Računalna aplikacija za RFID evidenciju studenata na nastavi

Nuić, Luka

Undergraduate thesis / Završni rad

2016

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:353766

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-18**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

RAČUNALNA APLIKACIJA ZA RFID EVIDENCIJU STUDENATA NA NASTAVI

Završni rad

Luka Nuić

Osijek, 2016.

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 2

2. PRIMIJENJENE TEHNOLOGIJE I ALATI 3

2.1. Odabir i detalji sklopa za očitavanje kartica 3

2.2. Alati i tehnologije potrebni za razvoj aplikacije 4

3. REALIZACIJA SUSTAVA 6

3.1. Izrada baze podataka 7

3.2. Izrada aplikacije 8

3.2.1. Izrada početnog sučelja 8

3.2.2. Izgled glavnog sučelja 10

3.2.3. Izrada upravljačkog sučelja 13

4. ZAKLJUČAK 15

LITERATURA 16

SAŽETAK 17

ABSTRACT 18

ŽIVOTOPIS 19

PRILOZI 20

1. UVOD

U ovom završnom radu cilj je prikazati način izrade aplikacije za praćenje dolaznost studenata na nastavi. Aplikacija će biti izrađena tako da se sa njom mogu služiti razni poduzetnici koji trebaju pratit nazočnost zaposlenika. Za rad aplikacije se koristi RFID senzor koji se spaja pomoću USB kabla na osobno računalo. Nakon spajanja korisnik treba pokrenuti aplikaciju te se prijaviti pomoću svoje identifikacijske kartice. Zatim ovisno o vrsti korisnika dobiva se drugačije sučelje. Obični korisnici mogu zapisivati nove dolaznosti sudionika i nadgledati sve prethodno zapisane podatke kao i listu svih sudionika i korisnika. Administrator osim svih mogućnosti korisnika također može upravljati zapisanim podatcima. Svi podatci sudionika i korisnika spremaju se na bazu podataka radi lakšeg prikaza i upravljanja. Također aplikacija omogućuje izvoz podataka o dolascima sudionika na korisnikov Google disk u obliku Excel tablice.

1.1. Zadatak završnog rada

U ovom završnom radu potrebno je napraviti aplikaciju koja će pomoću RFID čitača evidentirat sudionike na nastavi. Treba omogućiti običnim korisnicima jednostavno sučelje pomoću kojeg mogu evidentirati i nadgledati već upisane podatke o sudionicima. Administratorima treba omogućiti da upravljaju korisnicima i podatcima te treba stvoriti jasan prikaz dolaznosti pojedinih sudionika.

2. PRIMIJENJENE TEHNOLOGIJE I ALATI

2.1. Odabir i detalji sklopa za očitavanje kartica

Za čitanje kartica odabran je čitač pametnih kartica sa ugrađenim EM4100 modulom koji radi na 125 kHz. Sklop komunicira pomoću USB-a i ne zahtjeva dodatnu instalaciju programa za svoj rad. Prilikom očitanja kartice sklop čita prvih deset znamenki što je isto i identifikacijski broj kartice. Očitani broj šalje se na računalo te ga računalo prepoznaje kao da je uneseno pomoću tipkovnice i pritisnuta tipka „Enter“. Takvo ponašanje sklopa pojednostavljuje način kreiranja ovog sustava i omogućuje pristup točno traženim informacijama pametnih kartica.

Obilježja uređaja:

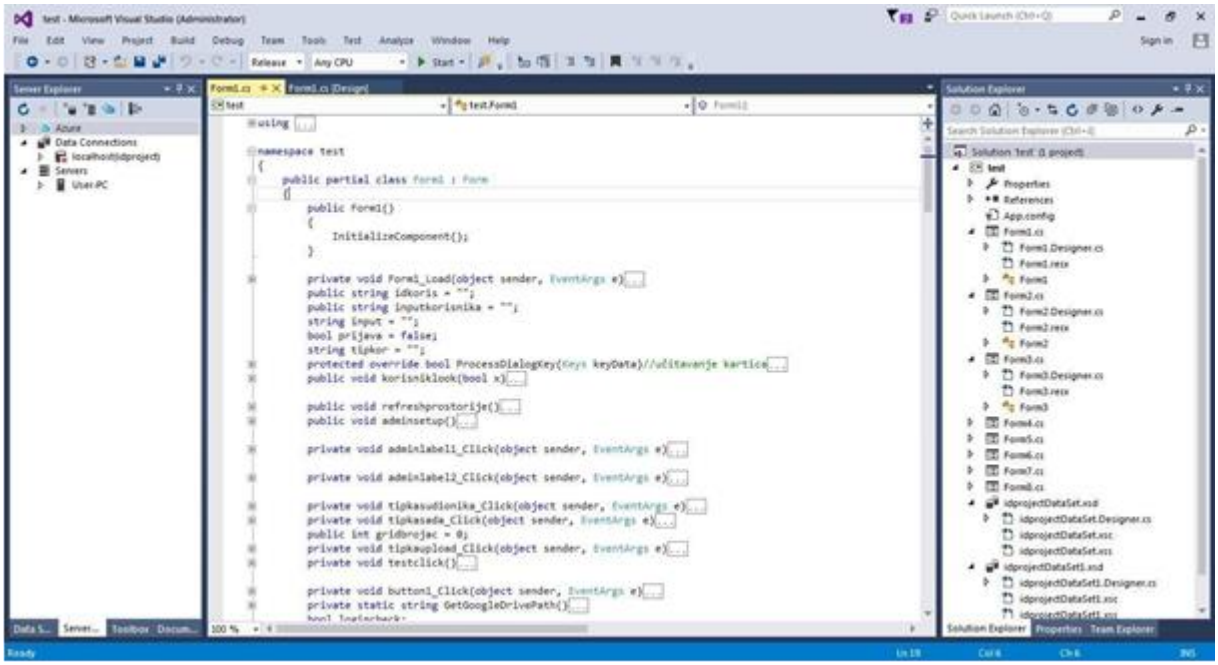
- USB komunikacija sa računalom (Plug and Play, podržava Windows 2000, 2003, XP, Vista, Windows7 i Linux)
- Čita sve 125Khz RFID EM4100 EM410X kartice/tagove
- LED dioda za definiranje stanja sklopa
- Ugrađeni „buzzer“
- Napajano preko USB kabla (tip: Mini B)
- Udaljenost čitanja: 5-8 cm
- Dimenzije: 10.5cm x 7cm x 1.2cm

