

SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA I RAČUNARSTVA

**DIPLOMSKI RAD**

**RAZVOJ ANDROID I DESKTOP  
APLIKACIJE ZA LIJEČNIČKU  
ORDINACIJU**

Mijo Šabić

Mostar, rujan 2018.

**SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA I RAČUNARSTVA**

**DIPLOMSKI RAD**

**RAZVOJ ANDROID I DESKTOP  
APLIKACIJE ZA LIJEČNIČKU  
ORDINACIJU**

**Mentor:**

**Dr.sc. Goran Kraljević**

**Student:**

**Mijo Šabić**

**Mostar, rujan 2018.**

## IZJAVA

*Ja Mijo Šabić izjavljujem da sam diplomski rad izradio samostalno i uz pomoć dostupne literature. Također se zahvaljujem mentoru dr.sc Goranu Kraljeviću koji mi je bio od velike pomoći pri izradi diplomskog rada.*

---

Mijo Šabić

SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA I RAČUNARSTVA

Diplomski studij:	Računarstva
Smjer:	Programsko inženjerstvo i informacijski sustavi
Ime i prezime:	Mijo Šabić
Broj indeksa:	341-RM

ZADATAK DIPLOMSKOG RADA

Naslov: RAZVOJ ANDROID I DESKTOP APLIKACIJE ZA LIJEČNIČKU

DEVELOPMENT OF ANDROID AND DESKTOP APPLICATION FOR  
DOCTORAL ORDINATION

Zadatak: U diplomskom radu potrebno je navesti i opisati tehnologike (Visual Studio 2015, MySQL Androi Studio) za razvoj desktop i android aplikacije. Korištenjem navedenih alata i tehnologija u diplomskom radu je potrebno realizirati konkretan primjer funkcionalnih aplikacija.

Prijava rada:

Rok za predaju rada:

Rad predan:

Odbor za diplomski rad:

Mentor:

---

Dr.sc Goran Kraljević

## **RAZVOJ ANDROID I DESKTOP APLIKACIJE ZA LIJEČNIČKU ORDINACIJU**

### **Sažetak:**

U ovom radu glavni zadatak je bio implementacija android i desktop aplikacije za liječničku ordinaciju. U radu je objašnjeno implementiranje aplikacija koje omogućavaju dodavanje, uređivanje i brisanje pacijenata, doktora, uputnica i sl.

## **DEVELOPMENT OF ANDROID AND DESKTOP APPLICATION FOR DOCTORAL ORDINATION**

### **Abstract:**

The goal of this paper was implementation of android and desktop application for doctoral ordination. Paper explains implementation of applications which allow adding, editing and deleting patients, doctors, referrals and similar things.

## Sadržaj

<b>1. UVOD</b>	1
<b>2. VISUAL STUDIO 2015</b>	2
2.1. Programski jezik C#	3
2.1.1. Tipovi podataka u C#	3
2.1.1. Dictionary, Array, List	3
2.2. Windows Forms	6
2.2.1. Dodavanje elemenata na formu	7
2.3. DevExpress	8
<b>3. ANDROID STUDIO</b>	9
3.1. Sustav Gradle	11
3.2. Activity class	11
3.2.1. Activity i XML Layouts	12
3.3. Java	13
<b>4. MySQL i Node.js</b>	14
4.1. Povezivanje Node.js i MySQL-a	17
4.2. Nodemailer	18
<b>5. RAZVOJ ANDROID I DEKTOP APLIKACIJE ZA LIJEČNIČKU ORDINACIJU</b>	19
5.2. Prijav u aplikacije	19
5.2. Administrator aplikacije	22
5.3. Doktor	25
5.4. Pacijent	28
<b>6. ZAKLJUČAK</b>	34
Literatura	35
POPIS SLIKA	36
POPIS TABLICA	37

# 1. UVOD

Liječnička ordinacije predstavlja skup raznih osoba koje ujedinjene čine jednu cjelinu za pomoć drugim osobama.

Problem koji se javlja kod liječničkih ordinacija je vođenje organizacije termina za pacijente odnosno kako na najjednostavniji način riješiti brojna čekanja u čekaonici, te kako liječniku omogućiti što jednostavnije praćenje pacijenata.

Drugi problem koji se javlja kod liječničkih ordinacija je praćenje dokumentacije o pacijentu. Cilj nam je bio omogućiti što jednostavnije pisanje i praćenje potrebne dokumentacije za svakog pacijenta (recepti, dijagnoze, uputnice itd.).

Kao najveći problem koji se javlja kod pacijenata je predugo čekanje u čekaonicama zbog slabe organizacije termina.

Zadatak ovog rada je bio olakšati posao liječnicima i pacijentima skratiti vrijeme čekanja u čekaonicama.

Pomoću ovog rada opisan je problem liječničkih ordinacija i naručivanja pacijenata, kao i vođenje sve potrebne dokumentacije o pacijentima. Aplikacije su izrađene na način da doktoru olakša vođenje papirologije o pacijentima, a pacijentima skрати broj dolazaka u ordinaciju. Aplikacije za doktora i administraciju su izrađene pomoću programa Visual Studio 2015 uz korištenje programskog jezika C#, .NET Frameworka (WindowsForms), DevExpress komponent. Aplikacija za pacijente je izrađena pomoću Android studija, a komunikacija između aplikacija i servera tj. Baze podataka koja je izrađena u MySQL-u je odrađena preko Node.js.

## 2. VISUAL STUDIO 2015

**Microsoft Visual Studio** jest integrirano razvojno okruženje (IDE) koga proizvodi Microsoft. Koristi se za razvoj računarskih programa za Windows, web-stranica, aplikacija i usluga. Koristi Microsoftove platforme za razvoj poput aplikativnih programskih interfejsa (API) za Windows, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Može stvarati nativni kôd i upravljači kôd (eng. *managed code*).

Visual Studio sadrži uređivač izvornog kôda koji podržava IntelliSense (komponenta koja predlaže ostatak kôda) kao i refaktoriranje kôda. Integrirani program za otklanjanje grešaka (debugger) radi na nivou izvornog i mašinskog koda. Program također sadrži alate poput dizajnera oblika koji se koristi za pravljenje aplikacija s grafičkom korisničkim interfejsom, web-dizajnera, dizajnera klasa i dizajnera shema baza podataka. Prihvata proširenja koja poboljšavaju funkcionalost na skoro svakom nivou dodajući podršku sistema za upravljanje izvornim kôdom (poput softvera subversion) i dodajući nove nizove alata poput tekstualnih uređivača i vizualnih dizajnera za jezik određenih domena ili za druge dijelove procesa razvoja softvera (poput klijenta *Team Explorer*).

Visual Studio podržava različite programske jezike i dozvoljava uređivaču kôda i debuggeru da podržava (u različitoj mjeri) gotovo bilo koji programski jezik, pod uvjetom da usluga za taj jezik postoji. Ugrađeni jezici su C, C++ i C++/CLI (preko Visual C++), VB.NET (preko Visual Basic .NET)-a, C# (preko Visual C#) i F# (počevši od programa Visual Studio 2010). Podrška za ostale programske jezike poput M-a, Pythona, i Rubyja kao i ostalih dostupna je instalacijom jezičkih servisa koji se mogu zasebno instalirati. Također podržava XML/XSLT, HTML/XHTML, JavaScript i CSS. [1]



## 2.1. Programski jezik C#

Programski jezik C# (C sharp) je proizvod tvrtke Microsoft i nastao je kao odgovor na nedostatke postojećih jezika kao što su C, C++ i Visual Basic, u isto vrijeme kombinirajući njihove dobre strane.

C# je potpuno objektno-orientirani programski jezik, što znači da svi elementi unutar njega predstavljaju objekt. Objekt predstavlja strukturu koja sadrži podatkovne elemente, kao i metode i njihove međusobne interakcije.

### 2.1.1. Tipovi podataka u C#

C# programski jezik sadrži osnovne tipove podataka kao što su:

- Numerički
- Znakovni
- Logički

Numerički tipovi podataka koriste se za prikazivanje cijelih i realnih brojeva. Za cijele brojeve koriste se tipovi podataka prikazani u tablici:

Tablica 1 Tipovi podataka cijelih brojeva

Tip	Opseg vrijednosti	Zauzeće memorije
sbyte	-128 do 127	1
byte	0 do 255	1
short	-32768 do 32767	2
ushort	0 do 65535	2
int	-2 147 483 648 do 2 147 483 647	4
uint	0 do 4 294 967 295	4
long	-9 223 372 036 854 775 808 do 9 223 372 036 854 775 807	8
ulong	0 do 18 446 744 073 709 551 615	8

Za realne brojeve koriste se tipovi float, double, decimal, a njihov opseg vrijednosti dan je u tablici:

Tablica 2 Tipovi podataka realih brojeva

Tip	Opseg vrijednosti
float	$\pm 1.5 \times 10^{-45}$ do $\pm 3.4 \times 10^{38}$
double	$\pm 5.0 \times 10^{-324}$ do $\pm 1.7 \times 10^{308}$
decimal	$\pm 1.0 \times 10^{-28}$ do $\pm 7.9 \times 10^{28}$

### 2.1.1. Dictionary, Array, List

Rječnikna klasa je struktura podataka koja predstavlja skup ključeva i vrijednosti par podataka. Ključ je identičan u paru ključ / vrijednost i može imati najviše jednu vrijednost u rječniku, ali vrijednost može biti povezana s mnogim različitim ključevima.

Ova klasa je definirana u System.Collections.Generic namespace, tako da trebate uvesti ili koristiti System.Collections.Generic namespace.

```
Dictionary<string, int> dict = new Dictionary<string, int>();  
dict.Add("one", 1);  
dict.Add("two", 2);  
dict.Add("three", 3);  
dict.Add("four", 4);
```

Kod 1. Primjer rječnika

U rješavanju raznih problema javlja se potreba za postojanjem većeg broja podataka istog tipa koje predstavljaju jednu celinu. Zbog toga se u programskim jezicima uvodi pojam niza ili u općem slučaju pojam polja. Elementi niza numerirani su sa 0,1,2,...,N-1. Ovi brojevi se nazivaju indeksima elemenata niza. Broj elemenata u niz predstavlja njegovu dužinu. Nizovi mogu biti različitih dimenzija. Najčešće se koriste jednodimenzionalni nizovi ili vektori i dvodimenzionalni nizovi ili matrice.

```
int[] prostibrojevi = new int[10] { 1, 2, 3, 5, 7, 11, 13, 17, 19, 23 };
```

Kod 2. Primer niza

Kolekcije classes su skupina klasa dizajniranih posebno za grupiranje objekata i obavljanje zadataka na njima. Popis klasa je zbirka i definirana je u System.Collections.Generic namespace i pruža metode i svojstva poput ostalih klasa prikupljanja kao što su dodavanje, umetanje, uklanjanje, pretraživanje itd. To je zamjena za polja, „linked list“, redove i većinu drugih jednodimenzionalnih struktura podataka. [2]

```
List<int> iList = new List<int>();  
iList.Add(2);  
iList.Add(3);  
iList.Add(5);  
iList.Add(7);
```

Kod 3. primjer liste

### 2.1.1. Klasa

Klasa predstavlja skup objekata sa zajedničkom građom i ponašanjem. Klasa određuje strukturu objekta navođenjem varijabli koje su sadržane u svim instancama klase i određuje ponašanje objekata preko metode instance koje izražavaju ponašanje objekata. Ovo je moćna ideja, ali nešto poput ovog se može postići u većini programskih jezika. Glavna novost objektno orijentiranog programiranja u odnosu na tradicionalno je da klase mogu izražavati sličnosti između objekata koji imaju zajedničke neke, ali ne sve dijelove strukture i ponašanja. Ovakve sličnosti mogu biti izražene pomoću nasljeđivanja (inheritance) i polimorfizma.

