# LimeDent App

## Login podaci

Username – test

Password – test

## Kratki opis

LimeDent aplikacija je aplikacija za managment dentalne ordinacije i njenih pacijenata te generiranje ponuda u PDF formatu i spremanja istih. Aplikacija je izrađena koristeći Java programski jezik. Za bazu podataka se koristi MySQL baza, a GUI sučelje je napravljeno koristeći custom Java Swing komponente. Kod programiranja same aplikacije se pratio MVC design pattern. Također, u izradi aplikacije su se koristili i Command i Observer design pattern-i te Multithreading, kako bi se osiguralo što bolje korisničko iskustvo.

## Kratka povijest

Aplikacija LimeDent koja je prezentirana u završnom projektu nastala je na zahtjev jedne dentalne ordinacije zbog potrebe za aplikacijom koja će uzimati input korisnika te generirati ponudu u PDF formatu.

Prva verzija aplikacije bila je napravljena s osnovnim Java Swing komponentama te je imala samo mogućnost izrade PDF ponuda te spremanje istih u datoteku na Desktop-u.

Tu dolazi do ideje za izradom cijele Desktop aplikacije s potrebnim svojstvima i mogućnostima potrebnim za vođenje dentalne ordinacije te njenih pacijenata.

## View – GUI dizajn

### Vanjski libraryji

Dizajn same aplikacije je napravljen vlastitom nahođenju s uputama potencijalnih krajnjih kupaca. Za potrebe toga, napravljene su i korištene već napravljene custom Java Swing komponente. Iste se mogu naći u ***CustomJavaSwing.jar*** i ***GlassPanePopUp_Raven_ikojic000.jar*** libraryjima koje se nalaze u ***/libs/Swing*** datoteci projekta. ***GlassPanePopUp_Raven.jar*** library je vanjska datoteka koja sadrži klase za korištenje skočnog prozora unutar same aplikacije kojim korisnik potvrđuje određenu radnju. Navedeni *library* je napravljen od strane osobe/a iza ***DJ-Raven*** profila na GitHub-u te ***Ra Ven*** Youtube kanala Na navedenim izvorima se može pronaći mnoštvo korisnih Java Swing komponenti, izrađenih GUI sučelja kao i upute kako ih napraviti te kako ih koristiti.

***CustomJavaSwing.jar*** library je vanjska datoteka sastavljena od custom izrađenih Java Swing komponenti te se može pronaći na mom GitHub profilu (https://github.com/ikojic000/JavaSwing). Komponente su napravljene i/ili sastavljene koristeći upute i postojeće komponente sa DJ-Raven profila i Youtube kanala te stavljene u jedan zajednički .jar file za lakše korištenje. Dokumentacija i upute za korištenje se također mogu pronaći kod DJ-Raven korisnika te je planirana izrada vlastite dokumentacije koja će

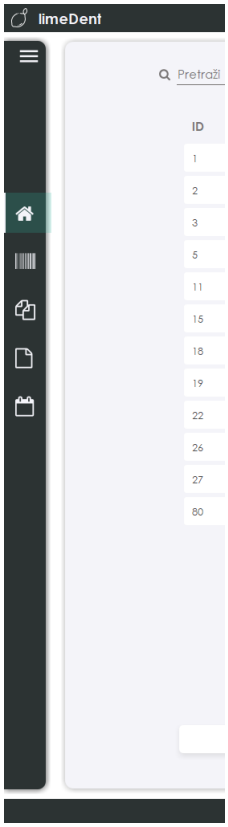
biti unutar navedenog libraryja za lakše korištenje. U libraryju se mogu pronaći custom Java Swing komponente korištene u izradi aplikacije, podijeljene u pakete po kategoriji kojoj pripadaju. Podijeljeno je na **button**, **comboBox**, **messageDialog**, **notification**, **panel**, **scroll**, **table/action table** i **txtInput** pakete s pripadajućim klasama i komponentama. Za navedene komponente su se koristili MiG Layout i GridBag Layout, SwingX te Timing Framework detaljno opisan u knjizi ***Filthy Rich Clients***.

Za ikone na meniju je korišten vanjski library ***gradient-icon-font.jar*** također preuzet sa DJ-Raven Github profila. Upute za korištenje se nalaze na navedenom profilu i Youtube kanalu.

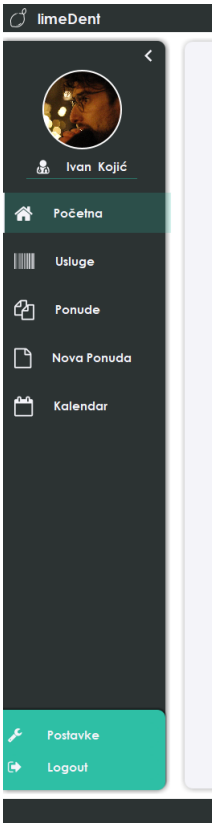
Da bi ***JFileChooser*** komponenta izgledala kao nativni Windows10 prozor, korišteni su ***jna-5.5.0.jar*** i ***swing-jnafilechooser.jar*** vanjski libraryji.

## Dizajn

Da bi aplikacija bila *full responsive* koristili su se ***GridBag Layout*** te ***MiG Layout*** (*Menu panel*). Koristeći navedene *layout manager*-e i ***TimingFramework*** se omogućilo korisniku da *Menu panel* može povećati te smanjiti po vlastitom ukusu ili potrebi.



