Srednja šola za kemijo,
elektrotehniko in računalništvo

Model avtomatizirane parkirne hiše

Raziskovalna naloga

Avtorja:

David Slatinek

Gregor Zavolovšek

Mentor:

Gorazd Breznik, univ. dipl. inž. rač.

Mestna občina Celje, Mladi za Celje

Celje, marec 2019

Kazalo

Kazalo slik

Kazalo tabel

Povzetek

Pri tej raziskovalni nalogi sva se ukvarjala s pomanjkanjem parkirnih mest v mestih in prihranku na času. Najin cilj je bil narediti model parkirne hiše, ki bi čim bolj zadovoljila stranke in tako prispevala k izboljšanju prometnega položaja v mestih.

Vsak informacijski sistem vsebuje dobro zgrajen strežnik, zato je to prvi del najine raziskovalne naloge. Opisala sva namen strežnikov in njegove komponente, na koncu pa ga sestavila in ga prilagodila za namen te naloge.

Naslednji del je model parkirne hiše v fizični obliki. Za ta namen sva izbrala mikrokrmilnik Arduino. Pojasnila sva kaj je, predstavila strojni del, opisala komponente uporabljene v projektu in fizično naredila model parkirne hiše ter predstavila delovanje programa.

Zatem sva opisala načrtovanje podatkovne baze za ta projekt.

Nato sva zgradila aplikacijo za krmiljenje parkirne hiše s programskim jezikom C#. Predstavila sva diagram poteka programa in predstavila vizualni videz.

Nazadnje sva predstavila aplikacijo za uporabnika in opisala njene funkcije.

Ključne besede

C#, Arduino, podatkovne baze, SQL, strežnik, Android.

Abstract

In this research paper, we were dealing with the lack of parking spaces in cities and time saving. Our goal was to make a parking house model that would satisfy customers as much as possible and therefore contributed to improving traffic situation in cities.

Every information system contains a well-built server, so this is the first part of our research task. We described the purpose of the servers and its components, and in the end, we assembled the server and customized it for the purpose of this task.

The next part is the model of the parking house in physical form. For this purpose, we chose the Arduino microcontroller. We explained what it is, presented the hardware part,

described the components used in the project and physically made the parking model and presented the working of the program.

Then we described the design of the database for this project.

Then we built an application for controlling the parking house with the programming language C #. We presented the flow chart of the program and presented the visual appearance.

Finally, we presented the application for the user and described its features.

Keywords

C#, Arduino, databases, SQL, server, Android.

Kratice

RFID: Radio Frequency IDentification.

ALE: aritmetična logična enota.

SDA - Serial Data Line.

MOSI - Master-Out Slave-In.

MISO - Master-In Slave-Out.

SCK - Serial Clock.

Vcc - Voltage Common Collector.

RAM – Random Access Memory.

DRAM – Dynamic Random-Access Memory.

SRAM – Static Random-Access Memory.

OS – operacijski sistem.

AREF – analog reference.

1 Uvod

1.1 Predstavitev raziskovalnega problema

Avtomobili so najbolj priljubljen način prevoza potnikov po vsej EU: predstavljajo približno 72 % vseh prevozov potnikov. V Ljubljani je v povprečju avtomobil neuporabljen 22 ur in 30 minut. Vsak dan se v Ljubljano pripelje med 120.000 in 140.000 vozil.

Vsi zgornji podatki nam povedo, da promet kroji veliko večino našega časa. Prevoz v šolo, službo, do sorodnikov, po opravkih … promet je na vsakem koraku. Tudi svetovna populacija narašča. Vse to vodi k več prometa, več zastojev, manj parkirnih prostorov.

Tudi področje samovozečih vozil se hitro razvija.

Ljudje veliko časa preživijo v šoli, službi itd. Ne glede na to dejstva, ima en dan še vedno 24 ur. Vendar se nam vedno zdi, da nam primanjkuje časa. Tisti, ki pa lahko ljudem ponudi čas, ima velik potencial, da kroji usodo večine ljudi.

Model avtomatizirane parkirne hiše se ubada s tem, da ljudem v čim večji meri poenostavi parkiranje in ljudje tako pridobijo čas, ki bi ga drugače zapravili z nepotrebnimi stvarmi.

1.2 Namen raziskovalne naloge

Namen raziskovalne naloge je izdelati model parkirne hiše z lastnostmi, ki bodo v čim večjem merilu poenostavile parkiranje.

Parkirne hiše se v zadnjih preteklih letih niso veliko spreminjale. Uporabnik se pripelje do parkirne hiše, kjer ga čaka napis, koliko parkirnih mest je še na voljo. Če ima srečo, je še vsaj eno mesto prosto, če pa te sreče nima, mora poiskati drugo parkirno hišo, s tem pa izgublja dragocen čas, ki bi ga lahko izkoristil na boljši način.

Nato pritisne na tipko, natisne se listek, ki vsebuje čas prihoda. Zapornica se odpre, stranka pa lahko parkira. Ko opravi z opravili, odide na blagajno, kjer poravna za uporabljene storitve.

Vse zgoraj naštete stvari bi se lahko poenostavile, stranka bi pridobila mnoge ugodnosti, podjetje, ki upravlja s to parkirno hišo pa bi lahko na podlagi plačanih storitev, ta denar vlagalo v nadaljnji razvoj podjetja.

1.3 Hipoteze

Za čim bolj natančne rezultate sva pred začetkom postavila nekaj hipotez, ki jih bova na koncu dokazala ali odvrgla.

1. Proces parkirnih hiš je mogoče poenostaviti.
2. Stranke bodo za storitev najraje plačale s pomočjo aplikacije.
3. Stroški modela ne bodo presegli 80 €.
4. V mestih vlada pomanjkanje parkirnih mest.

Prvo hipotezo bo mogoče potrditi v primeru, da bo model omogočal prihranek na času v primerjavi z večino ostalih parkirnih hiš.

Druga in četrta hipoteza sta odvisni od rezultatov raziskave.

Tretja hipoteza je odvisna od cene komponent, uporabljenih v projektu.

2 Teoretična zasnova

2.1 Lastnosti strežnika

Strežnik je računalnik, ki daje na voljo svoje storitve ostalim uporabnikom. Dostop do njega je lahko lokalen, tj. dostop je dovoljen samo računalnikom iz istega omrežja, ali globalen, tj. dostop do njega je dovoljen vsem računalnikom iz drugih omrežij.

Strežnik je v osnovi računalnik, ki pa se v lastnostih komponent razlikuje od namiznih ali prenosnih računalnikov.

Prva očitna razlika je velikost. Strežniki so veliko večji kot namizni računalniki, zasedejo celotno sobo, imajo ogromno potrošnjo glede električne energije in zaradi tega potrebujejo hlajenje z veliko ventilatorji, kar posledično pomeni, da so precej glasni.

Ker morajo zagotoviti hitro delovanje, imajo veliko procesorjev (na desetine), ki delujejo z visoko frekvenco in imajo veliko jeder in niti. Prav tako imajo veliko RAM pomnilnika, ki zagotavlja hiter dostop do podatkov.

Ker velikokrat shranjujejo ogromne količine podatkov, imajo diske velikosti nekaj Tera[1] bajtov.

Slika 1: Strežnik.

[1] 10^{12} količine podatkov.

2.2 Osnovna strojna oprema računalnika

2.2.1 Matična plošča

Matična plošča je glavni povezovalni del med komponentami. Služi kot povezava med komponentami računalnika, saj si morajo elementi računalnika deliti računalniške vire, prav tako pa morajo komunicirati med sabo in si pošiljati podatke.

Večina matičnih plošč že ima vgrajene različne vmesnike za priklop naprav:

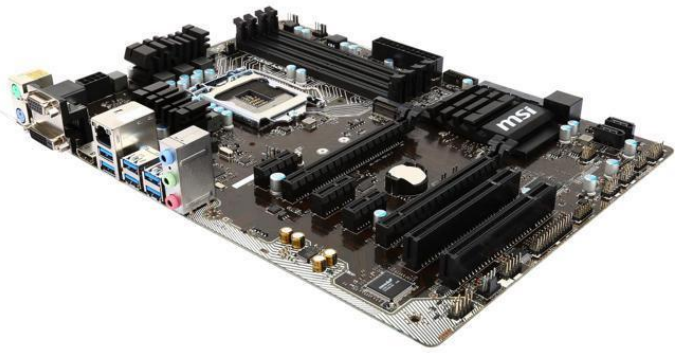
- Vmesnik USB
- Omrežni vmesnik
- Grafični vmesnik
- …

Lastnosti matične plošče so odvisne od dveh čipov, ki predstavljata sistemski nabor:

- Severni most – povezan z:
    - Procesorjem
    - RAM pomnilnikom
    - Grafičnim vmesnikov
    - Južnim mostom
- Južni most – povezan z:
    - Integriranim grafičnim vmesnikom
    - Razširitvenimi režami
    - Vmesniki: USB, miška, Ethernet …
    - Trdim diskom

Sistemski nabor je potreben za komunikacijo med komponentami, npr. procesor s pomnilnikom, pomnilnik z vmesnikom USB itd.

Prav tako sistemski nabor določa osnovne lastnosti matične plošče in katere komponente lahko uporabimo v računalniku (vrsto in količino pomnilnika, katere procesorje podpira itd.).



Slika 2: Matična plošča.

2.2.2 Procesor

CPE oziroma centralna procesna enota skrbi za pravilno delovanje sistema. Izvaja logične in aritmetične operacije, komunicira z ostalimi napravami, obdeluje podatke.

Sestavljen je iz predpomnilnika, aritmetične logične enote, registrov in kontrolne enote.

Današnji pomnilniki delujejo s 3 GHz ali še celo več, kar je veliko več kot hitrost delovanja RAM pomnilnika. Procesor v RAM pomnilnik shranjuje ali pridobiva podatke, potrebne za delovanje sistema.

Predstavljajmo si naslednji primer: procesor potrebuje podatke o določenem procesu, ki so v pomnilniku. Iz pomnilnika pridobi podatke, ker pa je teh podatkov veliko in je hitrost delovanja RAM pomnilnika počasnejša kot hitrost delovanja CPE, mora procesor čakati na podatke, kar je seveda nezaželeno, ker takrat procesor ne počne nič, posledično pa mora tudi uporabnik čakati dlje, da se določeno opravilo izvede.

Za ta namen so proizvajalci ustvarili predpomnilnike (do tri). Procesor ima vgrajen majhen pomnilnik (nekaj MB), ki pa ima izredno hitro delovanje, prav tako pa je dostopni čas do podatkov v predpomnilniku majhen.

Predpomnilnik je po zgradbi narejen drugače kot RAM, kar ga naredi hitrejšega, a tudi dražjega. V predpomnilnik si procesor shranjuje podatke, za katere meni, da jih bo v naslednjih ukazih potreboval.

Aritmetična logična enota je samostojna enota, ki izvaja vse aritmetične in logične operacije. Današnji procesorji večinoma vsebujejo več aritmetičnih enot. Osnova za njihovo delovanje so logična vrata, ki na podlagi vhodnih podatkov in pravilnostnih tabel na izhodu posredujejo rezultate dela.

Register je priročen pomnilnik s hitrim dostopom. V procesorju poznamo več registrov:

- Akumulator – shranjevanje operandov (npr. 5)
- Programski števec – shramba pomnilniškega naslova naslednjega ukaza
- Indeksni števec – pomoč za določanje naslova operanda
- Statusni register – opis rezultata zadnjega izvedenega ukaza
- Ukazni register – shramba ukaza

Slika 3: Procesor.

2.2.3 RAM pomnilnik

Bralno-pisalni pomnilnik je vrsta elektronskega pomnilnika, ki se uporablja za tekoče delo z računalnikom. Procesor lahko naslovi poljubno celico in vanj shrani podatek ali pa ga pridobi. Ob izklopu računalnika se podatki pozabijo.

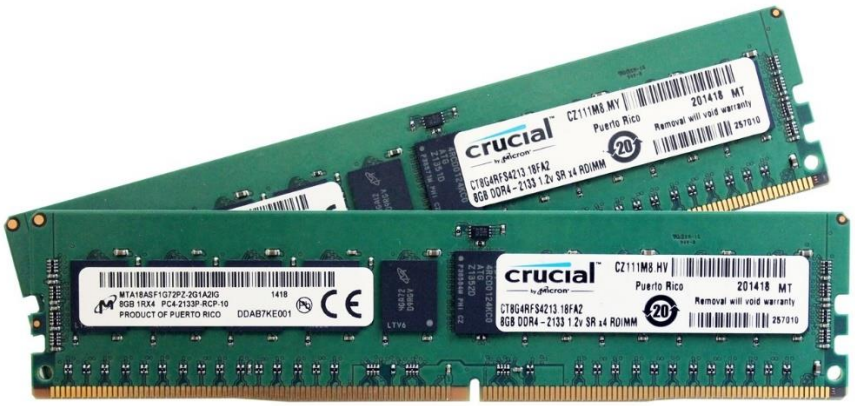
Glede na delovanje poznamo dve vrsti RAM: dinamični (DRAM) in statični (SRAM) pomnilnik. Opis lastnosti je viden v tabeli 1.

Tabela 1: Primerjava DRAM in SRAM pomnilnika.

	DRAM	**SRAM**
Uporaba	Delovni pomnilnik	Predpomnilnik[2]
Cena	Nižja	Višja
Kapaciteta	Velika	Majhna
Hitrost	Manjša	Visoka

[2] Glej poglavje o procesorju.

Pomnilna celica	Kondenzator	Posebno vezje
Posebnost	Potrebuje osveževanje[3]	Ne potrebuje osveževanja



Slika 4: RAM pomnilnik.

2.2.4 Zunanje pomnilne naprave

Zunanje pomnilne naprave so bile razvite za trajno hranjenje podatkov, saj jih RAM pomnilnik po izklopu pozabi. Dandanes se najpogosteje uporablja disk, ki pride v dveh izvedbah: HDD ali SSD.

Disk HDD za delovanje uporablja magnetenje magnetnega materiala. Sestavljen je iz kovinskih plošč z magnetno prevleko in bralno-pisalne glave. Glava se vrti znotraj ohišja s 5400, 7200 ali več obrati na minuto.

Prednosti tega diska so predvsem velikost (TB) in cena, saj so veliko cenejši od SDD diskov.

A imajo kar nekaj slabosti. Prva takšna bi bila segrevanje. Ker se glava vrti, to povzroča segrevanje, ki je lahko pri velikih hitrostih težavno. Druga težava je hitrost, saj so veliko počasnejši kot SDD diski. Prav tako pri delovanju oddajajo vibracije, so manj odporni proti udarcem, manj energetsko učinkoviti, prav tako obstaja nevarnost uničenja podatkov zaradi magnetizma.

SSD diski izboljšujejo težave diskov HDD, a prinašajo nekaj drugih težav. So zelo hitri in odzivni, neslišni, saj uporabljajo pomnilna integrirana vezja, neobčutljivi na udarce, se manj segrevajo …

[3] Vsebina se prebere in ponovno zapiše.

V primerjavi s HDD so dražji, a cena pada, prav tako se njihov nakup obrestuje pri hitrejšem delovanju računalnika.

Slika 5: HDD in SSD disk.

2.3 Operacijski sistem

Operacijski sistem je programska oprema, ki nadzoruje delovanje celotnega sistema. Deluje kot vmesnik med programsko in strojno opremo, je odgovoren za usklajevanje procesov, opravil, dodeljuje računalniške vire procesom, rešuje konflikte (istočasni dostop do virov) … Uporabnik za delo z operacijskim sistemom uporablja grafični vmesnik ali pa ukazni vmesnik, t. i. terminal.

Ubuntu Mate je odprtokodni operacijski sistem, ki temelji na jedru operacijskega sistema Linux. Linux je prav tako odprtokoden, kar pomeni, da lahko vsak dobi izvorno kodo in si ga prilagodi za lasten namen. Ker pa je to zahtevno, trajajoče itd., se programerji združijo in vsak naredi določeno opravilo za pravilno delovanje sistema in take nastane distribucija operacijskega sistema.