Naziv nasljeđivanje odnosi se na činjenicu da jedna klasa može naslijediti dio ili svu strukturu i ponašanje od druge klase. Klasa koja nasljeđuje zove se podklasa (subclass) klase od koje nasljeđuje. Ako je klasa B podklasa klase A onda je klasa A nadklasa(superclass) klase B. Podklasa može nadopunjavati strukturu i ponašanje klase koju nasljeđuje, a može i zamijeniti ili izmijeniti naslijeđeno ponašanje, ali ne i naslijeđenu strukturu. Odnos između podklase i nadklase je često prikazan kao dijagram u kojem je podklasa prikazana ispod i povezana na nadklasu. [3]

```
class Osoba  
{  
    private string ime;  
    private string prezime;  
    private int godine;  
}
```

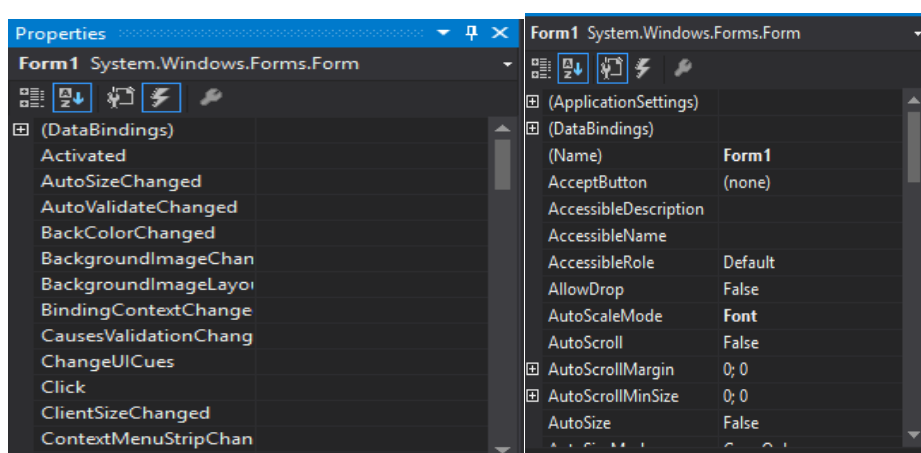
Kod 4. primjer klase

## 2.2. Windows Forms

U sustavu Windows Forms razvijaju se pametni klijenti. Pametni klijenti grafički su bogati aplikacije koje se lako implementiraju i ažuriraju, mogu funkcionirati kada su povezane s Internetom ili su odspojene s Interneta te mogu pristupiti resursima na lokalnom računalu na sigurniji način od tradicionalnih aplikacija temeljenih na sustavu Windows. Windows Forms je tehnologija pametnog klijenta za .NET Framework, skup upravljanih biblioteka koji pojednostavljaju uobičajene zadatke aplikacije kao što su čitanje i pisanje u datotečni sustav. Kada koristite razvojno okruženje kao što je Visual Studio, možete stvoriti aplikacije pametnih klijenata sustava Windows Forms koje prikazuju informacije, zahtijevaju unos korisnika i komuniciraju s udaljenim računalima putem mreže.

U Windows Forms *formi* je vizualna površina na kojoj korisnicima prikazujete informacije. Obično izrađujete aplikacije sustava Windows Forms dodavanjem kontrola na obrasce i razvoju metode na radnju korisnika, kao što su klikovi mišem ili pritisci na tipke. Mnoge aplikacije moraju prikazivati podatke iz baze podataka, XML datoteke, XML web servisa ili drugog izvora podataka. Windows Forms pruža fleksibilnu kontrolu koja se zove kontrola DataGridView za prikaz takvih tabličnih podataka u tradicionalnom obliku retka i stupca tako da svaki dio podataka zauzima vlastitu ćeliju. Kada upotrebljavate DataGridView, možete prilagoditi izgled pojedinačnih ćelija, zaključati proizvoljne redove i stupce na mjestu i prikazati složene kontrole unutar ćelija, među ostalim značajkama. [4]

Svaka forma koju stvorimo ima svoja određena svojstva poput dimenzija, boje, naziva, teksta koji se ispisiuje u zaglavlju forme itd. Svojstvima neke forme pristupamo preko Properties prozora.

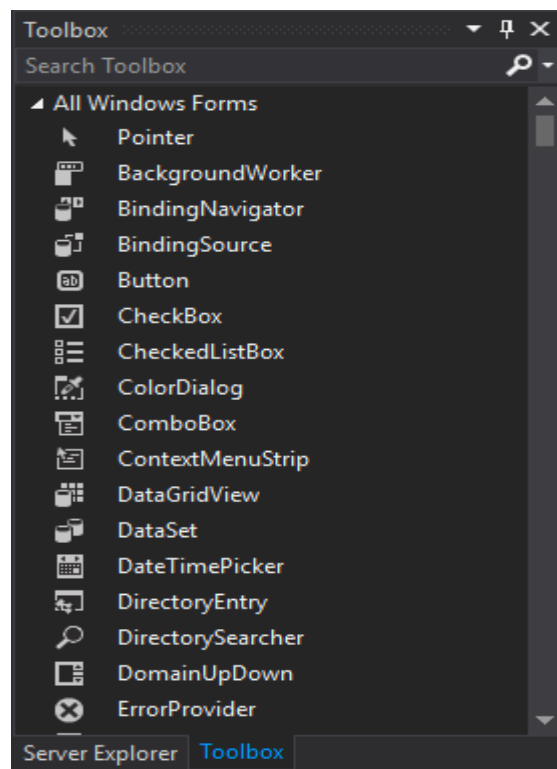


Slika 1 Svojstva formi

### 2.2.1. Dodavanje elemenata na formu

S lijeve strane Visual Studio vidjet ćete i alatnu traku. Toolbox sadrži sve kontrole koje se mogu dodati u Windows Forms. Kontrole poput teksta ili naljepnice samo su neke od kontrola koje se mogu dodati u Windows Forms. Elementi su zapravo instance različitih klasa. Elemente dodajemo na formu jednostavnim „drag-n-drop“ principom ili „dvoklikom“ na element. Elemente možemo razmještati po formi, prilagoditi im širinu i visinu, upisati neki tekst u njih i još mnoštvo toga. Svaki element ima svoja određena svojstva koja možemo mijenjati.

Kao i kod Windows Formsa, i nad elementima se može kontrolirati izvršavanje koda na neki događaj.

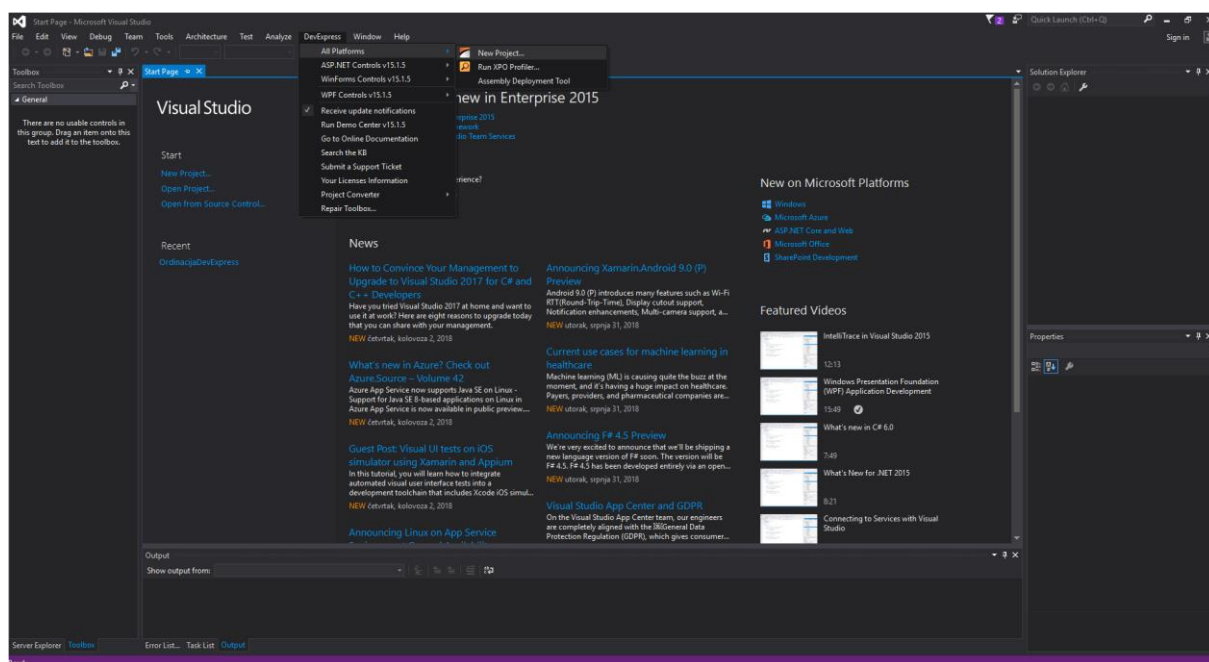


Slika 2 Toolbox

## 2.3. DevExpress

DevExpress (Developer Express Inc) je softver za razvoj aplikacija koji je osnovan 1998. godine sa sjedištem u Glendaleu, Kalifornija . DevExpress je u početku počeo proizvoditi UI kontrole za Borland Delphi / C ++ Builder i ActiveX kontrole za Microsoft Visual Studio . Trenutno DevExpress ima proizvode koji koriste programeri za razvoj u okruženjima kao što su: Delphi / C ++ Builder, Visual Studio i HTML5 / JavaScript tehnologije. [5]

Pokretanje DevExpressa vrši se pomoću Visual Studia na način da u Start Page odaberemo DevExpress kao što je prikazano na slici ispod.



Slika 3 Pokretanje DevExpressa

Nako odabira željenog oblika aplikacije i otvaranja dijaloga za izradu aplikacije možemo krenuti s radom i korištenjem DevExpress kontrola. Kao što smo već opisa kod izrade običnih Windows forma tako i kod DevExpressa kod pokretanja na lijevoj strani se nalazi prozor s ponuđenim kontrolama koje možemo pozvati u našu aplikaciju te ih uređivati i dodavati određene funkcionalnosti.

### 3. ANDROID STUDIO

Android Studio službeno je integrirano razvojno okruženje (IDE) za razvoj aplikacija za Android, temeljeno na IntelliJ IDEA . Osim vrhunskog alata za uređivanje koda i alata za razvojne programere tvrtke IntelliJ, Android Studio nudi još više značajki koje poboljšavaju produktivnost prilikom izgradnje aplikacija za Android.

Svaki projekt u programu Android Studio sadrži jedan ili više modula s datotekama izvornog koda i datotekama resursa. Vrste modula uključuju:

- Moduli aplikacije za Android
- Knjižnični moduli
- Moduli Google App Engine

Prema zadanim postavkama, Android Studio prikazuje vaše projektne datoteke u prikazu Androidovog projekta. Taj je pregled organiziran pomoću modula koji omogućuju brz pristup izvornim datotekama vašeg projekta.

Sve datoteke gradnje vidljive su na najvišoj razini pod **Gradle Scripts** i svaki modul aplikacije sadrži sljedeće mape:

- **manifestira** : Sadrži AndroidManifest.xml datoteku.
- **java** : Sadrži datoteke izvornog koda Java, uključujući JUnit test kôd.
- **res** : Sadrži sve resurse koji nisu kodovi, kao što su XML izgled, žice korisničkog sučelja i bitmap slike.

