UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Robert Kreslin

APLIKACIJA ZA UPRAVLJANJE RFID PAMETNIH KARTIC PRI KONTROLI PRISTOPA V HOTELU

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Igor Rožanc

Ljubljana, 2011

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00163/2011
Datum: 09.09.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ROBERT KRESLIN**

Naslov: **APLIKACIJA ZA UPRAVLJANJE RFID PAMETNIH KARTIC PRI KONTROLI PRISTOPA V HOTELU**

THE RFID SMART CARD MANAGEMENT APPLICATION FOR THE HOTEL ACCESS CONTROL

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Kontrola pristopa v objekte z uporabo pametnih kartic je sodobna rešitev, ki jo pogosto srečamo. V diplomski nalogi predstavite razvoj aplikacije, ki podpira uporabo pametnih RFID kartic za kontrolo pristopa v manjšem hotelu. V ta namen najprej predstavite različne tehnologije s tega področja, analizo uporabnikovih zahtev, izbiro najustreznejše rešitve ter sistematično izdelavo aplikacije, ki omogoča celovit nadzor pristopa v hotel.

Mentor:

viš. pred. dr. Igor Rožanc

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU
diplomskega dela

Spodaj podpisani Robert Kreslin, z vpisno številko 63040262, sem avtor diplomskega dela z naslovom:

APLIKACIJA ZA UPRAVLJANJE RFID PAMETNIH KARTIC PRI KONTROLI PRISTOPA V HOTELU

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., ang.), povzetek (slov., ang.) ter ključne besede (slov., ang.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 19. 09. 2011

Podpis avtorja:

ZAHVALA

Najprej bi se rad zahvalil dr. Igorju Rožancu za mentorstvo in pomoč pri izdelavi diplomske naloge.

Posebna zahvala gre trem, brez katerih študij na FRI-ju ne bi bil tak, kot je bil; Erika Pogorelc, Miha Vydra-Stančič in Peter Krebelj, hvala vam, da ste me v času študija prenašali, mi pomagali in me spodbujali. Zahvala tudi vsem ostalim sošolkam in sošolcem, profesorjem in asistentom, ki so mi kakorkoli pomagali pri študiju na Fakulteti za računalništvo in informatiko.

Hvala tudi mojim najbližjim, dekletu Doris za spodbudo pri pisanju; na koncu pa bi se rad spomnil tudi dedka, ki zaradi bolezni diplomske naloge ni dočakal.

KAZALO

SEZNAM KRATIC

.NET	Aplikacijsko ogrodje za izdelavo aplikacij.
ABC	Nekdanji sistem za avtomatsko cestninjenje na avtocestah.
ActiveX	Programsko ogrodje, ki zagotavlja uporabo programskih komponent neodvisno od programskega jezika.
C	Programski jezik.
C#	Programski jezik.
C++	Programski jezik.
CLR	Common Language Runtime - Programsko okolje.
DB	Database - podatkovna baza.
DNK	Kratica za molekulo deoksiribonukleinske kisline.
ID	Kratica za identifikacijo.
IDC70	Tip RFID kartice.
IDC80	Tip RFID kartice.
IDE	Integrated Development Environment - Integrirano razvojno orodje.
IDK50	Tip RFID kartice.
IDTECK	Znamka strojne opreme kontrole pristopa.
IMC125	Tip RFID kartice.
IP	Internetni protokol.
MDI	Multiple Document Interface Container - Večnamenski vmesnik za dokumente .
MS	Microsoft – Podjetje.
PIN	Personal Identification Number – Osebna identifikacijska številka.
RAD	Rapid Application Development – orodje za hitro izdelavo programov.
RF TINY	Tip RFID čitalca.
RF10	Tip RFID čitalca.
RF20	Tip RFID čitalca.
RF30	Tip RFID čitalca.

RFID	Radio Frequency Identification – radio frekvenčna identifikacija.
RS232	Recommended Standard 232 – standard za komunikacijo.
RS422	Recommended Standard 422 – standard za komunikacijo.
SDK	Software Development Kit – programsko razvojno orodje.
SQL	Structured Query Language – strukturni poizvedovalni jezik.
TCP	Transmission Control Protocol – protokol za nadzor prenosa.
VB	Programski jezik.

POVZETEK

V diplomskem delu smo želeli prikazati projekt, ki smo ga izdelali za manjši hotel v Novi Gorici. Naloga je bila izdelava aplikacije za upravljanje kontrole pristopa po želji naročnika ter vpeljava sistema v obstoječo infrastrukturo.

Najprej smo se seznanili s tem, kaj pravzaprav je kontrola pristopa. V grobem jo delimo na RFID – radiofrekvenčno identifikacijo ter biometrično identifikacijo. Vsaka posebej ima svoje prednosti in slabosti. Za izdelavo aplikacije za upravljanje kontrole pristopa smo nato izbrali primerna orodja za razvoj. Odločili smo se za Microsoft Visual Studio 2008 ter ogrodje Microsoft .NET, pri katerem smo uporabili programski jezik C#. Analizirali smo zahteve našega naročnika, definirali funkcionalne, nefunkcionalne zahteve ter pripravili podatkovno bazo. Največ dela je seveda prinesel razvoj aplikacije, spoznavanje strojne opreme ter komunikacija z njo. Za kontrolo pristopa smo izbrali strojno opremo proizvajalca IDTECK, saj so že dobrih 20 let vodilni na področju izdelave in razvoja za kontrolo pristopa. Njihovi produkti so cenovno ugodni, hkrati pa razvijalcem programske opreme ponujajo programsko razvojno orodje SDK, s katerim lahko razvijalci programske rešitve nadgradijo po želji naročnika.

Z diplomskim delom je tako nastal pregled razvoja aplikacije za upravljanje pametnih RFID kartic, ki je bil razvit po željah naročnika. Izdelali smo aplikacijo, ki poleg kontrole pristopa omogoča tudi pregled nad uporabniki oziroma gosti hotela, ter možnost avtomatskega onemogočanja uporabnikov po določenem datumu, kar sama strojna oprema kontrole pristopa brez programske rešitve ne omogoča. Skozi raziskovanje različnih možnosti smo spoznali precej podrobnosti v zvezi s kontrolo pristopa, veliko pa smo se naučili tudi o razvoju aplikacij v orodju Microsoft Visual Studio. Skozi razvoj smo ugotovili, da obstaja veliko možnosti, ki bi jih v povezavi s kontrolo pristopa lahko realizirali, od statistike prisotnosti do urnikov delovnega časa ter s tem avtomatskega obračuna delovnih ur.

Ključne besede:
Kontrola pristopa
RFID – radiofrekvenčna identifikacija
Biometrična identifikacija
Nadzor
IDTECK

ABSTRACT

In this thesis we wanted to present the project that was made for a smaller hotel in Nova Gorica. The goal was to create an application for managing access control according to customer's wishes as well as to introduce the system into the existent infrastructure.

The first step was to define what access control actually means. In broad terms it is divided into RFID – radio-frequency identification and biometric identification. Both have their strengths and their weaknesses. Next step was choosing the right tools to develop the application for managing access control. We picked Microsoft Visual Studio 2008 and Microsoft .NET framework using C# programming language. We analyzed the customer's requirements, defined the functional and non-functional requirements and set up a database. The most demanding part was of course developing the application, studying the hardware and how to communicate with it. We chose the hardware created by IDTECK, which has been at the forefront of manufacturing and developing access control hardware for more than 20 years. Their products are moderately priced and they provide software developers with the SDK development kit that allows the developers to upgrade their solutions according to customer's wishes.

The thesis thus offers an overview of the development of an application for managing smart RFID card, made to suit the customer's wishes. We created an application that offers more than access control, enabling an overview of its users i.e. hotel guests as well as automatically disabling the users after a set date, which is something that access control hardware does not offer without a software solution. While researching various options we learned many details of access control and learned a lot about developing applications in Microsoft Visual Studio environment. In development we learned that there are many various other useful functions that access control allows, from calculating presence to timetables of working time and along with that an automatic calculation of the working hours.

Keywords:
Access control
RFID – radio-frequency identification
Biometric identification
Control
IDTECK

1 UVOD

Danes se vsako podjetje sooča s težavami, kako zavarovati svoje prostore, poslovne skrivnosti in osebje pred nepooblaščenimi osebami. Večje je podjetje, večji postaja omenjeni problem, saj se osebje in oddelki med seboj ne poznajo. V ta namen je večina podjetij v svoje vrste zaposlila receptorje – varnostnike, ki opravljajo nadzor nad osebjem in prostori. Veliko težavo pa v takih sistemih povzroča vstop v prostore. Ti so po navadi zaklenjeni s ključavnicami, za katere morajo zaposleni imeti ključe, ki se hitro izgubijo, njihova slaba lastnost pa je tudi ta, da lahko na enostaven način pridemo do dvojnika (kopije). Hkrati se pojavlja tudi problem, na kakšen način lahko nadziramo osebje in njihovo prisotnost oziroma delovne ure. V nekaterih tovarnah še danes uporabljajo posebne kartončke, s pomočjo katerih zaposlenim beležijo prihod in odhod z dela.

Takšne in podobne težave so pripeljale do razvoja nove veje tehnologije – kontrole pristopa (*ang. access control*). Ta odpravlja glavne pomanjkljivosti starih načinov delovanja, na daljši rok pa tudi zmanjšuje stroške podjetjem. Kontrola pristopa je še posebej pisana na kožo hotelom, kjer se na dan izmenja ogromno število gostov.

2 KONTROLA PRISTOPA

Kontrola pristopa (*ang. access control*) je besedna zveza, ki jo zadnje čase vse večkrat slišimo predvsem v poslovnem, pa tudi v civilnem svetu. Kljub popularnosti veliko ljudi ne ve, kaj izraz pravzaprav pomeni. Kontrola pristopa označuje strojno in programsko opremo, ki skupaj omogočata ali onemogočata fizični dostop do različnih objektov in prostorov v določenih časovnih intervalih. Če bi pojem želeli poenostaviti bi lahko rekli, da je kontrola pristopa nekakšen računalniški varnostnik oziroma ključar.

Varovanje prostorov s kontrolo pristopa se je razvilo iz službe varnostnika, ključev in ključavnic. Včasih smo dostop v objekte varovali s pomočjo ključavnic in ključev, varnostnih služb ter varnostnikov, ki so na podlagi identifikacije, pisnega seznama obiskovalcev za določen dan ali vizualnega prepoznavanja zaposlenih osebi dovolili ali zavrnili vstop v objekt. Nekateri problemi takšnega varovanja so:

- 24 urni nadzor varnostnika, kar predstavlja velik finančni zalogaj (za 24 urni nadzor potrebujemo 4 osebe za 8 urni delovnik)

- nenatančen vpogled prisotnosti zaposlenih in obiskovalcev (nimamo natančnega vpogleda v to, kdaj je kdo prisoten)
- nezmožnost varnostnika biti hkrati na več lokacijah (sprejemati gosta na recepciji ter odpirati vrata arhiva v prvem nadstropju)

Kontrolo pristopa v večini primerov sestavljajo električni prijemniki (*ang. door locks*), kontrolne enote (*ang. access controller*), identifikacijski čitalci (*ang. ID readers*) ter programska oprema (*ang. software*). Kontrolo pristopa v glavnem zanimajo tri glavna vprašanja:

- KDO?
- KJE?
- KDAJ?

Na podlagi teh vprašanj se omogoči ali zavrne dostop do objekta ali prostora. Na vprašanje KDO in KJE odgovarjajo identifikacijski čitalci glede na svojo lokacijo ter podatke, ki jih prejmejo in posredujejo kontrolni enoti. Najbolj razširjeni so čitalci pametnih kartic (*ang. smart card readers*) in biometrični čitalci (*ang. biometric readers*). Na vprašanje KDAJ pa odgovarjajo kontrolne enote s pomočjo programske opreme.

2.1 IDENTIFIKACIJA

Na vprašanje KDO pri kontroli pristopa odgovarjamo z identifikacijo (*ang. identification*). Identifikacija uporabnika je najpomembnejši odgovor kontroli pristopa. Identifikacija pomeni, da osebo identificiramo, razberemo oziroma ugotovimo, kdo je. To lahko ugotovimo na tri načine:

- nekaj, kar oseba ve/pozna, na primer PIN koda (*ang. PIN number*)
- nekaj, kar oseba ima, na primer pametno kartico (*ang. smart card*)
- nekaj, kar oseba je, na primer prstni odtis (*ang. fingerprint*)

Dolgo časa je veljalo, da je prstni odtis najvarnejša oblika identifikacije, vendar se izkazalo, da ga je moč dokaj enostavno ponarediti, zato je še vedno ena izmed najbolj varnih in tudi najbolj razširjenih oblik identifikacije t.i. RFID pametna kartica (*ang. RFID smart card*). [1]

2.1.1 RFID – RADIOFREKVENČNA IDENTIFIKACIJA

Kratica RFID (*ang. Radio Frequency Identification*) označuje tehnologijo, ki omogoča prenos podatkov med čitalcem in elektronsko oznako (pametno kartico), ki jo sestavlja integrirano vezje, ki hrani in procesira podatke. RFID lahko deluje na različnih frekvencah, najbolj pogoste so nizkofrekvenčne od 125 – 134.2 kHz. [2] [3]

Slika 1: Primer RFID elektronske oznake

RFID elektronske oznake se uporabljajo v različne namene, najbolj razširjeno je označevanje živali, uporaba v namene kontrole pristopa, sledenje vozil itd. Eden od primerov je označevanje izdelkov v trgovini, kot prikazuje slika 1. Za namene kontrole pristopa pa se običajno uporablja RFID elektronske oznake velikosti kreditne kartice, kot prikazuje slika 2.

Slika 2: Primer RFID kartice za kontrolo pristopa [4]

2.1.1.1 AKTIVNI RFID

Aktivni RFID oddajnik (*ang. active RFID*) za svoje delovanje potrebuje anteno, čip in baterijo, ki napaja vezje oddajnika. Aktivni RFID je prav zaradi baterije večji in dražji, ima pa nekaj prednosti:

- večjo oddajno moč,
- daljši domet,
- zanesljivejše delovanje v neidealnih pogojih (bližina kovin, tekočin).

Primer aktivne RFID kartice je stari sistem cestninjenja v Sloveniji, poimenovan ABC. [2]

2.1.1.2 PASIVNI RFID

Pasivni RFID (*ang. passive RFID*) za svoje delovanje energijo prejme od signala, ki se inducira v anteni, zato za svoje delovanje ne potrebuje baterije, kar občutno zmanjša ceno in velikost. Sprejet izmenični signal se usmeri in dovede do čipa, ki se vzbudi in prične delovati. V primerjavi z aktivnim oddajnikom je slaba lastnost pasivnega predvsem v tem, da ima veliko krajši domet in manjšo odpornost na napake. Pasivni RFID je najpogosteje uporabljen pri označevanju živali in pri uporabi kontrole pristopa. [2]

2.1.1.3 INDUKTIVNI RFID

Induktivni RFID sistem za prenos informacije uporablja princip magnetne indukcije. Dve bližnji tuljavi sta induktivno sklopljeni, ko magnetni pretok, ki ga povzroča tok prve tuljave, doseže drugo tuljavo in na njenih priključnih sponkah inducira napetost. Komunikacija deluje v bližnjem polju (približno 0.16 valovne dolžine), zato je domet induktivnih sistemov razreda nekaj deset centimetrov in pada z naraščanjem frekvence (krajšanje valovne dolžine). Oddajnik informacijo prenese čitalcu prek uporabe bremenske modulacije (*angl. load modulation*), ki v praksi pomeni spreminjanje sklopnega faktorja med tuljavama v ritmu podatkov. [2]

2.1.1.4 ELEKTROMAGNETNI RFID

Elektromagnetni RFID (*ang. electromagnetic RFID*) sistem komunicira z uporabo elektromagnetnih valov. Čitalec oddaja elektromagnetne valove, ki dosežejo oddajnik in se od

njega odbijejo. Ta odboj lahko izkoristimo za prenos informacije od oddajnika do čitalca. V trenutku, ko se oddajnik vzbudi, začne svojo lastno impedanco spreminjati v ritmu podatkov ter na ta način spreminja svojo oscilacijsko frekvenco. Signal, ki se odbije, je zato moduliran, celoten pojav pa imenujemo modulacijski odboj. [2]

2.1.2 BIOMETRIČNA IDENTIFIKACIJA

Biometrija (*ang. biometrics*) je veda o načinih prepoznave ljudi na podlagi njihovih telesnih, fizioloških ter vedenjskih značilnosti, ki jih imajo vsi posamezniki. So edinstvene in stalne za vsakega posameznika posebej, z njimi je možno določiti posameznika, zlasti z uporabo prstnega odtisa, posnetka papilarnih linij s prsta, šarenice, očesne mrežnice, obraza, ušesa, DNK ter značilne drže. *[4]* Biometrične podatke različni čitalci berejo različno. Najpopularnejši so čitalci prstnih odtisov, ki jih čitalci berejo s pomočjo optičnih ali silikonskih bralnikov. *[5]*

2.1.2.1 ČLOVEŠKE ZNAČILNOSTI UPORABLJENE V BIOMETRIJI

Človeške značilnosti, ki jih biometrija uporablja za identifikacijo, delimo na dve vrsti. Prva vrsta so človekove fizične značilnosti:

- prstni odtis
- dlan
- podoba obraza
- šarenica
- očesna mrežnica
- uho
- preplet ven na roki
- vonj
- DNK

Druga vrsta pa so človekove vedenjske značilnosti:

- lastnoročno podpisovanje
- govor (glas)
- gibanje
- tipkanje

Pri vsem tem je potrebno poudariti, da niso vse značilnosti enako unikatne, ter da vseh naštetih značilnosti za potrebe identifikacije ne moremo uporabiti zgolj samostojno, ampak v kombinaciji z drugimi značilnostmi. Najbolj unikatni značilnosti sta DNK in očesna mrežnica, vendar tudi ti dve nista nujno absolutno unikatni. [4]

2.1.3 UPORABA BIOMETRIJE V SLOVENIJI

Vprašanje, ki se poraja marsikateremu podjetju je, ali lahko biometrijo uporablja za evidentiranje delovnega časa zaposlenih. Po določbi 80. člena ZVOP-1 se biometrijski ukrepi lahko izvajajo le, če so nujno potrebni za opravljanje dejavnosti, za varnost ljudi ali premoženja ali za varovanje tajnih podatkov ali poslovne skrivnosti. S takšno določbo je zakonodajalec sledil načelu sorazmernosti (3. člen ZVOP-1) in načelo konkretiziral glede obdelave posebne vrste osebnih podatkov, to je biometričnih podatkov, ter s tem omejil možnost prekomernih in neupravičenih posegov v zasebnost in dostojanstvo posameznika pri izvajanju biometrijskih ukrepov. Obstajati mora torej resnično upravičen razlog, ki terja, da je biometrično preverjanje oziroma ugotavljanje identitete nujno potrebno in da namena, ki ga upravljalec zasleduje, ni mogoče doseči zadovoljivo tudi z drugimi načini preverjanja oziroma ugotavljanja identitete, ki ne vključujejo posegov v zasebnost in dostojanstvo posameznika.