Prednosti:

- Odprtokoden OS – vsakdo si ga lahko prilagodi po željah
- Varnost – bolj odporen proti napadom
- Boljše delovanje s starejšimi komponentami
- Zastonj
- Zasebnost – Linux ne zbira (oz. minimalno število) osebnih podatkov

- Prilagoditev uporabniškega vmesnika – npr. GNOME, KDE
- Stabilnost

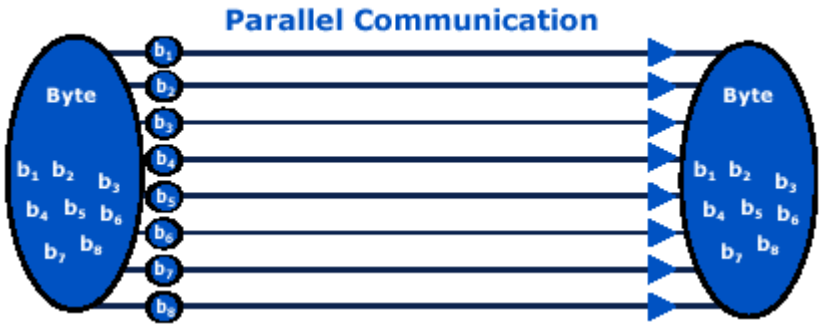
Zaradi navedenih prednosti večina podjetij za svoje strežnike uporablja distribucijo operacijskega sistema Linux.

2.4 Komunikacijski protokoli

V vsakdanjem življenju, službi ali na drugih področjih so vsepovsod prisotni računalniški sistemi. Prej ali slej morajo ti sistemi komunicirati med sabo in si deliti podatke. Da je ta operacija uspešna, morajo računalniški sistemi imeti skupen komunikacijski protokol. Ti so lahko dveh vrst: vzporedni ali serijski.

2.4.1 Vzporedna komunikacija

Vzporedna komunikacija prenese več bitov naenkrat. Podatki se pošiljajo po vodilih, najpogosteje po 8, 16 ali več kanalih. Prednost te komunikacije je hitrost in enostavna gradnja. Pomanjkljivost predstavlja več vhodno/izhodnih linij in neuporabnost za večje razdalje zaradi večjih dolžin kablov (cena) in problemov s sinhronizacijo. To je zelo pomembno pri sistemih, kjer smo omejeni s strojno opremo naprave. Delovanje vzporedne komunikacije je predstavljeno na sliki 6.



Slika 6: Vzporedna komunikacija.

2.4.2 Serijska komunikacija

Serijska komunikacija prenaša en bit na enkrat. Ta vmesnik lahko deluje samo z eno žico. Uporablja se za daljše razdalje in v primerih, kadar je težko zagotoviti sinhronizacijo.

Prednosti serijske komunikacije so, da zahteva manj žic, kar pomeni nižjo ceno vmesnika in zavzame manj prostora.

Še ena dodatna prednost: motnje med žicami niso mogoče.

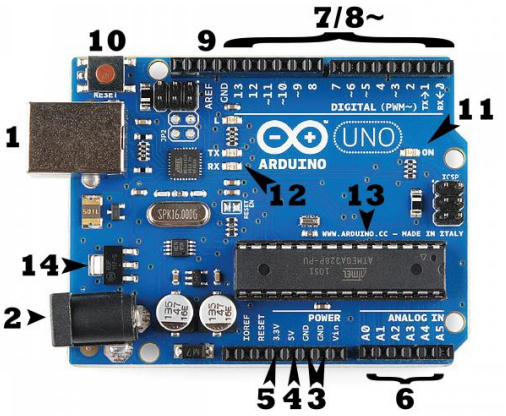
Slika 7: Serijska komunikacija.

Slika 7 prikazuje delovanje serijske komunikacije.

2.5 Mikrokrmilnik Arduino Uno

Arduino je odprtokodno razvojno okolje za gradnjo elektronskih projektov. Arduino plošča lahko bere iz vhodov (npr. gumbi, senzorji …) in na podlagi teh signalov in nameščene kode, izvede določen proces. Lahko prižge ali ugasne LED-diode, pošlje SMS, poišče navodila za pot na internetu, začne predvajati določeno skladbo iz SD kartice ali pa preko serijskega vmesnika pošlje določen niz znakov na računalnik.

Arduino UNO uporablja 8-bitni ATmega328P mikrokontroler. Ustrezna vhodna napetost za delovanje je med 7 in 12 voltov. Maksimalna vhodna napetost je 20 voltov. Deluje z 5 volti, kolikor lahko prenaša vmesnik USB. Ima 14 digitalnih vhodov (t. i. pini). Hitrost mikroprocesorja je 16 MHz. Nazadnje, ima 2 KB RAM pomnilnika.



Slika 8: Arduino UNO.

Vhod za napetost (1, 2) (USB / napajalni priključek)

Vsaka elektronska naprava potrebuje napajanje. Arduino UNO lahko napajamo iz računalnika preko vmesnika USB ali preko napajalnega priključka.

Pini (3 - 9) (5V, 3.3V, GND, analogni, digitalni, PWM, AREF)

Pini se uporabljajo za povezavo z drugimi elektronskimi komponentami. Večinoma so vse komponente na testni ploščici. Vsak pin ima svojo funkcijo.

- GND (3) – ozemljitev.
- 5V (4) in 3.3V (5): Prvi pin oskrbuje Arduina s 5V, drugi pa s 3V. Katerega uporabimo, je odvisno od porabe električnega toka naših komponent.
- Analogni pini (6): uporabljajo se za branje signalov iz analognih senzorjev (npr. senzor za temperaturo).
- Digitali pini (7): pini od 0 do 13 se uporabljajo za digitalni vhod ali izhod.
- PWM (8): pulzno širinska modulacija; uporabljamo jih za simuliranje analognih izhodov (npr. počasno prižiganje in ugašanje LED-diode).
- AREF (9): primerjava napetosti.
- Gumb za resetiranje (10): ob pritisku se ta pin začasno poveže na zemljo in nameščen program se ponovno naloži.
- Kazalnik napajanja (11): LED-dioda je prižgana, kadar ima Arduino napajanje.
- LED-diode za oddajanjein sprejemanje (12): TX (transmit - oddajanje) dioda utripa, ko Arduino oddaja podatke, RX (receive - sprejem) dioda utripa, ko Arduino sprejema podatke.
- Intergirano vezje (13): množica elektronskih elementov, ki skupaj sestavljajo čip za računanje in obdelovanje podatkov.
- Regulator napetosti (14): nadzira količino napetosti, ki prihaja v Arduino. Blokira odvečno napetost in tako prepreči okvaro.

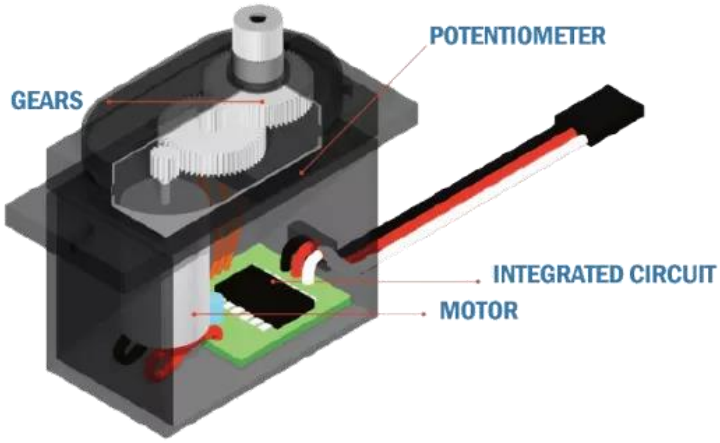
Vsaka komponenta je vidna na sliki 8.

2.6 Servo motor

Servo motor je komponenta, ki se uporablja za spreminjanje položaja gredi. Na podlagi potenciometra lahko nadziramo položaj in hitrost vrtenja.

Sestavljen je iz štirih delov:

- Motor: upravlja z menjalnikom; ima veliko hitrost, a nizek navor.
- Menjalnik: povezovalnik med motorjem in gredjo; ima visok navor.
- Potenciometer: nadzor hitrost in položaja.
- Nadzorno vezje: nadzor delovanja.

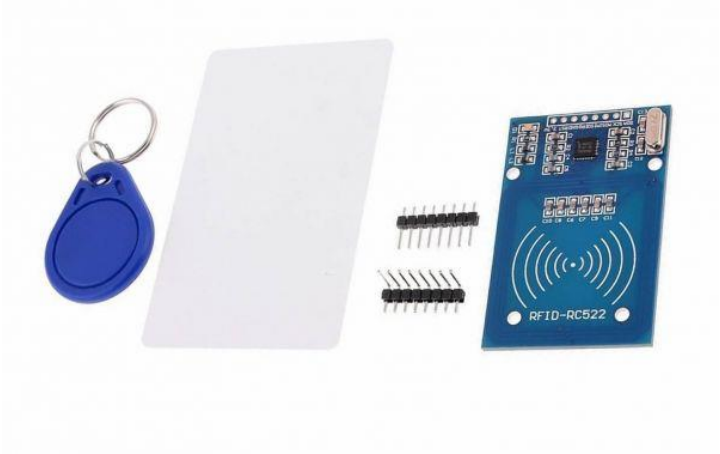


Slika 9: Servo motor.

2.7 RFID bralnik

RFID ali radiofrekvenčna identifikacija je tehnologija prenosa podatkov med čitalnikom in čipom. Čitalnik lahko prebere podatke kartice z nekaj centimetrov.

Sistem RFID je zgrajen iz dveh delov: čipa (kartica, poseben ključek) in čitalnika. Kartica je opremljena z anteno, ki lahko oddaja ali sprejema podatke. Čitalnik vsebuje dve komponenti: mikročip, ki uravnava delovanje sistema, in anteno, ki sprejema in oddaja radio signale.



Slika 10: RFID bralnik.

Antena na čitalniku oddaja radijske signale. Ko določena kartica pride do določene razdalje, ta signal zazna in odgovori s svojo številko.

Čitalnik nato pošlje številko naprej do računalnika, kjer se na podlagi programa, izvede določeno dejanje (npr. odprejo se vrata).

2.8 Ultrazvočni senzor

Ultrazvočni senzor se uporablja za ugotavljanje razdalje do določenega objekta. Senzor oddaja ultrazvočne signale pri 40 000 KHz. Ko ti signali pri potovanju po zraku, naletijo na oviro in se odbijejo nazaj. Na podlagi časa potovanje in ob pomoči formule se lahko enostavno izračuna razdalja do objekta.

Slika 11: Ultrazvočni senzor.

Oddajni del senzorja pošlje ultrazvočne signale, sprejemni del pa sprejme odmev.

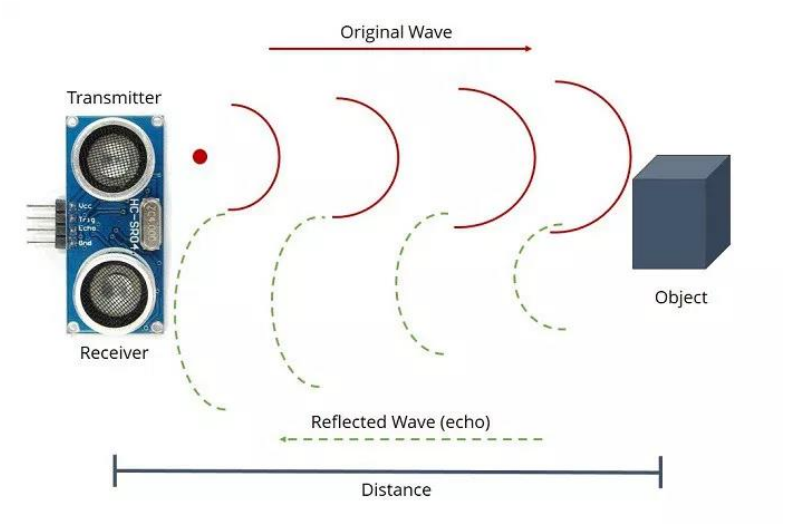
Izračun razdalje:

$$L = \frac{1}{2} * T * C$$

L = razdalja,

T = čas od oddaje signala do sprejetja odmeva,

C = hitrost zvoka v mediju.



Slika 12: Delovanje ultrazvočnega senzorja.

2.9 Senzor dotika

Senzor dotika, kot že samo ime pove, zaznava dotik. Je steber današnjih pametnih telefonov oz. vseh naprav na dotik.

Kondenzator je elektrotehniški element, ki shranjuje energijo v obliki električnega polja. Zgrajen je iz dveh prevodnikov, ki ju ločuje izolator.

Senzor dotika deluje na principu kondenzatorja. Ko se s prevodnim elementom (npr. prstom) dotaknemo zgornje plasti senzorja, se kapacitivnost poveča, kar zazna elektronika, vezana na senzor.

Slika 13: Senzor dotika.

2.10 Objektno orientirano programiranje

Programske jezike, pri katerih osnovni koncept predstavlja objekt, imenujemo objektni jeziki.

Glavna ideja objektno orientiranega programiranja je v tem, da program predstavlja celoto, sestavljeno iz številnih samostojnih enot oziroma objektov. Vsak objekt se lahko preko sporočil sporazumeva z okoljem, procesira podatke in oddaja sporočila drugim objektom.

Osnovan je na objektih in relacijah med njimi. Če ga primerjamo s klasičnim, proceduralnim programiranjem, vnaša vanj nove koncepte in termine, kot so: objekt, razred, vmesnik, dedovanje …

Glavna ideja se je pojavila že leta 1967 z nastankom programskega jezika Simula-67, ki sta ga razvila Ole-Johan Dahl in Kristed Nygaard z Norveškega računalniškega centra v Oslu. V sklopu projekta so analizirali učinek različnih atributov na povezavi med ladjami. Pri tem sta različne tipe ladij razvrstila v posamezne razrede ter vsakemu razredu določila lastnosti in postopke. To so bili prvi koncepti dedovanja, razredov in objektov.

Objektni jeziki so res zaživeli šele z nastankom objektno usmerjenega jezika C++. Bjarne Stroustrup je leta 1979 z vpeljavo razredov v jezik »C with Classes« nadgradil programski jezik C. Jezik je bil prvič uradno implementiran konec leta 1983, medtem ko je pravi razmah doživel po letu 1985. V devetdesetih letih je bil OOP prevladujoč vzorec programiranja in C++ najbolj množično uporabljan objektno usmerjeni jezik.

Lastnosti objektno orientiranih jezikov so bile dodane tudi drugim že obstoječim jezikom Ada, BASIC, Lisp, Fortran, Pascal itd.

Objekte opredeljujeta obnašanje in stanje objekta. Obnašanje nam pove, kaj objekt »zna«. Programsko obnašanje je izvedeno s programsko kodo oziroma podprogrami ali metodami.

Objekt (kot primer razreda) lahko razumemo kot stvar, ki izvaja določen nabor dejavnosti na primer, objekt stranka nam lahko pove ime, priimek, stanje na računu …

```
public class Stranka
{
    8 references
    public string EMSO { get; private set; }
    3 references
    public string Ime { get; private set; }
    3 references
    public string Priimek { get; private set; }
    3 references
    public string Email { get; private set; }
    2 references
    public string Kljuc { get; private set; }
    5 references
    public decimal Stanje { get; private set; }
    private static readonly string povezavaString = ConfigurationManager.ConnectionStrings["povezava"].ConnectionString;
```

Slika 14: Podatki objekta.

```
private void Racun(decimal stanje)
{
    Stanje -= stanje;
    try
    {
        using (MySqlConnection povezava = new MySqlConnection(povezavaString))
        using (MySqlCommand cm = new MySqlCommand("UPDATE stranka SET stanje = @stanje WHERE EMSO = @EMSO", povezava))
        {
            cm.Parameters.AddWithValue("@stanje", Stanje);
            cm.Parameters.AddWithValue("@EMSO", EMSO);
            povezava.Open();
            cm.ExecuteScalar();
        }
    }
    catch (Exception e)
    {
        MessageBox.Show("Napaka: " + e.ToString(), "Prišlo je do napake", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Slika 15: Metoda objekta.

2.11 Vmesnik IDisposable v programskem jeziku C#

Vmesnik IDisposable služi za sprostitev neuporabljenih računalniških virov. Ko se določen odsek kode konča, ta vmesnik počisti neuporabljene vire. To nam posebej pride v pomoč, kadar imamo odprto povezavo na podatkovno bazo. Če je povezava vedno odprta, to pomeni, da zaseda določen odstotek virov strežnika, kar je seveda nezaželeno, saj lahko pride do preobremenitve strežnika.

Druga težava, ki se pojavi pri tem je, da ostale naprave dobijo manj virov in tako posledično traja dlje, da pridobijo podatke, kar ima za rezultat počasnejše delovanje določene aplikacije.

Vmesnik IDisposable se v programskem jeziku C# uporablja s t. i. stavki using, kar je vidno na sliki 19. Prav tako je primerno uporabljati try-catch stavke, saj ti ulovijo napake in preprečijo sesutje programa.

