Univerza na Primorskem
Fakulteta za matematiko, naravoslovje in informacijske tehnologije

Tehnična dokumentacija in izvedba projekta sledljivosti proizvodnje

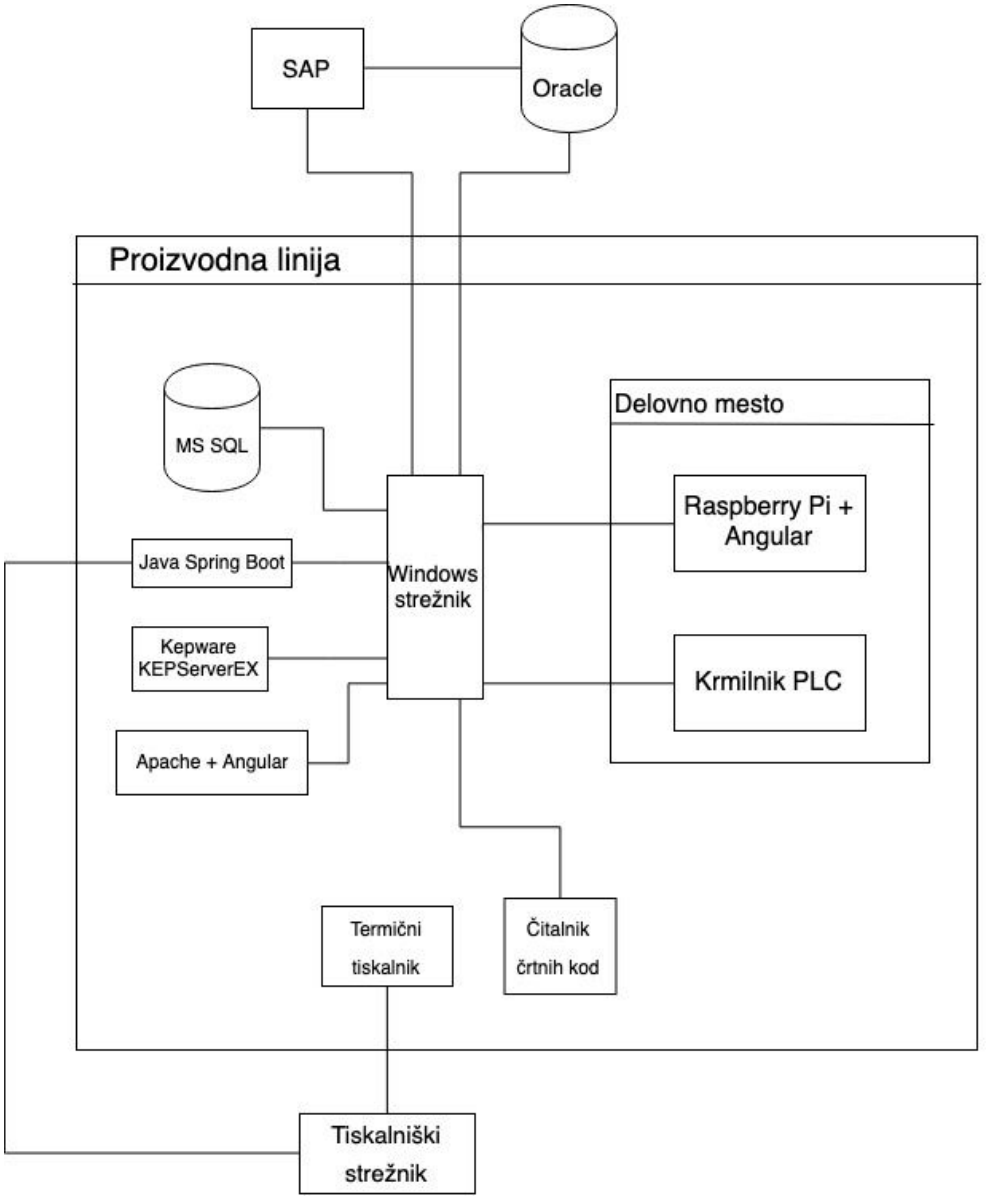
Avtor: Jan Bratina
Mentor: dr. Peter Rogelj

4.4.2019

Kazalo vsebine

1 UVOD

Namen tehnične dokumentacije je predstaviti uporabljeno strojno ter programsko opremo sistema. Strojna oprema sistema je izbrana tako, da predstavlja čim nižji strošek za vpeljavo sistema, sočasno pa mora biti dovolj zmogljiva, da poganja celoten sistem. Sistem je programsko modularno zasnovan, torej lahko dele sistema uporabimo tudi pri drugih projektih. Drugi prednost tega sistema je, da ga lahko poganjamo na raznoraznih klientih, saj temelji na uporabi spletnih tehnologij. Na sliki 1 vidimo, kako sta strojna in programska oprema sistema umeščena v proizvodno linijo ter ostale poslovne sisteme podjetja.