Struktura Androida projekta na disku razlikuje se od ove spljoštene reprezentacije. Da biste vidjeli stvarnu strukturu datoteke projekta, odaberite **Project** iz padajućeg izbornika **projekta**.

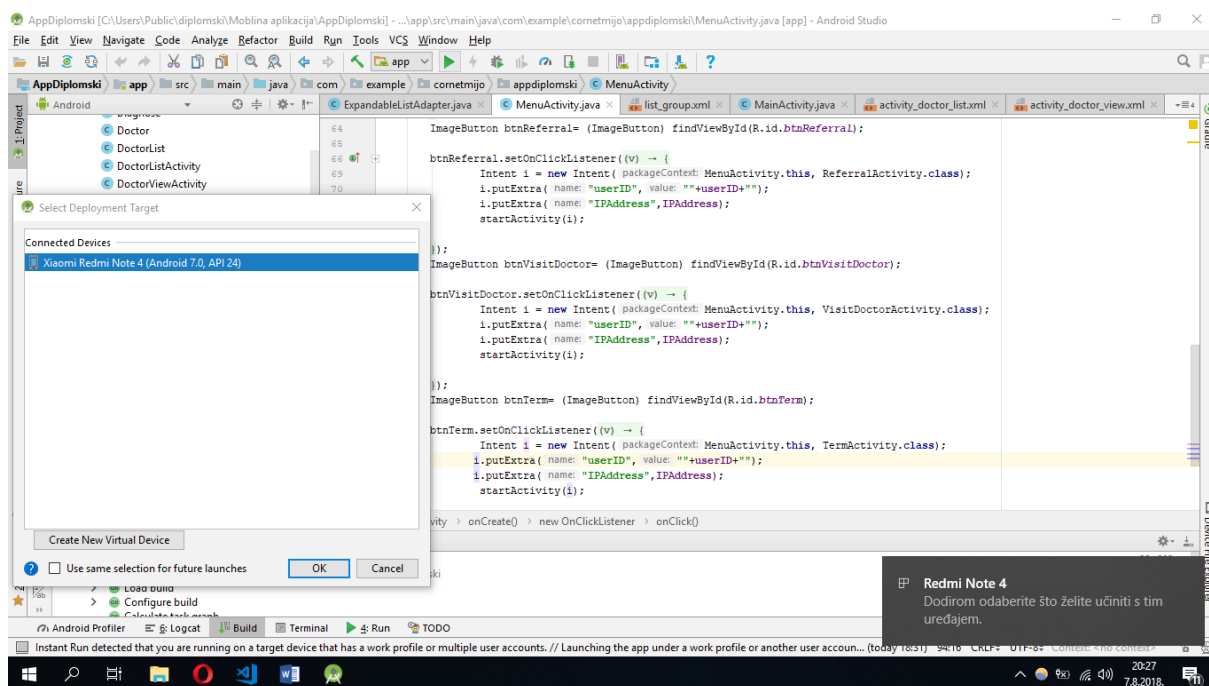
Također možete prilagoditi prikaz projekata kako biste se usredotočili na određene aspekte razvoja aplikacije. Na primjer, odabirom opcije **Problemi** gledanja vašeg projekta prikazuju se veze na izvorne datoteke koje sadrže bilo koju priznatu pogrešku kodiranja i sintakse, kao što je oznaka zatvaranja XML elementa koji nedostaje u datoteci izgleda.

Glavni prozor možete organizirati kako biste dobili više prostora na zaslonu sakrivanjem ili premještanjem alatnih traka i prozora alata. Tipkovne prečace možete koristiti i za pristup većini IDE značajki.

U bilo kojem trenutku možete pretraživati izvorni kôd, baze podataka, radnje, elemente korisničkog sučelja i tako dalje dvostrukim pritiskom na tipku Shift ili klikom na povećalo u gornjem desnom kutu Android Studio prozor. To može biti vrlo korisno ako, primjerice, pokušavate pronaći određenu IDE radnju koju ste zaboravili kako pokrenuti. [6]

Svaki put prilikom isprobavanja aplikacije ili njenog konačnog testiranja potrebno je aplikaciju instalirati na određeni uređaj koji na sebi posjeduje android sustav. Imamo više načina pokretanja odnosno instaliranja aplikacije. Prvi način instaliranja aplikacije odnosno njenog pokretanja je da aplikaciju pokrenemo pomoću emulatora koji dolazi u samom Android studiju.

Drugi način je da instaliram neki od emulatora koje nam nude razni proizvođači te da nam ona zamijeni naš mobilni uređaj ili emulator koji se koristi u Android studiju. Treći način koji nam omogućuje instaliranje tj. pokretanje aplikacije je taj da aplikaciju pokrenemo na našem mobilnom uređaju koji također koristi Android susta. Za pokretanje aplikacije na mobilnom uređaju potrebno je taj isti uređaj povezati s računalom na kojem se nalazi Android studijo i kod naše aplikacije. Nakon povezivanja računala i mobitela potrebno je na mobitelu omogućiti opciju razvojnog programera. Opcija razvojni programer se omogućuje na način da odemo u postavke mobilnog uređaja i kliknemo na gumb „O uređaju“, nakon toga pronađemo liniju na kojoj piše „Broj podverzije“ i na tu liniju brzo klikamo dok nam se ne pojavi poruka da smo aktivirali opciju razvojnog programera.



Slika 4 Pokretanje aplikacije preko mobilnog uređaja



### 3.1. Sustav Gradle

Android Studio koristi Gradle kao temelj sustava izgradnje, s dodatnim mogućnostima specifičnim za Android, koje dodjeljuje Android dodatak za Gradle. Ovaj sustav za izgradnju funkcionira kao integrirani alat iz izbornika Android Studio i neovisno od naredbenog retka. Značajke sustava gradnje možete koristiti za sljedeće:

- Prilagodite, konfigurirate i proširite postupak izrade.
- Izradite više APK-ova za svoju aplikaciju, s različitim značajkama koje koriste isti projekt i module.
- Ponovno upotrijebite kod i resurse na svim izvorima.

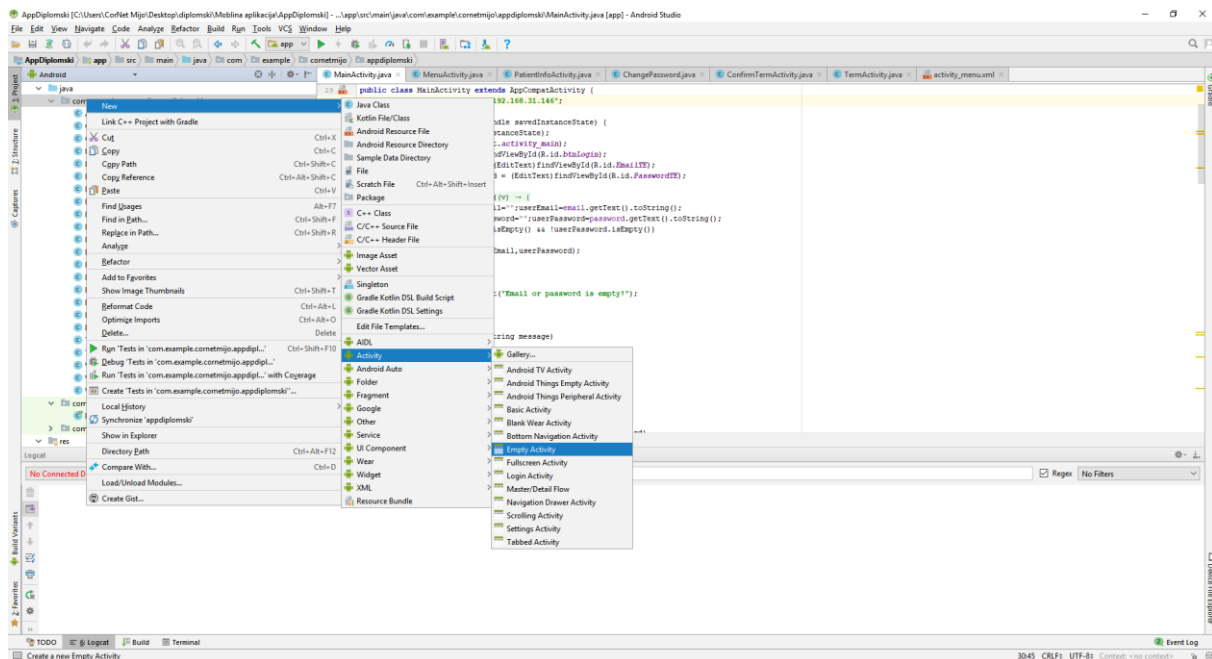
Koristeći fleksibilnost Gradlea, sve to možete postići bez izmjene osnovnih datoteka svoje aplikacije. Imenovane su datoteke za izgradnju Androida Studio `build.gradle`. To su obične tekstualne datoteke koje koriste Groovy sintaksu za konfiguriranje gradnje s elementima koje pruža Android dodatak za Gradle. Svaki projekt ima jednu datoteku za izgradnju najviše razine za cijeli projekt i odvojene datoteke gradnje na razini modula za svaki modul. Kada uvezete postojeći projekt, Android Studio automatski generira potrebne datoteke za izgradnju. [6]

### 3.2. Activity class

Activity class je ključna komponenta aplikacije za Android, a način na koji su aktivnosti pokrenuta i staviti zajedno je temeljni dio platforme modela aplikacija. Za razliku od paradigmi programiranja u kojima se aplikacije pokreću `main()` metodom, sustav Android pokreće kod u Activity primjeru pozivajući se na određene metode povratnog poziva koji odgovaraju određenim fazama svog životnog ciklusa. Ovaj dokument uvodi koncept aktivnosti, a zatim daje lagane upute o tome kako raditi s njima.

Kada jedna aplikacija pozove drugu, pozivnica poziva aktivnost u drugoj aplikaciji, a ne aplikaciji kao atomskoj cjelini. Na taj način, aktivnost služi kao ulazna točka za interakciju aplikacije s korisnikom. Aktivnost provodite kao podrazred Activity razreda. [6]

Dodavanje novog Activity-a vrši se na način da kliknemo na karticu File -> New -> Activity i odaberemo željeni tip Activity-a kao što je prikazano na slici ispod. Dodavanjem novog Activity-a dobijemo dvije datoteke jedna koja se nalazi u folderu java i druga koja se nalazi u folderu res/layout.



Slika 5 Dodavanje novog Activity-a

### 3.2.1. Activity i XML Layouts

Layout definira strukturu korisničkog sučelja u vašoj aplikaciji, primjerice u nekoj aktivnosti. Svi elementi u layout kreiraju se pomoću hijerarhije View i ViewGroup objekata. View obično privlači nešto korisnik može vidjeti i komunicirati s tim. Dok ViewGroup je nevidljivi spremnik koji definira strukturu izgleda View i druge ViewGroup objekte.

Pomoću XMLovog rječnika za Android možete brzo izraditi raspored UI-a i elemente zaslona koji sadrže, na isti način na koji izrađujete web stranice u HTML-u - s nizom ugniježđenih elemenata.

Svaka datoteka layout-a mora sadržavati točno jedan element korijena, koji mora biti objekt View ili ViewGroup. Kada definirate element korijena, dodatni predmeti ili widgeti izgleda kao elemente, za dijete možete postupno graditi hijerarhiju prikaza koja definira vaš izgled. Kada sastavite aplikaciju, svaka datoteka XML layout-a se stavlja u View resurs. U svojem postupku Activity.onCreate() povratnog poziva trebali biste učitati resurs izgleda iz koda aplikacije.

