Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žan Zadnikar

Informacijski sistem za avtomatizacijo kontrole vstopa in nadzora prometa

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Informacijski sistem za avtomatizacijo kontrole vstopa in nadzora prometa.

Tematika naloge:

Avtomatizacija sistemov delovanja z informacijsko-komunikacijsko tehnologijo je ključnega pomena za izboljšanje njihove varnosti, učinkovitosti in analize. Hitri razvoj računalniških tehnologij omogoča, da podjetja in ustanove avtomatizirajo procese, ki so bili prej stroškovno neopravičljivi. V okviru diplomske naloge predstavljam informacijsko rešitev za optimizacijo logističnih procesov s pomočjo računalniškega vida.

Zahvaljujem se družini in prijateljem.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
OCR	Optical Character Recognition	Optično prepoznavanje znakov
IT	Information Technology	Informacijska tehnologija
PHP	Hypertext Preprocessor	Predprocesor za hiperbesedilo
PDO	PHP Data Objects	Podatkovni objekti PHP
JS	JavaScript	JavaScript
JSON	JavaScript Object Notation	JavaScript objektna notacija
CSS	Cascading style sheets	Kaskadne stilske predloge
API	Application programming interface	Vmesnik za namensko programiranje
OpenCV	Open Source Computer Vision Library	Odprtokodna knjižnica za računalniški vid
OpenALPR	Open Source Automatic License Plate Recognition	Odprtokodna knjižnica za prepoznavo registrskih tablic
DBMS	Database Management System	Sistem za upravljanje s podatkovnimi zbirkami
DNS	Domain Name System	Sistem domenskih imen
NAT	Network Address Translation	Prevod omrežnega naslova

Povzetek

Naslov: Informacijski sistem za avtomatizacijo kontrole vstopa in nadzora prometa

Ključnega pomena za uspešnost organizacije glede na konkurenco je zavedanje o nujnosti nenehnega izboljševanja poslovnih procesov. S to mislijo je ob pomoči informacijsko-komunikacijskih tehnologij v diplomski nalogi predstavljen informacijski sistem za optimizacijo logističnega procesa avtomatizacije kontrole vstopa in nadzora/analize prometa z uporabo različnih tehnologij (PHP, Bootstrap, JS, MariaDB, openALPR ...). Te bodo predstavljene v prvem delu diplomske naloge. V nadaljevanju naloge se bomo posvetili podrobnemu opisu razvoja sistema, prikazan pa bo tudi poslovni primer uporabe.

Ključne besede: informacijski sistem, nadzor kontrole vstopa, php, javascript, mariadb, openalpr, debian.

Abstract

Title: Information System for Entry Control Automation and Traffic Control

A continual business process improvement is one of the most important key elements of successful businesses. This was the focal point during the whole development of this thesis: how to use the power of information and communication technologies (PHP, Bootstrap, JS, MariaDB, openALPR ...) together with good practices to optimize logistic processes of entry control automation, traffic control and traffic analysis to provide optimal efficiency for the company.

Keywords: information system, entry control system, php, javascript, mariadb, openalpr, debian.

Poglavje 1

Uvod

Informacijske tehnologije so v mnogih podjetjih predstavljene kot obvezen strošek, ki bremeni proračun podjetja. Na drugi strani se vodstveni kadri uspešnih podjetij zavedajo moči prave informacije, ki jo s pomočjo informacijskih tehnologij lahko nudijo podjetju za povečanje poslovne vrednosti. Za takšna podjetja je značilno, da razumejo in upravljajo tveganja z informacijsko tehnologijo. Ključno vlogo pri optimiziranem upravljanju ima ustrezna uporaba razpoložljivih virov IT, vključno z informacijami, infrastrukturo, aplikacijami in ljudmi. Pri zasnovi diplomskega dela se je upoštevalo prav to: kako zgraditi stabilen informacijski sistem, ki bo ob možnosti človeške interakcije z aplikacijami ponudil podjetju le najkoristnejše informacije.

Ideja diplomskega dela o optimizaciji logističnega procesa se je razvila v času študentskega dela v enem izmed večjih slovenskih podjetij, kjer za nadzor kontrole vstopa in nadzora prometa ne izkoriščajo vseh tehnologij, ki jih imajo na voljo. Takšen sistem delovanja namreč ne omogoča optimalnosti, saj mora delo, ki bi lahko bilo avtomatizirano, opravljati človek. Prav tako je sistem podvržen visokemu faktorju človeške napake in s tem povezanimi napačnimi vnosi. Cilj končnega proizvoda diplomskega dela je bil tako postavljen, namreč razvoj informacijskega sistema, ki bo ponudil učinkovito upravljanje in avtomatizacijo kontrole vstopa in nadzora prometa.

Pri oblikovanju diplomskega dela so bili upoštevani naslednji informacijski

kriteriji [7]:

- Zahteve po kakovosti: uspešnost sistema se nanaša na informacije, ki so pomembne za poslovni proces, ob optimalni uporabi razpoložljivih virov (učinkovitost).
- Zahteve po varnosti: varovanje občutljivih informacij (preprečitev nepooblaščenega dostopa) s poudarkom na celovitosti podatkov (pravilnost in popolnost). Informacije morajo biti razpoložljive vsakokrat, ko jih potrebujemo.
- Zahteve po skladnosti: skladnost z zakoni, predpisi in pogodbenimi dogovori, ki veljajo za poslovni proces.

Poglavje 2

Tehnologije

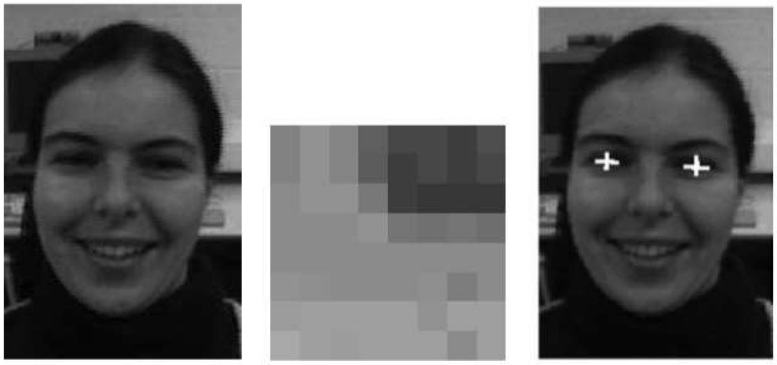
To poglavje je namenjeno predstavitvi in pojasnitvi ključnih tehnologij, uporabljenih v procesu izdelave končnega produkta diplomskega dela.

2.1 Računalniški vid

Začetki računalniškega vida segajo v šestdeseta leta 20. stoletja z doktorsko disertacijo Lawrencea Robertsa o pridobivanju 3D geometrijskih informacij iz 2D perspektive. Številni znanstveniki so sledili temu delu in tako omogočili začetek razvoja računalniškega vida, ki je v zadnjih letih zelo intenziven, predvsem zaradi cenovne dosegljivosti velikih procesorskih moči, zmogljivejših in natančnejših kamer ter neprestanih izboljšav algoritmov za potrebe računalniškega vida.

Kaj računalniški vid pravzaprav je? Osnovni cilj računalniškega vida je ustrezno odločanje na podlagi prepoznave oblik/predmetov iz zajema slike [10]. Naš namen je torej pridobiti karseda veliko vsebinskih informacij, ki nam pomagajo pri nadaljnjem odločanju (npr. medicinska diagnostika za odkrivanje rakavih celic, kontrola kakovosti v proizvodnih sistemih, vojaška industrija ...). Ta proces pridobivanja informacij pa je bistveno kompleksnejši, kot si lahko to kot vizualna bitja pogosto zavajajoče predstavljamo. Vse, s čimer računalniški sistem lahko predstavi neko sliko, je tabela vredno-

sti slikovnih točk s podatki o barvi in intenziteti.

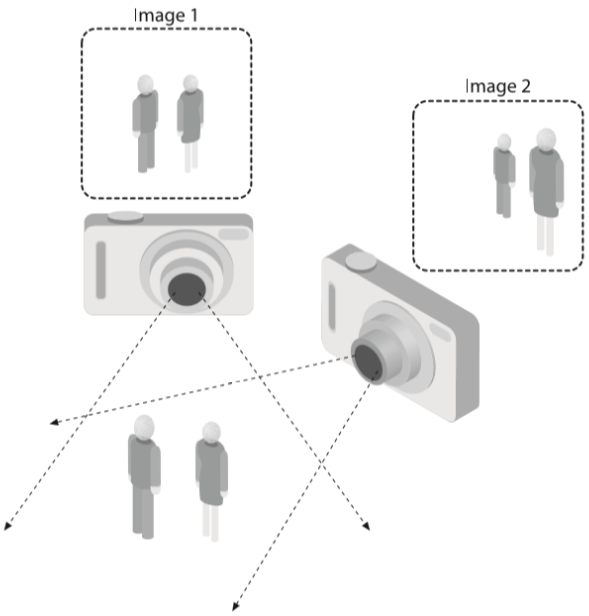


	0	1	2	3	4	5	6	7
0	130	146	133	95	71	71	62	78
1	130	146	133	92	62	71	62	71
2	139	146	146	120	62	55	55	55
3	139	139	139	146	117	112	117	110
4	139	139	139	139	139	139	139	139
5	146	142	139	139	139	143	125	139
6	156	159	159	159	159	146	159	159
7	168	159	156	159	159	159	139	159

Slika 2.1: Prikaz kodiranja dela slike (8x8 slikovnih točk) za predstavitev v računalniškem sistemu.

Iz teh vrednosti slikovnih točk je treba sedaj pridobiti dojetje predmetnega sveta, kar predstavlja nerešljiv problem [3], saj iz 2D pogleda ne moremo konstruirati 3D oblike. Tudi če razpolagamo s popolnimi podatki, je ta proces nemogoč, saj lahko ista 2D slika predstavlja neskončno število kombinacij 3D oblike. Prav tako je potrebno upoštevati motnje pri zajemanju slike (šum, popačenja, nepravilnosti na objektivu/kameri ...) zaradi dejavnikov, ki otežujejo proces prepoznave oblik s fotografije. Da bi razvili neki sistem z uporabo računalniškega vida je potrebno vedeti točen namen in kontekst uporabe, v okviru katerega se bo sistem uporabljal. Tako se (vsaj delno) izognemo procesiranju motenj, ki nastanejo pri zajemu slike, kjer to ni potrebno. Npr., sistem za prepoznavo človeškega obraza lahko pričakuje človeka oziroma obraz samo na točno določenem mestu fotografije. V vseh drugih

conah fotografije se procesiranje ne izvaja. Tako na teh delih ne more priti do napačnih informacij na podlagi motenj, ki bi tam lahko bile.



Slika 2.2: Isti 3D objekt ima neskončno 2D pogledov.