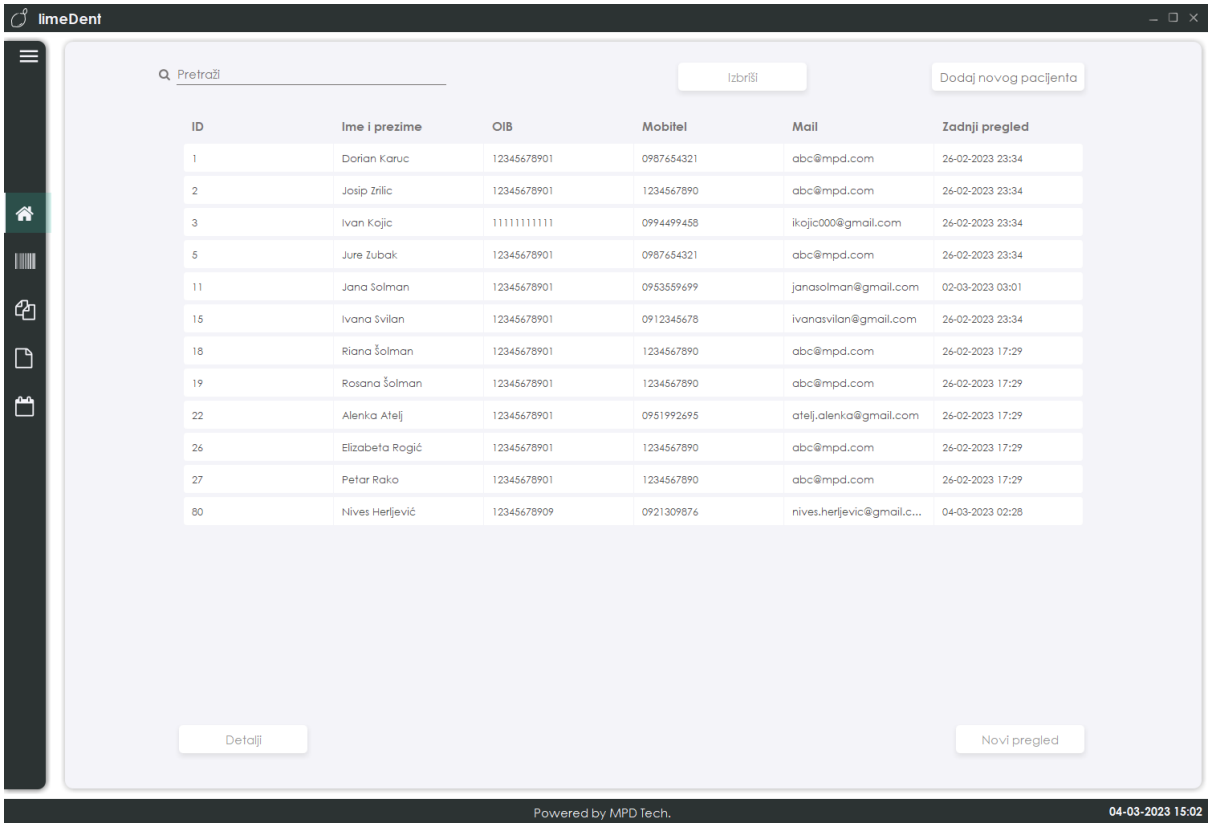

*MenuPanel 1*




*MenuPanel 2*

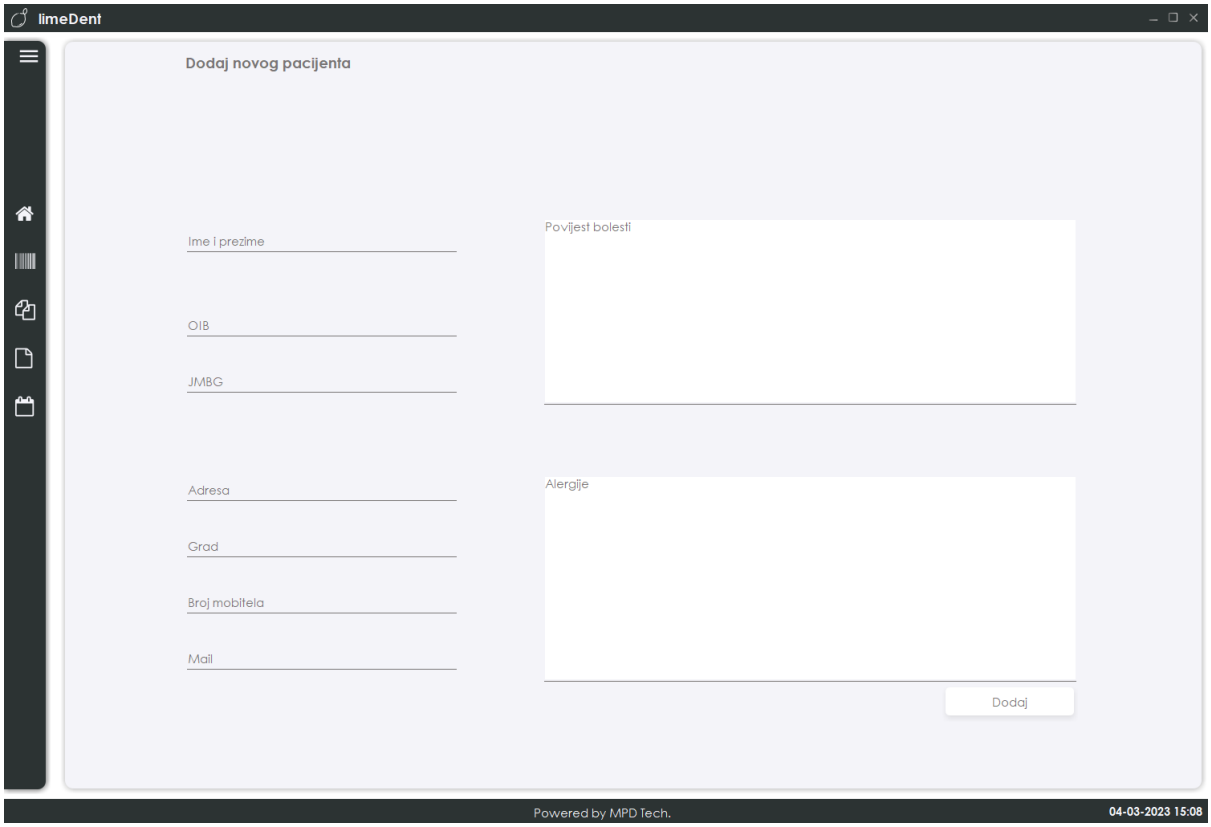
*Menu panel* se sastoji od ***header***, ***menu*** i ***footer*** dijelova. *Header* na sebi prikazuje podatke prijavljenog korisnika kao što su ime i prezime te profilna fotografija. *Footer* ima gumbe koje korisnika vode na *UserSettingsPanel* na kojem mijenja vlastite postavke te gumb za izlazak iz aplikacije. *Glavni / Menu* dio se sastoji od gumbova koji vode korisnika na u ostale dijelove aplikacije kao što su *HomePanel*, *UslugePanel*, *PonudePanel*, *NovaPonudaPanel* te *CalendarPanel*.

Svi paneli su dodani na jedan panel koristeći ***Card Layout*** te je tako riješen problem prebacivanja s panela na panel.



