**Sveučilište u Mostaru**

**Fakultet prirodoslovno-matematičkih i odgojnih znanosti  
Studij Informatike**

Matej Madžarević

**JAVA DESKTOP APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA**

Diplomski rad

Mostar, 2021.

**Sveučilište u Mostaru**

**Fakultet prirodoslovno-matematičkih i odgojnih znanosti  
Studij Informatike**

Matej Madžarević

**JAVA DESKTOP APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA**

Diplomski rad

Mentor: dr. sc. Dražena Gašpar

Student: Matej Madžarević

Mostar, veljača 2021.

Ovaj diplomski rad izrađen je na Studiju informatike Fakulteta prirodoslovno-matematičkih i odgojnih znanosti Sveučilišta u Mostaru pod vodstvom dr. sc. Dražene Gašpar.

Sadržaj:

[1. UVOD 1](#_Toc64777965)

[2. PROGRAMSKI JEZIK JAVA 3](#_Toc64777966)

[2.1. Osnovne karakteristike Java jezika 4](#_Toc64777967)

[2.2. Prednosti i nedostaci programskog jezika Java 7](#_Toc64777968)

[2.3. Naredbe u Java-i 9](#_Toc64777969)

[2.4. Java platforma 17](#_Toc64777970)

[2.5. Integrirano razvojno okruženje NetBeans 20](#_Toc64777971)

[2.6. Ostala razvojna okruženja za Java-u 22](#_Toc64777972)

[2.7. MVC Arhitektura 25](#_Toc64777973)

[3. JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA 27](#_Toc64777974)

[3.1. Opis problema 27](#_Toc64777975)

[3.2. Relacijski model baze podataka 28](#_Toc64777976)

[3.3. Dijagram klasa 29](#_Toc64777977)

[3.4. Prozori aplikacije 32](#_Toc64777978)

[3.4.1. Prijava 32](#_Toc64777979)

[3.4.2. Evidencija vremena - Administrator 34](#_Toc64777980)

[3.4.3. Pregled korisnika- Administrator 35](#_Toc64777981)

[3.4.4. Dodaj korisnika - Administrator 39](#_Toc64777982)

[3.4.5. Izmjeni korisnika - Administrator 40](#_Toc64777983)

[3.4.6. Početna - Korisnik 41](#_Toc64777984)

[3.4.7. Evidencija vremena - Korisnik 45](#_Toc64777985)

[3.4.8. Postavke - Korisnik 46](#_Toc64777986)

[3.5. MVC arhitektura u aplikaciji 46](#_Toc64777987)

[3.6. Slučajevi korištenja 51](#_Toc64777988)

[3.7. Mogućnosti proširenja aplikacije 53](#_Toc64777989)

[4. ZAKLJUČAK 54](#_Toc64777990)

[5. LITERATURA 55](#_Toc64777991)

**SAŽETAK**

Temeljna svrha ovog rada je istraživanje Java programskog jezika kako bi se uspješno implementirala Java desktop aplikacija za evidenciju radnog vremena. Istraživanje je provedeno kroz teorijski i praktični dio rada.

Cilj ovog rada bio je implementirati funkcionalnu desktop aplikaciju za evidenciju radnog vremena koristeći Java programski jezik za implementaciju desktop aplikacije u integriranom razvojnom okruženju za Java-u, odnosno u Netbeans-u.

Za kreiranje i oblikovanje dizajna aplikacije je korišten JavaFX Scene Builder 2.0 koji pruža znatno veće mogućnosti od Swing frameworka, dok je za bazu podataka korištena MySQL baza podataka iz razloga što je najkompaktnija odnosno najlakša je za preuzimanje, instalaciju i korištenje.

Bitno je naglasiti kako je aplikacija urađena po MVC (Model-View-Controler, Model-Pogled-Upravitelj) obrascu software-ske (software-programska podrška) koji uvelike olakšava brzi razvoj računalnih programa i aplikacija, a koristi se i u brojnim drugim programskim jezicima.

Sustav korišten za implementaciju aplikacije je Windows 10, ali aplikaciju je moguće pokrenuti i na bilo kojem drugom uređaju koji ima instaliranu Java podršku.

**Ključne riječi**: Java programski jezik, Java platforma, MVC arhitektura, Java desktop aplikacija za evidenciju radnog vremena

# UVOD

U svakom ozbiljnom poduzeću, instituciji ili čak organizaciji (osobito u onima koje imaju veliki broj zaposlenika i čija su radna vremena ponekad i fleksibilna) potrebno je voditi evidenciju radnog vremena. Radna vremena zaposlenika je potrebno evidentirati iz više potencijalnih razloga poput npr. precizne isplate plaće, provjeravanja dolaska zaposlenih, provjeravanja intenziteta i obujma posla i sl.

Kroz povijest su se zaposlenici susretali s raznim oblicima evidencije radnog vremena, sve od klasičnog potpisivanja olovkom na papir pri dolasku na posao, preko kartica koje se provlače na ulasku u poslovne prostore pa sve do današnjih najmodernijih sustava koje imaju velike i uspješne kompanije, a koji su u mogućnosti putem primjerice računalnog vida automatski evidentirati svoje zaposlenike i zabilježiti od kada do kada je svaki od njih bio na poslu.

Upravo je problematika efikasne evidencije radnog vremena predmet istraživanja ovog diplomskog rada, a cilj rada je, kao što je već navedeno u sažetku, implementacija jednostavne Java desktop aplikacije koja omogućava efikasnu evidenciju radnih vremena svih zaposlenika u tvrtki.

Aplikacija je zamišljena tako da zaposlenici prilikom dolaska na svoje radno mjesto, a pod uvjetom da su prije toga registrirani u bazi podataka, u login formu unesu svoje korisničko ime i lozinku, kliknu mišem gumb koji označava početak radnog vremena, a aplikacija će im (u realnom vremenu) odbrojavati vrijeme provedeno na poslu.

Također je zamišljeno da korisnici (zaposlenici) imaju mogućnost pauze, odnosno pauziranja odbrojavanja radnog vremena, a shodno s tim i prestanak pauze.

Naravno, na kraju radnog vremena, zaposlenici moraju imati mogućnost završetka odbrojavanja radnog vremena jednostavnim klikom miša na gumb.

Svaki zaposlenik mora imati mogućnost pregleda svojih radnih vremena uz informaciju kojeg datuma i u kojem vremenskom intervalu je bio na poslu.

Sve zaposlenike u bazu podataka unosi administrator (ili više njih), koji također ima mogućnost pregleda postojećih korisnika i njihovih radnih vremena, ali i mogućnosti izmjene ili brisanja istih.

Za implementaciju je iz više razloga korišten Java programski jezik, prvenstveno zato što je jednostavan za korištenje te je lagan za naučiti, ali i zbog toga što ima moćne razvojne alate uz pomoć kojih se bez velikih poteškoća mogu razviti korisne aplikacije bogate dizajnom.

U prvom dijelu rada je pobliže opisan Java programski jezik, točnije njegove osnovne karakteristike, prednosti (ali i pojedini nedostatci), naredbe koje se koriste u njemu, Java platforma te razvojna okruženja u kojima se upotrebljava Java. U zasebnom odlomku detaljnije je objašnjen MVC arhitektonski obrazac koji olakšava razvoj aplikacija.

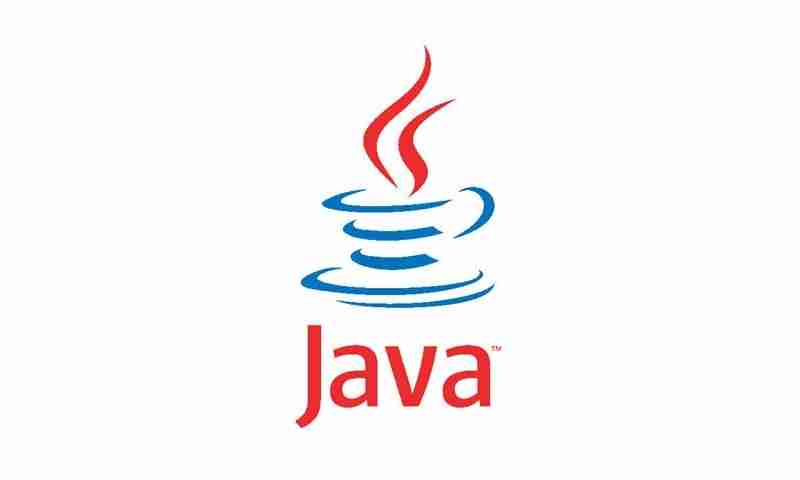
U drugom dijelu rada opisan je problem koji rješava Java desktop aplikacija razvijena u razvojnom okruženju Netbeans, relacijski model baze podataka, dijagram klasa te prozori aplikacije. Također su u posebnom odlomku drugoga dijela navedeni i objašnjeni paketi koji se nalaze unutar same aplikacije, odnosno paketi koji podrazumijevaju modele, poglede i upravitelje. U zasebnom odlomku su uz pomoć UML (Unified Modeling Langauge-grafički jezik koji služi za vizualizaciju i dokumentiranje objektno orijentiranih programskih rješenja) dijagrama i tekstualnih opisa objašnjeni slučajevi korištenja aplikacije. Dakako, kako je gotovo svaku aplikaciju moguće prošiti na određene načine, tako je isto slučaj i s aplikacijom koja je predmet istraživanja ovog diplomskog rada, stoga su u posebnom, ujedno i posljednjem odlomku rada, navedene i objašnjene mogućnosti proširenja aplikacije.

# PROGRAMSKI JEZIK JAVA

Programski jezik Java je jedan od najzastupljenijih programskih jezika, kako u poslovnom svijetu tako i u obrazovanju. Pokreće se na više milijardi uređaja te je od samih svojih početaka gotovo uvijek bio jedan od najpopularnijih programskih jezika na svijetu. Java predstavlja jedinstvenu računalnu platformu i programski jezik visoke razine koji se danas koriste za razvoj desktop aplikacija, mobilnih aplikacija, web aplikacija, igara i još mnogo toga.

Programski jezik Java je razvijen od strane James Gosling-a, Patrick Naughton-a i ostalih inženjera u kompaniji Sun Microsystems koji su 1991. godine počeli raditi na projektu novog programskog jezika (Stealth project). Cilj projekta je bio razviti programski jezik namijenjen za male uređaje (poput preklopnika za televizor) koji bi trošio malo memorije i radio na slabijim te različitim procesorima.[[1]](#footnote-1)

Prvobitno ime programskog jezika bilo je Oak, ali je kasnije otkriveno kako već postoji programski jezik pod istim nazivom. Zatim je nakon puno rasprava razvojnog tima i uz pokoju šalicu kave programski jezik preimenovan u Java pa tako i danas njegov logo podsjeća na šalicu kave.[[2]](#footnote-2) (Slika 1.)



**Slika 1. Java-in logo**

Prvi Java-in program se zvao \*7 i predstavljao je program za inteligentni daljinski upravljač za televizor. Nakon toga su se nizali različiti programi za usluge kabelske televizije. 1994. godine kompanija Sun razvija web preglednik nazvan HotJava- prvi program koji je imao mogućnost izvršavanja Java-inih programa unutar web preglednika (apleta)[[3]](#footnote-3). Projekt je konačno objavljen u studenom 1995. godine kada je izašla probna verzija programskog jezika (beta verzija). [[4]](#footnote-4)

Početkom 1996. godine izlazi prva službena verzija Java platforme i Java programskog jezika, a iste godine Java-u licenciraju važne tvrtke poput NetScape-a, IBM-a, Microsoft-a i drugih.

2010. godine kompanija Oracle kupuje kompaniju Sun Microsystems i od tada održava izdanja i podršku za Java-u.

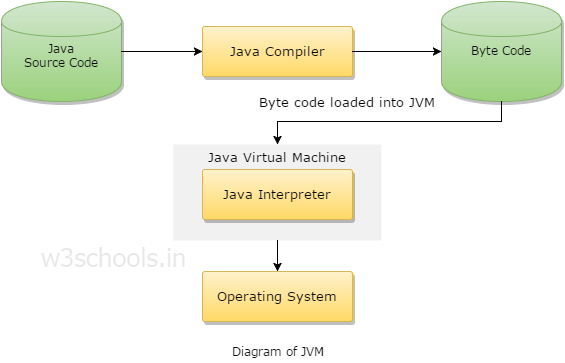
Zanimljiva je činjenica da se donedavno Java nije toliko koristila u razvoju Web poslužitelja. Međutim, s poboljšanjima u sučeljima za programiranje aplikacija (engl. API), a posebice Servlet API-ju, Java postaje gotovo neizostavan alat u Web programiranju.