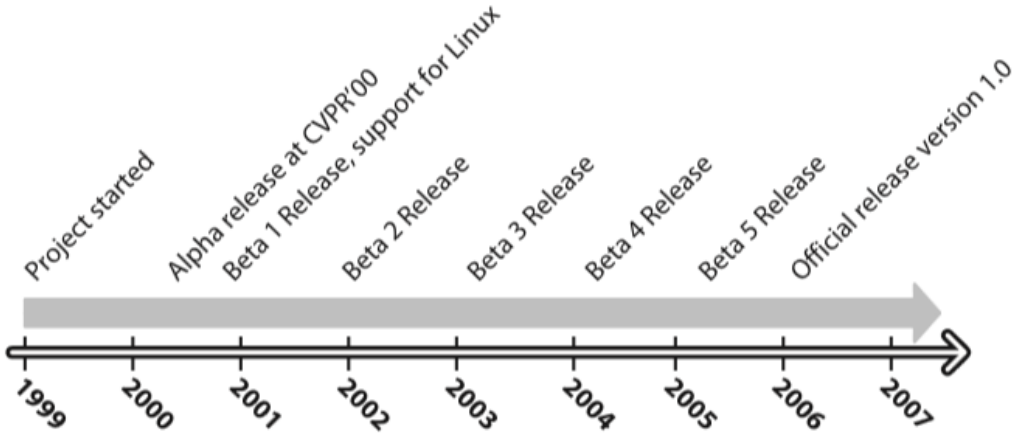
Naslednji velik izziv za računalniški vid predstavlja šum. Največkrat se težave s šumom odpravljajo z uporabo statistike oziroma statističnih metod. Npr., določanje robu na podlagi primerjave dveh sosednjih točk na fotografiji je lahko zaradi motenj izjemno težko oziroma nemogoče. Vendar če upoštevamo statistiko preko določenega območja fotografije, postane takšno zaznavanje robov bistveno enostavnejše in učinkovitejše.

Akcije in odločitve računalniškega vida so tako odvisne od namena uporabe oziroma od strategije, ki jo za nek sistem določimo. Sistem za prepoznavanje obrazov bo imel bistveno drugačno strategijo kot robot za iskanje določenih predmetov. Potrebno se je zavedati, da bolj ko je omejen kontekst računalniškega vida, bolj lahko problem poenostavimo, tako pa zagotovimo višjo zanesljivost delovanja končnega sistema.

2.2 OpenCV

Projekt OpenCV je nastal v okviru Intelovega razvojnega laboratorija Intel Research v letu 1999 z namenom razviti procesorsko intenzivne aplikacije. Predstavljenih je bilo več različnih projektov, med pomembnejšima z vidika računalniškega vida sta bila sledenje žarkov v realnem času in prikazovanje 3D objektov. Glavni doprinos k projektu je omogočilo sodelovanje Intelove razvojne ekipe z ruskimi strokovnjaki, ki so hkrati postavili cilje projekta OpenCV:

- Predstaviti odprtokodno in optimizirano infrastrukturo za napredne raziskave s pomočjo računalniškega vida.
- Razširitev znanja s platformo, ki je berljiva, premestljiva in zmožna nadgradnje s strani razvijalcev.
- Omogočitev razvoja komercialnih aplikacij na podlagi brezplačne optimizirane kode.



Slika 2.3: Časovnica razvoja projekta openCV.

OpenCV je danes objavljen pod BSD licenco in je brezplačen tako za akademsko kot komercialno uporabo. Knjižnica OpenCV je napisana v pro-

gramskem jeziku C/C++ s podporo C++, Python in Java vmesniki, prav tako jo je možno namestiti na vse razširjene operacijske sisteme (Microsoft Windows, Linux distribucije, Mac OS, iOS in Android).

2.3 OpenALPR

OpenALPR je odprtokodna knjižnica, napisana v programskem jeziku C++ z c#, Java, Node.js, Go in Python vmesniki. Specializira se za prepoznavanje avtomobilskih tablic, ki jih kot končni rezultat izpiše v tekstovni obliki skupaj z določeno mejo zaupanja. OpenALPR za svoje delovanje potrebuje knjižnici OpenCV in Tesseract OCR.

2.4 Tesseract OCR

Tesseract je odprtokodni modul, napisan v C/C++ za optično prepoznavanje znakov, razvit s strani podjetja Hewlett-Packard v letih 1984 in 1994. V devetdesetih letih 20. stoletja je bil deležen manjših popravkov, dokler ni konec leta 2005 Hewlett-Packard objavil Tesseract pod licenco Apache License v2.0 (odprtokodna programska oprema). Od leta 2006 naprej za njegov razvoj skrbi Google. Tesseract je možno namestiti na vse razširjene operacijske sisteme (Microsoft Windows, Linux distribucije in Mac OS) [1].

2.5 Bootstrap

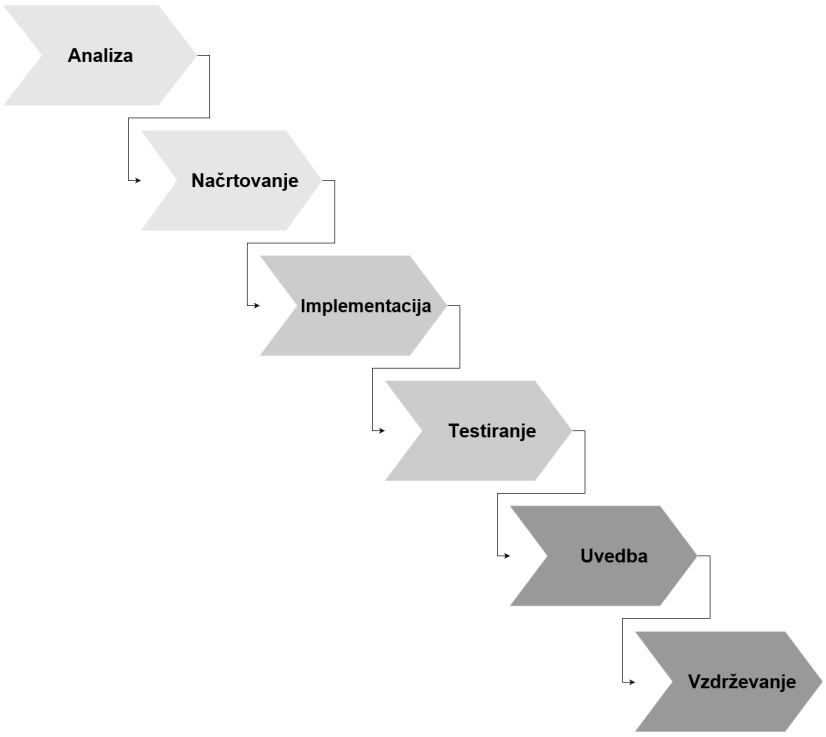
Bootstrap je brezplačno orodje/okvir za hitrejši in lažji razvoj grafične podobe spletnih aplikacij z uporabo HTML-ja, CSS-sa in JavaScript-a. Razvit je bil s strani podjetja Twitter (Mark Otto in Jacob Thornton), leta 2011 pa je bil kot odprtokodni produkt objavljen na spletni platformi GitHub. Sredi leta 2014 je bil Bootstrap na GitHab-u predstavljen kot najobetavnejši projekt. Med uporabniki se je uveljavil predvsem zaradi preproste uporabe, odzivne spletne strani na vseh napravah in zaradi kompatibilnosti z vsemi

modernimi brskalniki (Chrome, Firefox, Internet Explorer, Safari in Opera) [2].

Poglavje 3

Razvoj informacijskega sistema

Potek razvoja informacijskega sistema se je odvijal v skladu s tradicionalnim življenjskim ciklom razvoja sistema (angleško "waterfall model").



Slika 3.1: Tradicionalni življenjski cikel razvoja sistema.

3.1 Tradicionalni model

Metodologija je bila prvič uradno dokumentirana leta 1970 s strani Winstona W. Roycea in je osnova številnim metodologijam, ki so se razvile kasneje. Njena glavna prednost je enostavna uporabnost in razumljivost, saj gre za logično zaporednje ciklov, ki so povezani v celoto. V fazi prehoda na novi cikel mora biti predhodni cikel končan, saj metodologija ne dopušča prekrivanj med posameznimi cikli. Takšen način razvoja rešitve je primeren predvsem za projekte, ki imajo točne in dobro definirane končne cilje [9].

Glavne prednosti:

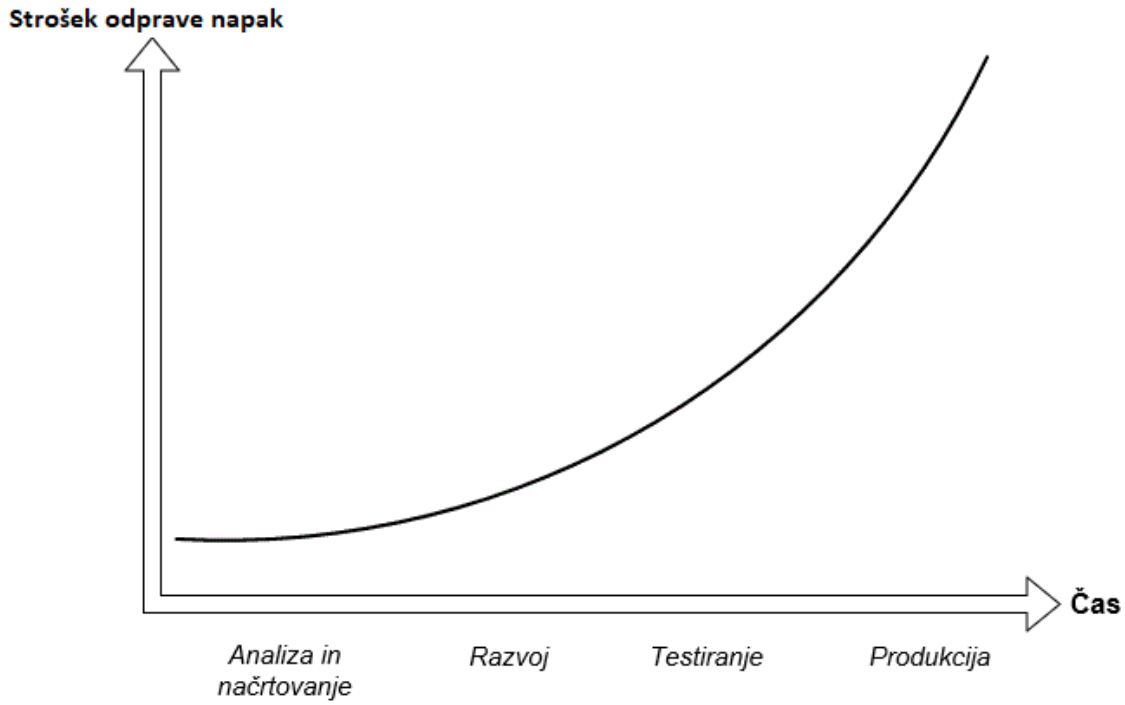
- Lahek za razumevanje in implementacijo.
- Splošno sprejet in poznan (vsaj v teoriji).
- Učinkovito upravljanje modela zaradi točno določenih ciljev znotraj posameznega cikla.
- Krepi dobre prakse: "**define before design**"in "**design before code**".

Glavne pomanjkljivosti:

- Ko je razvoj aplikacije že dovolj daleč, se je izredno težko vrniti nazaj in popraviti napake, ki so bile posledica slabo premišljenega načrta (učinek slapu).
- Delujoča rešitev je v življenjskem ciklu razvoja dostavljena pozno.
- Neprimeren model za dolgotrajne in hitro razvijajoče se projekte.
- Neprimeren model za projekte z visoko možnostjo vmesnih sprememb.
- Veliko administracije, kar lahko predstavlja časovno in stroškovno potratnost za manjše ekipe in projekte.

