Izrada web aplikacije za sustav gospodarenja željeznim otpadom

Bašić, Damir

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:616688

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-13**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Izrada web aplikacije za sustav gospodarenja željeznim otpadom

Diplomski rad

Damir Bašić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 31.08.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Damir Bašić
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. Pristupnika, godina upisa:	D-1358, 07.10.2021.
OIB studenta:	52242463789
Mentor:	izv. prof. dr. sc. Emmanuel Karlo Nyarko
Sumentor:	,
Sumentor iz tvrtke:	Matija Rajčevac
Predsjednik Povjerenstva:	izv. prof. dr. sc. Damir Filko
Član Povjerenstva 1:	izv. prof. dr. sc. Emmanuel-Karlo Nyarko
Član Povjerenstva 2:	izv. prof. dr. sc. Ratko Grbić
Naslov diplomskog rada:	Izrada web aplikacije za sustav gospodarenja željeznim otpadom
Znanstvena grana diplomskog rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Potrebno je izraditi web aplikaciju i pripadajuću bazu podataka, tzv. Level2 aplikaciju ili "Operator Workstation" za sustav gospodarenja željeznim otpadom. (Tema rezervirana za: Damir Bašić) (Sumentor iz tvrtke: Matija Rajčevac, Danieli-Systec d.o.o, Vinež 601, 52220 Labin, Hrvatska)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	31.08.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum: 14.09.2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 14.09.2023.

Ime i prezime studenta:	Damir Bašić
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1358, 07.10.2021.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada web aplikacije za sustav gospodarenja željeznim otpadom**

izrađen pod vodstvom mentora izv. prof. dr. sc. Emmanuel Karlo Nyarko

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD *1*

2. PREGLED POSTOJEĆIH RJEŠENJA VIZUALIZACIJE PODATAKA U PODRUČJU AUTOMATIZACIJE INDUSTRIJSKIH POSTROJENJA *4*

2.1. Acciaieria Arvedi Shredder Plant **4**

3. IZRADA WEB APLIKACIJE ZA SUSTAV GOSPODARENJA ŽELJEZNIM OTPADOM *6*

3.1. Definiranje podataka za razmjenu između servisne aplikacije i blokova podataka PLC-a **8**

3.2. Pravila pri razmjeni podataka između servisne aplikacije i blokova podataka PLC-a **9**

3.2.1. Kružni spremnik 10

3.2.2. Podatkovni spremnik 12

3.3. Projektiranje baze podataka **13**

3.4. Razvoj komunikacijskog softvera (servisne aplikacije) **14**

3.4.1. Siemens S7.Net 14

3.4.2. Konekcija s bazom podataka 17

3.4.3. Razvoj komunikacijskog zapisnika 19

3.4.4. Osiguravanje pravilnog rada servisne aplikacije u nepredviđenim okolnostima 20

3.5. Razvoj Web aplikacije **21**

4. TESTIRANJE RADA WEB APLIKACIJE ZA SUSTAV GOSPODARENJA ŽELJEZNIM OTPADOM *23*

4.1. Testiranje funkcionalnosti servisne aplikacije **23**

4.2. Testiranje funkcionalnosti Web aplikacije **24**

4.2.1. Testiranje funkcionalnosti autentifikacije 25

4.2.2. Testiranje funkcionalnosti manipulacije parametrima za konfiguraciju rada postrojenja 25

4.2.3. Testiranje funkcionalnosti prikaza *Delay* i *Production* zapisa 29

4.2.4. Testiranje funkcionalnosti stvarnovremenskog strujanja komunikacijskog zapisnika 31

5. ZAKLJUČAK *33*

LITERATURA *34*

1. UVOD

Zamjena fizičkog rada sa radom strojeva ideja je koji potječe još od samih početaka civilizacije. Grk *Ktesibios* 250. godina prije nove ere konstruirao je vodeni sat sa regulatorom protoka koji je predstavljao najtočniju spravu za mjerenje vremena sve do 17. stoljeća [1].

Ktesibios se smatra ocem automatike. Vodeni sat sa regulatorom protoka bio je jedinstven proizvod budući da ga je osim preciznog mjerenja te prikaza vremena karakterizirao i sustav za dodatno obavještavanje (različita zvonca). S godinama koje su prolazile, za standarde tog vremena, pojavljivali su se napredniji i napredniji izumi. U drugoj polovici 18. stoljeća počinje globalna zamjena ručne proizvodnje sa parnim strojevima (tzv. era mehanizacije) čime su se iz korijena promijenili politički, gospodarski i društveni sustavi predstavljajući prvu industrijsku revoluciju. Problematika koja se pojavljivala u periodu prve industrijske revolucije je činjenica da je čovjek bez obzira na uvelike smanjen opseg fizičkog rada i dalje sveprisutan u proizvodnom procesu. Potreba za potpunim ili djelomičnim oslobađanjem čovjeka od rada u proizvodnom procesu sa vremenom je postajala sve veća, pa se analiziranjem postojećih operacija u proizvodnom procesu nastojao konstruirati model proizvodnog sustava kako bi se čovjekova zadaća većim dijelom svodila samo na nadzor proizvodnog procesa. Upravo taj period druge polovice 19. stoljeća te početka 20. stoljeća naziva se periodom automatizacije tj. druge industrijske revolucije.

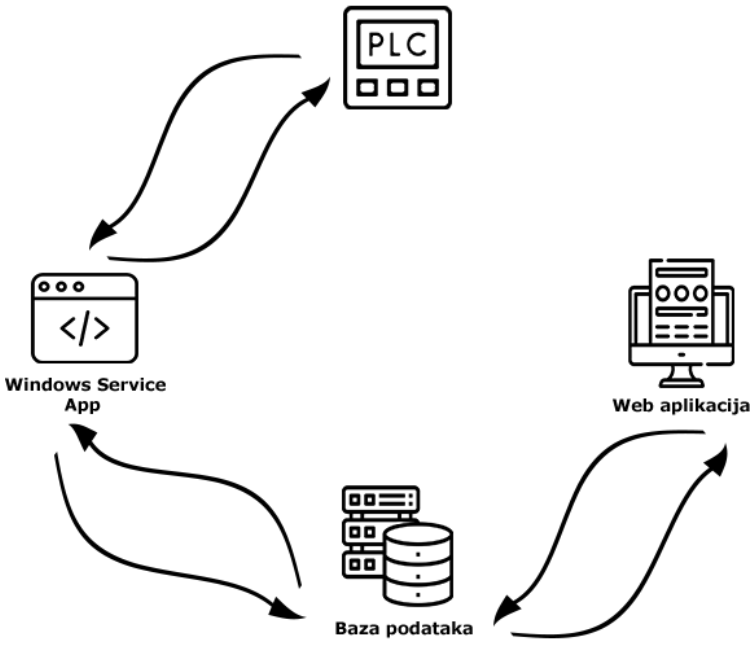
Automatizacija sa sobom donosi veliki broj prednosti u svim aspektima, samo neki od njih su: povećanje proizvodnje, povećanje sigurnosti, povećanje kvalitete proizvoda te ušteda energije. Zbog navedenih razloga razvijanje i unaprjeđivanje sustava za automatsko upravljanje nastavljeno je i nakon druge industrijske revolucije te traje i dan danas. Programirljivi logički kontroleri (engl. *PLC - Programmable Logic Controllers*), industrijska računala projektirana za teške uvjete rada, svoju primjenu prvi puta nalaze u sektoru automobilske industrije krajem 60-ih godina prošlog stoljeća te su od tada osnovni (temeljni) dio sustava (procesa) za automatsko upravljanje [2]. Idući zahtjevi takvih sustava, 90-ih godina prošlog stoljeća bili su mogućnosti nadzora, mjerenja i upravljanja pomoću računalnog sustava. Slični sustavi postoje još od 60-ih godina prošlog stoljeća, i ostvareni su pomoću HMI/SCADA (engl. *Human Machine Interface - Supervisory Control And Data Acquisition*) [3] sustava ali sa vrlo ograničenim funkcionalnostima. Razvojem bržih i efikasnijih računala potreba za naprednijim funkcionalnostima bila je sve veća i veća. S druge strane, nova razvojna okruženja i novi programski jezici 2000-ih godina postavile su temelj za projektiranje radnih stanica za operatere (engl. *Operator workstation*) aplikacija koje su pružale brojne prednosti u odnosu na HMI/SCADA sustave.

Zadatak diplomskog rada je projektirati i razviti radnu stanicu za operatere – aplikaciju za nadzor i kontrolu sustava gospodarenja željeznim otpadom. Diplomski rad ostvaren je uz suradnju sa tvrtkom *Danieli Systec*. Unutar pogona postoje određena postrojenja koja se sastoje od različitih strojeva poput glodalica, preša i sortirki upravljanih PLC-ovima čineći sustav za gospodarenje željeznim otpadom. Za takve sustave trenutno postoji tzv. *Level 1* (HMI/SCADA) sustav, te je osnovna ideja na postojeći *Level 1* sustav nadograditi i razviti tzv. *Level 2* (*Operator Workstation*) sustav.

Zahtjevi jednog takvog sustava su:

- izrada potrebne tehničke dokumentacije,
- definiranje podataka za razmjenu,
- razvoj komunikacijskog softvera (pozadinske aplikacije) za upravljanje tih podataka i
- izrada web aplikacije za vizualni prikaz i kontrolu podataka.

Na slici 1.1 prikazani su glavni dijelovi sustava.



Slika 1.1 Glavni dijelovi sustava te informacijski tokovi između pojedinih dijelova

Diplomski rad čini 5 poglavlja. Prvo poglavlje daje općenite informacije o automatizaciji, potrebi za vizualizacijom podataka te detaljno opisuje zadatak diplomskog rada. Drugo poglavlje opisuje trenutna postignuća u području razvoja radnih stanica za operatere (engl. *Operator Workstation*). U trećem poglavlju u cijelosti je opisan prijedlog rješenja Web aplikacije za sustav gospodarenja željeznim otpadom. Četvrto poglavlje detaljno opisuje načine testiranja rada predloženog rješenja te prikazuje dobivene rezultate, dok je u petom poglavlju dan zaključak.

2. PREGLED POSTOJEĆIH RJEŠENJA VIZUALIZACIJE PODATAKA U PODRUČJU AUTOMATIZACIJE INDUSTRIJSKIH POSTROJENJA

U području nadzora, kontrole i upravljanja industrijskih procesa postoji veliki broj postojećih rješenja budući da danas postoji ogroman broj industrijskih procesa u svim granama industrije. S obzirom da je područje diplomskog rada čelična industrija te, kako je spomenuto, već postoji aplikacija za konkretan sustav gospodarenja željeznim otpadom, u ovom poglavlju obraditi će se navedeno postojeće rješenje.

2.1. Acciaieria Arvedi Shredder Plant

Acciaieria Arverdi Shredder Plant je pogon unutar kojeg postoje postrojenja za sustav gospodarenja željeznim otpadom. Postojeći *HMI/SCADA* sustav razvijen je u *AVEVA inTouch* razvojnom okruženju, te kao takav predstavlja referentni model za Web aplikaciju koju je potrebno razviti u okviru diplomskog rada. U ovom potpoglavlju prikazati će se vizualni identitet same aplikacije te ukratko opisati njene funkcionalnosti. Definiranje samih nedostataka aplikacije biti će od posebne važnosti budući da nedostatci i ograničenja postojeće aplikacije predstavljaju temelj za razvijanje nove aplikacije. Na slici 2.1 prikazana je tablica koja služi za konfiguraciju načina rada samog postrojenja (tzv. recepti).

HYDRAULIC ACTUATORS						
		Table 1	Table 2	Table 3	Table 4	Table 5
Table name		100% - H	100% - M	100% - L	90%	80%
Infeed conveyor target speed	%	32	28	23	25	21
Infeed conveyor automatic start delay after CFR	ms	2000	2000	2000	2000	2000
Infeed conveyor automatic stop delay after CFR	ms	4000	4000	4000	4000	4000
Compression feed roller target speed	%	70	80	90	85	80
Compression feed roller stall pressure	bar	100	100	100	100	100
Compression feed roller stall delay	ms	500	500	500	500	500
Swing frame cylinder target opening position	%	60	60	60	60	60
Swing frame auto up time at CFR restarting	ms	2000	2000	2000	2000	2000
Swing frame auto down time for CFR clamping	ms	1000	1000	1000	1000	1000
Main motor speed	rpm	500	500	500	500	500
Main motor current for SF clamping	A	440	440	440	380	330
Main motor current gap for SF clamping	A	40	40	40	40	40
Mill rotor speed for SF clamping	rpm	460	460	460	460	460
Mill rotor speed gap for SF clamping	rpm	15	15	15	15	15
Cyclone fan target speed	%	70.0	70.0	70.0	70.0	70.0
Cascade fan target speed	%	85.0	85.0	85.0	85.0	85.0
Cascade Bag Filter fan target speed	%	80.0	80.0	80.0	80.0	80.0
Transport Conveyor C1A Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C1B Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C2 Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C3A Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C3B Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C4A Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C4B Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C5A Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Transport Conveyor C5B Belt target speed	%	100.0	100.0	100.0	100.0	100.0
Light Fraction Conveyor CD1 Belt target speed	%	60.0	60.0	60.0	60.0	60.0
Non ferrous Conveyor NF3 Belt target speed	%	60.0	60.0	60.0	60.0	60.0

Slika 2.1 Tablica koja služi za konfiguraciju načina rada samog postrojenja

Kao što je vidljivo na slici, prvi nedostatak je mogućnost prikaza isključivo ograničenog broja recepata (u ovom slučaju 5). U praksi, to je često nedovoljno te je potrebno relativno često dodavati nove recepte. Ne postoji forma unutar aplikacije za dodavanje novih recepata, nego se recepti uvoze iz vanjske datoteke – što također predstavlja nedostatak. U prilogu 1 nalazi se slika koja prikazuje *Delay* zapise. Navedena stranica omogućuje prikazivanje zapisa vremenskih intervala prestanka rada postrojenja iz bilo kojeg razloga. Podaci su generirani na način da se prate točno određeni parametri unutar blokova podataka PLC-a koji služe za indikaciju prestanka rada samog postrojenja. U trenutcima kada sustav prepozna prestanak rada postrojenja pokreće se brojač koji bilježi vrijeme do ponovnog početka rada postrojenja. Na taj način konstruiran je „prazni“ *delay* zapis koji se prikazuje na samoj stranici. Dodatna funkcionalnost je da sami operateri mogu uređivati pojedini *delay* zapis na način da mu dodaju generalnu grupaciju razloga prestanka rada te same detalje razloga prestanka rada. Nadalje, operateri imaju mogućnost filtriranja podataka po raznim parametrima, ručnog dodavanja *delay* zapisa te izvoza svih *delay* zapisa.

