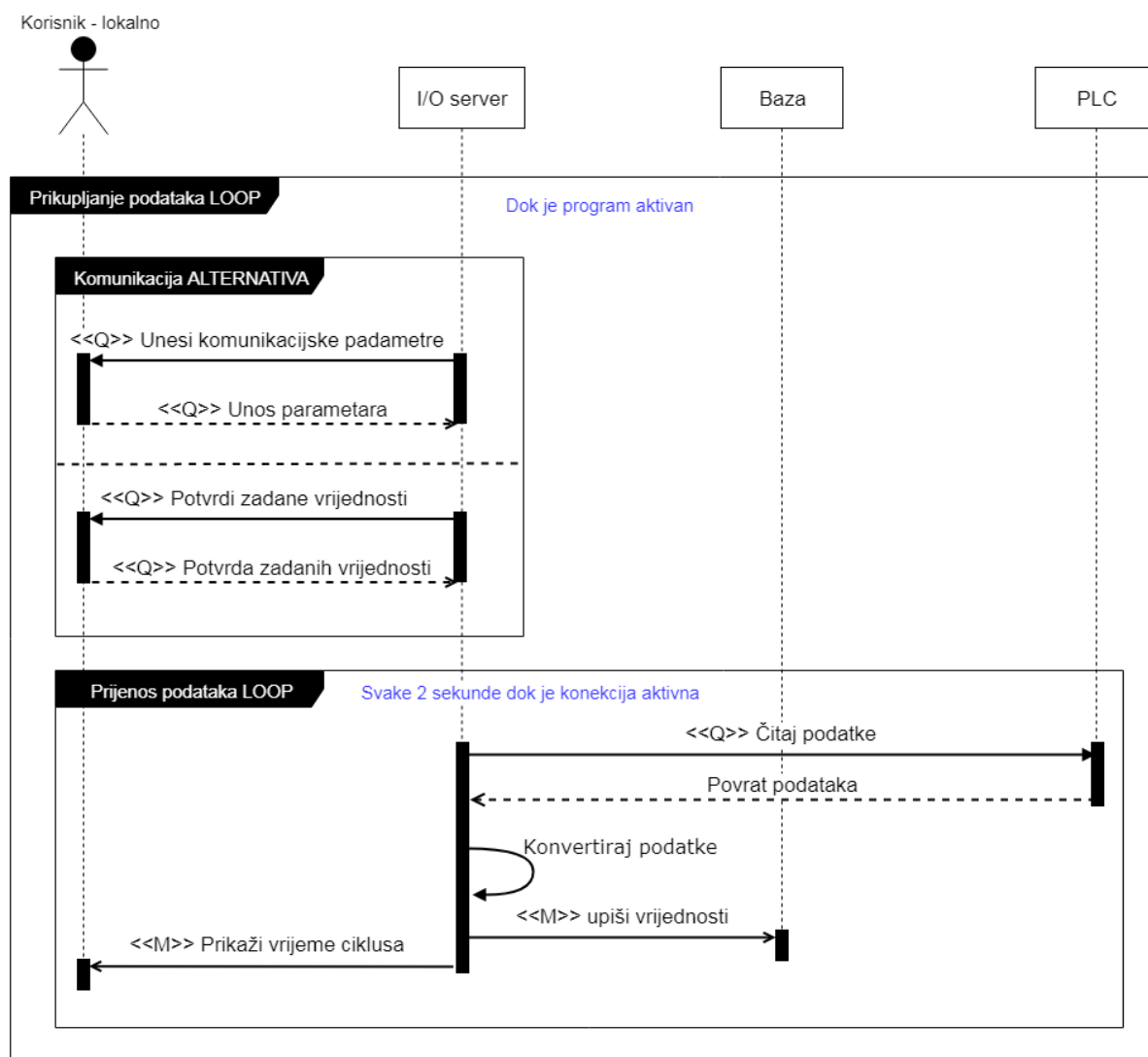


Slika 3: Sequence dijagram – Web aplikacija: dio 1



Slika 4: Sequence dijagram – Web aplikacija: dio 2

Na gornjem dijagramu vidljiv je slijed aktivnosti u interakciji sustava i korisnika web aplikacije. Time je prikazano na koji se način, putem web klijenta (*front end*) registrira uočena kritična situacija, te daljnji slijed koji dovodi do pohranjivanja modela iste, te u konačnici generiranja upozorenja ako u cikličkoj provjeri postoji poklapanje pohranjenih vrijednosti sa aktualnima.



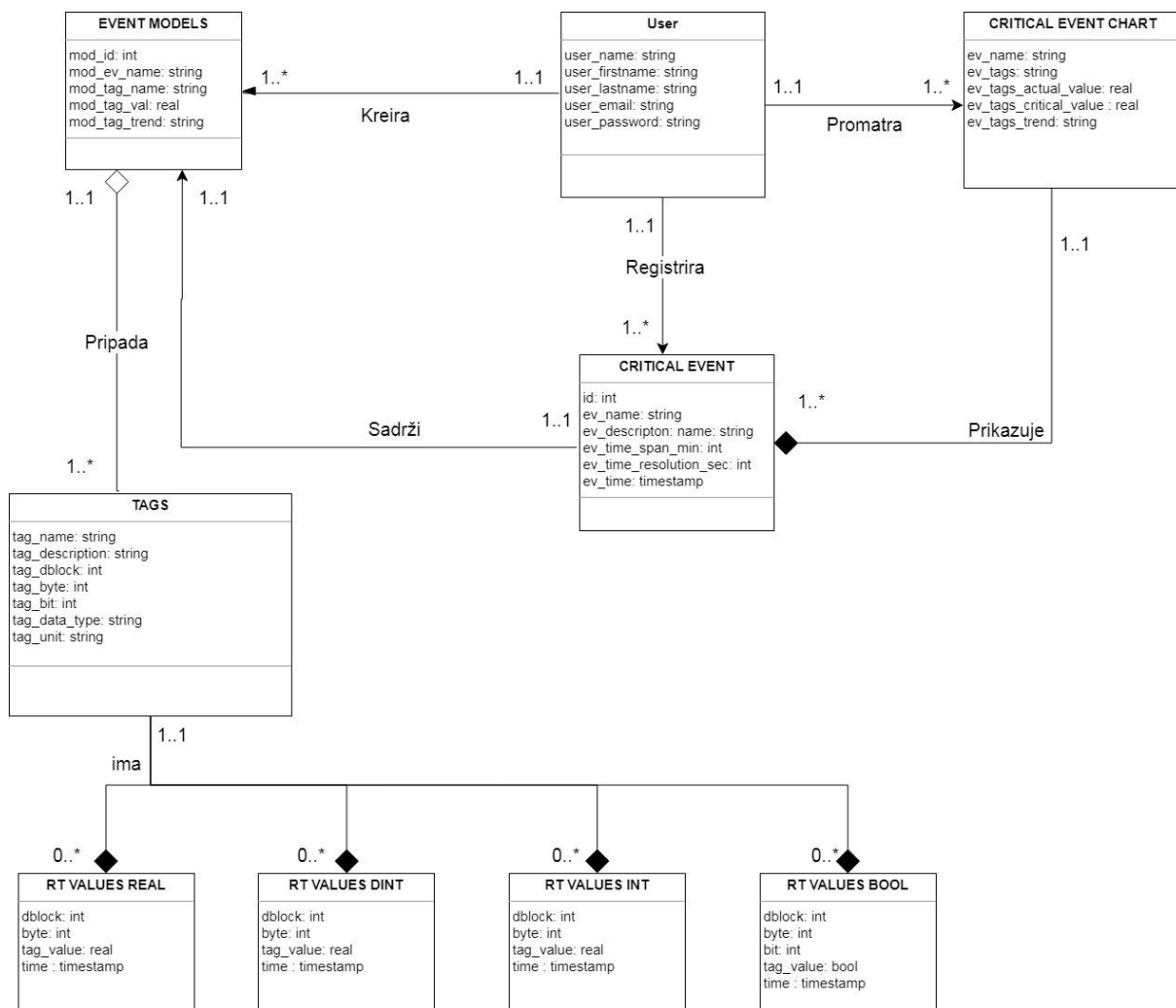
Slika 5: Sequence dijagram – I/O server (konzolna aplikacija)

Gornji dijagram prikazuje slijed aktivnosti u čitanju vrijednosti iz PLC-a, te pohranjivanje istih u SQL bazu podataka. Vidljivo je kako jednom pokrenuta, aplikacija ciklički čita podatke, te ih nakon konverzije pohranjuje u bazu. Nakon svakog ciklusa, koji se ponavlja u periodima od dvije sekunde, prikazuje se vrijeme trajanja ciklusa čitanja i upisa u bazu (redovito kraće od dvije sekunde).