V praksi podjetja uvajajo biometrijske ukrepe za evidentiranje delovnega časa zgolj zato, ker je takšen način bodisi bolj praktičen od sistema z brezkontaktnimi karticami ali pa želijo preprečiti zlorabe s posojanjem kartic, pri čemer slednji razlog zgolj pavšalno navedejo kot razlog uvedbe, kar pa ne zadosti pogojem iz 80. člena ZVOP-1.

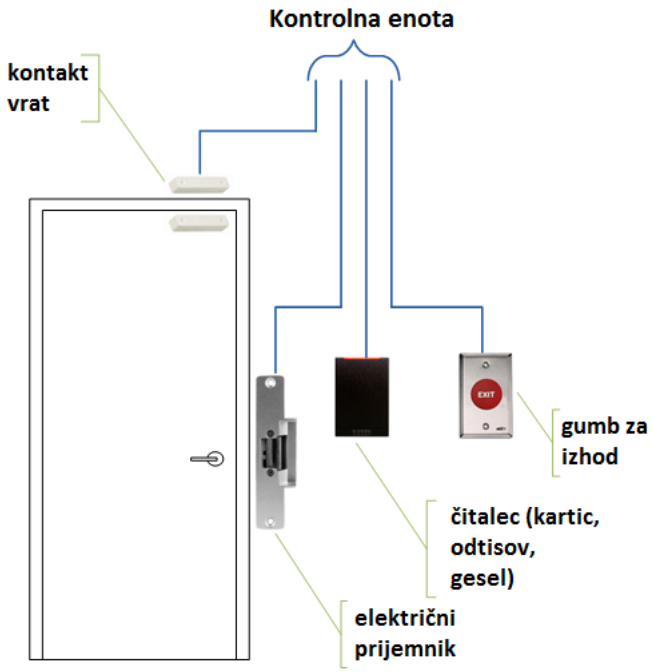
Na tem mestu je vredno omeniti še to, da četudi podjetje zadosti pogojem za uvedbo biometrične identifikacije, to lahko izvaja le nad zaposlenimi osebami. [4]

2.2 STROJNA OPREMA

Kontrola pristopa za svoje delovanje potrebuje strojno opremo (*ang. hardware*). Pod strojno opremo pojmujemo vse elemente, ki so za delovanje kontrole pristopa potrebni. Osnovni element vsake kontrole pristopa je **kontrolna enota** oziroma **kontroler** (*ang. access controler*). Njegova naloga je komuniciranje s čitalci, ki preverjajo identiteto, napajanje le-teh, preverjanje uporabnikov na podlagi hranjenih podatkov, posredovanih s strani uporabniških programov za nastavitev delovanja kontrole pristopa, ter odpiranje vrat (ključavnic).

Na kontrolno enoto so vezani **čitalci** oziroma čitalniki (*ang. access readers*), ki preverjajo identiteto. Najpogostejši so čitalci RFID kartic (*ang. RFID readers*), čitalci prstnih odtisov (*ang. fingerprint readers*), čitalci očesne mrežnice (*ang. iris readers*) in drugi.

Na kontrolno enoto imamo lahko opcijsko povezane tudi senzorje, ki preverjajo status odpiranja/zapiranja vrat.
Pri vsakem sistemu kontrole pristopa pa potrebujemo tudi naprave, ki fizično omogočajo vstop v varovane prostore. V večini primerov so to električne ključavnice, bolj pravilno **električni prijemniki** (*ang. electric door locks*), električna vrata ali druge naprave.



Slika 3: Primer uporabljene strojne opreme za kontrolo pristopa [7]

Tipične komponente strojne opreme (čitalec, električni prijemnik, kontakt vrat ter gumb za izhod), uporabljene pri kontroli pristopa, prikazuje slika 3. Gumb za izhod se uporablja v primerih, kjer nimamo čitalcev na obeh straneh vrat, kot je na primer garažna hiša, odpiranje vrat obiskovalcem objekta ali vstop v prostor znotraj objekta, kjer je predhodno že bila opravljena identifikacija uporabnika.

3 IMPLEMENTACIJA KONTROLE PRISTOPA V HOTELU

Do tretjega poglavja smo se podrobneje seznanili s kontrolo pristopa ter načini identifikacije, v tretjem poglavju pa začenjamo z implementacijo.

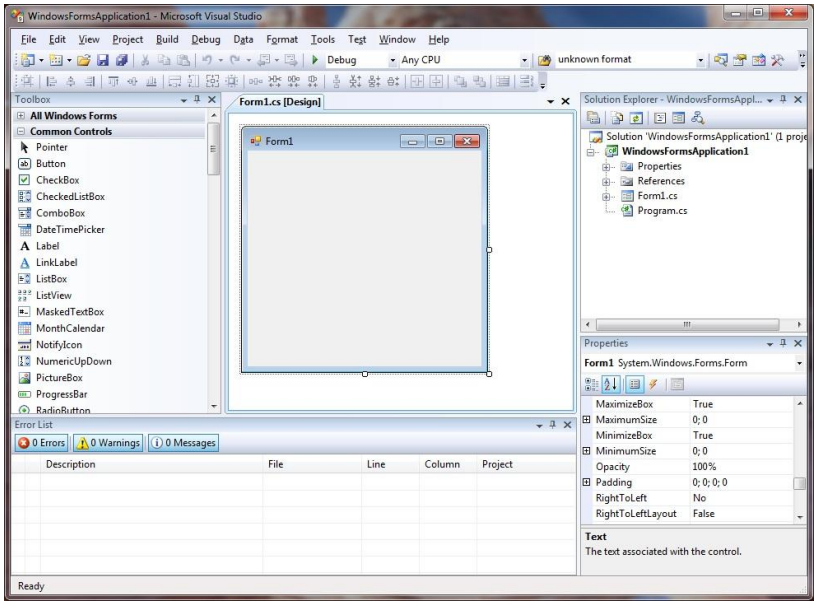
Za vpeljavo kontrole pristopa v hotelu smo najprej potrebovali ustrezno strojno opremo, ki bo omogočala delovanje po zahtevah naročnika, hkrati pa tudi programsko opremo, ki bo enostavna za uporabo in pisana na kožo naročnika. Tako smo že na začetku ugotovili, da iščemo strojno opremo, ki bo ustrezala tako zahtevam naročnika, kot tudi zahtevi, da jo je moč upravljati z lastno razvito aplikacijo.

3.1 ORODJA ZA RAZVOJ APLIKACIJE

Za razvoj aplikacije za upravljanje RFID pametnih kartic je bilo potrebno izbrati primerno ogrodje in orodja za povezavo aplikacije s strojno opremo. V tem poglavju predstavljamo izbrano ogrodje ter orodja, ki so pomagala pri razvoju aplikacije.

3.1.1 MICROSOFT VISUAL STUDIO 2008

Microsoft Visual Studio je integrirano razvojno orodje (*ang. Integrated Development Environment ali IDE*), ki ga razvijalci programske opreme najpogosteje uporabljajo za razvoj konzolnih aplikacij, aplikacij z grafičnim vmesnikom, internetnih aplikacij, internetnih storitev (*ang. web services*) ter mobilnih aplikacij. Uporablja se za razvoj aplikacij na osnovi Microsoft .NET ogrodja. [6]



Slika 4: Orodje za razvoj aplikacij Microsoft Visual Studio 2008

Microsoft Visual Studio vsebuje:

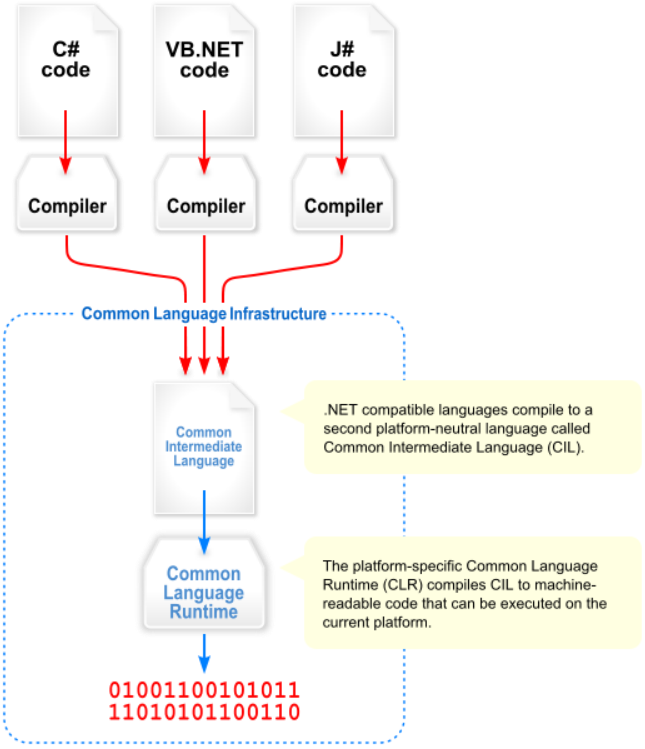
1. urejevalnik programske kode
2. orodje za načrtovanje grafičnih vmesnikov
3. orodjarno
4. orodje za načrtovanje projektov
5. razhroščevalnik
6. prevajalnik
7. pregledovalnik objektov, dokumentov, rešitev
8. druga orodja, ki pomagajo pri načrtovanju programskih aplikacij

Nekatere od naštetih stvari prikazuje slika 10. Microsoft Visual Studio 2008 nudi podporo različnim programskim jezikom, kot so Visual C#, Visual C++, VB - Visual Basic ter drugim.

3.1.2 .NET OGRODJE

Ogrodje .NET (*ang. .NET Framework*) je programsko ogrodje, ki primarno teče v okolju Microsoft Windows. Vsebuje veliko knjižnic in podpira različne programske jezike. Programi, ki so napisani za .NET ogrodje tečejo v programskem okolju imenovanem CLR (*ang. Common Language Runtime*), ki omogoča pomembne storitve kot so varnost,

upravljanje s pomnilnikom ter upravljanje z napakami. Zbirka knjižnic skupaj s CLR okoljem sestavlja .NET ogrodje.



Slika 5: .NET CLR - Common Language Runtime [9]

Namen CLR-ja, ki ga prikazuje slika 5 je, da se vsi programski jeziki se v ogrodju .NET prevajajo v vmesni jezik, zato je ena izmed prednosti .NET ogrodja ta, da so lahko deli programa napisani v različnih programskih jezikih. [7]

3.1.3 UPORABLJEN PROGRAMSKI JEZIK - C#

Pri izdelavi aplikacije za upravljanje RFID pametnih kartic smo uporabili programski jezik C#. Že samo ime nakazuje, da programski jezik izhaja iz sintakse jezika C++. S tem so razvijalcem, ki so prej uporabljali jezika C in C++ želeli olajšati prehod na nov jezik. C# poenostavlja nekatere težavne dele kot so novo samodejno zbiranje "smeti" t.i. pomnilnika, ki ga koda ne potrebuje več ter samodejno deklariranje spremenljivk. V sintaksi naj bi se zgledovali tudi po Visual Basicu in orodjem za hitro izdelavo programov RAD. S tem se C# po strukturi in načinu dela močno približuje Javi vendar Microsoft zatrjuje, da C# javi ni tekmec. Visual C# razvijalcem omogoča dostop do funkcij ogrodja Microsoft .NET

Framework ter sodoben in intuitiven sistem tipov, ki temelji na predmetih in odpravlja potrebo po zapletenih kazalcih. [8]

3.1.4 ACTIVE X KONTROLA

Za komunikacijo programske opreme s strojno opremo potrebujemo množico ukazov, ki jih kontrolna enota razume za komunikacijo. V ta namen je podjetje IDTECK poleg svoje strojne opreme na trg ponudilo `StarInterface.ocx` kontrolo, ki je bila izdelana na ogrodju ActiveX (*ang. ActiveX Control*), ki zagotavlja uporabo programskih komponent neodvisno od programskega jezika. Za komponento `StarInteface.ocx` so izdali obsežna navodila na kar 87. straneh, kjer opisujejo vse funkcije, ki jih ponuja komponenta. Za primer si poglejmo opis funkcije, s pomočjo katere na kontrolno enoto vpišemo RFID pametno kartico:

`CardIDDownload(CardID As String, Password As String, TimeSchedule As String, Reader As String, Finger As String) As String` [9]

Funkcijo `CardIDDownload` kličemo s parametri:

- `CardID` tipa `string`
  - številka RFID kartice, ki jo želimo shraniti v kontroler
- `Password` tipa `string`
  - geslo, ki ga želimo uporabiti v kombinaciji s kartico (samo pri nekaterih strojnih komponentah)
- `TimeSchedule` tipa `string`
  - urnik, po katerem naj deluje kartica (v našem primeru na kontrolni enoti iCON100 lahko nastavimo do 10 različnih urnikov)
- `Reader` tipa `string`
  - na katerem čitalcu naj kartica deluje (kontrolna enota ima 1-2 čitalca)
- `Finger` tipa `string`
  - informacija, ali se uporablja prstni odtis (v našem primeru kontroler tega ne podpira)

Funkcija nam vrača string, ki je lahko:

- `"0"` `NACK`: napaka
- `"Z"` `ACK`: operacija uspešna

Podobno zgornji so v navodilih proizvajalca navedene vse funkcije, ki jih je mogoče uporabiti z ActiveX kontrolo.

3.2 RAZVITA APLIKACIJA

V poglavju 3.2 predstavljamo celoten postopek razvoja aplikacije od zahtev do razvitega sistema. Strojno opremo, ki je bila uporabljena pri razvoju aplikacije za upravljanje RFID pametnih kartic predstavimo v poglavju 3.3.

3.2.1 ZAHTEVE NAROČNIKA

Celoten projekt temelji na željah oziroma zahtevah naročnika. Ideja naročnika je bila, da bi v hotel sezonskih delavcev v Novi Gorici namestili kontrolo pristopa, ki bi v kombinaciji z videonadzorom omogočala:

- *Odpravo 24 urnega fizičnega nadzora (vratar/receptor);*
  Vratar oziroma receptor je moral biti na objektu prisoten ves čas, saj je moral gostom odklepati vhodna vrata, ki so bila drugače z zunanje strani zaklenjena
- *Avtomatično blokiranje gostov, ki niso plačevali obveznosti;*
  Vratarji niso vedeli za vsakega gosta ali ima poravnane vse finančne obveznosti do hotela, zato niso mogli onemogočiti dostopa gostu
- *Nepooblaščen vstop in bivanje;*
  Gostje so ključe sob večkrat izdelovali na črno in tako omogočali dostop v hotel tretjim osebam, s katerimi so nato delili stroške
- *Kraje;*
  Občasno so se v sobah dogajale kraje, zaradi več ključev pa storilcev niso mogli najti

3.2.2 ANALIZA ZAHTEV

Za uspešno izpeljavo projekta je analiza zahtev prvi korak, ki ga moramo narediti. S tem se seznanimo s problemom in definiramo zahteve naročnika. V fazi analize določimo strojno opremo in operacijski sistem na katerem bo tekla programska rešitev. [10]

Analizo zahtev smo izdelali s pomočjo komunikacije s tistimi uporabniki, ki so bili del starega sistema (sistema ključev), ki je bil potreben prenove, torej tistimi ljudmi, ki stvari poznajo, in bodo nov sistem (kontrolo pristopa) uporabljali v vsakdanji praksi. S pomočjo pogovorov in predlogov naročnika smo prišli do vseh potrebnih informacij, ki smo jih potrebovali za vpeljavo sistema.

3.2.2.1 FUNKCIONALNE ZAHTEVE

Funkcionalne zahteve so tiste zahteve, ki se nanašajo na funkcionalnost sistema, torej tiste zahteve, ki jih poda naročnik. V konkretnem primeru so bile zahteve dokaj nejasno zastavljene, zato smo morali stopiti v kontakt ne samo z naročniki ampak tudi z uporabniki, ki so obstoječi sistem dejansko poznali in vedeli, kako stvar deluje pred implementacijo novega sistema.

S skupno analizo zahtev smo z naročniki in uporabniki določili funkcionalne zahteve aplikacije, ki se delijo na dva sklopa: upravljalski ter avtomatski modul.

ZAHTEVE ZA UPRAVLJALSKI MODUL

Uporabniku so omogočene funkcionalnosti, ki se delijo na dve veji; funkcionalnosti, ki jih želi naročnik, ter funkcionalnosti, ki so zahtevane za pravilno delovanje strojne opreme.

Funkcionalnosti, ki jih želi naročnik, so:

- Prijava uporabnika (administratorja) v sistem;
  Vsak upravljalec se v aplikacijo za upravljanje kontrole pristopa prijavi s svojim uporabniškim imenom, saj je naročnik želel, da je pri vsakem gostu razvidno, kdo ga je vnesel v sistem.
- Možnost branja RFID kartic z dodatnim čitalcem;
  Za lažje vpisovanje novih RFID kartic v sistem smo se odločili za dodaten čitalec, s katerim aplikacija prebere RFID kartico in jo prenese v obrazec za vpis novega gosta.
- Vnos gosta s parametri;
  - Priimek
  - Ime
  - Telefonska številka

  - RFID kartica
  - Opombe
  - Slika
- Nastavljanje dostopa do prostorov za vsakega uporabnika posebej;
  Uporabnik gostu določi, katera vrata lahko s svojo kartico odpira (na primer "Glavna vrata hotela", "Stranska vrata hotela", "Zapornica parkirišča", "Soba 102") ter ob katerih časovnih intervalih to lahko počne.
- Možnost vnosa datuma, ko gostu poteče RFID kartica;
  Uporabnik lahko gostu določi datum, ko mu kartica preneha veljati.

Za pravilno delovanje pa je bilo potrebno izdelati tudi nekaj funkcionalnih zahtev, ki se na uporabnika neposredno ne vežejo, so pa predpisane s strani proizvajalca strojne opreme in so nujno potrebne za pravilno delovanje kontrole pristopa. Te funkcionalnosti so:

- **Nastavitev kontrolnih enot**

  KONTROLNA ENOTA
  Uporabniški vmesnik mora omogočati vnos N kontrolnih enot, pri čemer je za vsako enoto potrebno vnesti podatke:
  - Naziv enote
  - Opis enote
  - ID enote (zaporedna številka enote, ki se fizično nastavi na kontrolni enoti)
  - IP naslov enote (naslov ILAN-a, na katerega je enota vezana)
  - TCP vrata ILANA
  - "Anti-Passback" funkcija (ali enota deluje v "Anti-Passback" načinu)

  ČITALCI
  Uporabniški vmesnik mora omogočati vnos čitalcev, ki so vezani na določeno kontrolno enoto, pri čemer je za vsak čitalec potrebno vnesti podatke:
  - Naziv čitalca
  - Opis čitalca
  - Številka čitalca (vhod kontrolne enote, na kateri je čitalec)
  - Dogodki, ki jih čitalec sproži.

  URNIKI KONTROLNE ENOTE
  Vsaka kontrolna enota ima možnost vnosa do 10 različnih časovnih urnikov, pri čemer nam vsak urnik pove, kako se enota obnaša ob določeni uri v določenem dnevu oziroma ob praznikih.

TABELE PRAZNIKOV

Vsaka kontrolna enota ima možnost vnosa do 10 različnih tabel praznikov, kjer vsaka tabela lahko vsebuje največ 32 prazničnih dni, ki se nato uporabljajo v kombinaciji z urniki kontrolne enote.