### 3.3. Java

**Java** je objektno orijentirani programski jezik koji su razvili James Gosling, Patrick Naughton i drugi inženjeri u tvrtci Sun Microsystems. Razvoj je počeo 1991, kao dio projekta Green, a objavljen je u studenom 1995.

Tvrtka Sun posjeduje trademark na ime Java, ali samo okruženje je moguće bez plaćanja skinuti sa Sunovih internet poslužitelja.

Velika prednost u odnosu na većinu dotadašnjih programskih jezika je to što se programi pisani u Javi mogu izvoditi bez preinaka na svim operativnim sustavima za koje postoji **JVM** (Java Virtual Machine), dok je klasične programe pisane primjerice u C-u potrebno prilagođavati platformi (Operacijskom sustavu) na kojem se izvode.

Time i bogatim skupom klasa za rad s mrežnim komunikacijama u jednom trenutku je Java bila najbolji izbor za široku lepezu mogućih aplikacija. Microsoft je stoga razvio svoj C# i .NET platformu kao odgovor na open source alternative.

Java je jedan od najkorištenijih programskih jezika. Procjene i izvješća o broju korisnika kreću se od gotovo 7 do preko 10 milijuna.

Na današnjem tržištu, Java se koristi široko i dosljedno tamo gdje preteže brzina razvoja programskog sustava nad zahtjevima do brzine rada programa. Iako inspirirana jezikom C, Java pruža bolji stupanj sigurnosti i pouzdanosti zahvaljujući VM-u i hermetički zatvorenom okolišu u kome svaki program operira: na Javi se brže razvija program s manje pogrešaka.

Upravo zbog toga je popularna za razvoj programa na mobilnim telefonima i kod financijskih kompanija. Javlja se kao osnovni jezik za programiranje Googleovog sustava Android. [7]

Uporaba Jave u Android studiju omogućuje nam izradu raznih funkcionalnosti koje našoj aplikaciji daju interakciju i funkcionalnost.

## 4. MySQL i Node.js

**MySQL** je besplatan, open source sustav za upravljanje bazom podataka. Uz PostgreSQL MySQL je čest izbor baze za projekte otvorenog koda, te se distribuira kao sastavni dio serverskih Linux distribucija, no također postoje inačice i za ostale operacijske sustave poput Mac OS-a, Windowse itd.

MySQL baza je slobodna za većinu uporaba. Ranije u svom razvoju, MySQL baza podataka suočila se s raznim protivnicima MySQL sustava organiziranja podataka jer su joj nedostajale neke osnovne funkcije definirane SQL standardom. Naime, MySQL baza je optimizirana kako bi bila brza nauštrb funkcionalnosti. Nasuprot tome, vrlo je stabilna i ima dobro dokumentirane module i ekstenzije te podršku od brojnih programskih jezika: PHP, Java, Perl, Python...

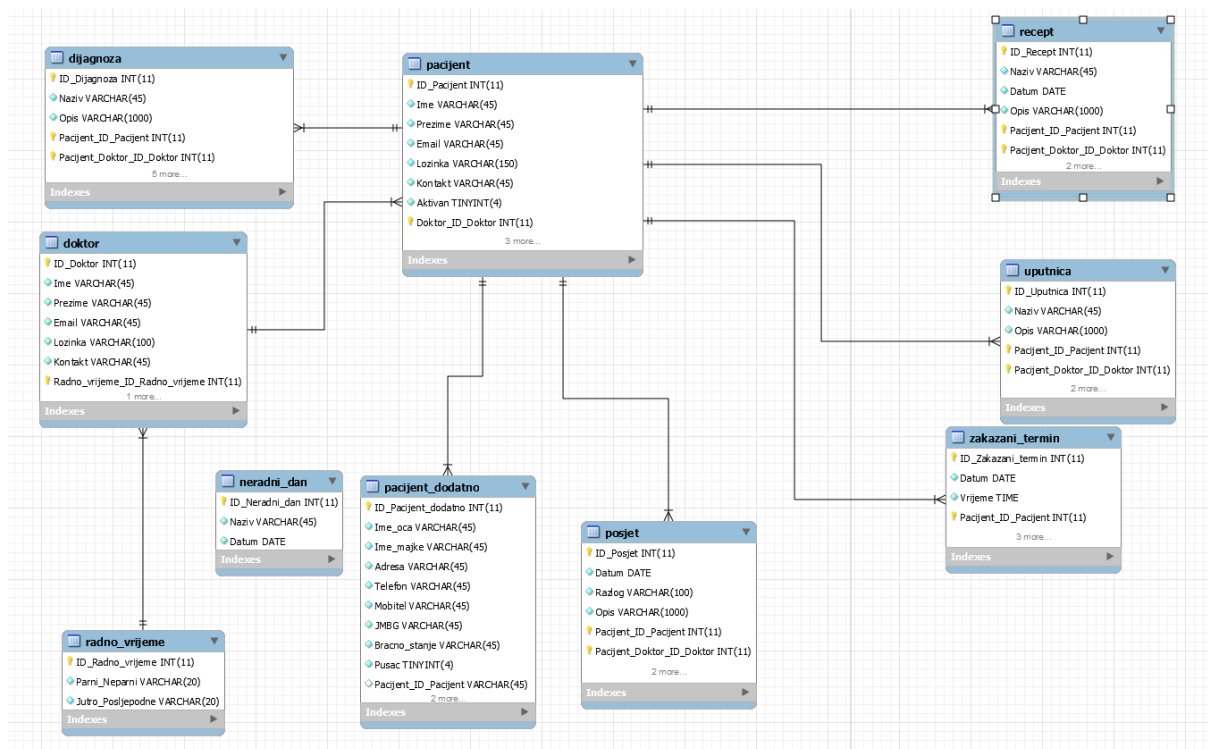
MySQL baze su relacijskog tipa, koji se pokazao kao najbolji način skladištenja i pretraživanja velikih količina podataka i u suštini predstavljaju osnovu svakog informacijskog sustava, tj. temelj svakog poslovnog subjekta koji svoje poslovanje bazira na dostupnosti kvalitetnih i brzih informacija.

MySQL i PHP su osvojili veliki dio tržišta jer su open source, dakle, mogu se besplatno koristiti.

Prije upuštanja u rad sa bilo kojim DBMS sistemom, pa tako i sa MySQL-om potrebno je dizajnirati odgovarajući izgled baze podataka, odnosno napraviti **shemu** baze, koja se u kasnijem postupku prevodi u određen broj tablica koje se koriste za pohranjivanje podataka. Osnovi element koji se pohranjuje u bazi naziva se **entitet**, entitet može biti bilo što: osoba, neki objekt, događaj, služba u nekoj organizaciji i sl. dakle stvari iz stvarnog života o kojima želimo čuvati informacije. Drugi važan pojam u teoriji baza podataka jeste **relacija**. Kao što u stvarnom životu postoje određeni međusobni odnosi između dvije ili više osoba, događaja isl. tako se i u bazama podataka mogu pojaviti određeni odnosi ili **relacije** između raznih entiteta, koji se na odgovarajući način predstavljaju unutar same baze.

Prema vrsti, relacije se mogu podijeliti na relacije **jedan prema jedan, jedan prema više odnosno više prema jedan te više prema više**. Uzmimo za primjer da modeliramo bazu

koja sadrži dvije tabele, jedna za pohranu informacija o zaposlenim osobama, a druga informacije o službama koje postoje u toj određenoj organizaciji. [7]



Slika 6 Modle baze

Relacijski model baze podataka aplikacije sastoji se od 10 tablica koje su nam omogućile jednostavnije funkcioniranje aplikacije kao i lakšu izvedbu iste.

- Tablica Dijagnoza sastoji se od ID\_Dijagnoza (int 11), Naziv (varchar 45), Opis (varchar 1000) također se sastoji od dva strana ključa ID\_Pacijent (int 11) i ID\_Doktor (int 11). ova tablica nam služi da možemo voditi evidenciju svih dijagnoza koje doktor dodaje za pacijenta.
- Tablica Doktor nam služi za pohranu svih informacija od doktorima. Sastoji se ID\_Doktor (int 11) Ime, Prezime, Email, Kontakt (varchar 45), Lozinke (varchar 100) kao i od stranog ključa ID\_Radno\_vrijeme (int 11) koji nam označuje koje radno vrijeme ima koji doktor u bazi.
- U tablicu pacijente pohranjujemo sve pacijente koje je dodao doktor ili administrator aplikacije, a sadrži ID\_Pacijent (int 11), Ime, Prezime, Email, Kontakt (45), Lozinka (varchar 150), polje Aktivan (tinyint 4) nam služi za razlikovanje aktivnih i neaktivnih pacijenata koji su upisani u bazu. Kao i većina ostalih tablica tablica pacijent sadrži

strani ključ, a to je ID\_Doktor (int 11) on nam ukazuje kod kojeg doktora je pacijent prijavljen.

- Tablica Radno vrijeme sastoji se od tri polja i to su ID\_Radno\_Vrijeme (int 11) Parni\_Neparni, Jutro\_Posljepodne (varchar 20). Pomoću ove tablice određujemo moguće kombinacije radnog vremena (parni dani jutri, parni dani posljepodne, neparni dani jutri, neparni dani posljepodne)
- Neradni dani je tablica koja nije poveza s niti jednom drugom tablicom u modelu baze. Sastoji se od ID\_Neradni\_Dan (int 11), Naziv (varchar 45) i Datum (Date). Ova tablica služi za pohranu svih dana koji su neradni, a tu spadaju praznici, blagdani i slično. Tablicu ima pravo popunjavati samo administrator aplikacije i odnosi se na sve doktore koji koriste aplikaciju.
- Tablica Pacijent\_dodatno se popunja prilikom prijave pacijenta u mobilnu aplikaciju tj. prilikom registracije u aplikaciju. Tablica se sastoji od ID\_Pacijent dodatno (int 11), Ime\_oca, Ime\_majke, Adresa, Telefon, Mobitel, JMBG, Bracno\_stanje( varchar 45), sadrži još i polje Pusac (tinyint 4) koji označuje dali je pacijent pusac ili ne, te ima i strani ključ ID\_Pacijent da znamo o kojem se pacijentu točno radi.
- U tablici Posijet pokranjuju se sve posjeti pacijenta kod njegovog doktora, a sama tablica sadrži ID\_Posijet (int 11), Datum (Date), Razlog (varchar 100), Opis(1000) i dva strana ključa ID\_Pacijent i ID\_Doktor (int 11).
- Tablica Recep sastoji se od ID\_Recept (int 11), Naziv (varchar 45), Datum (Date), Opis (varchar 1000) i još ima dva strana ključa koji određuju pacijenta i doktora ID\_Pacijent i ID\_Doktor (int 11).
- Tablica Uputnica se sastoji od ID\_Uputnica (int 11), Naziv (varchar 45), Opis(varchar 1000) i od dva strana ključa ID\_Pacijent i ID\_Doktor (int 11).
- Tablica Zakazani\_termin nam služi za što jednostavnije rješenje problema ovog diplomskog rada pomoću nje pamtimo sve zakazane termine nekog pacijenta kao i sve zauzete termine nekog doktora. Tablica Zakazani\_termin se sastoji od ID\_Zakazani\_termin (int 11), Datum (Date), Vrijeme (Time) i stranih ključeva ID\_Pacijent i ID\_Doktor (int 11).