2.4 Klasni dijagram

Donjim dijagramom opisana je logička struktura sustava sa odnosima između klasa. Iz dijagrama je uočljivo kako su sa aspekta veza između klasa ključne dvije skupine istih. Veze između korisnika(user), kritične situacije(critical event), njenog modela (event model), te aktualnog prikaza pohranjene kritične situacije (critical event chart). Navedena skupina veza sa pripadnim klasama realizira temeljnu funkciju sustava registriranja i pohranjivanja kritične situacije (događaja) te pregleda istog u realnom

vremenu. Drugu ključnu skupinu predstavljaju veze između klasa modela kritične situacije (event model), tagova i njima pripadnih vrijednosti (rt values real, rt values dint, rt values int, rt values bool). Navedena skupina veza omogućuje definiranje kritične situacije i praćenje pohranjenog modela iste u realnom vremenu.



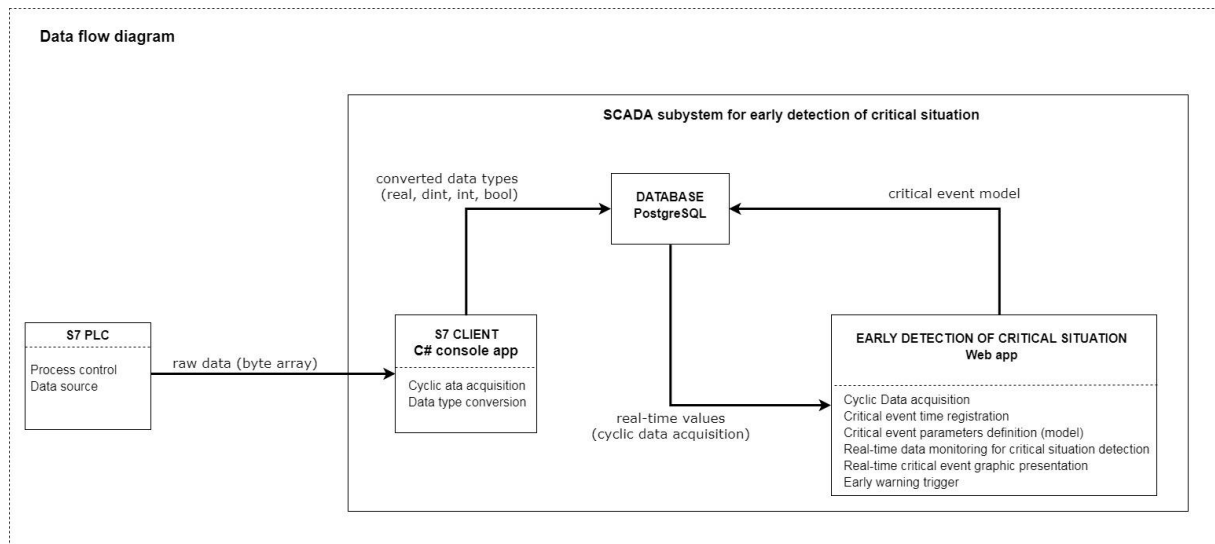
Slika 6: Klasni dijagram

Na gore prikazanom dijagramu vrijedi izdvojiti nekoliko uočljivih tipova veza:

- agregacija između klasa „event models“ i „tags“ čime je jasno vidljivo kako je struktura taga sastavni dio modela kritičnog događaja
- kompozicija između tagova i njihovih vrijednosti, iz čega je vrijednost zabilježena u realnom vremenu sastavni dio strukture taga, a time dalje i modela kritičnog događaja.
- kompozicija između klasa „critical event“ i „critical event chart“ iz čega je evidentno da je pregled aktualnih vrijednosti moguć samo ako postoji pohranjena kritična situacija, odnosno, njen model.

2.5 Dijagram tijeka podataka (*data flow*)

Donjim dijagramom cilj je pojasniti tijek podataka od izvora procesnih vrijednosti koje se generiraju u PLC-u, odnosno prikupljaju sa mjernih uređaja i senzoristike u polju, do cikličkih usporedbi aktualnih i pohranjenih tih istih vrijednosti nakon što su konvertirane i pohranjene u SQL bazu.



Slika 7: Dijagram tijeka podataka

Prema gore prikazanom, konzolna aplikacija (S7 client) prikuplja podatke iz PLC-a u obliku polja bajtova (row data – byte array). Prikupljeni podaci se konvertiraju u poznate tipove podataka (real, long integer, integer, bool), te se pohranjuju u SQL bazu. Uz konverziju podataka u spomenute tipove, bitno je napomenuti da se u Siemens PLC-u, za koji je I/O server razvijen, zapis složenih tipova podataka izveden u BIG-Endian formatu, te da se u konzolnoj aplikaciji konvertira u LITTLE-Endian kako je to uobičajeno za interni PC format zapisa. Time završava jedna cjelina u tijeku podataka koja je izolirana od tijeka podataka unutar Web aplikacije, uz SQL bazu kao zajedničku točku razmjene podataka. Drugi dio odnosi se na sustav detekcije kritične situacije iz podataka dostupnih u bazi. Da bi isto bilo realiziran, postoji ciklički tijek podataka iz baze prema serverskoj strani aplikacije koji čita podatke u realnom vremenu (*de facto realtime*) na način da se čita posljednji upis u bazu od strane konzolne aplikacije. Ti se podaci unutar web aplikacije uspoređuju sa pohranjenima, te se prema potrebi generira upozorenje. Uz opisani ciklički tijek podataka (između serverske strane i baze), postoji i tijek podataka kojim se kreira i pohranjuje model kritičnog događaja (između klijentske strane i baze) koji služi za usporedbu podataka sa onima prikupljenim u cikličkom djelu tijeka podataka.

2.6 Model kritičnog događaja

Temeljeni cilj predmetnog rada jeste otkrivanje/detektiranje kritičnog događaja prije nego isti nastupi, odnosno prije nego se ispune svi uvjeti da nastane situacija istovjetna onoj koja je prethodno registrirana i pohranjena. Da bi to bilo realno moguće, potrebno je postaviti određene preduvjete glede definiranja kritične situacije. Naime, da bi se kritična situacija registrirala i opisala (definirao model) ključno je odrediti koji signali (tagovi) sudjeluju u istoj, te sa kojim vrijednostima u trenutku opažanja kritične situacije. Navedeno je dovoljno za evidentiranje kritične situacije, ali nedovoljno za potrebe definiranja modela sa ciljem preventivnog otkrivanja budućih istih događaja. Da bi se događaj detektirao prije ispunjenja uvjeta, neophodno je reagirati prije nego se dosegnu zabilježene vrijednosti uključenih signala. Međutim, neke vrijednosti mogu doseći kritično visoku razinu, dakle trend kretanja vrijednosti je uzlazan, dok je za druge silazan u slučaju dosezanja kritično niske razine vrijednost. Dakle, uz informaciju o kritičnim vrijednostima signala, također je potrebna informacija o trendu kretanja svakog pojedinačno. Da bi se zabilježio trend kretanja vrijednosti očigledno je potrebno definirati i vremenski okvir koji se uzima u obzir. S obzirom da je korisnik aplikacije osoba sa poznavanjem procesa koji se kontrolira, time je i mjerodavan za određivanje relevantnog vremenskog perioda koji se može uzeti u obzir za računanje trendova kretanja vrijednosti signala (vrijeme u kojemu su vrijednosti počele izlaziti iz okvira uobičajenog). Prema gore navedenom, kritični zadovoljavajući opis kritičnog događaja treba sadržavati podatke iz donje tablice.