- **Pošiljanje nastavitev**

Po vnosu vseh parametrov potrebnih za delovanje, je potreben vmesnik, ki nastavitve pošlje na kontrolne enote. Na enote se pošiljajo naslednje nastavitve:

- Datum in ura
- Nastavitve kontrolerjev (koliko čitalcev, kaj sprožijo)
- Urniki enot
- Tabele praznikov
- RFID kartice uporabnikov
- Funkcija za tovarniško ponastavitev enote

ZAHTEVE ZA AVTOMATSKI MODUL

Ker strojna oprema kontrole pristopa ne omogoča, da bi določene RFID kartice samodejno brisala po določenem datumu, je bilo potrebno izdelati programski modul, ki neodvisno od same aplikacije pregleda in izbriše pretečene kartice iz kontrolnih enot.

Zato mora tak programski modul omogočati:

- Dostop do podatkovne baze izdelane s pomočjo upravljalskega modula

Modul se mora znati povezati z aktualno podatkovno bazo.

- Pridobivanje seznama vseh gostov, ki so jim potekle RFID kartice
- Brisanje poteklih RFID kartic
    - Možnost avtomatskega zagona ob določeni uri
    - Možnost ročnega zagona
- Grafični izpis brisanih kartic iz kontrolnih enot

3.2.2.2 NEFUNKCIONALNE ZAHTEVE

Nefunkcionalne zahteve so tiste zahteve, ki se nanašajo na tehnične in druge nevsebinske zahteve sistema.

Strojna in programska oprema
Glede strojne opreme so zahteve naslednje: potrebujemo osebni računalnik z mrežno kartico za komunikacijo s kontrolerjem prek mrežnega vmesnika iLAN oziroma s serijskim vmesnikom za komunikacijo s kontrolerjem prek RS232 vmesnika. Kar se tiče programske opreme potrebujemo Windows operacijski sistem s .NET Framework 2.0 ogrodjem ter programskim paketom za upravljanje z `.mdb` (Microsoft Access DB) datotekami (Microsoft Office, MS Database Engine).

Varnost
Podatkovna baza je osnova za delovanje programa, v njej so med drugim tudi osebni podatki gostov, zato je pomembno, da je primerno zaščitena pred dostopom nepooblaščenih oseb.

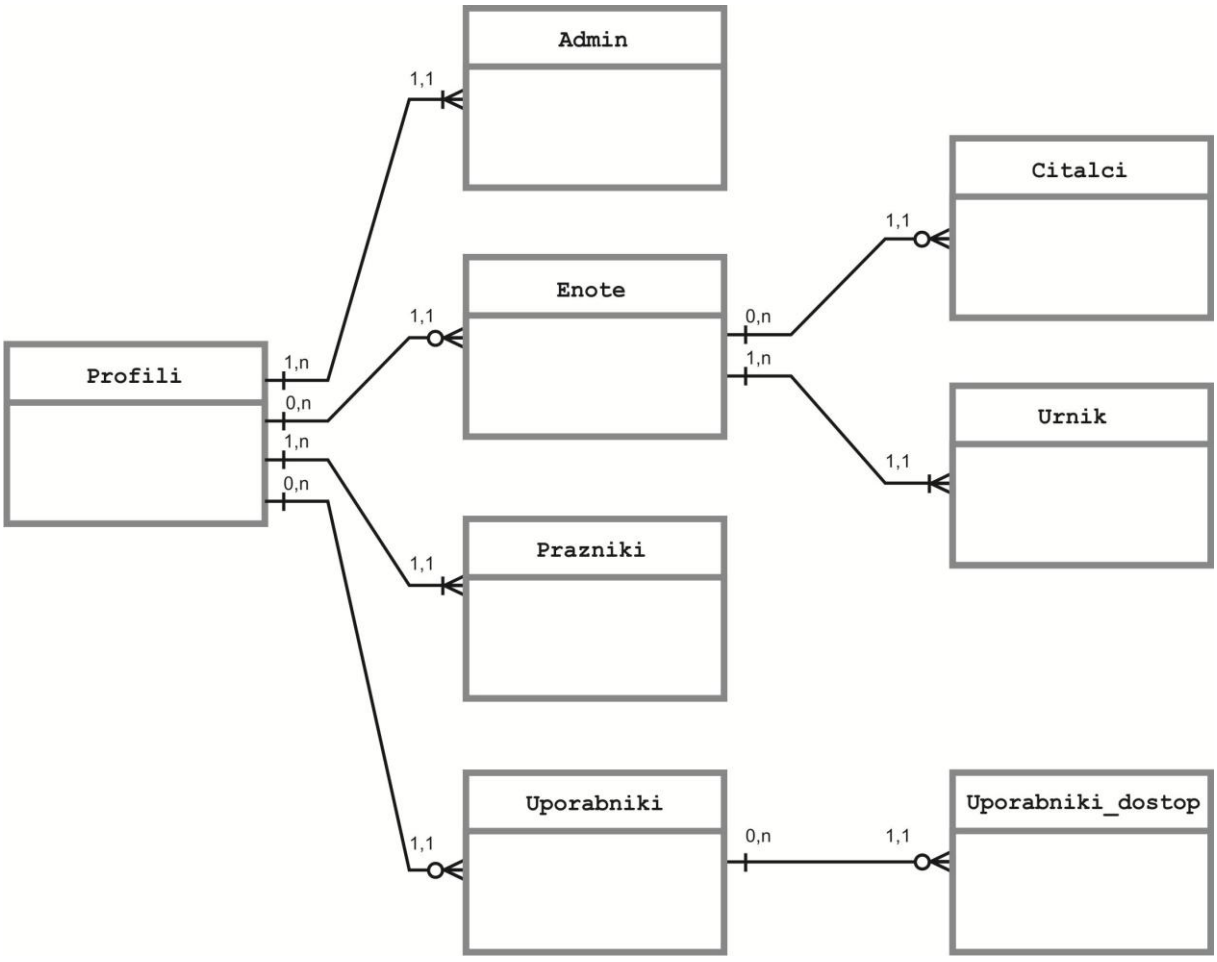
Uporabniški vmesnik
Uporabniški vmesnik mora biti intuitiven in enostaven za uporabo, omogočati mora hiter dostop do želenih podatkov.

Odzivnost
Program mora biti odziven in v hitrem času uporabniku podati zahtevane informacije. Najšibkejši člen programa je pošiljanje in prejemanje podatkov iz enot, ki lahko v določenih primerih traja do nekaj sekund. Zato mora biti uporabniški vmesnik izdelan tako, da uporabnik nima občutka, da se program ne odziva.

Zanesljivost
Program mora ob vsakem zagonu omogočati vse funkcije, ki jih ima, zato je nujno potrebno, da se ga v času razvoja dobro testira na različnih operacijskih sistemih pod različnimi pogoji.

3.2.3 PODATKOVNA BAZA

Za hrambo podatkov skrbi podatkovna baza Microsoft Access Database, ki je realizirana kot `.mdb` datoteka, zato za uporabo programa ne potrebujemo "prave" podatkovne baze, ki bi jo bilo potrebno nameščati na strežnik in vzdrževati. Access-ova podatkovna baza je enostavna za arhiviranje, saj jo uporabnik lahko kot datoteko iz programa izvozi prek menija. Seveda za veliko količino podatkov ni primerna, za potrebe naše aplikacije pa je več kot uporabna. Podatkovna baza je sestavljena iz osmih tabel, relacije med njimi pa prikazuje slika 5. Polja posameznih tabel so opisana v nadaljevanju.



Slika 5: Podatkovna baza - relacije med tabelami

3.2.3.1 TABELA PROFILI

V tabeli `Profili` hranimo podatke o profilih uporabnikov programa. Predstavljajmo si, da kot vzdrževalec kontrole pristopa servisiramo več podjetij. S pomočjo profilov si lahko na enostaven način izdelamo bazo podatkov vseh kontrolnih enot, s katerimi upravljamo.
Polja v tabeli `Profil` so opisana v tabeli 1.

Ime polja	Tip	Opis
id_profil	Long Integer	ID zapisa profil
ime_profil	Text	Ime profila
opis_profil	Memo	Opis profila
mapa_profil	Text	Pot do mape, kjer se nahaja podatkovna baza profila in slike uporabnikov

Tabela 1: Opis polj v tabeli Profili

3.2.3.2 TABELA ADMIN

V tabeli `Admin` hranimo vse skrbniške račune uporabnikov, torej tistih uporabnikov, ki upravljajo z aplikacijo. Polja v tabeli `Admin` so opisana v tabeli 2.

Ime polja	Tip	Opis
id_admin	Long Integer	ID zapisa admin
id_profil	Long Integer	ID zapisa profil
admin_priimek	Text	Priimek skrbnika
admin_ime	Text	Ime skrbnika
admin_upime	Text	Uporabniško ime skrbnika
amin_geslo	Text	Geslo skrbnika

Tabela 2: Opis polj v tabeli Admin

3.2.3.3 TABELA ENOTE

V tabeli `Enote` so shranjene vse kontrolne enote kontrole pristopa izbranega profila. Vsaka enota ima poleg svojega naziva in opisa tudi nastavitve, ki so predpisane s strani proizvajalca IDTECK. Ta polja so označena z velikimi črkami, polja so opisana v tabeli 3.

Ime polja	Tip	Opis
id_enota	Long Integer	ID zapisa enota
id_profil	Long Integer	ID zapisa profil
naziv_enota	Text	Naziv kontrolne enote
opis_enota	Memo	Opis kontrolne enote
BOARD_ID	Text	ID enote s stikali nastavljiv na kontrolni enoti
IP_ADDRESS	Text	IP naslov vmesnika, kjer se nahaja kontrolna enota
TCP_PORT	Number	TCP vrata, kjer se nahaja kontrolna enota
ANTI_PASSBACK	Yes/No (bit)	Nastavitev, ali enota uporablja funkcijo AntiPassback

Tabela 3: Opis polj v tabeli Enote

3.2.3.4 TABELA CITALCI

V tej tabeli hranimo čitalce posameznih kontrolnih enot in njihovo delovanje ob dogodkih, ki se vršijo ob pristopu. Za vsak čitalec lahko nastavimo časovni interval delovanja posameznega releja (signala za odpiranje vrat) na kontrolni enoti, ki ga podamo v sekundah. Polja so opisana v tabeli 4.

Ime polja	Tip	Opis
id_citalec	Long Integer	ID zapisa čitalec
id_enota	Long Integer	ID zapisa enota
naziv_citalec	Text	Naziv čitalca
opis_citalec	Memo	Opis čitalca
RELAY_NR	Number	Številka vhoda, kjer je priklopljen čitalec
OK1	Text	Dogodek, ki se izvrši na releju 1 ob pristopu registrirane RFID kartice, katere urnik je veljaven
OK2	Text	Dogodek, ki se izvrši na releju 2 ob pristopu registrirane RFID kartice, katere urnik je veljaven
UNREG1	Text	Dogodek, ki se izvrši na releju 1 ob pristopu neregistrirane RFID kartice
UNREG2	Text	Dogodek, ki se izvrši na releju 2 ob pristopu neregistrirane RFID kartice
TIMESCH1	Text	Dogodek, ki se izvrši na releju 1 ob pristopu registrirane RFID kartice, katere urnik ni veljaven
TIMESCH2	Text	Dogodek, ki se izvrši na releju 2 ob pristopu registrirane RFID kartice, katere urnik ni veljaven

Tabela 4: Opis polj v tabeli Citalci

3.2.3.5 TABELA URNIK

V tabeli `Urnik` so shranjeni vsi urniki, ki pripadajo določeni kontrolni enoti. Vsaka kontrolna enota je omejena na 10 različnih urnikov (omejitev s strani strojne opreme), ki so razdeljeni po dnevih od ponedeljka do nedelje z dodatnim poljem "praznik", ki uveljavlja poseben urnik ob prazničnih dneh (vnešeni v tabeli prazniki). Vsak urnik se lahko sklicuje na 1 tabelo praznikov. Vsak posamezen dan urnika ima lahko do 5 časovnih intervalov. `Urnik` ima polja, opisana v tabeli 5.

Ime polja	**Tip**	**Opis**
id_urnik	Long Integer	ID zapisa urnik
id_enota	Long Integer	ID zapisa enota
ID_TECK_SCHEDULE	Text	Številka urnika v kontrolni enoti
id_praznik	Long Integer	ID zapisa praznik
naziv_urnik	Text	Naziv urnika
opis_urnik	Memo	Opis urnika
dan_urnik	Text	Dan urnika
sch11	Text	Časovni interval 1 (od)
sch12	Text	Časovni interval 1 (do)
sch21	Text	Časovni interval 2 (od)
sch22	Text	Časovni interval 2 (do)
sch31	Text	Časovni interval 3 (od)
sch32	Text	Časovni interval 3 (do)
sch41	Text	Časovni interval 4 (od)
sch42	Text	Časovni interval 4 (do)
sch51	Text	Časovni interval 5 (od)
sch52	Text	Časovni interval 5 (do)

Tabela 5: Opis polj v tabeli Urnik

3.2.3.6 TABELA PRAZNIKI

V tej tabeli hranimo tabele praznikov. Vsaka tabela ima največ 32 datumov (omejitev s strani strojne opreme), vsaka kontrolna enota pa ima lahko največ 10 tabel. Polja tabele `Prazniki` so opisana v tabeli 6.

Ime polja	Tip	Opis
id_praznik	Long Integer	ID zapisa čitalec
id_profil	Long Integer	ID zapisa profil
ID_TECK_HOLIDAY_CODE	Text	Številka tabele praznikov v kontrolni enoti
ime_praznik	Text	Ime tabele praznikov
opis_praznik	Memo	Opis tabele praznikov
d1_opis	Text	Ime praznika 1
d1	Text	Datum praznika 1
. . .		
d32_opis	Text	Ime praznika 32
d32	Text	Datum praznika 32

Tabela 6: Opis polj v tabeli prazniki

3.2.3.7 TABELA UPORABNIKI

V tabeli `Uporabniki` hranimo uporabnike, njihove podatke, številko ter veljavnost RFID kartice za vsakega uporabnika. Tu hranimo tudi nastavitev, ali je uporabnik omogočen, ter podatek o tem, kateri skrbniški račun je zadnji na uporabniku izvedel spremembo. Tabela 7 opisuje vsa polja te tabele.

Ime polja	Tip	Opis
id_uporabnik	Long Integer	ID zapisa uporabnik
id_profil	Long Integer	ID zapisa profil
priimek_uporabnik	Text	Priimek uporabnika
ime_uporabnik	Text	Ime uporabnika
tel_uporabnik	Text	Telefon uporabnika
opombe_uporabnik	Memo	Opombe uporabnika
slika_uporabnik	Text	Ime datoteke s sliko

ID_TECK_RFID	Text	RFID številka kartice
omogocen_uporabnik	Yes/No (bit)	Nastavitev, ali je uporabnik omogočen
veljavnost	Date/Time	Nastavitev, kdaj uporabniku poteče RFID kartica
id_admin	Long Integer	ID zapisa admin, ki je uporabnika vnesel/spreminjal

Tabela 7: Opis polj v tabeli Uporabnik

3.2.3.8 TABELA UPORABNIKI_DOSTOP

V tej tabeli hranimo dostopne pravice uporabnikov na posameznih enotah in čitalcih po določenih urnikih. Polja so opisana v tabeli 8.

Ime polja	**Tip**	**Opis**
id_uporabnik_dostop	Long Integer	ID zapisa uporabnik_dostop
id_uporabnik	Long Integer	ID zapisa uporabnik
id_enota	Long Integer	ID zapisa enota
citalec1_vstop	Yes/No (bit)	Pravica do vstopa na čitalcu 1
citalec2_vstop	Yes/No (bit)	Pravica do vstopa na čitalcu 2
ID_TECK_SCHEDULE	Text	Številka urnika v kontrolni enoti

Tabela 8: Opis polj v tabeli Uporabniki_dostop

Strojna omejitev določa, da ima uporabnik na enoti pravice do enega ali drugega čitalca le po enem urniku, ki velja za oba čitalca, zato na primer ni mogoče na isti enoti uporabniku na različnih čitalcih nastaviti različen urnik.

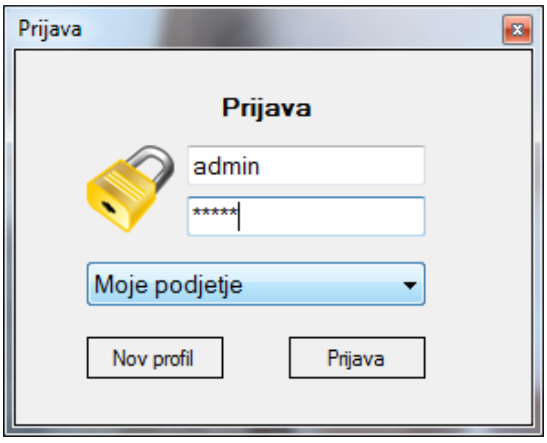
3.2.4 UPRAVLJALSKI MODUL

Upravljalski modul je razvit kot samostojna aplikacija, ki smo jo poimenovali Vratar Mini. Mini zato, ker sama po sebi nima možnosti vpogleda v zgodovino dogodkov, ki so se zgodili na posamezni enoti, ampak le upravlja z uporabniki, RFID karticami, kontrolo pristopa in njenimi nastavitvami. Tako lahko s pomočjo aplikacije Vratar Mini upravljamo s skrbniškimi računi, s pomočjo katerih določimo, katere osebe lahko s kontrolo pristopa in uporabniki sploh upravljajo, spreminjamo nastavitve kontrolnih enot in čitalcev, upravljamo s časovnimi urniki posameznih kontrolnih enot ter tabelami praznikov. Prav tako s pomočjo upravljalskega modula vse nastavitve prek TCP/IP povezave pošiljamo na kontrolne enote. V nadaljevanju si bomo pogledali izgled in delovanje upravljalskega modula Vratar Mini ter programsko logiko, ki stoji za njim.

3.2.4.1 IZGLED IN DELOVANJE

PRIJAVA

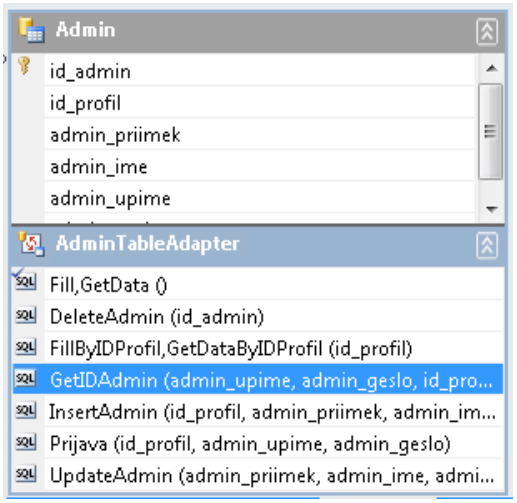
Za prijavo v aplikacijo Vratar Mini (slika 6) potrebujemo uporabniško ime in geslo skrbnika, ki je vezano na podjetje oziroma profil. Pri prvem zagonu programa je v profilu "Moje podjetje" že izdelan skrbniški račun z uporabniškim imenom in geslom "admin", ki ga po prijavi lahko spremenimo.