```
private void Posodobi()
{
    try
    {
        using (MySqlConnection povezava = new MySqlConnection(povezavaString))
        using (MySqlCommand cm = new MySqlCommand("DELETE FROM parkiranje WHERE EMSO = @EMSO", povezava))
        {
            cm.Parameters.AddWithValue("@EMSO", EMSO);
            povezava.Open();
            cm.ExecuteScalar();
        }
    }
    catch (Exception e)
    {
        MessageBox.Show("Napaka: " + e.Message + "\nVir: " + e.TargetSite, e.GetType().ToString());
    }
}
```

Slika 16: Del kode C# programa.

2.12 Operacijski sistem Android

Android je operacijski sistem za pametne telefone, ter ostale prenosne naprave, ki je zgrajen Linuxovemu jedru. Za razvoj Androida je pomemben Google, saj je to hitro rastoče podjetje prevzel pod svoje okrilje leta 2007.

Po večini so aplikacije za Android napisane v programskem jeziku Java. Aplikacije so sestavljene v Android paket s končnico »apk«. Vsaka aplikacija se požene v svojem procesu. Operacijski sistem požene proces takrat, ko mu je poslana zahteva za izvajanje aplikacije. Ko aplikacija konča z izvajanjem, se proces zapre. To omogoča rabo pomnilnika tudi drugim aplikacijam. Vsak posamezni proces se prevede posebej, kar omogoči izoliranje aplikacij, da delujejo med sabo neodvisno. Vsaki aplikaciji se ob zagonu ustvari tudi lastna identifikacijska koda, ki se ji nato dodajo pravice za uporabo strojne opreme.

Prednosti:

- Večinoma so programi za uporabnike brezplačni
- Je enostaven, odziven in omogoča večopravilnost
- Mogoča je samodejna sinhronizacija z Googlovimi storitvami

Slabosti:

- Nekatera okužena programska oprema na portalu Google Play

- Na trgu najdemo izjemno bogato paleto telefon z OS Android in nudenje tehnične pomoči za posamezen model je veliko težje kot v primeru, če imamo na voljo le Microsoftove ali Applove mobilne telefone.
- Začetna nastavitev je v primerjavi s preostalimi konkurenčnimi mobilnimi operacijskimi sistemi precej zapletena.

2.13 Android Studio

Android studio je uradno razvojno okolje za Googlov operacijski sistem Android, ki temelji na JetBrains' IntelliJ IDEA programski opremi, ki je bila ustvarjena točno za razvoj Androida. Na voljo je na več različnih platformah (Windows, IOS, Linux).

2.13.1 Programski jeziki v Android Studiu

Javo je razvil James Gosling s sodelavci v podjetju Sun Microsystems. Projekt, ki se je na začetku leta 1991 imenoval Oak (hrast), je bil razvit kot zamenjava za C++. Je objektno usmerjen programski jezik, ki ga ne smemo zamenjevati z jezikom JavaScript, ki ima podobno ime.

Je moderen programski jezik, ki se je v zadnjem desetletju zelo razširil v svetu računalništva. Java je opremljena z bogato standardno knjižnico programskih struktur in funkcij (za delo z datotekami, za dostop do podatkovnih baz, za mrežne povezave, hkratno izvajanje več programov, grafične aplikacije, aplikacije, vgrajene v brskalnike, aplikacije za telefone …).

Poznamo 3 vrste jave:

- J2SE - standardna različica jave za osebne računalnike
- J2ME - različica jave za mini naprave (mobiteli, pametni televizorji ...)
- J2EE - poslovna različica jave

KOTLIN je programski, ki ga je zasnoval JetBrains. Prvič se je pojavil leta 2011, v Android Studiu pa se je pojavil od verzije 3.0 naprej (2017). Podpira več platform, je statično napisan (preverja pravilnost izvorne kode), objektni programski jezik, ki podpira tudi standardni java prevajalnik.

3 Empirični del

3.1 Sestava računalnika

Večino komponent je eden izmed avtorjev te raziskave (David Slatinek) dobil od prijatelja, določene pa sva kupila.

Začela sva z izbiro ohišja računalnika. Doma sva imela dve različni ohišji, ampak glede na faktor oblike matične plošče, je bil samo eden ustrezen.

Naslednja stvar, za katero sva poskrbela, je bila matična plošča. Teh sva imela kar nekaj, na koncu sva se zaradi združljivosti z drugimi komponentami, odločila za matično ploščo podjetja Intel, model D915GLVG. Matično ploščo sva pričvrstila v ohišje računalnika.

Nato sva začela z izbiro procesorja. Na podlagi podnožja matične plošče (LGA 775), sva izbrala 32 bitni Pentium 4 530J, ki ima frekvenco 3 GHz in tehnologijo gradnje 90 nm. Preden sva procesor vstavila v podnožje, sva nanj še nanesla termalno pasto, saj bo ta pomagala pri odvodu toplote.

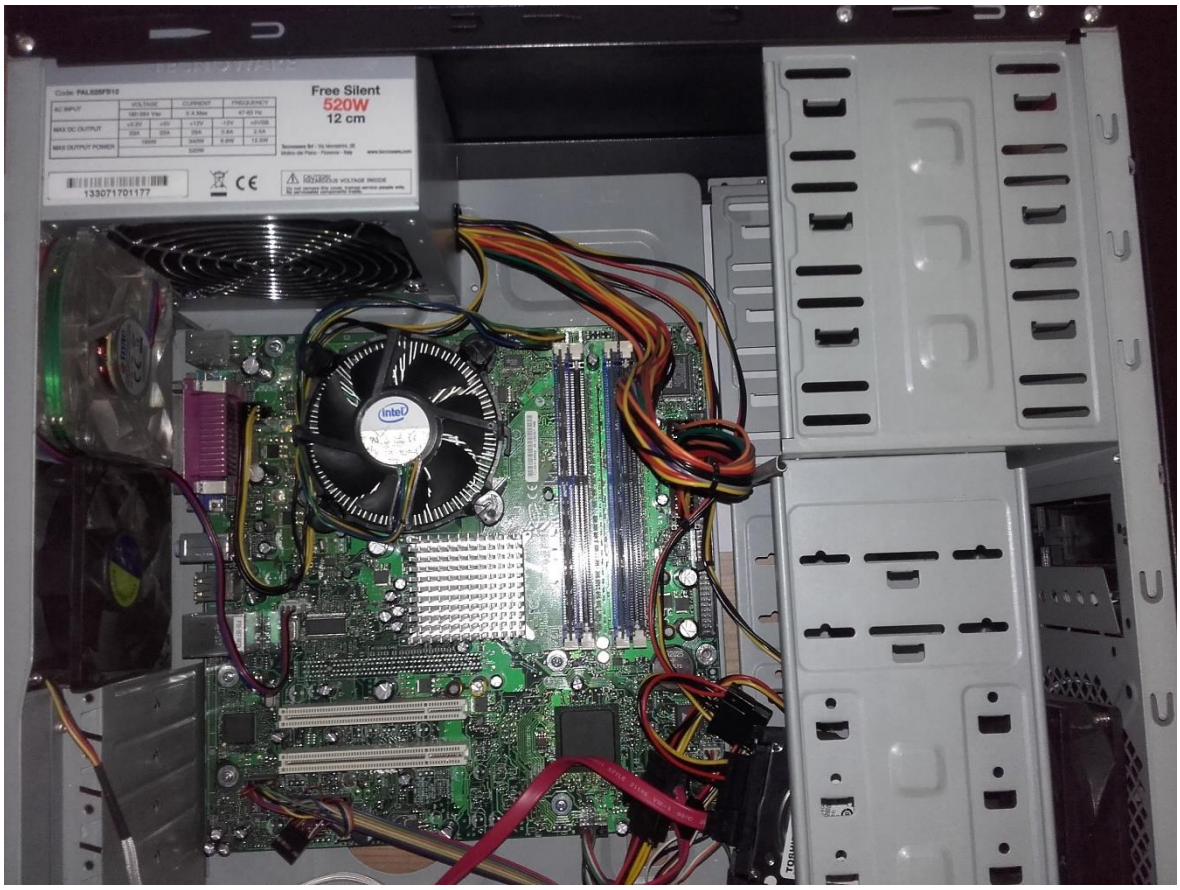
Zatem sva v pomnilniške reže vstavila RAM pomnilnike, ki so v najinem primeru vrste DDR, njihova hitrost pa znaša 400 MHz.

Nato sva v ohišje dodala napajalnik Tecnoware Free Silent 520W in priključila glavni napajalni priključek na matično ploščo.

Dodala sva še trdi disk modela MQ01ABF050, velikosti 500 GB, 5400 rpm[4] in ga ustrezno priključila na matično ploščo preko SATA vodila in mu priskrbela napajanje.

Nazadnje sva poskrbela za ustrezno hlajenje računalniškega sistema in vgradila ventilatorje in jim zagotovila ustrezno napajanje ter povezala sprednji del ohišja na matično ploščo.

[4] Revolutions per minute – število obratov na minuto.



Slika 17: Sestavljeni računalnik.

3.2 Nameščanje operacijskega sistema in prilagoditev računalnika

V kontrastu z izkušenimi skrbniki sistemov, ki ne uporabljajo grafičnega vmesnika, ampak samo ukazni vmesnik (npr. Ubuntu Server), sva se odločila, da bova izbrala operacijski sistem Ubuntu Mate, saj je ustrezen za začetnike, prav tako pa grafični vmesnik ni pretirano zahteven za komponente računalnika.

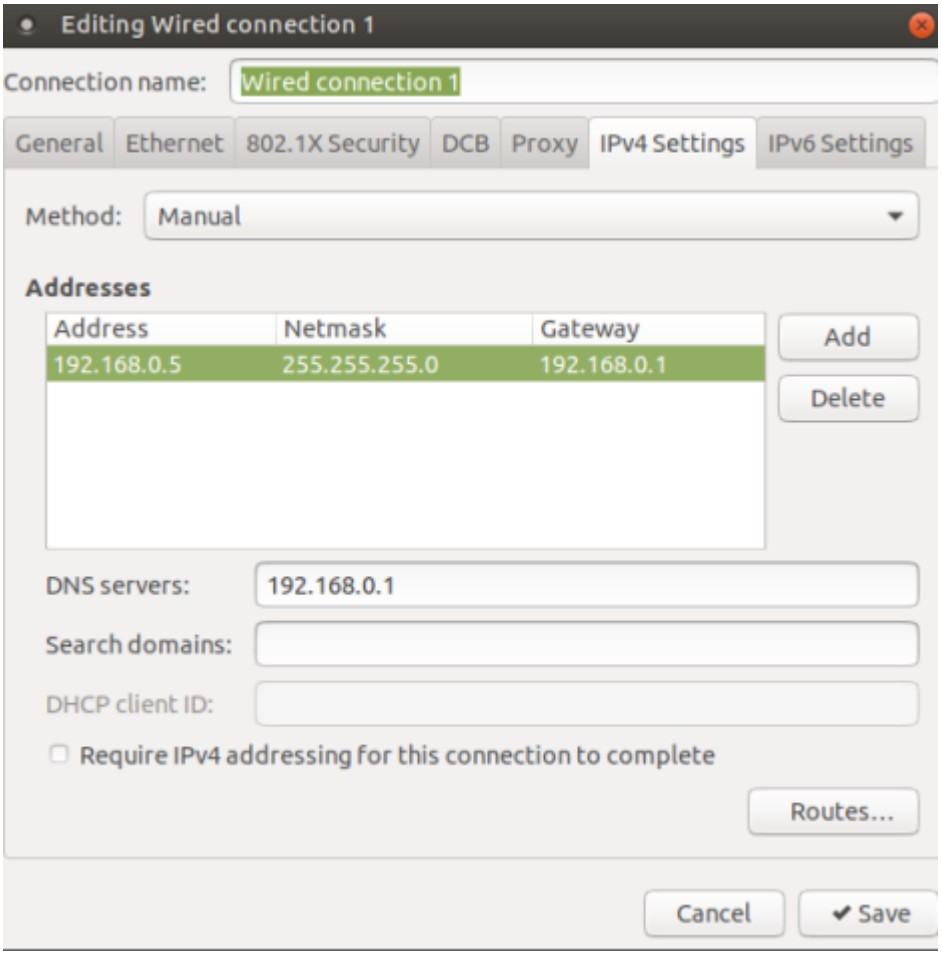
Najprej sva z uradne spletne strani prenesla Ubuntu Mate. Nato sva še prenesla orodje Rufus, s katerim sva nato ustvarila zagonski ključek.

Nato sva namestila operacijski sistem in po namestitvi sva morala nastaviti statičen IP[5] strežnika, saj bo samo tako usmerjevalnik vedel, komu poslati paket z zahtevo, ko bo ta prišel na določena vrata[6].

[5] Naslov, s katerim se predstavlja računalnik, na primer: 195.45.6.2.
[6] Vmesnik za sprejem in oddajo podatkov.

Omrežne nastavitve se nastavijo pod »Control Center«, natančneje pod »Internet and Network«.

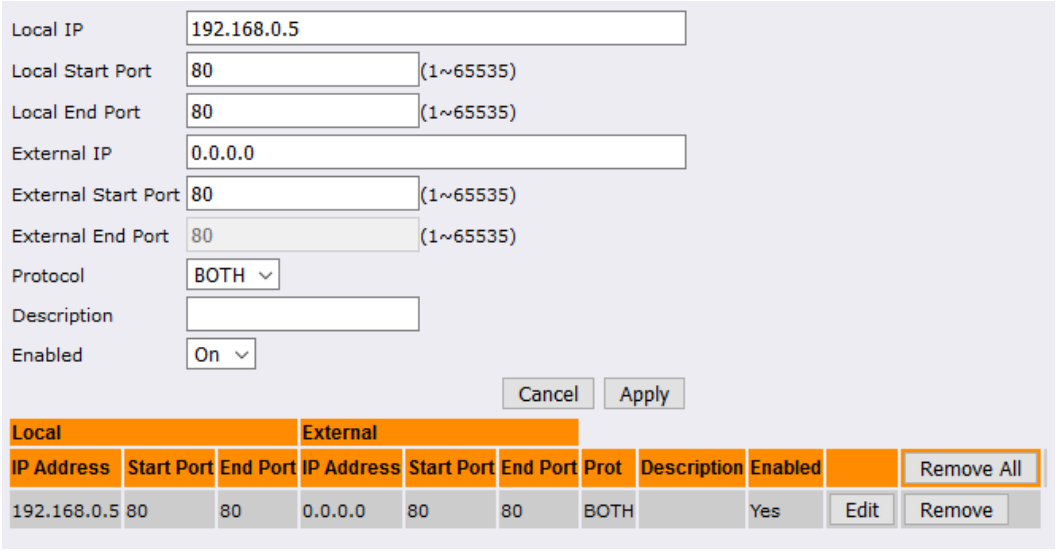
Za IP naslov sva izbrala 192.168.0.5 (izven domene dinamičnih naslovov, ki jih dodeljuje usmerjevalnik), za omrežno masko sva pustila privzeto, prehod[7] in DNS[8] strežnik pa naslov usmerjevalnika.



Slika 18: Nastavljanje statičnega IP naslova..

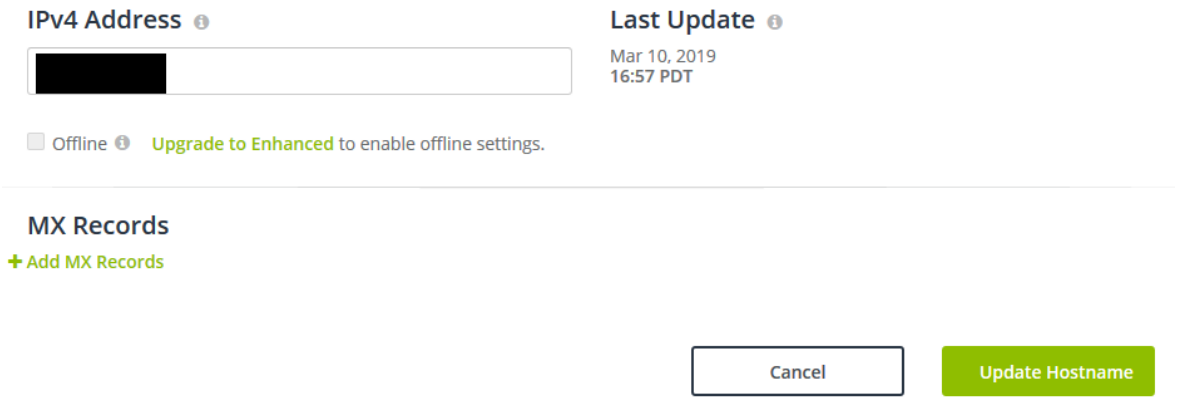
Nato sva uredila port forwarding. Ta funkcija v nastavitvah usmerjevalnika nam omogoča, da usmerjevalnik posreduje paket do končne postaje, ko prejme paket na določena vrata. V najinem primeru, če bo prejel paket na vrata 80, ga bo posredoval strežniku.

[7] Vmesnik med različnimi omrežji.
[8] Servis, ki skrbi za povezavo med domeno in IP naslovom.

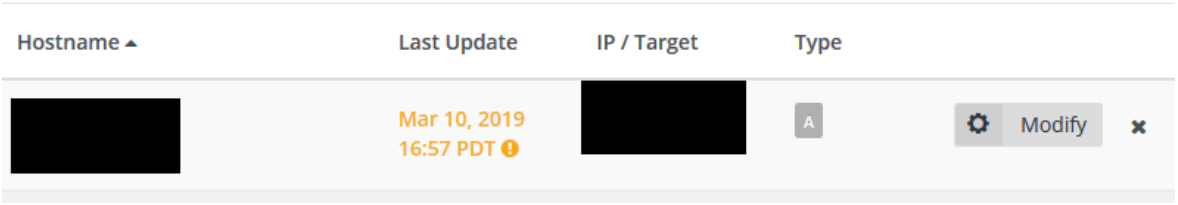


Slika 19: Port forwarding.

Nato sva z uporabo storite No-IP omogočila povezavo na strežnik z uporabo domene (npr. program.ddns.net). To pride tudi v pomoč, saj imava oba statičen naslov omrežja, kar pomeni, da v aplikacije ne bi mogla preprosti napisati IP naslov omrežja, saj se ta spreminja. Ta storitev pa spremlja spremembo IP naslova in to posreduje svojemu strežniku.



Slika 20: Nastavitev domene.



Slika 21: Podatki o domeni.

Določeni podatki so prekriti zaradi varnostnih razlogov.

Nazadnje sva namestila Apache spletni strežnik za upravljanje s podatkovno bazo.