Tablica 1: podaci za definiranje kritičnog događaja

Naziv kritičnog događaja	string
Vrijeme registracije kritičnog događaja	dateTime
Ime taga (signala)	string
Aktualna vrijednost taga	real, long integer, integer, bool
Trend kretanja vrijednosti (pozitivno /negativno)	char (n/p)
Relevantan vremenski period (<i>time span</i>) u minutama	integer

Opcionalno se uz gornje podatke može dodati i vrijeme ciklusa čitanja (*time resolution*) koje će biti dovoljno da zadovolji relativnu promjenu vrijednosti taga koja se najbrže mijenja. Time bi se izbjegla opasnost da se brzom promjenom aktualnih vrijednosti u kombinaciji sa niskom frekvencijom ciklusa usporedbe sa pohranjenim modelom, kritična situacija ne prepozna na vrijeme. Međutim, to se odnosi na vrlo brze procese koji nisu cilj ovog projekta. Općenito, za široku primjenu u proizvodnim procesima, ciklus čitanja (rezolucija) od pet sekundi zadovoljiti će najveći dio analognih signala.

3. Implementacija

U fazi implementacije, teorijski opis sustava dobiven modeliranjem prenosi se u konkretno programsko rješenje izvedeno u odgovarajućoj tehnologiji za zadani problem. Sukladno tome, odabir tehnologije mora zadovoljiti postavljene zahtjeve.

3.1 Zahtjevi u odabiru tehnologije

- Besplatan razvojni alat

Jedan od ciljeva, i ideja vodilja, je ostvariti mogućnost za razvoj SCADA podsustava koji bi bio u najvećoj mjeri besplatan. Značajan korak u tom smjeru je odabir besplatnog razvojnog alata. Ukoliko se u samom početku moraju izdvojiti određena sredstva da bi se uopće započelo eksperimentirati sa razvojem softvera, teže će se donijeti odluka o odabiru upravo tog smjera razvoja istog.

- Mala veličina aplikacije (eng. *footprint*)

Imajući u vidu da je svaki kontrolirani proces također sustav koji se kontinuirano mijenja i nadograđuje, isto će vrijediti za sustav koji ga kontrolira, a time i pripadni podsustavi koji na bilo koji način ovise o istom izvoru podataka iz kontroliranog procesa. U tom slučaju, iznimno je korisno da se aplikacija može lako i brzo prenositi uobičajenim kanalima komunikacije, bez potrebe za posebnim procedurama i trećim servisima za prebacivanje, često i više GB podataka (koliko neki SCADA sustavi danas zauzimaju).

- Rješenje za razvoj I/O servera za komunikaciju prema kontroleru mora biti zaseban modul ili aplikacija kojoj se na jednostavan način može pristupiti koja na jednostavan način može uspostaviti komunikaciju sa bazom

podataka. Na taj način cilj je omogućiti jednostavna proširenja na više podržanih PLC kontrolera.

- Ažuriranja i nadogradnja

Kada je riječ o velikim i kompleksnim SCADA sustavima, podrazumijeva se da postoji potreba za povremenim intervencijama i ažuriranjima samog sustava (ne u samu aplikaciju). Takve se aktivnosti planiraju, i imaju svoju cijenu održavanja. Međutim, u ovom slučaju takva praksa nije prihvatljiva, te stoga softversko rješenje mora biti takvo da omogući normalno funkcioniranje sustava u dogledno vrijeme bez potrebe za intervencijama takovoga tipa.

3.2 Razvojne tehnologije – I/O server

Sukladno postavljenim zahtjevima, I/O server je realiziran kao konzolna aplikacija u razvojnom sučelju kako je prikazano u tablici.

Tablica 2: Softverske tehnologije za razvoj I/O servera

Naziv	verzija
Windows OS	10x
.Net framework	4.6.01038
C#	7.3
Visual Studio Community 2017	15.8.7
- NuGet Package manager	4.6.0
- Npgsql	4.0.8
- GitHub.VisualStudio	5.2.5.2500
- SQL Server Data Tools	15.1.61808.07020
PostgreSQL	11.4
- PgAdmin 4	4.10

Uz gore prikazano softversko okruženje i razvojne alate, programsko rješenje konzolne aplikacije razvijeno je korištenjem klase Sharp7.cs koja pripada projektu Sharp7 autora Davide Nardella, objavljenom pod licencom GNU (eng. *General Public License*). Navedena klasa korištena je za čitanje polja bitova iz Siemens S7 serije PLC-a.

```

Sharp7.cs Client.cs
Sharp7.MsgSocket

/*=====
PROJECT Sharp7 1.0.0
=====
Copyright (C) 2016 Davide Nardella
All rights reserved.
=====
Sharp7 is free software: you can redistribute it and/or modify
it under the terms of the Lesser GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

It means that you can distribute your commercial software which includes
Sharp7 without the requirement to distribute the source code of your
application and without the requirement that your application be itself
distributed under LGPL.

Sharp7 is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
Lesser GNU General Public License for more details.

You should have received a copy of the GNU General Public License and a
copy of Lesser GNU General Public License along with Sharp7.
If not, see http://www.gnu.org/licenses/
=====*/

```

Slika 8: Sharp7 zaglavlje klase
(Izvor: <https://sourceforge.net/projects/snap7/files/Sharp7>)

3.3 Razvojne tehnologije – Web aplikacija

Web aplikacija rane detekcije kritične situacije razvijena je korištenjem niže prikazanih razvojnih alata, softverskih tehnologija, programskih jezika i okvira (eng. *framework*) .

Tablica 3: Softverske tehnologije i razvojni alat za razvoj web aplikacije

Naziv	verzija
Windows OS	10x
.Net framework	4.6.01038
Visual studio code	1.36.1
PostgreSQL 11.4	11.4

Tablica 4: Programski jezici (framework) i tehnologije za razvoj web aplikacije

Naziv	verzija
Server (back end)	
- Node.js	10.15.1
- Express.js	4.16.4

- Socket.io	2.2.0
- Socket.io-client	2.2.0
- Axios	0.18.1
- Sequelize	7.7.4
Klijent (front end)	
- Vue.js	2.6.10
- Bootstrap	4.3.1
- Bootstrap-vue	2.0.0-rc.19
- postgresql	0.0.1
- pg	7.10.0
- vue-google-charts	0.3.2
- vuex	3.1.1
- vue-socket.io	3.0.7
- vue-router	3.0

3.4 Arhitektura web aplikacije

Sukladno postavljenim zahtjevima, web aplikacija rane detekcije kritičnog događaja razvijena je u troslojnoj MCV arhitekturi (eng. *Model View Controller*) kako je prikazano na slijedećoj slici. *Model* i *Controller* pripadaju serverskome djelu koji je razvijen u Node.js programskom okviru, dok je dio koji se odnosi na *View*, razvijen u Vue.js programskom. Pojedinačno slojevi MCV arhitekture u ovom slučaju obavljaju slijedeće zadaće:

- Controller - Svaka pojedinačna komponenta unutar direktorija sadrži programski kod koji izvršava/obrađuje odgovarajući zahtjev zaprimljen od strane View komponenti putem usmjerivača (route.js), te vraća rezultat. Također, pojedinačna komponenta realTimeMonitoring.controler.js na zahtjev serverske strane(server.js) vrši potrebne zadaće za detekciju kritične situacije na temelju pohranjenog modela, te generira upozorenje ako su za to ispunjeni uvjeti.
- Model – predstavlja ORM model baze podataka u kojem je svaka pojedinačna komponenta model jedne relacije (tablice) kojoj je moguće pristupiti kao objektu, te na taj način izvršiti CRUD operacije (eng. *Create Read Update Delete*).

- View – sadrži skup vue komponenti iz kojih se u web pregledniku generiraju stranice koje sadrže programski kod kojim se generiraju zahtjevi prema komponentama sadržanima u controller direktoriju.