Slika 6: Prijava v program Vratar Mini

Prijava uporabnika je uspešna, če v podatkovni bazi obstaja natanko en zapis, ki se ujema z uporabniškim imenom, geslom in profilom. To preverimo s poizvedbenim stavkom

poimenovanim Prijava, ki je shranjen v komponenti Visual Studia za upravljanje s podatkovnimi tabelami imenovanim `DataSet`, ki v sebi vsebuje več vmesnikov za tabele (TableAdapters). Tak vmesnik znotraj `DataSet`-a v Visual Studiu lahko vsebuje več poizvedbenih stavkov, na katere se nato programer sklicuje. Vmesnik za tabelo `Admin` prikazuje slika 7. [11]



Slika 7: Vmesnik za tabelo (TableAdapter) Admin

Vsak poizvedbeni stavek v sebi skriva SQL poizvedbo. Tako na primer "Prijava" vsebuje naslednji poizvedbeni stavek:

```
SELECT      COUNT(*) AS Expr1
FROM          Admin
WHERE      (id_profil = ?) AND (admin_upime = ?) AND (admin_geslo = ?)
```

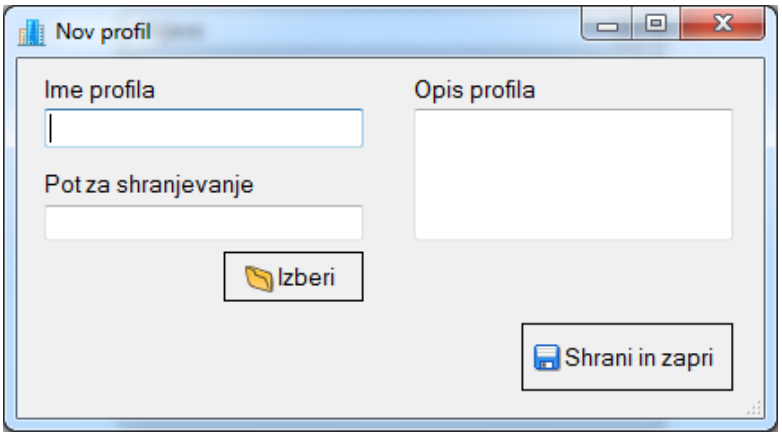
Ob pritisku gumba "Prijava" na sliki 6 pokličemo proceduro Preveri uporabnika, ki ji posredujemo podatke o uporabniškem imenu, geslu in profilu:

```
private void PreveriUporabnika(string up_ime, string geslo, int idp)
        {
            if (adminTableAdapter1.Prijava(idp,up_ime,geslo).Value==1)
            {
                if ((int)profiliTableAdapter.NiDirektorija(idp) == 1)
                    NastaviDirProfila(idp);

if
(!Directory.Exists(profiliTableAdapter.PodatkiProfila(idp)[0].mapa_profil.ToString(
)))
                    NastaviDirProfila(idp);
```

```
                Form1 f = new Form1();
                f.id_profil = idp;
                f.id_admin = adminTableAdapter1.GetIDAdmin(up_ime, geslo,
idp)[0].id_admin;
                f.Show(this);
                this.Hide();
                textBox1.Clear();
                maskedTextBox1.Clear();
            }

        }
```

Procedura v prvem koraku naredi poizvedbo na podatkovni bazi, kjer preveri avtentikacijo. V drugem koraku preveri, če ima izbrani profil nastavljeno pot za shranjevanje podatkov (to preverjanje se po navadi izvrši pozitivno le pri prvem zagonu programa), nato prikaže glavno zaslonsko masko programa, kateremu nastavi parametra `id_profil` in `id_admin`, s katerima določimo nad katerim profilom prijavljen uporabnik operira ter kdo je ta uporabnik.

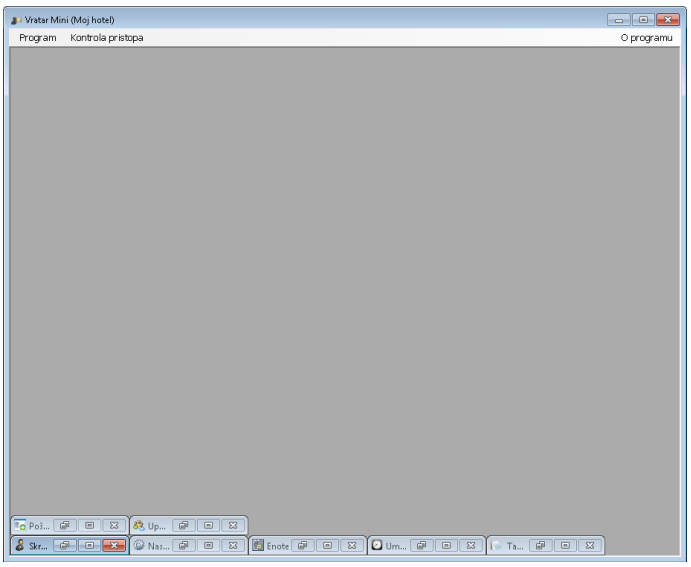
Prijavno okno ima poleg gumba `Prijava` tudi gumb `Nov profil` (slika 6), s pomočjo katerega lahko izdelamo nov profil, ki mu določimo ime in pot za shranjevanje podatkov (slika 7). Podatkovna baza v sebi lahko nosi več različnih profilov, kar pomeni, da si lahko na enem računalniku nastavimo več podjetij oziroma več lokacij, s katerimi upravljamo.



Slika 8: Kreiranje novega profila

Po uspešni prijavi v prijavnem oknu (slika 6) se nam odpre glavno okno programa (slika 9), ki je namenjeno upravljanju z vsemi funkcijami, ki jih program premore. Glavno okno je načrtovano kot večnamenski vmesnik za dokumente (*ang. MDI Container, Multiple*

Document Interface Container), saj nam tak vmesnik omogoča strukturirano delo in lažjo uporabo.



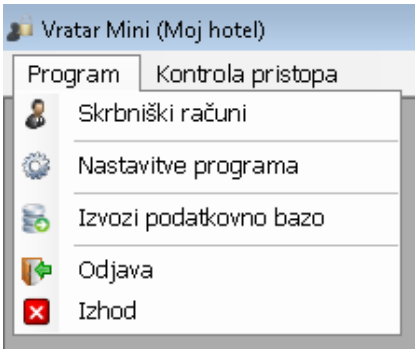
Slika 9: Glavno okno programa

Vsa okna, ki so del programa, po potrebi lahko poljubno premikamo, vedno pa se nahajajo znotraj glavnega okna (slika 9). Glavno okno ima 2 glavna menija:

- "Program" in
- "Kontrola pristopa".

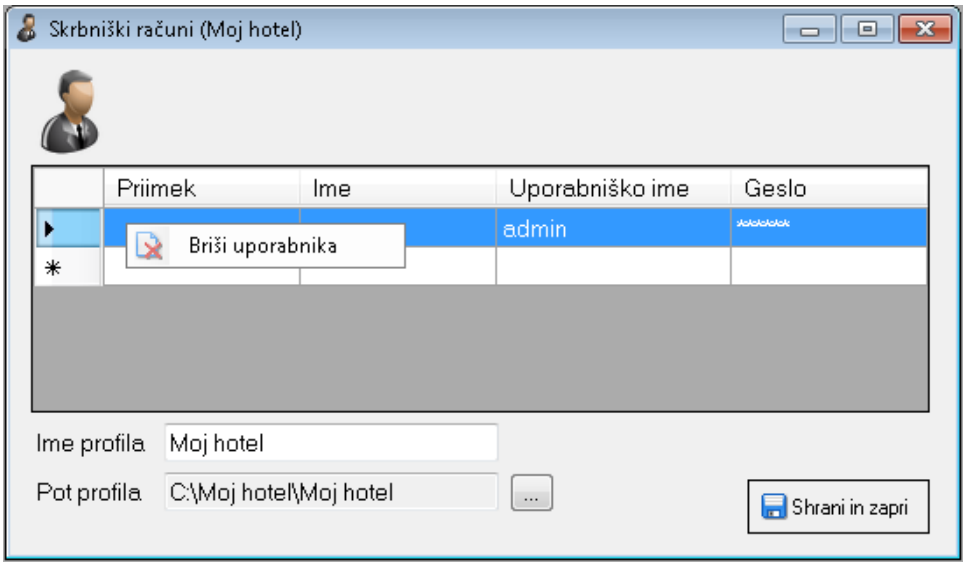
MENI PROGRAM

Meni `Program` sestavlja 5 podmenijev (slika 10). Ti podmeniji odpirajo nova okna z nastavitvami oziroma neposredno izvajajo svoje funkcije.



Slika 10: Podmeniji menija Program

Meni "Skrbniški računi" odpira novo okno, kjer nastavljamo skrbniške račune oziroma uporabnike, ki so skrbniki kontrole pristopa oziroma sistema. Uporabnikov je lahko več, vsakemu pa lahko vnesemo podatke kot so priimek, ime ter uporabniško ime in geslo, ki sta obvezni polji. Uporabnika lahko izbrišemo tako, da ga izberemo ter z desnim klikom nanj izberemo meni `Briši uporabnika` (slika 11). Uporabnika lahko izbrišemo le, če po brisanju računa v tabeli skrbniških računov ostane vsaj en račun.

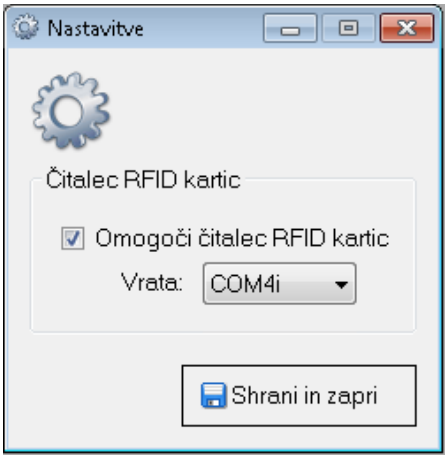


Slika 11: Okno skrbniški računi

V oknu `Skrbniški računi` (slika 11) lahko spremenimo tudi ime profila in lokacijo za shranjevanje podatkov. Glavna komponenta okna je prikazna tabela (*ang. Datagridview*), ki prikazuje skrbniške račune. Shranjevanje in posodabljanje podatkov o uporabniških računih, profilu in poti se sproži ob pritisku na gumb `Shrani iz zapri`, medtem ko se brisanje uporabniškega računa izvaja neposredno ob pritisku na meni `Briši uporabnika`. Programska koda, ki se skriva za gumbom `Shrani in zapri` je naslednja:

```
foreach (DataGridViewRow r in dataGridView1.Rows)
            {
                if (!(r.Cells["id_admin"].Value == null))
                    if ((int)r.Cells["id_admin"].Value < 0)
                    {
                        if (r.Cells["up_ime"].Value.ToString() != "" &&
r.Cells["geslo"].Value.ToString() != "")
                            adminTableAdapter.InsertAdmin(id_p,
r.Cells["priimek"].Value.ToString(),
                                r.Cells["ime"].Value.ToString(),
r.Cells["up_ime"].Value.ToString(), r.Cells["geslo"].Value.ToString());
                    }
                    else
                    {
                        if (r.Cells["up_ime"].Value.ToString() != "" &&
r.Cells["geslo"].Value.ToString() != "")

adminTableAdapter.UpdateAdmin(r.Cells["priimek"].Value.ToString(),
r.Cells["ime"].Value.ToString(),
                                r.Cells["up_ime"].Value.ToString(),
                                r.Cells["geslo"].Value.ToString() != "******" ?
r.Cells["geslo"].Value.ToString() : r.Cells["geslo"].Tag.ToString(),
Convert.ToInt32(r.Cells["id_admin"].Value));
                    }

            }

            if (textBox1.Text.Length == 0)
                textBox1.Text = "Moje podjetje";
            profiliTableAdapter1.UpdateImeProfila(textBox1.Text, id_p);
            f.Text = "Vratar Mini (" + textBox1.Text + ")";
            this.Close();
```

Meni "Nastavitve programa" odpre okno (slika 12), kjer lahko nastavimo vrata RS232 čitalca pametnih kartic, če ga imamo. Program nato ob vnosu uporabnika dovoljuje možnost, da v polje "številka kartice" čitalec posreduje RFID številko, kar olajša vnos skrbniku.



Slika 12: Nastavitve

V programu ob nalaganju okna `Nastavitve programa` s pomočjo .Net knjižnice `System.IO.Ports` od računalnika, kjer teče aplikacija, prejmemo seznam vseh komunikacijskih vrat, ki obstajajo na tem računalniku, in nato vrata prenesemo v izbirni seznam (*ang. Dropdown Menu*). S pomočjo programske kode to izvedemo v dveh vrsticah:

```
foreach (string s in SerialPort.GetPortNames())
                    comboBox1.Items.Add(s);
```

Za razliko od ostalih nastavitev te nastavitve ne shranjujemo v bazo podatkov, ampak v konfiguracijsko datoteko, ki se ob zagonu programa prebere z diska. Ob nalaganju programa se tako kliče procedura, ki preveri obstoj konfiguracijske datoteke in branje nastavitev:

```
private void NaloziNastavitve()
        {
            if
(File.Exists(System.Environment.GetFolderPath(Environment.SpecialFolder.LocalApplic
ationData) + "\\VratarBCfg.xml"))
            {
                //branje xml
                XmlTextReader textReader = new XmlTextReader(
System.Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) +
"\\VratarBCfg.xml");

                string sName = "";
                while (textReader.Read())
                {
                    switch (textReader.NodeType)
                    {
                        case XmlNodeType.Element:
                            sName = textReader.Name;
                            break;
```

```
                    case XmlNodeType.Text:
                        switch (sName)
                        {

                            case "bralec":
                                reader = Convert.ToBoolean(textReader.Value);
                                break;
                            case "bralec_vrata":
                                com_port = textReader.Value;
                                break;
                        }
                        break;
                }
            }
            textReader.Close();
        }
    }
```

Meni "Izvozi podatkovno bazo" za razliko od prejšnjih dveh ne odpira novega okna, ampak kliče t.i. dialog za shranjevanje (*ang. SaveFileDialog*), ki je v .Net ogrodju knjižnica, ki olajša delo pri shranjevanju podatkov. S tem ukazom lahko uporabnik s pomočjo enega klika izvozi celotno podatkovno bazo. To pride prav še posebej takrat, kadar želimo podatke prenašati iz enega računalnika na drugega oziroma kadar želimo izdelati varnostno kopijo. Ob pritisku na meni se sproži naslednja programska koda:

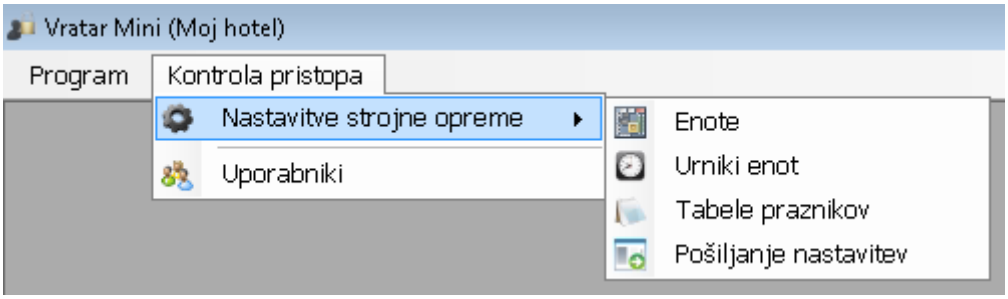
```
SaveFileDialog s = new SaveFileDialog();
            s.FileName = "Vratar.mdb";
            s.Filter = "Database File|*.mdb";
            if (s.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    File.Copy(Directory.GetCurrentDirectory() + "\\Vratar.mdb",
                       s.FileName, true);
                    MessageBox.Show("Baza uspešno shranjena.", "Shranjevanje baze",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Napaka pri shranjevanju baze:\r\n"+ex.Message,
"Shranjevanje baze", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }

            }
```

Meni "Odjava" je namenjen odjavi trenutnega skrbniškega računa. Ob pritisku na ta meni program zapre glavno okno in ponovno prikaže Prijavno okno (slika 6).

Meni "Izhod" pa povzroči izhod iz aplikacije oziroma njeno ustavitev.

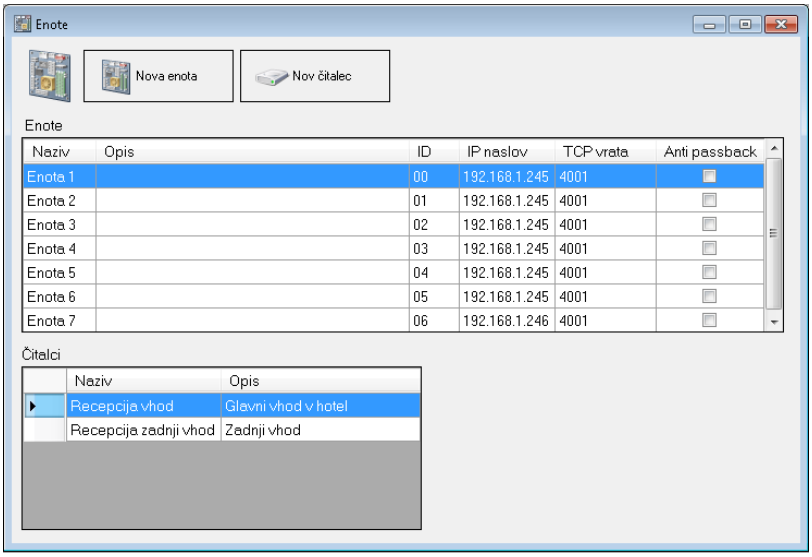
MENI KONTROLA PRISTOPA

Meni "Kontrola pristopa" sestavljata dva podmenija, prvi pa je sestavljen še iz štirih dodatnih podmenijev (slika 13).



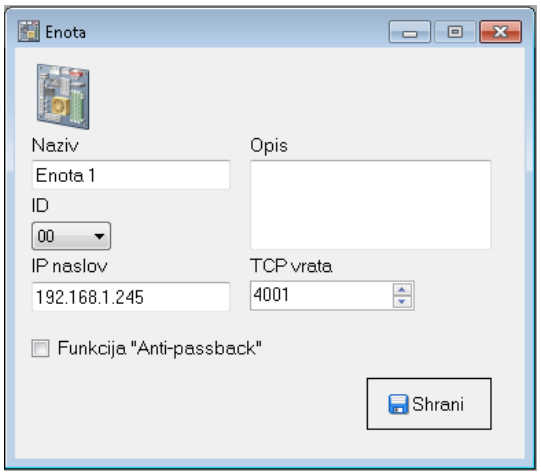
Slika 13: Meni Kontrola pristopa

Meni "Nastavitve strojne opreme/Enote" odpira novo okno, kjer se nastavlja osnovne nastavitve kontrolnih enot (slika 14). Tukaj se dodaja nove kontrolne enote, ureja njihove nastavitve in dodaja čitalce. Zaradi strojne omejitve ima lahko ena enota največ 2 čitalca pametnih kartic, zato eni enoti ni mogoče dodati več kot 2 čitalca.