```
sudo apt install apache2
```

Slika 22. Namestitev Apache strežnika preko terminala.

```
sudo apt install mysql-server
```

Slika 23: Namestitev servisa Mysql preko terminala.

3.3 Načrtovanje podatkovne baze

Podatkovna baza je organizirana zbirka podatkov. Znotraj podatkovne baze imamo tabele, ki so večinoma povezane druga z drugo. Znotraj tabel imamo posamezne stolpce, ki predstavljajo atribute za določeno tabelo.

Vse podatke lahko pregledujemo, posodabljamo, vstavljamo, brišemo (pod pogojem, da imamo ustrezne pravice).

Načrtovanja sva se lotila s postopkom od zgoraj navzdol – velik problem razdelimo na manjše dele in to ponavljamo, dokler ne dobimo rešljivega dela celote.

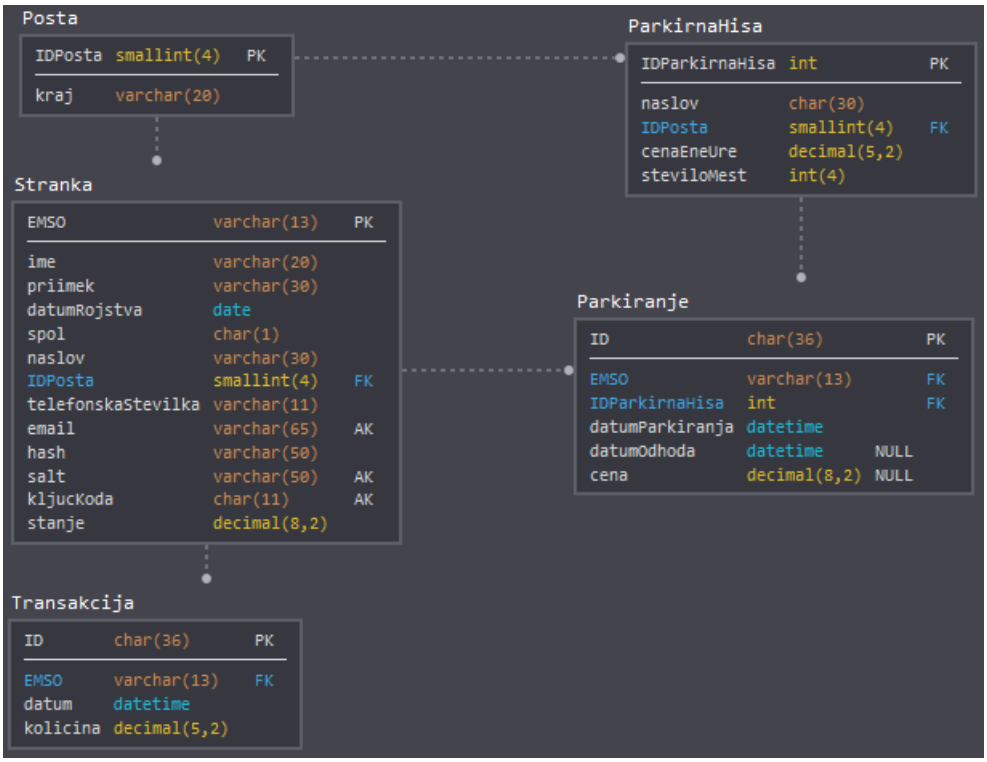
Najprej sva ustvarila tabelo Stranka, saj je ta zagotovo najpomembnejša. V tabelo sva dodala atribute, ki so potrebni za ustrezno delovanje, hkrati pa sva pazila, da sva izbrala ustrezne podatkovne tipe, saj je to velikega pomena pri ohranjanju prostora na trdem disku.

Nato sva dodala tabeli Pošta in Parkirna hiša. Tabela Parkirna hiša bo vsebovala vse potrebne atribute za nemoteno delovanje programa, ki jih bo računalnik pridobil med delovanjem, med drugim tudi ceno za uro in število mest v parkirni hiši.

Zatem sva ustvarila tabelo Parkiranje. Tabela Parkiranje se bo uporabljala za tekoče delovanje programa. V njej se bodo shranjevali podatki o vstopu in izstopu, stranki, ki je parkirala in ceno za parkiranje. Pri tem pa sva naletela na določeno težavo. Odločiti sva se morala ali naj podatkovno bazo narediva tako, da lahko stranka naenkrat parkira samo v eni parkirni hiši ali v več. Po temeljitem razmisleku sva se odločila za prvo možnost, da

lahko stranka naenkrat parkira samo v eni parkirni hiši, saj, statistično gledano, ni ravno verjetno, da bo stranka parkirala, nato pa z drugim vozilom parkirala na drugi lokaciji.

Nazadnje sva še dodala tabelo Transakcija. Ta bo v pomoč, kadar bo hotel uporabnik naložiti denar na svoj račun.



Slika 6: Načrt podatkovne baze.

3.4 Sestava modela in izdelava Arduino programa

Na Arduino bo povezan servo motor, ki bo skrbel za dvig in spust zapornice.

Naslednji senzor bo čitalnik kartic RFID. Ta bo skrbel za branje kode RFID s kartice stranke.

Naslednja dva senzorja sta: senzor dotika in senzor razdalje. Senzor dotika bo pri vhodu v parkirno hišo in bo skrbel, da bo obvestil Arduina, kdaj je stranka zapustila parkirno hišo. Tako bo Arduino vedel, kdaj lahko zapre zapornico. Podobno nalogo ima senzor razdalje. Ta bo obvestil Arduina, kdaj je stranka zapeljala v parkirno hišo, nato pa lahko Arduino oz. servo motor zapre zapornico.

Najprej sva iz Arduina zagotovila napajanje in ozemljitev na testni ploščici, saj bi drugače imel premalo pinov.

Nato sva na testno ploščico povezala servo motor, ki ima tri povezave. Ena za napajanje (3.3V), ena je za ozemljitev, zadnja je za nadzor samega motorja. Zadnja žica mora biti povezana na pin PWM, izbrala sva pin številka 2 na Arduinu.

Naslednja komponenta, ki sva jo povezala, je RFID čitalnik, ki ima kar nekaj povezav.

- Pin RST: služi za ponastavitev naprave
- Pin SDA: prenos podatkov
- Pin MOSI: nadrejena naprava sporoča, podrejena posluša
- Pin MISO: nadrejena naprava posluša, podrejena sporoča
- Pin SCK: sinhronizacija ure

Nato sva povezala senzor razdalje. Ta ima štiri priključke.

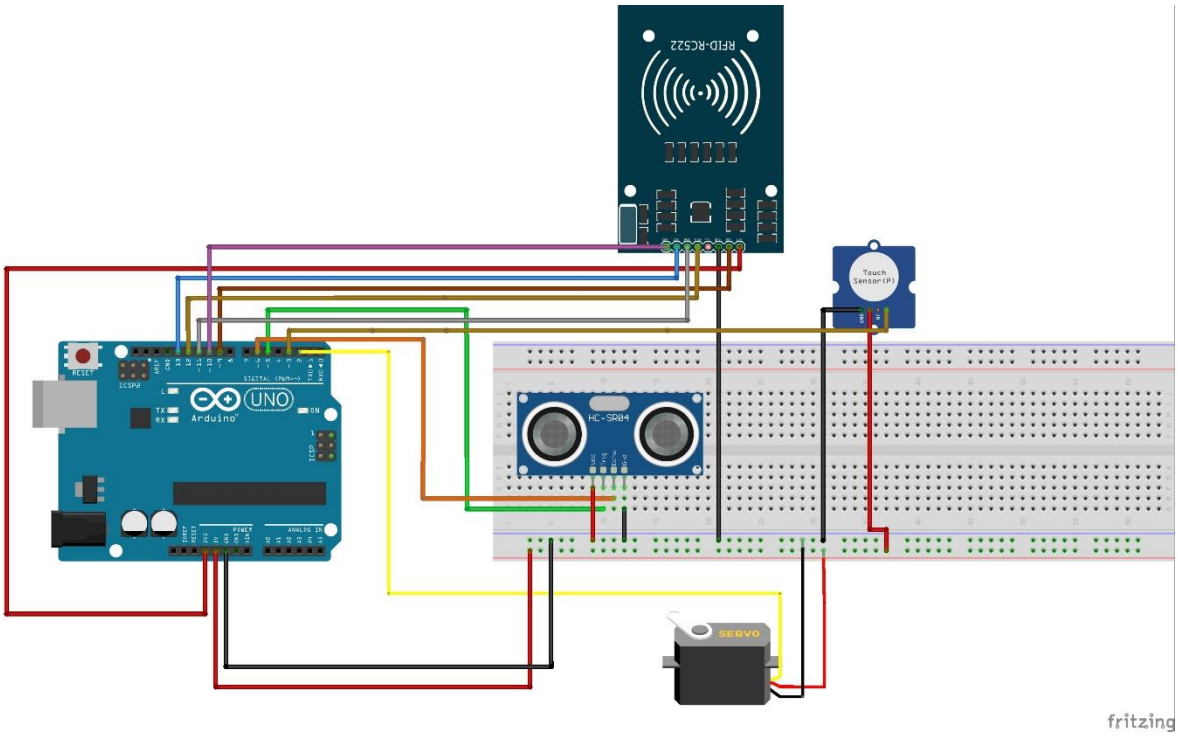
- Vcc: napajanje (3.3V)
- Gnd: ozemljitev
- Trig priključek: oddajanje ultrazvočnih zvokov
- Echo priključek: sprejem odboja ultrazvočnih zvokov

Trig priključek sva povezala na pin 5, echo pa na 6.

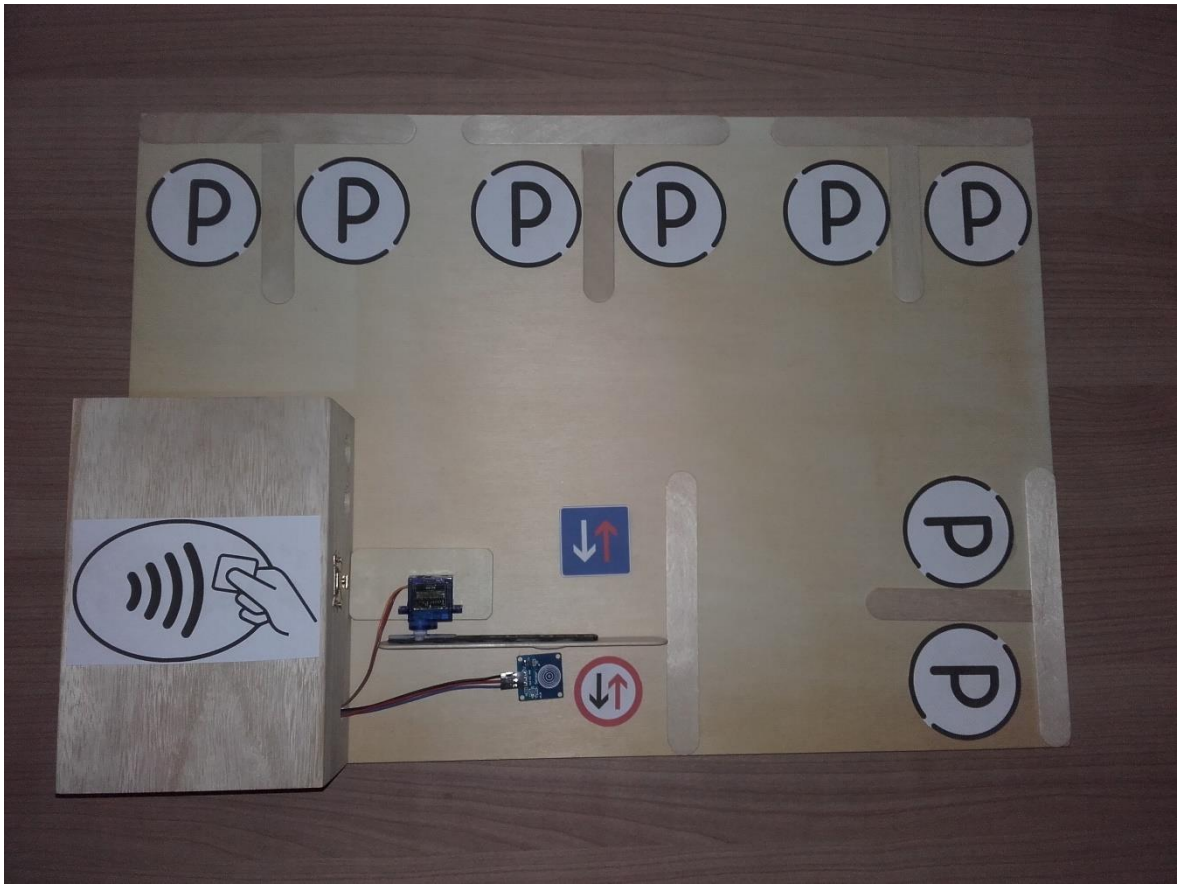
Nazadnje sva še povezala senzor dotika. Ta ima tri priključke.

- Vcc: napajanje (3.3V)
- Gnd: ozemljitev
- Sig: signal; visoko stanje, ko je pritisnjen oz. nizko stanje kadar ni

Celotna vezava elementov je vidna na sliki 16.

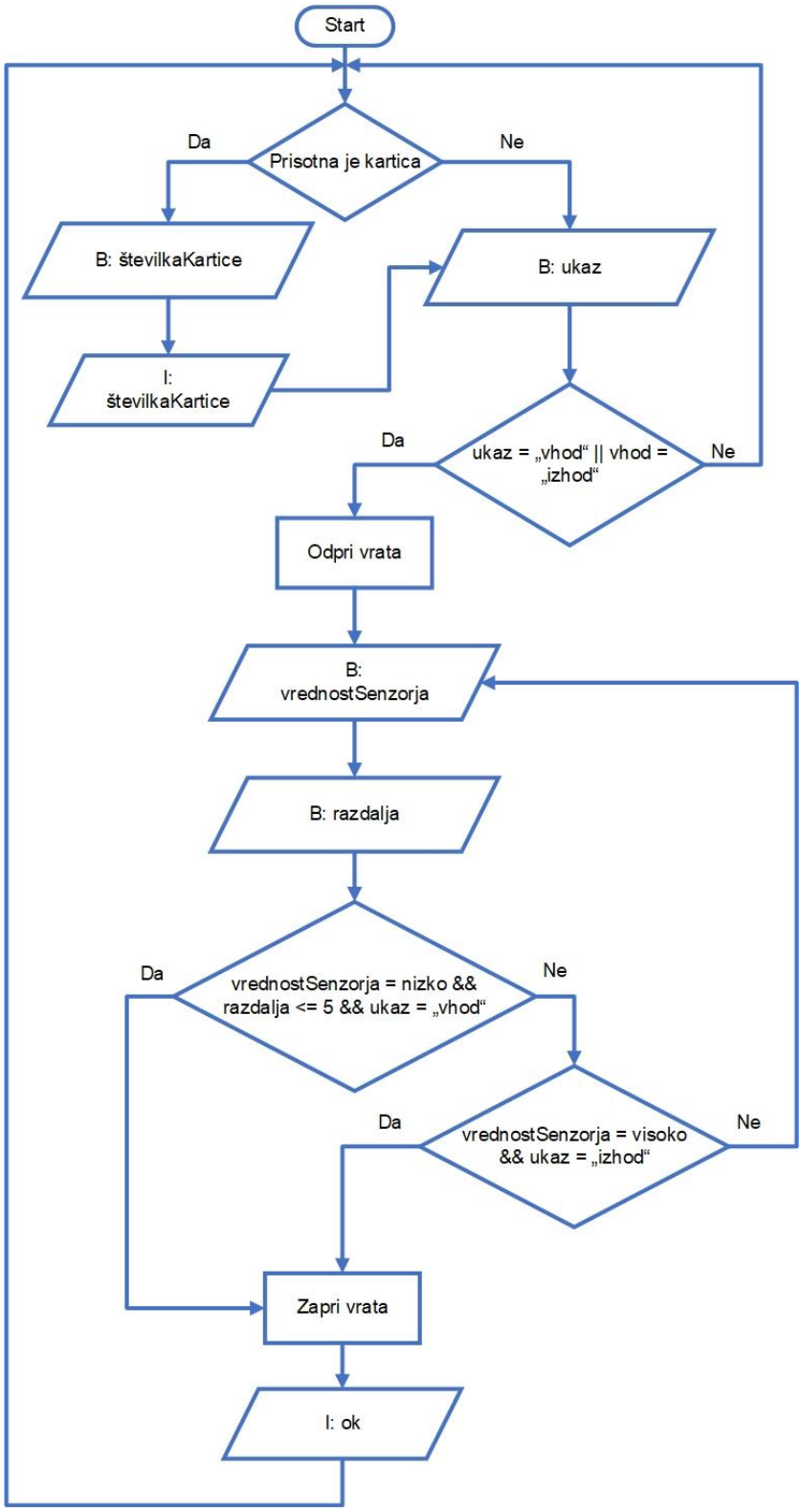


Slika 24: Vezava komponent na Arduino.



Slika 25: Končni videz modela.

Zatem sva se lotila pisanja programa za nadzor Arduina. Ker je kode kar veliko, sva se odločila, da narediva diagram poteka, predstaviva delovanje in prikaževa najpomembnejše dele kode.



Slika 26: Diagram poteka Arduino programa.

V primeru, da Arduino zazna novo kartico RFID, jo prebere in pošlje na računalnik, kjer jo prevzame C# program, ki nato nadaljnjo odreagira. Če kartica ni prisotna, Arduino prebere ukaz, pridobljen iz serijske komunikacije. Če je ukaz karkoli drugega kot vhod ali izhod, se program vrne na začetek.

Če ni tako, servo motor odpre vrata. Nato program prebere stanje senzorja dotika (visoko – high ali nizko - low) in razdaljo s pomočjo temu namenjenega senzorja.

Če je ukaz enak besedi vhod in če je stanje senzorja dotika nizko in je razdalja manjša ali enaka 5 cm, to pomeni, da je voznik zapeljal v parkirno hišo in servo motor lahko zapre zapornico.

Če je ukaz enak besedi »izhod« in če je stanje senzorja dotika visoko, to pomeni, da je voznik zapeljal iz parkirne hiše in zapornica se lahko zapre.

Če pa nobeden od zgornjih pogojev ni izpolnjen, program ponovni prebere vrednosti senzorjev in cikel se ponovi.

Po zaprtju zapornice program pošlje besedo »ok« v računalnik, da lahko nato ta ali pozove uporabnika, da lahko parkira ali pa se od njega poslovi in opravi transakcijo za opravljeno storitev.