Slika 9: MCV arhitektura – struktura direktorija

3.5 Poziv metode za detekciju kritičnog događaja

Detekcija kritičnog događaja, odnosno usporedba aktualnih vrijednosti i pripadnih trendova rasta/pada sa pohranjenim modelom kritičnog događaja odvija se u serverskom djelu (node.js) u komponenti „realTimeMonitoring.controller.js“, na poziv iz server.js komponente, a prilikom aktivacije socket.io modula kako je prikazano dolje.

```
// Loading socket.io
var io = socket(server);

io.on('connection', function (socket) {
  console.log('Client connected to socket.io !');
  setInterval(async function() {
    let cWarning = await RealTimeMonitoring.socketSendToClient()
    console.log(cWarning);
    if(cWarning !== 0){
      socket.emit("earlyWarningAlert", cWarning);
    }
  }, 5000);
});
```

Iz gornjeg je programskog koda vidljivo da se metoda socketSendToClient() poziva asinkrono i u ciklusima od 5 sekundi, te generira (emitira) upozorenje prema klijentu (*front end*) ukoliko je povratna vrijednost različita od nule. Vrijeme ciklusa od 5 sekundi definirano je sukladno objašnjenjima danim u poglavlju 2.6 Model kritičnog događaja.

3.6 Algoritam detekcije kritičnog događaja

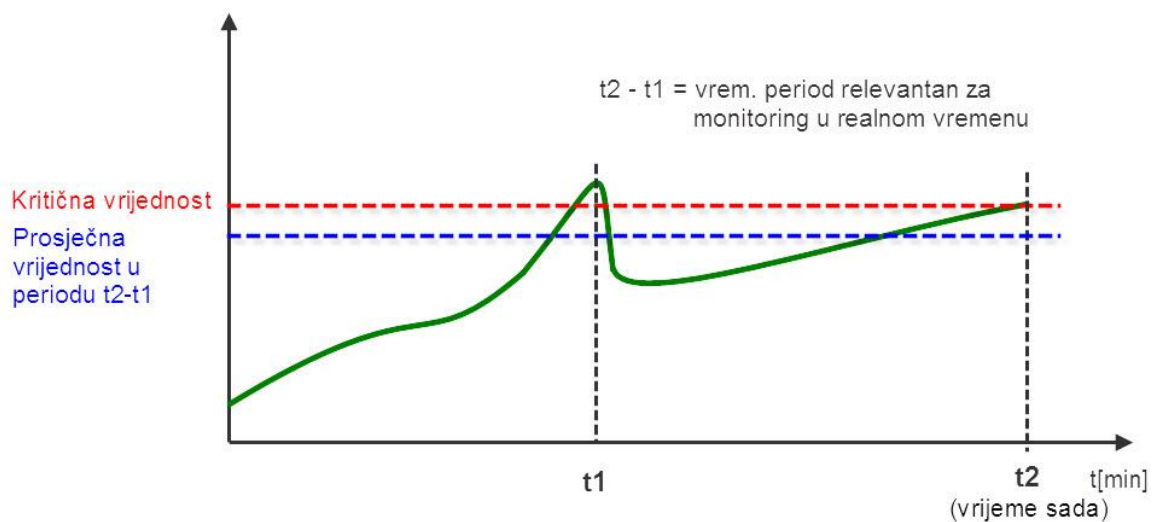
Kako je već rečeno, niže prikazan algoritam detekcije kritičnog događaja nalazi se u komponenti „realTimeMonitoring.controller.js“ gdje se u realnom vremenu uspoređuju aktualne vrijednosti za svaki pohranjeni model kritične situacije. Algoritam radi na način da za svaki evidentirani kritični događaj učitava sve ključne parametre (prema tablici 4), Putem imena taga (jedan od parametara) upitom na bazu dolazi se do adrese i tipa podatka za svaki pojedini tag što je potrebno za prikupljanje vrijednosti tagova u zadanom vremenu (*time span*). Za učitane vrijednosti računa se jedinstvena prosječna vrijednost u zadanom vremenu (posljednjih *time span* minuta), te se usporedbom te i aktualne vrijednosti dolazi do

pozitivnog ili negativnog trenda kretanja vrijednosti u promatranom vremenu od značaja za nastanak kritične situacije. Usporedbom dobivenih veličina sa pohranjenim modelom, dobiva se postotak ispunjenja uvjeta za svaki pojedini tag. U slučaju pozitivnog trenda, smatrati će se da je uvjet za generiranje upozorenja ispunjen ako aktualna vrijednost dosegne iznad 80% pohranjene vrijednosti u slučaju pozitivnog trenda, te ispod 120% pohranjene vrijednosti u slučaju negativnog trenda. Upozorenje će se generirati u slučaju da vrijednosti svih promatranih tagova pređu zadane pragove.

```
exports.socketSendToClient = async function(){
  let cEventDetails = [];
  let actWarningArr = [];
  let cParam = await getCriticalParam();
  for (let i = 0; i < cParam.length; i++) {
    cEventDetails.push(cParam[i].dataValues);
  }
  let cEventTagDetails = [];
  for (let k = 0; k < cEventDetails.length; k++) {
    let cEventTags = await getCriticalTagsParam(cEventDetails[k].ev_name);
    for (let j = 0; j < cEventTags.length; j++) {
      cEventTagDetails.push(cEventTags[j].dataValues);
    }
    for (let i = 0; i < cEventTags.length; i++) {
      let tagDetails = await getTagDetails(cEventTags[i].mod_tag_name);
      switch(tagDetails.tag_data_type) {
        case "Real":
          actualState.actVal = await getTagActValueReal(tagDetails);
          actualState.actAvg = await getTagAvgReal(cEventDetails[k],tagDetails)
          if (actualState.actVal > actualState.actAvg &&
cEventTags[i].mod_tag_trend == "P" && (actualState.actVal /
cEventTags[i].mod_tag_val) > 0.8){
            actWarningArr.push(cEventTags[i].mod_tag_name);
            console.log("- A L A R M - P -" + cEventTags[i].mod_tag_name);
          }
          if (actualState.actVal < actualState.actAvg &&
cEventTags[i].mod_tag_trend == "N" && (actualState.actVal /
cEventTags[i].mod_tag_val) < 1.2){
            actWarningArr.push(cEventTags[i].mod_tag_name);
            console.log("- A L A R M - N -" + cEventTags[i].mod_tag_name);
          }
        }
      }
    }
  }
```

Gore prikazan segment programskog koda računa navedeno u slučaju taga čija je vrijednost realnog tipa podatka. Ono što je preostalo za objasniti u opisanim koracima kojima se dolazi do odluke o potrebi generiranja upozorenja jeste

računanje prosječne vrijednosti pojedinog taga za promatrani period vremena, koja se koristi za usporedbu sa aktualnom vrijednošću da bi se dobio trend kretanja vrijednosti. Naime, isti trend bi se jednostavnije mogao dobiti usporedbom aktualne vrijednosti sa vrijednošću na početku *time span* perioda (najstarija promatrana vrijednost). Međutim, budući se radi o analognim vrijednostima u proizvodnim procesima, situacija ne mora biti tako jednostavna. Naime kada se radi o vrijednostima pritiska, protoka, temperature, br. okretaja i sl., uvijek postoji mogućnost trenutnih oscilacija koje mogu u određenom trenutku preći zadane granice a da u tom kratkom vremenu nemaju značajni utjecaj na nastanak kritične situacije. Jedan takav primjer dan je slijedećom slikom.