Kdaj torej uporabiti tradicionalni model? V našem primeru gre za končni produkt z jasnimi in dobro definiranimi zahtevami. V razvojnem postopku ne pričakujemo sprememb, ki bi bistveno spremenile potek razvoja jedra informacijskega sistema. Uporabljene tehnologije v procesu razvoja so razumljive in dobro poznane. To je nekaj od razlogov, na podlagi katerih smo se odločili za razvoj z uporabo tradicionalnega modela življenjskega cikla.

Ker je pri tradicionalnem modelu ena izmed ključnih faz za končno uspešnost produkta v veliki meri odvisna od dobro premišljene analize in ustreznega načrta, bo tej fazi namenjena še posebna pozornost. Želeli bomo preprečiti vsa morebitna presenečenja, ki bi vodila k spremembam ključnih elementov sistema. Potrebno se je zavedati, da predstavlja reševanje napak pozno v razvoju sistema veliko časovno in stroškovno obremenjenost.



Slika 3.2: Tradicionalna stroškovna krivulja sprememb v povezavi s fazo razvoja [5].

3.2 Analiza

3.2.1 Opis delovanja trenutnega sistema

Domena: Postopek prehodov motornega prometa preko vstopne točke podjetja.

Opis trenutnega stanja: Podjetje, katerega glavni vir prihodkov je prodaja lastnih produktov avtomobilski industriji v tujino, se dnevno srečuje z veliko prehodi preko njihove edine vstopne točke v podjetje. Preko te točke vstopajo in izstopajo tako zunanja podjetja (transport produktov, zunanji partnerji, gostje ...) kot določeni zaposleni v podjetju, ki imajo ekskluzivne pravice do vstopa (vodstveni kadri). Za varnost na vstopni točki skrbi zunanji izvajalec, ki vsak prehod zabeleži v Excel tabelo. Za proces identifikacije vozila sta potrebna vsaj dva uslužbenca, saj tisti za sprejemnim oknom neposredno ne vidi registrske tablice vozila. Varnostna služba prav tako zabeleži podatke o vozniku, zaradi česar mora voznik zapustiti vozilo in priti do sprejemnega okna. V tem času se za stoječim vozilom postavi več vozil, ki pogosto blokirajo normalno pretočnost prometa (še posebej ob prometnih konicah). Varnostna služba ima poleg identifikacije vozil in voznikov tudi druga opravila, kot so identifikacija zaposlenih, ki so pozabili službeno kartico, vnos zunanjih partnerjev v sistem podjetja, izvajanje video nadzora ipd.

3.2.2 Identifikacija pomanjkljivosti

Podjetja, ki dnevno prepuščajo veliko prometa preko kontrolnih točk v svoja območja, bi morala zagotoviti visoke varnostne standarde in mehanizme, ki bi jih v primeru morebitnih zlorab znala ustrezno obravnavati (npr., koliko časa se je tovorno vozilo zadrževalo v določenih conah podjetja, ali ima tovorno vozilo pravico dostopa do določenih con podjetja ...). Trenutni sistem ne omogoča največje učinkovitosti delovanja, saj je za dokončanje postopka prehoda vozila potrebno veliko človeške interakcije, kar predstavlja visoko

časovno zahtevnost v primerjavi s sistemom, ki bi v ta proces vključil avtomatizacijo. Prav tako ni orodij, ki bi izboljšala učinkovitost na podlagi zgodovine obiskov (npr. samodejna izpolnitev podatkov o vozilu ali vozniku). S takšnimi samodejnimi vnosi bi se povečala učinkovitost delovanja sistema, saj bi varnostno osebje potrebovalo bistveno manj časa za identifikacijo, s tem pa bi pridobili čas za druga, pomembnejša dela. Zmanjšala bi se tudi možnost vnosa napačnih podatkov v sistem, saj ima ročno vnašanje visok faktor človeške napake (utrujenost, slaba vidljivost zaradi vremenskih razmer, površnost ...). Poskrbeti bi bilo potrebno tudi za ustrezno shranjevanje podatkov, saj Excel ni produkt, ki bi nudil dovolj dobro zaščito pred morebitnim vdorom.

3.2.3 Identifikacija morebitnih težav pri vpeljavi novega sistema

Pred načrtovanjem novega informacijskega sistema je treba jasno določiti namen sistema ter določiti vse funkcionalnosti, ki bodo sistem predstavljale. V ta proces načrtovanja in vpeljave je potrebno vključiti vse akterje, ki bodo v neposrednem ali posrednem stiku z novim sistemom (vodstveni kadri, končni uporabniki ...). S takim vključevanjem se želimo izogniti morebitni slabi pripravljenosti ali zavračanju novega načina dela s strani končnih uporabnikov, ki bodo najbolj občutili spremembo dela zaradi vpeljave novega informacijskega sistema.

Pri vpeljavi bo potrebno prav tako poskrbeti za ustrezen prenos znanja uporabe informacijskega sistema, za kar bodo določeni ustrezno kvalificirani strokovnjaki s strani izvajalca, ki bodo na uporabniku prijazen način predstavili uporabo in delovanje sistema.

3.2.4 Cilji podjetja

Podjetje želi vpeljati informacijski sistem, ki bo obravnaval vse pomanjkljivosti trenutnega načina prehoda motornega prometa v območje podjetja.

Takšen sistem bo predstavljal avtomatizirano rešitev ob uporabi dobrih praks in priznanih tehnologij. Ta bo optimiziran za potrebe učinkovitega delovanja zaposlenih na teh delovnih mestih ter bo nudil funkcionalnosti, ki bodo pripomogle k izboljšanju varnosti.

3.2.5 Funkcionalne specifikacije

Opredelitev funkcionalnih specifikacij za namen načrtovanja in razvoja informacijskega sistema:

- Sistem mora omogočati beleženje prehodov s pomočjo zajema slike ali videa.
- Sistem mora omogočati pomoč pri prepoznavi registrskih tablic.
- Sistem mora omogočiti integracijo z zalednimi sistemi (podatkovna baza).
- Sistem mora omogočati prijavo uporabnikov (avtentikacija).
- Sistem mora omogočati vnos podatkov o obiskovalcih.
- Sistem mora omogočati spremembo podatkov s strani ustrezno pooblaščene osebe.
- Sistem mora omogočati pogled "v živo"preko spletne aplikacije.
- Sistem mora podatke varovati pred nepooblaščenim dostopom.
- Sistem mora omogočati razpošiljanje e-pošte.

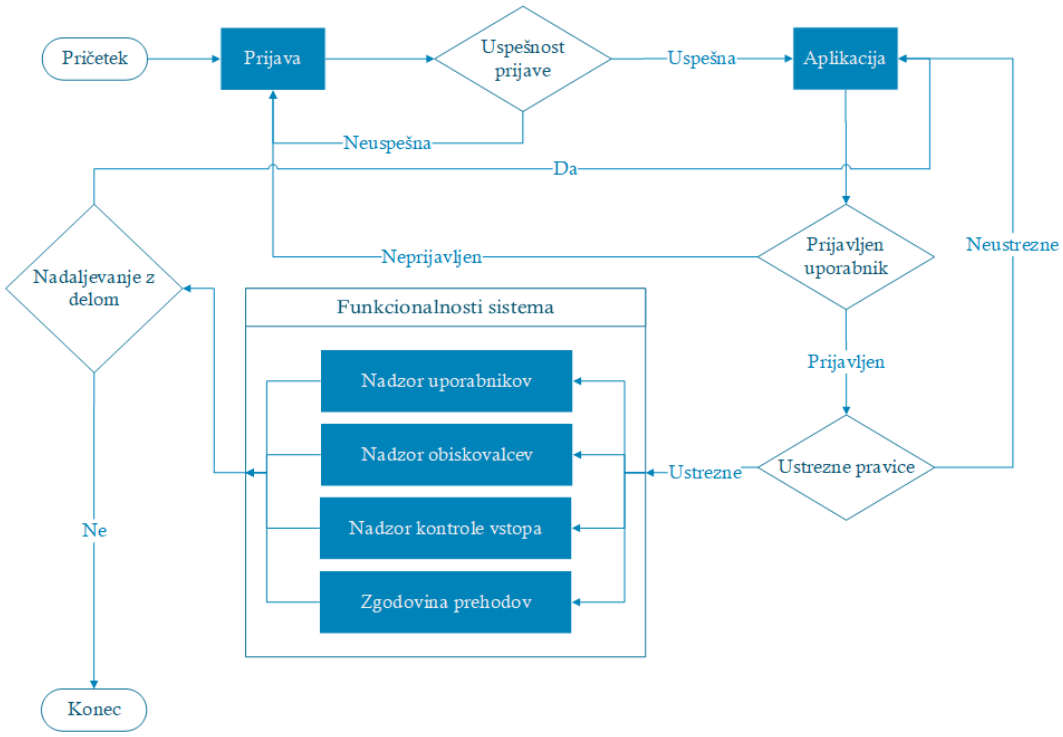
3.2.6 Nefunkcionalne specifikacije

- Rešitev mora biti v celoti dosegljiva preko spletnega brskalnika (brez dodatnih namestitev).
- Delovanje aplikacije za končnega uporabnika mora biti neodvisno od operacijskega sistema.

- Delovanje aplikacije v brskalniku Google Chrome ali Mozilla Firefox.
- Podpora šumnikov.
- Strežnik mora biti kompatibilen za delovanje znotraj hipervizorja Microsoft Hyper-V.

3.2.7 Diagram poteka

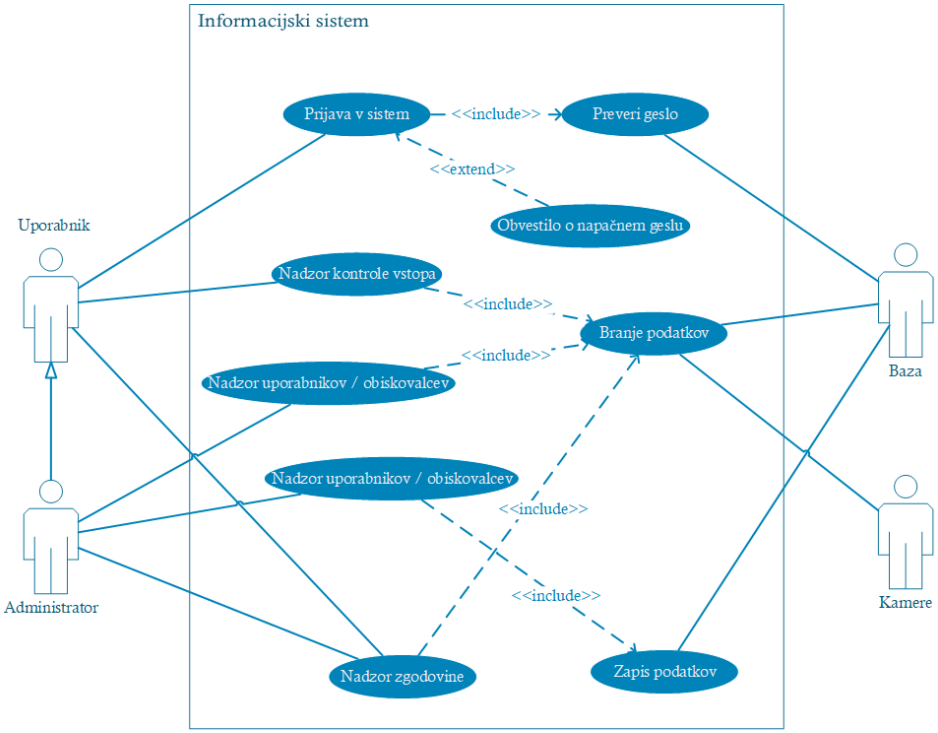
Za preglednejši in učinkovitejši razvoj informacijskega sistema tega najprej predstavimo z grafičnim prikazom v obliki diagrama poteka. Takšen diagram pomembno prispeva h komunikaciji vseh udeleženih v projektu, saj predstavi logiko informacijskega sistema na jasen in pregleden način.