```
void loop() {
  if(RFID.PICC_IsNewCardPresent() && RFID.PICC_ReadCardSerial())  {
    for (byte i = 0; i < 4; i++)  {
      poljeZaKodo[i] = RFID.uid.uidByte[i];
      koda += String(poljeZaKodo[i], HEX) + " ";
    }
    koda.toUpperCase();

    Serial.println(koda);
    koda = "";

    RFID.PICC_HaltA();
    RFID.PCD_StopCryptol();
  }

  ukaz = Serial.readString();

  if (ukaz == "vhod" || ukaz == "izhod") {
    OdpriVrata();
    ZapriVrata();
    Serial.println("ok");
  }
}
```

Slika 27: Glavna metoda programa.

```
void OdpriVrata() {
  servoMotor.attach(servoPin);

  for (int i = servoMin; i <= servoMax; i += 1) {
     servoMotor.write(i);
     delay(10);
  }
}
```

Slika 28: Odpiranje vrat.

3.5 Program za komuniciranje z Arduinom in podatkovno bazo

Zaradi obsežnosti programa, ki skupaj z Arduinom nadzira delovanje, sva s programom Microsoft Visio sestavila diagram poteka.

Celoten program se izvaja na svoji niti[9]. Ko Arduino pošlje ukaz in tako sproži prekinitev[10], se izvede koda, navedena znotraj te niti oz. metode.

Če je ta ukaz enak besedi »ok«, kar pomeni, da je uporabnik prišel ali odšel, se spremeni vsebina besedila na grafičnem vmesniku programa, ki se izvaja na računalniku. Besedilo pozove naslednjega uporabnika, naj približa kartico k senzorju.

V nasprotnem primeru se ustvari objekt razreda Stranka. Nato se preveri, če ta stranka sploh obstaja, saj so ključki za ta senzor cenovno zelo ugodni in bi se hitro lahko zgodilo, da bi ga kdo kupil in poskušal vstopiti, ki pa sploh nima dovoljenja vstopa. Če stranka ne obstaja v podatkovni bazi, program to izpiše.

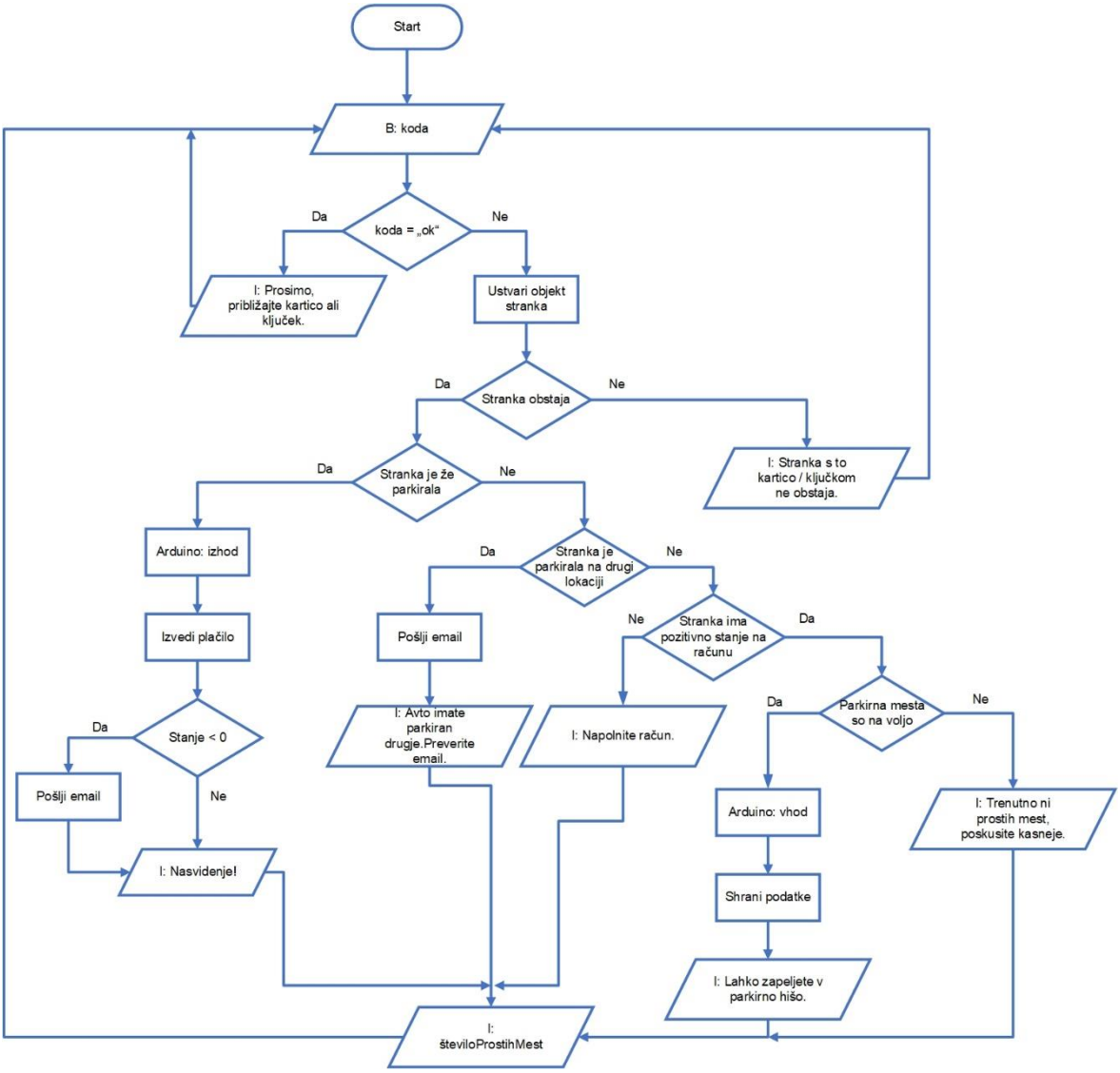
Nato se preveri ali je stranka že parkirala in želi zapustiti parkirno hišo. Če da, program obvesti Arduina za ta namen z besedo »izhod« in Arduino lahko odpre zapornico. Nato program izvede plačilo za parkiranje in po potrebi, če je novo denarno stanje stranke negativno, obvesti stranko o tem ter se poslovi.

V nasprotnem primeru se preveri, če je stranka že parkirala na kateri drugi lokaciji. Po potrebi se jo obvesti po elektronski pošti, kje je parkirala.

[9] Proces, ki ga procesor izvaja.
[10] Dogodek, ki sproži začasno prekinitev izvajanja.

Za tem se pogleda denarno stanje stranke. Če je pozitivno in so še prosta mesta za parkiranje, Arduino odpre zapornico, program pa shrani vse potrebne podatke v podatkovno bazo. Če pa stanje ni pozitivno ali pa ni voljo prostih mest, se to obvesti uporabnika.

Na koncu se na grafičnem vmesniku izpiše novo število prostih parkirnih mest.

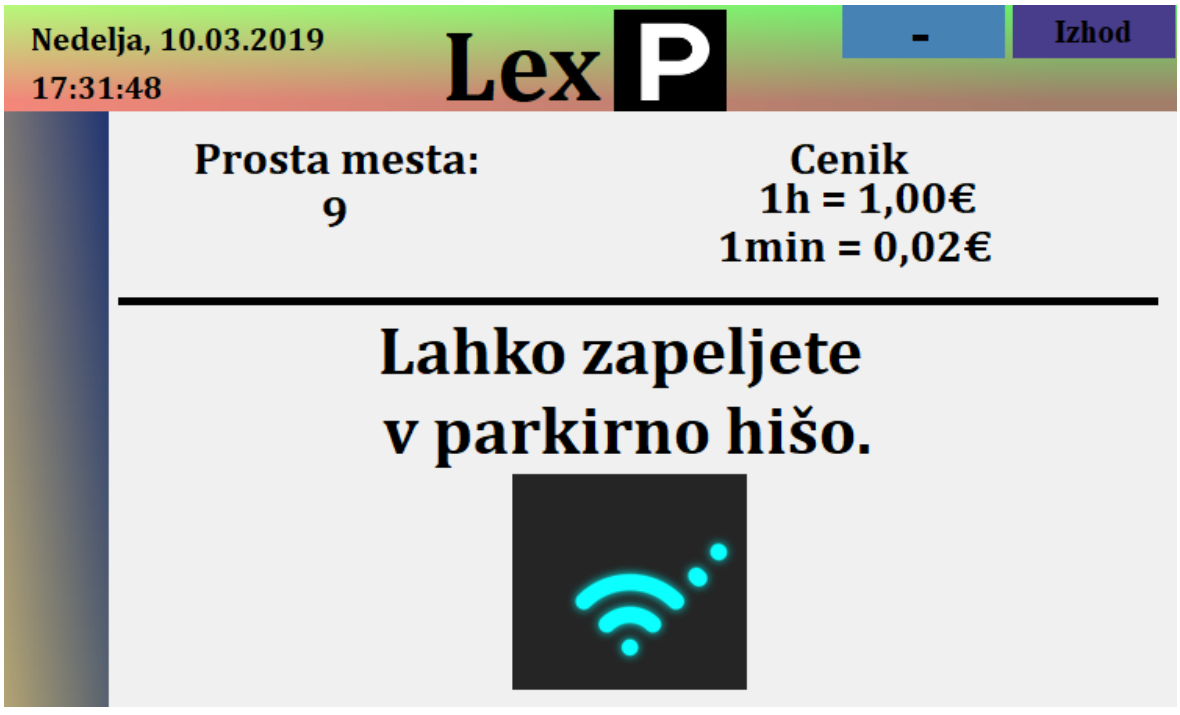


Slika 29: Diagram poteka C# programa.

Slika 21 prikazuje glavno okno programa, ki komunicira s stranko.

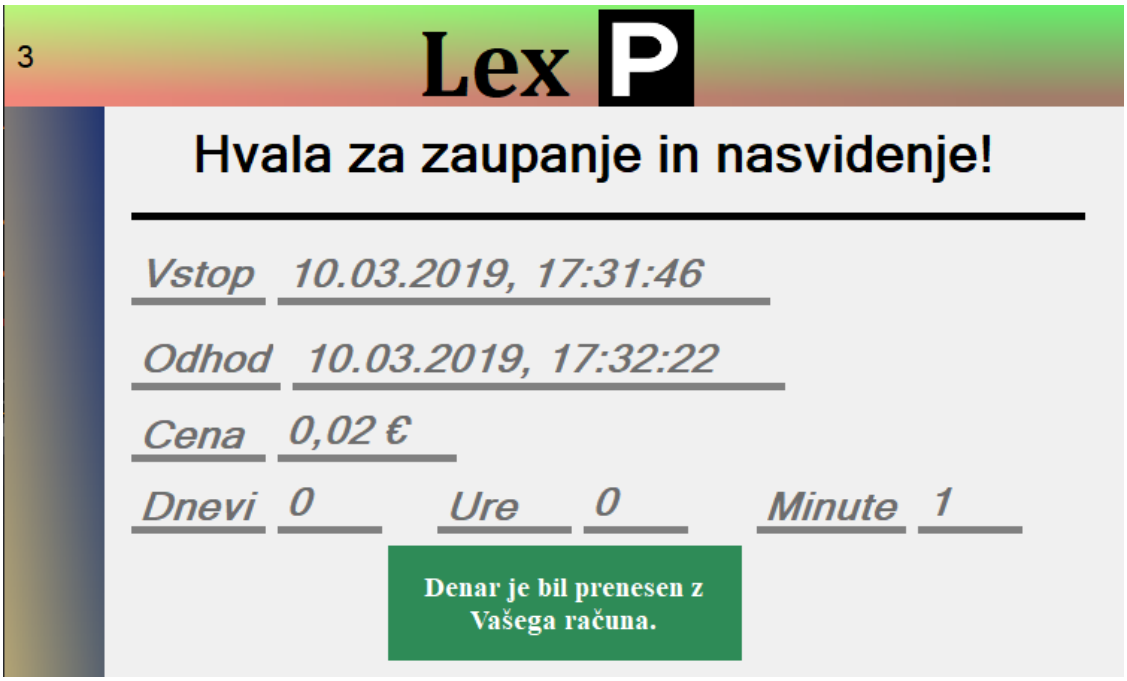
Slika 30: Glavno okno.

Slika 22 prikazuje videz glavnega okna, ko stranka vstopa v parkirno hišo.



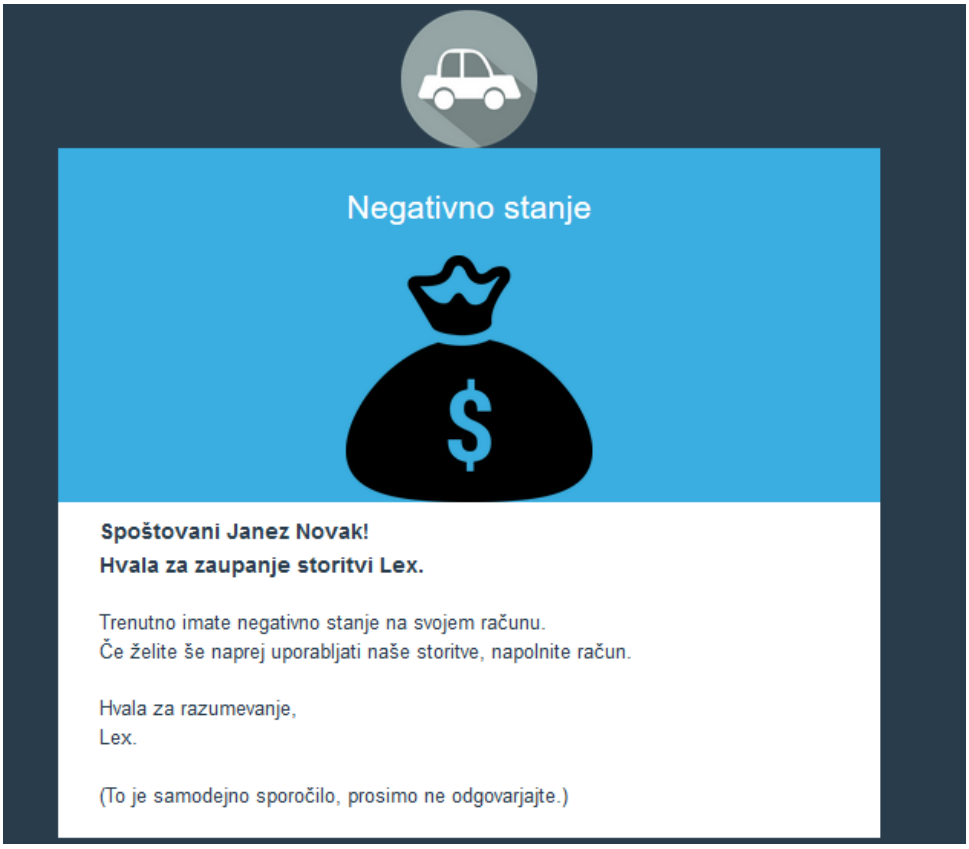
Slika 31: Vhod v parkirno hišo.

Ko stranka zapusti parkirno hišo, ji program izpiše podatke o parkiranju (slika 23).



Slika 32: Izhod iz parkirne hiše.

Lahko pa se zgodi, da po plačilu stranka nima več pozitivnega stanja na računu. Takrat se jo o tem obvesti po elektronski pošti (slika 24).



Slika 33: Email o negativnem stanju računa.

Če pa stranka želi izstopiti, kjer ni parkirala, se ji pošlje elektronska pošta o naslovu parkiranega vozila (slika 25).

Slika 34: Email, kadar hoče stranka izstopiti iz parkirne hiše, kjer ni parkirala.

3.6 Aplikacija za uporabnika

Aplikacija je narejena z namenom, da uporabnikom olajša marsikateri problem v parkirnih hišah. Od iskanja parkirnega mesta do plačevanja in vodenja podatkov o zgodovini parkiranja. Njen videz je bil ročno zasnovan in pozneje upodobljen v Android Studiu. Cilj tega je bil preprost videz, saj je aplikacija namenjena širšemu krogu ljudi.

Kaj vse nam omogoča:

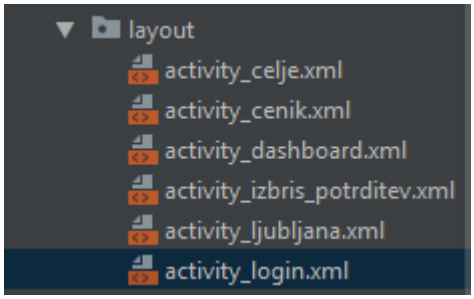
- Preko aplikacije lahko na račun nalagamo denar
- Vodi zgodovino naših transakcij

- Vodi zgodovino parkirnih mest
- Preko nje lahko izvemo lokacijo parkiranja
- Lahko si ogledamo podatke o določeni parkirni hiši

Za najin logo sva se odločila narisati preprost simbol za parkiranje, ki je poznan množici in predstavlja glavni namen aplikacije. Logo sva narisala s pomočjo programa 3D-Slikar in ga pozneje dodala v projekt v Android studiu.

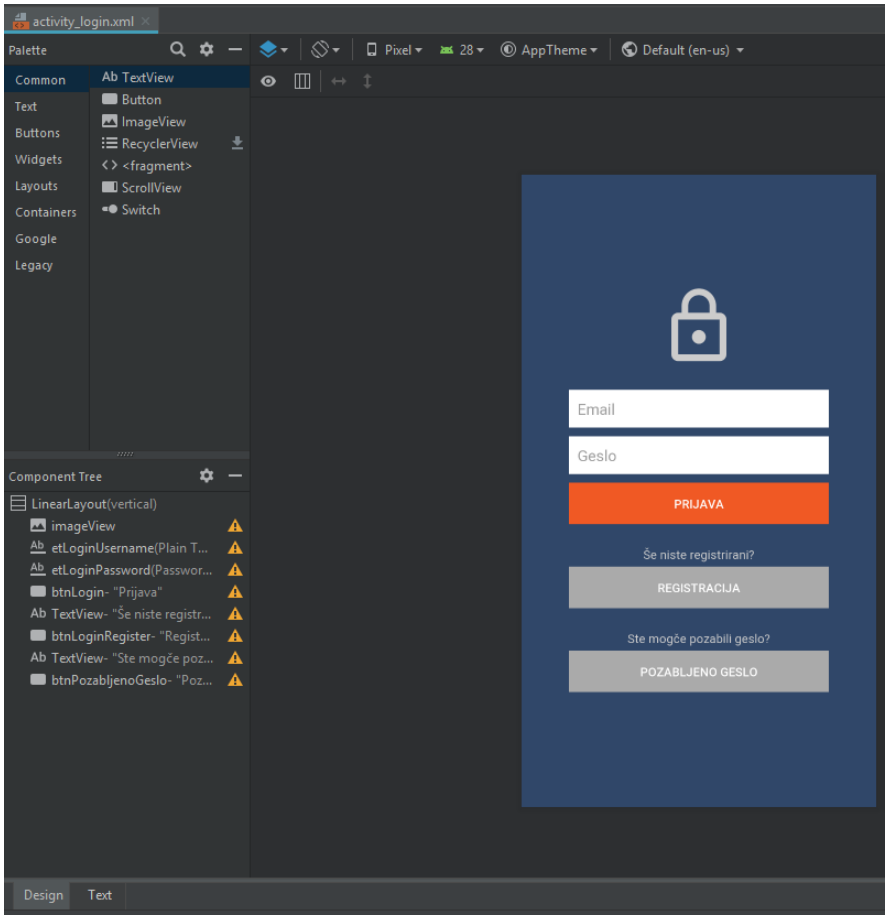
Slika 35: Logo aplikacije.

Nato sva se lotila videza glavnega okna, ki se požene ob zagonu aplikacije. To je prijavno okno, ki ima podoben videz kot večina mobilnih aplikacij. Videz aplikacije lahko zasnujemo s kodo ali pa ga oblikujemo ročno. Sprva v podmapi »layout« Izberemo našo xml datoteko.