Slika 10: Pozitivan trend kretanja vrijednosti

Iz gornjeg primjera da se zaključiti kako je moguće doći do zaključka da postoji pozitivan i negativni trend kretanja vrijednosti, ovisno o načinu računanja i trenutku mjerenja. U slučaju kada bi se trend računao jednostavnom usporedbom između aktualne vrijednosti u vremenu t_2 (koja je dosegla pohranjenu kritičnu vrijednost), i vrijednosti u vremenu t_1 ($t_1 = t_2 - \text{timeSpan}$), dolazi se do zaključka da se vrijednost u tom rasponu vremena kreće silazno. Takvim rezultatom, izostati će ispunjenje uvjeta za generiranje upozorenja premda je aktualna vrijednost dosegla kritičnu, ali algoritam očekuje da uz silazan trend, ta vrijednost bude 120% kritične. Računanjem prosječne vrijednosti u cijelom periodu $t_2 - t_1$ takva se greška zbog povremenih kraćih oscilacija isključuje. Promatrajuću plavu isprekidanu liniju koja predstavlja okvirnu prosječnu vrijednost, očigledno je da u zadanom periodu $t_2 - t_1$ promatrani signal ima

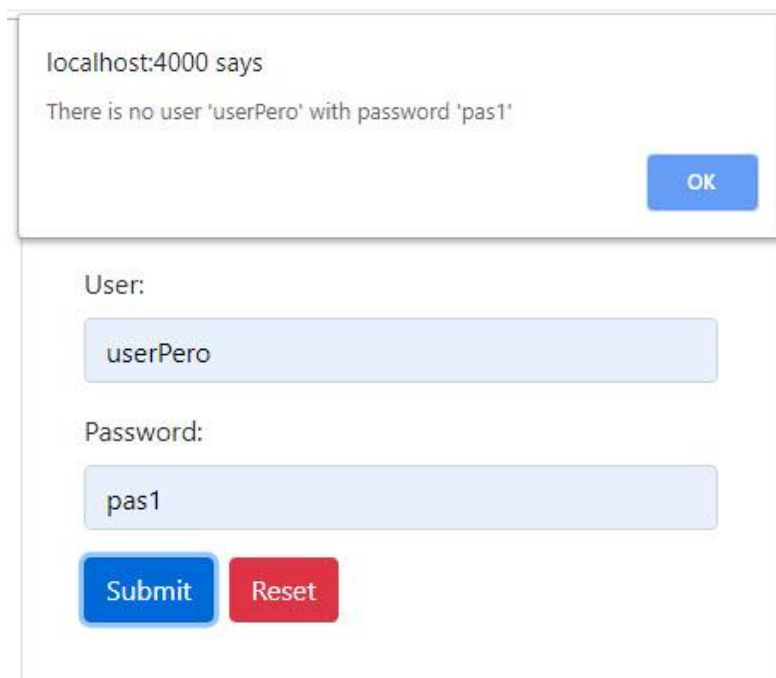
pozitivni trend kretanja vrijednosti, što posljedično ispunjava uvjet za generiranje upozorenja nakon prelaska granice od 80% pohranjene vrijednosti signala.

4. Korisničke upute

Upute dane u nastavku teksta imaju za cilj pružiti cjelovit pregled sučelja, sa svim prikazima i objašnjenjem funkcionalnosti svakog od njih, te pripadnih elemenata. Sadržaj je prvenstveno usmjeren na korisnika aplikacije, na način da ga pripremi za redovan rad korištenjem raspoloživih funkcionalnosti.

4.1 Prijava (Logon)

Korisnik (koji postoji u bazi) mora se prijaviti korisničkim imenom i lozinkom. Krivi unos generirati će poruku upozorena kao na slici.



The image shows a web application interface for a login form. At the top, there is a light gray message box with the text "localhost:4000 says" and "There is no user 'userPero' with password 'pas1'", and a blue "OK" button. Below this, the login form itself is visible. It has a "User:" label followed by a text input field containing "userPero". Below that is a "Password:" label followed by a text input field containing "pas1". At the bottom of the form are two buttons: a blue "Submit" button and a red "Reset" button.

Slika 11: Logon forma

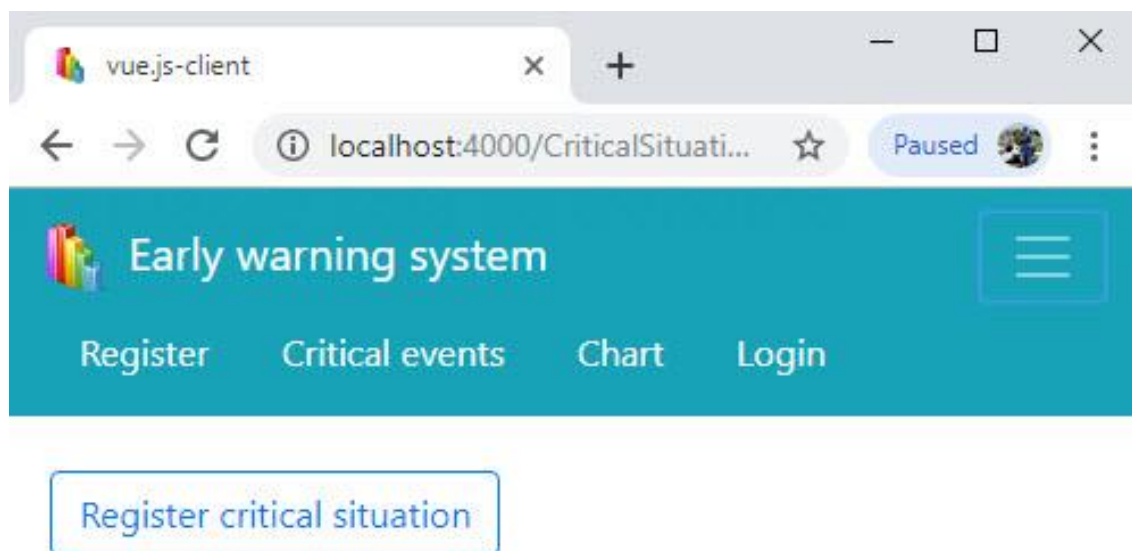
4.2 Registriranje kritične situacije

Ulaskom u sustav, korisnik inicijalno ima mogućnost registrirati kritičnu situaciju koja se trenutno manifestira jednostavnim klikom na dugme. Time se registrira trenutno vrijeme, te se korisnika otvaranjem formi za unos vodi kroz postupak unosa svih relevantnih podataka za dani kritični događaj.



Slika 12: Registriranje kritične situacije

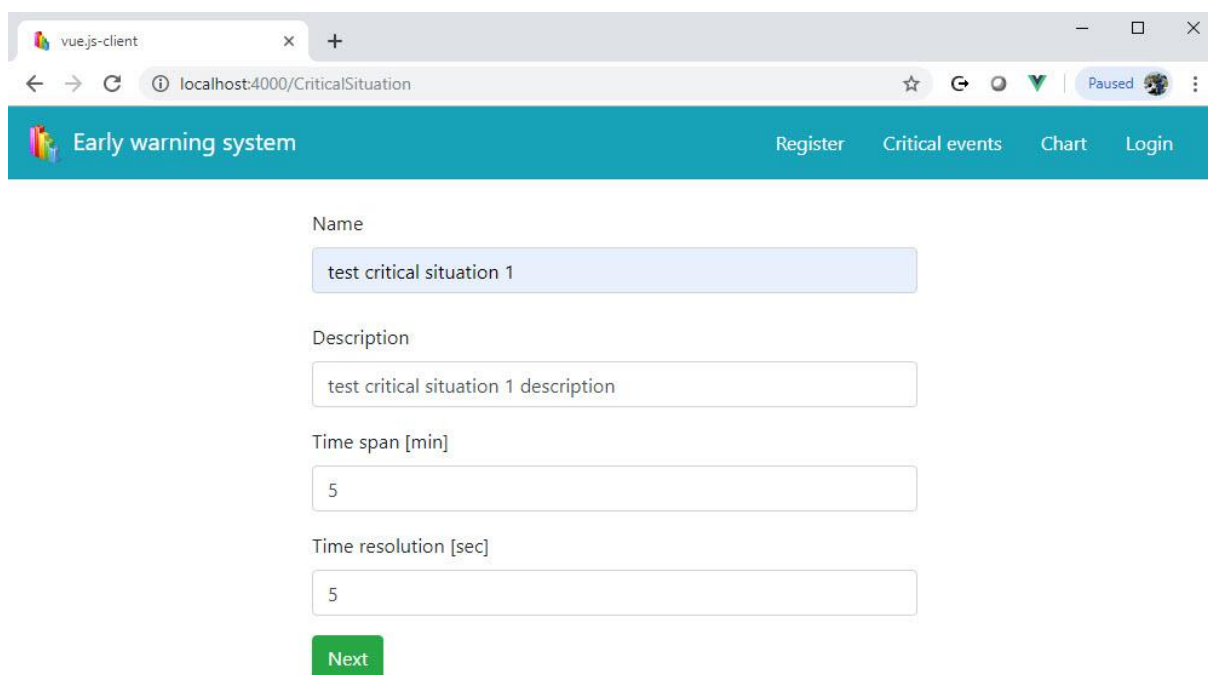
U slučaju da se korisnik u sustav rane detekcije nije prijavio radi evidentiranja noge kritične situacije, u gornjem djelu postoji navigacijska traka sa ostalim opcijama za pregled postojećih modela kritičnih situacija, te praćenje istih u realnom vremenu. Aplikacija je također prilagođena za manje ekrane, te se gornji prikaz može i grafički skalirati prema donjoj slici, čime se u ovom primjeru navigacijska traka prilagodila izgledom.