Danas je, prema podacima Git Hub-a, Java treći programski jezik po zastupljenosti među programerskom populacijom i dosada je poboljšavan kroz svojih 17 verzija pa je tako najnovija verzija Java-e (Java 15) izašla studenom 2020. godine.

## Osnovne karakteristike Java jezika

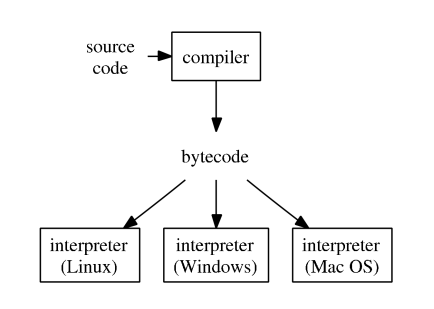
Kako bi se izvorni kod nekog programa (bilo koji niz izraza napisan u nekom od programskih jezika) mogao izvršiti na odgovarajućoj računalnoj platformi potrebno ga je najprije prevesti u jedini jezik kojeg računala mogu razumjeti- strojni programski jezik, odnosno jezik nula i jedinica. Takvo transformiranje izvornog koda se ostvaruje prevođenjem ili interpretiranjem, a tu zadaću obavlja prevoditelj odnosno interpreter.[[5]](#footnote-5) Ovisno o tome prevodi li se izvorni kod u strojni jezik uz pomoć prevoditelja ili interpretera programski jezici se mogu podijeliti u dvije skupine; prevedene (kompilirane) i intepretirane programske jezike. Java programski jezik je po ovoj podjeli izrazito specifičan iz razloga što se može svrstati u bilo koju od navedenih skupina, točnije za Java-u se može reći da je i kompilirani i interpretirani programski jezik. Razlog tome leži u činjenici da je programski jezik Java dizajniran tako da se programi napisani u njemu mogu jednostavno izvoditi kroz različite tipove platformi bez unošenja ikakvih promjena na tim programima. Za ovakvo izvođenje Java-inih programa zaslužan je Java virtualni stroj (engl. JVM- Java Virtual Machine), koji može biti instaliran na bilo kojem od postojećih operacijskih sustava.

Prilikom prevođenja izvornog koda u strojni kod, Java-in prevoditelj (javac) prevodi izvorni kod u među-format poznatiji kao bytecode. Bytecode nije izvršni kod, već visoko-optimizirani skup instrukcija dizajniran za izvršavanje unutar Java virtualnog stroja. Na taj način JVM predstavlja svojevrsnog interpretera za bytecode.[[6]](#footnote-6) (Slika 2.)



**Slika 2. Prevođenje Java izvornog koda uz pomoć javac-a i interpretiranje uz pomoć JVM-a**

Zahvaljujući ovakvom načinu prevođenja izvornog koda programa napisanog u Java-i u strojni jezik računala, programski jezik Java postiže svoje najznačajnije karakteristiku- prenosivost. Prenosivost proizlazi iz platformne neovisnosti, koja se odnosi na računalne platforme na kojim se izvorni kod može izvršiti (Slika 3.)[[7]](#footnote-7) i arhitekturne neutralnosti koja se odnosi na arhitekturu računala ili vrstu procesora na kojoj se izvode programi napisani u Java-i.



**Slika 3. Pokretanje koda na različitim operacijskim sustavima**

Pored prenosivosti sljedeća važna karakteristika Java programskog jezika jeste objektna orijentiranost. Osnovni koncepti Objektno orijentiranog programiranja su sljedeći:

1. Objekt
2. Klasa
3. Nasljeđivanje
4. Polimorfizam
5. Apstrakcija
6. Učahurivanje ili enkapsulacija

Naime, programski jezik Java se temelji na klasama i u njemu su obavezni objektno orijentirani principi. Sav izvorni kod u Java-i se piše unutar klasa, što znači da svaki program napisan u Java-i mora sadržavati barem jednu (glavnu) klasu.

U usporedbi s ostalim programskim jezicima, sintaksa kao i programska logika jezika Java-e su najsličnije onima u C i C++ programskom jeziku koji su postojali prije Java-e. Java je upravo od njih zadržala neke značajke koje su predstavljale prednosti prilikom pisanja koda, a uklonila je pri tom i neke nedostatke koji su oni imali pruživši pri tom i brojne dodatne funkcionalnosti. Konstrukcije i mehanizmi koji su prilikom kodiranja u C-u ili C++-u često dovodili do grešaka, a koji su u Java-i ukinuti ili zamijenjeni alternativnim rješenjima su: [[8]](#footnote-8)

* Predprocesorske naredbe (typdef i define)
* Strukture i unije
* Višestruko nasljeđivanje
* Goto naredba
* Preopterećenje operataora
* Pokazivači

Programski jezik Java karakterizira i ugrađena podrška nacionalnih znakova. Programi u Java-i jednako se lako pišu za korištenje u raznim dijelovima svijeta, gdje se ljudi ne služe standardnim latiničnim pismom ili engleskim jezikom. Programi se mogu napraviti na način da podržavaju nekoliko svjetskih jezika i da se automatski prilagođavaju okruženju u kojemu se kod izvršava.[[9]](#footnote-9)

## Prednosti i nedostaci programskog jezika Java

Osim prenosivosti kao i objektne orijentiranosti programskog jezika, Java ima još mnogo karakteristika koje se ujedno mogu svrstati u njegove prednosti ili razloge zbog kojih bi se baš on trebao upotrebljavati. Najvažnije od njih su:[[10]](#footnote-10)

* Jednostavnost
* Sigurnost
* Robusnost
* Podrška za paralelno programiranje
* Visoke performanse
* Dinamičnost
* Mnoštvo API-ja na raspolaganju
* Moćni razvojni alati otvorenog koda
* Biblioteke otvorenog koda
* Besplatnost
* Podrška zajednice

Java je jednostavan za korištenje iz razloga što nema kompliciranu sintaksu, a budući da je sintaksa dosta slična sintaksi u C-u i C++-u izrazito je jednostavan za naučiti ukoliko se dobro poznaju spomenuti jezici. Java-inoj jednostavnosti pridonosi činjenica da su u Java-i uklonjeni eksplicitni pokazivači i preopterećenje operatora.[[11]](#footnote-11)

Osim jednostavnosti, odsustvo pokazivača kao i nemogućnost preopterećenja operatora ujedno doprinose i Java-inoj sigurnosti. I pored njih postoji nekoliko razloga zbog kojih se Java smatra sigurnim programskim jezikom:[[12]](#footnote-12)

* učitavač klasa u Java-i omogućava dinamično učitavanje klasa te razdvaja pakete za klase lokalnog datotečnog sustava od onih koji se uvoze s mrežnih izvora
* verifikator bytecode-a provjerava fragmente koda štiteći objekte od neovlaštenog pristupa
* u Java-i upravitelj sigurnosti određuje kojim resursima klasa može pristupiti poput primjerice čitanje s lokalnog diska i pisanje na njega

Za Java-u se može se reći kako je on snažan (robustan) programski jezik zato što Java koristi snažno upravljanje memorijom eliminirajući svaku mogućnost da dvije varijable posjeduju istu memorijsku lokaciju, što sprječava mnoge pogreške prilikom programiranja i što također predstavlja prednost korištenja Java-e.[[13]](#footnote-13)

Također, u Java-i je moguće je pisati programe koji rade na mnogim zadacima odjednom tako što se definiraju višestruke niti. Niti su važne za multimediju, Web aplikacije itd.[[14]](#footnote-14) Glavna prednost višenitnosti (engl. multithreading-a) je dijeljene iste memorije. Ova karakteristika pruža programima bolje funkcioniranje u stvarnom vremenu.[[15]](#footnote-15)

Java virtualni stroj koristi Just in time (točno na vrijeme) način prevođenja što Java-i omogućava izvršavanje koda tijekom samog prevođenja te na taj način pruža Java-i visoke performanse.[[16]](#footnote-16)

Prednost korištenja Java programskog jezika jeste i mnoštvo API-ja na raspolaganju koji se mogu koristiti za različite aktivnosti poput: povezivanja s bazom podataka, umrežavanja, Ulaza/izlaza, raščlanjivanja u XML-u (Extensible markup langauge- proširivi jezik za označavanje podataka) itd.[[17]](#footnote-17)

Konačno, Java programski jezik je besplatan gotovo u svakom pogledu. Moguće je koristiti besplatna integrirana razvojna okruženja otvorenog koda te su dostupni brojni besplatni online tečajevi na kojima se može naučiti programirati u Java-i. Java ima na raspolaganju i veliki broj knjižnica otvorenog koda, a proširivanju tih knjižnica doprinijele su i neke uspješne i poznate kompanije poput Apache-a, Google-a, Facebook-a, Yahoo-a, LinkedIn-a itd.[[18]](#footnote-18)

Zbog svoje popularnosti Java je stekla široku podršku zajednice programera, što novim programerima olakšava svladati umijeće programiranja, a nakon toga pružaju im i se brojne mogućnosti za poslovne pothvate.

Uz prednosti Java programski jezik ima i pojedine nedostatke, koji su gotovo zanemarivi i nisu svojstveni samo za Java-u ali uvelike mogu pomoći konstruktivnijem korištenju Java-u. To su:[[19]](#footnote-19)

* Java prevoditelj (eng. compiler) još nije dobro optimiziran u usporedbi s C ++
* Ne postoji odvajanje specifikacija od implementacije
* Upravljanje memorijom u Java-i je skupo
* Nedostatak predložaka (eng. template) može ograničiti sposobnost Java-e u stvaranju visokokvalitetnih struktura podataka

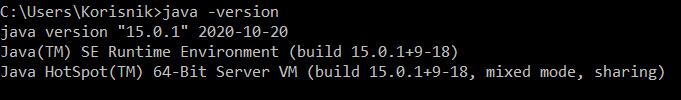
## Naredbe u Java-i

Program je niz naredbi koje računalo izvršava kako bi obavilo neku zadaću. Da bi računalo moglo izvršavati naredbe, one moraju biti pisane na računalu razumljiv način, tj. u programskom jeziku. Programski jezici se razlikuju od ljudskog po svojoj jasnoći i strogosti što je u programu dozvoljeno, a što nije. Pravila koja određuju što je dozvoljeno zovu se sintaksa jezika. Sintaksna pravila određuju osnovni rječnik jezika i način na koji se programi mogu stvarati koristeći petlje, grananja i potprograme. Sintaksno ispravan program je onaj koji je moguće kompajlirati ili izvršiti. Programi koji sadržavaju sintaksne greške će biti odbačeni uz poruku o grešci.[[20]](#footnote-20)

U programskom jeziku Java-i, naredba (ključna riječ) je bilo koja od 52 ključnih riječi koje su rezervirane i imaju unaprijed definirano značenje u Java-i[[21]](#footnote-21).To znači da niti jedna od ovih 52 rezerviranih ključnih riječi ne može biti upotrijebljena kao naziv varijable, metode ili klase niti može biti upotrjebljena kao identifikator.[[22]](#footnote-22)

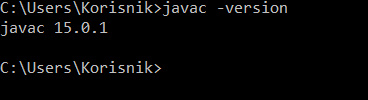
U ovom poglavlju su najprije objašnjene neke od osnovnih naredba u Java-i koje je moguće unijeti preko terminala. Nakon tih naredbi su u nastavku objašnjene naredbe i ključne riječi koje se koriste prilikom kodiranja u integriranim razvojnim okruženjima:

**Java –version** (verzija): Ovo je jedna od najosnovnijih Java naredbi koja se koristi za provjeru Java verzije instalirane na korisnikovom računalu. Ova naredba se također koristi kako bi se osigurala ispravnost instalacije i postavki PATH varijabli.



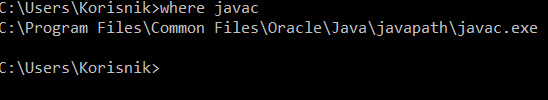
**Slika 4. Primjer unošenja naredbe Java –version u terminal**

**Javac -version:** Ova naredba se koristi za prikaz verzije prevoditelja koji je odgovoran za sastavljanje izvornog koda. Ovo je također dio Java razvojnog kompleta popularno zvanog JDK (Java Development Kit-Java razvojni komplet).