Slika 1. USB čitač pametnih kartica

2.2. Alati i tehnologije potrebni za razvoj aplikacije

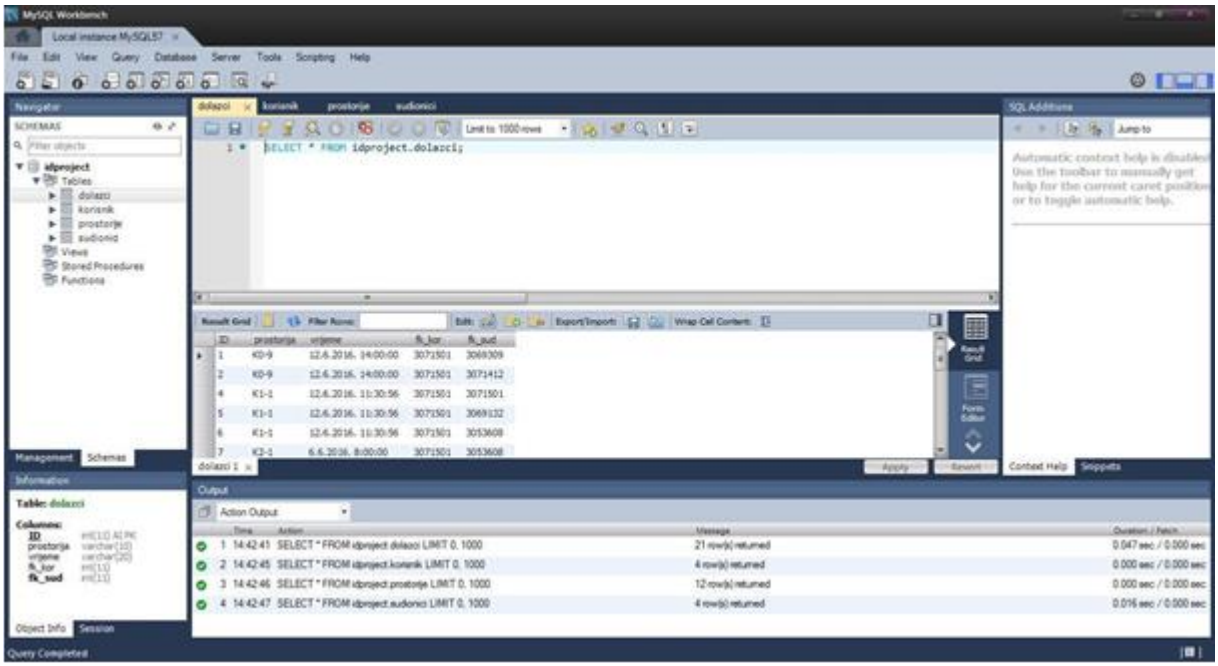
Za aplikaciju je potrebno razvojno okruženje koje ima mogućnost primanja identifikacijskih oznaka korisnika kao i sudionika prilikom kontrole pristupa. Također treba omogućiti običnim korisnicima jednostavno sučelje koje im omogućuje unos novih podataka. Administratori trebaju imati mogućnost, osim unosa podataka, kontrolu nad već unesenim podatcima, unos novih korisnika i sudionika,izmjenu podataka itd. Osim navedenih uvjeta aplikacija treba moći komunicirati sa bazom podataka i Microsoft Office programskim paketima. Za razvojno okruženje je izabran Visual Studio Community zbog velikog broja proširenja i jednostavnog sučelja za izradu Windows form aplikacija. Također omogućuje programiranje u raznim programskim jezicima kao C, C++, C#, Javascript i Python. Ova verzija Visual Studia je dostupna svima i podržava razvoj aplikacija na raznim platformama. Više o Visual Studiu na [1]. Prilikom stvaranja izgleda aplikacije potrebno je samo postaviti odgovarajuće objekte unutar forme. Takav način izrade omogućuje stvaranje jednostavnih aplikacija za manje sustave. Izgled sučelja prikazan je na Slici 2. Za stvaranje interakcije objekata korišten je programski jezik C#.

MySQL je izabran za upravljanje bazom podataka zbog toga što je besplatan i njegov izvorni kod je dostupan svima. Isto tako MySQL ima vlastito razvojno okruženje pod nazivom „Workbench“. Izgled MySQL Workbench sučelja prikazan je na Slici 3. Pomoću tog programa



Slika 2. Razvojno okruženje Visual Studio

može se, uz određeno znanje SQL naredbi, kreirati i nadgledati podatci unutar baze podataka. Sve baze unutar MySQL-a su relacijskog tipa zbog čega je skladištenje i pretraživanje velikog broja podataka brzo i sigurno. Još jedna od prednosti MySQL je podrška proširenja za razne programske jezike kao: PHP, Java , Perl, Python itd. Za više informacija o MySQL vidi [2]. Sama komunikacija između aplikacije i baze podataka biti će obavljana pomoću određenih naredbi unutar C# okruženja pomoću kojim se pokreće komunikacije te se omogućuje pokretanje određenih MySQL naredbi.



Slika 3. MySQL Workbench sučelje

3. REALIZACIJA SUSTAVA

Za kreiranje ovakvog sustava potrebno je kreirati bazu podataka i aplikaciju koja će upravljati svime. Aplikacija treba pružiti jednostavno sučelje za običnog korisnika i administratora. Zbog toga što sustav treba moći pratit dolaznost i zaposlenika, nazivi korišteni trebaju biti općeniti kao i dizajn aplikacije.

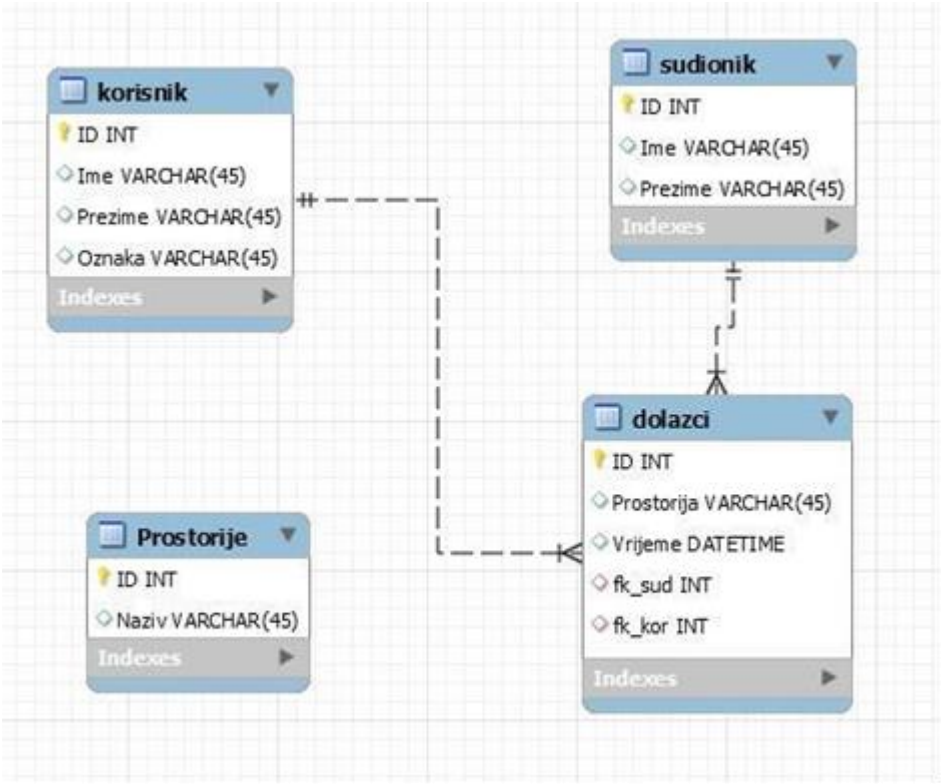
Korisnik sustava prvotno treba priključiti RFID uređaj u računalo. Nakon što se uređaj ispravno instalira, korisnik pokreće aplikaciju. Prilikom pokretanja aplikacije, korisniku će biti prikazan tekst koji će tražiti od njega da provuče svoju identifikacijsku karticu kako bi se prijavio. Nakon prijave korisnik ovisno o svojoj oznaci dobiva drugačije sučelje. Obični korisnici imaju mogućnost pregleda upisanih podataka pomoću pogleda na bazu podataka. Isto tako mogu unositi nove dolaske sudionika. Unos novih dolazaka pokreće se sa tipkom „Započni“ nakon čega svi prisutni sudionici provlače svoje identifikacijske kartice. Nadalje korisnik bira neku od ponuđenih prostorija te postavlja odgovarajuće vrijeme događaja. Zatim pritiskom na tipku

„Upload“ svi uneseni podatci se ažuriraju u bazi te ovisno o potrebama korisnika može ažurirati i tablicu na vlastitom Google disku. Nakon unosa korisnik može unositi nove podatke ili se odjaviti iz aplikacije.