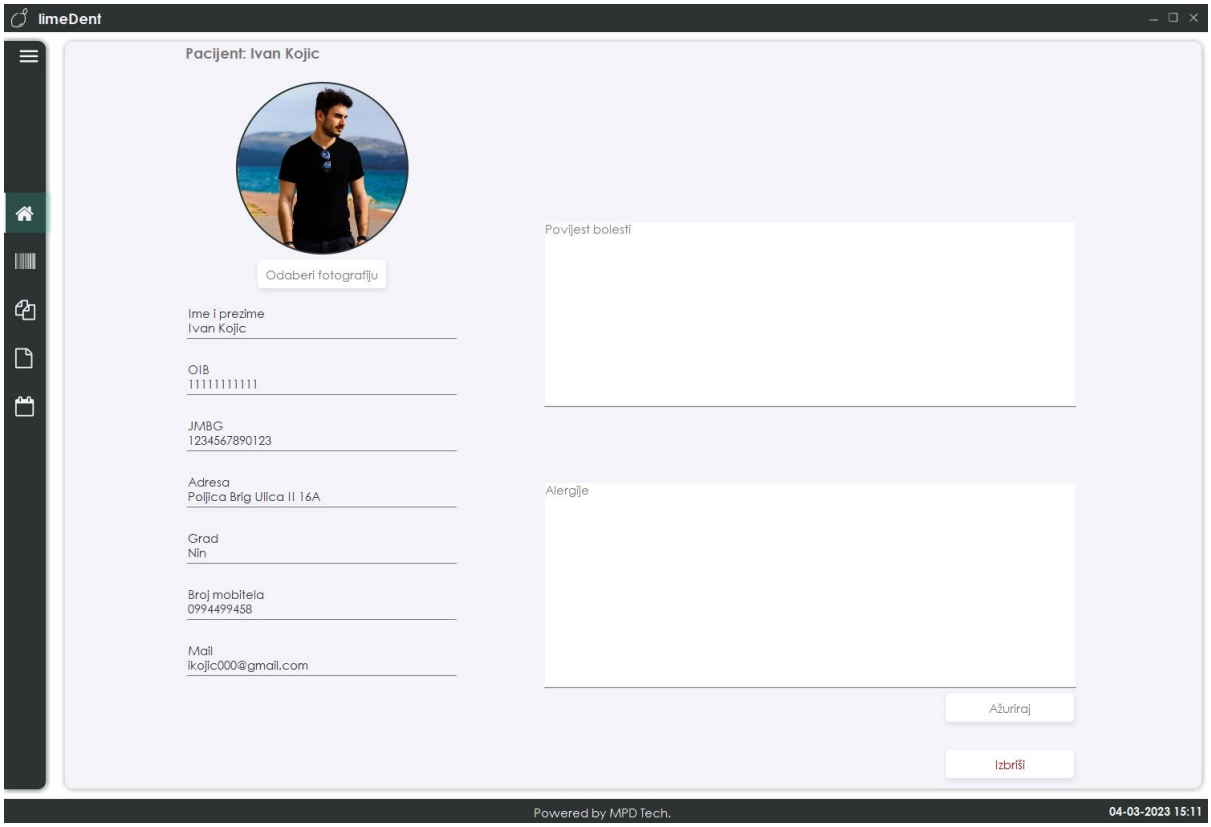

*HomePanel 1*

***HomePanel*** je prvo što korisnik vidi nakon uspješnog ulaska u aplikaciju. Na sebi prikazuje tablicu svih pacijenata s njihovim osnovnim podacima te vremenom zadnjeg pregleda. Također tu je i tražilica kojom korisnik može pretraživati pacijente unosom traženog imena i prezimena.



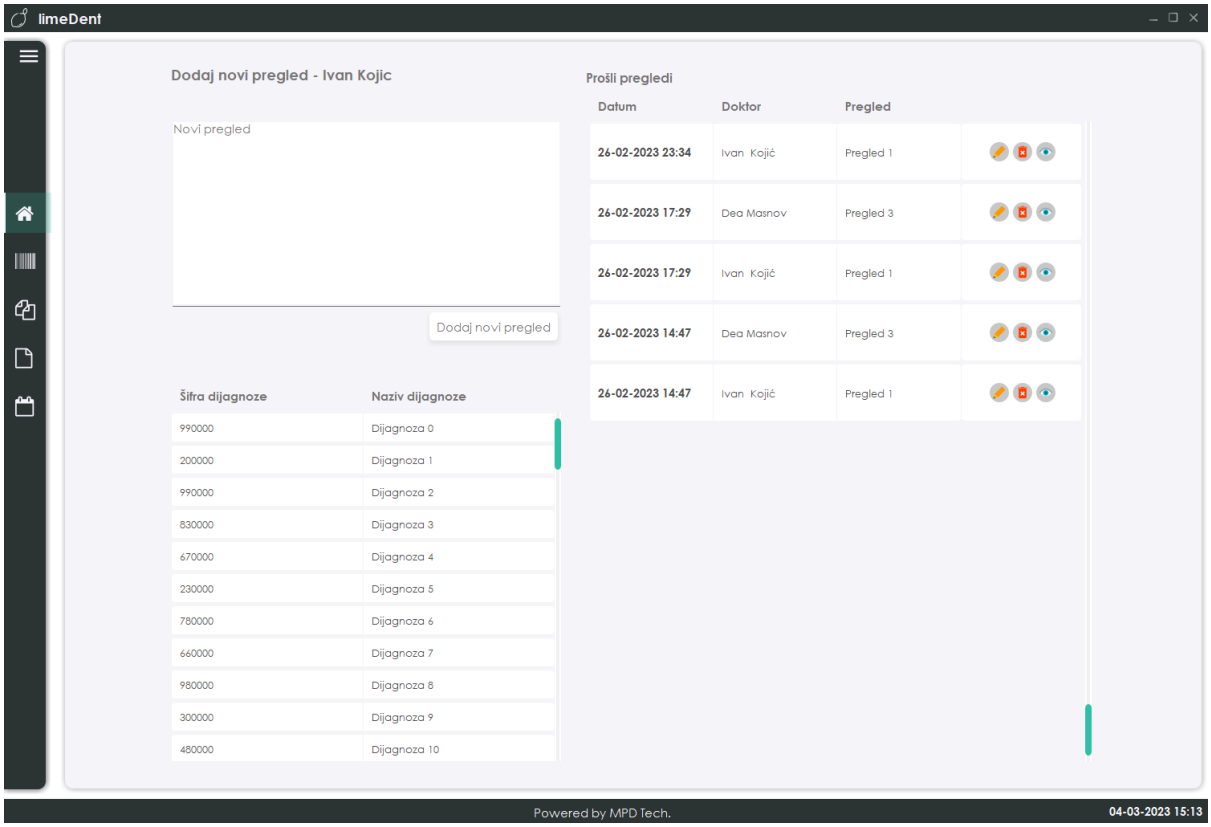

*NoviPacijentPanel 1*

Gumb „*Dodaj novog pacijenta*“ vodi korisnika na ***NoviPacijentPanel*** na kojem su polja za unos podataka o pacijentu (ime i prezime, OIB, JMBG, adresa, kontakt broj i mail, povijest bolesti te alergije) te koji služi za dodavanje novog pacijenta u bazu podataka aplikacije.