**Slika 5. Primjer korištenja naredbe Javac –version u CMD-u**

**Where:** Ova naredba se koristi za pronalaženje određene komponente unutar direktorija. U nastavku je uzet u obzir java prevoditelj.



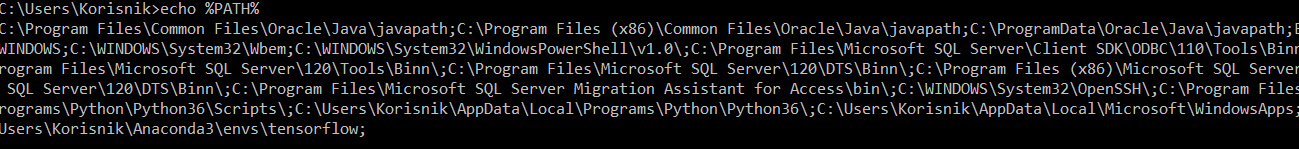
**Slika 6. Upotreba naredbe Where u CMD-u**

Za izvršavanje programa nakon što je pretvoren u bajt kod, koristimo ključnu riječ Javai potom naziv programa. U primjeru u nastavku, korišten je program s nazivom Hello World.



**Slika 7. Izvršavanje programa**

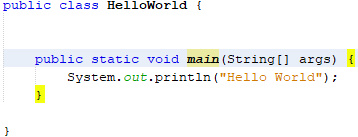
**Echo:** Ova naredba se ne koristi samo u Java-i, već je naredba koju se mora poznavati jer pomaže u prikazivanju podataka određene datoteke. Obično se koristi za provjeru PATH varijabli.



**Slika 8. Naredba Echo**

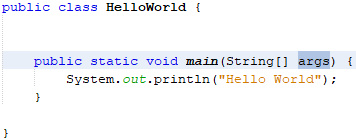
Nakon što su navedene osnovne naredbe koje je moguće unijeti preko terminala, slijede objašnjenja i prikazi ostalih naredbi, odnosno ključnih riječi koje se koriste u integriranim razvojnim okruženjima za izradu kodova:

**Main:** Kao što joj ime sugerira, Main je glavna funkcija programa u kojemu prevoditelj prvo dosegne i izvrši njezin dio programskog koda.



**Slika 9. Glavna funkcija programa- main**

**Args []:** koristi se za specificiranje argumenata sadržanih u glavnoj funkciji, a uglate zagrade podrazumijevaju niz:



**Slika 10. Specificiranje argumenata**

Jedini način na koji se u Java-i može spremiti podatak je korištenjem naredbe pridruživanja vrijednosti[[23]](#footnote-23). Ovisno o tipu podatka kojeg je potrebno pridružiti varijabli prilikom deklariranja varijable se koriste sljedeće naredbe:

**Byte, short, int** i **long**- Naredbe za deklariranje cjelobrojne varijable. Na primjer:

int brojac=0; Varijabli brojac pridružujemo vrijednost 0.

**Float** i **double** su naredbe za deklariranje varijable kojoj se pridružuje vrijednost s pomičnim zarezom. Međusobno se razlikuju po rasponu i preciznosti. Na primjer:

float prosjek\_ocjena; Nakon što je deklarirana, varijabli prosjek\_ocjena se može pridružiti vrijednost iz skupa realnih brojeva

Naredba **char** služi za deklariranje varijable kojoj pridružujemo vrijednost od jednog znaka iz Unicode skupa:

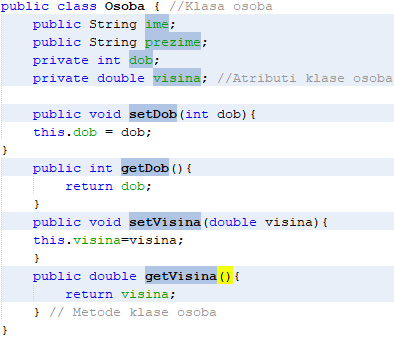
char znak=Y;

Ključna riječ **boolean** se koristi za deklariranje logičke varijable. Shodno tome, varijabli se može pridružiti jedna od dvije moguće vrijednosti tj. true ili false. Na primjer:

boolean prolazak= true;

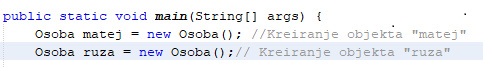
Važno je napomenuti kako svi navedeni tipovi podataka spadaju u skupinu tzv. primitivnih (jednostavnih) tipova podataka. Pored jednostavnih u Java-i postoje i složeni tipovi podataka, a to su polja, klase i sučelja. Za deklariranje polja u Java-i se koristi ključna riječ **Array**, za deklariranje klasa **Class** a za deklariranje sučelja **Interface**.

**Class:** Klasa je opći predstavnik nekog skupa objekata koji imaju istu strukturu i ponašanje.[[24]](#footnote-24) Za klasu se često kaže kako je ona logična cjelina koja predstavlja nacrt objekta. Upisujući ključnu riječ **Class** definiramo klasu. Uobičajeno, klasa sadrži atribute i metode kao na sljedećoj slici:



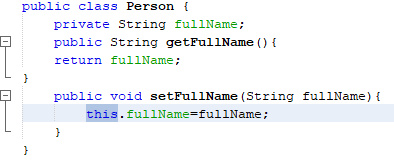
**Slika 11. Atributi i metode klase Osoba**

**New (Objekt):** Objekt je instanca klase. Sve u Java-i je objekt jer je to jedini fizički entitet od kojeg se sve sastoji. Predstavlja kombinaciju postupaka i podataka koji rade na već dostupnim podacima.



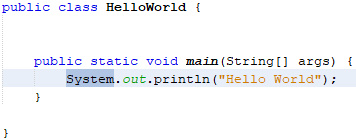
**Slika 12. Primjer kreiranja objekata u Java-i**

**This:** Ova ključna riječ u Java-i se koristi za upućivanje na trenutno stanje objekta ili na varijabilnu instancu trenutne klase ili konstruktor trenutne klase.



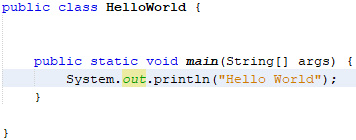
**Slika 13. Ključna riječ this**

**System:** Ovo je unaprijed izgrađena klasa u Java-i koja je prisutna u zadanom paketu lang koji se koristi za rukovanje standardnim ulazom, standardnim izlazima i drugim greškama u stream-u (strujanju).



**Slika 14. Klasa System u Java-i**

**Out:** Out je objekt i statički član klase System koji se koristi za ispis poruke na izlaznoj stazi koja je obično konzola ili datoteka.



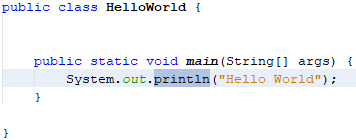
**Slika 15. Out, statički član klase System**

**Print:** Ova metoda se koristi za ispis sadržaja programa na slijedan način i kako kursor ne bi došao do sljedećeg retka.



**Slika 16. Metoda print**

**Println:** Ova se funkcija koristi za ispis sadržaja u novom retku.



**Slika 17. Metoda println**

Ovo su neke od Java naredbi tj. ključnih riječi koje Java programeri često koriste. Postoji nekoliko drugih naredbi koje se nazivaju upravljačke naredbe i koriste se za kontroliranje toka programa i njegovog grananja na osnovu promjena stanja programa. Java-ine upravljačke naredbe se mogu svrstati u sljedeće kategorije:

* 1. naredbe uvjetovanja (if i switch)
  2. naredbe ciklusa (for, while, do-while)
  3. naredbe skoka (break, continue, return).

Java podržava dvije naredbe uvjetovanja i to **if** i **switch**. Naredbom if se program može usmjeriti duž jednog od dva različita puta. Opći oblik naredbe if: [[25]](#footnote-25)

* If (uvjet) naredba 1;
* Else naredba 2;

Ako postoji više uvjeta može se koristiti if-else-if naredba .Međutim kako bi se izbjegla ljestvica if-else-if naredbi koristi se naredba switch.

Java-ine naredbe ciklusa obuhvaćaju: for, while i do-while. Ovim naredbama se prave petlje koje izvršavaju isti skup instrukcija onoliko puta koliko je potrebno kako bi se ispunio zadani uvjet. Petlja **while** je Java-ina osnovna petlja. Ona ponovljeno izvršava naredbu ili blok sve dok je upravljački izraz točan.[[26]](#footnote-26)

Njen opći oblik je:

while (uvjet) {

tijelo petlje

}

Na primjer:

int brojac=0;

while(brojac<10) {

f=x-brojac;

Brojac ++;

}

Petlja **do-while** uvijek izvršava tijelo petlje barem jednom, jer se uvjetni izraz nalazi na njenom kraju.[[27]](#footnote-27)

Njen opći oblik je:

do {

// tijelo

} while (uvjet);

Na primjer:

int k=0;

do {

System.*out*.println(”k= ”+k);

k++;

} while (k<=10);

Petlja **for** radi na sljedeći način: kada petlja započne rad, izvršava se inicijalizacija koja djeluje kao brojač ciklusa petlje. Zatim se ispituje uvjet koji uspoređuje upravljačku varijablu petlje sa zadanom ciljnom vrijednošću. Ako je izraz točan, izvršava se tijelo petlje. Ako je netočan, petlja se završava. Zatim se izvršava iteracija.[[28]](#footnote-28) Na primjer:

for (int n=10;n>0;n–)

Kada se varijabla deklarira unutar petlje for, tada njena važnost prestaje sa završetkom petlje for.

Java podržava tri naredbe za skokove: **break**, **continue** i **return**. Ove naredbe omogućavaju da se izvršavanje programa ne nastavi sa sljedećom po redu, već s nekom drugom naredbom.

Naredba **break** u Java-i ima tri namjene. Prva, za završavanje niza naredbi u naredbi switch. Drugo, može poslužiti za izlazak iz petlje i treće, kao „civiliziran” oblik naredbe goto. Njen opći oblik je: break oznaka.[[29]](#footnote-29)

Oznakom se označava blok naredbi. Kada se izvrši ovaj oblik naredbe break, izvršavanje programa se nastavlja poslije označenog bloka. Rijetke su situacije u kojima je opravdana primjena naredbe **continue**, jer Java posjeduje bogat skup naredbi za petlje koje mogu riješiti većinu potreba. Posljednja naredba je **return**, koja se koristi za bezuvjetan izlazak iz metode.[[30]](#footnote-30)

## Java platforma

Definicija računalne (digitalne) platforme glasi: Platforma je software-sko ili hardware-sko okruženje u kojemu se računalni program pokreće. Iz navedene definicije lako se može zaključiti kako postoje dva osnovna tipa računalnih (digitalnih) platformi:

1. Digitalne platforme zasnovane na software-u
2. Digitalne platforme zasnovane na hardware-u

Pod pojmom Java ne podrazumijeva se isključivo programski jezik Java te njegova biblioteka klasa. Pojam Java podrazumijeva i cjelokupnu software-sku platformu čiji je temelj Java virtualni stroj.[[31]](#footnote-31) Java platforma podrazumijeva skupinu programa koji omogućuju izvođenje aplikacija napisanih u Java-i. Razlikuje se od većine ostalih računalnih platformi po tome što je ona software-ski zasnovana platforma koja radi na ostalim (hardweare-ski zasnovanim) platformama. Ovisno o namjeni upotrebe, Java platforma je dostupna u 3 izdanja:

1. Java standardno izdanje (Java Standard Edition - Java SE)- izdanje je namijenjeno osobnim računalima
2. Java mikro izdanje (Java Micro Edition – Java ME)- namijenjeno je mobilnim uređajima
3. Java poslovno izdanje (Java Enterprise Edition- Java EE) namijenjeno je poslužiteljima za mrežne i poslovne aplikacije

Osnovne komponente koje zajedno čine Java platformu jesu:[[32]](#footnote-32)

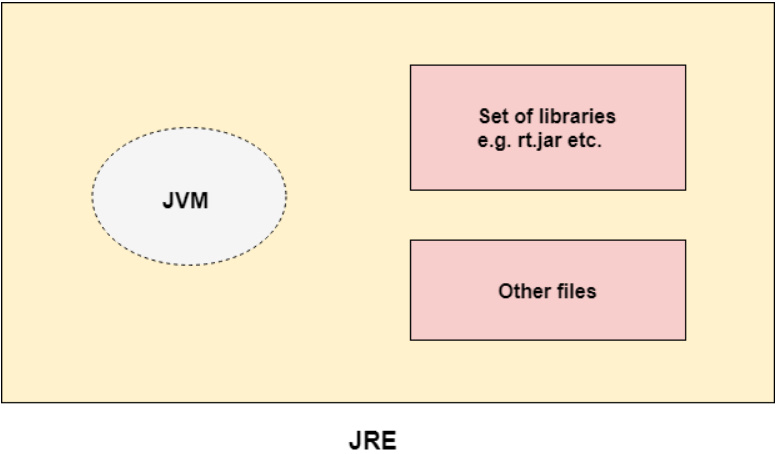
1. Java programski jezik
2. Java razvojni komplet (JDK – Java Development Kit)
3. Java okruženje za izvršavanje (JRE- Java Runtime Enviorment)
4. Java prevoditelj
5. Java virtualni stroj (JVM- Java Virtual Machine)

Kako bi se stekla jasnija slika o cjelokupnoj Java platformi od navedenih komponenti je važno razlikovati komponente JVM, JRE i JDK.