Administrator sustava sa svim mogućnostima korisnika može i upravljati podatcima. Zbog toga ima prošireno sučelje sa dodatnim opcijama. Administrator aplikacije može dodavati i brisati prostorije, korisnike i sudionike. Isto tako ima mogućnost promjene pojedinog obilježja korisnika ili sudionika. Tako u slučaju nekakve pogreške prilikom unošenja moguće je promijeniti samo ime ili prezime određenog korisnika. Ostale detalje rada aplikacije moguće je pogledati sljedećim poglavljima.

3.1. Izrada baze podataka

Prvo se definiraju elementi baze podataka. Korisnici se smatraju osobama koji koriste ovaj sustav dok su sudionici sve osobe kojima se prati dolaznost. Tablicu korisnika i sudionika povezivat će tablica dolazaka koja će isto tako sadržavati vrijeme i prostoriju tog događaja. Korisnici i sudionici imaju jednake elemente tablice(identifikacijski broj, ime i prezime) dok korisnici imaju dodatni element koji definira dali je korisnik administrator ili ne. Tablica dolazaka se sastoji od stranih ključeva koji su povezani sa identifikacijskim brojevima korisnika i sudionika. Tako je količina podataka unutar baze smanjena te je pretraživanje jednostavnije. Također napravljena je tablica sa listom prostorija kako bi administrator mogao upravljati sa unesenim prostorijama. Relacijski dijagram tablice prikazan je na slici 4.



Slika 4. ER dijagram baze podataka

3.2. Izrada aplikacije

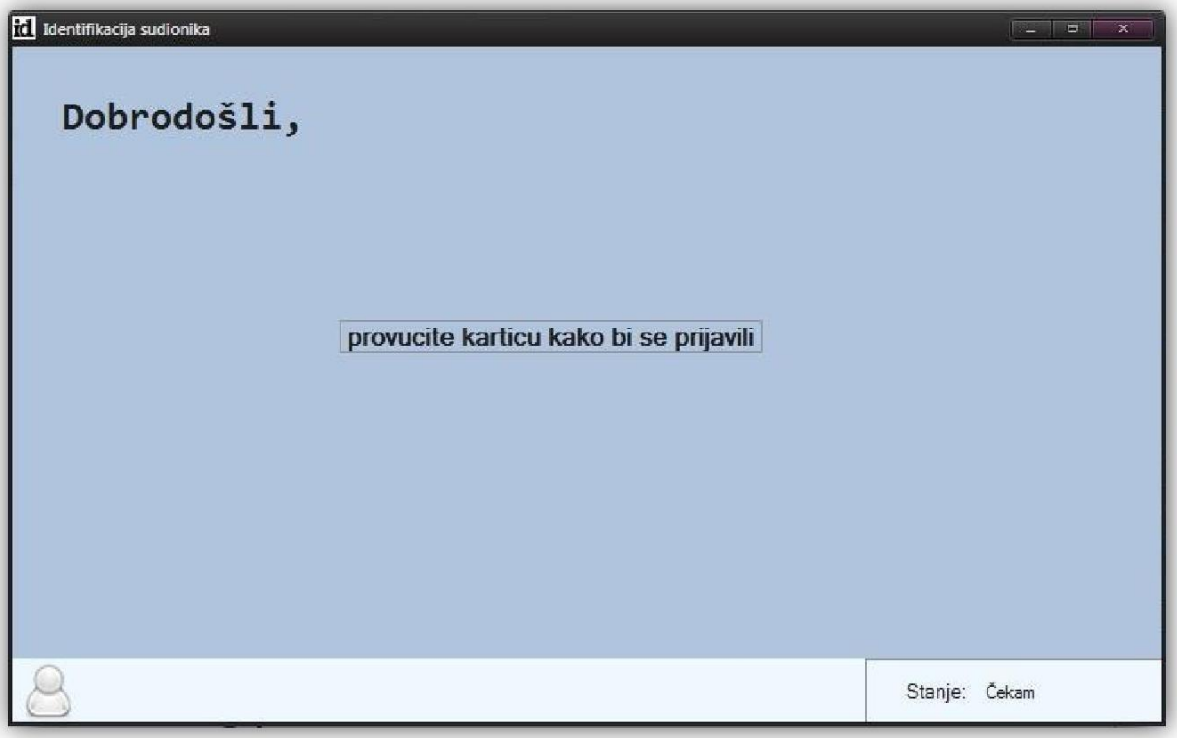
Aplikacije će se sastojati od tri glavna sučelja pa se analogno tome može podijeliti na tri dijela izrade. Prvo sučelje je početno sučelje koje će samo tražiti od korisnika da učita svoju karticu. Također prilikom izrade ovog sučelja definiraju se dimenzije i dizajn aplikacije. Statusna traka se dodaje kako bi se korisniku dala povratna veza o događajima i kako bi korisnik dobio opciju odjave iz aplikacije. Drugo sučelje dolazi poslije prijave korisnika i pojavljuje se bez obzira na vrstu korisnika. Služi za kontrolu pristupa sudionika kao i za prikaz liste korisnika, sudionika i osnovni prikaz dolazaka. Treće sučelje dobivaju kao opciju jedino korisnici sa oznakom administratora te tu imaju mogućnost unosa i izmjene podataka. Sučelje je podijeljeno na tri dijela: prvo za upravljanje korisnicima, drugo za upravljanje sudionicima i treće za upravljanje prostorijama.