Slika 1: Topološki pregled umestitve sistema na proizvodno linijo

2 UPORABLJENA STROJNA OPREMA

Za lokalni strežnik na proizvodni liniji potrebujemo navaden osebni računalnik z operacijskim sistemom Windows z vsaj 8GB pomnilnika ter dva trda diska, ki jih postavimo v polje RAID 1. Na ta način smo nekoliko zaščiteni pred nenadno odpovedjo enega od trdih diskov. Za večjo varnost se sliko trdega diska redno varnostno kopira na oddaljen strežnik. Lokalni strežnik mora imeti tudi tri fizično ločene mrežne kartice. Prva se uporablja za dostop do poslovnega omrežja, druga za dostop do vseh klientov ter optičnih bralnikov kod ter tretja za dostop do krmilnikov PLC. Priporočljivo je tudi, da imamo na lokalni strežnik priključen na UPS v primeru nenadnega izpada električne energije. Na vsakem delovnem mestu se za klienta uporablja Raspberry Pi 3, na njega pa je priklopljen zaslon občutljiv na dotik. Raspberry Pi je uporabljen predvsem zaradi lažjega upravičevanja stroškov strojne opreme, zmogljivostno pa zadošča trenutnim potrebam. Dobra lastnost uporabe Raspberry Pi-ja je v tem, da ga lahko zelo hitro zamenjamo v primeru okvare. Tudi če pride do okvare pomnilnika, v tem primeru SD kartice, jo lahko enostavno zamenjamo in proizvodnja lahko nemoteno deluje naprej. Zaslon občutljiv na dotik uporablja optično tehnologijo zaznave dotika. To je pomembno zato, ker zaslon mora zaznati dotik, tudi če uporabnik uporablja rokavice. Tudi uporaba teh zaslonov na dotik je pogojena z lažjim upravičevanjem stroškov strojne opreme, saj so ti zasloni veliko cenejši od klasičnih zaslonov na dotik. Nazadnje so tu še termični tiskalniki za nalepke ter optični bralniki kod. Optične bralnike kod potrebujemo na delovnem mestu pakiranja ter za preverjanje pravilnega materiala na delovnih mestih.

3 UPORABLJENA PROGRAMSKA OPREMA TER TEHNOLOGIJE

V tem poglavju sledi opis uporabljene programske opreme ter tehnologij pri izdelavi sistema za sledljivost proizvodnje.

3.1 Kepware KEPServerEx

KEPServerEX je produkt, ki zagotavlja vmesnik med našim sistemom ter PLC krmilniki. Razvilo ga je podjetje Kepware, ki se ukvarja z razvojem programskih rešitev za povezovanje različnih naprav v svetu industrije. Produkt KEPServerEx, ki je v vlogi OPC strežnika, preko svojega uporabniškega vmesnika omogoča povezovanje, spremljanje in nadzorovanje povezanih naprav, oziroma v našem primeru PLC krmilnikov podjetja Siemens [1]. Povezava med našim sistemom, ki je v vlogi OPC klienta ter PLC krmilniki je omogočena preko Siemens TCP/IP Ethernet gonilnika. Gonilnike za povezovanje na naprave različnih proizvajalcev podjetje

Kepware prodaja posebej, torej moramo za vsak tip narave dokupiti gonilnik. Izmenjevanje podatkov med našim sistemom ter PLC krmilniki torej poteka preko komunikacijskega protokola OPC, ki je podrobneje opisal v podpoglavju 6.1.2. Podatki iz PLC krmilnika so predstavljeni s tako imenovanimi značkami, pri čemer ima vsaka značka določen podatkovni tip ter naslov v katerem podatkovnem bloku PLC krmilnika se podatek nahaja. Strežnik KEPServerEx nam omogoča, da značke konstantno beremo z nekim določenim časovnim zamikom lahko pa jih preberemo na zahtevo.

3.2 OPC

OPC je standard za varno in zanesljivo izmenjavo podatkov med industrijskimi napravami ter ostalimi aplikacijami. Namen OPC protokola je ustvariti skupen vmesnik, ki deluje med različnimi ponudniki industrijskih krmilnikov ter ostalo programsko opremo [2]. Za razvoj in vzdrževanje skrbi OPC Foundation, ki je 1996 izdala prvo specifikacijo protokola, ki jo danes poznamo kot OPC DA [2]. Ta specifikacija temelji na Microsoftovi tehnologiji DCOM, posledično deluje samo na Microsoftovem operacijskem sistemu Windows [3]. To se je kasneje izkazalo kot omejitev, zato je bila leta 2008 izdana specifikacija OPC UA, ki deluje na vseh operacijskih sistemih [2]. Naš sistem omogoča uporabo tako OPC DA, kot tudi OPC UA komunikacijskega protokola, konkretno na opisani liniji pa je uporabljen OPC DA protokol, saj so izbrani PLC krmilniki podpirajo ta protokol brez dodatne konfiguracije. Prav tako nismo omejeni z operacijskim sistemom, saj je na strežniku na proizvodnji liniji nameščen operacijski sistem Windows.