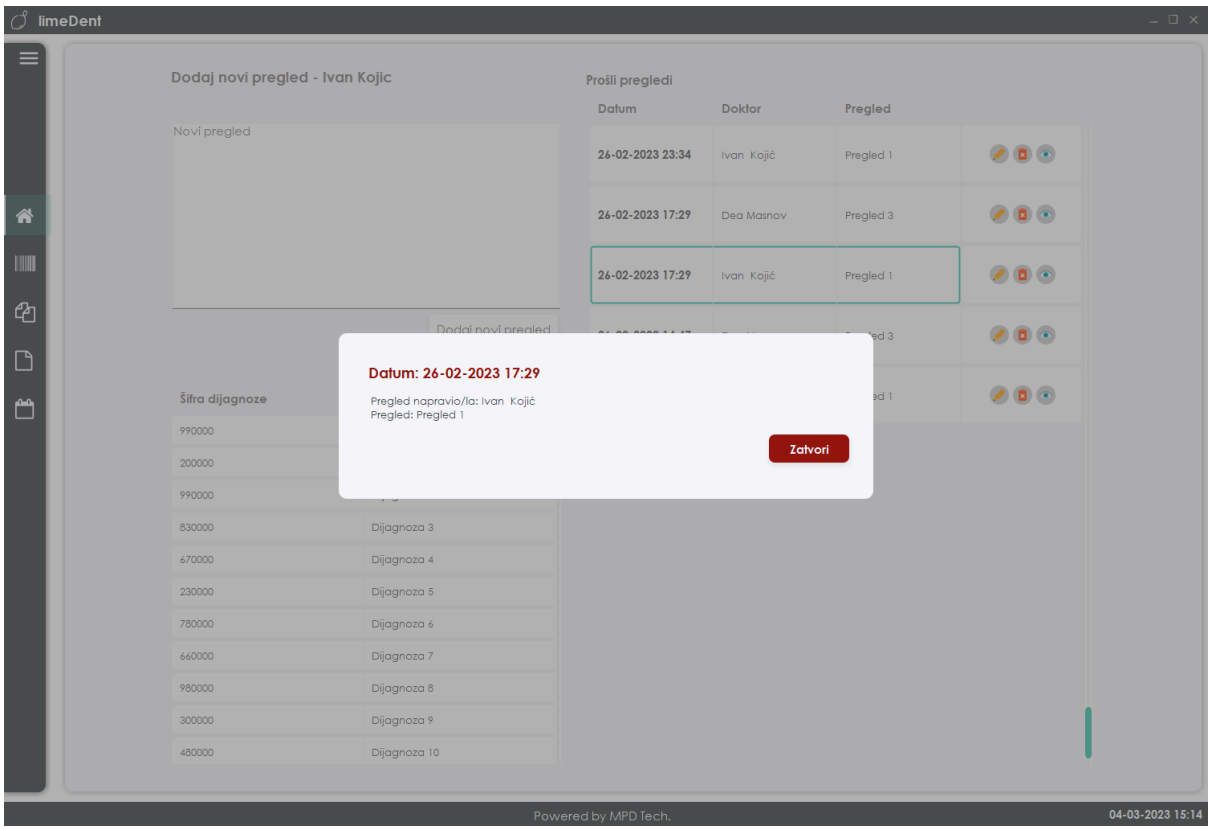



*DetaljiPanel 1*

Gumb „*Detalji*“ vodi korisnika na ***DetaljiPanel*** nakon odabira željenog pacijenta. Tamo može promijeniti podatke te fotografiju, kao i izbrisati odabranog pacijenta.