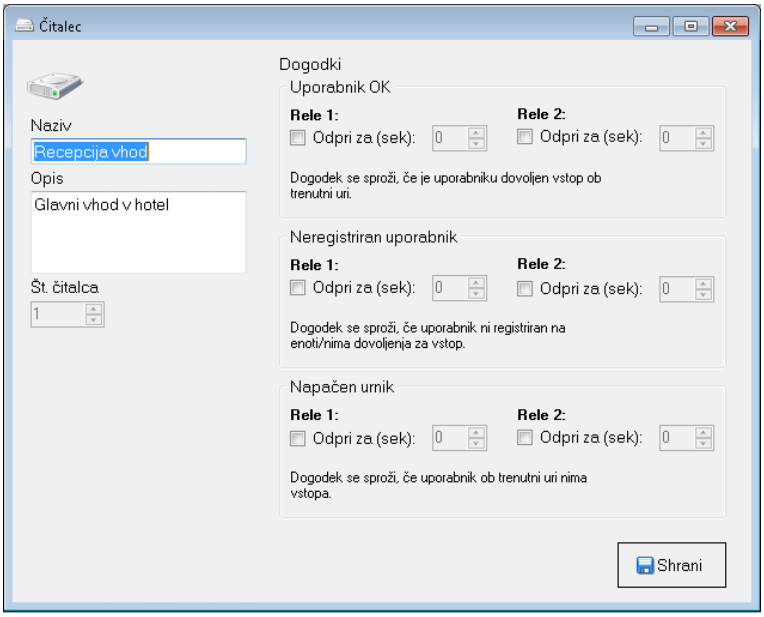
Slika 14: Kontrolne enote

S klikom na gumb "Nova enota" se odpre novo okno, kjer vnašamo podatke o enoti; naziv, opis, ID, IP naslov, TCP vrata, Anti-Passback funkcija. Zelo pomembno je, da za vsako enoto pravilno vnesemo polja ID, IP naslov in TCP vrata. Polje ID je definirano kot dvomestna številka, teče pa od 00 do 10. Gre za številko enote, ki jo na kontrolni enoti nastavimo s pomočjo mikro stikal. Vsaka kontrolna enota je vezana na TCP/IP vmesnik, ki ima svoj IP naslov in TCP vrata. En TCP/IP vmesnik lahko komunicira z več kontrolnimi enotami, vendar mora imeti vsaka enota v skupini svoj ID. Tako je lahko v našem primeru na en vmesnik priklopljenih 11 različnih kontrolnih enot. Več o uporabljenem TCP/IP vmesniku in kontrolni enoti lahko najdemo v poglavju 3.3. Funkcija `Anti-Passback` je poseben način delovanja kontrolne enote, ki je opisan v poglavju 3.3.3.



Slika 15: Urejanje kontrolne enote

Ko imamo vnešeno kontrolno enoto, lahko enoti dodamo do dva čitalca. Vsak čitalec ima svoje nastavitve (slika 16).



Slika 16: Nastavitve čitalca

Vsakemu čitalcu lahko poleg naziva in opisa vnesemo nastavitve, kako se kontrolna enota obnaša ob dogodkih, ki so lahko za vsako kartico, ki jo prebere čitalec naslednji:

- Registriran uporabnik
  Dogodek se sproži takrat, kadar čitalec kartic kontrolni enoti posreduje številko kartice, ki je na kontrolni enoti registrirana, hkrati pa je trenutni čas registracije kartice znotraj urnika, ki je določen kartici
- Neregistriran uporabnik
  Dogodek se sproži takrat, kadar čitalec kartic kontrolni enoti posreduje številko kartice, ki na kontrolni enoti ni registrirana
- Napačen urnik
  Dogodek se sproži takrat, kadar čitalec kartic kontrolni enoti posreduje številko kartice, ki je na kontrolni enoti registrirana, trenutni čas registracije kartice pa ni znotraj urnika, ki je določen kartici

Za vsakega od zgoraj napisanih dogodkov lahko nastavljamo delovanje releja 1 in 2. Releji so na kontrolni enoti nekakšna stikala, ki ključavnici vrat pošiljajo signal, ali so vrata odprta ali ne. Za vsako stikalo lahko ob vsakem dogodku nastavimo, koliko časa (sekund) naj bo aktivno.

Po vnosu nastavitev se ob pritisku na gumb `Shrani in zapri` najprej sproži procedura `DoWork`, ki olajša kasnejše shranjevanje podatkov v bazo:

```
private void DoWork()
        {
            //preveri evente
            if (numericUpDown1a.Value < 10)
                ok1 = "0" + numericUpDown1a.Value.ToString();
            else
                ok1 = numericUpDown1a.Value.ToString();

            if (numericUpDown2.Value < 10)
                ok2 = "0" + numericUpDown2.Value.ToString();
            else
                ok2 = numericUpDown2.Value.ToString();

            if (numericUpDown3.Value < 10)
                ner1 = "0" + numericUpDown3.Value.ToString();
            else
                ner1 = numericUpDown3.Value.ToString();

            if (numericUpDown4.Value < 10)
                ner2 = "0" + numericUpDown4.Value.ToString();
            else
                ner2 = numericUpDown4.Value.ToString();

            if (numericUpDown5.Value < 10)
                ur1 = "0" + numericUpDown5.Value.ToString();
            else
                ur1 = numericUpDown5.Value.ToString();

            if (numericUpDown6.Value < 10)
                ur2 = "0" + numericUpDown6.Value.ToString();
            else
                ur2 = numericUpDown6.Value.ToString();
        }
```

Procedura si v začasne spremenljivke shrani čas v obliki string, saj ga v taki obliki pri pošiljanju nastavitev na kontrolne enote posredujemo naprej.

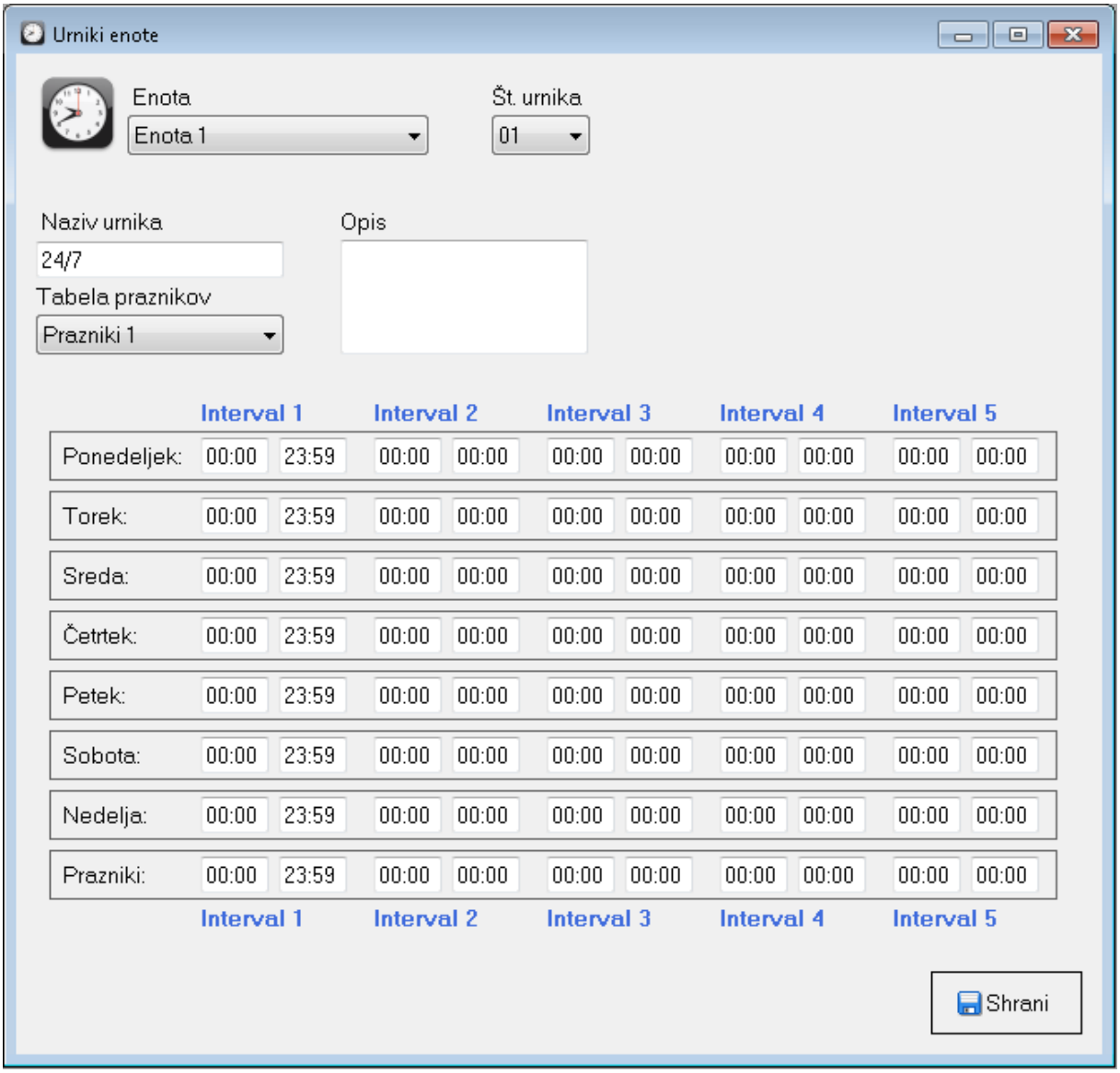
Ko se procedura `DoWork()` zaključi, je program pripravljen na shranjevanje podatkov v bazo, prej je seveda potrebno preveriti še, da ima čitalec vnešeno ime. Glede na vrednost spremenljivke `id_c`, ki predstavlja id čitalca, katerega vrednost je -1, če gre za nov čitalec, drugače pa id, ki ga ima čitalec v bazi, programska koda preveri, ali gre za vnos novega čitalca ali za posodabljanje polj obstoječega čitalca. Programska koda, ki preverja, shranjuje in posodablja čitalce je naslednja:

```
if (textBox1.Text.Length > 0)
            {
                textBox1.BackColor = Color.White;
                if (id_c == -1)
                {
                    citalciTableAdapter1.InsertCitalec(id_e, textBox1.Text,
textBox2.Text, (short)numericUpDown1.Value,
                        ok1, ok2, ner1, ner2, ur1, ur2);
                    id_c = citalciTableAdapter1.LastIDCitalec().Value;
                    button2.Enabled = false;
                    this.Close();
                }
                else
                {
                    citalciTableAdapter1.UpdateCitalec(id_e, textBox1.Text,
textBox2.Text, (short)numericUpDown1.Value,
                        ok1, ok2, ner1, ner2, ur1, ur2, id_c);
                    button2.Enabled = false;
                    this.Close();
                }
            }
            else
            {
                textBox1.BackColor = Color.Red;
            }
```

Meni "Nastavitve strojne opreme/Urniki enot" odpira novo okno, preko katerega nastavljamo urnike dostopa kontrolnih enot (slika 17). Vsaka kontrolna enota ima lahko do največ 10 različnih urnikov, zato so urniki posamezne enote označeni s številko od 01 do 10. Vsak urnik je poleg imena in opisa sestavljen iz osmih postavk (dnevi od ponedeljka do nedelje ter praznični dan), vsaka postavka pa ima lahko največ 5 časovnih intervalov (od – do). Poleg naštetega moramo vsakemu urniku določiti tabelo praznikov, kjer so zapisani prazničnidnevi, saj ob takih dnevih po navadi velja poseben urnik. Na tem mestu si oglejmo, kako programska koda na podlagi izbrane številke iz izbirnega menija (*ang. Dropdown List*) v podatkovni bazi preveri, če za to številko obstaja urnik ter nato podatke urnika prikaže v oknu. Zaradi obsežne kode polnjenja osemdesetih vnosnih polj (*ang. TextBox*), je ta koda zamenjana s komentarjem "Polnjenje podatkov posameznega urnika". Programska koda za preverjanje obstoja urnika na številki n je naslednja:

```
VratarDataSet.UrnikDataTable urnik;
            urnik =
urnikTableAdapter1.GetDataByIDEnoteIDTSCH(Convert.ToInt32(comboBox1.SelectedValue),
comboBox2.Text);

            if (urnik.Rows.Count > 0)
            {
                textBox1.Text = urnik[0].naziv_urnik;
```

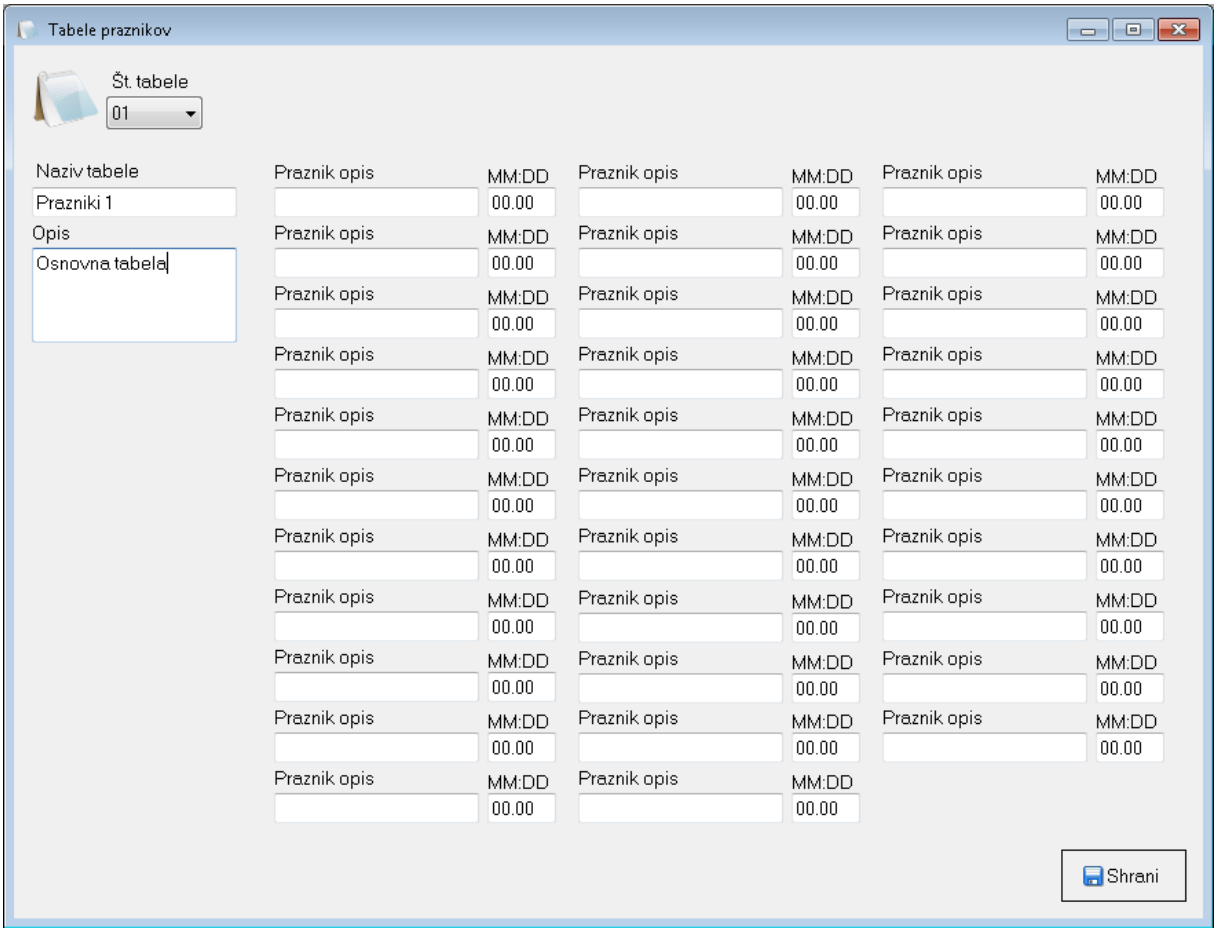
```
textBox2.Text = urnik[0].opis_urnik;
comboBox3.SelectedText = urnik[0].id_urnik.ToString();
foreach (DataRow r in urnik.Rows)
{
  // Polnjenje podatkov posameznega urnika
}
```



Slika 17: Urnik enote

Meni "Nastavitve strojne opreme/Tabele praznikov" odpira okno, kjer lahko urejamo tabele praznikov. Ob pristopu uporabnika kontrolna enota najprej preveri, če je uporabnik registriran, nato preveri datum in uro in urnik, ki ga ima uporabnik. Na podlagi teh podatkov

pa v tabeli praznikov preveri, ali gre za prazničen dan ali ne. Vsaka kontrolna enota ima lahko 10 različnih tabel praznikov, zaradi enostavne uporabe pa je implementacija tabel praznikov izvedena tako, da imajo vse kontrolne enote enake tabele praznikov. Omejitev strojne opreme je, da ima ena tabela praznikov lahko do 32 različnih datumov. V bazo podatkov pa poleg datuma shranimo še opis datuma oziroma praznika ter splošen naziv in opis tabele. Okno z vnosnimi polji prikazuje slika 18.



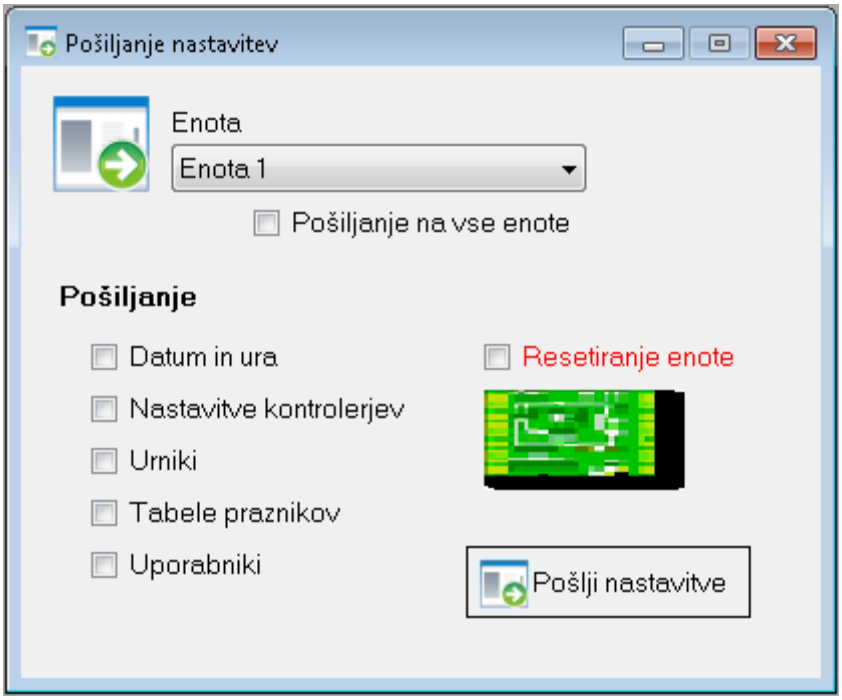
Slika 18: Tabele praznikov

Meni "Nastavitve strojne opreme/Pošiljanje nastavitev" odpira okno, ki je za kontrolo pristopa in sploh celotno delovanje najpomembnejši del aplikacije, saj se tu skriva programska koda, ki komunicira s strojno opremo. Ker je programska koda v tem oknu najpomembnejši del aplikacije, hkrati pa gre za kar nekaj vrstic kode, je ta koda predstavljena v poglavju 5.