3.3 SPRING BOOT

Ogrodje Spring Boot nudi podporo pri izgradnji spletnih javanskih aplikacij [4]. Je del širšega aplikacijskega ogrodja Spring. Namen ogrodja Spring Boot pa je, da čim hitreje ter z malo konfiguracije vzpostavimo Spring aplikacijo oziroma spletni servis [4]. To je bil tudi eden glavnih razlogov za uporabo tega ogrodja. Vsebuje že konfigurirane, najpogosteje uporabljene knjižnice, ki pa jih lahko po želji tudi spremenimo oziroma zamenjamo [4]. Za upravljanje z odvisnostmi, knjižnicami ter za prevajanje kode v našem projektu skrbi orodje Maven. Za zagon spletne aplikacije ter spletnih servisov skrbi vgrajen strežnik Tomcat. Za potrebe našega projekta smo morali v projekt vključiti tako knjižnice, ki nam jih ponuja ogrodje Spring, kot tudi nekaj knjižnic izven tega ogrodja. Nekaj knjižnic, ki smo jih uporabili v projektu:

- *spring-boot-starter-web* – knjižnica je uporabljena za izdelavo spletnih aplikacij ter RESTful aplikacij [5],
- *spring-boot-starter-batch* – knjižnica nam pomaga pri implementaciji funkcij, ki se izvajajo v nekem zaporedju, naj bo to časovno zaporedje ali pa vezano na nek prožilec,

- *spring-boot-starter-test* – knjižnica, ki je uporabljena za testiranje Spring aplikacij. Med drugim vsebuje tudi ogrodje za testiranje JUnit [5],
- *spring-boot-starter-jdbc* – knjižnica, ki nam omogoča pisanje in klicanje SQL poizvedb znotraj javanske kode,
- *spring-boot-starter-integration* – ta knjižnica nam omogoča integracijo zunanjih sistemov v našo aplikacijo. V našem primeru to pomeni konfiguracijo OPC strežnika, povezavo na optične bralnike kod itd. [6],
- *spring-boot-starter-websocket* – ta knjižnica nam omogoča vzpostavitev WebSocket protokola med našo aplikacijo ter ciljno napravo [5],
- *jtds* – je odprtokodna knjižnica, ki prinaša v našo aplikacijo JDBC gonilnik za Microsoft SQL Server, ki je lokalna baza v tem sistemu [7],
- *ojdbc* – ta knjižnica prinaša v našo aplikacijo JDBC gonilnik za Oracle podatkovno bazo, ki je poslovna podatkovna baza,
- *openscada* – knjižnica, ki nam omogoča povezavo in pisanje podatkov na OPC strežnik ter branje podatkov iz OPC strežnika [8],
- *jackson* – knjižnica, ki prinaša podporo razčlenjevanju JSON datotek v javanski kodi. V našem projektu je uporabljena pri upravljanju z recepti, saj so recepti za delovna mesta napisani v JSON formatu,
- *jpedal* – ta knjižnica nam omogoča pretvarjanje PDF dokumentov v slikovne dokumente, ki jih nato lažje prikažemo na delovnem mestu,
- nazadnje je tu še lastna, v podjetju razvita, javanska knjižnica, ki se jo uporablja tudi v drugih projektih znotraj podjetja. Ta knjižnica vsebuje vse potrebne objekte, servise ter objekte za dostop do podatkov (angl. Data Access Object - DAO), ki so vmesnik med aplikacijo ter podatkovno bazo.

Spring Boot aplikacija ima v našem sistemu vlogo zalednega (angl. backend) sistema. V glavnem nam omogoča povezavo na OPC strežnik, na deljene mrežne diske in obdelavo datotek, na optične bralnike kod, vzpostavitev WebSocket povezave itd. Ena najpomembnejših lastnosti te aplikacije je omogočanje dostopa do funkcij preko unikatnega URL naslova. V spletni aplikaciji se nato uporabljajo klici standardnih HTTP metod, in sicer GET, POST, PUT in DELETE. To je omogočeno z uporabo arhitekture za izmenjavo podatkov med spletnimi storitvami (angl. Representational state transfer – REST).