Node.js je open-source, cross-platforma JavaScript run-time okruženja koja izvršava JavaScript kod izvan preglednika. Povijesno gledano, JavaScript je prvenstveno upotrebljavao skriptiranje na strani klijenta, u kojem su skripte napisane u JavaScript ugrađene u HTML web stranice i pokreću web-preglednik korisnika pokretačkom programom JavaScripta. Node.js omogućuje razvojnim programerima JavaScript za pisanje alata za naredbeni red i za skriptiranje na strani poslužitelja -pokretanje skripti na strani poslužitelja za izradu dinamičnog sadržaja web stranica prije nego što se stranica šalje korisnikovom web pregledniku. Slijedom toga, Node.js predstavlja paradigmu "JavaScript svugdje", ujedinjujući razvoj web aplikacija oko jednog programskog jezika, a ne na različitim jezicima za strani poslužitelja i klijentske skripte.

Node.js prvenstveno se koristi za izgradnju mrežnih programa kao što su web poslužitelji. Najveća razlika između Node.js i PHP je da većina funkcija u PHP bloku do završetka (naredbe izvršiti samo nakon prethodne naredbe preciznosti), a Node.js funkcije neblokirajući.

#### 4.1. Povezivanje Node.js i MySQL-a

Povezivanje Node.js i MySQL-a vrši se na način da se iz npm registra instalira modu MySQL. On se instalira uz pomoć naredbe koja se upiše u naredbeni redak koji pokazuje na točno određenu putanju.

```
$ npm install mysql
```

Kod 6. instalacija mysqla

Nakon same instalacije potrebno je napisati upravljački kod za povezivanje node.js i MySQL-a koji izgleda ovako:

```
const mysql = require('mysql');  
const http = require('http');  
var connection = mysql.createConnection({  
  host : 'localhost',  
  user : 'root',  
  password : '',  
  database : 'ordinacija',  
  port : '3309'});  
connection.connect();
```

Kod 7. Konekcija baze i Node.js

## 4.2. Nodemailer

Nodemailer je modul za Node.js aplikacije koji omogućuju jednostavno slanje e-pošte u obliku torte. Projekt je započeo 2010. godine kada nije bilo jednostavne opcije slanja poruka e-pošte, a danas je rješenje većina korisnika Node.js po defaultu. Nodemailer je licenciran pod licencom MIT.

Nodemailer se instalira na jednostavan način kao i većina modula za Node.js. U naredbeni redak upiše se naredba:

```
npm install nodemailer --save
```

Kod 8. Instalacija nodemailea

A sami kod za slanje mail poruka izgleda ovako:

```
const nodemailer = require('nodemailer');
const sendMail = function (sendTo, subject, text, htmlText) {
  nodemailer.createTestAccount((err) => {
    let transporter = nodemailer.createTransport({
      service: 'gmail',
      auth: {
        user: 'email@gmail.com', // generated ethereal user
        pass: 'password.' // generated ethereal password
      }
    });
    let mailOptions = {
      from: 'email94@gmail.com', // sender address
      to: sendTo, // list of receivers
      subject: subject, // Subject line
      text: text, // plain text body
      html: htmlText // html body
    };
    transporter.sendMail(mailOptions, (error, info) => {
      if (error) {return console.log(error); }
      console.log('Message sent: %s', info.messageId);
      console.log('Preview URL: %s', nodemailer.getTestMessageUrl(info));});});
  module.exports = {sendMail: sendMail}
```

Kod 9. Slanje emaila



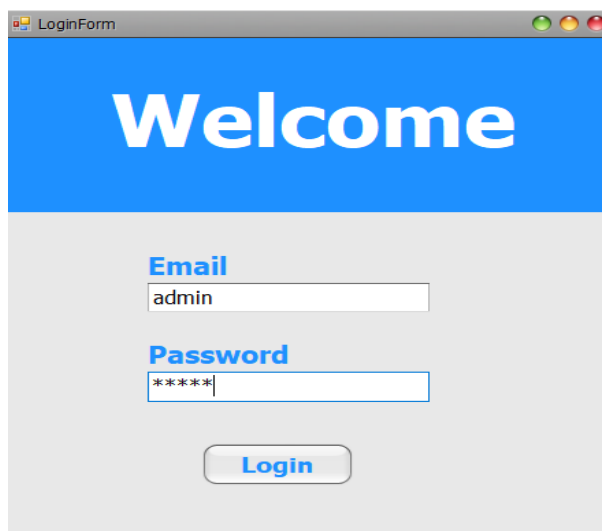
## 5. RAZVOJ ANDROID I DEKTOP APLIKACIJE ZA LIJEČNIČKU ORDINACIJU

Problem ovog diplomskog rada je razrađen kroz tri tipa korisnika. Administrator sustava kao i liječnik u aplikaciju se prijavljuju preko desktop aplikacije koja je izrađena u Visual Studio 2015 u dodavanje DevExpress komponenti i njihovo programiranje C# programskim jezik. Treći tip korisnika je pacijent koji se u aplikaciju prijavljuje pomoću mobilne aplikacije koja je izrađena u Android studiu.

- Desktop aplikacija se sastoji od sučelja za prijavu korisnika te od raznih drugih sučelja koji omogućavaju rješenje određene problematike ovoga rada.
- Androdi aplikacija se sastoji također od login sučelja te raznih drugih koje ćemo detaljnije opisati u radu.
- Komunikaciju između aplikacija nam je omogućio Node.js

### 5.2. Prijav u aplikacije

Prijava u aplikaciju aplikacije se vrši na više načina ovisno o tipu korisnika koji se prijavljuje. Za prijavu administratora u aplikaciju potrebno je u sučelje za prijavu unijeti polje Email upisati „admin“ kao i u polje za Password. Primjer je prikazan na slici ispod.



Slika 7 Prijava administratora

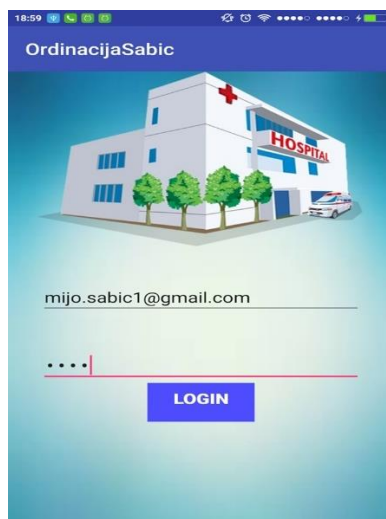
Za prijavu liječnika u aplikaciju potrebno je potrebno je u polje Email upisati vlastitu email adresu, au polje Password potrebno je upisati lozinku koju je primio na tu istu email adresu. Kod prijave u desktop aplikaciju vrši se provjera valjanosti unesenih podatak. Ako uneseni podaci prođu provjeru provjerava se koji je tip korisnika se prijavio te će se u slučaju da su podaci za prijavu ispravno upisani otvoriti određena aplikacija.

```
private void Login()
{
    if (EmailTE.Text == "admin" && PasswordTE.Text == "admin")
    {
        MainForm mf=new MainForm();
        mf.Show();
        this.Hide();
    }
    else
    {
        string pass = _DB.Login(EmailTE.Text, PasswordTE.Text);
        if (pass.Split(';')[0]== Base64Encode(CalculateMD5Hash("1950th"+ EmailTE.Text + ";" + PasswordTE.Text)))
        {
            FormDoctor.MainFormDoctor mfd = new FormDoctor.MainFormDoctor(pass.Split(';')[1]);
            mfd.Show();
            this.Hide();
        }
        else
        {
            XtraMessageBox.Show("Username and password are invalid! Please try again!");
        }
    }
}
```

Kod 10. Login metoda

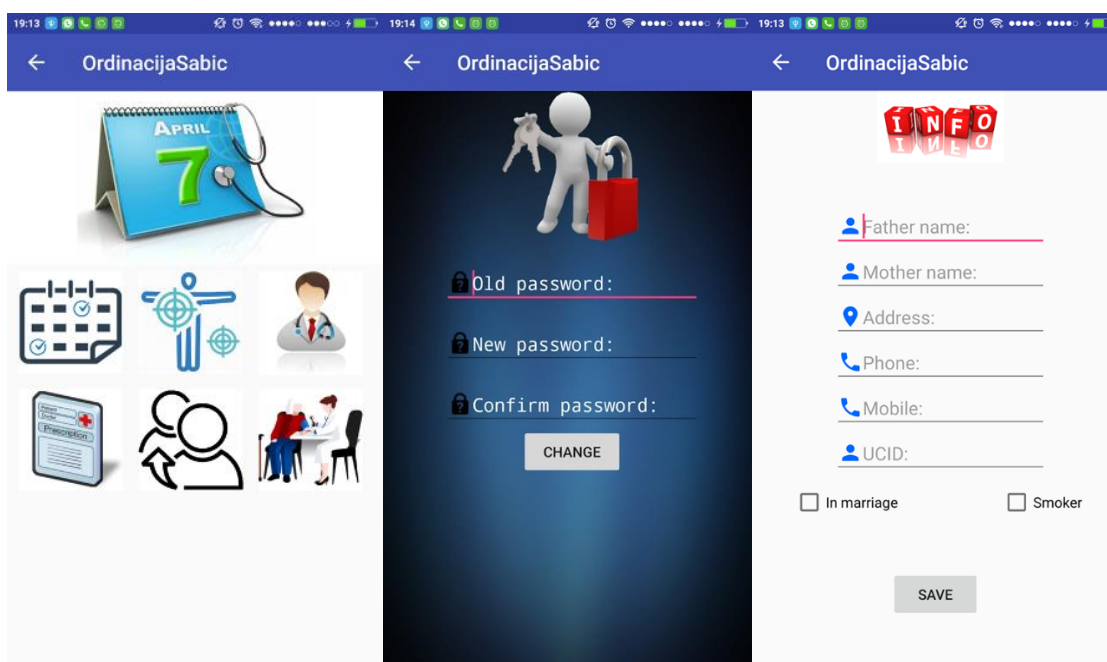
Metoda login radi na principu da provjeri jesu li u TextEdit Email i u TextEdit Password upisani podaci „admin“ i „admin“, ako jesu otvara se forma za administratora i omogućavaju mu se određene funkcionalnosti u aplikaciji. U slučaju da nisu upisani navedeni podaci provjerava se je li se onda u aplikaciju pokušao prijaviti liječnik sa svojom email adresom i lozinkom, u slučaju da je otvara se forma za prijavljenog liječnika. Ako uneseni podaci ne odgovaraju ni za administratora ni za liječnika ispiše se poruka da korisničko i lozinka nisu ispravni te da se pokuša ponovno prijaviti u aplikaciju.

Prijava pacijenta u aplikaciju vrši se preko aplikacije koja se nalazi na mobitelu. Prijava se vrši na način na koji se prijavljuje i liječnik tj. unosi se vlastita email adresa u polje za email te se u polje za password unese lozinka koju je primio na taj isti email. Primjer prijave se nalazi na slici ispod.