Na oknu je izbirni seznam (*ang. Dropdown List*), kjer se nahaja seznam vseh vnešenih kontrolnih enot. Pri pošiljanju nastavitev lahko izbiramo, ali bomo nastavitve pošiljali na eno

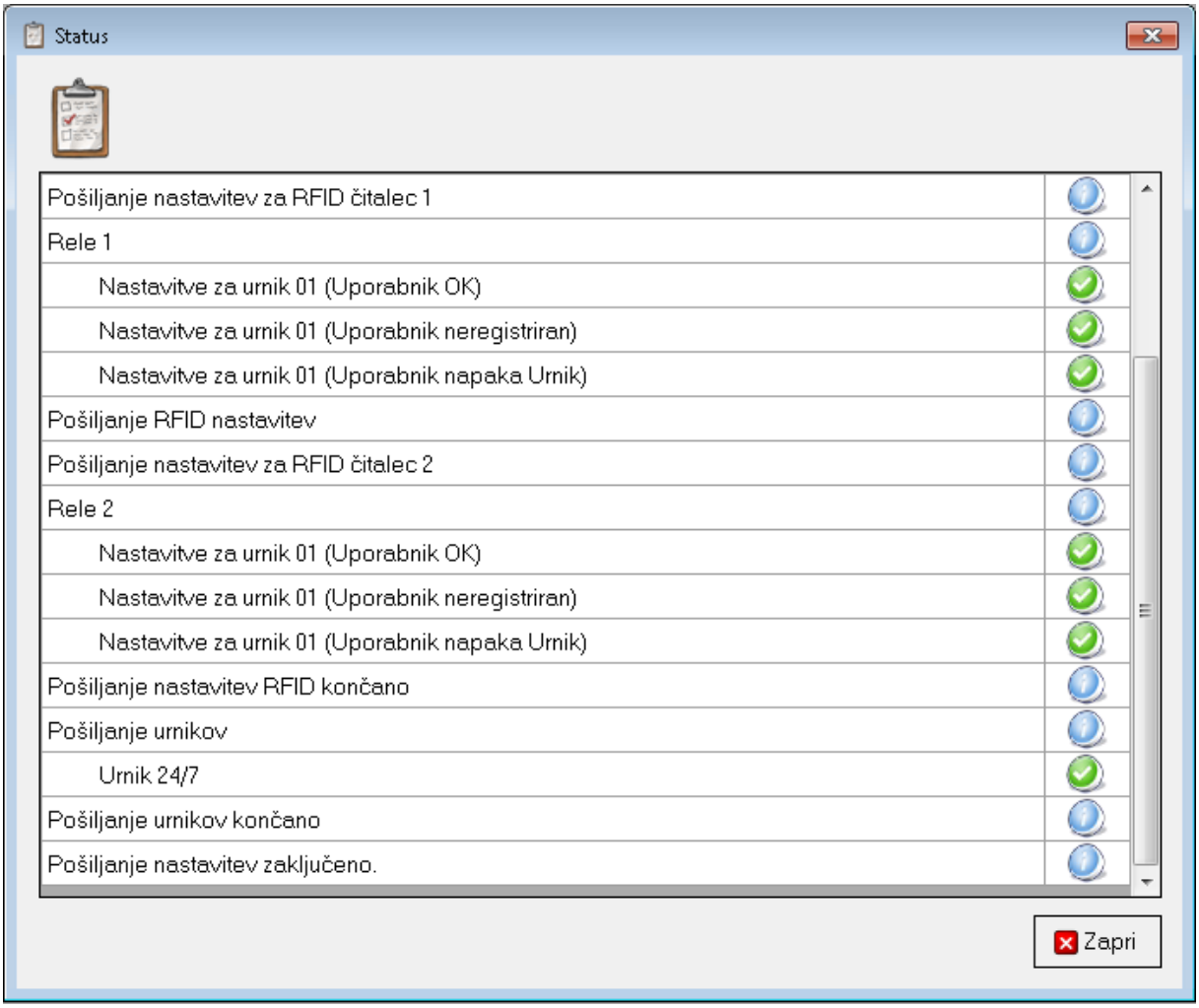
samo enoto ali pa na vse enote. To dosežemo z izbiro kljukice na polju "Pošiljanje na vse enote". Nastavitve, ki jih lahko prek tega okna pošiljamo na kontrolne enote so:

- Datum in ura
  Za pravilno delovanje urnikov kontrolna enota potrebuje pravilen datum in uro. Pri pošiljanju te nastavitve uporabljamo proceduro `NastaviDatumUro()`, ki je prikazana v prilogi 5I.
- Nastavitve kontrolerjev
  Nastavitve, ki jih nastavljamo v oknu Enote (slika 14) ter oknu "Čitalec" (slika 16), prek te nastavitve pošljemo kontrolni enoti, za kar skrbi procedura `NastaviNastavitveCitalca(int index,int id_e)` prikazana v prilogi 5C.
- Urniki
  Urniki, ki jih kontrolnim enotam nastavljamo prek okna "Urniki enote" (slika 17), se s pomočjo procedure `PosljiUrnikRocno(int id_e)`, ki je prikazana v prilogi 5D, ob izbiri te nastavitve pošljejo na kontrolno enoto.
- Tabele praznikov
  Prazniki, ki so vnešeni v oknu "Tabele praznikov" (slika 18), se s to nastavitvijo na kontrolno enoto pošljejo prek procedure `PosljiPraznik()`, ki je prikazana v prilogi 5E.
- Uporabniki
  Prek te nastavitve kontrolni enoti pošljemo vse uporabnike oziroma njihove RFID kartice, ter pripadajoče nastavitve, ki jih nastavimo prek okna "Uporabniki", ki je opisano v nadaljevanju (slika 21).
  Pošiljanje uporabnikov izvedemo s proceduro `PosljiUporabnike(int id_e)`, ki je opisana v prilogi 5G.
- Resetiranje enote
  Prek te nastavitve s pomočjo procedure `Reset()`, ki je opisana v prilogi 5F, kontrolni enoti povrnemo tovarniške nastavitve.



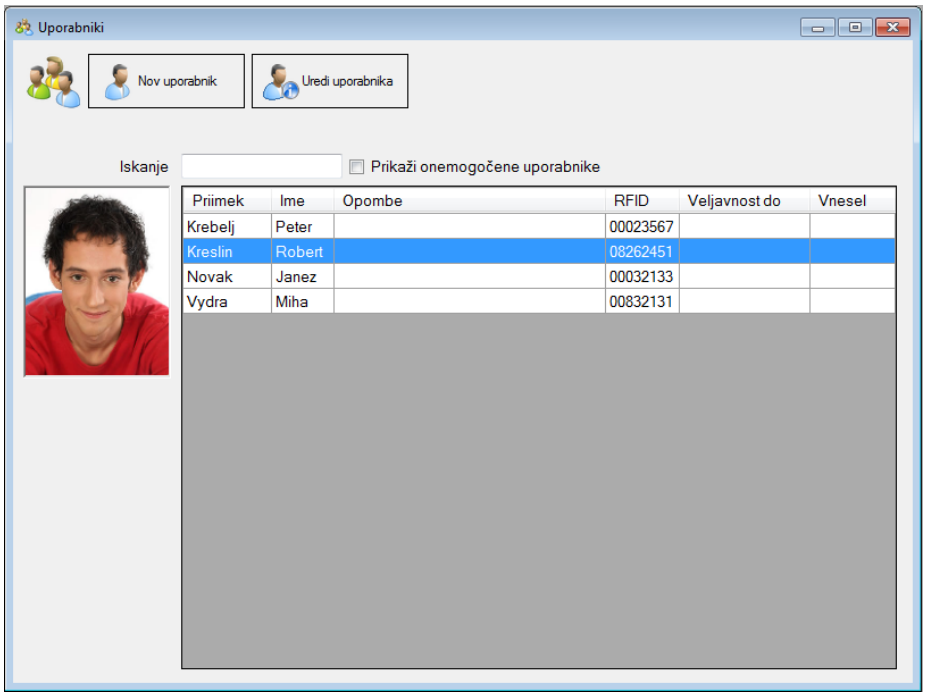
Slika 19: Pošiljanje nastavitev kontrolnim enotam

Ob pritisku gumba "Pošlji nastavitve" se glede na izbiro začnejo izvajati procedure, ki smo jih opisali, odpre se novo okno "Status", kjer je prikazan status izvedenih procedur (slika 20). Ker lahko pride pri komunikaciji s kontrolnimi enotami do napak, je v ta namen implementirana funkcija `string ProcessRezultat(string r)`, ki vrača tekstovni opis odgovora, ki ga posreduje kontrolna enota. Funkcija je opisana v prilogi 5A. Pred vsakim pošiljanjem podatkov pa se je potrebno na kontrolno enoto povezati, za kar skrbi ActiveX kontrola, ki jo uporabljamo tako, kot je prikazano v prilogi 5B.

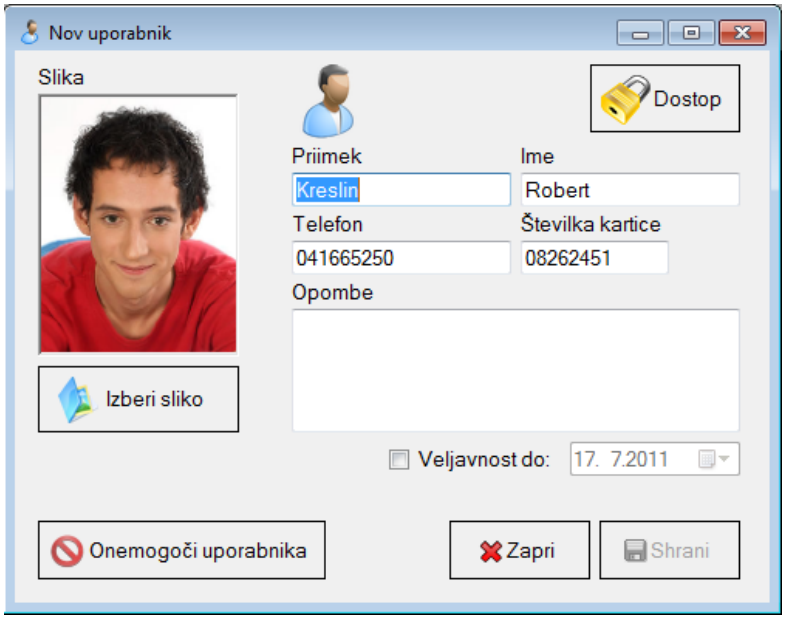


Slika 20: Status pošiljanja nastavitev

Meni "Uporabniki" odpira novo okno, kjer vidimo seznam vseh vpisanih uporabnikov kontrole pristopa oziroma gostov, ki jih lahko urejamo, lahko pa tudi dodajamo nove (slika 21). V tem oknu lahko uporabnike tudi iščemo z vnosom v polje "Iskanje". Pri urejanju ali dodajanju uporabnika se nam odpre novo okno (slika 22), kjer lahko vnesemo osnovne podatke: priimek, ime, telefon, številka kartice, veljavnost uporabnika, slika. Ob pritisku na gumb `Dostop` pa se nam odpre novo okno (slika 23), kjer imamo seznam vseh kontrolnih enot, čitalcev ter njihovih urnikov. Za dodelitev dostopa pritisnemo na želeno polje, ki se nato obarva z zeleno barvo. Urnik uporabniku za vsako enoto izbiramo posebej z izbirnim seznamom. Vsak uporabnik kontrole pristopa je lahko omogočen ali onemogočen. Ta status lahko nastavimo s tipko `Onemogoči uporabnika` (slika 22). Onemogočeni uporabniki na seznamu uporabnikov niso vidni, zato je za vpogled potrebno izbrati opcijo "Prikaži onemogočene uporabnike" (slika 21). Za avtomatično onemogočanje uporabnikov skrbi avtomatski modul, ki si ga bomo ogledali v nadaljevanju.



Slika 21: Upravljanje uporabnikov



Slika 22: Urejanje uporabnika

Slika 23: Nastavitve dostopa uporabniku

Okno za dostop je izdelano s pomočjo prikazne tabele, s kjer lahko ob različnih dogodkih, ki se izvajajo nad tabelo, sprožimo različno programsko kodo, v samo tabelo pa je moč vstavljati tudi druge objekte, kot je na primer izbirni seznam za izbiro urnika. [12] Ob shranjevanju dostopa (slika 23) nas program vpraša, če želimo uporabnika poslati na kontrolne enote. Vsakega uporabnika, ki je v sistemu dodan na novo, je seveda smiselno poslati kar ob vnosu in ne prek menija "Pošiljanje nastavitev", zato je del programske kode, ki komunicira s kontrolnimi enotami tudi na tem oknu. Če se vnašalec odloči, da želi uporabnika na kontrolne enote poslati takoj po shranjevanju, program pokliče proceduro `PosljiUporabnika(int id_u, int id_e)`, ki je podobna tisti, opisani v prilogi 5G.

3.2.5 AVTOMATSKI MODUL

Podobno kot upravljalski modul je tudi avtomatski modul razvit kot samostojna aplikacija z imenom Neveljavne kartice. Namen programa je en sam – brisanje kartic s kontrolnih enot, ki jim je potekel rok veljavnosti, ki se ga vnese ob vnosu uporabnika (slika 22). Predviden način delovanja programa "Neveljavne kartice" je, da ga zaganjamo s pomočjo opravil (*ang. Task Scheduler*) v Windowsih. Tako lahko določimo, ob kateri uri bodo kartice, ki jim je rok veljavnosti potekel, izbrisane.

3.2.5.1 IZGLED IN DELOVANJE

ZAGON PROGRAMA

Program je za razliko od aplikacije Vratar Mini izdelan tako, da mora ob zagonu kot argument prejeti pot do podatkovne baze, na kateri briše neveljavne kartice. Če uporabnik poti do podatkovne baze z argumentom ne poda, se pot do podatkovne baze nastavi kar na mapo, s katere se program zaganja. [13] [11] Programska koda, ki ob zagonu preverja argumente in nastavlja pot do podatkovne baze je naslednja:

```
public Form1(string[] args)
        {
            InitializeComponent();

            try
            {
                for (int i = 0; i < args.Length; i++)
                    p += args[i] + " ";
            }
            catch
            {
                p = System.IO.Directory.GetCurrentDirectory() + "\\Vratar.mdb";
            }

            if (p.Length==0)
                p = System.IO.Directory.GetCurrentDirectory() + "\\Vratar.mdb";

        }
```

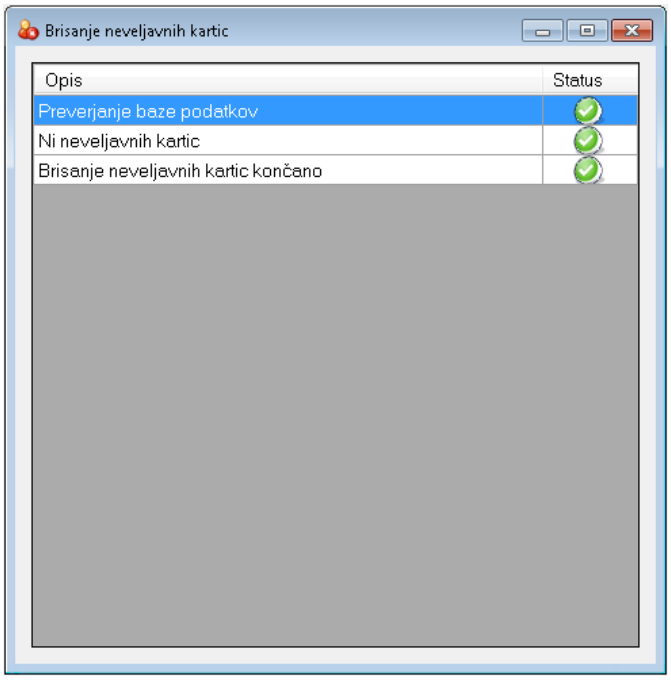
Ob preverjanju argumentov je seveda potrebno pomisliti tudi na to, da podana pot ni veljavna oziroma da ne obstaja. V takem primeru program ne more opraviti svoje naloge, zato se

samodejno zapre. Programska koda, s katero preverjamo obstoj poti do baze ter nastavljamo povezovalno pot programa (*ang. Connection String*) je naslednja:

```
if (File.Exists(p))
            {
                Properties.Settings.Default.VratarCS =
                    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + p.ToString()
                    + ";Persist Security Info=True;Encrypt Password=False;Mask
Password=False;Jet OLEDB:Database";
            }
            else
            {
                timerExit.Start();
            }
```

STATUSNO OKNO

Statusno okno uporabniku grafično prikazuje delovanje. Vsak zagon programa se najprej začne s preverjanjem poti podatkovne baze. V primeru veljavne poti do baze se v polju "Status" v statusnem oknu prikaže zelena kljukica (slika 24), v nasprotnem primeru se na mestu kljukice prikaže rdeč x, v polju opis pa opis napake (slika 25). Programska koda, ki v avtomatskem modulu briše uporabnike, je predstavljena v prilogi 5H.



Slika 24: Brisanje neveljavnih kartic - statusno okno

Slika 25: Izpis v primeru napake

3.3 UPORABLJENA STROJNA OPREMA PROIZVAJALCA IDTECK

IDTECK je eno izmed vodilnih podjetij pri izdelavi produktov za kontrolo pristopa. Njihovi produkti so v uporabi v več kot 70. državah sveta, zastopa pa jih več kot 200 distributerjev. Od leta 1989 so prevzeli vodilno vlogo na področju izdelave in razvoja opreme za kontrolo pristopa, od biometričnih izdelkov (čitalci prstnih odtisov, obraza, ...), čitalcev pametnih kartic in ostale strojne opreme, pa tudi razvoja programske opreme. Ena od prednosti je prav gotovo tudi možnost lastnega razvoja programske opreme s pomočjo njihovih vnaprej pripravljenih komponent, kot je na primer activeX kontrola za komuniciranje kontrolne enote z računalnikom - programsko opremo. Pri projektu "Aplikacija za upravljanje RFID pametnih kartic pri kontroli pristopa v hotelu" smo uporabili kontrolne enote iCON100 (na eno enoto je moč priklopiti do 2 čitalca), RFID čitalce RF TINY, RF10, RF20 in RF30, komunikacijski TCP/IP vmesnik iLAN422, RFID pametne kartice in obeske ter standardne 12 voltne električne prijemnike.

3.3.1 KONTROLNA ENOTA iCON100

Kontrolno enoto iCON100 prikazuje slika 26. iCON100 lastnosti:

- 8 bitni mikroprocesor
- od 10.000 do 50.000 uporabnikov
- od 10.000 do 50.000 dogodkov
- RS232/RS422 komunikacija
- nadzor dogodkov/stanj v realnem času
- sistem Anti-Passback
- avtomatični reset enote ob napakah s pomočjo t.i. Watchdog-a

Slika 26: Kontrolna enota IDTECK iCON100 [17]

Na kontrolno enoto lahko s pomočjo programskega vmesnika prenesemo/shranimo od 10 do 50.000 uporabnikov. Število je odvisno od nastavitvenega razmerja število uporabnikov:število shranjenih dogodkov. Večje, kot je število dogodkov, ki jih želimo shraniti v pomnilnik kontrolne enote, manjše je število uporabnikov, ki jih ima kontrolna enota shranjenih v spominu. Za komunikacijo lahko uporabimo počasnejšo RS232 ali hitrejšo RS422 komunikacijo, preko katere lahko upravljamo z enoto, nastavljamo nastavitve in v realnem času spremljamo dogajanje.