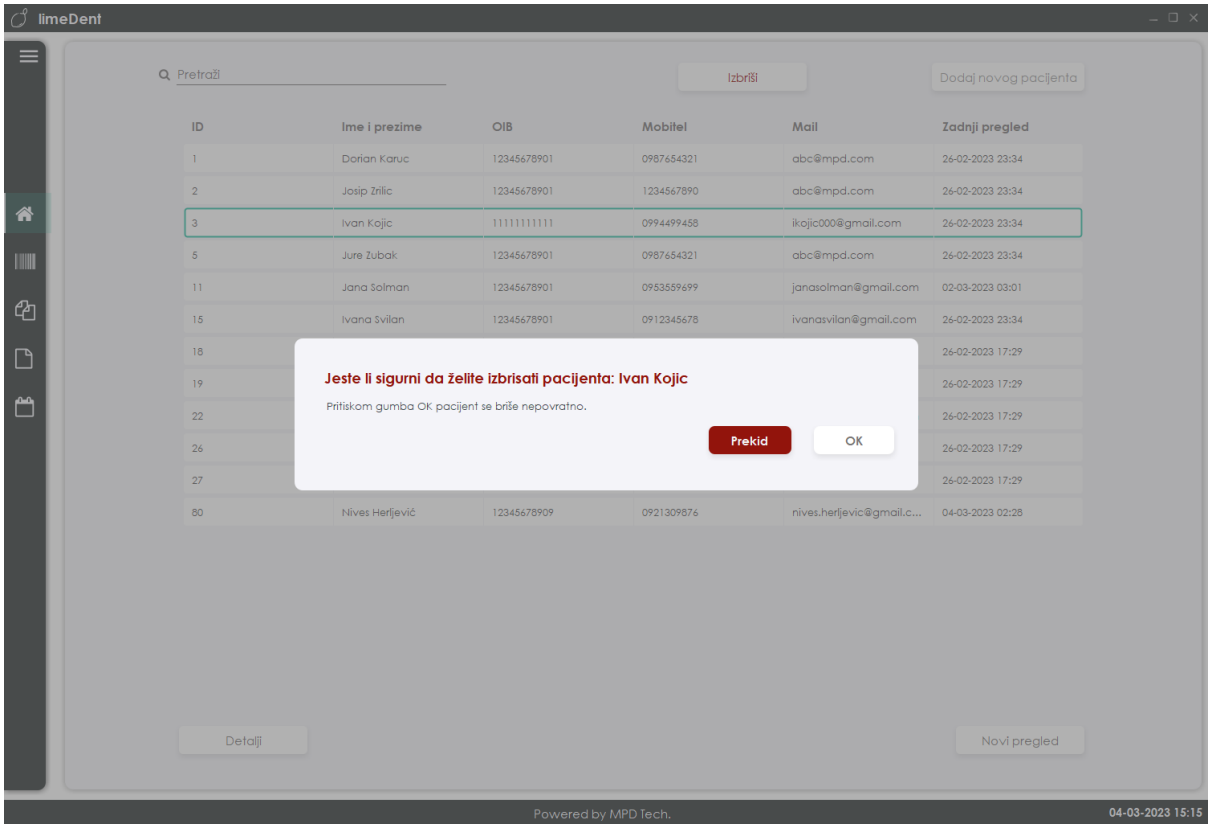



*NoviPregledPanel 1*



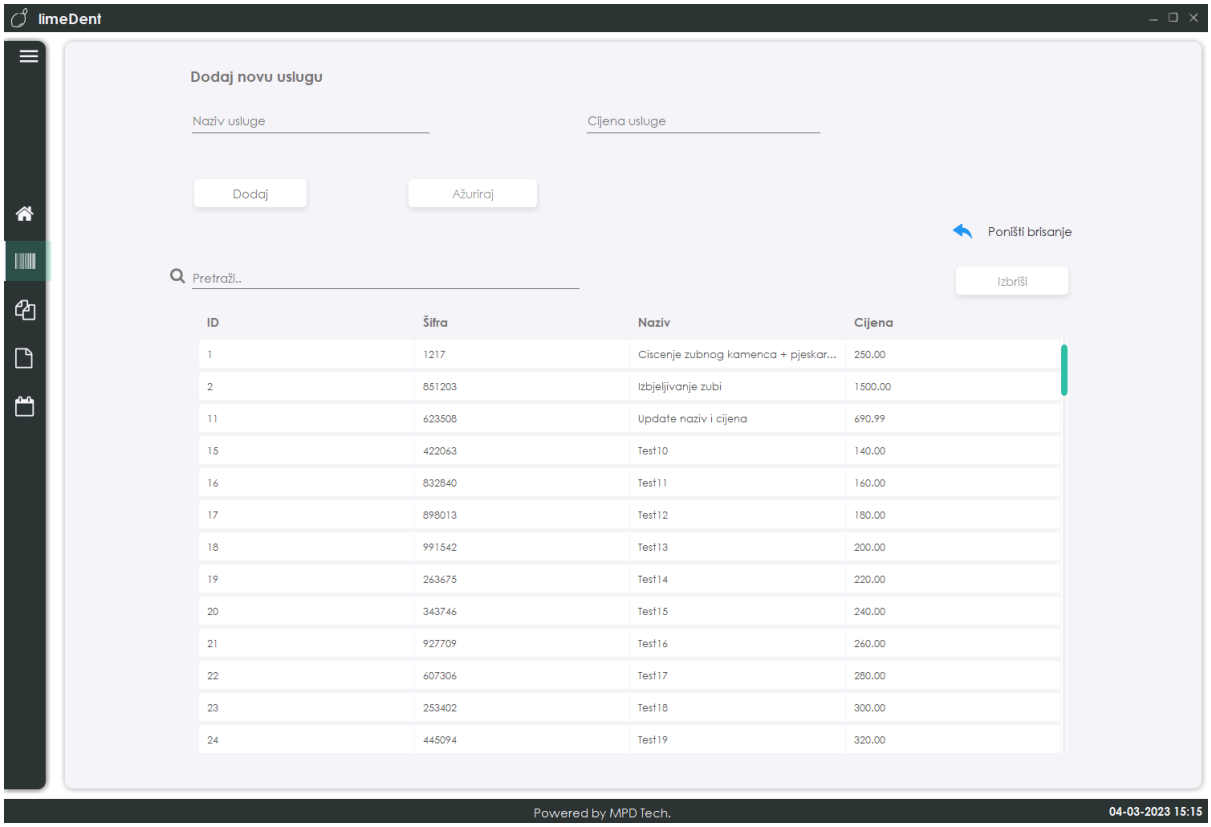

*NoviPregledPanel 2*

Gumb „*Novi pregled*“ vodi korisnika na ***NoviPregledPanel*** na kojem je pregled svih spremljenih pregleda odabranog pacijenta, input za dodavanje novog pregleda te tablica s mogućim dijagnozama koja popunjava input na dvostruki klik odabrane dijagnoze. Također korisnik ima mogućnost brisanja pregleda i otvaranja pregleda u skočnom prozoru.



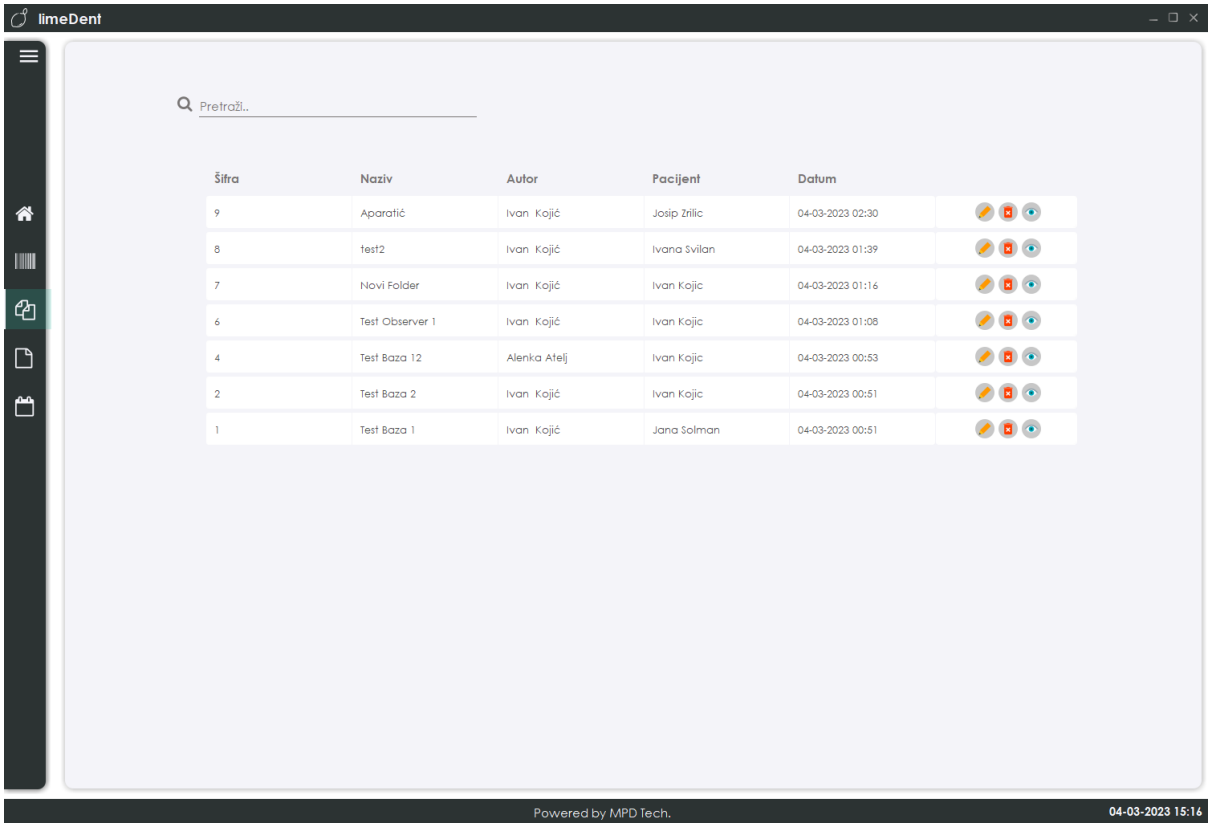

*HomePanel 2*

Gumb „*Izbriši*“ daje korisniku opciju brisanja odabranog pacijenta iz baze aplikacije.