Slika 8 Login pacijent

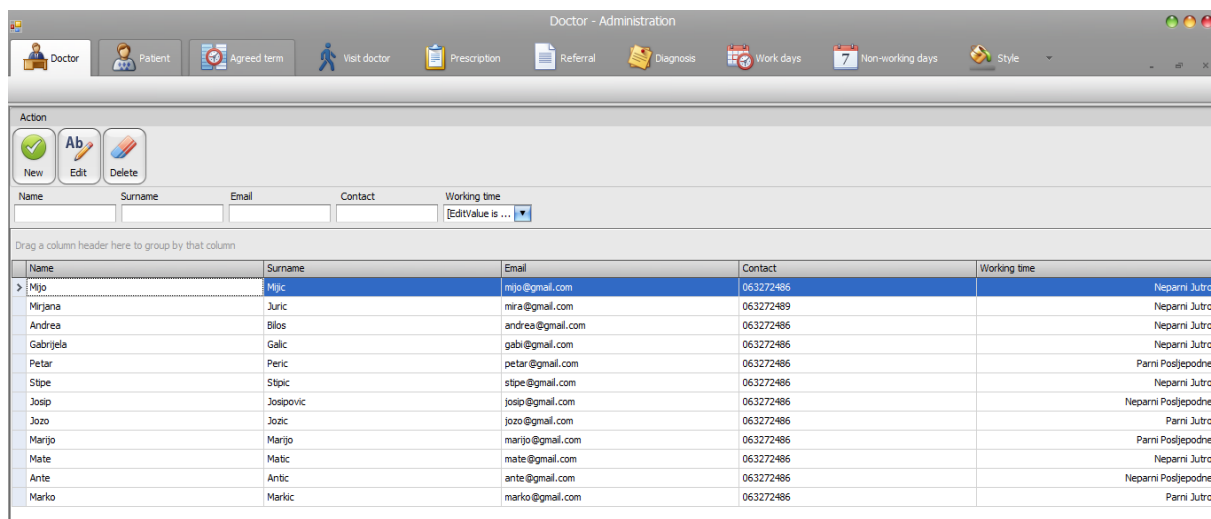
Kao i kod prijave korisnika u desktop aplikaciju tako i kod prijave pacijenata u mobilnu aplikaciju vrši se provjera ispravnosti podataka te se vrši provjera postoji li pacijent koji se želi prijaviti u aplikaciju. U slučaju da su podaci za prijavu ispravno upisani otvara se određeni prozor za pacijenta koji je se prijavio, ako je se prijavio aktivirani pacijent njemu se otvori glavni menu, a ako je se prijavio pacijent koji još nije aktiviran otvara se prozor za unos osobnih podataka te nakon unosa podataka otvara se prozor za promjenu lozinke. Nakon promjene lozinke pacijent postaje aktiva ne se treba ponovno prijaviti u aplikaciju da bi se pokrenu glavni menu.



Slika 9 Prijave u mobilnu aplikaciju

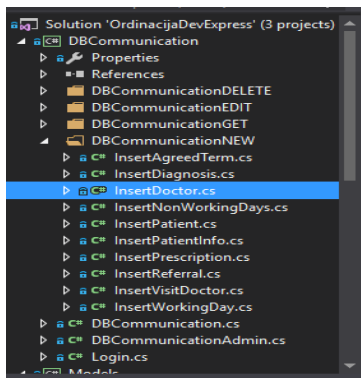
## 5.2. Administrator aplikacije

Kada se administrator na već opisani način prijavi u aplikaciju otvori mu se sučelje koje mu omogućuje dodavanje, editiranje i brisanje svih podataka koji se nalaze u bazi podataka. Na taj način administratoru se omogućuje dodavanje novih doktora i novih pacijenata za određen doktor, omogućuje mu se dodavanje neradnih dana, radnih termina i pregled svih tih i drugih podatak. Kod dodavanja doktora u bazu podataka potrebno je upisati sve potrebne podatke koji se traže u aplikaciji, potrebno je među ostalima upisati email adresu doktora na koju će prilikom dodavanja biti poslan email s odgovarajućom lozinkom koja će mu omogućiti prijavu u vlastitu desktop aplikaciju. Na isti način se unose i pacijenti.



Slika 10 Sučelje za prikaz i određene akcije s doktorima

Koda dodavanja novog doktora tako i pacijenta podaci prođu određene metode kao što su provjera valjanosti podataka. Nakon provjere valjanosti podaci odlaze u solution DBComunnication te se iz njega pozove klasa InserDoctor.cs koja se nalazi u Folderu DBComunnicationNEW kao što je prikazano na slici ispod.



Slika 11 Class InserDoctor.cs

Klasa InsertDoctor.cs prima objekt Doctor koji sadrži sve potrebne informacije za dodavanje novog doktora u bazu podataka. Nakon pozivanja klase i prosljeđivanja određenog parametra šalje se WebRequest koji je implementiran u Node.js i izvršava upit na bazu requestu se prosljeđuju potrebni parametri za insert doktora kao što su ime, prezime, email, kontakt, radno vrijeme i sl.. Prije samo inserta na server strani pozove se metoda za generiranje passworda koji se pomoću metode za slanje maila koja je također kreirana u Node.js pošalje na odgovarajuću email adresu. Poslije slanja emaila izvršava se MD5hash metoda koja kriptira lozinku te je prosljeđuje upitu na bazu podataka i pohranjuje u samu tablicu baze.

```
2 references | msabic, 18 days ago | 1 author, 1 change
class InsertDoctor
{
    1 reference | msabic, 18 days ago | 1 author, 1 change
    public bool Execute(Doctor dr)
    {
        try
        {
            var request = WebRequest.Create("http://localhost:3000/InsertDoktor");
            request.Headers["name"] = dr.Name;
            request.Headers["surname"] = dr.Surname;
            request.Headers["email"] = dr.Email;
            request.Headers["contact"] = dr.Contact;
            request.Headers["workinghours"] = dr.Working_time.ToString();
            request.GetResponse();
            return true;
        }
        catch(Exception ex)
        {
            return false;
            throw ex;
        }
    }
}
```

Kod 11. Klasa InsertDoctor.cs

```

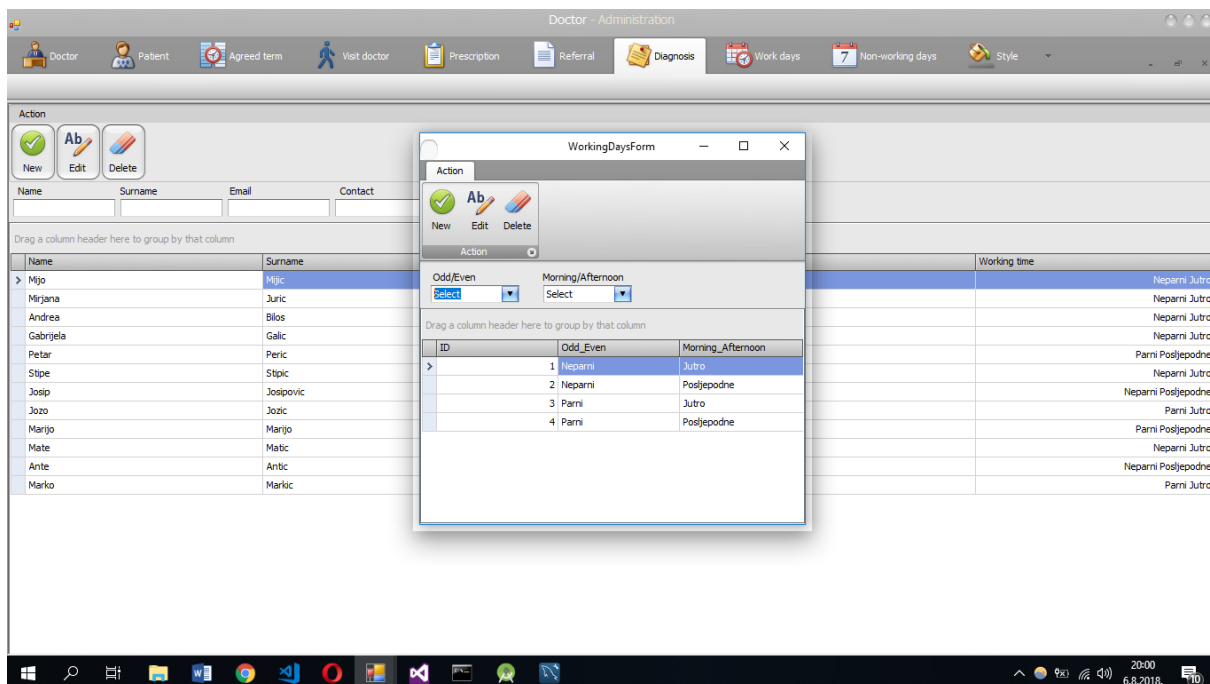
app.get('/InsertPacijent', function(req,res) {
  if(req.headers.name && req.headers.surname && req.headers.email && req.headers.contact && req.headers.doktor){

    const password=generatedPassword;
    console.log(password);
    console.log(md5(password));
    db.InsertPacijent(req.headers.name,req.headers.surname,req.headers.email,md5(password),
    req.headers.contact,req.headers.doktor).then((resoult) => {
      mailer.sendMail(req.headers.email,
        'Kreiranje računa',
        '<h3>U aplikaciju se možete prijaviti pomoću lozinke ${password}<h3>',
        '<h3>Kreiranje računa</h3>'
        '<p>U aplikaciju se možete prijaviti pomoću lozinke ${password}</p>'
      );
    });
    res.send(resoult);
  }).catch((err) => {
    console.log('Error: ', err);
    res.status(400).send('Bad Request');
  });} else {
    res.status(400).send('Bad Request');
  }})

```

Kod 12. Metoda za pozivanje insert query u Node.js

Na sličan način administratoru je omogućeno dodavanje pacijenata, termina, uputnica kao i brisanje istih. Administrator je dakle osoba koja ima zadatak dodati svakog doktora i dodati sve neradne dane u godini kao i odrediti sve moguće radno vrijeme. Moguće radno vrijeme bit će prikazani na slici. Kod dodavanja doktora u bazu potrebno mu je odabrati određeno radno vrijeme koje je već dodano, a ako administrator dodaje novog pacijenta također mu je potrebno odrediti doktora.



Slika 12 Radno vrijeme

Kod rezervacije termina za doktora potrebno je odabrati željenog doktora iz padajućeg izbornika. Nakon odabira doktora iz padajućeg izbornika pojavi nam se kalendar te još dva padajuća izbornika. Poslije odabira željenog datuma provjeri se dali je taj datum ispravan tj. jeli taj odabrani dan radni ili ne. Ako je dan radni odnosno nije vikend i ne nalazi se u tablici neradnih dana onda se provjeri koje je radno vrijeme za odabrani dan (jutro, poslijepodne) ako je radno vrijeme jutri u padajući izbornik se napune svi slobodni termini tog dana od 08:00 do 13:45, a ako je radno vrijeme poslijepodne onda se padajući izbornik napuni terminima od 14:00 do 19:45.