Sistem Anti-Passback je funkcija oziroma varnostni mehanizem, ki preprečuje, da bi bila pristopna kartica za vstop v varovano območje uporabljena več kot enkrat. Ko uporabnik s pristopno kartico vstopi v varovano območje, kontrolna enota tej kartici dovoljuje le še izstop in ne ponovnega vstopa. Ta funkcija je še posebej dobrodošla takrat, ko se s kontrolo pristopa preverja evidenco delovnega časa, saj s funkcijo Anti-Passback natančno vidimo, kdaj je bila oseba prisotna in kdaj odsotna. [14]

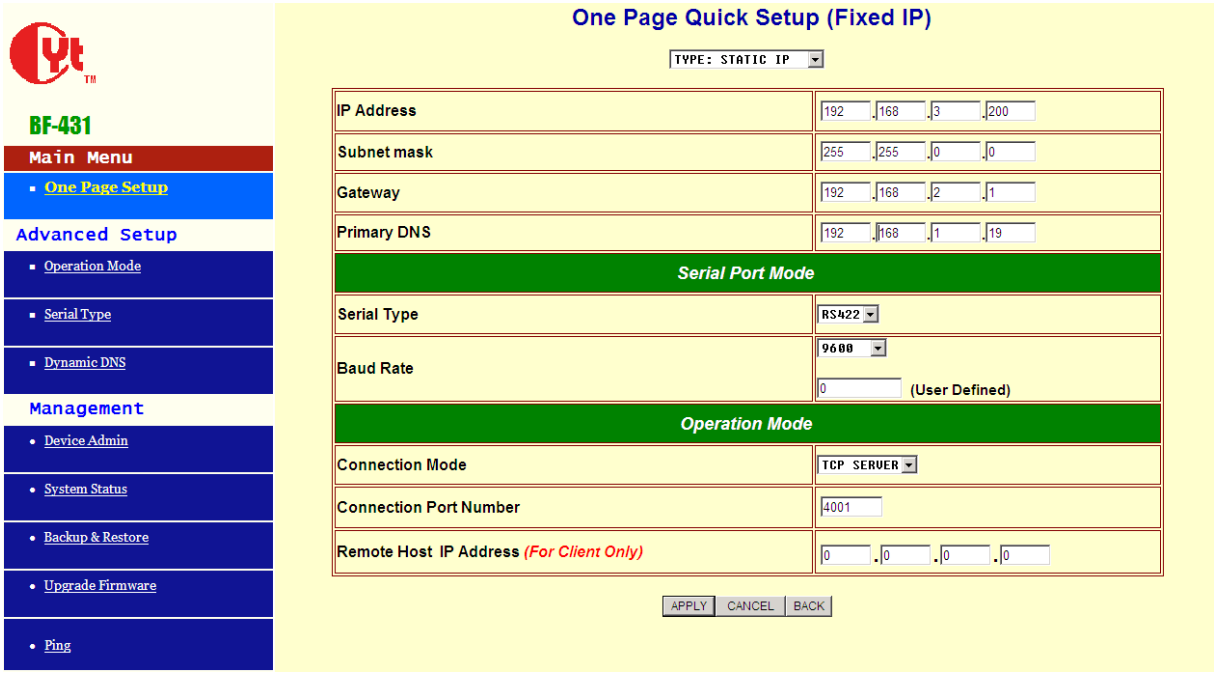
3.3.2 ILAN422 TCP/IP VMESNIK

Kontrolna enota za komunikacijo s programsko opremo uporablja RS232/RS422, vendar je v večini primerov to neprimerno, saj nas tak tip povezave omejuje. Računalnik, kjer teče programska oprema, mora imeti komunikacijsko kartico oziroma primerna vrata, poleg tega lahko z enoto upravljamo samo prek računalnika, na katerega imamo enoto priklopljeno. Zato uporabljamo mrežni TCP/IP vmesnik, ki nam omogoča, da z enoto upravljamo prek lokalnega

omrežja. Podjetij, ki ponujajo take vmesnike je sicer več, IDTECK pa priporoča uporabo posebej za to narejenega vmesnika, ki so ga poimenovali ILAN422 in je prikazan na sliki 27.

Slika 27: TCP/IP vmesnik za komunikacijo s kontrolno enoto prek lokalnega omrežja [19]

Nastavitve mrežnega vmesnika spreminjamo s pomočjo vgrajene spletne strani (slika 28), ki nam omogoča spremembe različnih parametrov, od IP naslova, tipa komunikacije, načina delovanja itd.



Slika 28: Spletna stran vmesnika ILAN422

3.3.3 RFID ČITALCI PAMETNIH KARTIC

Za preverjanje identitete uporabnikov uporabljamo RFID čitalce pametnih kartic. IDTECK ponuja vrsto različnih čitalcev z različnimi lastnostmi, uporabljeni pa so bili čitalci tipa RF10, RF20, RF TINY (slika 29) ter RF30 (slika 30).

Slika 29: IDTECK čitalci RFID pametnih kartic [20]

RF10, RF20 ter RF TINY so čitalci enakega tipa, ki se razlikujejo le po velikosti, delujejo na 125KHz, primerni so tako za notranjo kot tudi zunanjo uporabo, saj so vodoodporni, vgrajen imajo zvočnik za pisk, ki ga prek kontrolne enote lahko uporabljamo, ravno tako lahko spreminjamo barve led diod (rdeča/zelena). Delujejo na razdalji od 7-10cm, odvisno od tipa kartice.

Slika 30: IDTECK RF30, čitalec z dometom do 30cm [20]

RF30 pa se od zgoraj omenjenih čitalcev razlikuje predvsem po razdalji delovanja, saj v primerjavi z zgornjimi čitalci RF30 zazna kartice do razdalje 30 cm, kar je predvsem uporabno v situacijah, kot je na primer dovoz v garažo, kjer mora uporabnik kartico iz avtomobila približati čitalcu. RF30 je bil uporabljen za odpiranje dvižne zapornice pred hotelom.

3.3.4 RFID PAMETNE KARTICE

Za preverjanje identitete se pri projektu uporabljajo RFID pametne kartice različnih velikosti. IDC170 je format kartice primeren za nošenje okrog vratu oziroma na srajci. IDC80 je kartica velikosti kreditne kartice, kar je najbolj običajen tip RFID pametne kartice. Vse bolj popularne so kartice velikosti obeska za ključe, kot na primer IDK50, za posebne primere pa so na voljo tudi manjši formati, kot je na primer pametna kartica IMC125 v velikosti evrskega kovanca.

Slika 31: IDTECK RFID pametne kartice [21]

Slika 31 prikazuje različne velikosti RFID kartic. Vse kartice delujejo na frekvenci 125KHz in nimajo lastnega napajanja, saj delujejo po principu pasivnega RFID, ki za svoje delovanje prejme energijo od signala, ki se inducira v anteni.

Skozi celotno implementacijo kontrole pristopa v hotelu smo ugotovili, da moramo celotno področje zares dobro poznati, saj nam zgolj programersko znanje ne zadošča. V prvi vrsti se moramo seznaniti s strojno opremo in njenim delovanjem, da lahko kasneje z njo vzpostavimo komunikacijo ter razumemo celoten proces delovanja. Prav tako moramo spoznati in preizkusiti obnašanje tako strojne, kot programske opreme v ekstremnih situacijah (odpoved električnega toka, odpoved komunikacije), kar pa brez realnega testiranja ni mogoče.

4 ZAKLJUČEK

Cilj diplomske naloge je bil razvoj aplikacije za upravljanje RFID pametnih kartic ter implementacija kontrole pristopa, ki bi podjetju (hotelu) olajšala delo v administraciji, povečala varnost objekta in odpravila 24 urno prisotnost varnostnika oziroma receptorja.

Tako razvojno orodje Microsoft .NET, kot izbrana strojna oprema proizvajalca IDTECK, sta bila prava izbira, saj se je izkazalo, da je .NET ogrodje zares enostavno in zmogljivo, ter da je moč do nekaterih rešitev priti že z nekaj vrsticami programske kode. Proizvajalec IDTECK ni zaman vodilno podjetje v svojem segmentu, saj strojna oprema deluje odlično, njihova ActiveX povezovalna kontrola pa je bila v veliko pomoč pri implementaciji komunikacije v aplikaciji.

Med razvojem aplikacije je bila potrebna stalna komunikacija z naročniki, saj so se vprašanja odpirala skozi celoten razvoj. Tako smo tudi zaradi komunikacije že med razvojem dodali nekaj funkcionalnosti, kot je na primer možnost nadzora nad več podjetji.

Razvoj aplikacij Vratar Mini in Neveljavne kartice je trajal približno 3 mesece. Največ časa je seveda vzela sama komunikacija s strojno opremo in povezovanje ActiveX komponente. Potrebno je bilo raziskati, kako deluje komunikacija, kako se strojna oprema obnaša v ekstremnih situacijah, ter kakšne so možne komunikacijske in druge napake, ki se lahko med delovanjem pripetijo.

Po zaključku razvoja aplikacije se je izkazalo, da je možnosti za nadaljnji razvoj aplikacij za kontrolo pristopa veliko. V razviti aplikaciji ni statistike dogodkov, torej pristopne statistike, saj naročnik za svoje delo tega ni potreboval, kljub temu, da izbrana strojna oprema to omogoča. Vsekakor je statistika dogodkov naslednji korak pri nadgradnji obstoječe aplikacije, ki je že v razvoju. Predvsem zanimiv segment pa je tudi avtomatski obračun delovnih ur.

5 PRILOGE

A) FUNKCIJA ZA TOLMAČENJE REZULTATOV

```
public string ProcessRezultat(string r)
        {
            switch (r)
            {
                case "Z":
                    return "Uspešno";
                case "C":
                    return "Uporabnik že izbrisan";
                case "0":
                    return "Napaka pri komunikaciji (NACK)";
                case "A":
                    return "Napaka Checksum";
                case "D":
                    return "ID ne obstaja";
                case "H":
                    return "Napaka v dolžini podatkov";
                case "I":
                    return "Number Owerflow";
                case "Q":
                    return "Master Flag Error";
                case "B":
                    return "Spomin kontrolne enote je poln";
                default:
                    return "Neznana napaka";
            }
```

B) POVEZOVANJE NA KONTROLNO ENOTO

```
StarInterface1.SocketInitialize();
StarInterface1.WorkIndex = 0;
StarInterface1.BoardIndex = r["BOARD_ID"].ToString();

if (StarInterface1.SocketConnect(r["IP_ADDRESS"].ToString(), Convert.ToInt16(
                        r["TCP_PORT"].ToString())))
{
  PosljiUporabnika(id_uporabnik, Convert.ToInt32(r["id_enota"]));
  StarInterface1.SocketClose();
}
```

C) POŠILJANJE NASTAVITEV KONTROLNI ENOTI

```
private void NastaviNastavitveCitalca(int index,int id_e)
        {
```

```
            s.dataGridView1.Invoke(new EventHandler(delegate
                {
                    s.dataGridView1.Rows.Add("Pošiljanje RFID nastavitev",
s.imageList1.Images[0]);

                    //anti_passback
                    if (index == 1)
                    {
                        string ares = "";
                        if
((bool)enoteTableAdapter.GetDataByIDEnote(id_e)[0].ANTI_PASSBACK)
                        {
                            s.dataGridView1.Rows.Add("Pošiljanje nastavitev za Anti
PassBack", s.imageList1.Images[0]);

                            ares = StarInterface1.AntiPassbackDownload("1");

                            if (idtek.ProcessRezultat(ares) == "Uspešno")
                                s.dataGridView1.Rows.Add("          Nastavljanje
nastavitev za AntiPassBack", s.imageList1.Images[1]);
                            else
                            {
                                s.dataGridView1.Rows.Add("          Nastavljanje
nastavitev za AntiPassBack", s.imageList1.Images[2]);
                                s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(ares);
                            }

                        }
                        else
                        {

                            s.dataGridView1.Rows.Add("Pošiljanje nastavitev za Anti
PassBack", s.imageList1.Images[0]);

                            ares = StarInterface1.AntiPassbackDownload("0");

                            if (idtek.ProcessRezultat(ares) == "Uspešno")
                                s.dataGridView1.Rows.Add("          Brisanje
nastavitev za AntiPassBack", s.imageList1.Images[1]);
                            else
                            {
                                s.dataGridView1.Rows.Add("          Brisanje
nastavitev za AntiPassBack", s.imageList1.Images[2]);
                                s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(ares);
                            }
                        }
                    }
                    //anti_passback
```

```
                    s.dataGridView1.Rows.Add("Pošiljanje nastavitev za RFID čitalec
" + index.ToString(), s.imageList1.Images[0]);
                    string result;
                    VratarDataSet.CitalciDataTable tab;
                    tab = citalciTableAdapter1.GetDataByIDEnotaRNR(id_e,
(short)index);

                    if (tab.Rows.Count > 0)
                    {
                        if (index == 1)
                        {
                            s.dataGridView1.Rows.Add("Rele 1",
s.imageList1.Images[0]);
                            VratarDataSet.UrnikDataTable urniki_enote;
                            urniki_enote =
urnikTableAdapter1.GetDataByIDenota(id_e);

                            foreach (DataRow r in urniki_enote.Rows)
                            {
                                string sch = r["ID_TECK_SCHEDULE"].ToString();
                                //uporabnik ok
                                result = StarInterface1.InOutputTableDownload("06",
sch, tab[0].OK1.ToString(),
                                    tab[0].OK2.ToString(), "00", "00", "00");

                                if (idtek.ProcessRezultat(result) == "Uspešno")
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik OK)", s.imageList1.Images[1]);
                                else
                                {
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik OK)", s.imageList1.Images[2]);
                                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count
- 1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                                }

                                //uporabnik neregistriran
                                result = StarInterface1.InOutputTableDownload("07",
sch, tab[0].UNREG1.ToString(),
                                    tab[0].UNREG2.ToString(), "00", "00", "00");

                                if (idtek.ProcessRezultat(result) == "Uspešno")
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik neregistriran)", s.imageList1.Images[1]);
                                else
                                {
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik neregistriran)", s.imageList1.Images[2]);
                                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count
- 1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
```

```
                                }
                                //urnik napaka
                                result = StarInterface1.InOutputTableDownload("08",
sch, tab[0].TIMESCH1.ToString(),
                                    tab[0].TIMESCH2.ToString(), "00", "00", "00");

                                if (idtek.ProcessRezultat(result) == "Uspešno")
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik napaka Urnik)", s.imageList1.Images[1]);
                                else
                                {
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik napaka Urnik)", s.imageList1.Images[2]);
                                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count
- 1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                                }
                            }
                        }
                        else if (index == 2)
                        {
                            s.dataGridView1.Rows.Add("Rele 2",
s.imageList1.Images[0]);
                            VratarDataSet.UrnikDataTable urniki_enote;
                            urniki_enote =
urnikTableAdapter1.GetDataByIDenota(id_e);

                            foreach (DataRow r in urniki_enote.Rows)
                            {
                                string sch = r["ID_TECK_SCHEDULE"].ToString();
                                //uporabnik ok
                                result = StarInterface1.InOutputTableDownload("10",
sch, tab[0].OK1.ToString(),
                                    tab[0].OK2.ToString(), "00", "00", "00");

                                if (idtek.ProcessRezultat(result) == "Uspešno")
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik OK)", s.imageList1.Images[1]);
                                else
                                {
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik OK)", s.imageList1.Images[2]);
                                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count
- 1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                                }

                                //uporabnik neregistriran
                                result = StarInterface1.InOutputTableDownload("11",
sch, tab[0].UNREG1.ToString(),
                                    tab[0].UNREG2.ToString(), "00", "00", "00");

                                if (idtek.ProcessRezultat(result) == "Uspešno")
```

```
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik neregistriran)", s.imageList1.Images[1]);
                                else
                                {
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik neregistriran)", s.imageList1.Images[2]);
                                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count
- 1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                                }
                                //urnik napaka
                                result = StarInterface1.InOutputTableDownload("12",
sch, tab[0].TIMESCH1.ToString(),
                                    tab[0].TIMESCH2.ToString(), "00", "00", "00");

                                if (idtek.ProcessRezultat(result) == "Uspešno")
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik napaka Urnik)", s.imageList1.Images[1]);
                                else
                                {
                                    s.dataGridView1.Rows.Add("          Nastavitve
za urnik " + sch + " (Uporabnik napaka Urnik)", s.imageList1.Images[2]);
                                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count
- 1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                                }
                            }
                            s.dataGridView1.Rows.Add("Pošiljanje nastavitev RFID
končano", s.imageList1.Images[0]);
                        }
                        else
                            s.dataGridView1.Rows.Add("Napaka",
s.imageList1.Images[2]);

                    }
                    else
                        s.dataGridView1.Rows.Add("Ni priklopljenega čitalca",
s.imageList1.Images[0]);
                }));
        }
```