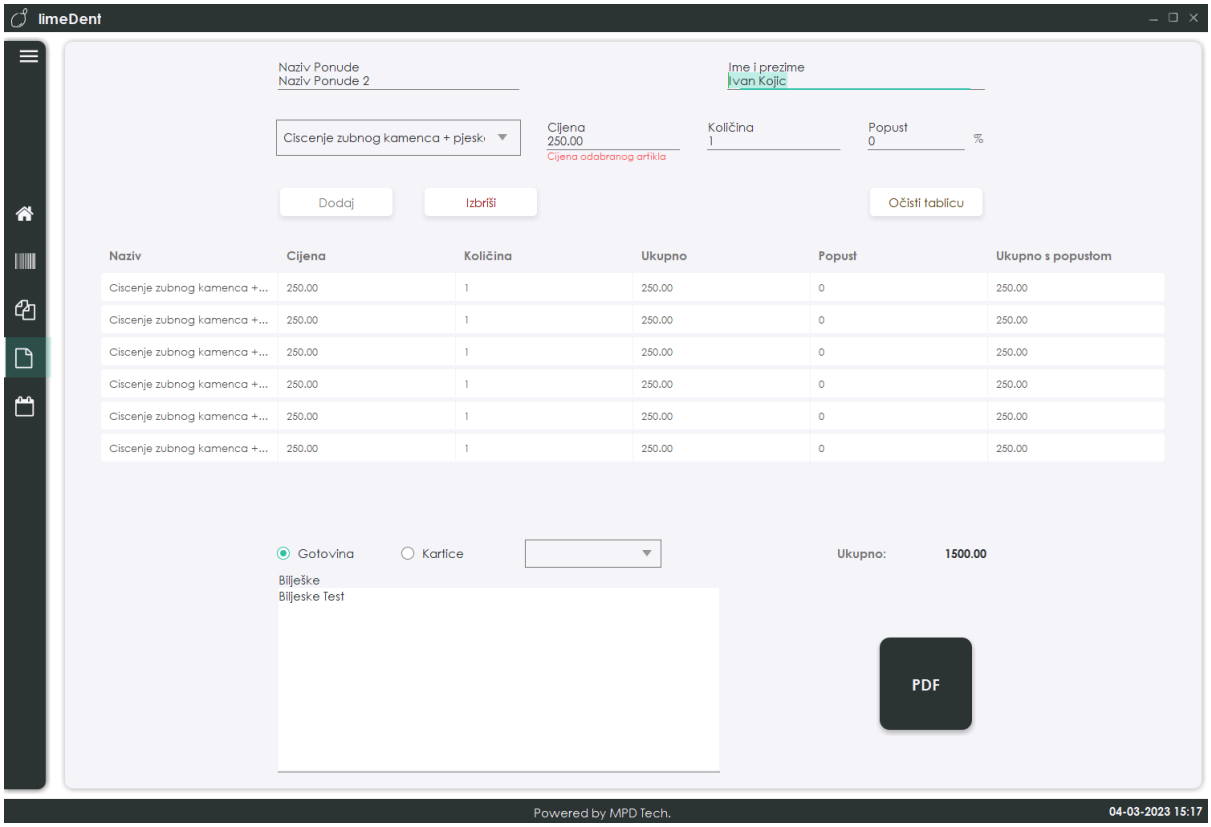

*UslugePanel 1*

***UslugePanel*** prikazuje tablicu sa svim uslugama koje ordinacija nudi. Korisnik ima opciju dodavanja nove usluge popunjavanjem polja te pritiskom „Dodaj“, ažuriranja odabrane ponude minjenjanjem vrijednosti na poljima te brisanja odabrane ponude. Također, korisnik ima opciju i povrata izbrisane usluge kao i mogućnost pretraživanja svih usluga pomoću naziva ili šifre.



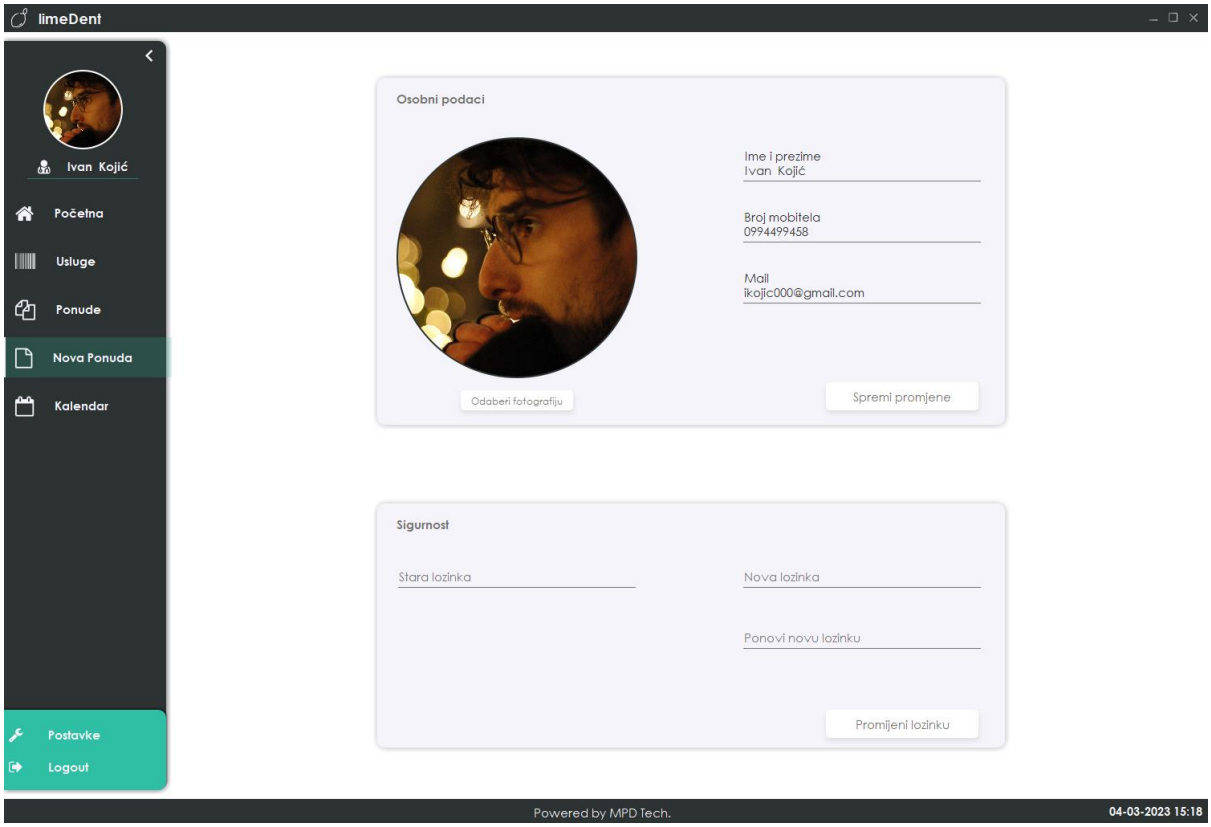

*PonudePanel 1*

***PonudePanel*** se sastoji od tablice koja prikazuje sve spremljene ponude. Korisnik ima mogućnost pretraživanja pomoću naziva ponude ili imena osobe za koju je ponuda bila namijenjena. Također, korisnik može navedenu ponudu izbrisati ili je otvoriti ako je spremljena na računalu.




*NovaPonudaPanel 1*

***NovaPonudaPanel*** sadrži nekoliko polja za popunjavanje kako bi ponuda bila kompletna. U „*Naziv ponude*“ upisuje se naziv ponude, a „*Ime i prezime*“ služi za unos imena i prezimena osobe za koju je ponuda namijenjena. To polje također ima mogućnost automatskog popunjavanja ako osoba već postoji u bazi pacijenata. Tu se nalazi i padajući izbornik koji sadrži sve usluge koje ordinacija nudi te također ima opciju automatskog popunjavanja. Polja za „*Količinu*“ i „*Popust*“ određuju, kako i sami kažu, količinu i popust odabrane usluge koja se u prezentacijsku tablicu dodaje gumbom „Dodaj“. Gumb „Briši“ služi za brisanje odabrane usluge iz prezentacijske tablice. Navedena tablica ima mogućnost ažuriranja količine i popusta odabrane usluge.
Nakon popunjavanja svih polja, i pritiskom gumba „*PDF*“, navedeni podaci se generiraju u PDF dokument koji se sprema na korisnikovo računalo te se odmah i otvara.  Također, napravljena ponuda se sprema u bazu.




*UserSettingsPanel 1*

***UserSettingsPanel*** na sebi ima nekoliko polja za promjenu korisnikovih podataka kao što su ime i prezime, broj mobitela, mail adresa te profilna fotografija. Također ima i polja za promjenu lozinke. Da bi se lozinka promijenila korisnik mora točno unijeti staru lozinku, te novu dva puta.