Slika 13: Registriranje kritične situacije - skalirano

4.3 Unos osnovnih podataka

Donji prikaz otvara se klikom na dugme „Register critical situation“ sa prethodne slike, te omogućuje unos osnovnih podataka potrebnih za evidentiranje i pohranu modela kritične situacije. Kako je već prethodno rečeno, podatak o vremenskoj rezoluciji ciklusa usporedbe pohranjenog modela sa realnim vrijednostima nije neophodan budući se za opću namjenu može smatrati da se radi o procesima kod kojih se vrijednosti značajno ne mijenjaju unutar pet sekundi, te je time i takav razmak ciklusa fiksno definiran u programskom kodu (eng, *hardcoded*)



The screenshot shows a web browser window with the address bar displaying 'localhost:4000/CriticalSituation'. The application header is teal with the text 'Early warning system' and navigation links: 'Register', 'Critical events', 'Chart', and 'Login'. The main form contains the following fields:

- Name:** A text input field containing 'test critical situation 1'.
- Description:** A text input field containing 'test critical situation 1 description'.
- Time span [min]:** A text input field containing '5'.
- Time resolution [sec]:** A text input field containing '5'.

At the bottom of the form is a green button labeled 'Next'.

Slika 14: unos osnovnih podataka

Unos u polje „Time span [min]“ odnosi se na vremenski period koji korisnik (operater koji kontrolira proizvodni proces) smatra relevantnim za predmetni kritični slučaj koji se registrira i pohranjuje. Vremenski period koji se ovdje unese definirati će vrijeme u kojemu se vrijednosti čitaju u realnom vremenu, odnosno vremenski odmak u natrag od aktualnog vremena.

4.4 Unos tagova (signala)

Uz osnovne podatke koji se unose putem gornje forme, klikom na dugme „Next“, otvara se slijedeća forma za odabir tagova koji se smatraju relevantni za nastanak kritične situacije, i čije će se vrijednosti dalje pratiti.

vue.js-client

localhost:4000/CriticalTags

Early warning system

Register Critical events Chart Login

Critical situation:

test critical situation 1

Description:

test critical situation 1 description

Select tags:

Tag	Description	Unit
PUMP1_OUTPUT_FLOW	Pump1 output flow	l/h
PUMP2_OUTPUT_FLOW	Pump2 output flow	l/h
PUMP3_OUTPUT_FLOW	Pump3 output flow	l/h

Add selected tag

Slika 15: unos tagova

Nakon unosa svakog novog taga odabranog se liste, otvara se mogućnost unosa dodatnog taga, ili opcija prelaska na slijedeći korak definiranja modela kritične situacije.

vue.js-client

localhost:4000/CriticalTags

Early warning system

Register Critical events Chart Login

Critical situation:

test critical situation 1

Description:

test critical situation 1 description

Select tags:

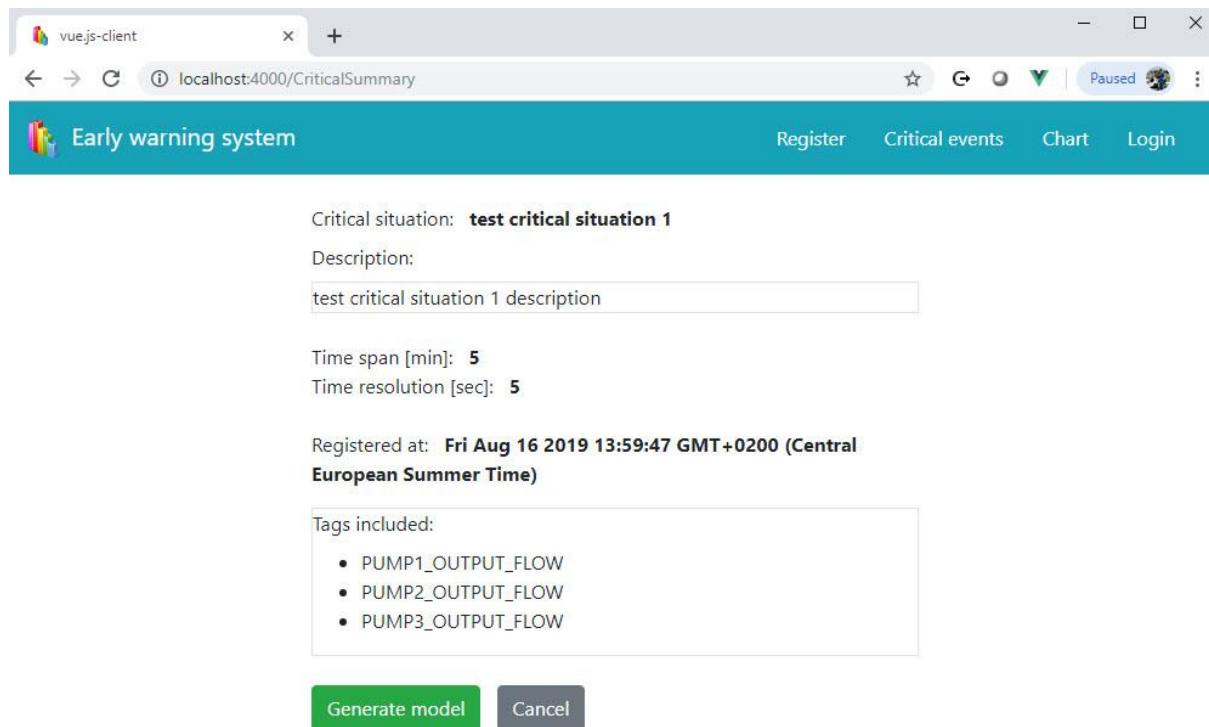
Tag	Description	Unit
PUMP1_OUTPUT_FLOW	Pump1 output flow	l/h
PUMP2_OUTPUT_FLOW	Pump2 output flow	l/h
PUMP3_OUTPUT_FLOW	Pump3 output flow	l/h

Add another Next step

Slika 16: unos dodatnog taga

4.5 Spremanje modela kritične situacije

Nakon što je u prethodnim koracima dovršen unos osnovnih podataka i svih relevantnih tagova, prije pohranjivanja modela korisnik dobiva na uvid sažetak informacija(model) za predmetnu kritičnu situaciju.