3.2.1. Izrada početnog sučelja

Pri stvaranju ovakvog sučelja potrebno je postaviti samo nekakvu oznaku korisniku kako bi znao što se traži od njega. Isto tako pravi se osnovni dizajn i dimenzije aplikacije pošto je prvo sučelje. Kao što je prikazano na slici 5 od korisnika se traži učitavanje kartice kako bi se prijavio. Prijava se obavlja na način da se postavlja upit na bazu podataka za listom svih korisnika. Postavljanje upita prikazano je na sljedećem kodu.

```
MySql.Data.MySqlClient.MySqlConnection conn; string myConnectionString;
myConnectionString = "server=localhost;user id=root;" +
"pwd=asd123;database=idproject;";
try
{
    conn = new MySql.Data.MySqlClient.MySqlConnection();
    conn.ConnectionString = myConnectionString;
    conn.Open();
    string stm = "SELECT * FROM korisnik WHERE ID='" + input.ToString() + "'";
    MySqlDataAdapter mda = new MySqlDataAdapter(stm, conn);
    System.Data.DataTable dt = new System.Data.DataTable();
    mda.Fill(dt);
    tipkor = dt.Rows[0][3].ToString();
    label4.Text = dt.Rows[0][3].ToString() + ", " + dt.Rows[0][1].ToString() + " " +
dt.Rows[0][2].ToString();
    conn.Close();
}
catch (MySql.Data.MySqlClient.MySqlException ex)
{
    MessageBox.Show(ex.Message);
}
```

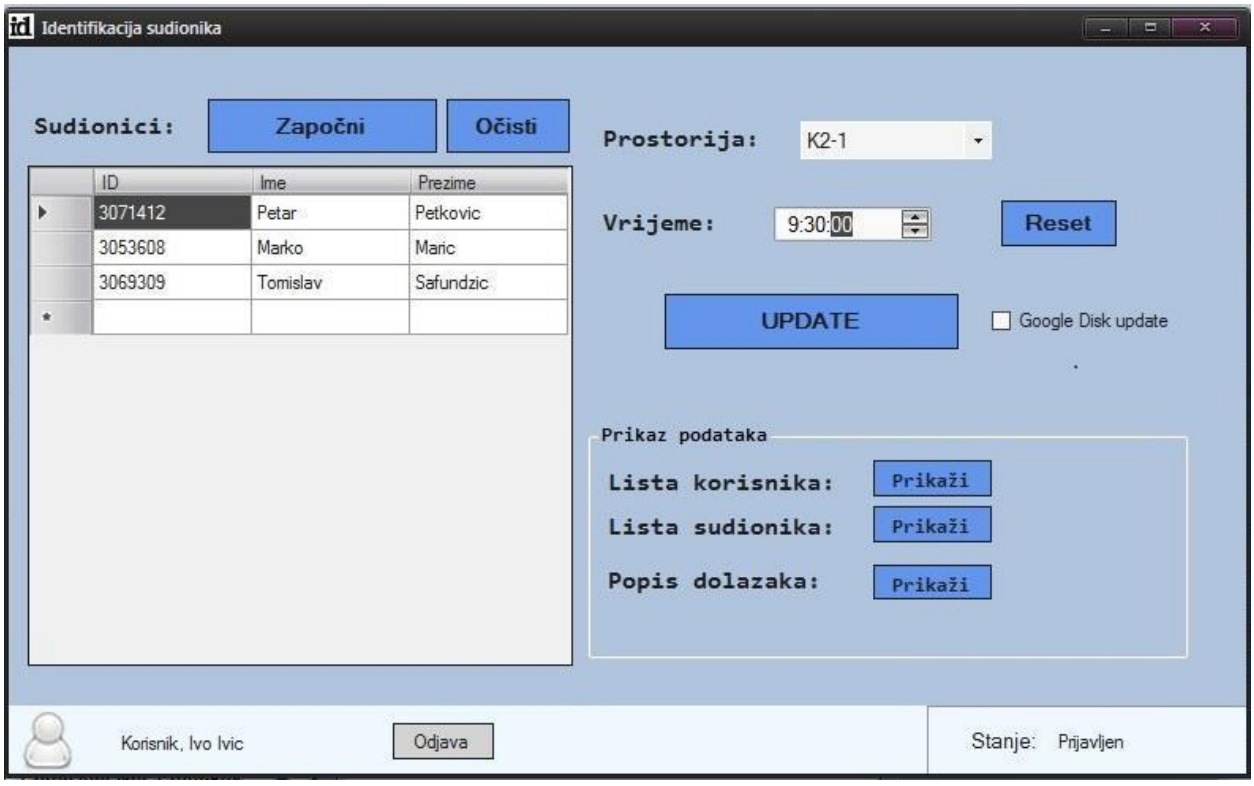
Prije bilo kakvog rada za MySQL bazama potrebno je uključiti MySQL biblioteku. Nakon uključivanja biblioteke potrebno je kreirati znakovni niz(eng. string) koji će sadržavati IP adresu servera, naziv korisnika i bazu podataka na koju se spajamo(ovdje je korištena i šifra što ne mora biti obavezno). Zatim kreira se objekt iz MySQL biblioteke te njegov string za konekciju mijenjamo sa prethodno stvorenim string-om i pozivamo funkciju za otvaranje veze. Ako se veza uspješno uspostavila stvara se string koji će predstavljati upit na bazu. Takav upit zatražit će entitet sa ID-om provučene kartice. Nadalje kreiraju se dvije vrste objekta, jedan skuplja sve podatke koje baza vrati na određeni upit(„DataAdapter“) dok je drugi predstavlja tablicu podataka. „DataAdapter“ objekt prilikom konstrukcije prima dva parametra, string upita i objekt veze. Najveći razlog korištenja „DataAdapter“ objekata je mogućnost povezivanja podataka baze i objekata ovog okruženja što se obavlja sa metodom „Fill“ koja ispunjava tablicu sa podatcima baze. Nakon uspješne komunikacije potrebno je samo korisnika obavijestiti o uspješnoj prijavi. Ovakav način komunikacije sa bazom, pošto daje točne podatke, će se koristiti kroz cijelu izradu aplikacije te više neće biti objašnjavana.



Slika 5. Izgled početnog zaslona aplikacije

3.2.2. Izgled glavnog sučelja

Nakon prijave korisnika automatski se pojavljuje sučelje za kontrolu pristupa. Samo mijenjanje sučelja u ovoj aplikaciji obavlja se promjenom svojstva vidljivosti prikazanih objekata. Pri prijavi običnog korisnika pojavljuje se sučelje prikazano na slici 6. Korisnik dobiva jednostavno sučelje pomoću kojeg može odmah unositi nove podatke o dolaznosti sudionika. Pritiskom na tipku „Započni“ aktivira se dok koda s kojim se unose sudionici u tablicu ispod. Unos sudionika pomoću RFID čitača unutar aplikacije je objašnjen u sljedećem poglavlju. Nakon unesenih sudionika korisnik bira prostoriju i vrijeme te ima opciju ažuriranja tablice na osobnom Google disku. Lista prostorija se automatski ažurira sa baze podataka prilikom prijavljivanja korisnika.