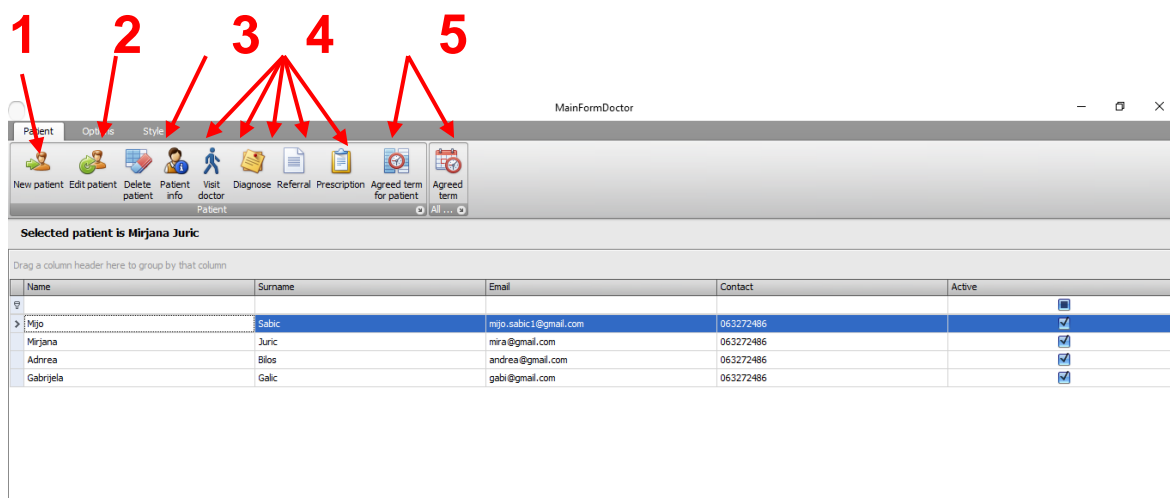
```
DateTime date = DateCalendar.SelectionStart;
if (!_DB.GetNonWorkingDay(date.ToString("yyyy-MM-dd")) && date.DayOfWeek.ToString() != "Saturday" &&
    date.DayOfWeek.ToString() != "Sunday") {
    if (date.Day % 2 == 0)
    {if (parni){jutro = true;}}else { jutro = false; }}else{ if (neparni){jutro = true;}}else{ jutro = false;}}
    if (jutro)
    {
        List<Term> temp = new List<Term>();
        foreach (Term t in _termAM)
        {
            foreach (AgreedTerm a in _agreedTerm)
            {
                if (t.Time == a.Time)
                {
                    temp.Add(t);
                }
            }
        }
        foreach (Term t in temp)
        {
            _termAM.Remove(t);
        }
        FreeTermLE.Properties.DataSource = _termAM;
    }
    else
    {
        List<Term> temp = new List<Term>();
        foreach (Term t in _termPM)
        {foreach (AgreedTerm a in _agreedTerm)
            {if (t.Time == a.Time)
                {
                    temp.Add(t);
                }
            }
        }
        foreach (Term t in temp)
        { _termPM.Remove(t);}
        FreeTermLE.Properties.DataSource = _termPM;
    }
}
```

Kod 13. Kod provjere radnog vremena

### 5.3. Dotror

Nakon što smo opisali neke od mogućnosti administratora aplikacije opisati ćemo i moguće izgleda aplikacije nakon prijave doktora. Kad se doktor prijavi u aplikaciju otvori se forma MainDoctor.cs koja prima ID prijavljenog doktora u aplikaciju. Nakon sam prijave vrši se „select“ doktora po primljenom ID-u, te se poziva i metoda za „select“ svih pacijenata koji su pacijenti od tog prijavljenog doktora, ta metoda nam vraća listu svih traženih

pacijenata koji se ispisuju u GridControl kontrolu (kontrola DevExpressa). Slika koja se nalazi ispod nam prikazuje sučelje koje se otvori nakon prijave doktora u aplikaciju.



Slika 13 Doktor aplikacija

Prvi redak u GridControl je prazan te služi kao tražilica podataka kada je u aplikacije prevelik broj pacijenta i omogućava nam lakše korištenje aplikacije. Funkcionira kao tražilica svakog stupca. Npr. ako u stupac Name upišemo slovo „A“ on će nam ispisati sve pacijente kojima ime počinje s tim upisanim slovom. To vrijedi za svaki stupac tablice kao i za stupac Active koji nam omogućuje pretraživanje aktivnih i neaktivnih pacijenata. GridControl nam omogućuje još raznih mogućnosti kao što su grupiranje podataka, filtriranja, traženje te skrivanje i prikazivanje željenih stupaca u gridu.

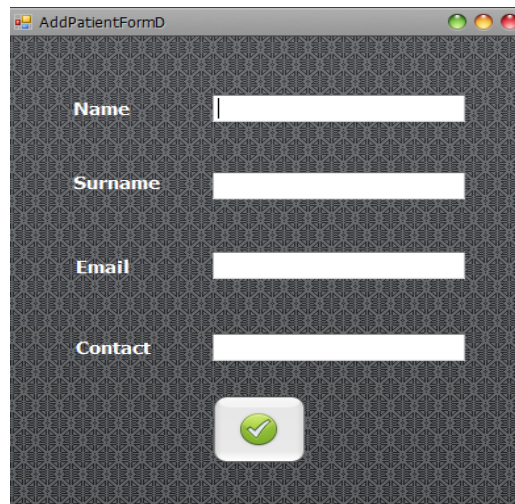
Na slici koja se nalazi ispod je prikazan primjer grupiranih podataka po prezimenu te je za razliku od prethodne slike skriven stupa s kontaktima.

Surname ▲			
Name	Email	Active	
▼			☑
☐ Surname: Bilos			
Adrea	andrea@gmail.com		☑
☐ Surname: Galic			
Gabrijela	gabi@gmail.com		☑
☐ Surname: Juric			
Mirjana	mira@gmail.com		☑
☐ Surname: Sabic			
Mijo	mijo.sabic1@gmail.com		☑

Slika 14 Grupirani podaci

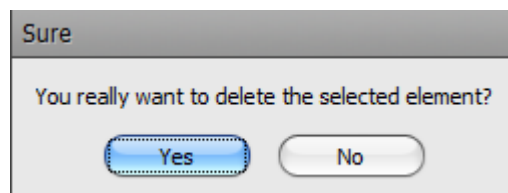


1. Ova opcija nam omogućava dodavanje novog pacijenta, kao što administrator ima mogućnost dodavanja novog pacijenta tako je ta mogućnost dana i prijavljenom doktoru. Za razliku od administratora doktor nema mogućnost dodavanja pacijenta drugim doktorima tj. znači da doktor koji je se prijavio u aplikaciju može dodati pacijenta samo se. Klikom na ikonu New patient otvori se dijalog za dodavanje novog pacijenta kao što je prikazan na slici ispod.



Slika 15 Dodavanje novog pacijenta

2. Nakon odabiranja željenog pacijenta u GridControlu koji se nalazi u sučelju aplikacije klikom na ikonu Edit patient otvara se sučelje koje omogućuje promjenu trenutno unesenih podataka o odabranom pacijentu.
3. Kod brisanja pacijenta stvar je također jednostavna odabere se željeni pacijent u bazi te se nakon klika na ikonu Delete patient pojavi MessageText gdje postoji mogućnost odabiranja željene akcije. Ako je akcija potvrđena odabrani pacijent će biti izbrisan iz baze te neće imati mogućnost prijave u mobilnu aplikaciju.



Slika 16 Provjera sigurnosti za brisanje

4. Na isti način kao i za editiranje i brisanje pacijenta, odaberemo ga iz GridControla te klikom na jednu od ikona (Patient info, Visit doctor, Referral, Diagnose, Perspection) otvara se određeni dijalog za dodavanje potrebnih informacija o pacijentu.
5. Pregled rezerviranih termina se vrši na dva načina. Rezervirane termine je moguće pogledati za odabranog pacijenta na način da odaberemo pacijenta te kliknemo na ikonu Agreed term for patient, a za sve sve rezervirane termine kliknemo na Agreed term.

## 5.4. Pacijent

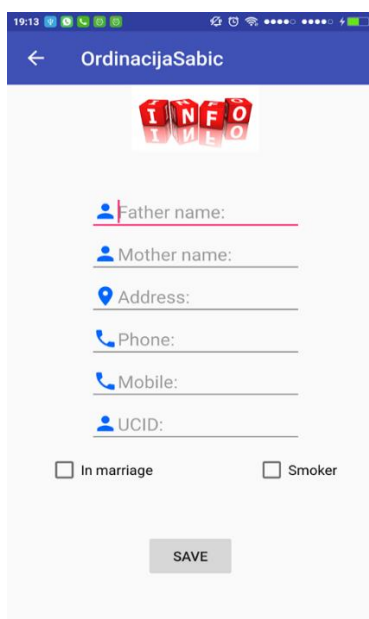
Aplikacija koja je namjenjena da je koriste pacijenti je izrađena u obliku mobilne aplikacije uz pomoć Adnroid studia i programskog jezika Java. Aplikacija se sastoji od tri dijela koja ćemo opisati pojedinačno, a djelovi su login, registracija i glavna aplikacija.

Kao što smo već opisali login u aplikaciju se sastoji od dva TextBox-a za unos email adrese i za unos lotinke. Nakon klika na dugme „Login“ provjerava se jesu li oba polja popunjena ako nisu ispiše se poruka „Sva polja trebaju biti popunjena!“, a u drugom slučaju ako jesu provjerava se jesu li uneseni podaci ispravni ako jesu okida se StringReques s određenim parametrima i čeka se odgovor. Ako smo privili željeni odgovor tj. ako nam se podudaraju dva hash koda onda ta osoba ima pravo pristupa u aplikaciju te se provjerava je li ona aktivirani član aplikacije ili ne.

```
String hashResult = md5( "1950th"+email+" "+password);
byte[] data = hashResult.getBytes(StandardCharsets.UTF_8);
String base64 = Base64.encodeToString(data, Base64.DEFAULT);
hashResult=base64.replace( target: "\n", replacement: "");
String hashResponse=response.split( regex: ";" ) [0];
int id_user=Integer.parseInt(response.split( regex: ";" ) [1]);
int active=Integer.parseInt(response.split( regex: ";" ) [2]);
if(hashResponse.equalsIgnoreCase(hashResult) )
{
    if(active==0){
        SharedPreferences.Editor editor = getSharedPreferences( name: "MyPrefsFile", MODE_PRIVATE).edit();
        editor.putString("userID", ""+id_user+"");
        editor.putString("IPAddress", IPAddress);
        editor.apply();
        Intent i = new Intent( packageContext: MainActivity.this, PatientInfoActivity.class);
        startActivity(i);
        finish();
    }
    else
    {
        SharedPreferences.Editor editor = getSharedPreferences( name: "MyPrefsFile", MODE_PRIVATE).edit();
        editor.putString("userID", ""+id_user+"");
        editor.putString("IPAddress", IPAddress);
        editor.apply();
        Intent i = new Intent( packageContext: MainActivity.this, MenuActivity.class);
        startActivity(i);
        finish();
    }
}
else
{
    MessageText("Please try again!");
}
```

Kod 14. Login

Nakon logiranja pacijenta i provjere jeli pacijent aktivan ili ne otvara se određeni prozor za pacijenta. Ako je pacijen nije aktivan potrebno je proći daljnu proceduru registracije kao što je popunjavanje svih osnovnih informacija o pacijentu kao što su ime oca, ime majke, adresa stanovanja, broj mobilnog telefona i fiksnog telefona, potrebno je upisati svoj JMBG i ako je pacijent ženjen potrebno je postaviti potvrdni znak pokraj „Relationship status“, a na isti nači je po potrebno postaviti potvrdu ako je pušač pokraj „Smoker“ u suprotnom ta dva polja ostaju prazna. Nakon što sva popunili sva potrebna polja klikom na dugme „Save“ potvrđujemo i pohranjujemo naše podatke u bazu.

The image shows a mobile application interface for a registration dialog titled "OrdinacijaSabic". At the top, there is a blue header bar with a back arrow and the title. Below the header is a red "INFO" banner. The form contains several input fields, each with a blue icon: "Father name:" (person icon), "Mother name:" (person icon), "Address:" (location pin icon), "Phone:" (phone handset icon), "Mobile:" (phone handset icon), and "UCID:" (person icon). Below these fields are two checkboxes: "In marriage" and "Smoker". At the bottom of the form is a grey "SAVE" button.