3.4 Microsoft SQL Server Express Edition

Za relacijsko podatkovno bazo smo izbrali Microsoft SQL Server Express Edition. To relacijsko podatkovno bazo se je v podjetju uporabljalo že pri prejšnjih projektih, kar je bil tudi eden od razlogov za izbiro tega sistema. Ena od omejitev te zastonjske verzije relacijske podatkovne baze je, da lahko shrani do 10GB podatkov. Ker se varnostno kopiranje izvaja v realnem času na poslovno podatkovno bazo Oracle nam omejitev Microsoft SQL Express Server-ja ne predstavlja velikega problema, saj lahko starejše podatke brišemo iz lokalne podatkovne baze. Za uporabniški vmesnik

baze je uporabljen Microsoft SQL Server Management Studio. Preko uporabniškega vmesnika je enostavno kreirati nove tabele. To orodje pa nudi pa nam še tudi naprednejša orodja, kot so analiza najzahtevnejših poizvedb v realnem času ter analiza porabljenega prostora.

3.5 Angular

Angular je platforma in ogrodje za izdelovanj spletnih aplikacij. Aplikacije se piše v programskem jeziku TypeScript, ki je nadgradnja JavaScript-a [9]. Programski jezik TypeScript nam omogoča uporabo tipov, vmesnikov, razredov, … Te lastnosti nam pomagajo pri hitrejšem ter lažjem programiranju. Napisana TypeScript koda se nato prevede v običajno JavaScript kodo. Ta korak je potreben zato, ker spletni brskalniki ne znajo prikazati aplikacije napisane v TypeScript-u [10]. Angular temelji na modularnem pristopu, katerega se poskušamo držati tudi pri razvoju lastne aplikacije. Za nadzor ter dodajanje novih modulov je v našem projektu uporabljen npm (angl. node package manager). Angular aplikacije so razdeljene na komponente (angl. components), katere so sestavljene iz aplikacijske kode v TypeScriptu, ter kode za prikaz, ki je sestavljena iz HTML-ja in CSS-ja. Angular nam ponuja kar nekaj značk znotraj HTML kode, ki pospešijo in olajšajo razvoj. Težimo k temu, da komponente pišemo čim bolj splošno, saj lahko na ta način komponento večkrat uporabimo in tako ne ponavljamo programske kode po nepotrebnem. Komponente, do katerih želimo dostopati preko URL naslova moramo dodati v poseben modul za preusmerjanje (angl. routing module), kjer jim tudi enolično določimo URL naslov. Pomemben del Angular aplikacije so tudi tako imenovane storitve (angl. services), ki so specializirani razredi namenjeni opravljanju točno določene funkcije [11]. Komponente in storitve so v Angular-ju ločene z namenom doseganja modularnosti ter ponovne uporabnosti kode [9]. Komponente navadno uporabljajo storitve za klic funkcij strežniške aplikacije. V našem primeru tako preko storitev in z uporabo standardnih HTTP metod kličemo pripravljene funkcije iz naše Spring Boot aplikacije. Na ta način pridobimo vse potrebne podatke iz strežniške aplikacije.
Pri razvoju Angular aplikacije nam zelo pomaga orodje Angular CLI, s pomočjo katerega lahko generiramo nove komponente ter storitve, končno prevedemo ter zgradimo projekt, itd. [12] To orodje nam omogoča tudi hitro gostovanje Angular aplikacij v fazi programiranja in testiranja. Pohitri nam predvsem fazo testiranja, saj ni potrebno zgraditi aplikacije za vsako spremembo, temveč se ob vsaki spremembi aplikacija avtomatsko osveži. V fazi produkcije pa Angular aplikacijo prevedemo in zgradimo z orodjem Angular CLI ter jo nato gostujemo na strežniku Apache. S tem ko Angular aplikacijo zgradimo za produkcijski način pridobimo na hitrosti nalaganja spletne aplikacije ter tudi na njeni odzivnosti. Tako gostovanje v fazi testiranja aplikacije ni praktično, saj je potrebno vedno znova zgraditi aplikacijo, kar je časovno potratno.

VIRI

[1] Kepware KEPServerEX, https://www.kepware.com/en-us/products/kepserverex/
[2] OPC Foundation, https://opcfoundation.org/about/what-is-opc/
[3] X. Hong, W. Jianhua: Using standard components in automation industry: A study on OPC specification
[4] Spring, https://spring.io/projects/spring-boot
[5] Spring Boot Starters, https://www.javatpoint.com/spring-boot-starters
[6] Spring Integration, https://spring.io/projects/spring-integration
[7] JTDS, http://jtds.sourceforge.net/
[8] Open Scada, http://oscada.org/wiki/About
[9] Angular, https://angular.io/guide/architecture
[10] Difference between TypeScript and JavaScript, https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/
[11] Algular, https://angular.io/guide/architecture-services
[12] Algular CLI, https://angular.io/cli