Slika 36: Podmapa layout - activity_login.xml.

Odpre se nam okno v katerem lahko urejamo ročno našo xml datoteko. Deluje na način Drag and drop funkcije, z uporabo miške izoblikujemo elemente, jim določimo velikost in jih postavimo v okno. Ker pa je postavitev elementov s kodo natančneje določena sva se odločila za možnost »text« oziroma tekstovni način urejevanja izgleda okna.



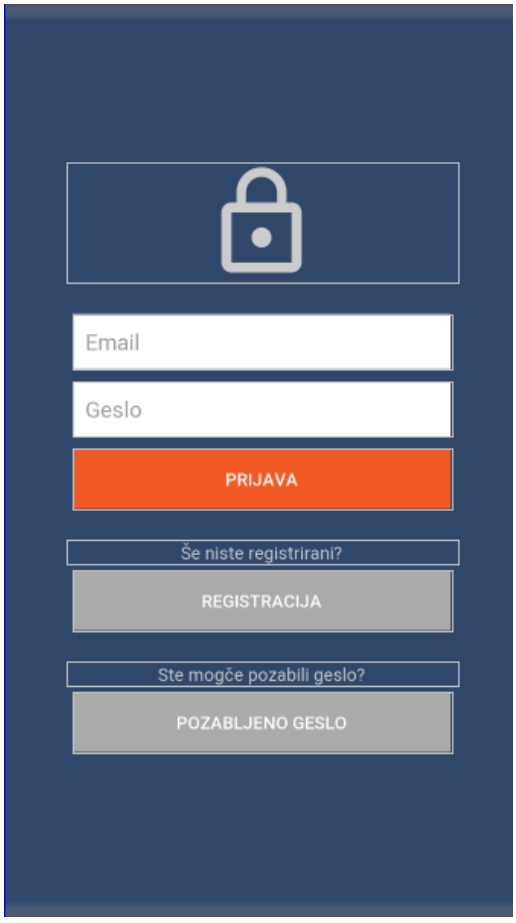
Slika 37: Urejanje XML Activity.

Vanj zapisujemo elemente s kodo. Pisanje kode je podobno kot pri HTML-ju (jezik za izdelavo spletnih strani) in CSS-ju (urejanje videza spletnih strani). Izberemo element in znotraj njega določimo lastnosti, kot so velikost (height in width), id (za uporabo v JAVI), potem barvo in podobno. Vse to se dodaja z besedo »android« za katero navedemo lastnost, ki jo bomo uredili.

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:src="@drawable/ic_lock_white" />
```

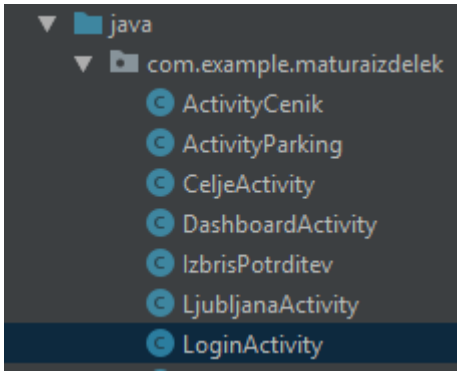
Slika 32: Primer XML elementa v tekstovni obliki.

Na okno prijava sva dodala dva EditText-a (eden za vpis elektronskega naslova, drug pa je za vpis gesla) in tri gumbe. Gumb Prijava preko katere program preveri vnesene podatke. Gumb za pozabljeno geslo, saj se marsikateremu lahko zgodi, da geslo pozabi in s tem na elektronski naslov dobi geslo. In še gumb Registracija za osebe, ki še niso registrirane.



Slika 33: Končni videz prijavnega okna.

Ko je bilo okno v končni obliki, je bilo treba sprogramirati posamezne elemente v oknu. To se naredi z javo datoteko, ki se je ustvarila skupaj z XML datoteko. Te datoteke najdemo v mapi »java«, bolj natančno v podmapi »com.example.maturaizdelek«.



Slika 38: Pot do LoginActivity-a

Treba je sprogramirati vsak gumb. Uporabila sva »setOnClickListener« metodo, ki ob pritisku na gumb izvede kodo, napisano znotraj te metode. Metoda lahko, na primer odpre novo okno, pridobi podatke iz podatkovne baze.

```
login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Vrne
        username = etUsername.getText().toString().toLowerCase().trim();
        password = etPassword.getText().toString().trim();
        if (validateInputs()) {
            login();
        }
    }
});
```

Slika 39: Primer metode setOnClickListener.

Ob pritisku na gumb metoda prebere elektronski naslov in geslo ter jih s pogojnim stavkom preveri. V primeru, da je vse pravilno vpisano, se pokliče metodo »login()«. Ta pošlje prošnjo za povezavo na podatkovno bazo. Za vnos v bazo in preverjanje uporabimo programski jezik PHP. V javi napišemo URL pot in se nanjo pozneje v kodi sklicujemo.

```php
<?php
define('DB_UPORABNIK', "");
define('DB_GESLO', "");
define('DB_BAZA', "");
define('DB_SERVER', "");  //Zaradi varnostnih razlogov, podatki niso vpisani!

$con = mysqli_connect(DB_SERVER,DB_UPORABNIK,DB_GESLO,DB_BAZA);


if(mysqli_connect_errno())
{
    echo "Povezava na strežnik ni uspela: " . mysqli_connect_error();
}

?>
```

Slika 40: Vzpostavitev povezave na strežnik.

Ko se povezava na bazo uspešno vzpostavi, se pokliče vsebina datoteke »login.php«. V tej datoteki se preveri, če se podatki iz podatkovne baze res ujemajo z vnesenimi in na podlagi tega se vrne rezultat Android Studiu.

```php
<?php
$response = array();
include 'db_connect.php';
include 'functions.php';

//Pridobi vhodne parametre
$inputJSON = file_get_contents('php://input');
$input = json_decode($inputJSON, TRUE); //pretvori JSON v polje

//Preverja pravilnost podatkov
if(isset($input['email']) && isset($input['geslo'])){
    $email = $input['email'];
    $geslo = $input['geslo'];
    $query = "SELECT email,hash, salt FROM Stranka WHERE email = ?";

    if($stmt = $con->prepare($query)){
        $stmt->bind_param("s",$email);
        $stmt->execute();
        $stmt->bind_result($email,$passwordHashDB,$salt);
        if($stmt->fetch()){
            //Preveri geslo
            if(password_verify(concatPasswordWithSalt($geslo,$salt),$passwordHashDB)){
                $response["status"] = 0;
                $response["message"] = "Uspešno ste se prijavili";
                $response["email"] = $email;
            }
            else{
                $response["status"] = 1;
                $response["message"] = "Napačen vnos emaila in gesla";
            }
        }
        else{
            $response["status"] = 1;
            $response["message"] = "Napačen vnos emaila in gesla";
        }

        $stmt->close();
    }
}
else{
    $response["status"] = 2;
    $response["message"] = "Manjkajo vhodni podatki";
}
//Display the JSON response
echo json_encode($response);
?>
```

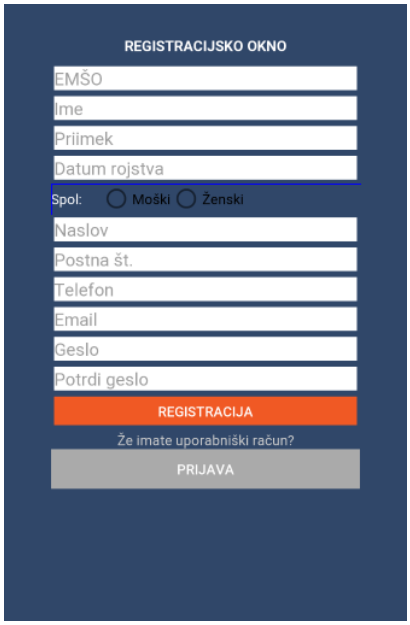
Slika 35: Koda za prijavo.