U prilogu 2 nalazi se slika koja prikazuje *Production* zapise. Navedena stranica omogućuje prikazivanje *Production* zapisa čiji parametri pomažu procijeniti uspješnost (kvalitetu) rada određene proizvodne linije i postrojenja općenito. *Production* zapis čini skup navedenih parametara u jednom ciklusu reciklaže. Podaci su generirani pomoću različitih senzora koji se nalaze duž proizvodne linije. Kao što je to slučaj kod *delay* zapisa, operateri imaju mogućnost filtriranja podataka, izvoza podataka i sl.

Zaključno, nedostatci navedene aplikacije su:

1. Ograničenost pri dugotrajnom arhiviranju proizvodnje.
2. Ograničenost pri pohrani različitih konfiguracija i načina rada samog postrojenja (maksimalno 5).
3. Nepouzdan način dodavanja novih konfiguracija i načina rada samog postrojenja (dodaju se pomoću vanjske datoteke, pri čemu ne postoji provjera ispravnosti pojedinog konfiguracijskog parametra – direktno upisivanje tih parametara u blokove podataka PLC-a može prouzrokovati nepravilan rad ili čak materijalnu štetu).
4. Nemogućnost automatskog prilagođavanja dizajna veličini ekrana/zaslona uređaja (aplikacija nije responzivna).
5. Relativno kompliciranа migracija cjelokupne aplikacije na druge (slične) projekte.

3. IZRADA WEB APLIKACIJE ZA SUSTAV GOSPODARENJA ŽELJEZNIM OTPADOM

Razvoju Web aplikacije za sustav gospodarenja željeznim otpadom prethodi niz radnji koji se može podijeliti u nekoliko cjelina:

1. Analiza postojećeg sustava za gospodarenje željeznim otpadom (engl. *Recycling Process*) te uviđaj potrebe za uvođenjem novog informacijskog sustava – Web aplikacije.
2. Pripremanje popratne tehničke dokumentacije
   - definiranje podataka za razmjenu unutar blokova podataka PLC-a.
3. Projektiranje baze podataka za pohranu informacija te implementacija komunikacijskog servisa za čitanje/pisanje iz blokova podataka PLC-a.
4. Izrada web aplikacije za sustav gospodarenja željeznim otpadom.

Različiti strojevi poput glodalica, preša i sortirki upravljane su PLC-ovima te čine postrojenje zaduženo za gospodarenje željeznim otpadom. Za takav sustav postoje različite konfiguracije (načini rada) samog postrojenja.

Skup navedenih konfiguracijskih parametara predstavlja tzv. recept. Web aplikacija mora sadržavati jednostavno sučelje za manipulaciju receptima (dodavanje novih recepata te uređivanje i brisanje postojećih recepata). Dodatno, operateri koji upravljaju takvim postrojenjima i sustavima imaju potrebu za uvid u parametre (uglavnom dobivene pomoću različitih senzora koji se nalaze na samoj procesnoj liniji) koji im pomažu procijeniti uspješnost (kvalitetu) rada određene proizvodne linije i postrojenja općenito. Skup takvih parametara (informacija) zove se *Production* zapis. S druge strane, za potpunu sliku o radu određene procesne linije i postrojenja općenito potrebno je imati informacije o vremenskim intervalima u kojima je došlo do prekida rada postrojenja iz bilo kojeg razloga (slučajnog ili planiranog). Takav skup parametara (informacija) zove se *Delay* zapis.

U tablici 3.1 detaljno su prikazani zahtjevi za funkcionalnostima Web aplikacije koji proizlaze iz prethodno navedenih nedostataka postojeće aplikacije.

Tablica 3.1 Zahtjevi za funkcionalnostima Web aplikacije

R. br.	Zahtjev	Opis
1.	Sustav autentifikacije korisnika	- predefinirani korisnici (administrator i operater) - implementacija administratorskih ovlasti (dozvola)
2.	Upravljanje receptima	- dodavanje novih recepata - uređivanje postojećih recepata - brisanje recepata - prikaz liste recepata (uz mogućnost filtriranja podataka) - mogućnost dodavanja prioriteta pojedinim receptima
2.1	Definiranje intervala za pojedine parametre u receptima	- dozvoljeno samo administratorima - definiranje minimalne i maksimalne vrijednosti (intervala) u kojem se svaki konfiguracijski parametar mora nalaziti
2.2	Stvarnovremenska verifikacija podataka prilikom dodavanja novog ili uređivanja postojećeg recepta	- primjenjujući funkcionalnost iz točke 2.1 osigurati stvarnovremensku verifikaciju podataka (prilikom unosa novog ili uređivanja postojećeg recepta svi parametri moraju biti unutar svojih pre-definiranih intervala)
3.	*Production* i *delay* zapisi	- omogućiti prikaz - omogućiti filtriranje podataka - omogućiti izvoz podataka u *Excel* (reportistika)
4.	Omogućiti stvarnovremensko strujanje komunikacijskog zapisnika	- komunikacijski servis za čitanje/pisanje u blokove podataka PLC-a generira komunikacijski zapisnik za koji je potrebno omogućiti stvarnovremensko strujanje
5.	Responzivnost	- omogućiti responzivni dizajn cjelokupne Web aplikacije

Zahtjevi navedeni u tablici predstavljaju inicijalne zahtjeve koji moraju biti postavljeni prilikom razvoja ovakvog tipa projekta. Uglavnom su usmjereni ka tome da se krajnjim korisnicima (operaterima) maksimalno pojednostavi i olakša upotreba same aplikacije, a samim time i kontrola te nadzor nad procesom. Web aplikacija kao takva predstavlja jedinu „dodirnu točku“ krajnjeg korisnika (operatera) i cjelokupnog procesa, tj. sustava.

3.1. Definiranje podataka za razmjenu između servisne aplikacije i blokova podataka PLC-a

Nakon analize postojećeg sustava za gospodarenje željeznim otpadom te nakon detaljnog definiranja zahtjeva za funkcionalnostima Web aplikacije potrebno je definirati podatke za razmjenu. Podaci koji se razmjenjuju između servisne aplikacije i blokova podataka PLC-ova definirani su unutar tzv. *LanDef-a*. Neke od osnovnih informacija koje su sadržane u *LanDefu* su:

- izvor (naziv PLC-a),
- identifikacijski broj bloka podataka,
- ukupna veličina bloka podataka (u bajtima) te
- popis podataka koji čine blok podataka.

Svaki podatak unutar bloka podataka sadrži informacije o adresi, opisu, tipu podatka, minimalnoj i maksimalnoj vrijednosti itd. Na taj način konstruirane su tzv. poruke koje mogu biti u 2 smjera, točnije:

- poruke usmjerene od strane PLC-a ka servisnoj aplikaciji te
- poruke usmjerene od strane servisne aplikacije ka PLC-u.

Tablica 3.2 Primjer poruke unutar *LanDefa*

Address	*Description*	*Fmt*	*Size*	*Min*	*Max*	*Unit*	*Remarks*
	CIRCULAR BUFFER HEADER						
DBW 0000	Change Counter	IN					
DBW 0002	Buffer Pointer	IN		1	5		actual array position
DBW 0004	Maximum Number of Elements	IN		5	5		number of array positions
	CIRCULAR BUFFER DATA STRUCTURE		5				5 RECORDS
DBD 0012	Input Coil L2 Identifier	DI					
DBB 0016:20	Input Coil L1 Identifier	MS	20				
	CIRCULAR BUFFER FOOTER						
DBW 0552	Auxiliary Counter	IN					

Kao što je prikazano tablicom 3.2 poruka je podijeljena u 3 segmenta:

1. Zaglavlje poruke
2. Podatkovni dio poruke
3. Podnožje poruke

Zaglavlje i podnožje poruke čine određeni strukturalni elementi koji su zaduženi za ispravnu komunikaciju te pomažu pri identificiranju promjena u podatkovnom dijelu poruke. Više o načinima razmjene podataka nalazi se u poglavlju 3.2.

Cjelokupan *LanDef* nalazi se u digitalnom formatu na CD-u.

3.2. Pravila pri razmjeni podataka između servisne aplikacije i blokova podataka PLC-a

Određene tehnološki značajne varijable poput sile, brzine i debljine određenih materijala zahtijevaju nadziranje i kontrolu u vrlo kratkom vremenskom intervalu (npr. 100 *ms*).

Problemi koji se javljaju pri procesu dohvaćanja podataka su:

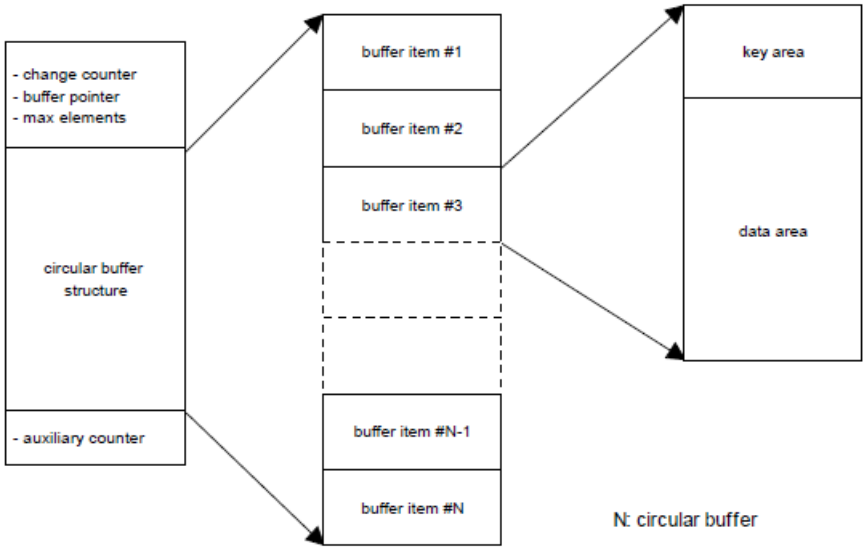
- kontinuiranost komunikacije (kratak vremenski interval u kojem je potrebno obaviti proces dohvaćanja i obrade podataka) te
- mogućnost nepovratnog gubitka određenog dijela podataka.

Da bi se to izbjeglo potrebno je osmisliti uniforman način razmjene podataka. Najčešće se koriste 2 načina:

1. Kružni spremnik.
2. Podatkovni spremnik.

3.2.1. Kružni spremnik

Kružni spremnik (engl. *Circular buffer*) predstavlja polje struktura koje se koriste za komunikaciju usmjerenu od strane PLC-a ka servisnoj aplikaciji. Struktura kružnog spremnika prikazana je na slici **Error! Reference source not found.**



Slika 3.1 Struktura kružnog spremnika

N predstavlja dimenziju kružnog spremnika (npr. 10). U zaglavlju kružnog spremnika nalaze se informacije o brojaču promjene (engl. *Change counter*) koji služi za prepoznavanje promjene u kružnom spremniku, o pokazivaču na spremnik (engl. *Buffer pointer*) te o dimenziji kružnog spremnika. U podatkovnom dijelu kružnog spremnika nalazi se polje struktura pri čemu svaka struktura sadrži identifikacijsku oznaku te sami podatkovni dio. Podnožje kružnog spremnika čini pomoćni brojač (engl. *Auxiliary counter)* koji služi za „potvrdu promjene“ u kružnom spremniku.

Ideja kružnog spremnika je da se logikom definiranoj na strani PLC-a na točno određen način ažuriraju podaci te da se logikom definiranoj na strani servis aplikacije isti podaci pravilno dohvate, pročitaju i obrade. U slučaju da je potrebno upisati neki novi podatak logika na strani PLC-a mora slijediti navedeni postupak:

1. Ažurirati brojač promjene u zaglavlju spremnika.
2. Dodati novu strukturu podataka u kružni spremnik.
3. Ažurirati pokazivač na spremnik (postaviti ga na vrijednost identifikacijske oznake prethodno dodane strukture podataka).
4. Ažurirati pomoćni brojač (na vrijednost brojača promjena) u podnožju spremnika.

Brojači u zaglavlju i podnožju su inkrementalni brojači koji se resetiraju kada dođu do maksimalne vrijednosti, npr. 65 536 (2^{16}) ili 4 294 967 295 (2^{32}).