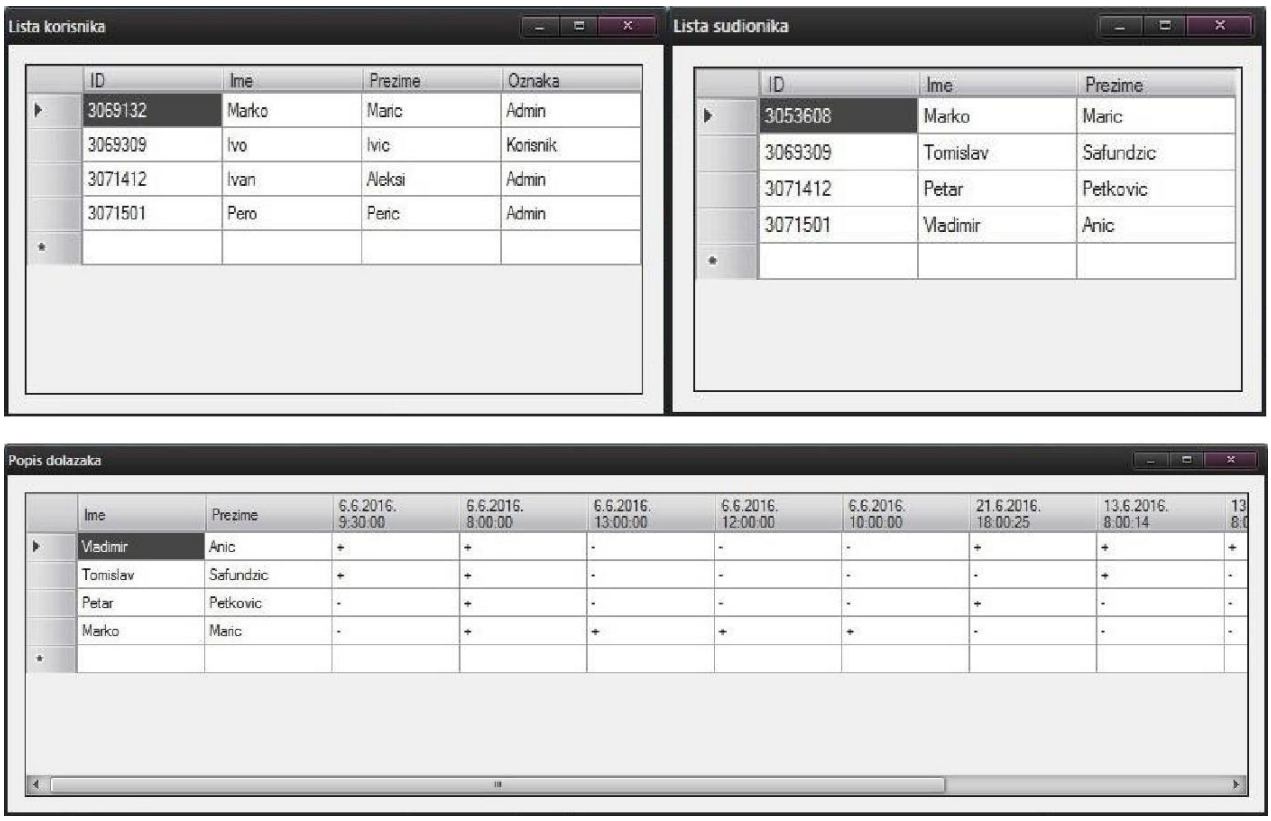
Slika 6. Korisničko sučelje aplikacije

Prilikom pritiska na tipku „update“ pokreće se petlja koja stvara string sa svakim korisnikom te otvara konekciju za bazom i pokreće ih kao naredbe. Sljedeći kod prikazuje string koji će biti pokrenut. Koristi se MySQL naredba za unos podataka sa vrijednostima koji se uzimaju iz

```
string Query = "insert into dolazci(prostorija, vrijeme, fk_kor, fk_sud)
values( '" + listaprostorija.SelectedItem.ToString() + "',
        '" + dateTimePicker1.Value.ToString() + "',
        '" + idkoris + "',
        '" + dataGridView1.Rows[gridbrojac].Cells[0].Value.ToString() + ');";
```

pojedinog elementa aplikacije. Za izvršenje naredbe koristi se „Reader“ objekt koji se inače koristi za spremanje podataka vraćenih od baze. Također stvara se objekt „Command“ koji prilikom konstrukcije kao parametre prima string konekcije koji je isti kao i u prethodnim poglavlju i string sa naredbom koji je sada stvoren.

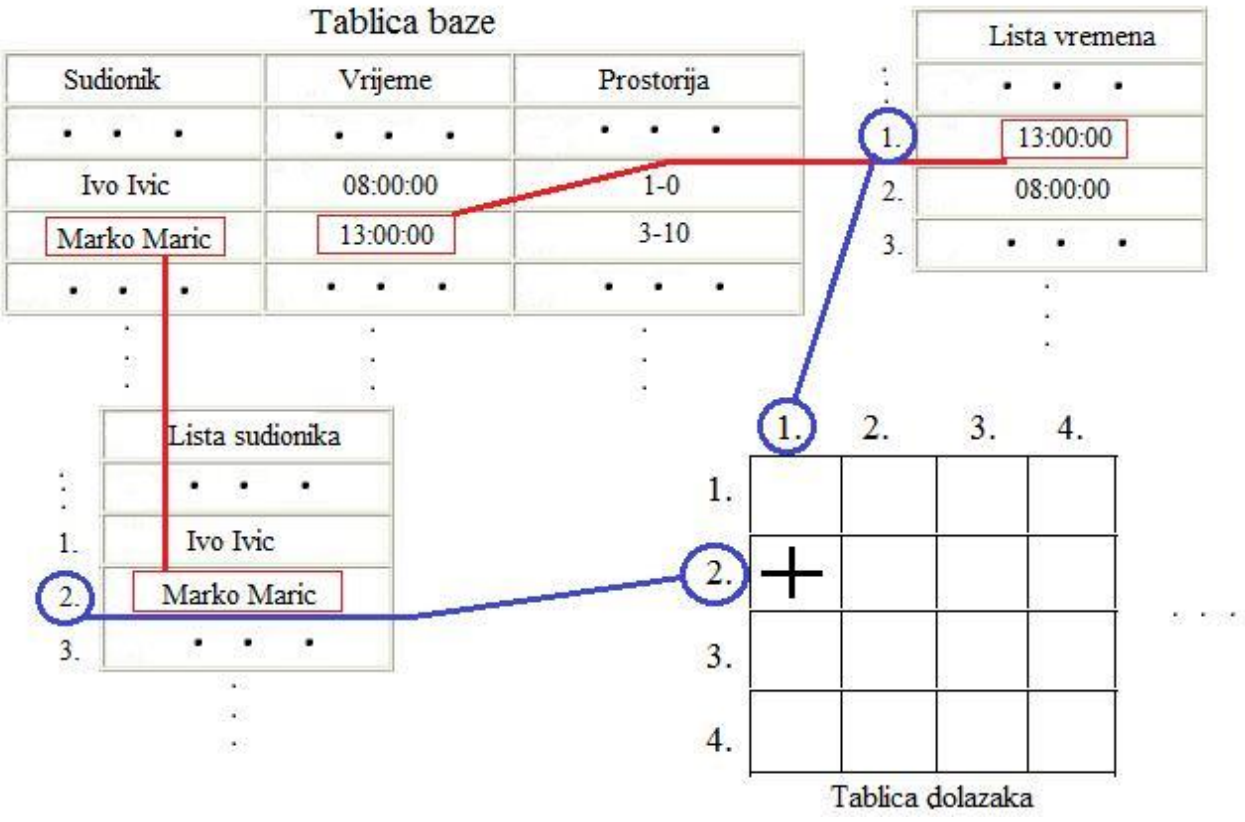
Kao što je već navedeno ovo sučelje sadrži dio koji korisniku omogućuje prikaz podataka iz baze. Ovdje obični korisnik može vidjeti listu korisnika i sudionika te listu dolaznosti pojedinih sudionika. Slika 7 prikazuje izgled tablica koje se stvaraju u trenutku kada korisnik stisne na jednu od tipaka iz grupe Prikaz podataka. Tipke za stvaranje prozora sa tablicom koriste metodu „ShowDialog“ pomoću koje se otvara forma ili prozor za stvoreni objekt.