Tako se ob uspešni prijavi ustvari tako imenovana seja z namenom, da se uporabniku ni treba ponovno prijaviti vsakič, ko odpre aplikacijo. Seja se konča ob pritisku na gumb »Odjava«, ki ga najdemo na ostalih oknih.

V primeru, da je uporabnik pozabil svoje geslo, lahko preko gumba »Pozabljeno geslo« uporabniku na elektronski naslov pošlje novo geslo.

Slika 36: Okno za pozabljeno geslo.

Nato sva se lotila okna »Registracija«. Na tem oknu se uporabnik registrira. Nato se podatki shranijo v bazo.

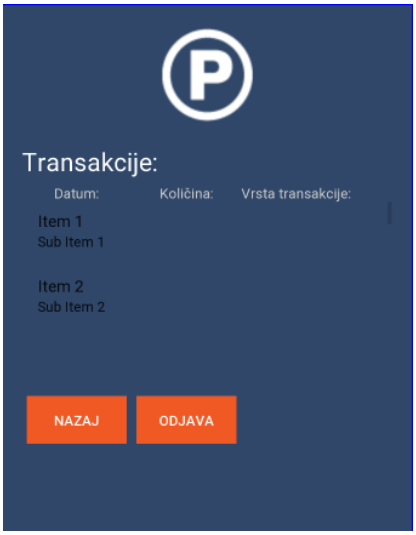


Slika 37: Okno za registracijo.

Ko pa se uporabnik prijavi v aplikacijo, ga pričaka okno, kjer lahko dodaja denar na račun, pregleda, ali je kje trenutno parkiran. Lahko pa tudi, ob kliku na ustrezen gumb, odide na druga okna.

Slika 41: Glavno okno.

V oknu transakcije lahko vidi svoja nalaganja denarja na račun.



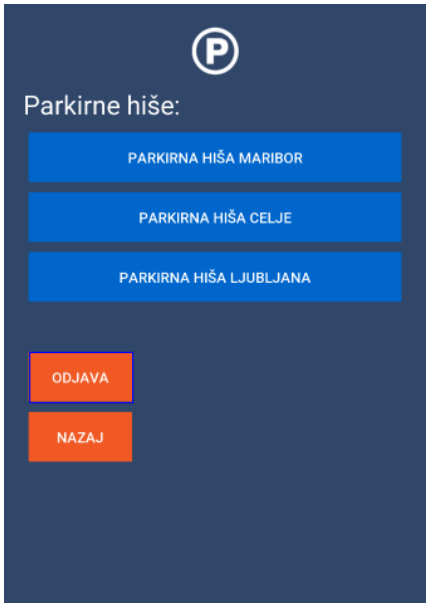
Slika 42: Transakcije.

V oknu Moj račun pa lahko ureja svoje podatke ali pa izbriše svoj račun.

Slika 43: Urejanje računa.

Nad gumbom »Odjava« lahko vidi trenuten parkirni status. V primeru, da je kje parkiran, se izpišejo podatki te parkirne hiše, v nasprotnem primeru se ne izpiše nič.

Gumb »Podatki o parkirnih hišah« nas odvedejo na novo okno, kjer izberemo med večjimi mesti, ki imajo parkirne hiše.



Slika 38: Parkirne hiše.

Ko stranka izbere želeno mesto, se v tabeli izpišejo vse parkirne hiše znotraj izbranega mesta. V tabeli je razvidno število prostih mest za posamezno parkirno hišo, prav tako lahko razbere lokacijo in ceno za uro parkiranja.

Slika 39: Podatki parkirnih hiš.

4 Stroški

Razvoj vsake projekta je povezan s financiranjem. Večji projekti oz. velika podjetja imajo na voljo večje količine finančnih sredstev, potrebnih za razvoj. Majhni projekti pa ravno nasprotno.

Cilj pri obeh pa ostaja enak. Čim hitreje, za čim manj stroškov razviti čim boljši projekt. Tega načela sva se tudi držala.

Imela sva srečo, da sva veliko komponent, potrebnih za ta projekt, že imela. Posebej velik zalogaj bi predstavljali stroški za nakup komponent, potrebnih za strežnik. Kar pa zadeva komponente za del z Arduinom, je globalizacija velik plus, saj so komponente s tujih trgov, predvsem s kitajskega, cenovno zelo ugodne. Edina težava je čas dostave, saj ta traja od dveh pa do štirih tednov ali pa tudi dlje, kar pa se sicer redko zgodi.

Tabela 2 prikazuje najine stroške.

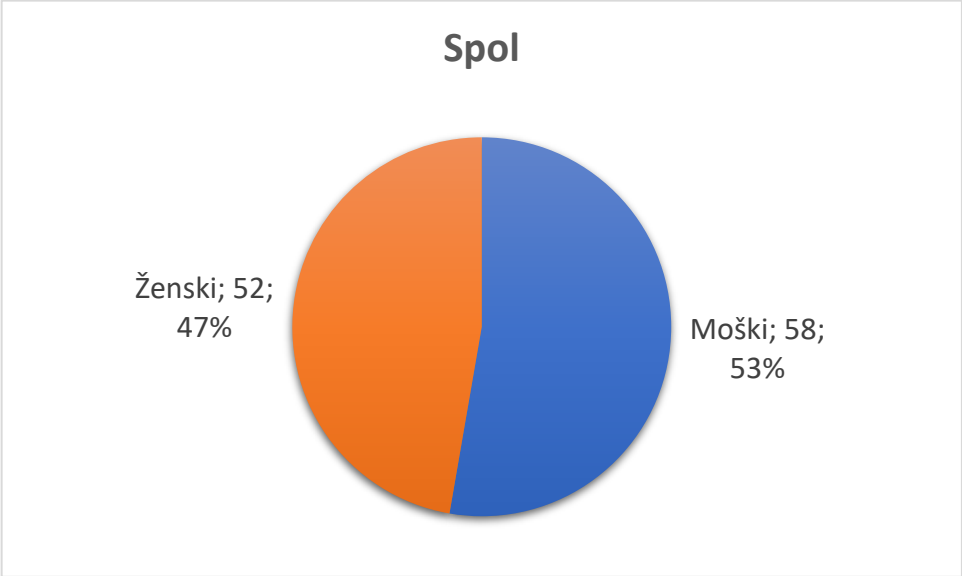
Tabela 2: Stroški.

Ime komponente	Cena	Nabava
Napajalnik	20,00 €	Mimovrste
Nosilec za disk	2,73 €	Mimovrste
RFID senzor	2,88 €	Ebay
Senzor razdalje	1,00 €	Ebay
Senzor dotika	1,00 €	Ebay
Servo server	1,30 €	Ebay
Žice	1,00 €	Ebay
Lesene palčke	2,00 €	Lekarna
RAM pomnilnika	6,00 €	Ebay
Sekundno lepilo	4, 00 €	Merkator
Lesena škatlica	7,00 €	Bead
Skupaj:	44,91 €	

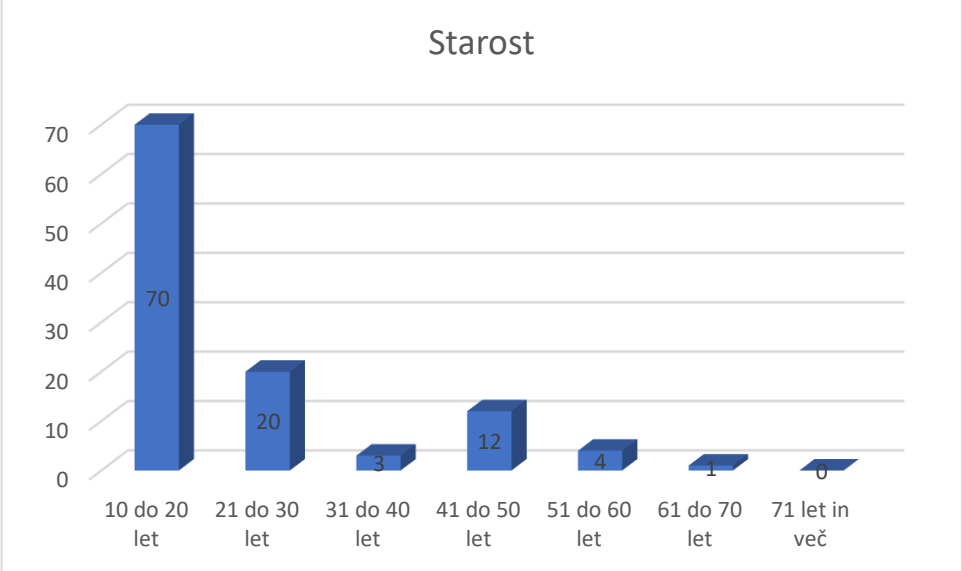
5 Analiza ankete

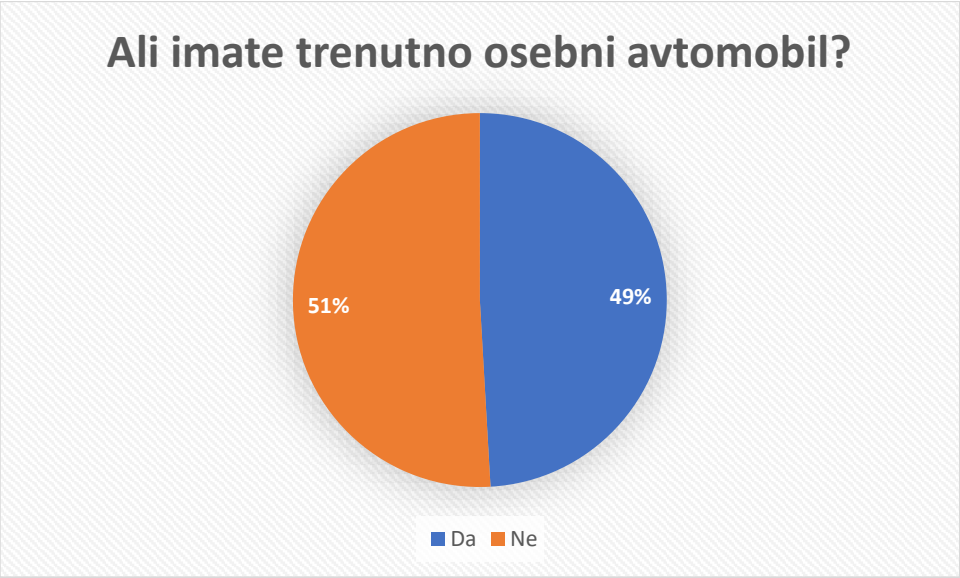
Anketa je vsebovala 9 vprašanj. Anketiranje je potekalo preko spletne strani 1ka od 4. 2. 2019 do 28. 2. 2019.

V anketi je sodelovalo 110 oseb, od tega 52 žensk (47 %) in 58 moških (53 %).

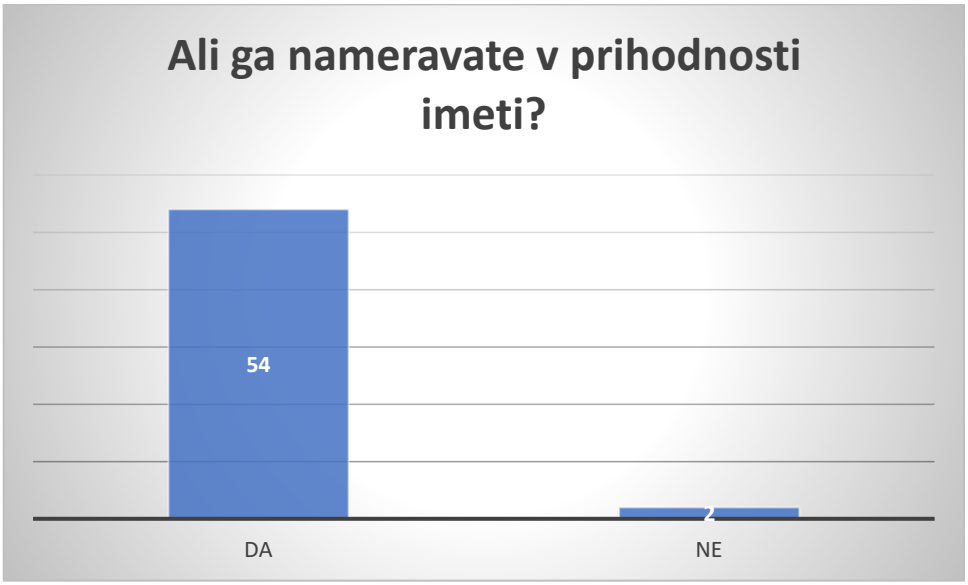


Največ anketirancev je bilo starih med 10 in 20 let.

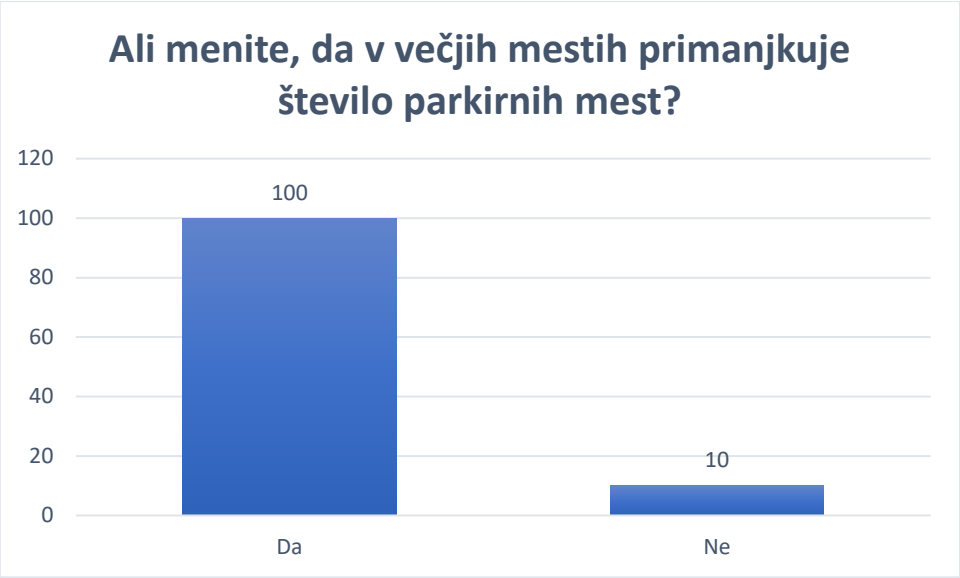




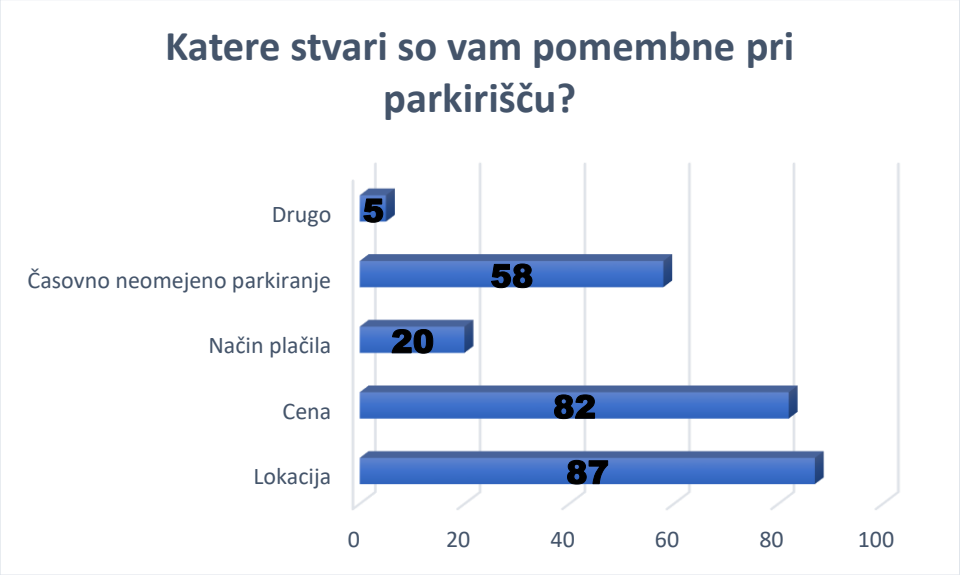
V povezavi s starostjo anketirancev je trenutno približno polovica oseb brez osebnega avtomobila.