The screenshot shows a web browser window with the address bar displaying 'localhost:4000/CriticalSummary'. The page title is 'Early warning system'. The navigation bar includes links for 'Register', 'Critical events', 'Chart', and 'Login'. The main content area displays the following information:

- Critical situation: **test critical situation 1**
- Description:
- Time span [min]: **5**
- Time resolution [sec]: **5**
- Registered at: **Fri Aug 16 2019 13:59:47 GMT+0200 (Central European Summer Time)**
- Tags included:
 - PUMP1_OUTPUT_FLOW
 - PUMP2_OUTPUT_FLOW
 - PUMP3_OUTPUT_FLOW

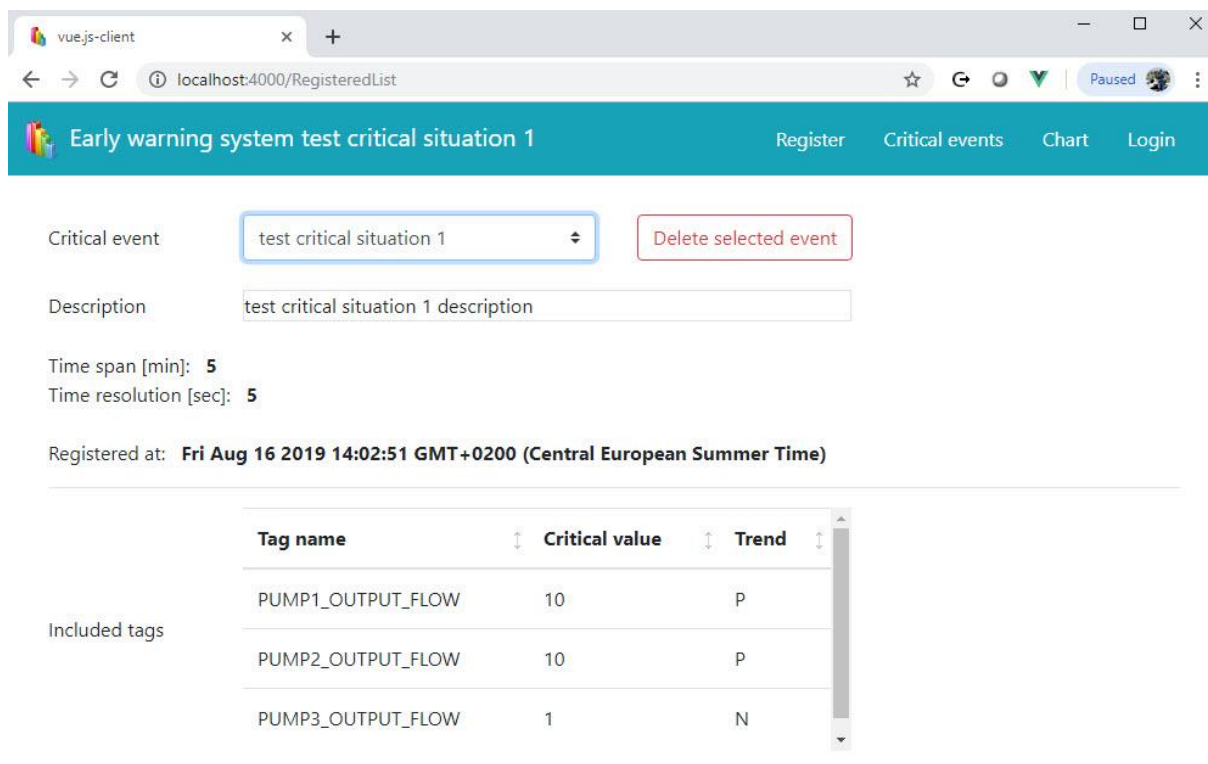
At the bottom, there are two buttons: 'Generate model' (green) and 'Cancel' (grey).

Slika 17: Sažetak podataka

Kritična situacija, čiji je sažetak prikazan na gornjoj slici, može se pohraniti, ili korisnik može odustati od registriranja iste. Do klika na dugme „Generate model“ u bazi podataka još uvijek ne postoji nikakav podatak o definiranoj kritičnoj situaciji. U trenutku kada se kritična situacija registrira, odmah započinje praćenje u realnom vremenu (eng. *real-time monitoring*). U slučaju da se kritične vrijednosti nisu mijenjale, prvo upozorenje će biti generirano nakon prvog ciklusa od pet sekundi.

4.6 Pregled modela pohranjenih kritičnih situacija

Sa navigacijske trake je u svakom trenutku moguće dobiti uvid u pohranjene modele kritičnih događaja. Svaki će model biti prikazan sa svim unesenim podacima i točnom vremenu opažanja prve pojave pohranjene kritične situacije. Dodatno, u tablici sa donje slike biti će vidljive i kritične vrijednosti koje su zabilježene u trenutku opažanja, te izračunati trendovi (P-pozitivni / N-negativni) u definiranom vremenu (*time span*).



Slika 18: Pregled pohranjenih modela

Svaki gore prikazani pohranjeni model kritične situacije moguće je izbrisati iz baze, čime se praćenje iste u realnom vremenu prekida, te upozorenja više neće biti generirana.

4.7 Grafički prikaz aktualnih vrijednosti

Za svaki pohranjeni model kritične situacije koji je dostupan na pregled putem navigacijske trake, također je za isti, te na isti način, moguće dobit uvid u aktualne vrijednosti svih tagova uključenih u predmetnu kritičnu situaciju.



Slika 19: Odabir pohranjenog modela kritične situacije

Odabirom pohranjenih modela, te klikom na dugme „show chart“, pojavljuje se grafički prikaz(stupci) aktualnih analognih vrijednosti kako je prikazano na donjoj slici.

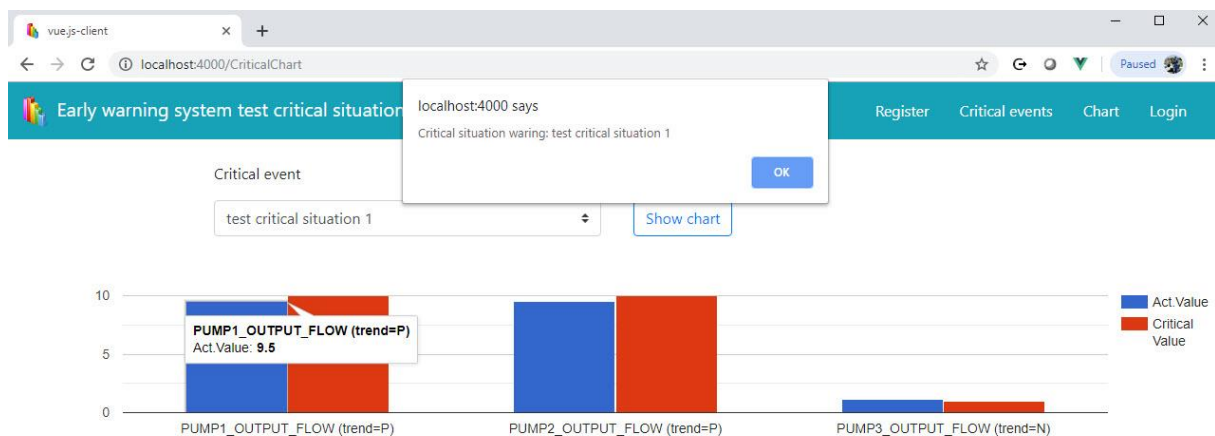


Slika 20: Grafički prikaz aktualnih vrijednosti pohranjenog modela

Za odabranu pohranjenu kritičnu situaciju, u stupcima se prikazuju pohranjene kritične vrijednosti (crveno) i aktualne vrijednosti u trenutku generiranja prikaza (plavo). Tekstualni opis na apscisi sadrži ime taga i njegov pohranjen trend kretanja vrijednosti u zadanom vremenu (*time span*), koji je definiran u vrijeme registriranja kritičnog događaja.

4.8 Upozorenje na kritični događaj

Sukladno prethodno navedenim zahtjevima te ispunjenju uvjeta u algoritmu detekcije kritičnog događaja, isti će biti generiran i prikazan sa nazivom pohranjenog modela kritičnog događaja.



Slika 21: Upozorenje na kritični događaj

Kako je vidljivo iz usporedbe aktualnih i pohranjenih vrijednosti tagova (donja slika), u trenutku generiranja upozorenja kritični događaj još nije nastao. Međutim, vjerojatnost nastanka istog je značajna prema zadanim uvjetima u algoritmu detekcije. Kako je prethodno rečeno u poglavlju 3.6, smatrati će se da je uvjet za generiranje upozorenja ispunjen ako aktualna vrijednost dosegne iznad 80%

pohranjene vrijednosti u slučaju pozitivnog trenda, te ispod 120% pohranjene vrijednosti u slučaju negativnog trenda. Upozorenje će se tada generirati u slučaju da vrijednosti svih promatranih tagova pređu zadane granice.