Naime, u središtu platforme se nalazi **JVM**. Naziva se virtualni zato što ne postoji u fizičkom smislu. Već je opisano kako se programi napisani u Java-i se ne prevode (kao što je slučaj kod većine drugih programskih jezika) u izvršni kod koji se izravno izvršava na procesoru računala, nego se oni prevode u posebnu vrstu izvršnog koda za JVM zvanu java bytecode ili međukod, kojeg onda konkretno računalo emulira. Rezultat toga je postizanje neovisnosti o vrsti procesora ili računalnoj platformi; kod se prevodi jednom a izvršava se na bilo kojem uređaju koji ima implementaciju JVM-a, bez obzira na vrstu procesora ili operacijskog sustava.[[33]](#footnote-33) Također, JVM može izvršiti i kod koji je napisan u nekom drugom programskom jeziku, pod uvjetom da je taj kod preveden u byte-code. Glavni zadaci JVM-a jesu:[[34]](#footnote-34)

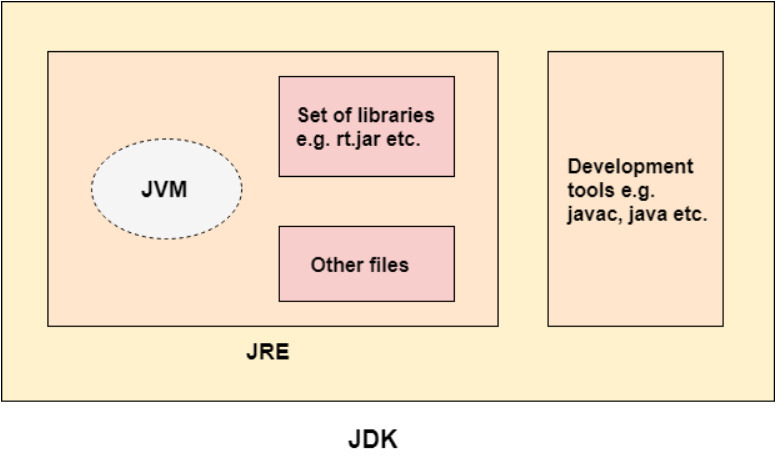
1. Učitavanje koda
2. Verifikacija koda
3. Izvršavanje koda
4. Pružanje okruženja za izvođenje

Nadalje, neizostavan dio Java platforme jeste Java-ino okruženje za izvršavanje odnosno **JRE** (također se rabi skraćenica RTE- RunTime Enviorment). Predstavlja okruženje u kojemu se pokreću Java aplikacije i apleti i ne sadrži nikakve razvojne alate poput primjerice prevoditelja i sl. Fizički postoji te predstavlja implementaciju JVM-a- Sastoji od JVM-a, biblioteka potrebnih za uspješno izvođenje programa i aplikacija te ostalih datoteka koje JVM koristi prilikom izvršavanja koda (Slika 16.).[[35]](#footnote-35)



**Slika 18. Java Runtime Enviorment (JRE)**

**JDK** predstavlja programsko razvojno okruženje koje se koristi za razvoj Java aplikacija i apleta. Baš kao i JRE, JDK fizički postoji u memoriji računala. JDK sadrži JRE (te sve njegove komponente) zajedno sa svim ostalim alatima potrebnih za razvoj Java aplikacija poput primjerice Java programskog prevoditelja- compiler-a. (Slika 17.) Predstavlja implementaciju bilo kojeg izdanja Java platforme (Java SE, Java ME ili Java EE).



**Slika 19. Java Development Kit (JDK)**

## Integrirano razvojno okruženje NetBeans

Za razvoj Java aplikacija je dovoljan JDK i obični tekstualni uređivač poput primjerice Windows-ovog bloka za pisanje. Naravno, takav razvoj aplikacija bi bio izuzetno spor, naporan i iscrpljujući. Zbog toga su se ljudi (još prije pojave Java jezika i platforme) dosjetili napraviti posebne aplikacije koje služe za razvoj drugih aplikacija. Takve aplikacije se nazivaju integrirana razvojna okruženja (skraćeno IDE- Integrated Development Enviorment). Jedno takvo razvojno okruženje je i NetBeans IDE.

Najvažnije prednosti korištenja integriranih razvojnih okruženja prilikom izrade aplikacija su sljedeće:[[36]](#footnote-36)

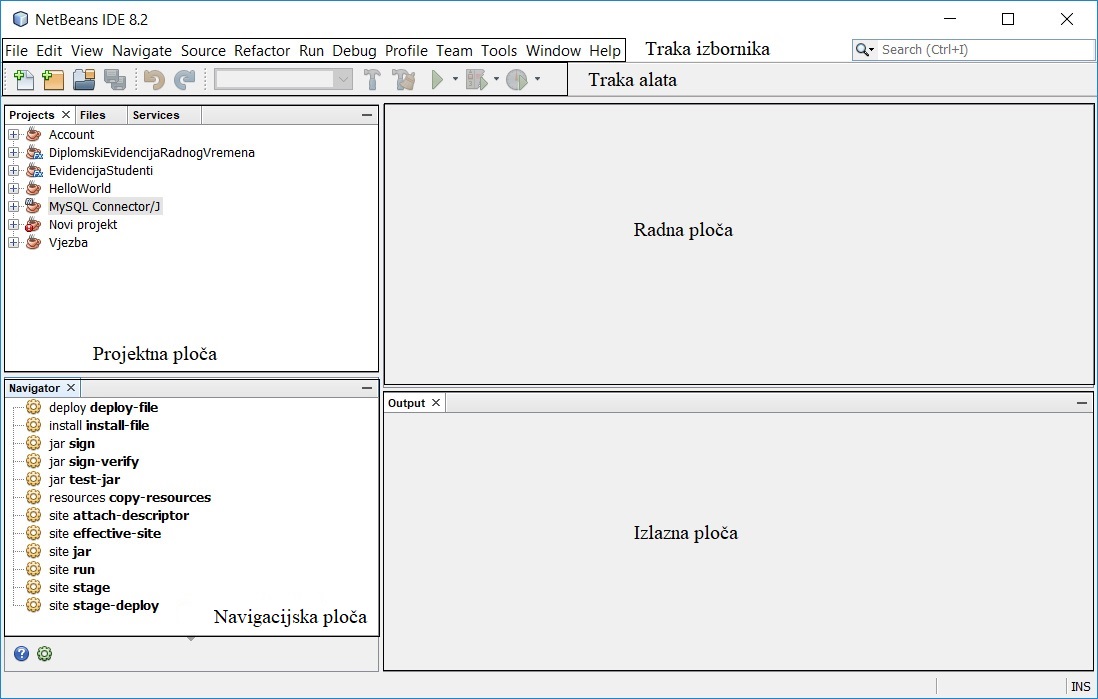
* Uređivanje koda, završavanje koda skraćenicama i prečacima, sintaksno naglašavanje, pomoć (hints)
* Pretraživanje i refaktoriranje koda
* Prevođenje i izvršavanje koda
* Čarobnjaci (Wizards) koji olakšavaju komplicirane zadatke
* Kreiranje izvršnih i instalacijskih paketa

NetBeans IDE je integrirano razvojno okruženje, namijenjeno prvenstveno razvoju Java tehnologija.[[37]](#footnote-37) Besplatan je, jednostavan za korištenje i učenje i sadrži sve potrebne elemente za razvoj aplikacija raznih opsega. Aplikacije utemeljene na NetBeans-u se mogu proširiti u skladu s potrebama, a impozantna je činjenica da se i sam NetBeans može proširiti s određenim dodacima. Osim programiranja u Java-i, u NetBeans-u je uz odgovarajuća proširenja moguće programirati i u ostalim programskim jezicima, kao što su C, C++, C#, HTML5, JavaScript, PHP te brojni drugi programski jezici.

NetBeans korijene vuče iz 1996. kada je započeo kao projekt grupe studenata Fakulteta za matematiku i fiziku na Sveučilištu u Pragu (tadašnja Čehoslovačka, današnja Češka) pod nazivom Xelfi. Cilj projekta je bio stvoriti integrirano razvojno okruženje za Java-u, što je grupi studenata i uspjelo 1997. godine, a uspjeh kao takav predstavljao je prvi IDE za Java-u ikada napravljen. Studenti su, nakon što su diplomirali, oko svog projekta osnovali tvrtku, a idealnu priliku za ulaganje u isplativi projekt pronašao je poduzetnik s iskustvom Roman Stanek te je tvorcima IDE-a pomogao da se proguraju na tržište. Jaroslav Tulach, tvorac arhitektura ovog IDE-a je tada predložio naziv NetBeans. 1999. godine tvrtka Sun kupuje NetBeans, ali i kreće sa sponzoriranjem NetBeans-a kao projekta otvorenog koda (Open source) i preuzima upravljanje nad infrastrukturom. Do lipnja 2000. izlazi i prva službena Internet stranica za ovaj NetBeans IDE. 2010. godine kupivši tvrtku Sun, tvrtka Oracle stekla je i prava na NetBeans IDE te ga i danas nastavlja sponzorirati kao projekt otvorenog koda i nastavlja izdavati podršku za njega.[[38]](#footnote-38) Tako je najnovije izdanje NetBeans IDE-a (NetBeans 8.2) izašlo u listopadu 2016. godine.

Glavni dijelovi NetBeans razvojnog okruženja su: (Slika 20.)

1. Traka izbornika
2. Traka alata
3. Projektna ploča
4. Navigacijska ploča
5. Radna ploča
6. Izlazna ploča



**Slika 20. Dijelovi glavnog prozora NetBeans IDE-a**

NetBeans-ov uređivač uvlači retke, podudara se s riječima i zagradama te naglašava izvorni kod sintaktički i semantički. Omogućuje jednostavno refaktoriranje koda s nizom zgodnih i moćnih alata, a također nudi i predloške koda, savjete za kodiranje i generatore koda.

Čuvanje jasnog pregleda velikih aplikacija, s tisućama mapa i datoteka, te milijunima redaka koda, zastrašujući je zadatak. NetBeansIDE pruža različite prikaze korisnikovih podataka, od više prozora projekata do korisnih alata za postavljanje aplikacija i njihovog učinkovitog upravljanja, omogućujući brzo i lako unošenje vlastitih podataka.