Poštujući navedena pravila kružnog spremnika znatno se olakšava i ubrzava proces prepoznavanja promjena u određenoj strukturi podataka, te se smanjuje mogućnost gubitka podataka. Servis aplikacija pomoću brojača promjene i pomoćnog brojača nedvojbeno može prepoznati promjenu i to na slijedeći način:

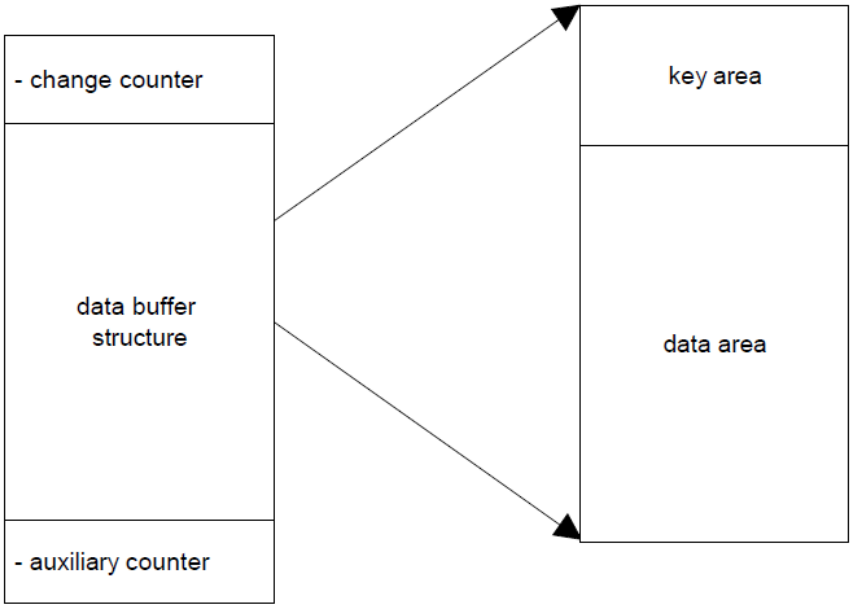
1. Potrebno je ciklički pratiti vrijednost brojača promjene i pomoćnog brojača te spremati posljednje očitane vrijednosti.
2. U slučaju da se u određenoj iteraciji cikličkog praćenja vrijednosti zaključi da je došlo do promjene vrijednosti (npr. ukoliko je posljednja očitana vrijednost brojača promjene bila 45, a očitana vrijednost brojača promjene u trenutnoj iteraciji je 46) to znači da je za nedvojbeno prepoznavanje promjene potrebno provjeriti još samo vrijednost pomoćnog brojača – ukoliko su te dvije vrijednosti jednake znači da došlo do promjene vrijednosti.

Navedeni brojači omogućavaju servisnoj aplikaciji da nedvojbeno prepozna promjenu cjelokupnog seta podataka uz praćenje isključivo samo dvije vrijednosti, dok bi bez brojača servisna aplikacija morala pratiti cjelokupni set podataka što bi znatno usporavalo njen rad - u nekim slučajevima (npr. kratkog vremenskog intervala za čitanje ili velikog broja različitih setova podataka za praćenje) servisna aplikacija ne bi bila u stanju u tako kratkom vremenskom intervalu procesuirati veliki broj podataka.

U slučaju da je u određenom trenutku vrijednost brojača promjene različita od vrijednosti pomoćnog brojača, to znači da je proces pisanja u blokove podataka PLC-a još uvijek u tijeku. Servisna aplikacija u tom slučaju neće napraviti ništa, te je na taj način spriječeno čitanje pogrešnih vrijednosti. U slučaju da su vrijednosti brojača promjene i pomoćnog brojača jednake, tada je potrebno pročitati vrijednost pokazivača na spremnik te ovisno o njemu pročitati podatke strukture na koju on usmjerava. Ukoliko servisna aplikacija iz nekog razloga propusti očitati pravilnu promjenu brojača promjene i pomoćnog brojača, te ukoliko se logikom na PLC strani ukaže potreba za upisati nove podatke – oni će biti upisani na iduće mjesto u polju struktura, tako da će i prethodni podaci ostati u memoriji te mogu biti retroaktivno procesuirani od strane servis aplikacije.

3.2.2. Podatkovni spremnik

Podatkovni spremnik (engl. *Data buffer*) predstavlja strukturu koja se koristi za komunikaciju usmjerenu od strane servisne aplikacije ka PLC-u. Struktura podatkovnog spremnika prikazana je na slici 3.3.2

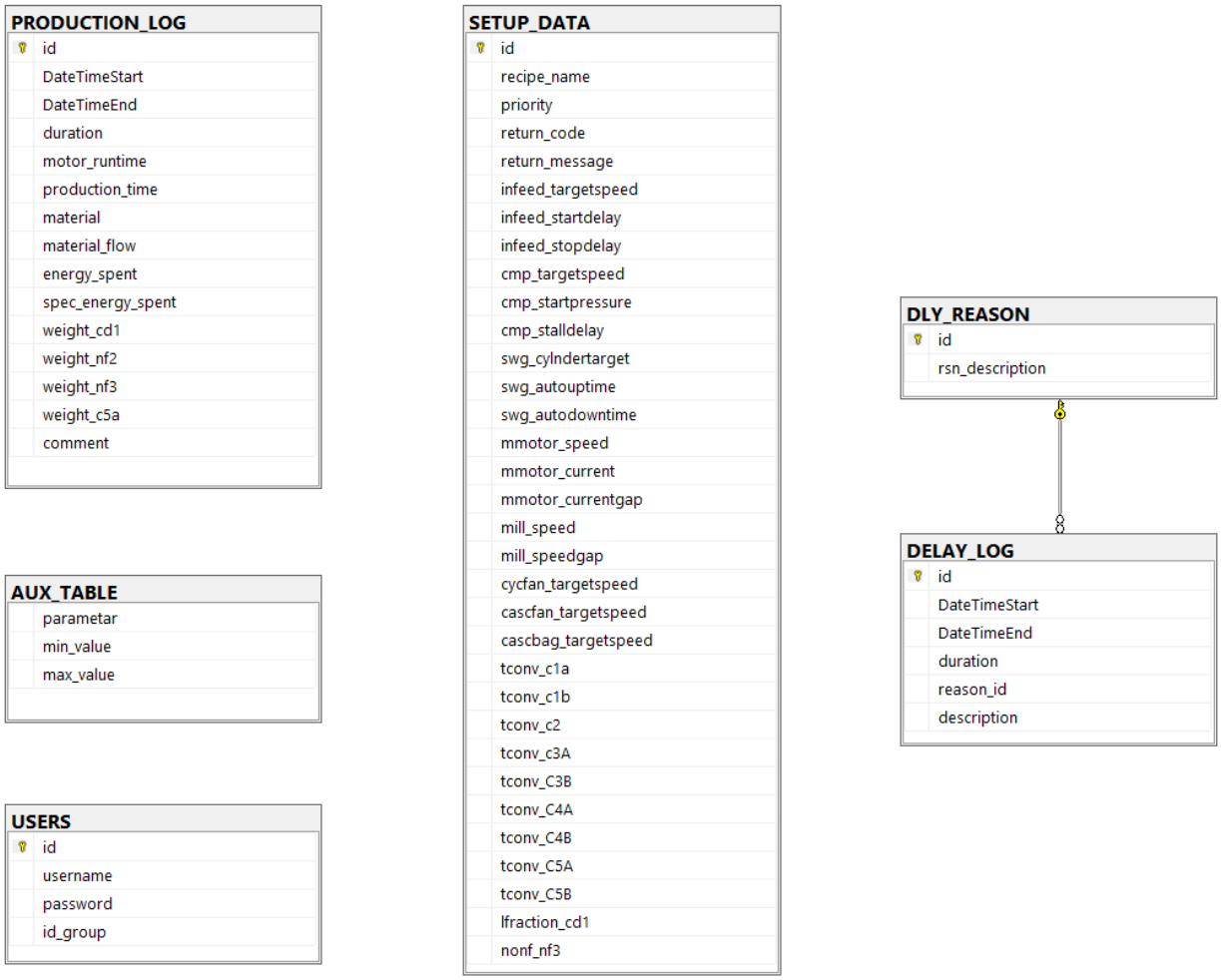


Slika 3.2 Struktura podatkovnog spremnika

Struktura podatkovnog spremnika jednostavnija je od strukture kružnog spremnika. Za informaciju usmjerenu od strane servisne aplikacije ka PLC-u nije potrebno polje struktura, nego samo jedan element (tj. samo jedna struktura). No, mehanizam brojača promjene i pomoćnog brojača je identičan. U ovom slučaju, zadaća servisne aplikacije je, prema pravilima i načinima navedenim u prethodnom koraku, inkrementirati brojače i ažurirati podatke. Razlog jednostavnosti podatkovnog spremnika je u činjenici da rijetko postoji potreba za brzom izmjenom podataka umjerenom od strane servisne aplikacije ka PLC-u.

3.3. Projektiranje baze podataka

Nakon definiranja podataka za razmjenu između servisne aplikacije i blokova podataka PLC-a potrebno je projektirati bazu podataka. Za potrebe diplomskog rada koristi se relacijska SQL (engl. *Structured Query Language*) baza podataka. Na slici **Error! Reference source not found.** prikazan je dijagram baze podataka.



Slika 3.3 Dijagram baze podataka

Bazu podataka čini 6 entiteta:

1. Entitet *PRODUCTION_LOG*,
2. Entitet *DELAY_LOG*,
3. Entitet *AUX_TABLE*,
4. Entitet *USERS*,
5. Entitet *DLY_REASON* te

6. Entitet *SETUP_DATA*.

Bazu podataka zajednički koriste servisna i Web aplikacija. Entitet *USERS* koristi isključivo Web aplikacija budući da se u njemu nalaze informacije (atributi) o korisnicima za potrebe prijave korisnika u Web aplikaciju. Entitet *AUX_TABLE* također koristi isključivo Web aplikacija budući da se u njemu nalaze informacije o minimalnoj i maksimalnoj vrijednosti pojedinog parametra iz recepta.

3.4. Razvoj komunikacijskog softvera (servisne aplikacije)

Nakon projektiranja baze podataka slijedi razvoj komunikacijskog softvera (servisne aplikacije). Servisna aplikacija razvijena je u C# programskom jeziku unutar *Visual Studio 2019* razvojnog okruženja. Za razvijanje servisnih aplikacija unutar C# postoje različite predefinirane metode. Neke od njih su:

1. Metoda *onStart* – definira akcije koje će biti izvršene pri pokretanju izvođenja servisa.
2. Metoda *onStop* – definira akcije koje će biti izvršene pri zaustavljanju izvođenja servisa.
3. Metoda *onTimer* – definira akcije koje će biti izvršene nakon isteka definiranog vremena brojača.

Za komunikaciju servisne aplikacije sa PLC-om koristi se Siemens S7.Net biblioteka.

3.4.1. Siemens S7.Net

Siemens S7.Net je .NET biblioteka koja se koristi za komunikaciju Siemensovih S7 PLC-ova sa .NET aplikacijama. PLC-ovi koji su podržani S7.Net bibliotekom su: S7-200, S7-300, S7-400, S7-

Linija ***Kod***

```
1:      Plc plc;
2:      public PlcConnection(string ipAddress)
3:      {
4:         plc = new Plc(CpuType.S71500, ipAddress, 0, 1);
5:      }
```

1200 te S7-1500. Uvjet korištenja Siemens S7.Net biblioteke je *Ethernet* konekcija PLC-a sa

Slika 3.4 Kôd za kreiranje instance PLC-a

računalom. Na slici 3.4 prikazan je kod za kreiranje instance PLC-a.

Za potrebe diplomskog rada koristi se PLC Siemens S7-1500. Za ostvarivanje i prekid konekcije koriste se metode *Open()* i *Close()* prikazane na slici 3.5.

Linija ***Kod***

```
1:      plc.Open();
2:      // logika programa
3:      plc.Close();
```

Slika 3.5 Kôd ostvarivanje i prekid konekcije

Za čitanje i pisanje u blokove podataka PLC-a na raspolaganju je nekoliko metoda. Najčešće korištene metode su:

1. *Read* – koristi se za čitanje jedne varijable iz bloka podataka PLC-a.
2. *Write* – koristi se za upisivanje jedne varijable u blok podataka PLC-a
3. *ReadClass* – koristi se čitanje korisnički definiranog tipa podataka iz bloka podataka PLC-a
4. *WriteClass* – koristi se za upisivanje korisnički definiranog tipa podataka u blok podataka PLC-a

Na slici 3.6 prikazano je korištenje metoda *Read* i *Write*.

Linija ***Kod***

```
1:      int variableToRead, variableToWrite = 5, DB = 1, startAddress = 2;
2:      variableToRead = plc.Read(DataType.DataBlock, DB, startAddress,
3:                                VarType.Int, 1);
4:      plc.Write(DataType.DataBlock, DB, startAddress, variableToWrite,
5:                VarType.Int, 1);
```

Slika 3.6 Primjer korištenja metoda Read i Write

Metoda *ReadClass* ima razne prednosti budući da je izvođenje metode znatno brže od čitanja varijabli jednu po jednu pomoću metode *Read*. Kod razvijanja servisnih aplikacija izuzetno je bitno obratiti pozornost na optimizaciju vremena izvođenja programa s obzirom da se brojač okida u relativno kratkom vremenskom intervalu. U tom vremenskom intervalu okidanja potrebno je obaviti cijeli niz operacija u nekoliko smjerova (učitati podatke iz baze podataka, upisati ih u blokove podataka PLC-a ili obrnuto). Iz tog razloga koriste se metode *ReadClass* i *WriteClass* kod kojih treba obratiti pozornost na pravilno modeliranje korisnički definiranih tipova podataka (klasa) koje trebaju „1 na 1“ odgovarati podacima unutar blokova podataka PLC-a. Ukoliko se to ne osigura, podaci se neće pravilno čitati iz blokova podataka PLC-a, ali isto tako neće se pravilno

niti upisivati u iste. Na slici 3.7 prikazano je korištenje metode *ReadClass*. Metoda *WriteClass* koristi se analogno kao i metoda *ReadClass*.