# Model

## Model u MVC

Kako je aplikacija napravljena prateći MVC design pattern, jedan od dijelova je i Model. Model klase su smještene u „Model“ paket te predstavljaju klase za objekte s kojima se u aplikaciji radi. Također tu se nalaze i klase „*Database*“ i „*DatabaseConnectionThread*“. Unutar paketa se nalazi i paket *tableModels.*

## Database

***Database*** klasa se nalazi u paketu Model te je prati *Singleton pattern*. Metoda *getDatabase()* se koristi za provjeru postoji li instanca klase u aplikaciji te ukoliko ne postoji ju kreira. Koristeći *Singleton pattern* osigurano je da tijekom životnog ciklusa aplikacije postoji samo jedna instanca klase. Također imamo metode za spajanje i zatvaranje konekcije. Podatke kao što su *DatabaseUsername* i *DatabasePassword* dohvaćamo iz dokumenta *config.properties* kako bi izbjegli direktno upisivanje podataka u sam kod.

***DatabaseConnectionThread*** klasa koja se koristi za stvaranje novog *Thread*-a koji se stvara pri pokretanju aplikacije. Navedena klasa, napravi novi *Thread* koji radi paralelno sa *EDT*

*Thread*-om te provjerava jeli uspostavljena veza s bazom. Ako veza nije upostavljena, pokušava ju opet uspostaviti.

### Model klase

Model klase predstavljaju klase za objekte s kojima aplikacija radi. Oni djelomično predstavljaju i tablice iz baze podataka. Modeli s kojima limeDent aplikacija radi su *User*, *Product*, *Patient*, *Offer*, *OfferTblPreviewData* i *MedicalExam*.

Svaka klasa u sebi sadrži odgovarajuća polja i atribute kao i razne konstruktore koji koriste određene atribute. Klasa User služi za kreiranje objekata koji predstavljaju krajnje korisnike, klasa Patient kreira objekte koji predstavljaju pacijente, MedicalExam predstavlja preglede pacijenata, Offer predstavlja ponude, a OfferTblPreviewData objekte koje stvaramo prilikom izrade ponude. Prezentacijska tablica u NovaPonudaPanel se popunjava objektima upravo ove klase.

### TableModels

*TableModels* paket u sebi sadrži klase za modele svih tablica korištenih u aplikaciji. Svaka klasa produžuje *AbstractTableModel* te ima svojstva koja trebaju tablici kojoj će pripadati.

***PatientTableModel*** pripada tablici na *HomePanel*-u za prikaz svih pacijenata, ***ProductTableModel*** pripada tablici na *UslugePanel*-u za prikaz svih usluga, ***MedicalExamTableModel*** tablici na *NoviPregledPanel*-u za prikaz pregleda odabranog pacijenta, ***OfferTableModel*** tablici na *PonudePanel*-u za prikaz svih ponuda te ***OfferTblPreviewModel*** tablici na *NovaPonudaPanel*-u u koju stavljamo objekte klase *OfferTblPreviewData* prilikom dodavanja usluge na ponudu.

### DAO

Paket ***DAO*** ,odnosno ***DataAccessObject*** sebi sadrži klase rad s podacima iz baze podataka. Kako u bazi podataka imamo tablice za *korisnike*(***users***), *usluge(**products***), *pacijente*(***patients***), *ponude*(***offers***) i *preglede*(***medicalExam***) tako imamo i DAO klase za sve te tablice. Unutar navedenih klasa imamo metode za dohvaćanje svih podataka iz tablice, za dohvaćanje podataka po zadanim uvjetima(npr. ID), dodavanje novih podataka u tablice kao i ažuriranje i brisanje željenih podataka. *DAO klase* kreiraju odgovarajuće objekte i/ili liste objekata *model* klasa te se koriste u *controller* klasama.

## Controller

### Controller u MVC

***Controller*** paket sadrži klase koje se koriste kao kontroler klase i kao spona između *View* i *Model* dijela aplikacije. Kontroler klase manipuliraju objektima *model* klasa te ih prosljeđuju ili prikazuju na *View* panelima.

### Controller klase

Svaki *ViewPanel*, odnosno svaka *View* klasa ima svoju kontroler klasu, neke čak i koriste više od jedne kontroler klase. Nazvane su tako da se lako raspozna koja kontroler klasa odgovara kojem *View* panelu (npr. ***HomeController*** – upravlja objektima ***User*** *model* klase te

ih prikazuje na ***HomePanel***-u). Kontroler klase koriste DAO klase kako bi dohvaćene podatke iz baze i spremljene u liste objekata prikazali i/ili manipulirali njima. Svaka kontroler klasa ima listu objekata s kojima radi nešto od navedenog. Također kontroler klase imaju razne metode za upravljanje i manipuliranje komponentama *ViewPanel*-a te se pozivaju u *View* klasama.

## Design pattern-i

Unutar *controller* paketa nalaze se i dva paketa nazvana ***command*** i ***observer.*** Unutar istih su klase i sučelja korištena za implementaciju istoimenih *design pattern*-a.

### Command Design Pattern

Aplikacija implementira ***Command Design Pattern.*** Kako je korisniku omogućeno brisanje usluga koje ordinacija pruža može doći do problema nastalog slučajnim brisanjem krive usluge. Kako usluga ima mnogo, može doći do korisničke pogreške te krivog odabira. Da bi korisnik bio osiguran od neželjenog ili krivog brisanja usluge korišten je *Command Design Pattern.* Unutar **UslugeController,** kontroler klase za ***UslugePanel*** postoji *stog(Stack)* koji prima ProductCommand tip objekta. Prilikom klika na gumb za brisanje kreira se novi objekt klase ***DeleteProductCommand*** koji u sebi sadrži podatke o izbrisanoj usluzi te metode za brisanje usluge iz baze podataka. Klikom „*undo*“ gumba na *UslugePanel*-u, poziva se *undo()* metoda na zadnje dodani objekt u zadani stog, odnosno iznova se doda usluga u bazu aplikacije te se poništi krivo i/ili neželjeno brisanje.

### Observer Design Pattern

*Observer Design Pattern* se koristi na tri različita mjesta unutar aplikacije. Kako postoje *ViewPanel*-i i kontroler klase koje mijenjaju svojstva objekta korištenog i prikazanog na više *ViewPanel*-a koristi se *Observer Design Pattern* kako bi obavijestio ostale kontroler klase i *View* klase o promjeni. U limeDent aplikaciji se koristi ***UserObserver***, ***PatientListObserver*** i ***OfferListObserver*** i ***OfferTblPreviewObserver***.

***UserObserver*** je sučelje koje određuje metode za ažuriranje korisnikove profilne fotografije i podataka tako da se prilikom promjene ažurira slika na *UserSettingsPanel*-u te na *Menu* panelu, kao i ime i prezime. Navedeno sučelje implementiraju *Header* (*Header panel* unutar *Menu panel*-a) i *UserSettingsPanel* klase koje pozivaju metode za ponovno dohvaćanje korisnikove profilne fotografije i prikazivanje te nove fotografije. Nakon odabira nove fotografije na *UserSettingsPanel*-u poziva se metoda *notifyObservers()* koja poziva upravo te metode za ponovno postavljanje fotografije. Također, ista metoda se poziva prilikom promjene podataka tako da se odmah promijeni korisnikovo ime na tim panelima. U ovom slučaju *User* klasa u *model* paketu služi kao i *Observable* klasa.