## Ostala razvojna okruženja za Java-u

Budući da je Java jedan od vodećih programskih jezika i budući da se on u velikoj mjeri koristi za razvoj snažnih i sigurnih aplikacija (aplikacije za radne površine, poslovne aplikacije, mobilne i web aplikacije) mora imati na raspolaganju mnoštvo razvojnih okruženja koji mu to omogućavaju. Iako su Eclipse, IntelliJ IDEA i NetBeans tzv. velika trojka, postoje i brojni drugi IDE-ovi za razvoj aplikacija koje učinkovito koriste Java-u.[[39]](#footnote-39) Pored već opisanog NetBeansau ovom radu će biti navedeni i opisani još neki od njih:

**IntelliJIDEA** je jedan od tri velika Java IDE-e, koji je izašao 2001. godine i koji radi na svim Linux, MacOS i Windows platformama. Dostupan je u dva različita izdanja, licenciranom izdanju zajednice Apache 2 i vlastitim komercijalnim izdanjem: [[40]](#footnote-40)

* Community Edition: Dostupno je besplatno i uglavnom ga programeri koriste za Java-u i Android.
* Ultimate Edition: Ultimate edition ima najnapredniji skup funkcija za razvoj web i desktop aplikacija. Podržava integraciju tzv. opružnog okvira, okvira za web razvoj poput Node.js-a, Angular-a i React-a te Java EE podršku poput JSF-a (Jakarta Server Face), JAX-RS-a (Jakarta RESTful Web Services), JPA-a (Java Persistence API), CDI-a (Contexts and Depedency Injection), itd.

Uključuje funkcije poput dovršavanja koda, dubinske statičke analize, inteligentnog refaktoringa, ispravljača pogrešaka, testnog pokretača itd.[[41]](#footnote-41) Kako bi se programerima omogućilo da dublje uđu u Java kod, IntelliJ IDEA može se pohvaliti refaktoringom na više jezika i značajkama analize protoka podataka. Ostale značajke koje nudi IntelliJ IDEA i koje olakšavaju posao Java programeru su dovršavanje lanca, ubrizgavanje jezika, pametno dovršavanje koda i statičko dovršavanje članova. Osim što podržava Java-u i mnoštvo Java okvira, IntelliJ IDEA pruža i podršku za ostale programske jezike temeljene na JVM-u kao što su Clojure, Groovy, Kotlin, Scala itd. [[42]](#footnote-42)

**Eclipse** je također jedan od tri velika Java IDE-a. Moderno integrirano razvojno okruženje dostupno je u desktop i cloud (oblak) izdanju. Oblak izdanje Eclipse-a, nazvano Eclipse Che, omogućava programerima razvijanje aplikacija putem web-preglednika. Eclipse se pojavio 2001. prve godine i također se može koristiti na Linux, MacOS, Windows i Solaris platformi. Oba izdanja Eclipse IDE-a su opremljena potrebnom / dodatnom funkcionalnošću pomoću dodataka te Eclipse ima mnoštvo dodataka na raspolaganju. Da bi se olakšalo inkrementalno prevođenje Java koda, Eclipse dolazi s prilagođenim prevoditeljem.[[43]](#footnote-43)

Za Java programere koji žele razviti specifičnu funkcionalnost za Eclipse, dostupan je PDE (Plugin Development Environment). Kako bi Java-ini programeri ubrzali razvoj aplikacija, Eclipse ima moćne alate za crtanje, modeliranje, izvještavanje i testiranje.

Eclipse podržava razvoj aplikacija u više programskih jezika putem svojih dodataka; Ada, C, C ++, Clojure, Groovy, Haskell, JavaScript, Julia, Perl, PHP, Ruby, Rust i Scala su neki od različitih programskih jezika koje podržava Eclipse.[[44]](#footnote-44)

**BlueJ** je također integrirano razvojno okruženje za Java-u. Iako je uglavnom dizajniran s obrazovnim namjerama, BlueJ je prikladan za razvoj software-a malih razmjera. Usprkos činjenici da je ovaj IDE razvijen gotovo za same početnike, iskusni Java programeri ga uvelike preferiraju. Prvi put se pojavio još 1999. godine.

Njegove glavne karakteristike su: [[45]](#footnote-45)

* Jednostavnost- BlueJ namjerno ima manje i jednostavnije sučelje od profesionalnih okruženja poput NetBeans-a ili Eclipse-a.
* Dizajniran je za podučavanje - Postoji popularni udžbenik dizajniran za podučavanje uvodnih, sveučilišnih i fakultetskih tečajeva s BlueJ-om te web stranica prepuna nastavnih sredstava.
* Mogućnost interakcije s objektima- BlueJ omogućava interakciju s objektima. Mogu se pregledati njihove vrijednosti, mogu se pozivati ​​metode na njih, mogu se proslijediti kao parametri i još mnogo toga. Java izrazi se mogu direktno pozivati ​​bez prevođenja. Tako BlueJ predstavlja snažnu i pouzdanu grafičku ljusku za Java-u.
* Prenosivost- BlueJ radi na Windows, Mac OS X, Linux i drugim platformama na kojima radi Java. Može se izvoditi i bez instalacije, izravno s USB sticka.
* Zrelost- BlueJ je stariji od petnaest godina, ali ga nastavlja ažurirati i podržavati tim koji je stalno zaposlen. Želi odgovoriti na sve zahtjeve za tehničku podršku u roku od jednog radnog dana.
* Inovativnost- BlueJ ima nekoliko funkcija koja ranije nisu viđena u drugim IDE-ovima. Objektna klupa, šifra jastuka i obojenje opsega bile su sve originalne značajke upravo BlueJ-a.

Primarni ekran BlueJ-a prikazuje klasnu strukturu aplikacije koja se u tom trenutku razvija. Omogućuje i interaktivno stvaranje i testiranje objekata. Ostale moćne značajke razvoja Java programa koje nudi BlueJ uključuju stvaranje dinamičkih objekata i vizualno skeniranje koda.

**JDeveloper** je besplatni Java IDE tvrtke Oracle. Nudi integrirani razvojni okvir s mnoštvom značajki i nekoliko alata za vizualni razvoj. JDeveloper pokriva cjelokupni ciklus razvoja: kodiranje, dizajniranje, uklanjanje pogrešaka, optimizaciju, profiliranje i implementaciju.[[46]](#footnote-46)

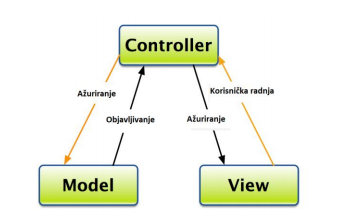
Oracle JDeveloper se može integrirati s Oracle-ovim okvirom za razvoj aplikacija (Oracle ADF) kako bi se dodatno pojednostavio razvoj aplikacija. Pored Java-e, JDeveloper se može koristiti i za razvoj aplikacija u GTML-u, JavaScript-u, PHP-u, SQL-u i XML-u.[[47]](#footnote-47)

## MVC Arhitektura

Model-prikaz-upravitelj (obično poznat kao MVC) je software-ski obrazac dizajna (engl. design pattern) koji se obično koristi za razvoj korisničkih sučelja i kao takav dijeli srodnu programsku logiku na tri međusobno povezana elementa (modele, prikaze i upravitelje). Ovakva podjela se vrši kako bi se odvojile interne reprezentacije informacija od načina na koji se informacije prezentiraju i prihvaćaju od korisnika. Slijedom MVC arhitektonskog uzorka odvajaju se ove glavne komponente koje omogućuju ponovnu upotrebu koda i paralelni razvoj.

Tradicionalno je korišten za grafička korisnička sučelja (Graphical User Interface) stolnih računala, a danas je ovaj obrazac postao popularan i za dizajn web aplikacija. Popularni programski jezici kao što su JavaScript, Python, Ruby, PHP, Java, C # i Swift imaju MVC okvire koji se koriste za razvoj web ili mobilnih aplikacija izravno iz okvira.

MVC aplikacija komunicira s korisnikom putem pogleda (eng. View), gdje korisnik unosom zahtjeva šalje naredbe upravitelju (eng. Controller), dok upravitelj od modela traži da obavi radnju i vrati mu rezultat. Rezultat radnje zatim upravitelj šalje u Pogled gdje ga korisnik može vidjeti (Slika 29).[[48]](#footnote-48)



**Slika 21. MVC okvir**

Komponenta Model odgovara svim logikama vezanim za podatke s kojima korisnik radi. To može predstavljati ili podatke koji se prenose između komponenata Prikaz i Upravitelja ili bilo koje druge podatke koji se odnose na poslovnu logiku. Na primjer, korisnikov model dohvatit će podatke o klijentu iz baze podataka, manipulirati i ažurirati podatke u bazu podataka ili ih koristiti za prikazivanje podataka.

Komponenta Prikaz koristi se za svu ulazno-izlaznu logiku aplikacije. Na primjer, prikaz kupca uključivat će sve komponente korisničkog sučelja poput tekstualnih okvira, padajućih postavki itd., s kojima krajnji korisnik komunicira.

Upravitelji djeluju kao sučelje između komponenti Model i Prikaz za obradu sve poslovne logike i dolaznih zahtjeva, manipuliraju podacima koristeći komponentu Model i komuniciraju s vlasničkim prikazima kako bi se dobio konačni izlaz. Na primjer, upravitelj kupca će upravljati svim interakcijama i ulazima iz pregleda klijenta i ažurirati bazu podataka koristeći model kupca. Isti upravitelj bit će korišten za pregled podataka kupca.

Neke od karakteristika (ujedno i prednosti) MVC aplikacija su: [[49]](#footnote-49)

* Olakšano upravljanje razvojem složenih aplikacija zbog mogućnosti posebnog razvoja svake od komponenti aplikacije, čime se olakšava organizacija posla razvoja aplikacije, a također i nadogradnja novim verzijama (ponekad je češće potrebno unaprijediti poglede i upravitelje dok Model ostaje isti)
* Olakšano testiranje aplikacije (tzv. Test-Driven Development - TDD), jer je moguće odvojeno testirati pojedine komponente aplikacije i nije uvijek za testiranje potrebno koristiti web server (neke komponente se mogu testirati lokalno)
* Olakšano procesiranje zahtjeva koje šalje korisnik, jer se za tu svrhu koristi tzv. Front Controller (prednji upravitelj) koji kroz jedan upravitelj šalje sve zahtjeve korisnika.

# JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA

Cilj ovog rada je, kao što je navedeno već u uvodnom dijelu, implementacija desktop aplikacije za evidenciju radnog vremena služeći se Java programskim jezikom, integriranim razvojnim okruženjem Netbeans i JavaFX Scene Builder-om 2.0 za kreiranje dizajna aplikacije. Java FX-scene Builder je korišten zato što pruža pregršt mogućnosti za dizajniranje aplikacija bogatih izgledom, a također se moguće služiti i opće poznatim stilom za oblikovanje izgleda- CSS-om (Cascading Style Sheet – Kaskadni stil oblikovanja).

Zbog svoje kompatibilnosti i jednostavnosti prilikom preuzimanja, instalacije i upotrebe odabrana je MySQL baza podataka (ali i zbog toga što je besplatna). Sustav korišten za implementaciju aplikacije je Windows 10, ali aplikaciju je moguće pokrenuti i na bilo kojem drugom uređaju koji ima instaliranu Java podršku.

## Opis problema

Kako su u prethodnom poglavlju navedeni svi alati koji su upotrijebljeni pri izradi i implementaciji Java-ine desktop aplikacije koja služi za evidentiranje radnog vremena, u ovom poglavlju je pobliže opisan problem kojega aplikacija rješava.

Naime, problem je vrlo razumljiv već na sam spomen naziva aplikacije koja je predmet istraživanja rada- evidentiranje radnog vremena zaposlenika u nekoj zamišljenoj kompaniji. S ovim problemom su se brojne tvrtke susretale duži niz godina, naročito one koje imaju veliki broj zaposlenika. U takvim tvrtkama koje broje i po nekoliko stotina zaposlenika u svojim brojnim prostorijama gotovo je nemoguće, ali zasigurno i neučinkovito na sam očigled provjeriti brojno stanje svojih zaposlenih. Potpisivanje olovkom na papir bi u ovakvom slučaju također moglo biti neučinkovito te bi isto zaposlenicima i njihovim nadređenima oduzimalo dragocjeno vrijeme. Danas, gotovo da nema tvrtke ili institucije s većim brojem zaposlenika ili korisnika koja nije na sličan način riješila ovaj problem. Jedna od takvih institucija je i Sabor Republike Hrvatske gdje je implementiran sustav za provjeru dolaska zaposlenih.

Aplikacija koja je predmet istraživanja ovog diplomskog rada rješava navedene i opisane probleme evidencije radnih vremena zaposlenika u nekoj zamišljenoj tvrtci, instituciji ili udruženju. Moguće je uz gotovo neznatno korištenje resursa osobnih računala zaposlenika na kojima (i ako) rade određene zadatke, ujedno i evidentirati njihovo vrijeme provedeno na poslu.