Linija Kod

```
1:    DB = 1;
2:    public class L2L1_SetupData
3:     {
4:        public int id { get; set; }
5:        public byte[] recipeName { get; set; }
6:        public int infeed_targetspeed { get; set; }
7:        public int infeed_startDelay { get; set; }
8:        public int infeed_stopDelay { get; set;}
9:      }
10:   L2L1_SetupData setupData = new L2L1_SetupData();
11:   plc.ReadClass(L2L1_SetupData, DB);
```

Slika 3.7 Primjer korištenja metode *ReadClass*

Navedena metoda *ReadClass* je efikasna kada je riječ o čitanju podataka iz određenog bloka podataka PLC-a, no, kada je potrebno u kratkom vremenskom periodu čitati podatke iz više različitih blokova podataka PLC-a ta metoda više nije efikasna. Za potrebe kontinuiranog praćenja vrijednosti brojača promjene i pomoćnog brojača koristi se metoda *ReadMultipleVars* koja u jednom zahtjevu čita podatke iz nekoliko različitih blokova podataka PLC-a. Na slici 3.8 prikazan je primjer korištenja *ReadMultipleVars* metode. Metoda kao parametar prima listu tipa *DataItem*, pri čemu je *DataItem* klasu čine atributi o bloku podataka, tipu varijable itd.

Linija Kod

```
1:    using S7.Net.Types;
2:    List <DataItem> multipleVarList = new List<DataItem>();
3:    static List<DataItem> listToRead = new List<DataItem>()
4:      {
5:        new DataItem
6:         {
7:           Count = 1, DataType = DataType.DataBlock, DB = 1,
8:           StartByteAdr = 0, VarType = Vartype.Int
9:         },
10:       new DataItem
11:        {
12:          Count = 1, DataType = DataType.DataBlock, DB = 2,
```

```
13:          StartByteAdr = 0, VarType = Vartype.Int
14:        }
15:      }
15:   multipleVarList = plc.ReadMultipleVars(listToRead);
```

Slika 3.8 Primjer korištenja ReadMultipleVars metode

Za provjeru konekcije i dostupnosti PLC-a koriste se metode *IsAvailable* te *IsConnected.* Navedene metode mogu nam pomoći u razvoju komunikacijskog zapisnika. Na slici 3.9 nalazi se primjer korištenja metoda *IsAvailable* te *IsConnected*.

Linija Kod

```
1:    if(plc.IsAvailable == false)
2:     {
3:       // logika - PLC nije dostupan
4:     }
5:    if(plc.IsConnected == false)
6:     {
7:       // logika - nije aktivna konekcija sa PLC-om
8:     }
```

Slika 3.9 Primjer korištenja metoda isAvailable te isConnected

3.4.2. Konekcija s bazom podataka

Unutar baze podataka nalaze se svi podaci koje je potrebno dohvatiti pomoću servisne aplikacije. Način na koji servisna aplikacija komunicira sa bazom podataka je pomoću biblioteka za rad sa bazama podataka. Konekcija se ostvaruje uz pomoć *MySqlConnection* klase. Za uspostavljanje konekcije ključno je definirati tzv. konekcijski string koji sadrži uobičajene podatke poput:

- naziva servera na kojem se nalazi baza podataka,
- naziva baze podataka na koju se potrebno spojiti te
- korisničkog imena i lozinke za spajanje na bazu podataka.

Na slici 3.10 nalazi se primjer konekcijskog stringa.

Linija ***Kod***

```
1:      string server   = "localhost";
2:      string database = "dbName";
3:      string username = "root";
4:      string password = "";
5:      string constr   = "SERVER=" + server + ";" + "DATABASE=" +
6:                         database + ";" + "UID=" + username + ";"
7:                         + "PASSWORD=" + password + ";";
```

Slika 3.10 Primjer konekcijskog stringa

Za primjer uzmimo metodu koja dohvaća konfiguracijske parametre s obzirom na identifikacijsku oznaku (engl. *ID*) recepta. Kao što je vidljivo na slici 3.11, potrebno je definirati SQL upit (engl. *Query*) koji se izvršava nad bazom podataka. Nakon definiranja upita, kreiramo objekt `connection` klase `MySqlConnection` kojem pridružujemo prethodno definirani konekcijski string. U idućem koraku definiramo objekt klase `MySqlCommand`, kojem prdružujemo prethodno definirani SQL upit te objekt `connection`. Nakon toga, moguće je otvoriti samu konekciju, te preko čitača (engl. *Reader*) dohvatiti vrijednosti iz baze podataka.

Linija ***Kod***

```
1:      public L2L1_SetupData PrepareSetupParameters(int id)
2:      {
3:        string query = string.Format("SELECT id, recipe_name,
4:                                      mill_speed, lfraction_cd1
5:                                      FROM setupdata WHERE id ={0}", id);
6:        L2L1_SetupData itm = new L2L1_SetupData();
7:        using (MySqlConnection connection = new MySqlConnection(constr))
8:         {
9:          MySqlCommand command = new MySqlCommand(query, connection);
10:         Connection.Open();
11:         MySqlDataReader reader = command.ExecuteReader();
12:         while(reader.Read())
13:          {
14:           itm.id = Convert.ToInt32(reader["id"]);
15:           itm.recipe_name = Convert.ToString(reader["recipe_name"]);
16:           itm.mill_speed = Convert.ToInt32(reader[„mill_speed"]);
17:           itm.lfraction_cd1 = Convert.ToInt32(reader["lfraction_cd1"]);
18:          }
19:        }
20:     }
```

Slika 3.11 Primjer metode za dohvaćanje konfiguracijskih parametara

3.4.3. Razvoj komunikacijskog zapisnika

Komunikacijski zapisnik (engl. *Logger*) standardni je zahtjev za servisne aplikacije. S obzirom da postoji veći broj međusobnih konekcija različitih entiteta (konekcija servisne aplikacije i PLC-a te konekcija servisne aplikacije i baze podataka) te postoji realna mogućnost prekida nekih od konekcija. Servisna aplikacija mora imati mehanizam prepoznavanja prekida konekcije sa bazom podataka ili PLC-om (ili prekida obje konekcije). Nakon toga slijedi razvoj komunikacijskog zapisnika koji zapisuje određene poruke o greški (ili neke druge bitne poruke koje ne moraju biti poruke o greški) u *.txt* datoteku.

Na slici 3.12 prikazan je kod koji predstavlja komunikacijski zapisnik.

Linija Kod

```
1:    public static class Logger
2:     {
3:       public static void WriteLog(string message)
4:        {
5:          string logPath = "..\log.txt" // putanja do datoteke
6:          using (var stream = new FileStream(logPath, FileMode.Append,
7:                 FileAccess.Write,  FileShare.ReadWrite))
8:             {
9:               using(var writer = new StreamWriter(stream))
10:                {
11:                   writer.WriteLine($"[{DateTime.Now}]: {message} ");
12:                   writer.Flush();
13:                }
14:            }
15:        }
16:     }
```

Slika 3.12 Primjer razvoja komunikacijskog zapisnika

3.4.4. Osiguravanje pravilnog rada servisne aplikacije u nepredviđenim okolnostima

Glavni izazov pri razvoju servisnih aplikacija je osigurati pravilan rad aplikacije u slučaju nepredviđenih okolnosti kao što su npr. prekid konekcije sa bazom podataka ili prekid konekcije sa PLC-om. Osim toga, niti jedna situacija prilikom izvođenja servisne aplikacije ne smije prouzrokovati nepravilan rad servisne aplikacije ili čak njen prekid rada. Pomoću komunikacijskog zapisnika potrebno je prikazati različite poruke u slučaju identifikacije nepravilnog rada ili prekida konekcije između entiteta. Na taj način omogućava se lakša i brža identifikacija samog problema.

Slika 3.13 Kritične konekcije između entiteta

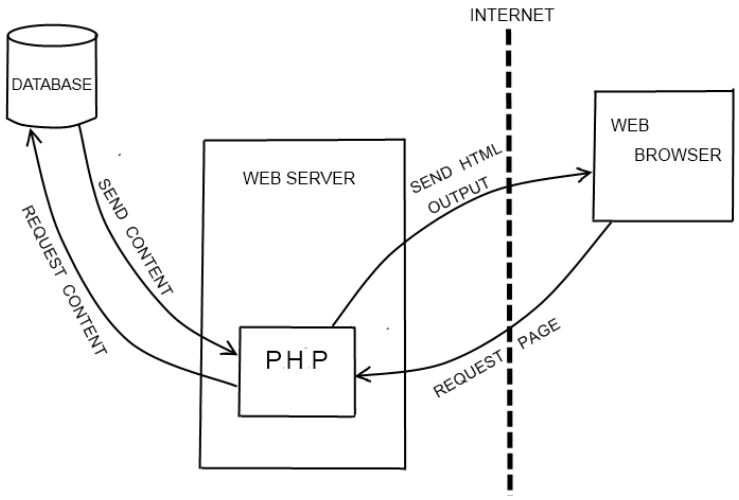
Kao što je vidljivo na slici 3.13, kritične su konekcije između servisne aplikacije i PLC-a te servisne aplikacije i baze podataka. Postoji više scenarija prekida konekcije, neki od njih mogu biti izrazito opasni te ih je stoga potrebno pravovremeno prepoznati i reagirati. Na primjer, razmotrimo scenarij u kojem je:

- konekcija servisne aplikacije i PLC-a aktivna te
- konekcija servisne aplikacije i baze podataka neaktivna.

Budući da servisna aplikacija ima mogućnost upisivanja podataka direktno u blokove podataka PLC-a ovaj scenarij može biti izrazito opasan budući da su podaci koji se upisuju zapravo parametri koji služe za konfiguraciju rada samog postrojenja (konfiguracijski parametri za različite mašine i sl.). Podaci koje servisna aplikacija upisuje u blokove podataka PLC-a se nalaze u bazi podataka, te ukoliko ne postoji konekcija servisne aplikacije i baze podataka ti se podaci ne mogu dohvatiti. U tom slučaju, izuzetno je važno prepoznati takvo stanje sustava kako servisna aplikacija ne bi upisala neke druge podatke (npr. bilo kakve slučajne ili remanentne podatke) u blokove podataka PLC-a te eventualno prouzrokovala materijalnu štetu.

3.5. Razvoj Web aplikacije

Nakon razvoja i cjelovitog testiranja rada servisne aplikacije potrebno je razviti Web aplikaciju pomoću koje će operateri imati mogućnost upravljanja i nadzora nad postrojenjem. Za razvoj Web aplikacije koristi se PHP (engl. *Hypertext Preprocessor*), HTML (engl. *The HyperText Markup Language*), Bootstrap te JavaScript. PHP u kombinaciji sa HTML-om omogućava dinamičko generiranje sadržaja koji se dohvaća iz baze podataka kako je prikazano na slici 3.14. *Bootstrap* pruža skup gotovih komponenata, stilova i alata koji omogućavaju razvoj responzivnih Web aplikacija, dok se JavaScript koristi za interaktivnost i animacije.



Slika 3.14 Način funkcioniranja PHP-a u kombinaciji sa HTML-om i bazom podataka [4]

Neki od najvažnijih koncepata koji su korišteni u razvoju Web aplikacije su:

1. Dinamičko prikazivanje podataka iz baze podataka (u obliku tablica).
2. Obrada formi.

Primjerice, dinamičko prikazivanje podataka u obliku tablica koristi se za prikaz liste recepata, *delay* te *production* logova itd. S druge strane, obrada formi koristi se kod dodavanja novih recepata, uređivanja postojećih itd.

Na slici 3.15 prikazan je kod za tablični prikaz liste recepata. Kao što je vidljivo u kodu, funkcija `mysqli_connect` služi za spajanje na bazu podataka. Dok se s pomoću funkcije `mysqli_query` izvodi upit nad bazom podataka. Nakon toga se `while` petljom dohvaća redak po redak iz baze podataka te se generiraju *HTML* elementi za tablični prikaz.

Linija Kod

```
1:    $conn = mysqli_connect($server, $username, $password, $database);
2:    $query = "SELECT id, recipe_name, priority FROM setupdata ORDER BY
3:              priority ASC";
4:    $result = mysqli_query($conn,$query);
5:    while($row = mysqli_fetch_assoc($result)) { ?>
6:      <td><?php echo $row["id"]; ?></td>
7:      <td><?php echo $row["recipe_name"]; ?></td>
8:      <td><?php switch($row["priority"]) {
9:                        case 1: echo 'Main'; break;
10:                       case 2: echo 'Normal'; break;
11:                       case 3: echo 'Low'; break; } ?>
12:     </td>
13:     <?php } ?>
```

Slika 3.15 Kod za tablični prikaz liste recepata

S druge strane, na slici 3.16 prikazan je kod za obradu forme za dodavanje *delay* zapisa. Uz pomoć PHP-a podaci poslani putem HTML obrasca su dohvaćeni te pomoću funkcije `mysqli_query` uneseni u bazu podataka.