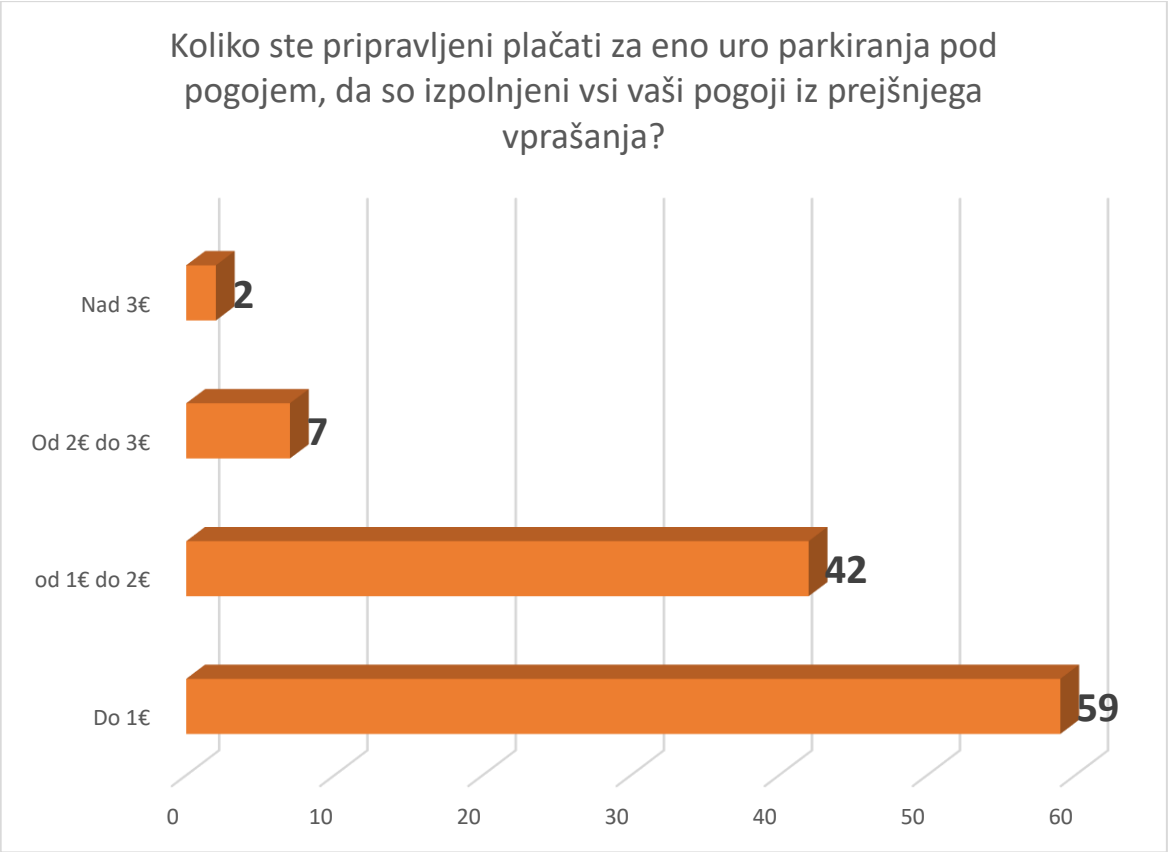
Od oseb, ki so odgovorile, da avtomobila nimajo, jih je kar 54 (od 56) odgovorilo, da ga v prihodnosti bodo imeli, kar vsekakor pomeni, da obstaja priložnost za uspeh najinega projekta.



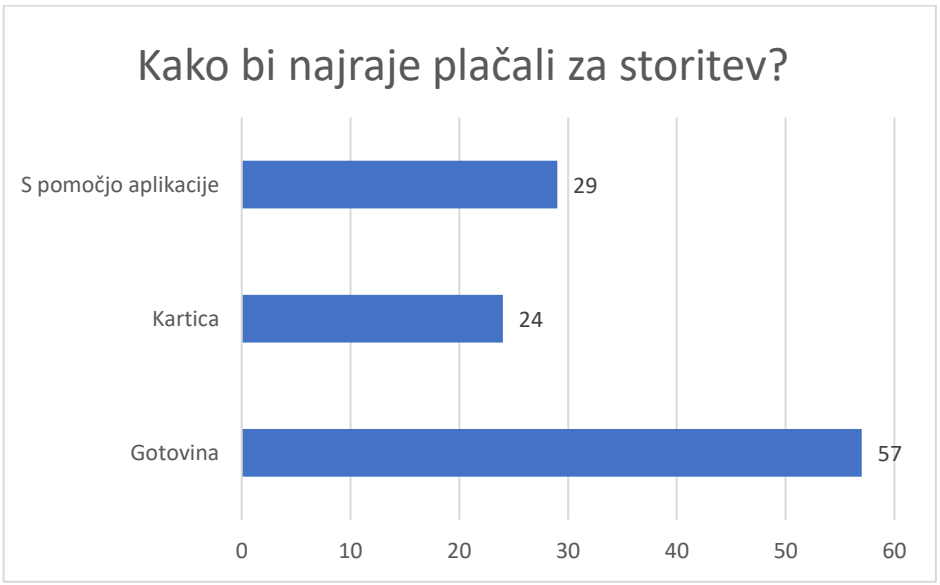
Težava večjih mest v Sloveniji še vedno ostaja pomanjkanje parkirnih mest.



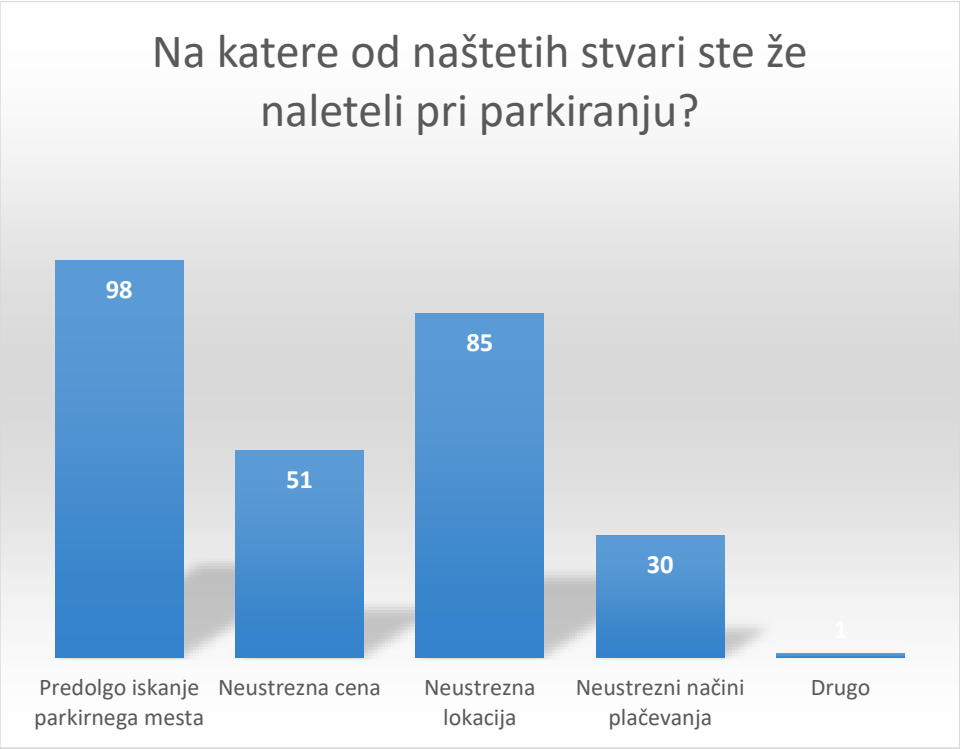
Najpomembnejša stvar anketirancev pri parkirišču je lokacija, tesno ji sledi cena. Na tretjem mestu je časovno neomejeno parkiranje, kar bo najina aplikacija omogočala.



Kot pričakovano je bil največkrat izbran odgovor »Do 1 €«, saj si vsi želijo najnižje stroške. Ta rezultat bo tudi prišel prav ob določitvi cene za parkirno hišo.



Večina še vedno najraje plačuje z gotovino. Rezultati tega vprašanja so bili presenečenje, posebej, ker je večina anketirancev mlajših od dvajset let.



Odgovori pod drugo:

- Neustrezno označena parkirišča

Večina oseb se je že soočila z naštetimi problemi, največji problem predstavljata predolgo iskanje in neustrezna lokacija. Te informacije so pomembne predvsem pri izvedbi projekta.

6 Zaključek

6.1 Hipoteze

1. Proces parkirnih hiš je mogoče poenostaviti.
2. Stranke bodo za storitev najraje plačale s pomočjo aplikacije.
3. Stroški modela ne bodo presegli 80 €.
4. V mestih vlada pomanjkanje parkirnih mest.

Prvo hipotezo potrjujeva, saj je uporaba najinega načina za parkirne hiše poenostavila celoten proces, osebe pa so prihranile na času.

Drugo hipotezo zavračava, saj bi anketiranci še vedno najraje plačali z gotovino. Meniva, da je rezultat takšen, ker so anketiranci navajeni na plačevanje z gotovino.

Tretjo hipotezo potrjujeva, saj so skupni stroški dosegli malo manj kot 45 €. Ob tem velja poudariti, da bi v primeru nakupa novih komponent, cena skokovito narasla in presegla mejo 80 €.

Četrto hipotezo potrjujeva, saj je iz raziskave vidno, da osebe menijo, da v mestih primanjkuje parkirnih mest.

6.2 Nadaljnjo delo in predlogi

Projekt ima velik potencial za nadaljnji razvoj. Prva stvar, ki jo predvidevava, je uporaba strojnega vida in kamer za prepoznavo avtomobilskih tablic. Na ta način se še dodatno izboljša prihranek pri času, saj vozniku sploh ne bi bilo treba ustaviti. Druga stvar bi bila aplikacija za skrbnika sistema.

6.3 Zaključna misel

Na koncu razvoja sva ugotovila, da so računalniki vsepovsod okoli nas. Mobilne naprave, pametne ure, pametni televizorji … so stvari, ki jih ljudje uporabljajo na dnevni ravni in se jim ne bodo kar tako odrekli.

Brez podatkovnih baz si današnjega interneta ne moremo predstavljati. Vse informacije morajo biti shranjene nekje z namenom, da jih lahko pozneje pridobimo z namenom pravilnega in ustreznega delovanja aplikacije.

Vsaka veriga je toliko močna kot njen najšibkejši člen. Če ta člen popusti, se veriga strga. Enak princip ostaja pri varovanju strežnikov. Noben upravitelj strežnika ne želi, da bi podatki prišli v napačne roke. Določene storitve uporabljajo močne varnostne protokole, druge ne. Močno šifriranje, varnostne kopije so stvari, ki zmanjšujejo šibkost sistema.

Za programiranje poznamo veliko programskih jezikov. Java, Python, C# … Vsak ima svoje prednosti in slabosti. V tem projektu je bil uporabljen C#, ker je primeren za začetnike, je dobro dokumentiran in omogoča veliko stvari.

V današnjem svetu ima skoraj vsak posameznik lasten telefon. Prav tako za skoraj vsak namen obstaja aplikacija. V svetu računalništva pa je zelo pogosto prepletanje različnih arhitektur. Na primer: doma imamo pametni termostat, preko aplikacije za telefon pa lahko nastavimo poljubno temperaturo.

Najin projekt oziroma raziskovalna naloga vsebuje nekaj vsega po malem. Od strojne opreme, programske opreme, operacijskih sistemov do omrežij. Najin namen je bil združiti različne veje računalništva in ustvariti projekt, ki do določenega problema pristopa na drugačen način, kot so ljudje vajeni in s tem izboljšati določeno področje.

7 Viri

1. ardunant. Arduining. [Elektronski] 2012. [Navedeno: 28. februar 2019.] https://arduining.com/2012/10/13/arduino-parking-lot-filled/.

2. B_E_N. sparkfun. [Elektronski] [Navedeno: 28. feburar 2019.] https://learn.sparkfun.com/tutorials/what-is-an-arduino.

3. JIMBO. sparkfun. [Elektronski] [Navedeno: 28. feburar 2019.] https://learn.sparkfun.com/tutorials/serial-communication.

4. Nick. Youtube. [Elektronski] 2017. [Navedeno: 28. feburar 2019.] https://www.youtube.com/watch?v=vHeG3Gt6STE.

5. Arduino. Arduino. [Elektronski] [Navedeno: 28. februar 2019.] https://www.arduino.cc/.

6. EPC-RFID INFO. [Elektronski] [Navedeno: 28. februar 2019.] https://www.epc-rfid.info/rfid.

7. Keyence. [Elektronski] [Navedeno: 28. februar 2019.] https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/.

8. Dejan. How to mechatronics. [Elektronski] 18. marec 2018. [Navedeno: 28. februar 2019.] https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/.

9. sistemc. Med.Over.Net. [Elektronski] 11. februar 2010. [Navedeno: 28. februar 2019.] https://med.over.net/forum5/viewtopic.php?t=6020060.

10. Borko, Leon. SERŠ e-gradiva. [Elektronski] 2002. [Navedeno: 28. februar 2019.] http://www.s-sers.mb.edus.si/gradiva/w3/sistemi/registri.html.

11. Norman, Harvey. Harvey Norman. [Elektronski] 6. avgust 2018. [Navedeno: 28. februar 2019.] https://www.harveynorman.si/blog/ssd-ali-hdd-disk-katero-komponento-za-shranjevanje-podatkov-izbrati-nasvet-navdih/.

12. Arnes. Arnes. [Elektronski] [Navedeno: 28. februar 2019.] http://www2.arnes.si/~osljis6s/priponke/rom02/roki1.html.

13. Rouse, Margaret. Tech Targer. [Elektronski] Avgust 2008. [Navedeno: 10. marec 2019.] https://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP.

14. Wikipedia. [Elektronski] 27. Februar 2019. [Navedeno: 10. Marec 2019.] https://en.wikipedia.org/wiki/Object-oriented_programming.

15. Wikipedia. [Elektronski] 10. Marec 2019. [Navedeno: 10. Marec 2019.] https://en.wikipedia.org/wiki/Android_(operating_system).

8 Priloge

Pozdravljeni!
V svoji raziskovalni/maturitetni nalogi se ukvarjava s parkirnimi hišami.
Da bi svojo raziskavo izvedela čim natančnejše in objektivno, vas prosiva, da odgovarjate resno.
Anketa je anonimna.
Potrebovali boste minuto, da rešite anketo.

1. Ali imate trenutno osebni avtomobil?

a) Da
b) Ne

2. Če ste odgovorili ne, ali ga nameravate v prihodnosti imeti?

a) Da
b) Ne

3. Ali menite, da v večjih mestih primanjkuje število parkirnih mest?

a) Da
b) Ne

4. Katere stvari so vam pomembne pri parkirišču?

a) Lokacija
b) Cena
c) Način plačila
d) Časovno neomejeno parkiranje

e) Drugo

5. Koliko ste pripravljeni plačati za eno uro parkiranja pod pogojem, da so izpolnjeni vsi vaši pogoji iz prejšnjega vprašanja?

a) Do 1 €
b) Od 1 € do 2 €
c) Od 2 € do 3 €
d) Nad 3 €

6. Kako bi najraje plačali za storitev?

a) Gotovina
b) Kartica
c) S pomočjo aplikacije

7. Na katere od naštetih stvari ste že naleteli pri parkiranju?

a) Predolgo iskanje parkirnega mesta
b) Neustrezna cena
c) Neustrezna lokacija
d) Neustrezni načini plačila
e) Drugo

Vaš spol:

M Ž

V katero starostno kategorijo spadate

10 do 20 let

21 do 30 let

31 do 40 let

41 do 50 let

51 do 60 let

61 do 70 let

71 let in več

IZJAVA*

Mentor Gorazd Breznik v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom Model avtomatizirane parkirne hiše, katere avtorja sta David Slatinek in Gregor Zavolovšek:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 8. 3. 2019

ŠOLSKI CENTER CELJE ŠOLA ZA MEHANIKO, ELEKTROTEHNIKO IN RAČUNALNIŠTVO CELJE, Pot na Lavo 22

žig šole

Podpis mentorja

Podpis odgovorne osebe

POJASNILO

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.