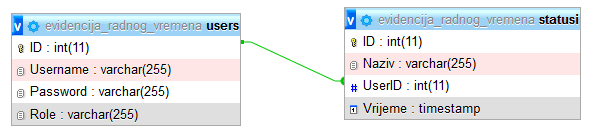
Aplikacija je osmišljena na način da razlikuje 2 vrste korisnika:

1. **Korisnik**
2. **Administrator**

Korisnik ima mogućnost prijave na sustav, unosa radnog statusa, te pregled svih svojih radnih statusa po vremenu.

Administrator ima mogućnost pregleda, unosa, izmjene i brisanja korisnika, te osim toga ima pregled svih radnih statusa od svih korisnika.

## Relacijski model baze podataka

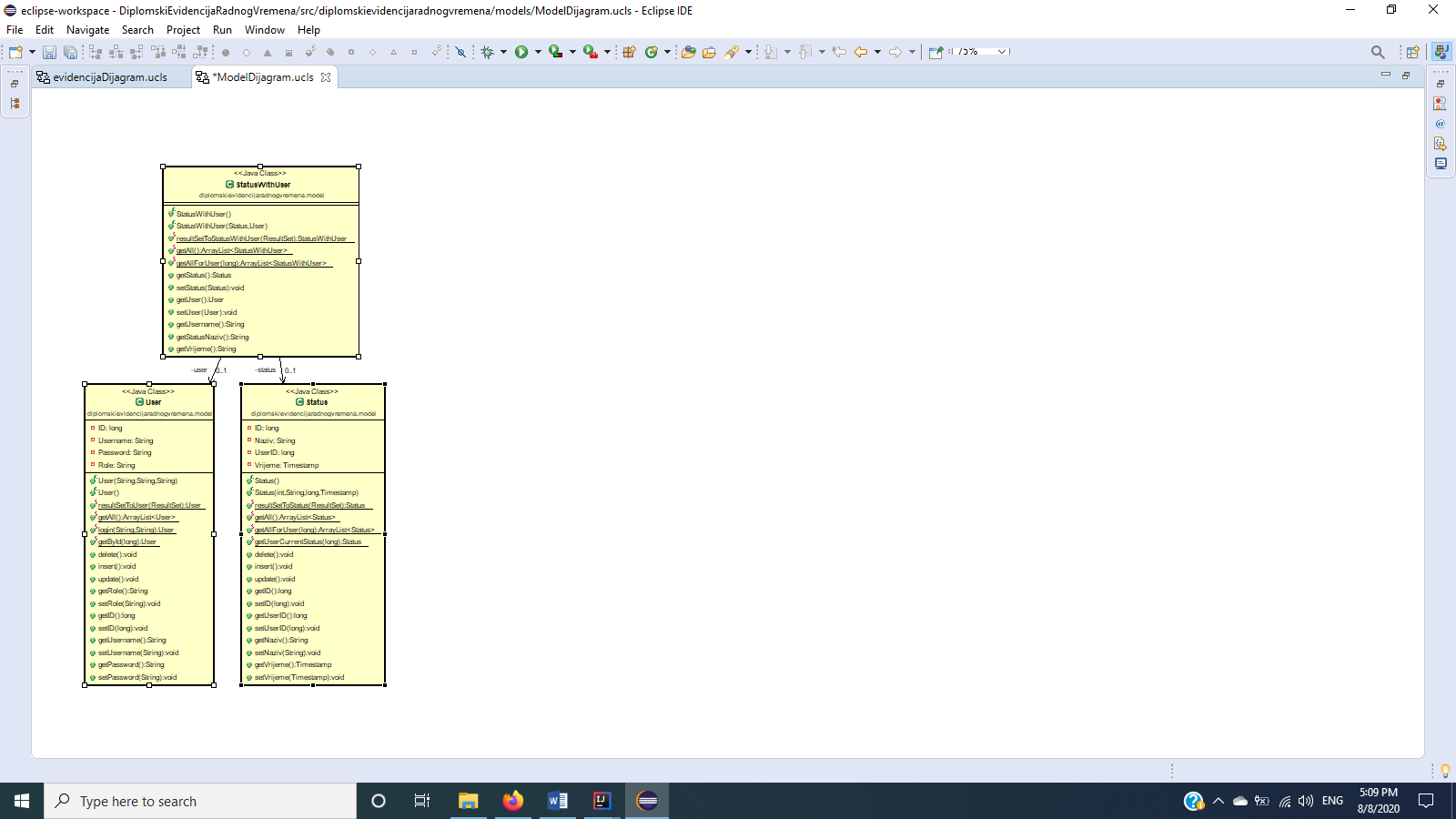


**Slika 22. Relacijski model baze podataka**

Prilikom implementacije korištene su dvije tablice unutar baze podataka. Tablica users, za spremanje podataka o korisnicima te tablica statusi za spremanje radnih statusa korisnika.

Statusi su vezani za korisnika pomoću stranog ključa te se na taj način mogu lako iz baze povući svi statusi za određenog korisnika.

## Dijagram klasa

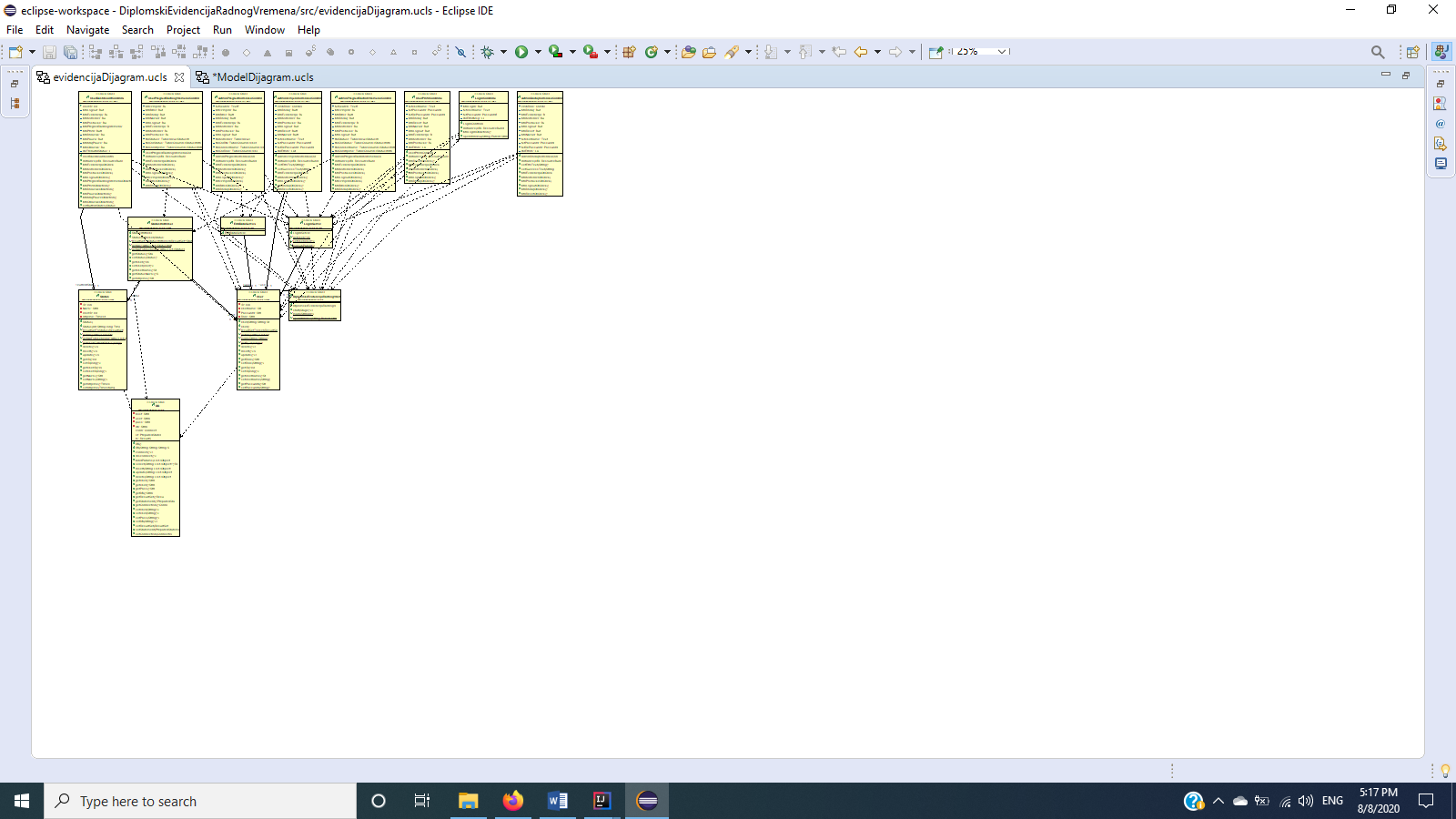


Slika 11 Dijagram klasa za paket model

Dijagram klasa za paket **model** je posebno izdvojen iz razloga što je potreban poseban osvrt na njega budući da on predstavlja kostur same aplikacije.

Kao što se može vidjeti iz prethodne slike paket **model** se sastoji iz tri klase:

1. *StatusWithUser –* predstavlja spoj klase *User* i klase *Status*
2. *User* – služi za rukovanje korisnicima sustava, te za interakciju istih sa bazom podataka
3. *Status* –služi za rukovanje radnim statusima korisnika sustava, te za interakciju istih sa bazom podataka



Slika 12 Dijagram klasa kompletne aplikacije

Dijagram klasa kompletne aplikacije nije jasno prikazan na prethodnoj slici jer ga nije moguće u potpunosti prikazati na papiru A4 formata, stoga je pobliže objašnjen u daljem tekstu. Naime, tu su prikazane sve datoteke aplikacije koje imaju ekstenziju *.java*.

Prvi red na prethodnoj slici su upravitelji, odnosno klase iz paketa *controller*, dole lijevo su klase paketa *models,* a dole desno se nalazi paket *services.*

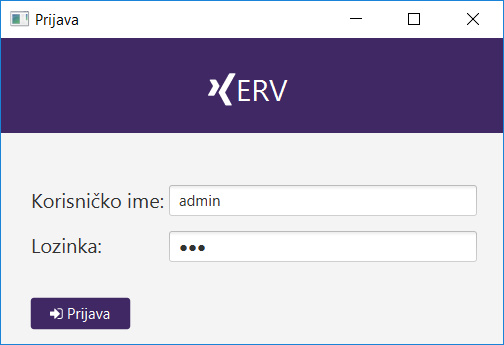
## Prozori aplikacije

U ovome odlomku su prikazani svi prozori aplikacije koji se pojavljuju u aplikaciji, kako prozori namijenjeni za administratore tako i oni za korisnike sustava. Uz tekstualni opis svih prozora, prikazane su i slike zaslona, a nakon pojedinih slika i uz pomoć online sintaksnog naglašivača prikazani su i prijepisi koda koji omogućavaju funkcioniranje aplikacije za evidenciju radnog vremena.