Linija Kod

```
1:    if(isset($_POST['delay_log_ins']) && $_POST['delay_log_ins'] == 1){
2:       $datetimestart = $_POST['datetimestart'];
3:       $datetimeend = $_POST['datetimeend'];
4:       $result = mysqli_query($conn,$query);
5:       $delayreason = $_POST['dly_reason_select'];
6:       $comment = $_POST['comment'];
7:       $ins_query="INSERT INTO delay(DateTimeStart, DateTimeEnd,
8:                   duration, reason_id, description) VALUES
9:                   ('$datetimestart', '$datetimeend', '$duration',
10:                  '$delayreason', '$comment');
11:       mysqli_query($conn,$ins_query);
12:    }
```

Slika 3.16 Kod za obradu forme dodavanja *delay* zapisa

4. TESTIRANJE RADA WEB APLIKACIJE ZA SUSTAV GOSPODARENJA ŽELJEZNIM OTPADOM

Nakon razvoja modela baze podataka, servisne aplikacije i web aplikacije potrebno je testirati cjelokupno rješenje. Za testiranje cjelokupnog sustava koristi se *TIA Portal 17* (*Totally Integrated Automation Portal 17)* [5] - softverski paket razvijen od tvrtke Siemens koji čini sveobuhvatno okruženje s integriranim alatima za sve faze razvoja automatizacijskih projekata. Jedan od tih integriranih alata je i *NetToPlcSIM* koji pruža usluge simulacije stvarnog ponašanja PLC-a, uključujući provjeru ispravnosti logike, reakciju na ulazne signale i generiranje odgovarajućih izlaznih signala. Unutar *TIA Portala* potrebno je kreirati identične blokove podataka kako je to prethodno definirano *LanDef-om*. Nakon toga se uz pomoć prethodno spomenutih alata provodi se simulacija konekcije te provjerava ispravnost rada cjelokupnog sustava.

4.1. Testiranje funkcionalnosti servisne aplikacije

Za testiranje funkcionalnosti servisne aplikacije potrebno je servisnu aplikaciju instalirati kao *Windows* servis. To se radi na način da se pokrene naredbeni redak (engl. *Command Prompt*) te se navigira do mape gdje se nalazi izvršna datoteka servisne aplikacije (engl. *.exe file*) i pokrene naredba `sc create` na slijedeći način:

```
sc create L2DServis BinPath=D:\LC\WindowService/L2DServis.exe
```

Nakon uspješne instalacije, servis je moguće pokrenuti, zaustaviti ili obrisati pomoću sljedećih naredbi:

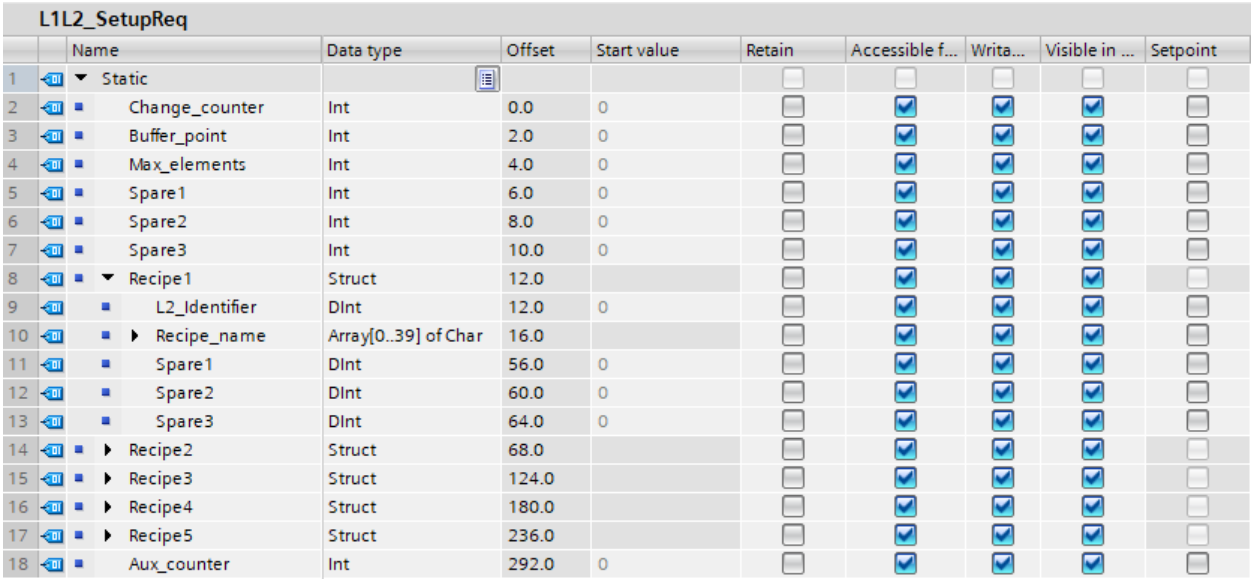
```
sc start TestService
sc stop TestService
sc delete TestService
```

Unutar *Services* aplikacije *Windowsa*, moguće je vidjeti popis aktivnih servisa te osnovne informacije o njima kako je prikazano na slici 4.1.

Korisnička usluga za Windows proslijeđene obavi...	This service hosts the Windows notification platform which provides support for local and push notifications. ...	Running	Automatic	Local System
Korisnički servis međuspremnika_75767	This user service is used for Clipboard scenarios	Running	Manual	Local System
KtmRm for Distributed Transaction Coordinator	Coordinates transactions between the Distributed Transaction Coordinator (MSDTC) and the Kernel Transaction...		Manual (Trig...	Network Service
L2DServis		Running	Automatic	Local System
Language Experience Service	Provides infrastructure support for deploying and configuring localized Windows resources. This service is star...		Manual	Local System
Link-Layer Topology Discovery Mapper	Creates a Network Map, consisting of PC and device topology (connectivity) information, and metadata descri...		Manual	Local Service

Slika 4.1 Popis aktivnih servisa unutar *Windows Services*

Idući korak je unutar razvojnog okruženja TIA Portal kreirati sve blokove podatka koji su definirani unutar *LanDefa*. Na slici 4.2. vidljiva je kreirana struktura poruke *L1L2_SetupRequest*.



L1L2_SetupReq

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static								
2	Change_counter	Int	0.0	0					
3	Buffer_point	Int	2.0	0					
4	Max_elements	Int	4.0	0					
5	Spare1	Int	6.0	0					
6	Spare2	Int	8.0	0					
7	Spare3	Int	10.0	0					
8	Recipe1	Struct	12.0						
9	L2_Identifier	DInt	12.0	0					
10	Recipe_name	Array[0..39] of Char	16.0						
11	Spare1	DInt	56.0	0					
12	Spare2	DInt	60.0	0					
13	Spare3	DInt	64.0	0					
14	Recipe2	Struct	68.0						
15	Recipe3	Struct	124.0						
16	Recipe4	Struct	180.0						
17	Recipe5	Struct	236.0						
18	Aux_counter	Int	292.0	0					

Slika 4.2 Struktura L1L2_SetupRequest poruke kreirana unutar TIA Portala

Nakon toga, potrebno je pomoću alata *NetToPLCsim* kreirati server pomoću kojeg se servisnoj aplikaciji omogućava spajanje na PLC. Nakon što je ostvarena konekcija, unutar TIA Portala potrebno se prebaciti u *online* način rada pri čemu je moguće u stvarnom vremenu pratiti i modificirati vrijednosti pojedinih parametara unutar strukture podatkovnog bloka. Na taj način moguće je testirati rad servisne aplikacije.

4.2. Testiranje funkcionalnosti Web aplikacije

U ovom potpoglavlju testiraju se funkcionalnosti Web aplikacije. Rezultati će biti prikazani pomoću slika Web aplikacije. S obzirom da je cjelokupna Web aplikacija responzivna, za svaku stranicu koja će biti prikazana u ovom potpoglavlju u prilogu će se, zbog preglednosti, pronaći ista stranica, ali u responzivnom obliku.

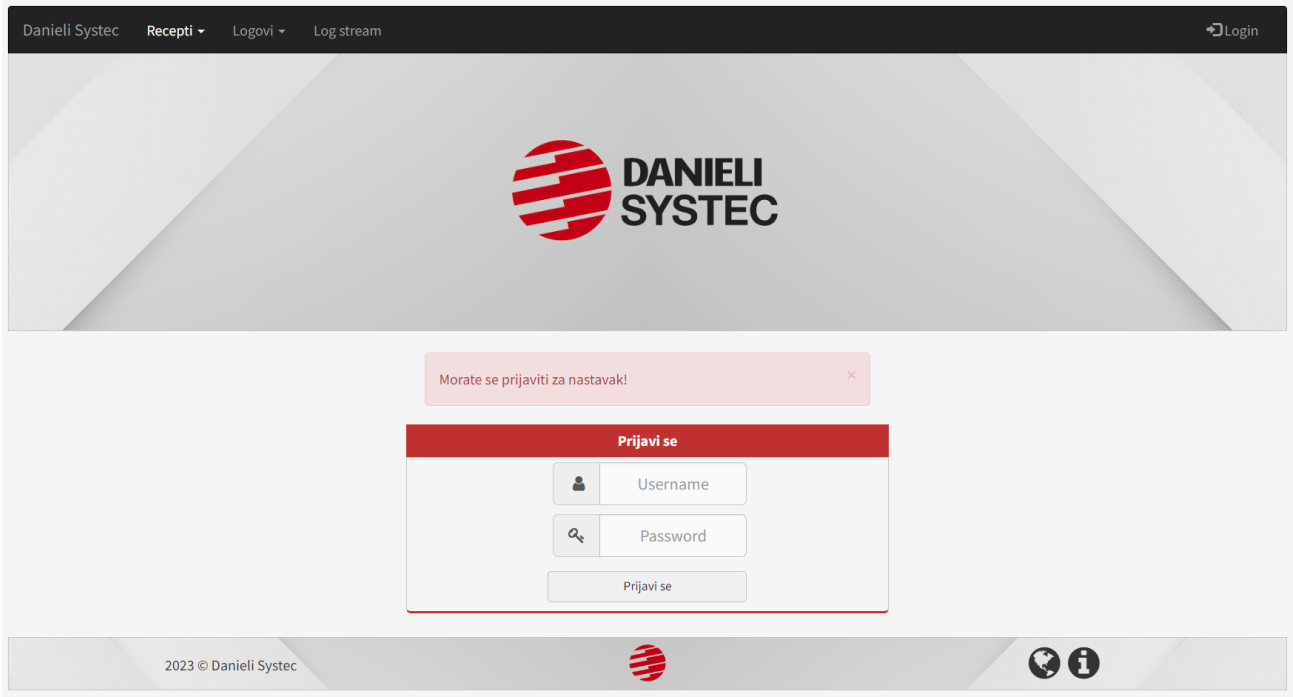
Testiranje funkcionalnosti Web aplikacije može se podijeliti u nekoliko cjelina:

1. Testiranje funkcionalnosti autentifikacije.
2. Testiranje funkcionalnosti manipulacije parametrima za konfiguraciju rada postrojenja (recepata).
3. Testiranje funkcionalnosti prikaza *Production* i *Delay* zapisa.
4. Testiranje funkcionalnosti stvarnovremenskog strujanja komunikacijskog zapisnika.

4.2.1. Testiranje funkcionalnosti autentifikacije

Temeljni mehanizam većine aplikacija je autentifikacija koja omogućava da samo ovlašteni korisnici imaju pristup podacima i funkcionalnostima Web aplikacije. Osim toga, autentifikacija omogućava pružanje različitih razina pristupa različitim korisničkim grupama. Kao što je definirano u zahtjevima za funkcionalnostima Web aplikacije, potrebno je imati pre-definirane korisnike (registracija korisnika treba biti onemogućena).

Na slici 4.3 prikazana je početna stranica za sve korisnike aplikacije koji se nisu prijavili. Kao što je vidljivo, korisnicima nije dozvoljen prikaz sadržaja sve dok se ne prijave.



Slika 4.3 Stranica „Login“

Pre-definirani korisnici su **sysadmin** (predstavlja administratora) te **sysoperator** (predstavlja operatera).

4.2.2. Testiranje funkcionalnosti manipulacije parametrima za konfiguraciju rada postrojenja

Najvažnija funkcionalnost Web aplikacije je sustav manipulacije parametrima za konfiguraciju rada postrojenja (sustav manipulacije receptima). Sastoji se iz nekoliko dijelova, neki od njih su:

- Prikaz liste recepata.
- Dodavanje novih recepata.
- Definiranje intervala za svaki od parametara koji čine pojedini recept.

Važnost sustava manipulacije parametrima za konfiguraciju rada postrojenja leži u činjenici da se upravo ti podaci spremaju u bazu podataka, te se preko servisne aplikacije upisuju direktno u blokove podataka PLC-a. Ukoliko podaci nisu verificirani i provjereni mogu direktno prouzročiti nepravilan rad ili čak materijalnu štetu.

Na slici 4.4 nalazi se prikaz liste parametara za konfiguraciju rada postrojenja (tablično).