Slika 3.3: Diagram poteka informacijskega sistema.

3.2.8 Diagram primera uporabe

Diagram primera uporabe je poenostavljena predstavitev uporabniške interakcije s funkcionalnostmi znotraj sistema, ki prikaže, kako naj bi naš informacijski sistem dejansko deloval. Z diagramom želimo torej posnemati uporabo v realnem svetu, pri čemer vsem udeležencem v projektu prikažemo, kako naj bi bil sistem strukturiran.



Slika 3.4: Diagram primera uporabe informacijskega sistema.

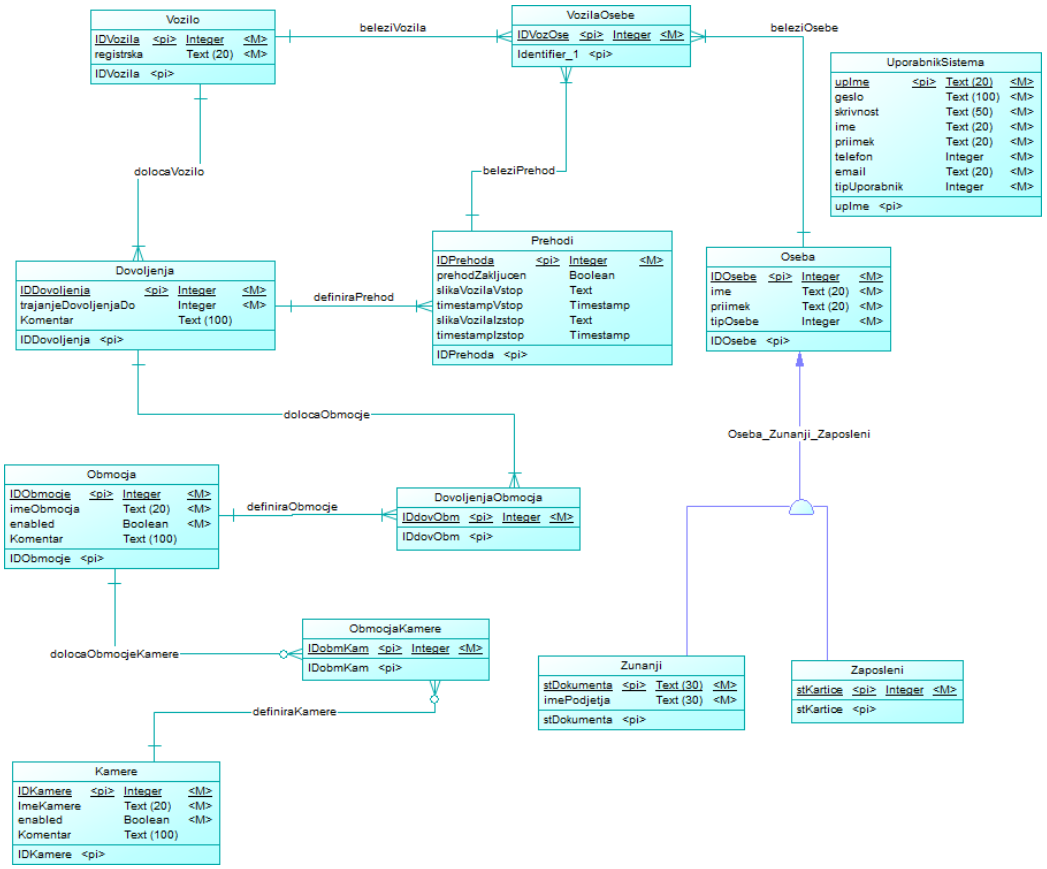
3.2.9 Podatkovna baza

Izdelati želimo podatkovni model za informacijski sistem za avtomatizacijo kontrole vstopa in nadzora prometa. Sistem bo o uporabniku hranil uporabniško ime, popolno ime, e-poštni naslov in telefonsko številko. E-pošta uporabnika in telefon sta obvezna podatka. Uporabniki so poleg tega razdeljeni na navadne uporabnike in administratorje. Administratorji imajo

možnost upravljanja in nadzora nad uporabniki sistema in obiskovalci. Sistem bo nadzoroval večje območje, sestavljeno iz različnih con. Vsaka cona ima unikatno identifikacijsko številko, naziv in komentar o lokaciji. Posamezno cono lahko prečijo le vozila z ustreznim dovoljenjem. Ob vnosu podatkov je trebno vnesti tudi čas veljave dovoljenja. V podjetje lahko z vozilom vstopijo tako zunanji obiskovalci kot zaposleni v podjetju. O zunanjih obiskovalcih se ob priložitvi identifikacijskega dokumenta zabeleži številka tega dokumenta, ime, priimek in podjetje, za katerega obiskovalci delajo. Za zaposlene znotraj podjetja se beleži le številka službene kartice oziroma ime in priimek zaposlenega. V primeru, da je v vozilu več oseb, se popiše vse osebe. Hkrati ob vstopu vozila zabeležimo še registrsko številko. En voznik lahko vozi več vozil, eno vozilo lahko pripada večim voznikom. Ob prehodih se beleži čas vstopa in izstopa.

Za izdelavo konceptualnega, logičnega in fizičnega modela smo uporabili program **Sybase PowerDesigner**. Nad relacijami se je prav tako izvedla normalizacija:

- I. normalna oblika.
  - Definiranje ključnih atributov.
  - Atributi v relacijah nimajo ponavljajočih skupin.
- II. normalna oblika.
  - Entitetnim tipom je bil dodeljen enolični identifikator.
  - Atributi relacije so odvisni od celotnega ključa.
- III. normalna oblika.
  - Preverjanje parcialnih odvisnosti.



Slika 3.5: Prikaz konceptualnega modela.

3.3 Načrtovanje in razvoj

3.3.1 Pregled produkcijskega okolja

Končni produkt diplomskega dela bo postavljen v poslovno navidezno omrežje 10. Za strežnik je rezerviran IP naslov 192.168.10.10. V istem omrežju je za nadzorne kamere namenjeno 20 rezerviranih IP naslovov v razponu od vključno 192.168.10.50 do vključno 192.168.10.70. IP naslovi za kamere se lahko določijo poljubno znotraj predvidenega razpona. Logično shemo produkcijskega okolja in rezerviranih naslovov prikazuje slika 3.6.

Za potrebe testnega obdobja bo rezervirana ena virtualna naprava s specifikacijami:

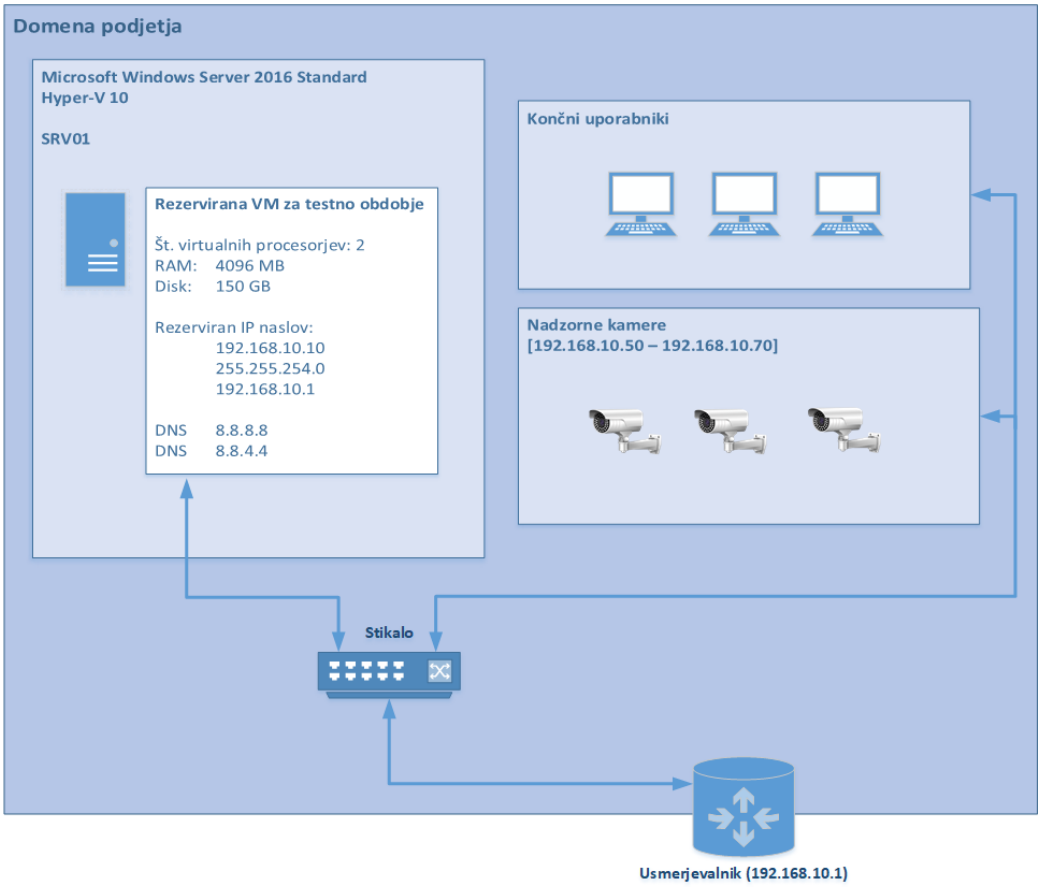
- Število navideznih jeder: 2.
- RAM: 4096 MB.
- Disk: 150 GB.
- Rezerviran IP naslov:
  - 192.168.1.10
  - 255.255.254.0
  - 192.168.10.1

Podatki o fizičnem strežniku:

- Model: IBM System x3650 M4
- Procesor: 2x Intel Xeon CPU E5-2620 v2 @ 2.10GHz
- Število procesorjev: 2x (6 jeder, 12 logičnih procesorjev).
- Diski strežnika:
  - Število diskov: 2.

  - Število navideznih diskov: 1.
  - Tip diskov: IBM SAS 278.875 GB
  - RAID: 1.