### Prijava

Pri pokretanju aplikacije najprije je potrebno prijaviti se. Postoje dva polja, polje *Username* i polje *Password* u koja je potrebno unijeti korisničko ime odnosno lozinku. Zatim kliknemo na dugme *Login* koji ima Event Handler u *LoginControler.java,* odatle se preko klase *User* provjerava da li u bazi postoji korisnik sa zadatim parametrima. Ukoliko ne postoji korisnik sa zadatim parametrima pojavit će se ispisane poruke kao što su:

* *Greška prilikom logiranja!* – ukoliko ne postoji korisnik sa zadatim parametrima
* *Unesite lozinku!* – ukoliko lozinka nije unijeta u odgovarajuće polje
* *Unesite korisničko ime!* – ukoliko korisničko ime nije unešeno u odgovarajuće polje



Slika 13 Prikaz prozora za prijavu

1. **package** diplomskievidencijaradnogvremena.controllers;
3. **import** diplomskievidencijaradnogvremena.models.User;
4. **import** diplomskievidencijaradnogvremena.services.LoginService;
5. **import** java.io.IOException;
6. **import** java.net.URL;
7. **import** java.util.ResourceBundle;
8. **import** java.util.logging.Level;
9. **import** java.util.logging.Logger;
10. **import** javafx.fxml.FXML;
11. **import** javafx.fxml.FXMLLoader;
12. **import** javafx.fxml.Initializable;
13. **import** javafx.scene.Parent;
14. **import** javafx.scene.Scene;
15. **import** javafx.scene.control.Button;
16. **import** javafx.scene.control.Label;
17. **import** javafx.scene.control.PasswordField;
18. **import** javafx.scene.control.TextField;
19. **import** javafx.stage.Stage;
21. **public** **class** LoginController **implements** Initializable {
23. @FXML
24. Button btnLogin;
26. @FXML
27. TextField txtUsername;
29. @FXML
30. PasswordField txtPassword;
32. @FXML
33. Label lblErrorMsg;
35. @Override
36. **public** **void** initialize(URL url, ResourceBundle rb) {
37. txtUsername.setText("admin");
38. txtPassword.setText("asd");
40. }
42. **public** **void** btnLoginOnAction() {
44. String username = String.valueOf(txtUsername.getText());
45. String password = String.valueOf(txtPassword.getText());
47. **if** (!username.equals("")) {
48. **if** (!password.equals("")) {
49. User user = User.login(username, password);
51. **if** (user != **null**) {
52. LoginService.setUser(user);
53. System.out.println(user.getRole());
54. **if** (user.getRole().equals("admin")) {
55. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Admin - Pregled radnih vremena", txtUsername.getParent(), "AdminPregledRadnihVremena");
56. } **else** **if** (user.getRole().equals("user")) {
57. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("User - Dashboard", txtUsername.getParent(), "UserDashboard");
58. }
59. } **else** {
60. lblErrorMsg.setText("Greška prilikom logiranja!");
61. }
62. } **else** {
63. lblErrorMsg.setText("Unesite lozinku!");
64. }
65. } **else** {
66. lblErrorMsg.setText("Unesite korisničko ime!");
67. }
68. }
70. **public** **void** openWindow(String title, Parent mainWindow, String path) {
71. }
73. }

Kao što je prethodno navedeno u klasi *LoginController.java* je vidljiv cjeloviti proces validacije prilokom prijave. Interakcija sa bazom podataka se vrši preko metoda u klasi *User* i klasi *DB.*

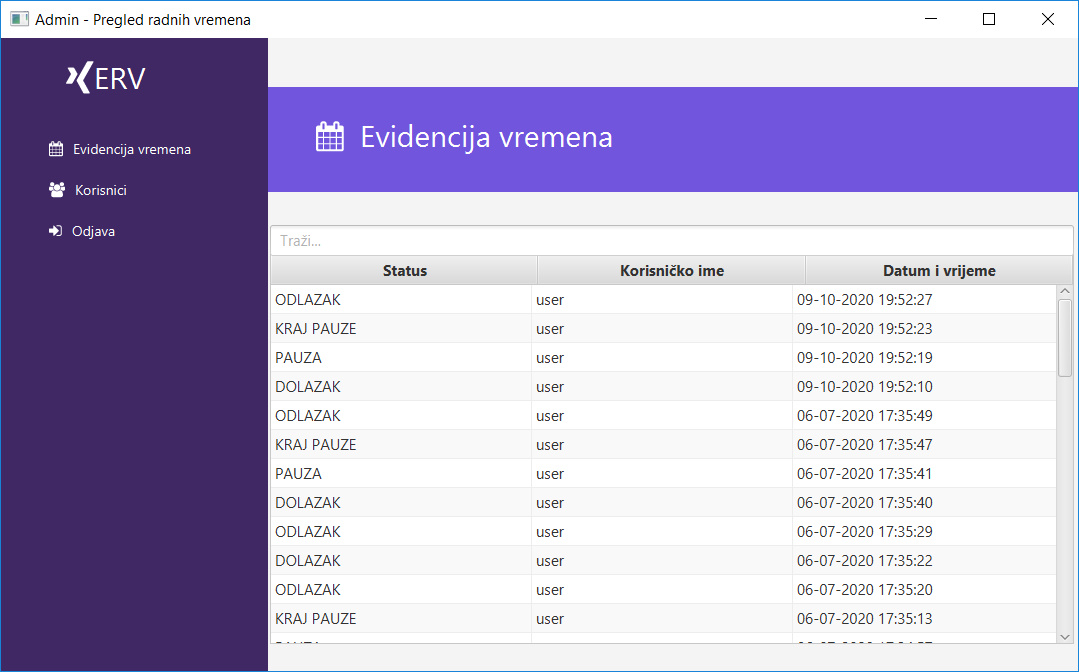
### Evidencija vremena - Administrator

Ukoliko je prijava uspješna i ukoliko je izvršena s korisničnog računa koji ima ulogu Administratora aplikacija će prikazati prozor *Evidencija vremena*, u kojoj se nalazi evidencija radnog vremena svih postojećih korisnika. Sa lijeve strane se može vidjeti navigacijski izbornik s tri opcije:

* *Evidencija vremena*  (trenutni prozor)
* *Korisnici* (za pregled, dodavanje i brisanje svih korisnika)
* *Logout* (za prekidanje sesije, odnosno odjavljivanje)

Prozor koji se trenutno prikazuje se sastoji od tablice sa svim ativnostima nad radnim vremenom. Tablica se sastoji od tri stupca:

1. *Status* – označava aktivnost koja se odigrala
2. *Username* – označava korisničko ime korisnika koji je izvršio gore navedenu aktivnost
3. *Datum i vrijeme* – označava vrijeme i datum kada se ta aktivnost odigrala

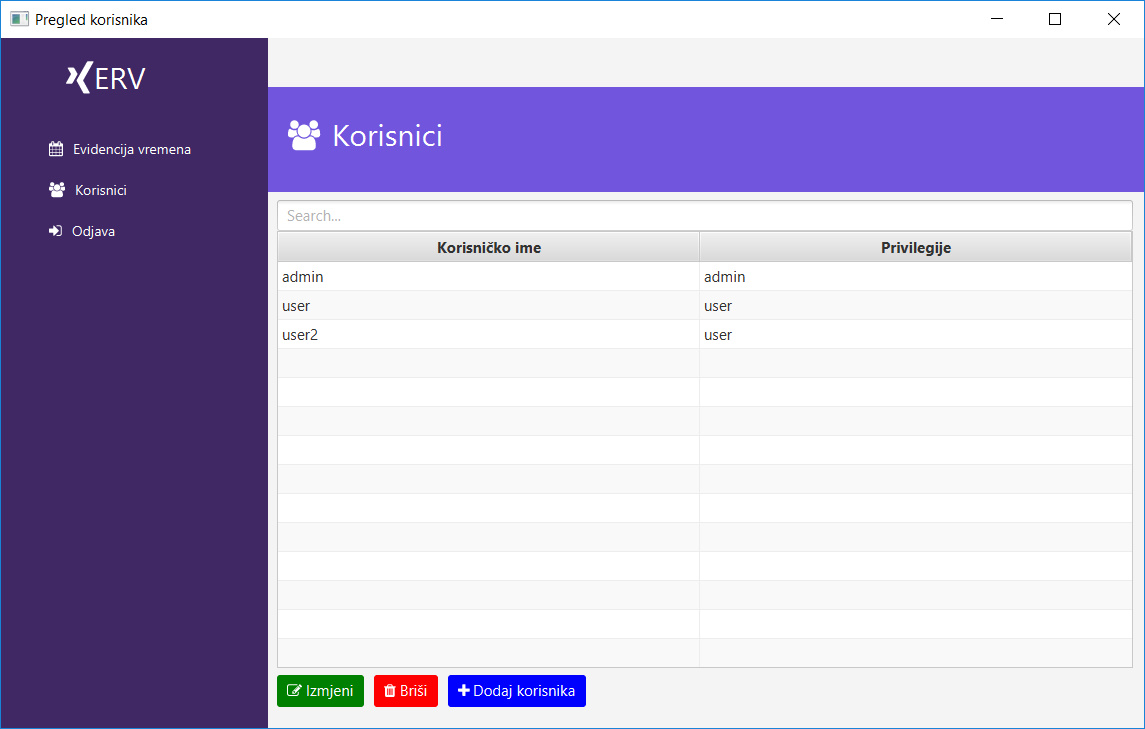


Slika 14 Admin Pregled radnih vremna

### Pregled korisnika- Administrator

Kada Administrator klikne mišom na opciju *Korisnici* u meniju s lijeve strane dobiva tablicu koja prikazuje sve korisnike zajedno sa njihovim ulogama u dva reda: *Korisničko ime* i *Privilegije*. Red *Privilegije* predstavlja ništa više no ulogu svakog korisnika. Ispod tabele, u dnu prozora imamo tri drugmeta:

1. *Izmjeni* – mjenjamo podatke o selektovanom korisniku
2. *Briši* – brišemo selektovanog korisnika iz baze podataka
3. *Dodaj korisnika* – dodajemo novog korisnika u bazu podataka



Slika 15 Administrator- Pregled korisnika

1. /\*
2. \* To change this license header, choose License Headers in Project Properties.
3. \* To change this template file, choose Tools | Templates
4. \* and open the template in the editor.
5. \*/
6. **package** diplomskievidencijaradnogvremena.models;
8. **import** diplomskievidencijaradnogvremena.DB;
9. **import** java.sql.ResultSet;
10. **import** java.sql.SQLException;
11. **import** java.util.ArrayList;
12. **import** java.util.Arrays;
13. **import** java.util.List;
14. **import** java.util.logging.Level;
15. **import** java.util.logging.Logger;
17. /\*\*
18. \*
19. \* @author owner
20. \*/
21. **public** **class** User {
22. **private** **long** ID;
23. **private** String Username;
24. **private** String Password;
25. **private** String Role;

28. **public** User(String username, String password, String role) {
29. **this**.ID = 0;
30. **this**.Username = username;
31. **this**.Password = password;
32. **this**.Role = role;
33. }
35. **public** User() {
36. **this**.ID = 0;
37. **this**.Username = "";
38. **this**.Password = "";
39. **this**.Role = "";
40. }

43. **public** **static** User resultSetToUser(ResultSet rs){
44. User user = **null**;
45. **try** {
46. user = **new** User(rs.getString("Username"), rs.getString("Password"), rs.getString("Role"));
47. user.setID(rs.getLong("ID"));
48. } **catch** (SQLException ex) {
49. Logger.getLogger(User.**class**.getName()).log(Level.SEVERE, **null**, ex);
50. }
51. **return** user;
52. }

55. **public** **static** ArrayList<User>getAll(){
56. ArrayList<User> users = **new** ArrayList<>();
57. List<Object> params = Arrays.asList();
58. DB db = **new** DB();
59. db.select("SELECT \* FROM users", params);
61. **try** {
62. **while**(db.getResultSet().next()){
63. users.add(resultSetToUser(db.getResultSet()));
64. }
65. } **catch** (SQLException ex) {
66. Logger.getLogger(User.**class**.getName()).log(Level.SEVERE, **null**, ex);
67. }
68. db.disconnect();
69. **return** users;
70. }