Slika 4.4 Tablični prikaz liste parametara za konfiguraciju rada postrojenja

Informacije koje su prikazane su:

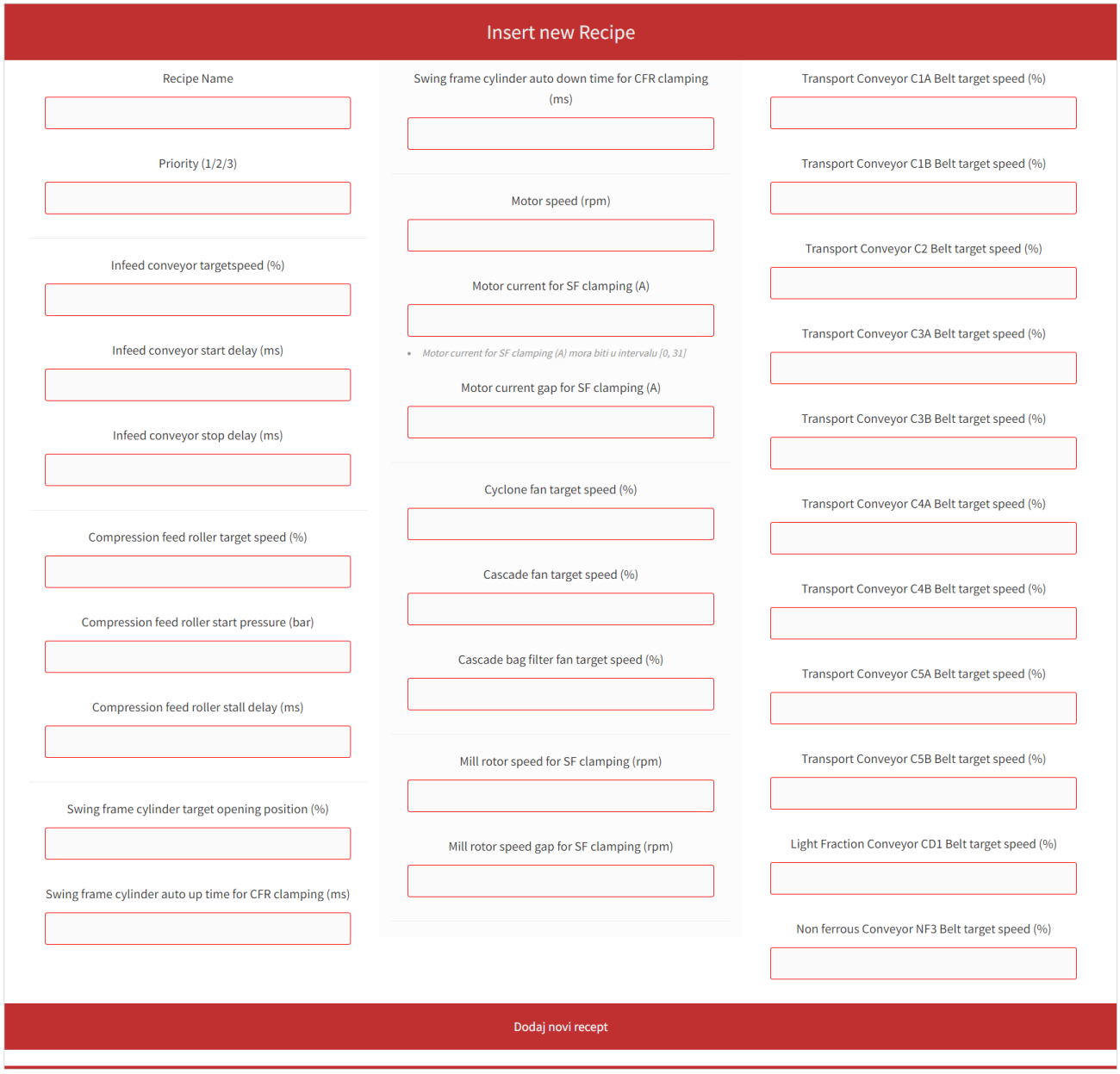
1. Jedinstvena identifikacijska oznaka recepta (engl. *Recipe ID*).
2. Opis pojedinog recepta (engl. *Description*).
3. Prioritet.
4. Postavke pojedinog recepta.

Jedinstvena identifikacijska oznaka recepta je dodijeljena automatski (automatsko inkrementiranje s obzirom na posljednje upisanu vrijednost u bazi podataka). Ostatkom informacija direktno upravlja korisnik. Informacije o jedinstvenoj identifikacijskoj oznaci recepta se zajedno sa opisom pojedinog recepta ciklički šalju u blokove podataka PLC-a kako bi *Level 1* strana imala informacije o dostupnim receptima. Ukupno se šalju informacije o maksimalno 20 recepata. Postupak dodjeljivanja prioriteta receptima ostvaruje se unutar postavki pojedinog recepta. Osim toga, postoji mogućnost filtriranja recepata s obzirom na to prikazuju li se na *Level 1* strani ili ne.

Razine prioriteta koje je moguće dodijeliti receptima su:

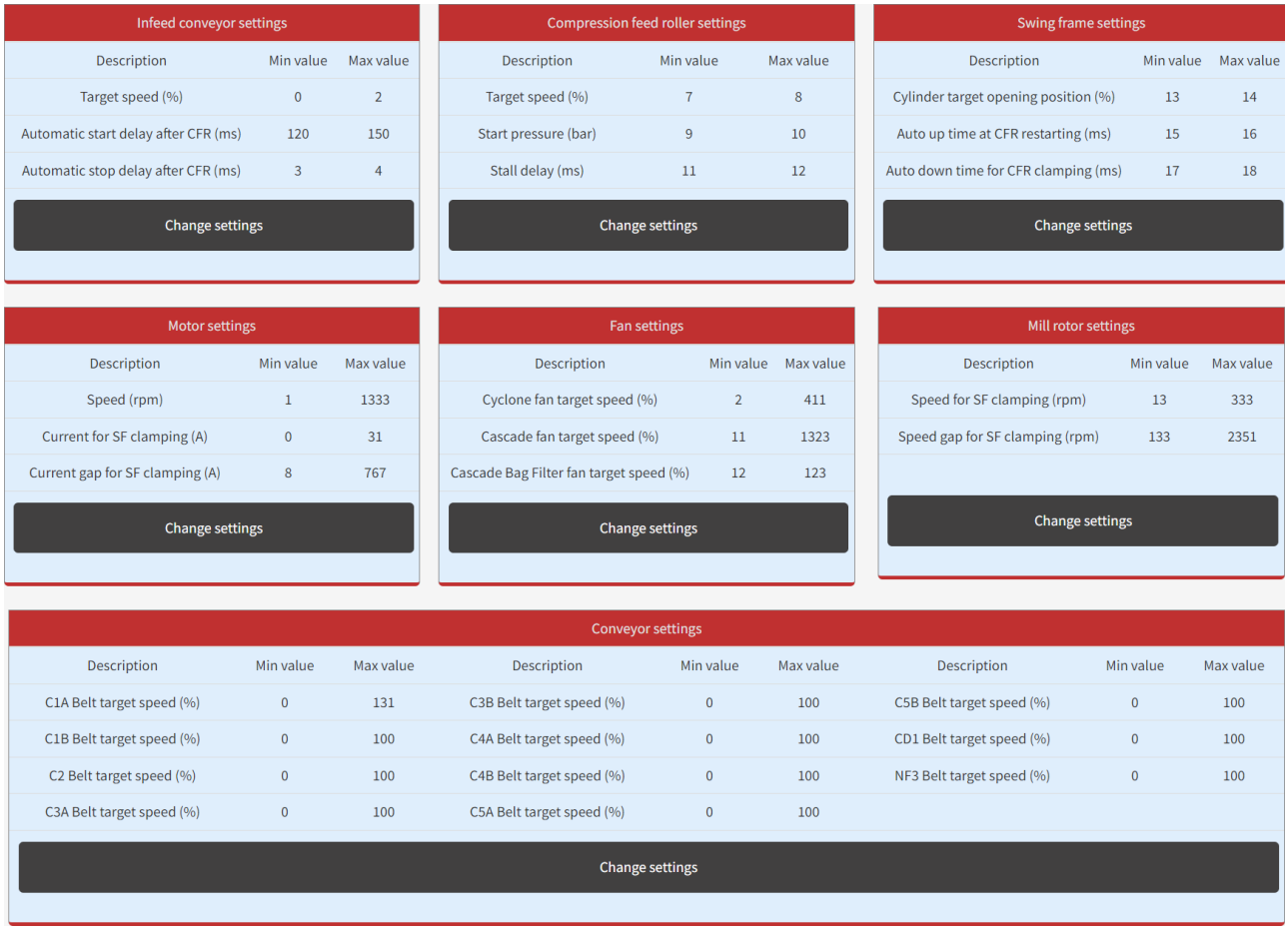
1. Glavni (engl. *Main*) prioritet.
2. Normalan (engl. *Normal*) prioritet.
3. Niski (engl. *Low*) prioritet.

Na slici 4.5 prikazana je forma za dodavanje novih recepata. Unutar forme moguće je dodati sve parametre za konfiguraciju rada postrojenja uz stvarnovremensku verifikaciju pojedinih parametara. Tek kada su svi parametri verificirani korisniku se omogućuje dodavanje novog recepta.



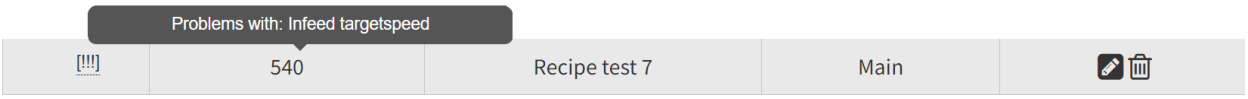
Slika 4.5 Forma za dodavanje novih recepata

Posljednja funkcionalnost vezana za manipulaciju parametrima za konfiguraciju rada postrojenja je definiranje intervala (minimalne i maksimalne vrijednosti) za svaki pojedini parametar koji čini recept, kao što je vidljivo na slici 4.6.



Slika 4.6 Sučelje za definiranje intervala pojedinih parametara koji čine recept

Cjelokupan set parametara je podijeljen u nekoliko smislenih cjelina (postavke pokretne trake, postavke motora, postavke ventilatora i sl.) te se na odabir postavki određene grupe mogu definirati konkretni intervali za svaki pojedini parametar koji pripada toj grupi. Kao što je već prethodno objašnjeno, da bi se recept uspješno dodao svi parametri moraju proći verifikaciju. No, postoji scenarij u kojem se nakon naknadne izmjene intervala (minimalne i maksimalne vrijednosti) određenog parametra neki od recepata moraju „izdvojiti“ budući da se „stara vrijednost“ unutar recepta mora prilagoditi novom intervalu. To se ostvaruje na način da se unutar tabličnog prikaza popisa recepta (slika 4.4) sa lijeve strane pojave oznake **[!!!]** pored recepata u kojima je potrebno prilagoditi određene parametre. Prelaskom miša preko oznake prikazuje nam se lista parametara koji su izvan intervala, kao što je prikazano na slici 4.7.



Slika 4.7 Prikaz parametara koji su izvan pre-definiranog intervala

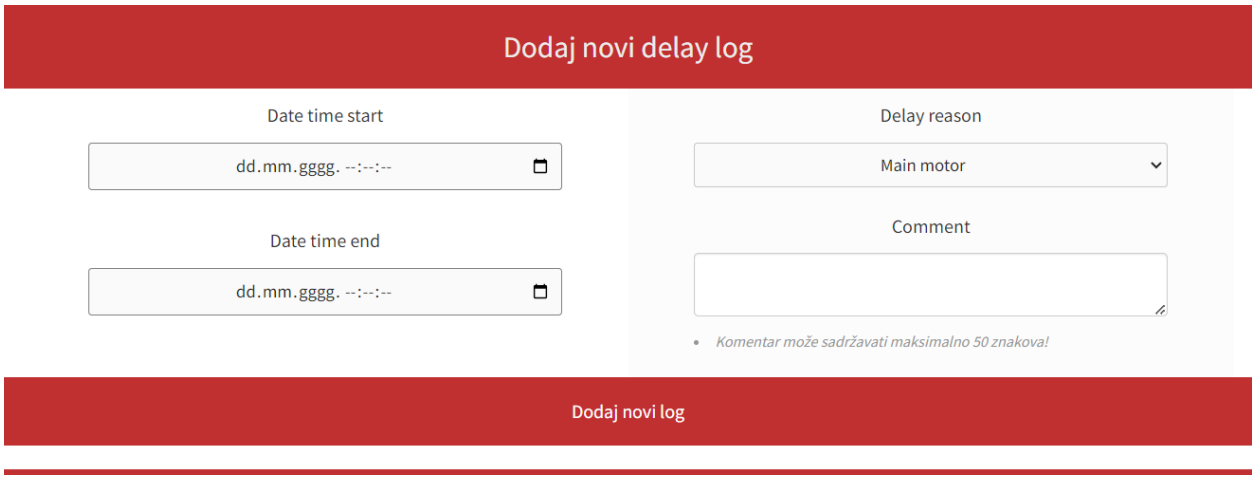
4.2.3. Testiranje funkcionalnosti prikaza *Delay* i *Production* zapisa

Unutar izbornika „Logovi“ dostupni su *Delay* te *Production* zapisi. Na slici 4.8 prikazani su *delay* zapisi (tablično) zajedno sa pripadajućim filterima. Operaterima se omogućava filtriranje podataka po vremenu početka/kraja zastoja, jedinstvenoj identifikacijskoj oznaci zapisa te razlogu zastoja.