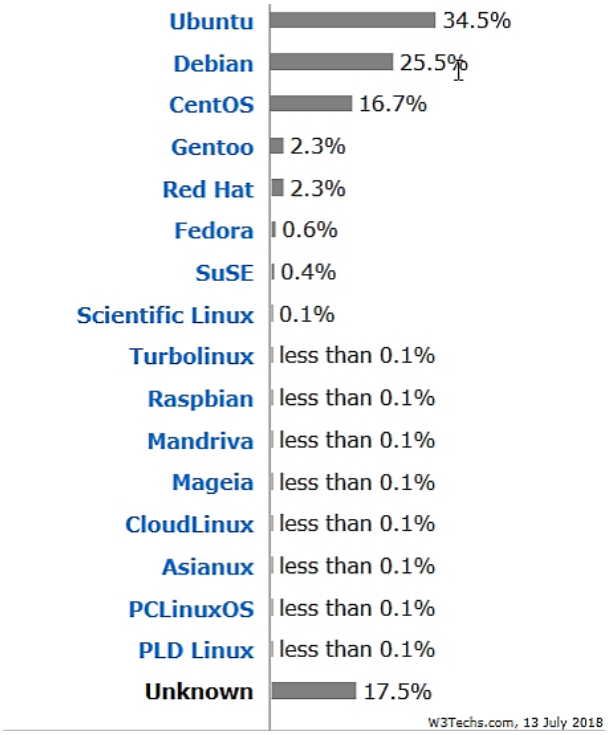
- Storage: IBM Storwize V3700
  - IBM SAS diski v skupni velikosti 12 TB.



Slika 3.6: Produkcijsko okolje - logična shema.

3.3.2 Izbira operacijskega sistema

Po podatkih spletnega portala W3Techs [12], ki svoja poročila posodablja dnevno, se v dobrih 68 odstotkih za uporabo spletnih strežnikov uporablja Unix (v isto kategorijo spadajo tudi Unix-like operacijski sistemi). Od tega s 40 odstotki prevladujejo Linux distribucije. Preostali delež z 32 odstotki pokriva operacijski sistem Microsoft Windows.



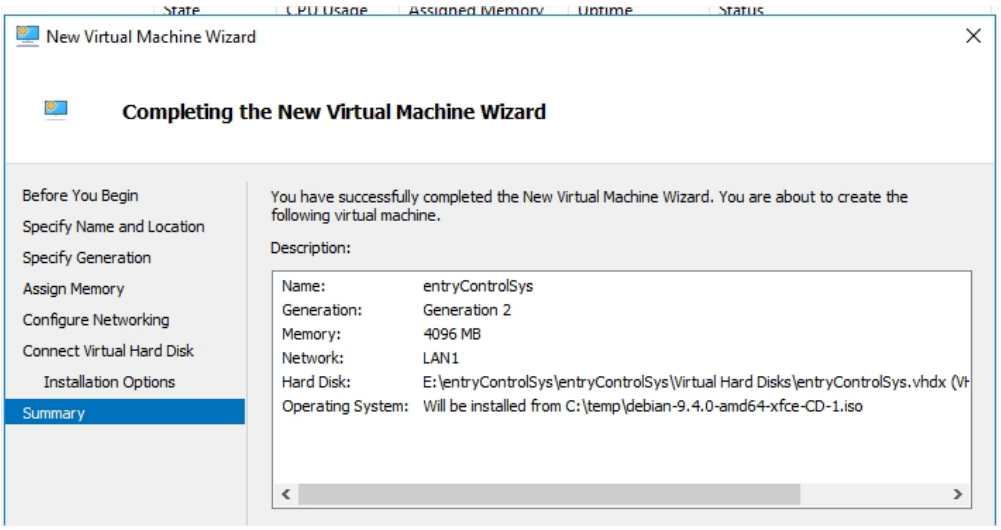
Slika 3.7: Prikaz najpogostejše uporabljenih Linux distribucij za namene spletnih strežnikov [5].

V našem primeru je bil za razvoj in kasnejšo produkcijo izbran operacijski sistem Linux Debian 9 zaradi poglavitnih razlogov:

- Posodabljanje.
  - Dobra dokumentacija za posodobitve preko sistema "apt-get". Ponovni zagon je ponavadi potreben le takrat, ko se posodobi jedro, kar ni zelo pogosto.
  - Redki cikli posodobitev.
- Varnost.
  - Paketi so preverjeni pred vsako Debian izdajo.
- Stabilnost.
- Samodejna konfiguracija programskih paketov.
  - Širok nabor programskih paketov (večina programov deluje brez dodatnih nastavitev konfiguracij po namestitvi na sistem).
  - Samodejna namestitev dodatnih paketov (angleško "dependencies"), ki jih aplikacija potrebuje za delovanje.

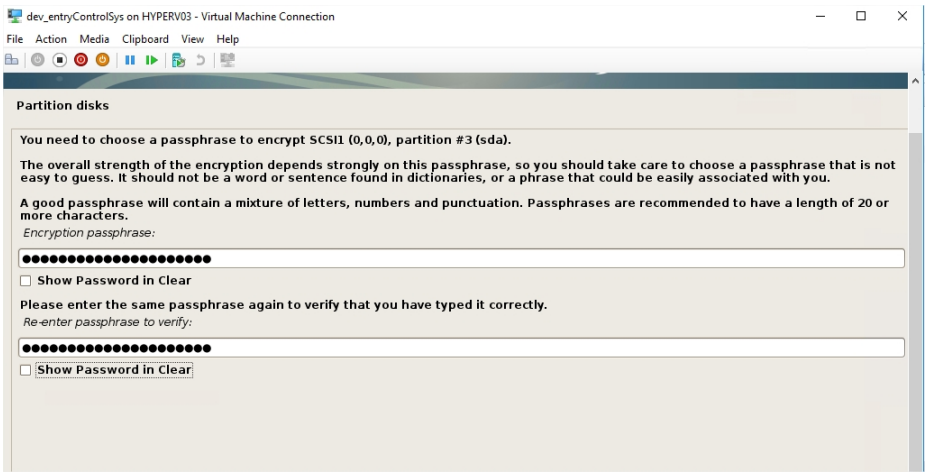
3.3.3 Namestitev in priprava operacijskega sistema

Želimo ustvariti identično okolje, kot nam bo na voljo v produkcijskem okolju. Tako smo na novo ustvarjeni virtualni napravi znotraj Microsoft Hyper-V strežnika dodelili enag obseg moči in virov, kot ji ga je namenjenega v produkcijskem okolju za čas testiranja informacijskega sistema (povzetek nastavitev virtualne naprave prikazuje slika 3.8). Strežnik bo v okolju predstavljen kot "**entryControlSys**".



Slika 3.8: Povzetek nastavitev virtualne naprave za potrebe razvojnega okolja.

Eno izmed glavnih vodil pri razvoju informacijskega sistema je tudi varnost. Tako je za preprečitev nepooblaščenega dostopa kriptiran celoten disk operacijskega sistema. S tem se zavarujemo, da bi v primeru fizične odtujitve strežnika nepridipravi prišli do občutljivih podatkov (oziroma jim je dostop do podatkov bistveno otežen).



Slika 3.9: Vnos kompleksnega gesla za kriptiranje diska.

Strežniku nastavimo statičen IP naslov. Omrežne nastavitve se v operacijskem sistemu Debian konfigurirajo preko datoteke **/etc/network/interfaces**.

```
~$ cat /etc/network/interfaces

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
   address 192.168.10.10
   netmask 255.255.254.0
   gateway 192.168.10.1
```

Prav tako ne smemo pozabiti na DNS strežnike. Za naš primer ni pomembno, da so vnešeni DNS strežniki hkrati tudi domenski, zato bomo uporabili Googleove DNS strežnike.

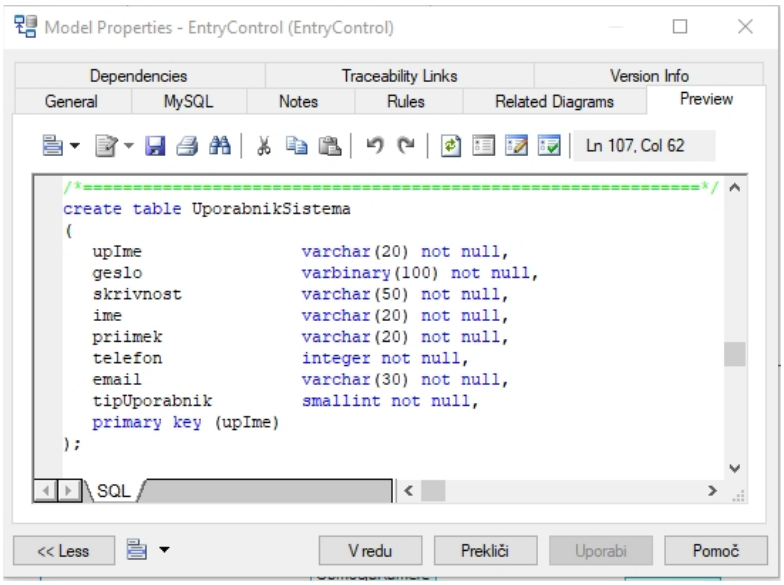
```
~$ cat /etc/resolv.conf

nameserver 8.8.8.8
nameserver 8.8.4.4
```

Razvojno okolje je s tem pripravljeno za začetek razvoja informacijskega sistema.

3.3.4 Izdelava fizičnega modela baze

Za izdelavo fizičnega modela smo uporabili orodje **"Model Properties - Preview"** znotraj programa **Sybase PowerDesigner**. Tako pridobimo

SQL kodo za izdelavo naše podatkovne baze. Predhodno smo za DBMS izbrali MySQL.



Slika 3.10: Prikaz SQL kode za izdelavo podatkovne baze znotraj programa.

3.3.5 Namestitev SQL strežnika

Namestitev in konfiguracijo podatkovne baze izvedemo preko Debian terminala.

```
~$ sudo apt-get install apache2 php libapache2-mod-php
   mysql-client mysql-server phpmyadmin
```

Posamezni paketi predstavljajo:

- apache2: Apache HTTP strežnik.
- php: Obvezen paket za podporo delovanja PHP-ja.
- libapache2-mod-php: PHP modul za podporo delovanja Apache HTTP strežnika.
- mysql-client: MySQL/MariaDB SQL klient.

- mysql-server: MySQL/MariaDB SQL strežnik.
  - MariaDB je sedaj privzeti SQL strežnik za "mysql-server"paket v različici Debian 9.x [8].
- phpmyadmin: Spletno administrativno orodje za podatkovne baze.

Po uspešni namestitvi in konfiguraciji delovanje baze preverimo z naslednjim ukazom.

```
~$ systemctl status mariadb
```

```
zadnikar@entryControlSys:~$ systemctl status mariadb
● mariadb.service - MariaDB database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
enabled)
   Active: active (running) since Wed 2018-09-12 15:59:39 CEST; 23min
 ago
 Main PID: 24317 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 28 (limit: 9830)
   CGroup: /system.slice/mariadb.service
           └─24317 /usr/sbin/mysqld
```

Slika 3.11: MariaDB je uspešno zagnana na našem strežniku.