Slika 7. Prikaz podataka

Najkompliciraniji dio se može smatrati dio kreiranja tablice dolazaka koja se može vidjeti na slici 7. Za tu tablicu potrebno je sva moguća vremena dolazaka studenata poredati u stupce iako se oni u bazi nalaze u redcima zbog načina na kojeg se evidentiraju. Cijeli kod stvaranja tablice nalazi se u prilogu 1. U početku se stvara konekcija koja zahtjeva sve podatke o dolascima iz baze. Nakon toga kreirane su 3 liste string-ova. Prve dvije liste sadržavaju imena i prezimena sudionika dok treća sva moguća vremena dolazaka. Ovdje nam dobro dolazi funkcija koju liste podržavaju, a to je „Exists“. Ta funkcija provjerava dali takav element unutar liste postoji te ako

postoji element neće biti unesen. Zatim se generira dvodimenzionalno polje koje ima redova koliko ima i sudionika i stupaca koliko ima i mogućih vremena. Svaka vrijednost polja postavlja se u „-“(minus) što predstavlja početnu vrijednost. Nakon toga treba provjeravati dali je sudionik bio u tom vremenu na predavanju i ako je vrijednost polja se mijenja u „+“. Sama provjera će se obavljati pomoću trostruke petlje. Prva petlja prolazi kroz svaki red tablice stvorene na početku prilikom uzimanja podataka iz baze. Nakon toga prolazi s kroz svaki element liste vremena i sudionika. Na mjestu gdje se vrijeme iz liste vremena podudara sa vremenom iz tablice baze i gdje se sudionik iz liste sudionika podudara sa sudionikom iz tablice, u prethodno kreiranom dvodimenzionalnom polju se postavlja „+“. Način stvaranja tablice dolazaka jasnije je pojašnjen na slici 8. Poslije stvaranja tablice dolazaka preostalo je samo ispisati sve podatke u konačnu tablicu. To je ostvareno sa dvije petlje gdje jedna postavlja vremena kao nazive stupaca, a druga nazive pojedinih sudionika sa njihovim dolascima. Ovaj način kreiranja tablice koristi se i prilikom kreiranja tablice za Google disk.



Slika 8. Izrada tablice dolazaka

3.2.3. Izrada upravljačkog sučelja

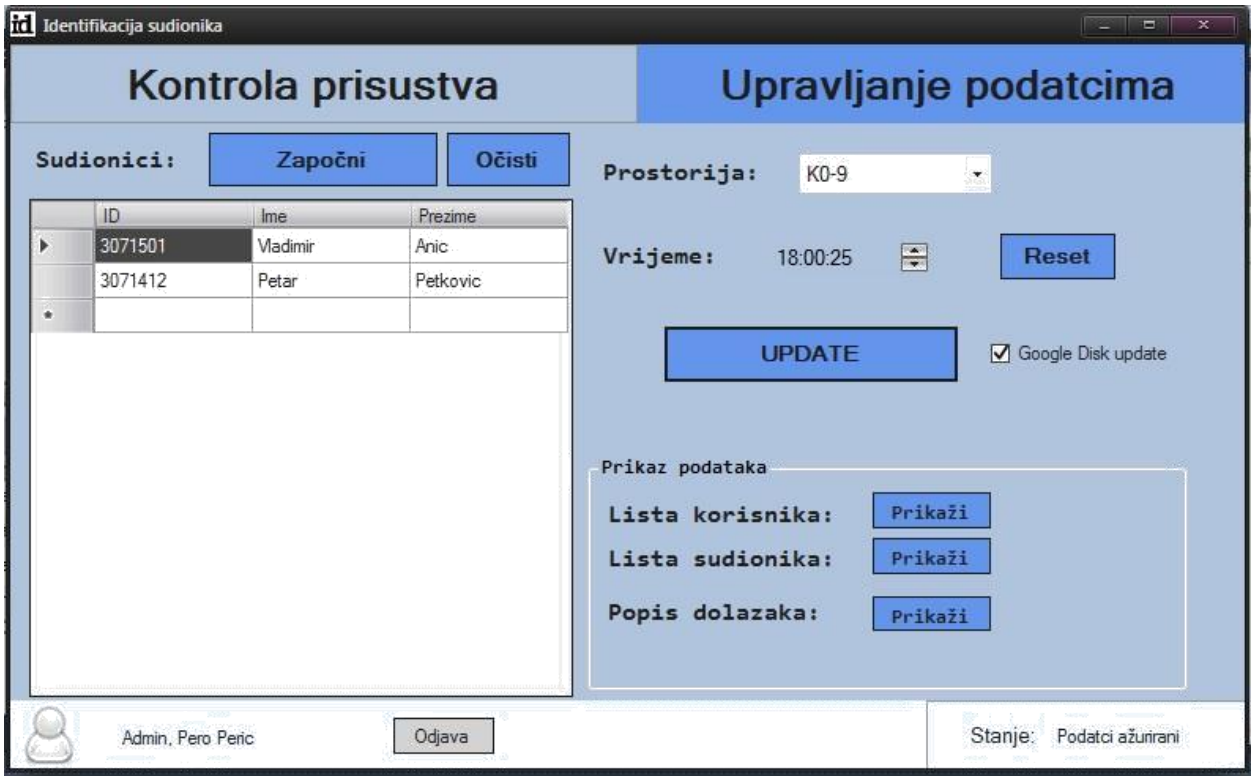
Pošto će sučelje koje će koristiti administrator imati također mogućnost upisa dolaznosti cijelo sučelje običnog korisnika se ostavlja. Stvara se dodatno sučelje za upravljanje podatcima. Zatim se postavljaju svi elementi koji će administratoru dati potpunu kontrolu nad upisanim podatcima. U ovom sučelju administrator može dodavati, izmijeniti i izbrisati korisnike, sudionike i prostorije. Ovaj dio programa za izradu nije kompliciran, ali je zato dosta vremenski zahtjevan. Sam proces izmijene podataka treba biti lako razumljiv i korisnik treba dobiti nekakvu povratnu informaciju kako bi bio siguran da su podatci stvarno izmijenjeni. Tu se to obavlja pomoću trake stanja gdje se ispisuju sve radnje aplikacije.

Za dodavanje korisnika administrator treba unesti ime, prezime i vrstu korisnika. Nakon toga kada pritisne tipku „Dodaj“ otvara se prozor koji će zatražiti ID broj korisnika. Tada je potrebno samo provući karticu novog korisnika. Ako je sve dobro obavljeno na traci stanja će se ispisati da je korisnik uspješno dodan. Sve radnje sa podatcima se obavljaju na isti način koji je već objašnjen. Mijenja se samo naredba koja će biti izvršena u MySQL okruženju. Nakon svake napravljene izmjene potrebno je osvježiti liste korisnika, sudionika i prostorija kako ne bi dolazilo do pogrešaka. To je ostvareno funkcijom koja ne prima ni ne vraća nikakve podatke te je sav njen kod prikazan u Prilogu 2.