Slika 4.8 *Delay* zapisi

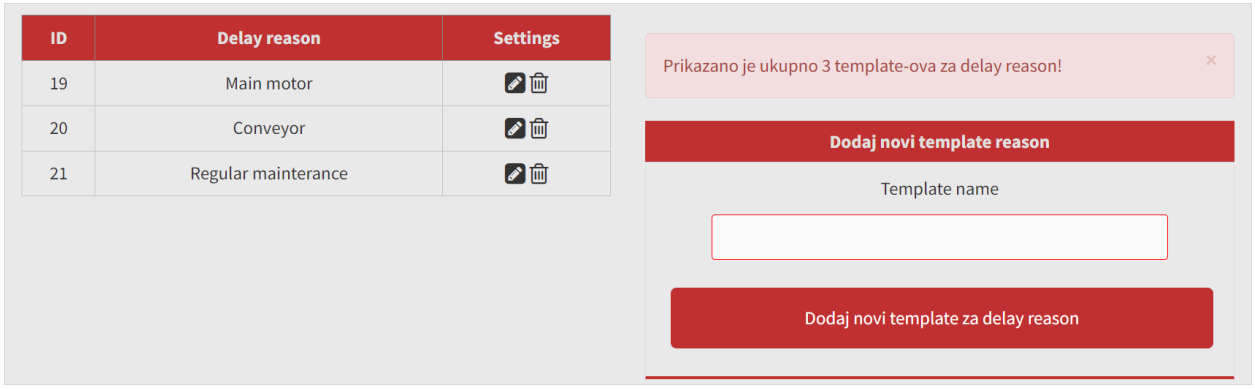
Navedene podatke o intervalima zastoja rada prikuplja servisna aplikacija i upisuje ih u bazu podataka. Istovremeno, Web aplikacija prikazuje podatke te omogućuje operaterima da naknadno dodaju razlog zastoja rada te pripadajući komentar o zastoju. Osim toga, operaterima je omogućen izvoz *delay* zapisa u *Excel* datoteku radi potreba dugotrajnijeg arhiviranja proizvodnje.

Na slici 4.9 prikazana je forma za dodavanje novih *delay* zapisa.



Slika 4.9 Ručno dodavanje *delay* zapisa

Predloške za dodavanje razloga zastoja (npr. *main motor*, *regular mainterance*) operateri mogu dodati ili obrisati unutar izbornika „Admin“ na način kako je prikazano slikom 4.10. Predlošci mogu značajno pomoći za bolje strukturiranje *delay* i *production* zapisa. Bez pravilnog strukturiranja zapisa, podaci biti će nepregledni i neupotrebljivi.



Slika 4.10 Dodavanje i lista predložaka za razloge zastoja

Analogno *delay* zapisima, *production* zapisi izvedeni su na identičan način – jedina razlika je u parametrima koji se prikazuju.

4.2.4. Testiranje funkcionalnosti stvarnovremenskog strujanja komunikacijskog zapisnika

Kao što je prethodno navedeno, zadatak komunikacijskog zapisnika je omogućiti korisniku lakšu dijagnostiku problema i greški. Osim toga, korisniku se mogu isporučiti neke druge bitne informacije. U ovom potpoglavlju prikazano je nekoliko glavnih poruka koje se korisniku prikazuju u komunikacijskom zapisniku.

1. Poruka o uspješnoj konekciji servisne aplikacije i PLC-a (slika 4.11)

```
[06/12/2022 21:42:52]: * * * * * * * * * * * * * * * * * * * * * * * *
[06/12/2022 21:42:52]: New PLC connection detected!
[06/12/2022 21:42:52]: Connection details -> CPU type: S71500, IP Address:
192.168.132.129, Slot: 1
[06/12/2022 21:42:52]: * * * * * * * * * * * * * * * * * * * * * * * *
```

Slika 4.11 Poruka o uspješnoj konekciji servisne aplikacije i PLC-a

2. Poruka o identifikaciji zahtjeva za receptom sa L1 strane (slika 4.12)

```
[06/12/2022 21:48:32]: New Setup request from Level 1 detected!
[06/12/2022 21:48:32]: Details -> Recipe Identifier: 17
```

Slika 4.12 Poruka o identifikaciji zahtjeva za receptom sa L1 strane

3. Poruka o grešci spajanja sa bazom podataka (slika 4.13)

```
[06/12/2022 21:47:24]: Error with database communication. Error
description: Unable to connect to any of the specified MySQL hosts.
```

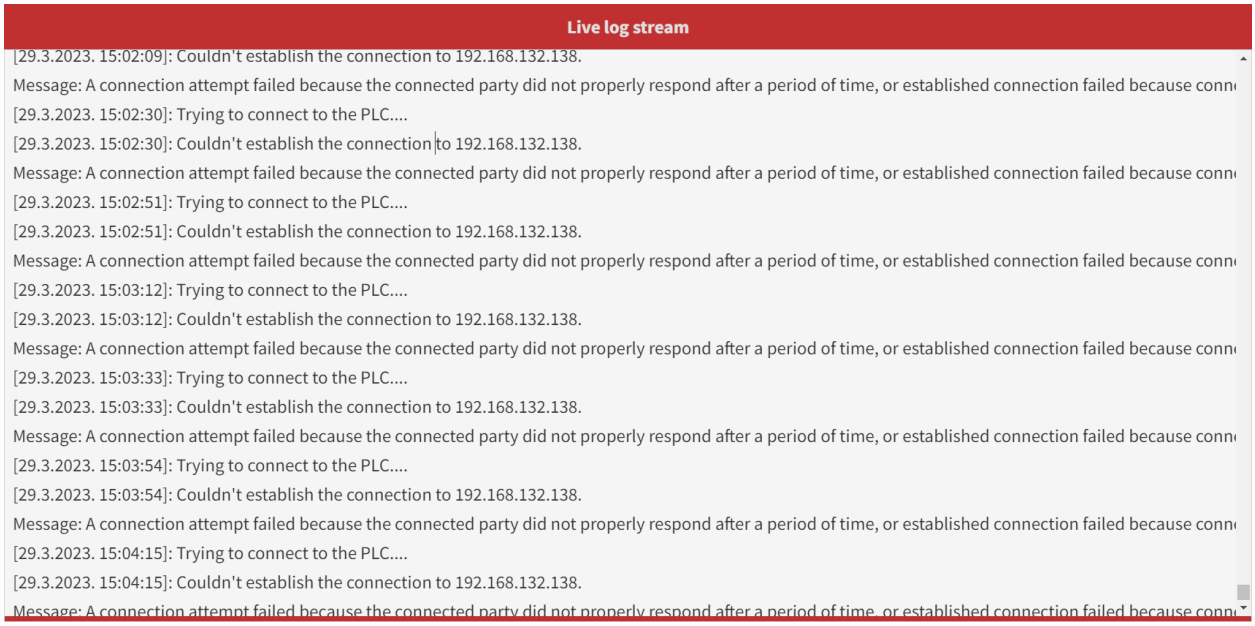
Slika 4.13 Poruka o grešci spajanja sa bazom podataka

4. Poruka o nedostupnosti PLC-a (slika 4.14)

```
[02/12/2022 15:22:09]: Error during executing ReadMultiVars (PLC IP:
192.168.132.129). Error description: Plc is not connected
[02/12/2022 15:22:09]: Trying to connect to the PLC again....
```

Slika 4.14 Poruka o nedostupnosti PLC-a

Komunikacijski zapisnik se zapisuje unutar *.txt* datoteke uz mogućnost stvarnovremenskog strujanja na Web aplikaciji kako je prikazano slikom 4.15.



Slika 4.15 Stvarnovremensko strujanje komunikacijskog zapisnika

5. ZAKLJUČAK

Razvojem radnih stanica za operatere (engl. *Operater workstation*) omogućava se upravljanje, nadzor i kontrola različitih industrijskih procesa i sustava. Razni analitički alati omogućavaju operaterima praćenje i analiziranje performansi cjelokupnog sustava. Zadatak diplomskog rada bio je proširiti postojeće funkcionalnosti *HMI/SCADA* sustava na način da se uz pomoć različitih svojstava i mogućnosti viših programskih jezika cjelokupan sustav dovede na jednu novu razinu. Mogućnosti dugotrajnog arhiviranja proizvodnje, fleksibilnost, prilagodljivost, integracija s drugim sustavima te responzivnost samo su neke od prednosti koje donosi izrada radnih stanica za operatere u višim programskim jezicima.

Jedna od mogućnosti unaprjeđenja diplomskog rada je razvoj dodatne funkcionalnosti automatskog prilagođavanja servisne i Web aplikacije promjenama u strukturi *LanDef-a*. Navedena funkcionalnost zahtijeva dodatni razvoj na svim razinama entiteta (baza podataka, servisna i web aplikacija) kako bi se omogućila lakša i brža migracija projekta. Definiranje struktura podataka moguće je razviti unutar postojeće web aplikacije ili neke druge (pomoćne) aplikacije. Nakon definiranja struktura poruka, set konfiguracijskih parametara potrebno je pohraniti unutar baze podataka te se pomoću navedenog seta konfiguracijskih parametara razvija mehanizam automatskog prilagođavanja servisne i Web aplikacije strukturama poruka unutar *Landef-a*. Osim lakše i brže migracije projekta, navedena funkcionalnost omogućava da ukoliko dođe do promjene strukture podataka unutar *LanDef-a* intervenciju prilagođavanja servisne i Web aplikacije može odraditi osoba bez programerskih vještina. Na način kako je trenutno razvijen cjelokupni sustav, potrebno je ručno modificirati bazu podataka, servisnu i Web aplikaciju ukoliko dođe do promjena unutar struktura poruka *LanDef-a*. Osim toga, potrebno je deinstalirati trenutnu verziju servisne aplikacije (prije modifikacija) te je potrebno instalirati novu (ažuriranu) verziju kao *Windows* servis. U slučaju razvijanja navedene funkcionalnosti, neće biti potrebno deinstalirati trenutnu verziju servisne aplikacije.

Diplomskim radom obuhvaćeni su samo osnovni elementi koji čine modernu radnu stanicu za operatere, kao što su na primjer poruke vezane za konfiguracije načina rada postrojenja, *production* i *delay* zapisi itd. No, ovako razvijen sustav podložan je unaprijeđenima u vidu povećanja broja poruka te samim time i funkcionalnosti. Kompleksne radne stranice za operatere su uglavnom vezane uz kompleksnije procesne linije. Za razvoj takvih radnih stanica neophodno je imati veći broj servisa koji međusobno komuniciraju pomoću softvera posrednika za razmjenu informacija (npr. *RabbitMQ*).

LITERATURA

[1] J. Norman, »Ctesibius Builds the First Truly Automatic Self-Regulatory Device,« History Of Information, [online]. Available: https://www.historyofinformation.com/detail.php?id=1946. [27.6.2023].

[2] »PLC Programming then & now: the history of PLC's,« c3Controls, [online]. Available: https://www.c3controls.com/white-paper/history-of-programmable-logic-controllers/. [Pokušaj pristupa 27 6 2023].

[3] »Općenito o SCADA sustavu,« Elektrokem, [Mrežno]. Available: https://elektrokem.hr/ek-sustavi/cijena/opcenito-o-scada-sustavu. [Pokušaj pristupa 27.6.2023].

[4] »The Best PHP Examples,« freeCodeCamp, [online]. Available: https://www.freecodecamp.org/news/the-best-php-examples/. [16.8.2023.]

[5] »Totally Integrated Automation Portal,« Siemens, [online]. Available: https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html. [16.8.2023.]

SAŽETAK

Diplomskim radom obuhvaćena je izrada radne stanice za operatere (*engl. Operater Workstation*) za sustav gospodarenja željeznim otpadom. Za potpunu funkcionalnosti radne stanice za operatere potrebno je izraditi detaljnu tehničku specifikaciju, definirati podatke za razmjenu, razviti komunikacijski softver (pozadinsku aplikaciju) za manipulaciju nad tim podacima te izraditi Web aplikaciju za vizualni prikaz i kontrolu podataka. Za komunikaciju aplikacije sa PLC-om koristi se Siemens S7.NET biblioteka koja podržava rad sa PLC-ovima Siemens serije S7. Komunikacijski softver (pozadinska aplikacija) razvijena je u C# programskom jeziku. S druge strane, Web aplikacija razvijena je u PHP programskom jeziku. Glavne karakteristike Web aplikacije su: autentifikacija korisnika, sustav naprednog upravljanja parametrima za konfiguraciju rada postrojenja (receptima), prikaz *production* i *delay* zapisa, stvarnovremensko strujanje komunikacijskog zapisnika te reponzivnost. Cjelokupan sustav je testiran pomoću alata *NetToPlcSim* koji omogućava simulaciju stvarne konekcije aplikacije sa PLC-om, dok su rezultati prikazani u dokumentaciji diplomskog rada.

Ključne riječi: automatizacija, PLC, radna stanica za operatere.

ABSTRACT

Development of Operater Workstation for Recycling Systems

The main topic of this thesis is the development of an Operator Workstation for recycling systems. To achieve complete functionality of the operator workstation, it was necessary to create a detailed technical specification, define data for exchange, develop communication software (background application) for handling this data, and design a web application for visual representation and data control. Communication between the application and the PLC (Programmable Logic Controller) was facilitated by the Siemens S7.NET library, which supports working with Siemens S7 series PLCs. The communication software (background application) was developed using the C# programming language. On the other hand, the Web application was developed using the PHP programming language. The main features of the web application include user authentication, an advanced parameter management system for configuring plant operations (recipes), display of production and delay records, real-time streaming of communication logs, and responsiveness. The entire system was tested using the NetToPlcSim tool, which allows simulating a real connection between the application and the PLC, with the results presented in the master's thesis documentation.