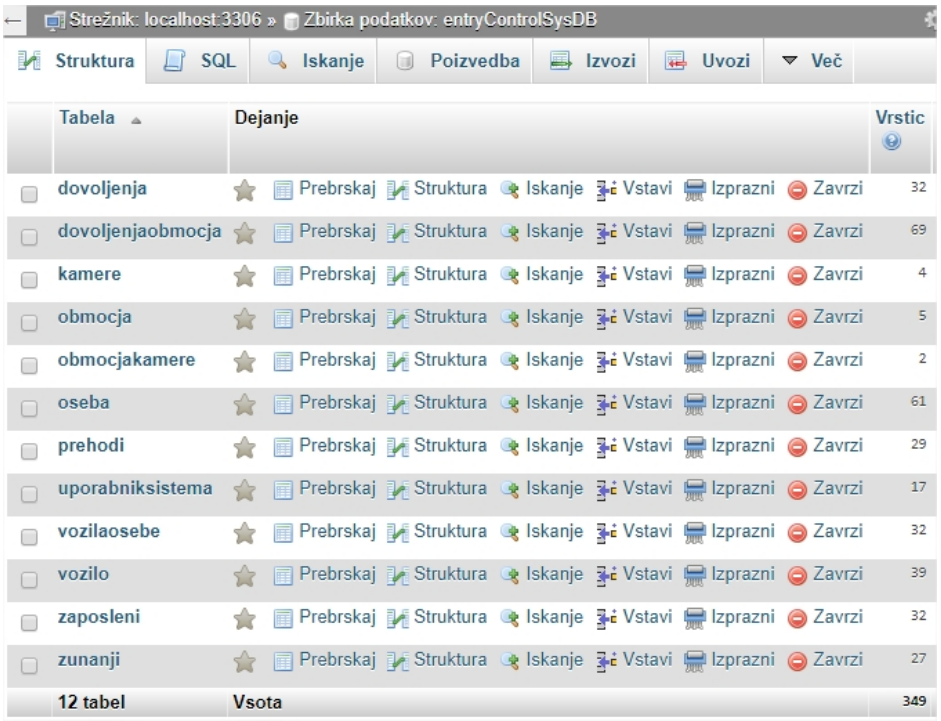
Do spletne administratorske konzole **phpMyAdmin** lahko dostopamo preko brskalnika na naslovu localhost/phpmyadmin. Predtem ustvarimo še uporabnika z root pravicami.

```
# Dostop do MariaDB konzole
~$ sudo mysql -u root -p
MariaDB> create user 'mariadb'@'localhost' identified by
    '**********';
MariaDB> grant all priviliges on *.* to 'mariadb'@'localhost';
# Izbrisemo predpomnilnik
MariaDB> flush privileges;
```

3.3.6 Izdelava podatkovne baze

Izdelava podatkovne baze je potekala preko spletnega vmesnika **phpMyAdmin**.

1. Ustvarimo novo bazo, ki smo jo poimenovali **"entryControlSysDB"**.
2. Zaženemo SQL kodo, pridobljeno v poglavju 3.3.4.
3. Preverimo log-e za morebitne napake.



Slika 3.12: Pregled na novo ustvarjenih tabel v spletnem vmesniku phpMyAdmin.

3.3.7 Spletni portal

Razvoj spletnega portala je potekal v skladu z diagramom poteka v poglavju 3.2.7. Struktura spletnega mesta je prikazana na sliki 3.13.

Slika 3.13: Struktura spletnega portala za nadzor kontrole vstopa.

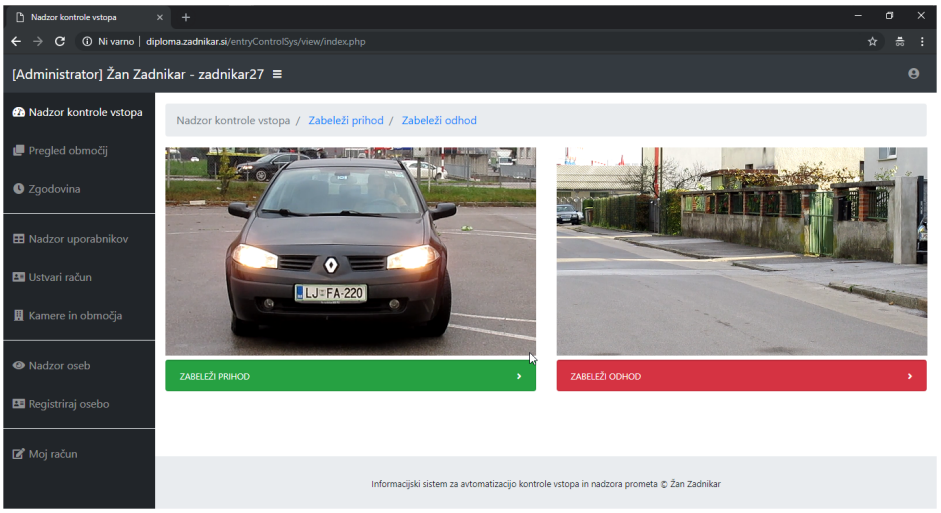
Za osnovo spletne strani smo uporabili Bootstrap predlogo startbootstrap-sb-admin, objavljena pod MIT licenco, ki med drugim dovoljuje uporabo za komercialne namene in distribucijo [11]. Predloga je dosegljiva na spletnem naslovu:

`https://github.com/BlackrockDigital/startbootstrap-sb-admin`.

Vsebino prenešenega paketa razširimo in kopiramo v Apache HTTP strežnik. Delovanje preverimo z odprtjem spletne strani `localhost`. Strukturo prenešene Bootstrap predloge sedaj preučimo in prilagodimo do te mere, da bo ustrezala

zamišljeni grafični podobi.

- Prevod angleških oznak v slovenščino.
- Odstranitev odvečnih elementov in prilagoditev našim zahtevam.



Slika 3.14: Grafični vmesnik po prilagoditvi.

- Prilagoditev prikaza tabel.
  - Želimo najboljšo uporabniško izkušnjo, zato bosta prikaz in morebitna sprememba urejena preko dinamične tabele po principu **"izberi in klikni"**.
  - Za delo s tabelami bomo uporabili knjižnico **DataTables** [4]. Za želeno interakcijo z uporabnikom bomo na spletni aplikaciji dodali JS funkcije, ki bodo skrbele za pridobitev podatka o tem, kaj je uporabnik kliknil.

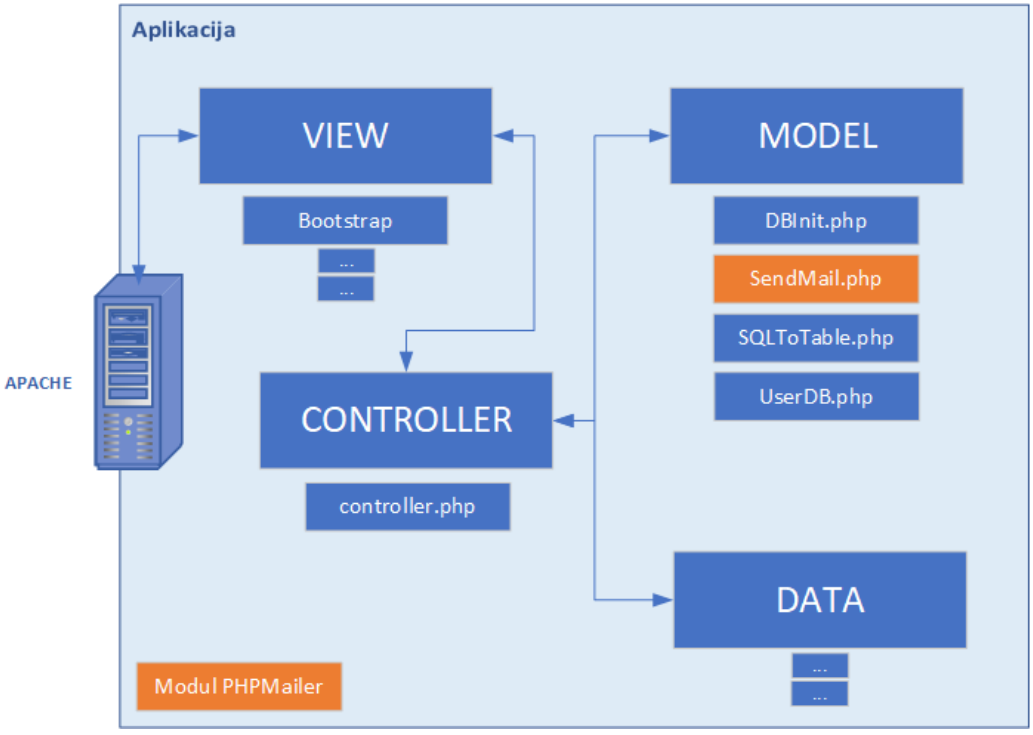
    Primer prepoznave uporabniškega klika v tabeli.

```
var table = $('#dataTable').DataTable();
$('#dataTable tbody').on('click', 'tr', function () {
    var data = table.row( this ).data();
```

```
        // data[0] je nas identifikator
        window.location.href = 'http://' +
            window.location.hostname +
            '/entryControlSys/view/editaccount.php?upIme=' +
            data[0];
} );
$('#dataTable tbody').css('cursor','pointer');
```

3.3.8 Implementacija funkcionalnosti

Implementacija logike in funkcionalnosti je potekala strukturirano v več posameznih korakih. PHP arhitektura naše spletne aplikacije je predstavljena na sliki 3.15.



Slika 3.15: Pregled glavnih modulov spletne aplikacije. Dobljen modul PHPMailer je objavljen pod LGPL 2.1 licenco [6].

Razlaga posameznih modulov:

- **Controller**: jedro delovanja spletne aplikacije. Vsako zahtevo s strani uporabnika, ki se pošlje na strežnik, obdela datoteka **controller.php**. V njej so napisana pravila in funkcije, ki ustrezno procesirajo nadaljnje korake. Uporabnik neposredne interakcije s preostalimi moduli nima.

  Osrednje funkcije Controllerja

  - **Preverjanje avtentikacije**: Uporabnik brez ustreznih prijavnih podatkov nima dostopa do našega sistema. Gesla o uporabniških računih se v SQL bazi hranijo šifrirana. Prav tako se baza nahaja na šifriranem disku.

    Primer preverjanja, ali je uporabnik že prijavljen v sistem.

    ```
    // Podatki o prijavljenem uporabniku se shranijo v
        spremenljivki $_SESSION[session_id()]
    if (!isset($_SESSION[session_id()]))header('Location:
        http://' . $_SERVER['SERVER_NAME'] .
        '/entryControlSys/view/login.php');
    ```

  - **Preverjanje avtorizacije**: Sistem mora znati ločevati med tipi uporabnikov (navaden uporabnik oziroma administrator) in prikazati vsebino glede na ustrezno avtorizacijo.

    Primer omejevanja prikaza vsebine, v kolikor gre za uporabnika brez ustreznih dostopnih pravic.

    ```
    // $_SESSION[session_id()][6] hrani podatke o tipu
        uporabnika
    if ($_SESSION[session_id()][6] === 0){
      echo 'Nepooblascen dostop!';
      exit;
    }
    ```

– **Razred Controller**: ključne funkcije, ki omogočajo delovanje sistema.

Primer funkcije, ki preveri ustreznost prijave.

```
class Controller {
   public static function login($upIme, $geslo){
      if (UserDB::validLoginAttempt($upIme, $geslo)){
         self::connectSessionIdUserInfo($upIme);
         self::addSessionId();
         return true;
      } else
         return false;
   }
}
```

– **Obravnava metod**: vse POST in GET zahteve se obdelajo v datoteki **controller.php**. Vsaka poslana zahteva s strani spletne aplikacije praviloma vsebuje podatek o njenem izvoru, kar omogoča identifikacijo in nadaljnje procesiranje.