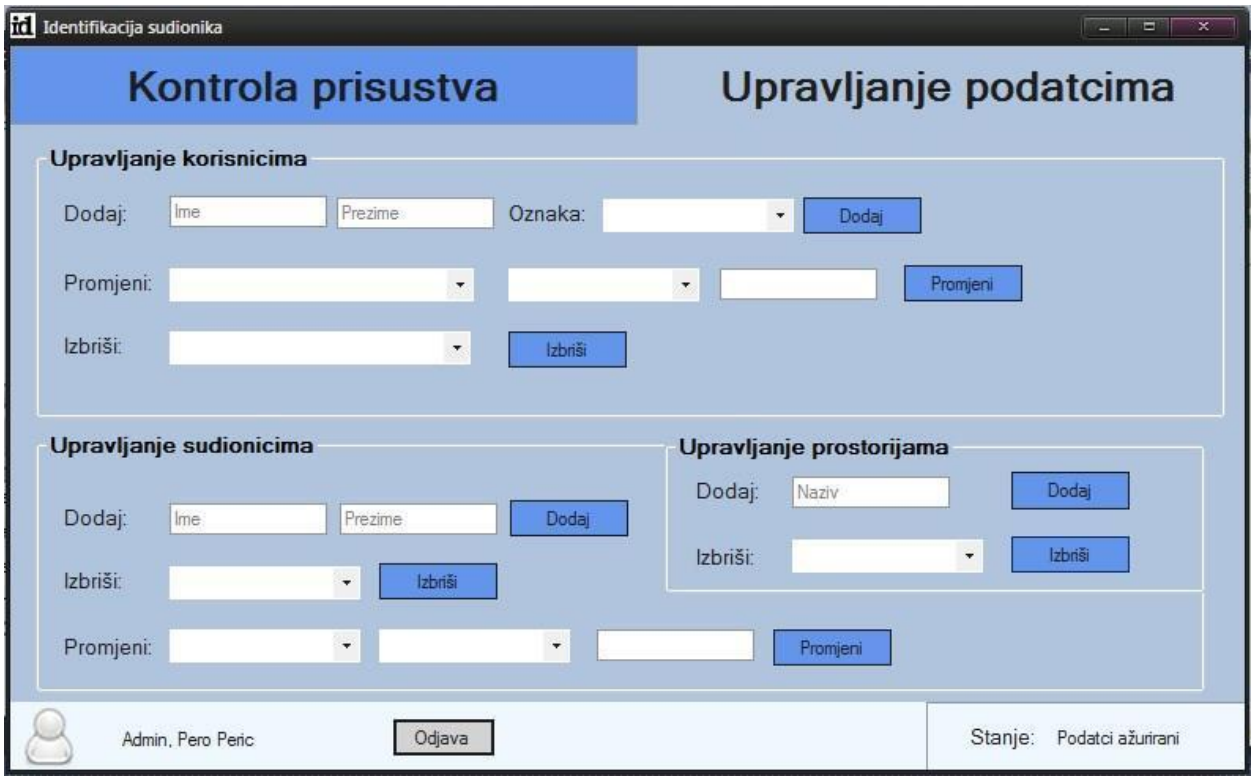
Primjer naredbe koja se koristi za dodavanje korisnika:

```
Query = "UPDATE korisnik SET Ime='" + textboximekor.Text + "',Prezime='" + textBoxprezkor.Text + "',tip='Admin' WHERE Ime='00000'; ";
```

Tijekom pisanja stringa naredbe treba paziti na sintaksu. String definiran u C# treba poprimiti ispravan oblik MySQL naredbe. Kako je već navedeno prilikom pritiska na tipku dodaj iskače prozor koji traži od korisnika da provuče karticu novog korisnika. Kada se provuče, u bazu se upisuje korisnik sa novim ID-om i za ime se stavlja '00000'. Nakon toga kada se vrati na početni prozor izvršava se ostali dio koda koji traži ime sa tim znamenkama i postavlja ispravne podatke.



Slika 9. Početno sučelje administratora



Slika 10. Sučelje za upravljanje podatcima

4. ZAKLJUČAK

U ovom radu izrađena je aplikacija za evidentiranje dolaznosti sudionika. Aplikacija koristi RFID uređaj za učitavanje kartica, a svi podatci se spremaju na bazu podataka. Sučelje aplikacije je jednostavno te sadržava sve potrebne funkcije za rad. Obični korisnici mogu vršiti evidenciju sudionika, a administratori mogu upravljati upisanim podatcima. Prednost ovog sustava je ta što ne zahtjeva instalaciju nikakvog dodatnog programa i cijena čitača kartica je vrlo niska. Mana sustava je nemogućnost detekcije različitih kartica odnosno čitač može detektirati samo kartice koje rade na frekvenciji od 125 kHz. Za detekciju drugih kartica moguće je proizvesti posebni sklop, ali to stvara dodatne troškove prilikom implementacije. Korištenjem ovog sustava ubrzava proces evidentiranja kada je broj sudionika jako velik što je pogodno za razne poslovne i obrazovne okolnosti.

LITERATURA

[1] Visual Studio Community razvojno okruženje, 27. lipanj 2016.

Link: https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx

[2] MySQL razvojno okruženje, 27. lipanj 2016.

Link: https://www.mysql.com/

[3] Naredbe i pomoću pri razvoju u SQL okruženju, 27. lipanj 2016.

Link: http://www.w3schools.com/sql/

[4] Osnovne metode i pomoć u C# okruženju, 27. lipanj 2016.

Link: https://msdn.microsoft.com/en-us/library/aa288436(v=vs.71).aspx

[5] Osnove programiranja C#-a sa primjerima, 27. lipanj 2016.

Link: http://www.tutorialspoint.com/csharp/

SAŽETAK

Naslov: Računalna aplikacija za RFID evidenciju studenata na nastavi

U ovom radu izrađena je aplikacija za evidenciju sudionika na nastavi. Aplikacija koristi RFID čitač koji detektira pametne kartice sudionika. Zatim aplikacija omogućuje korisniku da nadgleda upisane podatke te upisuje nove. Administratorima je dana mogućnost izmjene podataka te dodavanje i brisanje korisnika i sudionika. Svi podatci spremaju se na bazu podataka te je omogućen izvoz podataka u obliku tablice. Kroz rad je također objašnjena razvojna okruženja koja su korištena kao i uređaj za detekciju kartica. Isto tako opisani dijelovi koda koji su ključni za stvaranje ovakve aplikacije. Opisan je način na koji se stvara komunikacije između baze i aplikacije, način na koji se izvoze podatci, upravljanje podatcima i sl. Sustav ne zahtjeva dodatne program za rad, jeftin je za implementaciju i jednostavan za korištenje.

Ključne riječi: RFID, aplikacija, baza podataka, C#, Visual Studio, MySQL

ABSTRACT

Title: Computer application for recording student attendance using RFID technology

In this thesis an application for recording attendance of participants was developed. Application uses RFID reader which detects participants smart cards. Then application enables user to overlook written data and write new one. Administrators have an ability to modify data and add or delete users and participants. All data is stored onto data base and data export is enabled in form of table. Also, throughout the thesis, development environments were explained as well as the device used for card detection. System does not required no additional programs, its cheap for implementation and easy to use.