D) POŠILJANJE URNIKA KONTROLNI ENOTI

```
public void PosljiUrnikRocno(int id_e)
        {
            s.dataGridView1.Invoke(new EventHandler(delegate
                {

                    s.dataGridView1.Rows.Add("Pošiljanje urnikov",
s.imageList1.Images[0]);

                    string result;
```

```
                    VratarDataSet.EnoteDataTable tabelaEnota;
                    tabelaEnota = enoteTableAdapter.GetDataByIDEnote(id_e);

                    VratarDataSet.UrnikDataTable urnikEnote;
                    urnikEnote = urnikTableAdapter1.GetDataByIDenota(id_e);
                    urnik_nr.Clear();
                    foreach (DataRow r in urnikEnote.Rows)
                        urnik_nr.Add(r["ID_TECK_SCHEDULE"].ToString());

                    foreach (string nr in urnik_nr)
                    {
                        VratarDataSet.UrnikDataTable urnik;
                        urnik = urnikTableAdapter1.GetDataByIDEnoteIDTSCH(id_e,
nr);

                        string schedule = "";
                        string ned = "", pon = "", tor = "", sre = "", cet = "",
pet = "", sob = "", pra = "";
                        foreach (DataRow row in urnik.Rows)
                        {
                            switch (row["dan_urnik"].ToString())
                            {
                                case "Pon":
                                    pon = pon + row["sch11"].ToString();
                                    pon = pon + row["sch12"].ToString();
                                    pon = pon + row["sch21"].ToString();
                                    pon = pon + row["sch22"].ToString();
                                    pon = pon + row["sch31"].ToString();
                                    pon = pon + row["sch32"].ToString();
                                    pon = pon + row["sch41"].ToString();
                                    pon = pon + row["sch42"].ToString();
                                    pon = pon + row["sch51"].ToString();
                                    pon = pon + row["sch52"].ToString();
                                    break;
                                case "Tor":
                                    tor = tor + row["sch11"].ToString();
                                    tor = tor + row["sch12"].ToString();
                                    tor = tor + row["sch21"].ToString();
                                    tor = tor + row["sch22"].ToString();
                                    tor = tor + row["sch31"].ToString();
                                    tor = tor + row["sch32"].ToString();
                                    tor = tor + row["sch41"].ToString();
                                    tor = tor + row["sch42"].ToString();
                                    tor = tor + row["sch51"].ToString();
                                    tor = tor + row["sch52"].ToString();
                                    break;
                                case "Sre":
                                    sre = sre + row["sch11"].ToString();
                                    sre = sre + row["sch12"].ToString();
                                    sre = sre + row["sch21"].ToString();
                                    sre = sre + row["sch22"].ToString();
```

```
            sre = sre + row["sch31"].ToString();
            sre = sre + row["sch32"].ToString();
            sre = sre + row["sch41"].ToString();
            sre = sre + row["sch42"].ToString();
            sre = sre + row["sch51"].ToString();
            sre = sre + row["sch52"].ToString();
            break;
        case "Cet":
            cet = cet + row["sch11"].ToString();
            cet = cet + row["sch12"].ToString();
            cet = cet + row["sch21"].ToString();
            cet = cet + row["sch22"].ToString();
            cet = cet + row["sch31"].ToString();
            cet = cet + row["sch32"].ToString();
            cet = cet + row["sch41"].ToString();
            cet = cet + row["sch42"].ToString();
            cet = cet + row["sch51"].ToString();
            cet = cet + row["sch52"].ToString();
            break;
        case "Pet":
            pet = pet + row["sch11"].ToString();
            pet = pet + row["sch12"].ToString();
            pet = pet + row["sch21"].ToString();
            pet = pet + row["sch22"].ToString();
            pet = pet + row["sch31"].ToString();
            pet = pet + row["sch32"].ToString();
            pet = pet + row["sch41"].ToString();
            pet = pet + row["sch42"].ToString();
            pet = pet + row["sch51"].ToString();
            pet = pet + row["sch52"].ToString();
            break;
        case "Sob":
            sob = sob + row["sch11"].ToString();
            sob = sob + row["sch12"].ToString();
            sob = sob + row["sch21"].ToString();
            sob = sob + row["sch22"].ToString();
            sob = sob + row["sch31"].ToString();
            sob = sob + row["sch32"].ToString();
            sob = sob + row["sch41"].ToString();
            sob = sob + row["sch42"].ToString();
            sob = sob + row["sch51"].ToString();
            sob = sob + row["sch52"].ToString();
            break;
        case "Ned":
            ned = ned + row["sch11"].ToString();
            ned = ned + row["sch12"].ToString();
            ned = ned + row["sch21"].ToString();
            ned = ned + row["sch22"].ToString();
            ned = ned + row["sch31"].ToString();
            ned = ned + row["sch32"].ToString();
            ned = ned + row["sch41"].ToString();
```

```
                                    ned = ned + row["sch42"].ToString();
                                    ned = ned + row["sch51"].ToString();
                                    ned = ned + row["sch52"].ToString();
                                    break;
                                case "Pra":
                                    pra = pra + row["sch11"].ToString();
                                    pra = pra + row["sch12"].ToString();
                                    pra = pra + row["sch21"].ToString();
                                    pra = pra + row["sch22"].ToString();
                                    pra = pra + row["sch31"].ToString();
                                    pra = pra + row["sch32"].ToString();
                                    pra = pra + row["sch41"].ToString();
                                    pra = pra + row["sch42"].ToString();
                                    pra = pra + row["sch51"].ToString();
                                    pra = pra + row["sch52"].ToString();
                                    break;
                            }

                        }
                        //schedule = ned+pon+tor+sre+cet+pet+sob+pra;
                        schedule = pra + ned + pon + tor + sre + cet + pet + sob;
                        if (schedule.Length == 0)
                        {
                            for (int i = 0; i < 8; i++)
                            {
                                schedule = schedule +
"0000000000000000000000000000000000000000";
                            }
                            result =
idtek.ProcessRezultat(StarInterface1.TimeScheduleDownload(nr, schedule,

praznikiTableAdapter1.GetDataByIDHolidayCode(urnik[0].id_praznik)[0].ID_TECK_HOLIDA
Y_CODE.ToString()));

                            if (result == "Uspešno")
                                s.dataGridView1.Rows.Add("          Urnik ni
izdelan, brisanje polj urnika na enoti", s.imageList1.Images[1]);
                            else
                            {
                                s.dataGridView1.Rows.Add("          Urnik ni
izdelan, brisanje polj urnika na enoti", s.imageList1.Images[2]);
                                s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                            }
                        }

                        result =
idtek.ProcessRezultat(StarInterface1.TimeScheduleDownload(nr, schedule,

praznikiTableAdapter1.GetDataByIDHolidayCode(urnik[0].id_praznik)[0].ID_TECK_HOLIDA
Y_CODE.ToString()));
```

```
                        if (result == "Uspešno")
                            s.dataGridView1.Rows.Add("          Urnik " +
urnik[0].naziv_urnik.ToString(), s.imageList1.Images[1]);
                        else
                        {
                            s.dataGridView1.Rows.Add("          Urnik " +
urnik[0].naziv_urnik.ToString(), s.imageList1.Images[2]);
                            s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                        }
                    }

                    s.dataGridView1.Rows.Add("Pošiljanje urnikov končano",
s.imageList1.Images[0]);
                }));
        }
```

E) POŠILJANJE TABEL PRAZNIKOV KONTROLNI ENOTI

```
private void PosljiPraznik()
        {
            s.dataGridView1.Invoke(new EventHandler(delegate
                {

                    string result;
                    VratarDataSet.PraznikiDataTable tab;
                    tab = praznikiTableAdapter1.GetDataByIDProfilOnly(id_p);

                    s.dataGridView1.Rows.Add("Pošiljanje tabele praznikov",
s.imageList1.Images[0]);

                    foreach (DataRow r in tab.Rows)
                    {
                        VratarDataSet.PraznikiDataTable tabela;
                        tabela = praznikiTableAdapter1.GetDataByIDProfil(id_p,
r["ID_TECK_HOLIDAY_CODE"].ToString());
                        string praznik_string = "";

                        foreach (DataRow row in tabela.Rows)
                        {
                            for (int i = 1; i < 33; i++)
                            {
                                praznik_string = praznik_string + row["d" +
i.ToString()].ToString();
                            }

                        }
```

```
                        if (praznik_string.Length == 0)
                        {
                            for (int j = 1; j < 33; j++)
                            {
                                praznik_string = praznik_string + "0000";
                            }
                            result =
idtek.ProcessRezultat(StarInterface1.HolidayDownload(r["ID_TECK_HOLIDAY_CODE"].ToSt
ring(), praznik_string));

                            if (result == "Uspešno")
                                s.dataGridView1.Rows.Add("          Tabela
praznikov ni izdelana, brisanje tabele praznikov na enoti",
s.imageList1.Images[1]);
                            else
                            {
                                s.dataGridView1.Rows.Add("          Tabela
praznikov ni izdelana, brisanje tabele praznikov na enoti",
s.imageList1.Images[2]);
                                s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                            }
                        }


                        result =
idtek.ProcessRezultat(StarInterface1.HolidayDownload(r["ID_TECK_HOLIDAY_CODE"].ToSt
ring(), praznik_string));

                        if (result == "Uspešno")
                            s.dataGridView1.Rows.Add("          Pošiljanje tabele:
" + r["ime_praznik"].ToString(), s.imageList1.Images[1]);
                        else
                        {
                            s.dataGridView1.Rows.Add("          Pošiljanje tabele:
" + r["ime_praznik"].ToString(), s.imageList1.Images[2]);
                            s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                        }

                    }

                    s.dataGridView1.Rows.Add("Pošiljanje tabele praznikov končano",
s.imageList1.Images[0]);

                }));

        }
```

F) RESET KONTROLNE ENOTE

```
public void Reset()
        {
            string result =
idtek.ProcessRezultat(StarInterface1.SystemInitialize("1"));
            s.dataGridView1.Invoke(new EventHandler(delegate
            {
                if (result == "Uspešno")
                    s.dataGridView1.Rows.Add("          Inicializacija enote",
s.imageList1.Images[1]);
                else
                {
                    s.dataGridView1.Rows.Add("          Inicializacija enote",
s.imageList1.Images[2]);
                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                }

                result =
idtek.ProcessRezultat(StarInterface1.SystemInitialize("2"));
                if (result == "Uspešno")
                    s.dataGridView1.Rows.Add("          Brisanje RFID kartic",
s.imageList1.Images[1]);
                else
                {
                    s.dataGridView1.Rows.Add("          Brisanje RFID kartic",
s.imageList1.Images[2]);
                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                }

                result =
idtek.ProcessRezultat(StarInterface1.SystemInitialize("3"));
                if (result == "Uspešno")
                    s.dataGridView1.Rows.Add("          Brisanje dogodkov",
s.imageList1.Images[1]);
                else
                {
                    s.dataGridView1.Rows.Add("          Brisanje dogodkov",
s.imageList1.Images[2]);
                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                }

                result =
idtek.ProcessRezultat(StarInterface1.SystemInitialize("4"));
                if (result == "Uspešno")
                    s.dataGridView1.Rows.Add("          Brisanje urnikov",
s.imageList1.Images[1]);
                else
```

```
                {
                    s.dataGridView1.Rows.Add("          Brisanje urnikov",
s.imageList1.Images[2]);
                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                }

                s.dataGridView1.Rows.Add("Reset enote končan",
s.imageList1.Images[0]);
            }));
        }
```

G) POŠILJANJE UPORABNIKOV KONTROLNI ENOTI

```
private void PosljiUporabnike(int id_e)
        {
            s.dataGridView1.Invoke(new EventHandler(delegate
                {
                    s.dataGridView1.Rows.Add("Pošiljanje uporabnikov",
s.imageList1.Images[0]);
                    string result;
                    VratarDataSet.UporabnikiDataTable uporabniki;
                    uporabniki = uporabnikiTableAdapter1.GetDataByOmogocenUp(id_p,
true);

                    VratarDataSet.Uporabniki_dostopDataTable dostop;
                    dostop = uporabniki_dostopTableAdapter1.GetDataByIDEnota(id_e);

                    result =
idtek.ProcessRezultat(StarInterface1.SystemInitialize("2"));

                    if (result == "Uspešno")
                        s.dataGridView1.Rows.Add("          Brisanje vseh RFID
kartic", s.imageList1.Images[1]);
                    else
                    {
                        s.dataGridView1.Rows.Add("          Brisanje vseh RFID
kartic", s.imageList1.Images[2]);
                    }


                    foreach (DataRow d in dostop.Rows)
                    {
                        foreach (DataRow uporabnik in uporabniki.Rows)
                        {
                            string reader = "0";
                            if ((bool)d["citalec1_vstop"])
                                reader = "1";
                            if ((bool)d["citalec2_vstop"])
                                reader = "2";
```

```
                            if ((bool)d["citalec1_vstop"] &&
(bool)d["citalec2_vstop"])
                                reader = "3";

                            if ((d["id_uporabnik"].ToString() ==
uporabnik["id_uporabnik"].ToString()))
                            {
                                if (reader != "0")
                                {
                                    result =
StarInterface1.CardIDDownload(uporabnik["ID_TECK_RFID"].ToString(), "0000",
                                        d["ID_TECK_SCHEDULE"].ToString(), reader,
"1");

                                    if (idtek.ProcessRezultat(result) == "Uspešno")
                                        s.dataGridView1.Rows.Add("
Dodajanje uporabnika: " + uporabnik["ime_uporabnik"].ToString() +
                                        " " + uporabnik["priimek_uporabnik"],
s.imageList1.Images[1]);
                                    else
                                    {
                                        s.dataGridView1.Rows.Add("
Dodajanje uporabnika: " + uporabnik["ime_uporabnik"].ToString() +
                                        " " + uporabnik["priimek_uporabnik"],
s.imageList1.Images[2]);

s.dataGridView1.Rows[s.dataGridView1.Rows.Count - 1].Cells[1].ToolTipText =
idtek.ProcessRezultat(result);
                                    }

                                }
                            }
                        }

                        if (dostop.Rows.Count < 1)
                            s.dataGridView1.Rows.Add("          Ni uporabnikov za
izbrano enoto.", s.imageList1.Images[0]);


                        s.dataGridView1.Rows.Add("Pošiljanje uporabnikov končano.",
s.imageList1.Images[0]);

                }));
        }
```

H) BRISANJE NEVELJAVNIH UPORABNIKOV

```
VratarDataSet.UporabnikiDataTable up;
            up = uporabnikiTableAdapter1.GetNeveljavni();
```

```
            if (up.Rows.Count==0)
                dataGridView1.Rows.Add("Ni neveljavnih kartic",
imageList1.Images[0]);



            VratarDataSet.EnoteDataTable enote;

            foreach (DataRow upr in up.Rows)
            {
                enote =
enoteTableAdapter1.GetEnoteByProfil(Convert.ToInt32(upr["id_profil"]));
                foreach (DataRow er in enote.Rows)
                {
                    StarInterface1.SocketInitialize();
                    StarInterface1.WorkIndex = 0;
                    StarInterface1.BoardIndex = er["BOARD_ID"].ToString();

                    if (StarInterface1.SocketConnect(er["IP_ADDRESS"].ToString(),
Convert.ToInt16(
                        er["TCP_PORT"].ToString()))
                    {
                        string result =
StarInterface1.CardIDDelete(upr["ID_TECK_RFID"].ToString());

                        if (idtek.ProcessRezultat(result) == "Uspešno")
                        {
                            dataGridView1.Rows.Add("          Brisanje uporabnika "
+ upr["ime_uporabnik"].ToString() +
                            " " + upr["priimek_uporabnik"].ToString(),
imageList1.Images[0]);


uporabnikiTableAdapter1.UpdateOnemogociUporabnika(Convert.ToInt32(upr["id_uporabnik
"]));
                        }
                        else
                        {
                            dataGridView1.Rows.Add("          Brisanje uporabnika "
+ upr["ime_uporabnik"].ToString() +
                            " " + upr["priimek_uporabnik"].ToString(),
imageList1.Images[1]);
                            dataGridView1.Rows[dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                            if (idtek.ProcessRezultat(result)=="Uporabnik že
izbrisan")

uporabnikiTableAdapter1.UpdateOnemogociUporabnika(Convert.ToInt32(upr["id_uporabnik
"]));
```

```
                    }
                    StarInterface1.SocketClose();
                }
                else
                    dataGridView1.Rows.Add("Povezave z enoto ni bilo mogoče
vzpostaviti", imageList1.Images[1]);
            }
        }

        dataGridView1.Rows.Add("Brisanje neveljavnih kartic končano",
imageList1.Images[0]);
        timerExit.Start();
    }
```

I) POŠILJANJE DATUMA IN URE KONTROLNI ENOTI

```
private void NastaviDatumUro()
    {
        string result;
        string date = DateTime.Now.Year.ToString();

        if (DateTime.Now.Month.ToString().Length < 2)
            date = date + "0" + DateTime.Now.Month.ToString();
        else
            date = date + DateTime.Now.Month.ToString();
        if (DateTime.Now.Day.ToString().Length < 2)
            date = date + "0" + DateTime.Now.Day.ToString();
        else
            date = date + DateTime.Now.Day.ToString();
        string week = DateTime.Now.DayOfWeek.ToString();
        switch (week)
        {
            case "Sunday":
                week = "1";
                break;
            case "Monday":
                week = "2";
                break;
            case "Tuesday":
                week = "3";
                break;
            case "Wednesday":
                week = "4";
                break;
            case "Thursday":
                week = "5";
                break;
            case "Friday":
                week = "6";
                break;
```

```
                case "Saturday":
                    week = "7";
                    break;
                default:
                    week = "8";
                    break;

            }
            string time;
            if (DateTime.Now.TimeOfDay.Hours.ToString().Length < 2)
                time = "0" + DateTime.Now.TimeOfDay.Hours.ToString();
            else
                time = DateTime.Now.TimeOfDay.Hours.ToString();

            if (DateTime.Now.TimeOfDay.Minutes.ToString().Length < 2)
                time = time + "0" + DateTime.Now.TimeOfDay.Minutes.ToString();
            else
                time = time + DateTime.Now.TimeOfDay.Minutes.ToString();

            if (DateTime.Now.TimeOfDay.Seconds.ToString().Length < 2)
                time = time + "0" + DateTime.Now.TimeOfDay.Seconds.ToString();
            else
                time = time + DateTime.Now.TimeOfDay.Seconds.ToString();

            result = idtek.ProcessRezultat(StarInterface1.DateTimeDownload(date,
week, time));
            s.dataGridView1.Invoke(new EventHandler(delegate
            {
                if (result == "Uspešno")
                    s.dataGridView1.Rows.Add("          Nastavljanje datuma in
ure", s.imageList1.Images[1]);
                else
                {
                    s.dataGridView1.Rows.Add("          Nastavljanje datuma in
ure", s.imageList1.Images[2]);
                    s.dataGridView1.Rows[s.dataGridView1.Rows.Count -
1].Cells[1].ToolTipText = idtek.ProcessRezultat(result);
                }
            }));

        }
```

SEZNAM SLIK

SEZNAM TABEL

VIRI

[1] (2011) Access Control. Dostopno na: http://en.wikipedia.org/wiki/Access_control

[2] (2011) Radiofrekvenčna identifikacija. Dostopno na: http://sl.wikipedia.org/wiki/Radiofrekven%C4%8Dna_identifikacija

[3] (2004) Barcoding RFID And Beyond. Dostopno na: http://www.rmoroz.com/rfid.html

[4] (2008) Procontrol electronics. Dostopno na: http://www.procontrol.hu/gyartasfejlesztes/Termekeink/ProxTransponders/ProxTransponders_eng.htm

[5] (2011) Informacijski pooblaščenec RS. Dostopno na: http://www.ip-rs.si/varstvo-osebnih-podatkov/informacijske-tehnologije-in-osebni-podatki/biometrija/

[6] (2011) Biometric News portal. Dostopno na: http://www.biometricnewsportal.com/fingerprint_biometrics.asp

[7] (2008) Access control door wiring diagram. Dostopno na: http://upload.wikimedia.org/wikipedia/commons/1/10/Access_control_door_wiring.png

[8] Microsoft Visual Studio 2008 Wiki. Dostopno na: http://microsoft-visual-studio-2008.software.informer.com/wiki/

[9] (2008) Overview of the Common Language Infrastructure. Dostopno na: http://en.wikipedia.org/wiki/File:Overview_of_the_Common_Language_Infrastructure.svg

[10] (2011) .NET Framework. Dostopno na: http://en.wikipedia.org/wiki/.NET_Framework

[11] (2011) Razvoj in vpeljava programa za izdelavo poročila o potnem nalogu. Dostopno na: http://eprints.fri.uni-lj.si/1361/1/Panic1.pdf

[12] IDTeck, *ActiveX Control Reference Manual*. IDTeck, 2005.

[13] (2011) Vodenje projektov:Analiza zahtev. Dostopno na: http://www.e-studij.si/FRI:UNI:Vodenje_projektov:Analiza_zahtev

[14] K. Watson, *Beginning C# 2005 Databases*. Indianapolis: Wiley Publishing, Inc., 2006, Pogl. 6.

[15] M. MacDonald, *Pro .NET 2.0 Windows Forms & Custom Controls in C#*. New York: Springler-Verlag New York, Inc., 2005, pogl. 15.

[16] B. A. Joseph Albahari, *C# 3.0 in a Nutshell (Third Edition)*. Sebastopol: O'Reilly Media, Inc., 2007, Pogl. 4.

[17] (2011) IDTECK ICON100. Dostopno na: http://www.idteck.com/product/product_view.asp?productidx=24

[18] (2011) IDTECK. Dostopno na: http://www.idteck.com/product/product_view.asp?productidx=24

[19] (2011) IDTECK ILAN. Dostopno na: http://www.idteck.com/product/product_view.asp?productidx=151

[20] (2011) IDTECK Readers. Dostopno na: http://www.idteck.com/product/product_view.asp?productidx=29

[21] (2011) IDTECK Proximity Cards. Dostopno na:
http://www.idteck.com/product/product_view.asp?productidx=34