Slika 22: Detalji upozorena na kritični događaj

Prema gornjoj slici, aktualne vrijednosti prva dva taga (PUMP1_OUTPUT_FLOW i PUMP2_OUTPUT_FLOW) imaju pozitivne trendove, te su dosegle iznad 80% pohranjene kritične vrijednosti (9.5 : 10), dok je aktualna vrijednost trećeg taga (PUMP3_OUTPUT_FLOW) sa negativnim trendom, dosegla razinu nižu od 120% pohranjene kritične vrijednosti (1.1 : 1). Time je ispunjen uvjet prema kojemu vrijednosti svih tagova u promatranom vremenu (*time span*) moraju preći zadanu granicu da bi se smatralo kako vjerojatnost nastanka kritičnog događaja opravdava generiranje upozorenja.

5. Zaključak

Aplikacija SCADA podsustava za radnu detekciju kritične situacije izvedena je sukladno postavljenim zahtjevima glede funkcionalnosti i odabira tehnologija za izvedbu iste. Način na koji je predmetni SCAD podsustav razvijen, omogućava visoku razinu prilagodljivosti u odnosu na postojeće dijelove SCADA sustava kontrole proizvodnog procesa i akvizicije podataka. Uz navedeno, proširenje sustava u smislu komunikacije sa različitim vrstama PLC-a pojednostavljeno je odvajanjem I/O servera kao zasebnog modula. Na taj način, isključena je bilo kakva potreba intervencije u programski kod web aplikacije prilikom razvoja dodatnih komunikacijskih modula (I/O servera) za različite vrste PLC-a.

6. Popis slika

Slika 1: Industrijski kontrolni sustav	6
Slika 2: Use-case dijagram	10
Slika 3: Sequence dijagram – Web aplikacija: dio 1	11
Slika 4: Sequence dijagram – Web aplikacija: dio 2	12
Slika 5: Sequence dijagram – I/O server (konzolna aplikacija)	13
Slika 6: Klasni dijagram	14
Slika 7: Dijagram tijeka podataka	15
Slika 8: Sharp7 zaglavlje klase	19
Slika 9: MCV arhitektura – struktura direktorija	21
Slika 10: Pozitivan trend kretanja vrijednosti	24
Slika 11: Logon forma	25
Slika 12: Registriranje kritične situacije	26
Slika 13: Registriranje kritične situacije - skalirano	26
Slika 14: unos osnovnih podataka	27
Slika 15: unos tagova	28
Slika 16: unos dodatnog taga	28
Slika 17: Sažetak podataka	29
Slika 18: Pregled pohranjenih modela	30
Slika 19: Odabir pohranjenog modela kritične situacije	30
Slika 20: Grafički prikaz aktualnih vrijednosti pohranjenog modela	31
Slika 21: Upozorenje na kritični događaj	31
Slika 22: Detalji upozorena na kritični događaj	32

7. Popis tablica

Tablica 1: podaci za definiranje kritičnog događaja	16
Tablica 2: Softverske tehnologije za razvoj I/O servera	18
Tablica 3: Softverske tehnologije i razvojni alat za razvoj web aplikacije	19
Tablica 4: Programski jezici (framework) i tehnologije za razvoj web aplikacije	19

8. Literatura

Internet

1. <https://sourceforge.net/projects/snap7/files/Sharp7>
2. <https://openclassrooms.com/en/courses/2504541-ultra-fast-applications-using-node-js/2505653-socket-io-let-s-go-to-real-time>
3. <https://blogs.igalia.com/jfernandez/2012/12/19/node-js-socket-io-real-time-io/>
4. <https://support.industry.siemens.com/tf/ww/en/posts/application-in-c-windows-application-for-sending-receiving-data-to-plc-s7-1200/181756/?page=0&pageSize=10>
5. <https://www.vuescript.com/reactive-vue-js-wrapper-for-google-charts-gchart/>
6. <https://medium.com/@anaida07/mevn-stack-application-part-1-3a27b61dcae0>
7. <https://forum.vuejs.org/t/passing-data-back-to-parent/1201>
8. <https://vuejs.org/v2/guide/computed.html#Computed-Caching-vs-Methods>
9. <https://alligator.io/vuejs/rest-api-axios/>
10. <https://medium.com/@koreus/vue-js-there-and-back-again-in-1-5-years->

756c1582aa96

11. <https://madewithvuejs.com/blog/best-vue-js-datatables>
12. <https://flaviocopes.com/vuex/#create-the-vuex-store>
13. <https://nodejs.org/en/docs/guides/>
14. <https://medium.com/front-end-weekly/ajax-async-callback-promise-e98f8074ebd7>
15. <https://www.tutorialspoint.com/nodejs/>
16. <https://nodeaddons.com/calling-native-c-dlls-from-a-node-js-web-app/>
17. <https://www.codeproject.com/Tips/454011/Ajax-Asynchronous-postback>
18. <https://codeburst.io/javascript-arrow-functions-how-why-when-and-when-not-to-use-them-fb8c2de9dbdc>
19. <https://blog.logrocket.com/setting-up-a-restful-api-with-node-js-and-postgresql-d96d6fc892d8>
20. <https://nodejs.org/en/docs/guides/timers-in-node/>
21. <https://stackoverflow.com/questions/23442187/node-js-socket-io-and-express-sending-data-continuously-to-the-client-other-tha>
22. <https://medium.com/@onejohi/building-a-simple-rest-api-with-nodejs-and-express-da6273ed7ca9>
23. <https://node-postgres.com/features/connecting/>
24. <https://popsql.com/learn-sql/postgresql/how-to-query-date-and-time-in-postgresql/>
25. <https://stackoverflow.com/questions/28545489/c-sharp-best-way-to-run-a-function-every-second-timer-vs-thread>
26. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
27. <https://grokonez.com/frontend/vue-js/vue-js-nodejs-express-restapis-sequelize-orm-postgresql-crud-example>
28. <https://medium.com/tkssharma/writing-neat-asynchronous-node-js-code-with-promises-async-await-fa8d8b0bcd7c>
29. <https://sequelize.readthedocs.io/en/latest/docs/querying/>
30. <https://sequelize.readthedocs.io/en/2.0/docs/models-definition/>
31. <https://bootstrap-vue.js.org/docs/>

Sažetak

SCADA podsustav za ranu detekciju kritičnog događaja je razvijen sa namjerom praćenja proizvodnog procesa u realnom vremenu, i generiranja upozorenja korisniku prije nego što se ispune uvjeti za nastanak kritičnog događaja. Da bi se navedeno postiglo, u bazi podataka pohranjeni su modeli kritičnih situacija(događaja) koji sadrže sve relevantne signale koji sudjeluju u svakome od uočenih kritičnih događaja pojedinačno. Predmetni sustav je razvijen kao web aplikacija te I/O server u dva zasebna modula. Dok je web aplikacija zadužena za izvršavanje kalkulacije u realnom vremenu, detekciju uvjeta koji se približavaju pohranjenom modelu kritične situacije i grafičke prezentacije, funkcija I/O servera je prikupljanje podataka iz PLC-a u realnom vremenu, i pohrana u SQL bazu podataka. Osnovna prednost opisanog sustava je u njegovoj fleksibilnosti u odnosu na postojeće SCADA sustave u uporabi. Ne postoje posebni zahtjevi glede kompatibilnosti protokola ili baza podataka. Također, nema potrebe za instalaciju klijentske aplikacije na bilo koje računalo koje se koristi za pristup web aplikaciji. To se jednostavno postiže putem internetskog preglednika.

Summary

SCADA subsystem for early warning detection of critical situation is developed with intention of monitoring production proces in real time, and generate warning to the user before critical condition appears. In order to do so, critical condition(event) models are stored in database, containing all relevant signals participating in each observed critical condition individually. System in question is developed as a web application and I/O server application in two separate modules. While the web application is concerned for execution of the real time calculation, detection of conditions approaching to stored critical situation model and graphical presentation, the function of I/O server is real time data collection form PLC and storing in SQL database. Main advantage of the described system is in its flexibility in relation to existing SCADA systems that might be in use. There is no special requests regarding data transfer protocol, or database compatibility. Also, there is no need for client application software installation on any PC that is to be used to access the application. This can be simply done via web browser.