Keywords: RFID, application, database, C#, Visual Studio, MySQL

ŽIVOTOPIS

Luka Nuić rođen je 21. srpnja 1994. godine u Požegi. Od svog rođenja živi u mjestu Velika. U osnovnoj školi prvi put se susreće sa računarstvom te mu to postaje najveća zanimacija. U 2009. godini upisuje smjer računalnog tehničara u tehničkoj školi Požega. Tijekom srednje škole otkriva razne programske jezike kao i razvoj web stranica. Također je imao priliku sastavljanja jednostavnih sklopova tijekom vježbi. Stručnu praksu odrađuje u lokalnoj računalnoj prodavaonici gdje obavlja dijagnozu i popravak osobnih računala. Srednju školu završava redovno u 2013. godini i upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku.

PRILOZI

Ovdje se nalazi samo dio koda koji se koristi kao referenca kako bi se mogao bolje pratiti rad. Cijeli izvorni kod aplikacije nalazi se u CD priloženom uz ovaj rad.

Prilog 1. Kod forme za prikaz dolaznosti

```
public partial class Form4 : Form
{
    public Form4()
    {
        InitializeComponent();
    }

    private void Form4_Load(object sender, EventArgs e)
    {
        MySql.Data.MySqlClient.MySqlConnection conn1;
        string myConnectionString = "server=localhost;user id=root;" +
"pwd=asdf123;database=idproject;";

        try
        {
            conn1 = new MySql.Data.MySqlClient.MySqlConnection();
            conn1.ConnectionString = myConnectionString;
            conn1.Open();
            string stm = "select s.ime,s.prezime,d.vrijeme,d.prostorija from
sudionici as s,dolazci as d where s.id = fk_sud order by d.vrijeme DESC";
            MySqlDataAdapter mda = new MySqlDataAdapter(stm, conn1);
            System.Data.DataTable dt = new System.Data.DataTable();
            mda.Fill(dt);
            List<string> listaimena = new List<string>();
            listaimena.Add(dt.Rows[0][0].ToString());
            List<string> listaprezimena = new List<string>();
            listaprezimena.Add(dt.Rows[0][1].ToString());
            for (int i = 1; i < dt.Rows.Count; i++)
            {
                bool postoji = listaimena.Exists(element => element ==
dt.Rows[i][0].ToString());
                if (postoji == false)
                {
                    listaimena.Add(dt.Rows[i][0].ToString());
                }
            }
            for (int i = 1; i < dt.Rows.Count; i++)
            {
                bool postoji = listaprezimena.Exists(element => element ==
dt.Rows[i][1].ToString());
                if (postoji == false)
                {
                    listaprezimena.Add(dt.Rows[i][1].ToString());
                }
            }
            List<string> lista = new List<string>();
            lista.Add(dt.Rows[0][2].ToString());
            for (int i = 1; i < dt.Rows.Count; i++)
            {
```

```
                bool postoji = lista.Exists(element => element ==
dt.Rows[i][2].ToString());
                if (postoji == false)
                {
                    lista.Add(dt.Rows[i][2].ToString());
                }
            }
            string[,] dolazci = new string[listaimena.Count,lista.Count];
            for(int i = 0; i < listaimena.Count; i++)
            {
                for(int j = 0; j < lista.Count; j++)
                {
                    dolazci[i, j] = "-";
                }
            }
            for(int i = 0; i < dt.Rows.Count; i++)
            {
                for(int j = 0; j < lista.Count; j++)
                {
                    if (lista[j] == dt.Rows[i][2].ToString())
                    {
                        for(int k = 0; k < listaimena.Count; k++)
                        {
                            if (listaimena[k] == dt.Rows[i][0].ToString())
                            {
                                dolazci[k, j] = "+";
                            }
                        }
                    }
                }
            }

            dataGridView1.Columns.Add("Ime","Ime");
            dataGridView1.Columns.Add("Prezime", "Prezime");
            for(int j = 0; j < lista.Count; j++)
            {
                dataGridView1.Columns.Add(lista[j],lista[j]);
            }
            for(int i = 0; i < listaimena.Count; i++)
            {
                dataGridView1.Rows.Add(listaimena[i], listaprezimena[i]);
                for(int j = 0; j < lista.Count; j++)
                {
                    this.dataGridView1[j + 2, i].Value = dolazci[i,j];
                }
            }

            conn1.Close();
        }
        catch (MySql.Data.MySqlClient.MySqlException ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Prilog 2. Funkcija za ažuriranje upravljačkog sučelja

```
private void osvjeziuprav()
{
    MySql.Data.MySqlClient.MySqlConnection conn;
    string myConnectionString;
    myConnectionString = "server=localhost;user id=root;" +
"pwd=asdf123;database=idproject;";
    try
    {
        listsud.Clear();
        listpro.Clear();
        listkor.Clear();
        conn = new MySql.Data.MySqlClient.MySqlConnection();
        conn.ConnectionString = myConnectionString;
        conn.Open();
        string stmm = "SELECT * FROM korisnik";
        MySqlDataAdapter mdaa = new MySqlDataAdapter(stmm, conn);
        System.Data.DataTable dtt = new System.Data.DataTable();
        mdaa.Fill(dtt);
        comboizbkor.Items.Clear();
        comboprokor.Items.Clear();
        for (int i = 0; i < dtt.Rows.Count; i++)
        {
            listkor.Add(dtt.Rows[i][0].ToString());
            comboizbkor.Items.Add(dtt.Rows[i][1].ToString() + " " +
dtt.Rows[i][2].ToString() + ", " + dtt.Rows[i][3].ToString());
            comboprokor.Items.Add(dtt.Rows[i][1].ToString() + " " +
dtt.Rows[i][2].ToString() + ", " + dtt.Rows[i][3].ToString());
            var row = dtt.Rows[i];
        }
        string stmm1 = "SELECT * FROM prostorije";
        MySqlDataAdapter mdaa1 = new MySqlDataAdapter(stmm1, conn);
        System.Data.DataTable dtt1 = new System.Data.DataTable();
        mdaa1.Fill(dtt1);
        comboizbrisiprost.Items.Clear();
        for (int i = 0; i < dtt1.Rows.Count; i++)
        {
            listpro.Add(dtt1.Rows[i][1].ToString());
            comboizbrisiprost.Items.Add(dtt1.Rows[i][1].ToString());
            var row = dtt1.Rows[i];
        }
        string stmm2 = "SELECT * FROM sudionici";
        MySqlDataAdapter mdaa2 = new MySqlDataAdapter(stmm2, conn);
        System.Data.DataTable dtt2 = new System.Data.DataTable();
        mdaa2.Fill(dtt2);
        comboizbrsud.Items.Clear();
        comboBox1.Items.Clear();
        for (int i = 0; i < dtt2.Rows.Count; i++)
        {
            listsud.Add(dtt2.Rows[i][0].ToString());
            comboBox1.Items.Add(dtt2.Rows[i][1].ToString() + " " +
dtt2.Rows[i][2].ToString());
            comboizbrsud.Items.Add(dtt2.Rows[i][1].ToString() + " " +
dtt2.Rows[i][2].ToString());
            var row = dtt2.Rows[i];
        }


        conn.Close();
```

```
    }
    catch (MySql.Data.MySqlClient.MySqlException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```