Slika 17 Dijalog registracije

Nakon pohrane podataka u bazu otvara nam se drugi dialog tj. dialog za promjenu lozinke, on se sastoji od od tri TextBox-a i jednog dugmeta. U prvi TextBox potrebno je upisati staru lozinku koja je poslana na email adresu a u dva preostala potrebno je napisati novu lozinku, a te se lozinke trebaju podudarati da bi se promjena uspješno izvršila. Kod same promjene lozinke vrši se i aktivacija korisnika te se polje provedena akcije korisnik treba ponovno prijaviti u aplikaciju s novom lozinkom.

```

public void ChangePassword(final String newUserPassword)
{
    RequestQueue queue = Volley.newRequestQueue( context: this);
    //String url = "http://192.168.31.146:3000/InsertPacijentDodatno";
    String url = "http://"+IPAddress+":3000/ChangePasswordPacijent";
    StringRequest postRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>()
        {
            @Override
            public void onResponse(String response) {
                // response
                if(response.equalsIgnoreCase( anotherString: "true")){

                    Intent i = new Intent( packageContext: ChangePassword.this, MainActivity.class);
                    i.putExtra( name: "userID", value: ""+userID+"");
                    startActivity(i);
                }
                Log.d( tag: "Response", response);
            }
        },
        new Response.ErrorListener()
        {
            @Override
            public void onErrorResponse(VolleyError error) {

                Log.d( tag: "ERROR", msg: "error => "+error.toString());
            }
        }
    ) {
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("id", ""+patient_class.getID_Pacijent()+"");
            params.put("password", newUserPassword);

            return params;
        }
    };
    queue.add(postRequest);
}

```

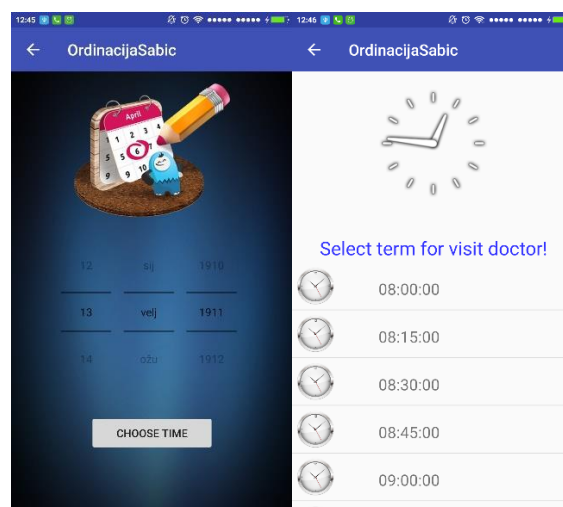
Kod 15. Metoda ChangePassword

Poslje promjene lozinke i ponovne prijave u aplikaciju ili u slučaju da je pacijent aktivira svoj račun otvara se glavni izbornik aplikacije u kojem se pacijentu omogućuju razne mogućnosti i rad s aplikacijom što je bio jeda od glavnih ciljeva ovoga rada. U glavnom izborniku koji izgleda kao ovaj što se nalazi na slici ispod pacijentu se omogućuju razne mogućnosti.



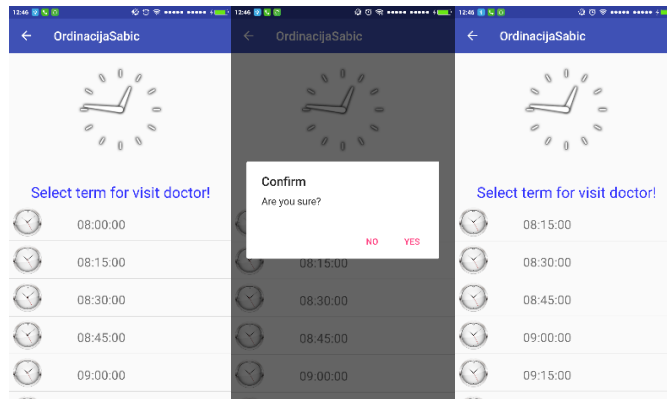
Slika 18 Menu

1. Kad u meniu kliknemo na ikonu koja se nalazi pod rednim broj 1 otvori nam se dialog gdje možemo odabrati željeni datum posjeta doktor tj. datum za rezerviranje termina. Kod odabira termina vrši se provjera valjanosti odabranog datuma ako je datum koji smo odabrali ispravan tj ako to spada u radne dane onda će nam se otvoriti novi dialog sa svim slobodnim terminima tog dana, a ako odabrani dan ne spada u radne dane ordinacije ispisat će se poruka „The selected date is not available!“ („Odabrani datum nije dostupan“).



Slika 19 Rezervacija termina

Na slici 19. Rezervacija termina lijevo prikazan je izgled dialoga za odabir željenog datuma, a na desnoj strani je prikazan dialog sa svim slobodnim terminima tog dana. Nakon odabira datuma na red dolazi odabir termina za taj dan. Odabir termina funkcioniše na sljedeći način: na popisu svih slobodnih termina odabere se željeni termin koji je ponuđen u listi termina. Klikom na željeni termin otvori se dialog gdje nas pita jesmo li sigurni da želimo rezervirati termin, odaberemo akciju NE onda se termin neće rezervirati a ako odaberemo akciju DA termin će se rezervirati te će se lista termina osvježiti i taj termin neće više biti u listi. Na slici ispod je prikazan navedeni postupak rezervacije termina. Na slici lijevo se nalazi popis prije rezervacije u sredini dialog potvrde kod rezervacije, a na kraju se nalazi popis termina nakon rezervacije.



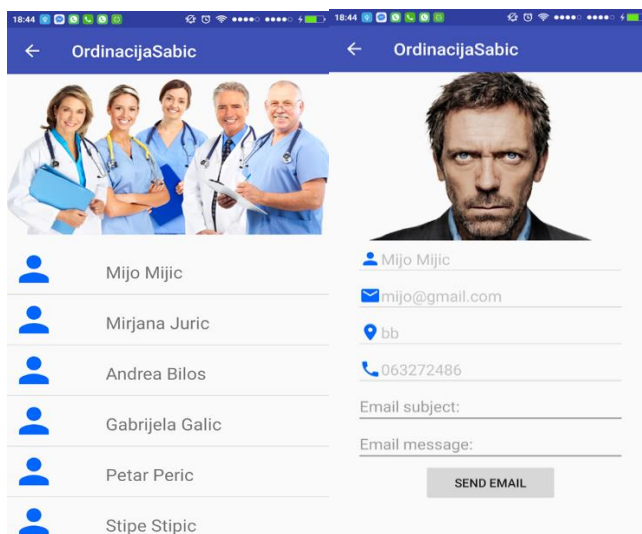
Slika 20 Rezervacija termina

2. Kad kliknemo na ikonu pod brojem 2 otvori nam se dijalog sa svim popisom dijagnoza koje je doktor upisao za nas. Activity dijagnoza sastoji se od dva dijela prvi dio je slika koja upućuje na to da se nalazimo u dijagnoza activity-u, a drugi dio je di ExpandableListView element koji prikazuje naslov dijagnoze, a klikom na njega otvara se padajući prozor u kojem se nalazi opis dijagnoze.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#ffffff" >
    <ImageView
        android:id="@+id/slika"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:maxHeight="100dp"
        android:minHeight="100dp"
        app:srcCompat="@drawable/diagnoses"/>
    <ExpandableListView
        android:id="@+id/lvExp"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:background="@drawable/back"/>
</LinearLayout>
```

Kod 16. ExpandableListView

3. Ako smo kliknuli na ikonu koja ima sličicu doktora onda nam se otvori dijalog koji sadrži popis svih doktora u našoj aplikaciji. Taj se dijalog također sastoji od dva dijela prvi dio je slika a drugi dio je ListView. Klikom na željenog doktora u ListView-u otvara nam se novi prozor u kojem se nalaze sve informacije o tom doktoru kao što su ime, prezime, email, adresa, kontakt broj i sl.. Kao dodatna opcija koju nam pruža ovaj prozor je slanje email poruke odabranom doktoru. Email poruka se šalje na način da se popune dva TextEdit-a, prvi TextEdit se odnosi na naslov poruke, a drugi TextEdit se odnosi na tekst poruke. Email se šalje pomoću Nodemailer\_a koji je opisan u poglavlju 4.2. .



Slika 21 Lista doktora

4. Klikom na ikonu pod brojem 4 otvori nam se popis svih recepata koje nam je propisao naš doktor
5. Klikom na ovu ikonu dobiti ćemo popis svih dosadašnjih uputnica kao i njihovo obrazloženje u padajućem prozoru
6. Kao i kod prethodnih ikona tako i kod ove klikom na nju dobijemo popis svih posjeta doktoru i kratki opis tj. obrazloženje koje je on zapisa prilikom posjeta

## 6. ZAKLJUČAK

Tema ovog diplomskog rada bila je implementacija android i desktop aplikacije za liječničku ordinaciju. Android aplikacija je izrađena u android studiu i može se upotrebljavati na mobilnim uređajima te je namijenjena uporabi pacijenata, a omogućuje im rezervaciju termina. Desktop aplikacija izrađena je uz pomoć Visual Studia 2015 i programskog jezika C#. Namijenjena je doktorima i administratorima ordinacije, a omogućuje im dodavanja, uređivanje i brisanje pacijenata, vođenje evidencije i slično.

Sigurnost korisnika i njihovih lozinki temelji se na pohrani u bazi podata u obliku MD5 hash koda.

U radu su također opisane sve tehnologije i alati koje su korištene prilikom izrade aplikacija.

Detaljan opis izrade i uporabe aplikacije može se pronaći u poglavu 5. Razvoj android i desktop aplikacije za liječničku ordinaciju.

Kao i kod većine sustava tako i kod ovog postoji mogućnosti za nadogradnju i dodatno napredovanje. Neke od stvari koje bih se mogle nadograditi su bolja organizacija termina i radnog vremena ordinacije onosno samog doktora, mogućnost fleksibilnosti termina kao i nadogradnja da pacijenti i doktori primaju obavijesti o sljedećem terminu (npr. termin je zakazan za sutra).



## Literatura

- [1] »Wikipedia,« 6 8 2018. [Mrežno]. Available: [https://bs.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://bs.wikipedia.org/wiki/Microsoft_Visual_Studio).
- [2] »Net-informations.com,« 6 8 2018. [Mrežno]. Available: <http://csharp.net-informations.com/>.
- [3] »Uvod u programiranje,« 6 8 2018. [Mrežno]. Available: <http://laris.fesb.hr/java/inheritance.htm>.
- [4] »Windows Forms Overview,« Microsoft, 6 8 2018. [Mrežno]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview>.
- [5] »Wikipedia,« 6 8 2018. [Mrežno]. Available: <https://en.wikipedia.org/wiki/DevExpress>.
- [6] »Developers,« 6 8 2018. [Mrežno]. Available: <https://developer.android.com>.
- [7] »Wikipedia,« 6 8 2018. [Mrežno]. Available: <https://hr.wikipedia.org/wiki/MySQL>.

## POPIS SLIKA

Slika 1 Svojstva formi.....	6
Slika 2 Toolbox .....	7
Slika 3 Pokretanje DevExpressa .....	8
Slika 4 Pokretanje aplikacije preko mobilnog uređaja.....	10
Slika 5 Dodavanje novog Activity-a .....	12
Slika 6 Modle baze .....	15
Slika 7 Prijava administratora.....	19
Slika 8 Login pacijent.....	21
Slika 9 Prijave u mobilnu aplikaciju .....	21
Slika 10 Sučelje za prikaz i određene akcije s doktorima .....	22
Slika 11 Class InsertDoctor.cs .....	22
Slika 12 Radno vrijeme .....	24
Slika 13 Doktor aplikacija .....	26
Slika 14 Grupirani podaci.....	26
Slika 15 Dodavanje novog pacijenta .....	27
Slika 16 Provjera sigurnosti za brisanje.....	27
Slika 17 Dijalog registracije .....	29
Slika 18 Menu .....	30
Slika 19 Rezervacija termina.....	31
Slika 20 Rezervacija termina.....	32
Slika 21 Lista doktora.....	33

## POPIS TABLICA

Tablica 1 Tipovi podataka cijelih brojeva.....	3
Tablica 2 Tipovi podataka realih brojeva .....	3