***PatientListObserver*** je sučelje koje određuje metodu za ažuriranje liste pacijenata prilikom dodavanja novog pacijenta. Također, imamo i klasu ***PatientListObservable*** koja služi za dodavanje i brisanje *Observer* objekata u/iz liste te za određivanje *notifyObservers()* metode. Ta klasa prati *Singleton Pattern* tako da smo sigurni da je samo jedna instanca klase kreirana tijekom životnog ciklusa aplikacije i da smo sigurni da na točnom objektu pozivamo potrebne metode. ***HomeController***, kontroler klasa implementira navedeno sučelje te poziva metode koje ažuriraju listu pacijenata.

***OfferListObserver*** je sučelje koje određuje metodu za ažuriranje liste ponuda prilikom dodavanja nove ponude u bazu aplikacije. Imamo i klasu ***OfferListObservable*** koja prati

*Singleton Pattern* i služi za dodavanje i brisanje *Observer* objekata i određivanje *notifyObserver()* metode. ***PonudeController*** klasa implementira to sučelje te poziva metode koje ažuriraju listu ponuda.

***OfferTblPreviewObserver*** sučelje se koristi kod promjena tablice kod izrade nove ponude. Klasa ***PonudeController*** ima metodu koja računa ukupnu vrijednost odabranih i unesenih artikala / usluga te osvježava UI komponentu *(JLabel)* s vrijednosti ukupnog iznosa. Kada korisnik promijeni neku od vrijednost u tablici koristimo *OfferTblPreviewObserver* kako bi osvježili odgovarajuću UI komponentu.

## PDF Generator

***PDFGenerator*** klasa u „*controller*" paketu je klasa koja se koristi za generiranje PDF dokumenata. Klasa koristi iTextPDF (verzija 5.5.13.3) vanjski library koji služi za upravljanje PDF dokumentima. Da bi navedeni library i navedena klasa radili potrebno je imati i ***activation.jar*** library uz ***itextpdf-5.5.13.3.jar***. Konstruktor klase PDFGenerator prima String koji predstavlja *url* do datoteke gdje će generirani PDF dokument biti spremljen, objekt klase *Offer* iz kojeg se potrebni podaci čitaju i stavljaju u PDF dokument te objekt klase *OfferTableModel* iz kojeg se čitaju usluge koje se stavljaju na PDF dokument. Klasa / *Library* generira dokument tako da koristi *PdfCell* objekte, te kako im i sam naziv kaže sastavljaju ćelije. Klasa ima metode za dizajn potrebnih ćelija. Uz navedene metode, klasa ima metode ***generatePDF()*** te ***openPDF()*** koje se pozivaju u konstruktoru klase te služe za generiranje i otvaranje PDF dokumenta.

## Multithreading

Kako bi se izbjeglo potencijalno blokiranje korisničkog sučelja, aplikacija koristi *multithreading* za radnje koje traju vremenski duže, koje trebaju više resursa ili predstavljaju potencijalni uzrok za blokiranje korisničkog sučelja.

Jedan od potencijalnih uzroka je gubljenje veze s bazom podataka. Kako bi se to izbjeglo, ***DatabaseConnectionThread*** služi kako bi kreirala odvojeni *thread* u kojem paralelno uz ***EDT thread***(*thread kojeg koristi Swing*) svakih 15 sekundi provjerava postoji li veza s bazom podataka te ukoliko ne postoji, pokušava je uspostaviti. Klasu iniciramo u Start klasi kao i *View* klasu.

Da bi se spriječila nepotrebna količina podataka, odnosno veliki podaci u bazi podataka, profilne fotografije pacijenata trebaju biti manje veličine, odnosno fotografije je potrebno kompresirati prije spremanja u bazu. Kako kompresija slike može biti zahtjevniji zadatak za aplikaciju i može trajati duže napravljena je klasa ***ImageCompressionThread***.

Navedena kasa kreira odvojeni *thread* u kojem se odvija kompresija prije spremanja u bazu. Nakon što kompresija završi, korisnik dobije obavijest kako je slika spremljena.

Kompresija ne smanjuje dimenzije fotografije, već smanjuje njenu kvalitetu. Kako je *panel* u kojem se prikazuje fotografija pacijenta manjih dimenzija, kvaliteta nije toliko bitna. Za kompresiju se koristi ***ImageIO*** klasa. Ako je odabrana fotografija veća od 512kB, metoda ***run()*** navedene klase stavlja ***ImageWriteParam*** na vrijednost ***0.5*** te provjeri veličinu novo napravljene slike. Ako je veličina veća i dalje od 512kB, petlja se ne prekida nego se *ImageWriteParam* smanjuje za 0.05 te se radi nova slika. Petlja se ponavlja sve dok slika ne

bude manja ili jednaka 512kb ili dok vrijednost *ImageWriteParam* ne postane manja od 0.05 nakon čega kompresija postaje nemoguća.

Klasa *ImageCompressionThread* se poziva kada korisnik potvrdno odabere sliku koju želi spremiti. Tu se koristi i ***SwingWorker*** klasa koja ima metode ***doInBackground()*** i ***done()***. Pomoću te klase možemo kreirati novi *thread* u pozadini dok *EDT thread* nesmetano radi. Pri završetku zadatka, navedeni *thread* se gasi te se izvršava metoda *done()*.

*FooterBar* panel također koristi *multithreading* u obliku ***TimerTask*** klase kako bi svakih 100ms osvježio i prikazao trenutno vrijeme korisniku. *TimerTask* je najbolje koristiti kod ponavljajućih stvari kao što je i u ovom primjeru.

## Sigurnost

Sigurnost je jedan od bitnih faktora kod poslovnih aplikacija. Lozinke u bazi nisu spremljene u izvornom obliku već je spremljen njihova *hash* verzija. Lozinke se *hash*-iraju uz pomoć **SHA-256** algoritma sa generiranim ***salt***-om. Pri ulasku u aplikaciju vrši se provjera korisničkog imena i lozinke tako da se dohvati *hash*-irana lozinka iz baze podataka, uzme se njen *salt*, *hash*-ira se upisana lozinka te se rezultati uspoređuju.

Također prilikom mijenjanja lozinke, stara lozinka se uspoređuje na isti način kao i prilikom ulaska u aplikaciju te se nova *hash*-ira s postojećim saltom.

## Glavne značajke

Glavne značajke limeDent aplikacije:

**Korisnici**:

- Prijava
- Odjava
- Promjena lozinke
- Promjena podataka i slike

**Sigurnost**:

- *Hashed* lozinka u bazi
- SHA-256 i generirani *salt*
- Provjera i mijenjanje lozinke uz pomoć postojećeg *salt*-a

**Pacijenti**:

- Dodavanje
- Brisanje
- Ažuriranje podataka i slike
- Pretraživanje
- Pregled svih pregleda po pacijentu
- Dodavanje novih pregleda
- Brisanje postojećih pregleda

**Ponude**:

- Kreiranje nove ponude
- Spremanje ponude u datoteku na računalu te otvaranje iste
- Pretraživanje
- Otvaranje postojećih ponuda
- Brisanje