Primer obdelave POST zahtevka, ki pride s strani editaccount.php.

```
if ($_SERVER["REQUEST_METHOD"] === "POST"){
   if(isset($_POST["izvor"])){
      switch ($_POST["izvor"]) {
      case "editaccount.php":
      if(UserDB::updateUserData($_POST["upIme"],
          $_POST["ime"], $_POST["priimek"],
          $_POST["telefon"], $_POST["email"],
          $_POST["tipUporabnik"])){
         if($_POST["upIme"] ===
             $_SESSION[session_id()][0]){
           Controller::connectSessionIdUserInfo($_POST["upIme"]);
```

```
                Controller::redirect(SERVER_NAME .
                    'entryControlSys/view/
                    myaccount.php?info=userUpdated');
                } else {
                Controller::redirect(SERVER_NAME .
                    'entryControlSys/view/
                    editaccount.php?userUpdated=' .
                    $_POST["upIme"]);
                }
            }
            break;
            }
        }
    }
```

- **Model**: vsebuje vse dodatne module, ki razširjajo funkcionalnosti sistema (priklop na bazo, pošiljanje e-pošte, klici na bazo ...) in predstavljajo ločene enote. Komunikacijo z moduli vzpostavi le **Controller**. Eden od ključnih modulov je povezava na **podatkovno bazo**. Za povezavo med PHP strežnikom in MariaDB skrbi **PDO**. Glavni prednosti uporabe PDO sta **varnost** (npr. zaščita pred SQL injekcijami) in **uporabnost** (številne funkcije, ravnanje z izjemami, podpora različnim bazam ...).

  Primer dodajanja novega prehoda v bazo.

```
// DBInit::getInstance() vrne instanco, ki predstavlja
   povezavo na naso bazo
class UserDB {
    public static function newPrehod($slikaVozilaVstop){
        $dbh = DBInit::getInstance();
        $statement = $dbh->prepare("INSERT INTO prehodi
            (slikaVozilaVstop, timestampVstop, prehodZakljucen)
```

```
            VALUES (:slikaVozilaVstop, NOW(), 0)");
        $statement->bindParam(":slikaVozilaVstop",
            $slikaVozilaVstop);
        $status = $statement->execute();
        $IDPrehoda = $dbh->lastInsertId();
        $_SESSION["IDPrehoda"] = $IDPrehoda;
        return $status;
    }
}
```

- **View**: je sestavljen iz številnih datotek in tehnologij (HTML, JS, Ajax ...), ki so gradniki dobre uporabniške izkušnje. Izpostaviti gre funkcijo za zajem in pošiljanje zajete slike na strežnik, ki se sproži asinhrono in tako uporabniku omogoči, da lahko nemoteno nadaljuje s svojim delom.

  Primer zajema in pošiljanja slike s strani index.php.

```
$('#zabeleziPrihod').on('click', function() {
    var video = $("#prihod01").get(0);
    var canvas = document.createElement("canvas");
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    canvas.getContext('2d').drawImage(video, 0, 0,
        canvas.width, canvas.height);
    var img = canvas.toDataURL("image/png");
    img = img.replace(/^data:image\/(png|jpg);base64,/,
        "");
     $.ajax(
      {
          type: 'POST',
          url: '../controller/controller.php',
          contentType:
              "application/x-www-form-urlencoded",
```

```
            data: {
                imageArrival: img
            },
            success: function(){
              window.location.href = 'http://' +
                  window.location.hostname +
                  '/entryControlSys/view/arrival.php';
            },
            error: function () { alert('Napaka.'); }
        });
    });
```

- **OpenALPR**: za prepoznavo registrskih tablic smo uporabili funkcionalnosti sistema OpenALPR, ki na podlagi slike vozila odgovori s podatki, ki so vezani na samo vozilo. OpenALPR omogoča tako lokalno namestitev na operacijski sistem kot uporabo spletnega APIja. Dokumentacija je dosegljiva na

  `http://doc.openalpr.com`.

  Na podlagi testiranj uspešnosti prepoznave tablic pri uporabi lokalne namestitve in spletnega APIja je bilo ugotovljeno, da je prepoznava bistveno boljša pri uporabi spletnega APIja, kar je bil glavni razlog za njegovo implementacijo.

  Povezava z našega PHP strežnika na OpenALPR API je urejena z ukazom **curl**. Do rezultatov dostopamo v formatu JSON.

```
$rezultati = shell_exec('curl -X POST -F image=@/' .
    $_SESSION["fullPathArrival"] . '
    "https://api.openalpr.com/v2/recognize?recognize_vehicle=1
    &country=eu&topn=1&secret_key=*****"');
```

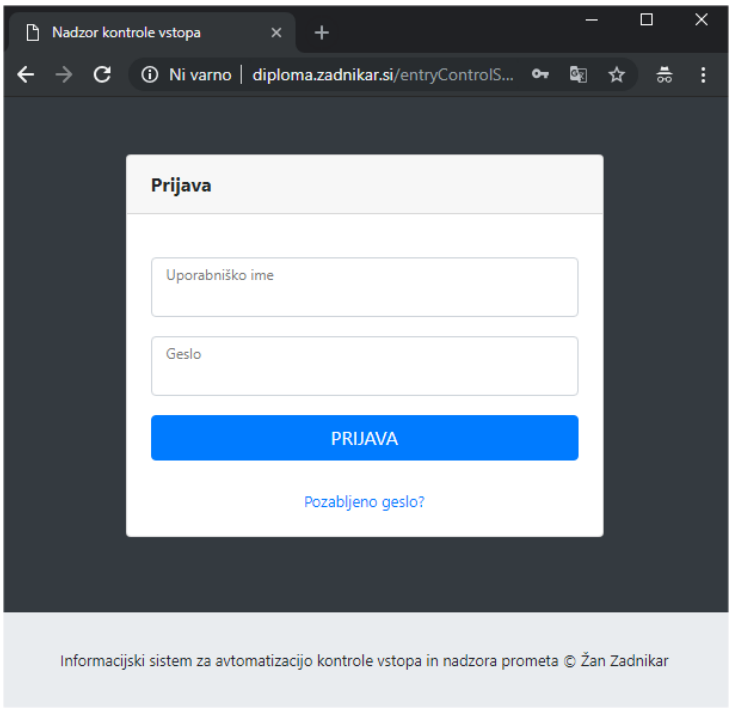
```
$rezJSON = json_decode($rezultati, true );
```

3.3.9 NAT in DNS

Za potrebe testiranja iz zunanjega sveta je bilo na usmerjevalniku dodano NAT pravilo, ki preusmerja HTTP promet (port 80) iz javnega IPja usmerjevalnika na lokalni naslov Apache strežnika. V tej fazi šifriranje prometa (HTTPS) ni nastavljeno, saj gre za obdobje testiranja.

Lokalni naslov strežnika: 192.168.10.10.
Javni naslov usmerjevalnika: 84.255.227.185.

Da bo dostop do spletne aplikacije lažji, smo v imenski strežnik (DNS) dodali vnos **diploma.zadnikar.si. TTL 3600 A 84.255.227.185**.



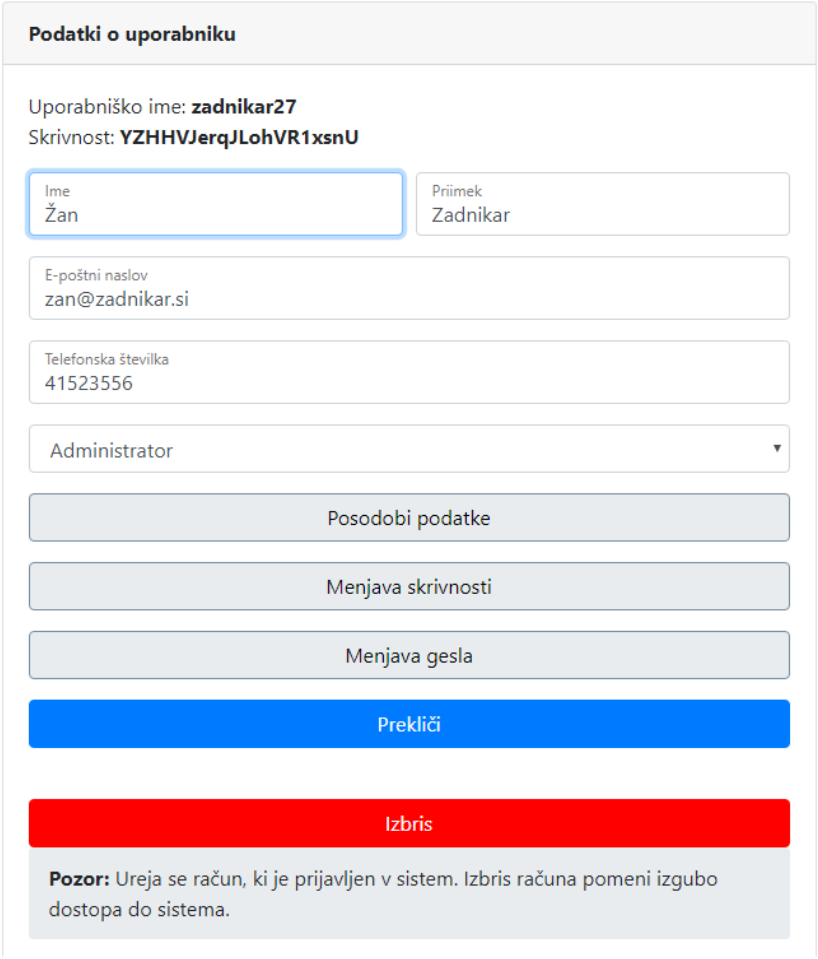
Slika 3.16: Prikaz brskalnika Google Chrome ob odprtju spletne aplikacije na naslovu `http://diploma.zadnikar.si`.

Poglavje 4

Funkcionalnosti sistema

Med razvijanjem informacijskega sistema in ob upoštevanju vseh funkcijskih specifikacij smo v želji po učinkovitem in uporabniku prijaznem informacijskem sistemu razvili naslednje glavne funkcionalnosti:

- **Prijava uporabnika v sistem**:
  - Samodejna generacija na novo ustvarjenih uporabniških imen.
  - Šifriranje uporabniških gesel.
  - Preverjanje ustreznosti prijavnih podatkov na vsaki podstrani spletne aplikacije.
- **Pridobitev gesla v primeru pozabe**.
- **Nadzor uporabnikov sistema**:
  - Sprememba osebnih podatkov.
  - Sprememba kontaktnih naslovov.
  - Spremembe podatkov o prijavi.
  - Izbris uporabniškega računa.
  - Pošiljanje podatkov o uporabniškem računu preko e-pošte.