Keywords: automation, operator workstation, PLC

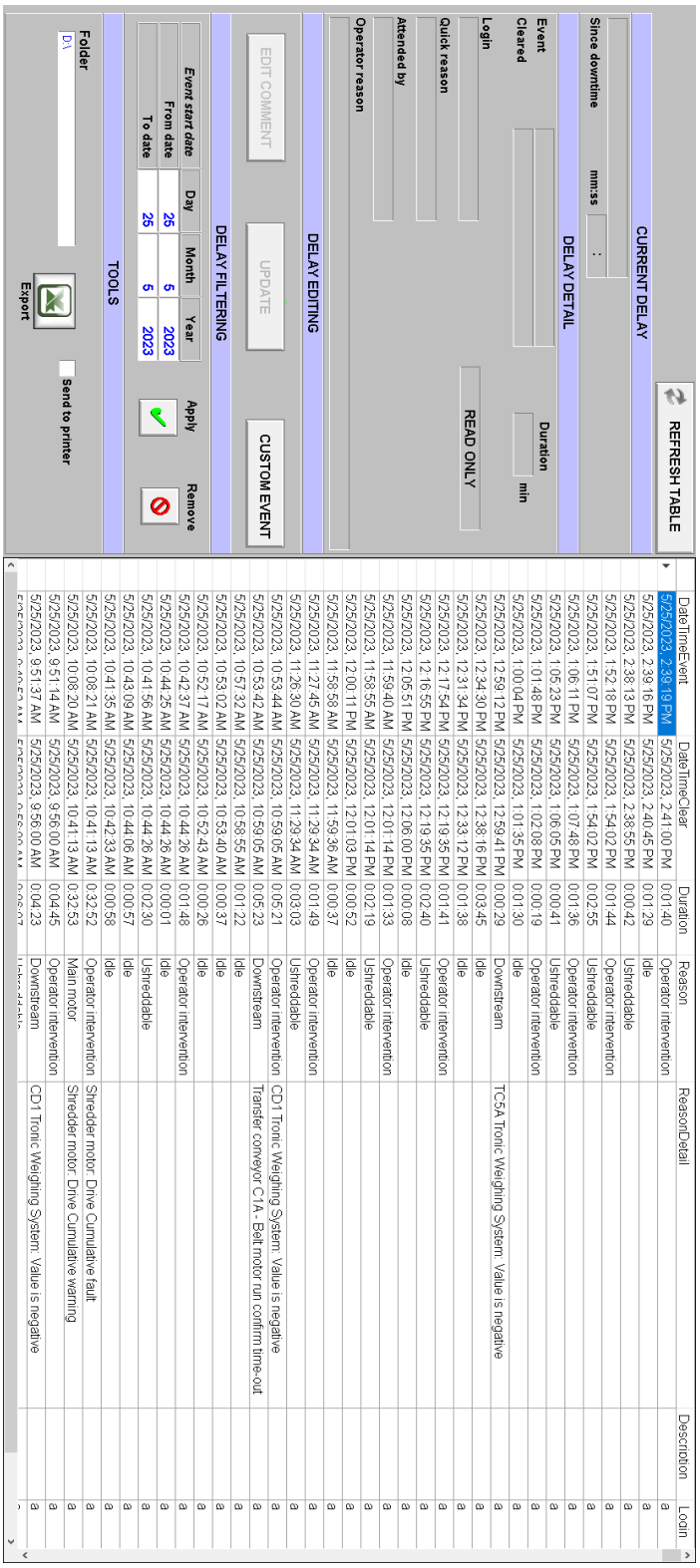
ŽIVOTOPIS

Damir Bašić rođen je 17. ožujka 2000. godine u Slavonskom Brodu. Pohađao je osnovnu školu „Blaž Tadijanović“ u Podvinju. Završio je Tehničku školu u Slavonskom Brodu, smjer elektrotehničar. Nakon završetka srednjoškolskog obrazovanja, pristupa ispitima državne mature te upisuje prvi odabir studija na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer elektrotehnika: komunikacije i informatika. Uspješno položenim završnim ispitom u rujnu 2021. godine stječe stručni naziv sveučilišnog prvostupnika inženjera elektrotehnike i informacijske tehnologije, zatim upisuje diplomski sveučilišni studij elektrotehnike, smjer komunikacije i informatika – izborni blok mrežne tehnologije.

PRILOZI

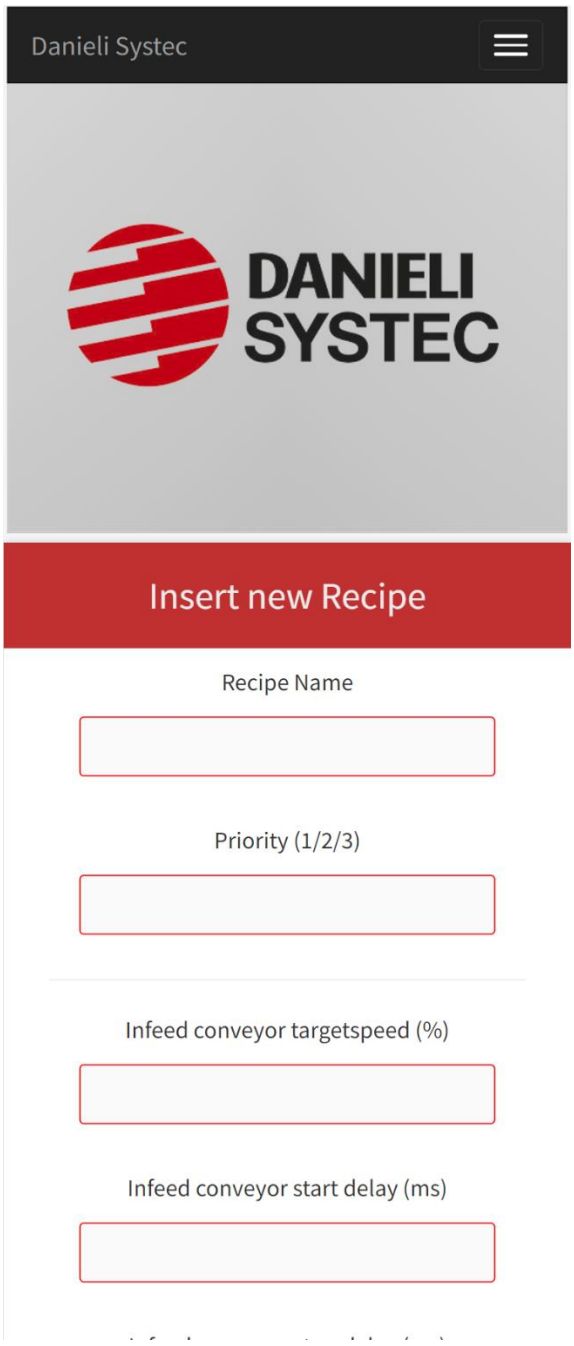
Cjelokupan izvorni kod servisne i web aplikacije, baza podataka te *LanDef* se nalaze na CD-u.

Prilog 1: Stranica koja prikazuje *Delay* zapise

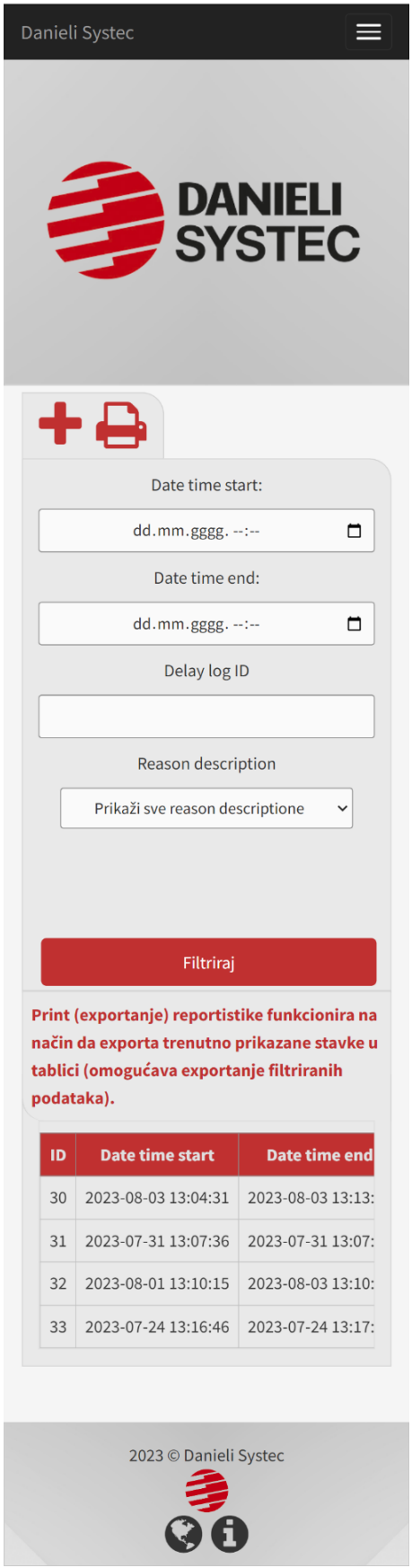


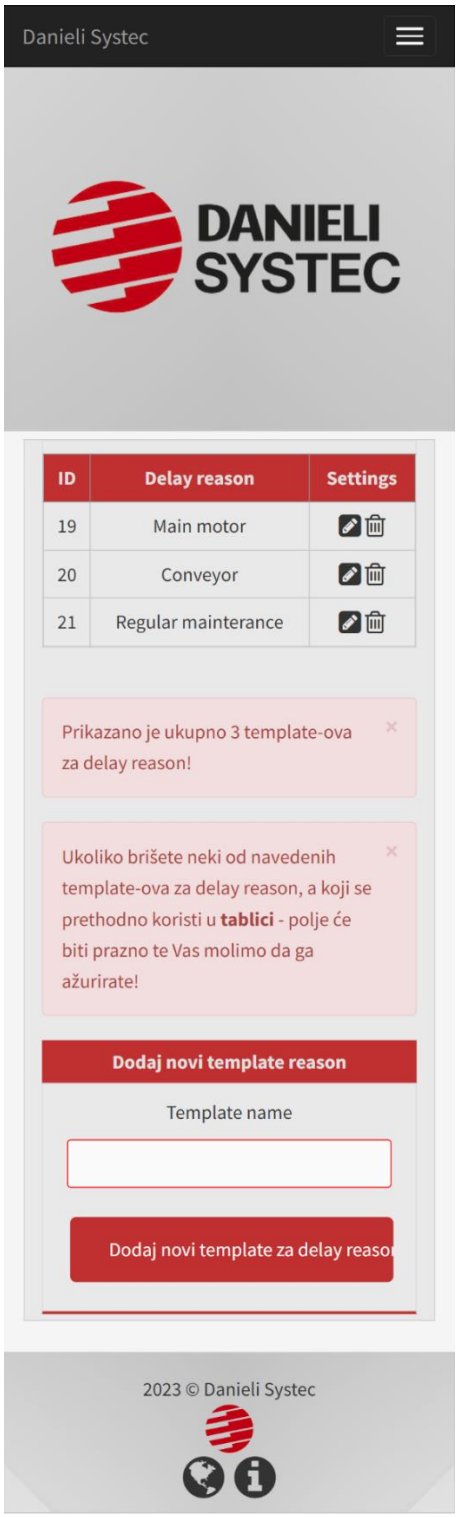
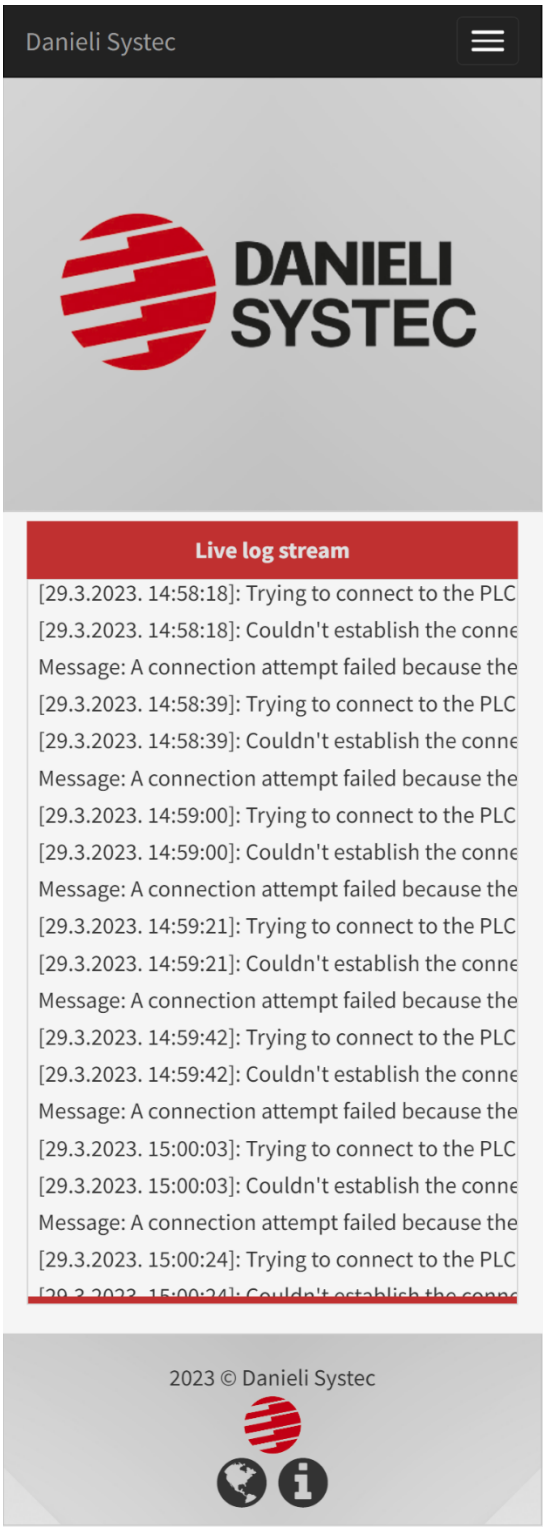
Prilog 2: Stranica koja prikazuje *Production* zapise

Prilog 3: Prikaz recepata i dodavanje novog recepta (responzivno)



Prilog 4: *Delay* zapisi i dodavanje novih predložaka za razloge zastoja (responzivno)





Prilog 5: Stvarnovremensko strujanje komunikacijskog zapisnika (responzivno)