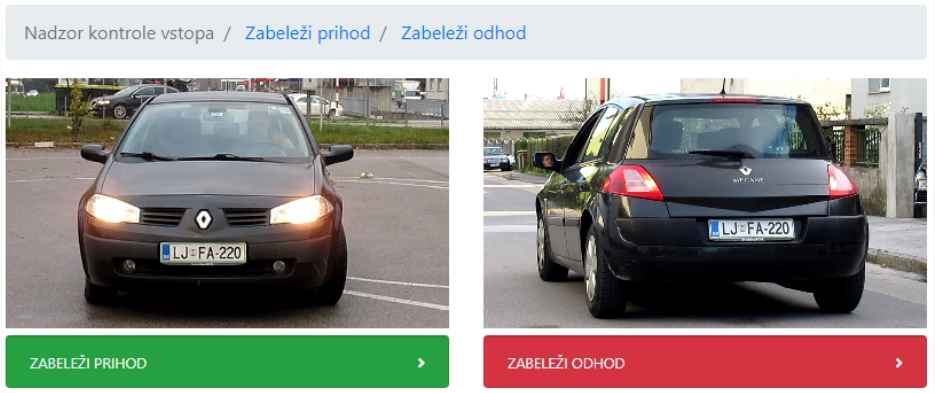


Slika 4.1: Grafični vmesnik za delo pri določenem uporabniku sistema.

- **Nadzor obiskovalcev**:
  - Kategorizacija obiskovalcev (zaposleni oziroma zunanji obiskovalci).
  - Sprememba osebnih podatkov.
  - Izbris obiskovalcev s strani uporabnika je zaradi beleženja zgodovine prehodov onemogočen.
- **Zgodovina prehodov**:
  - Pregledno iskanje v zgodovini prehodov na podlagi različnih kriterijev (osebni podatki, številka osebnega dokumenta, registrska

tablica ...).

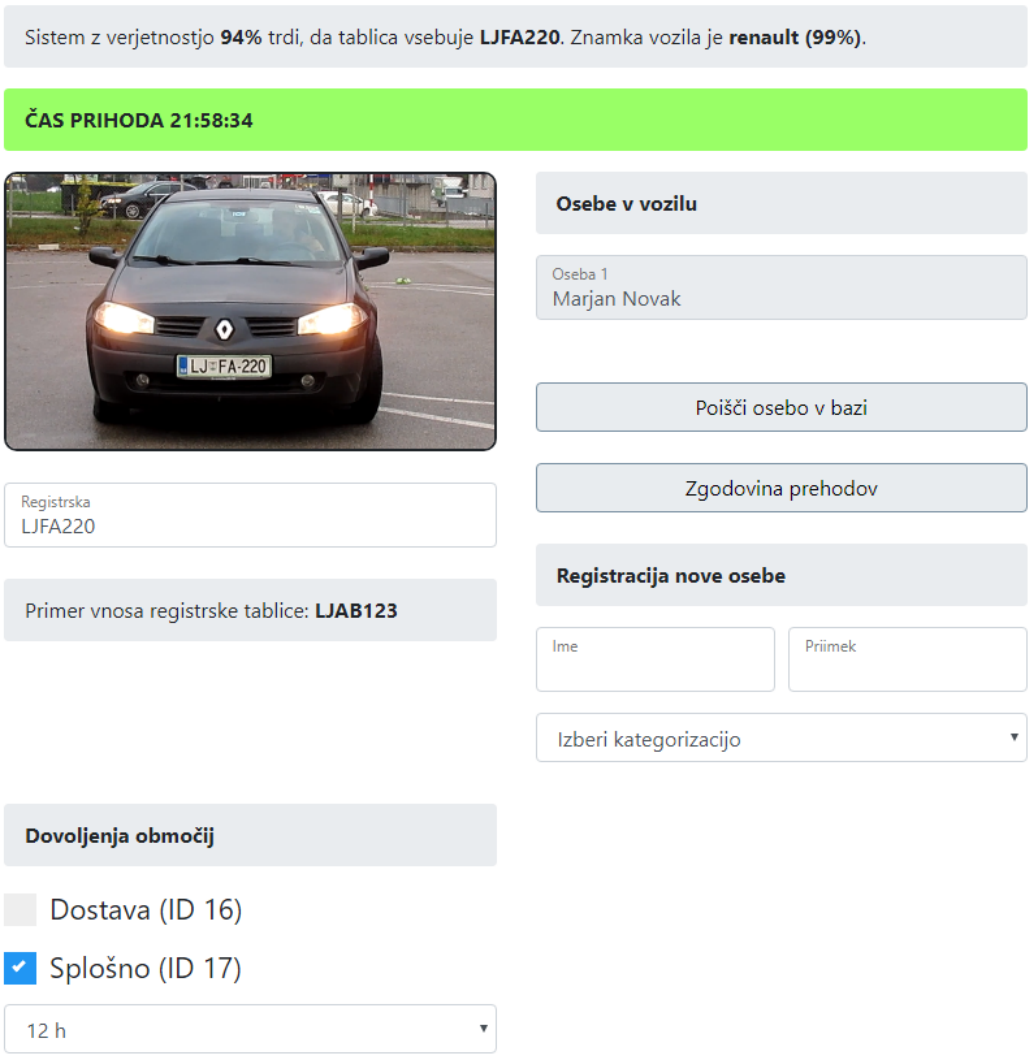
- **Nadzor kontrole vstopa**:
  - Nadzor kontrole vstopa je glavna funkcionalnost sistema, ki uporabnikom omogoča tekoče in učinkovito delo pri beleženju prihodov in odhodov.



Slika 4.2: Nadzor na vstopni in izstopni točki.

  - Ob zabeležitvi prihoda ali odhoda se zajame slika in sproži asinhrono procesiranje prepoznave tablice (slika 4.3). Nato izberemo osebe v vozilu na podlagi zgodovine prehodov oziroma iz baze oseb. V kolikor oseba prvič vstopa na območje našega podjetja, jo v sistem dodamo ročno. Vozilu prav tako določimo dovoljenja do posameznih območij znotraj podjetja.

- **Pregled območij**:
  - Možnost samodejnega zaznavanja vozil ob namestitvi ustrezne strojne opreme v produkcijsko okolje oziroma ob razvitju dodatnega modula za zaznavanje gibanja (trenutno omogočen ročni zajem slike). Funkcionalnost prepozna vozila na posameznih območjih in preveri, ali ima vozilo ustrezno dovoljenje. V primeru kršitev se izvede postopek, določen s strani podjetja (npr. obvestilo varnostni službi ...).



Slika 4.3: Grafični vmesnik za dodajanje podatkov o novem prihodu.

Poglavje 5

Sklep

Glavni cilj diplomskega dela je bil razviti varen in učinkovit informacijski sistem, ki bo s pomočjo informacijskih tehnologij ponudil učinkovito upravljanje in avtomatizacijo kontrole vstopa. Ob upoštevanju zahtev po kakovosti in varnosti smo skozi različne razvojne faze v skladu s tradicionalnim življenjskim ciklom prišli do stanja, ko se lahko informacijski sistem integrira v produkcijsko okolje za namene testiranja. Tu bi skupaj z vsemi akterji, neposredno ali posredno povezanimi z novim sistemom, preverili delovanje aplikacije in poskušali odgovoriti na vprašanje, ali novi sistem odpravlja vse tiste slabosti, ki smo jih izpostavili pri identifikaciji pomanjkljivosti. Istočasno bi preverili, ali sistem vključuje tiste funkcijske specifikacije, ki so bile podane ob začetku diplomskega dela. Sledilo bi daljše obdobje za testiranje aplikacije, kjer bi uporabniki lahko podali svoje mnenje in predloge, kako narediti sistem še boljši.

Bistvena izboljšava na novo razvitega sistema je vsekakor varnost. Podatki o prehodih, ki se v trenutno uporabljenem sistemu zapisujejo v različne Excel datoteke, so sedaj združeni v eni podatkovni bazi, do katere imajo dostop samo pooblaščene osebe. Baza se hkrati nahaja na šifriranem disku, ki se dnevno varnostno kopira (frekvenca varnostnega kopiranja v produkcijskem okolju bo določena s strani politike podjetja). Tako se izognemo morebitni izgubi podatkov. Uporaba novega sistema predvideva tudi zmanjšanje napak

pri vpisovanju registrskih tablic, saj je na voljo pomoč v obliki samodejnega zaznavanja tablic (poudarek na celovitosti podatkov). V kolikor sistem ni zmožen prepoznati registrske tablice zaradi tehničnih ali drugih ovir (npr. slabe vremenske razmere, sneg na tablici, nedelovanje IP kamere ...), lahko podatke o tablici ročno vnesejo uporabniki sistema. V vsakem primeru se zaradi varnosti v arhiv shrani tudi slika vozila ob vstopu in izstopu iz podjetja.

Možnosti za izboljšave in novosti je veliko. Prvi korak ob poteku testnega obdobja v produkcijskem okolju bi bil optimizacija tistih funkcionalnosti, ki so ključnega pomena za naš informacijski sistem. Nato bi vpeljali spremembe na podlagi povratnih informacij s strani uporabnikov, v kolikor bi ti imeli predloge glede izboljšanja delovanja sistema. Pri nadaljnjem razvijanju aplikacije bi bilo v bližnji prihodnosti smiselno dodati nove module. Eden izmed takšnih je modul za zaznavanje gibanja, ki bi glede na potrebe in želje naročnika razširil nabor novih funkcionalnosti informacijskega sistema (samodejno zaznavanje vozil, zaznavanje vdorov ...). Delo bi zaposlenim prav tako lahko olajšali z razvojem mobilne aplikacije, s katero bi omogočili zaznavanje tablic in vpisovanje podatkov tudi na terenu.

Dober informacijski sistem je živ informacijski sistem. To pomeni, da ga moramo nenehno usmerjati k izboljšavam in novostim. Le tako ga lahko ohranimo učinkovitega in uporabnega.

Literatura

[1] An Overview of the Tesseract OCR Engine. Dosegljivo: https://github.com/tesseract-ocr/docs/blob/master/tesseracticdar2007.pdf. [Dostopano: 17. 03. 2018].

[2] Bootstrap Get Started. Dosegljivo: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp. [Dostopano: 19. 03. 2018].

[3] Gary Bradski and Adrian Kaehler. Learning opencv: Computer vision with the opencv. In *Learning OpenCV: Computer Vision with the OpenCV*, pages 1–6. O'Reilly Media, 2008.

[4] DataTables Manual. Dosegljivo: https://datatables.net/manual/. [Dostopano: 15. 09. 2018].

[5] Examining the Agile Cost of Change Curve. Dosegljivo: http://www.agilemodeling.com/essays/costOfChange.htm. [Dostopano: 22. 03. 2018].

[6] GNU Lesser General Public License, version 2.1. Dosegljivo: http://www.gnu.org/licenses/lgpl-2.1.html. [Dostopano: 20. 11. 2018].

[7] IT Governance Institute. *COBIT® 4.1: Framework, Control Objectives, Management Guidelines, Maturity*. IT Governance Institute, 2007.

[8] Moving from MySQL to MariaDB in Debian 9. Dosegljivo: https://mariadb.com/kb/en/library/

moving-from-mysql-to-mariadb-in-debian-9/. [Dostopano: 12. 09. 2018].

[9] Mohammed Munassar and A. Govardhan. A comparison between five models of software engineering. In *A Comparison Between Five Models Of Software Engineering*, pages 94–101. IJCSI International Journal of Computer Science Issues, 2010.

[10] Linda Shapiro and George Stockman. Computer vision. In *Computer Vision*, pages 13–15. Pearson, 2001.

[11] The MIT License. Dosegljivo: https://opensource.org/licenses/MIT. [Dostopano: 13. 09. 2018].

[12] Usage of operating systems for websites. Dosegljivo: https://w3techs.com/technologies/overview/operating_system/all. [Dostopano: 13. 07. 2018].