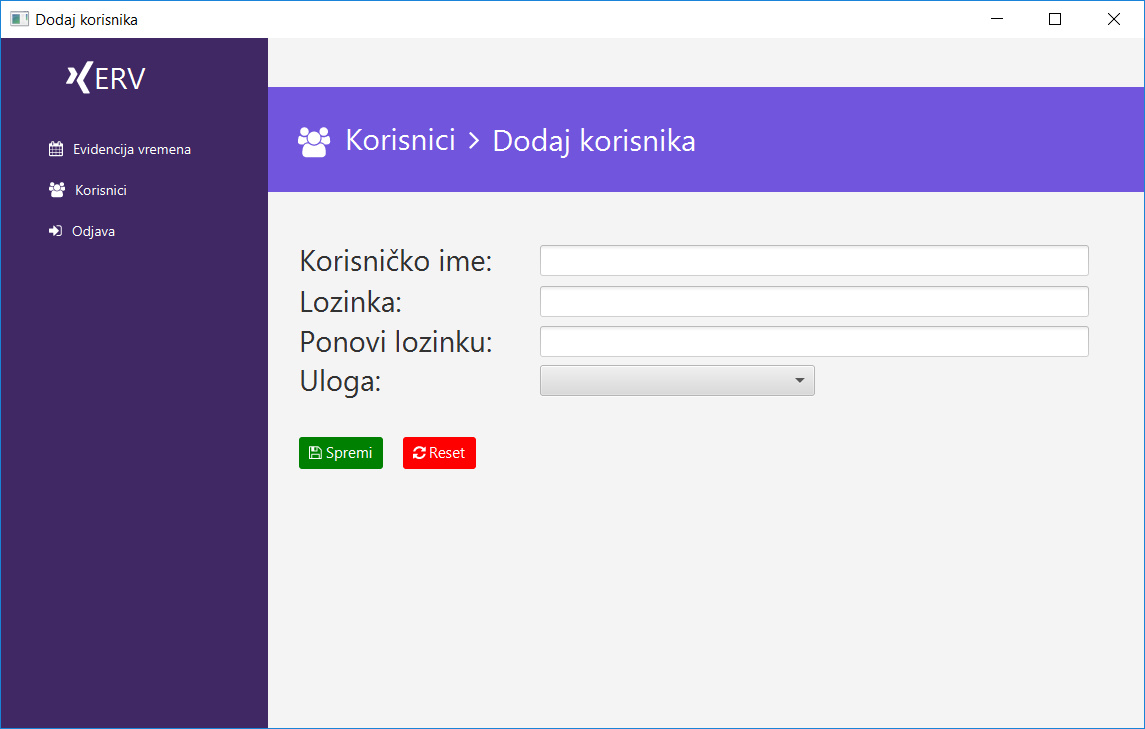
75. **public** **static** User login(String username, String password){
76. User user = **null**;
78. DB db = **new** DB();
79. List<Object> params = Arrays.asList(username, password);
80. db.select("SELECT \* FROM users WHERE Username=? AND Password=?", params);
81. **try** {
82. **while**(db.getResultSet().next()){
83. user = resultSetToUser(db.getResultSet());
84. }
85. } **catch** (SQLException ex) {
86. Logger.getLogger(User.**class**.getName()).log(Level.SEVERE, **null**, ex);
87. }
88. db.disconnect();
89. **return** user;
90. }
92. **public** **static** User getById(**long** ID){
93. User user = **null**;
94. DB db = **new** DB();
95. List<Object> params = Arrays.asList(ID);
96. db.select("SELECT \* FROM users WHERE ID=?", params);
97. **try** {
98. **while**(db.getResultSet().next()){
99. user = resultSetToUser(db.getResultSet());
100. }
101. } **catch** (SQLException ex) {
102. Logger.getLogger(User.**class**.getName()).log(Level.SEVERE, **null**, ex);
103. }
104. db.disconnect();
105. **return** user;
106. }
108. **public** **void** delete(){
109. DB db = **new** DB();
110. List<Object> params = Arrays.asList(**this**.ID);        //db.delete("DELETE FROM kolegij WHERE PredavacID=?", params);
111. db.delete("DELETE FROM users WHERE ID=?", params);
112. db.disconnect();
113. }
115. **public** **void** insert(){
116. DB db = **new** DB();
117. List<Object> params = Arrays.asList(**this**.Username, **this**.Password, **this**.Role);
118. db.insert("INSERT INTO users(Username, Password, Role) VALUES(?,?,?)", params);
119. db.disconnect();
120. }
122. **public** **void** update(){
123. DB db = **new** DB();
124. List<Object> params = Arrays.asList(**this**.Username, **this**.Password, **this**.Role, **this**.ID);
125. db.update("UPDATE users SET Username=?, Password=?, Role=? WHERE ID=?", params);
126. db.disconnect();
127. }
129. **public** String getRole() {
130. **return** Role;
131. }
133. **public** **void** setRole(String Role) {
134. **this**.Role = Role;
135. }
137. **public** **long** getID() {
138. **return** ID;
139. }
141. **public** **void** setID(**long** ID) {
142. **this**.ID = ID;
143. }
145. **public** String getUsername() {
146. **return** Username;
147. }
149. **public** **void** setUsername(String Username) {
150. **this**.Username = Username;
151. }
153. **public** String getPassword() {
154. **return** Password;
155. }
157. **public** **void** setPassword(String Password) {
158. **this**.Password = Password;
159. }
160. }

U kodu iznad možemo vidjeti sve metode za interakciju sa korisnicima koje su potrebne. Klasa *User* je jedna od najbitnijih klasa ove aplikacije.

### Dodaj korisnika - Administrator

Ukoliko administrator klikne na dugme *Dodaj korisnika* otvorit će mu se novi okvir gdje treba unijeti podatke o novom korisniku. Potrebno je unijeti korisničko ime, lozinku u dva navrata i odabrati ulogu novog korisnika, odnosno odrediti hoće li će on biti administrator ili korisnik.

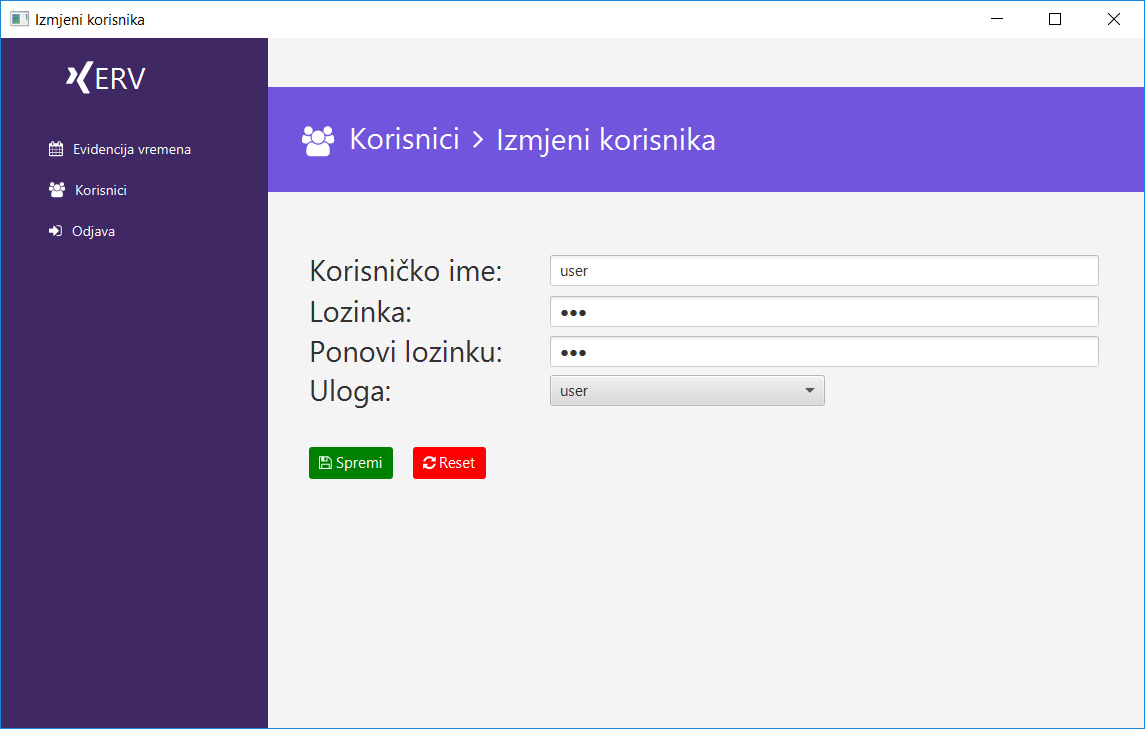
Nakon što su popunjena sva polja potrebno je kliknuti na dugme *Spremi* koje će se izvršiti u *AdminDodajKorisnikaController* odakle će novi korisnik preko klase *User* biti upisan u *MySQL* bazu podataka.



Slika 16 Administrator- Dodaj Korisnika

### Izmjeni korisnika - Administrator

Ukoliko Administrator želi izmijeniti podatke o odabranom korisniku, nakon što klikne na dugme *Izmjeni* iz menija *Korisnici*, pojavit će mu se sličan prozor kao i za dodavanje novog korisnika, jedina razlika je što su u ovom prozoru polja već popunjena, a od administratora se traži samo da prepravi polja koja želi promijeniti i da klikne na dugme *Spremi* nakon čega će podaci o odabranom korisniku biti prepravljeni u bazi podataka preko klase *User*.



Slika 17 Administrator- Izmjeni Korisnika

### Početna - Korisnik

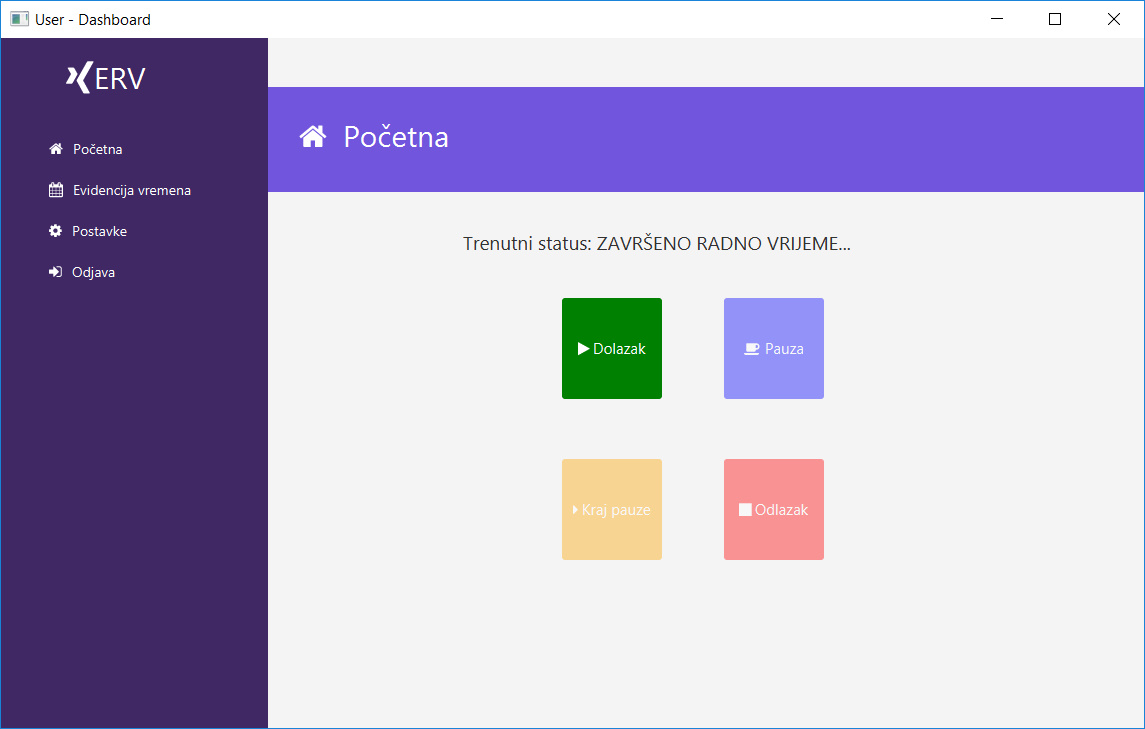
Ukoliko je prijava izvršena s podacima računa koji ima ulogu *Korisnik,* a ne *Administrator,* otvara se prozor koji ima nešto drugačije korisničko sulčelje. Prije svega treba primjetiti da imamo drugačije opcije u izborniku s lijeve strane:

1. *Početna –* sučelje za interakciju korisnika sa aplikacijom (za unos evidencije vremena)
2. *Evidencija vremena* – prikazuje evidenciju vremena prijavljenog korisnika
3. *Postavke –* za podešavanje računa, promjenu lozinke ili korisničkog imena
4. *Logout –* za prekidanje sesije (odjava)

Na slici ispod prikazan je prozor *Početna* koji nije dostupan za korisnike koji imaju ulogu *Administrator*. On predstavlja sučelje za unos evidencije vremena zaposlenog, tj. predstavlja jedan od glavnih prozora aplikacije.

Aplikacija ima četiri dugmeta (buttona):

1. *Dolazak* – predstavlja početak radnog vremena
2. *Pauza* – označava početak pauze
3. *Kraj pauze*
4. *Odlazak* – kraj radnog vremena



Slika 18 Početna stranica- Korisnik

1. /\*
2. \* To change this license header, choose License Headers in Project Properties.
3. \* To change this template file, choose Tools | Templates
4. \* and open the template in the editor.
5. \*/
6. **package** diplomskievidencijaradnogvremena.controllers;
8. **import** diplomskievidencijaradnogvremena.models.Status;
9. **import** diplomskievidencijaradnogvremena.services.LoginService;
10. **import** java.net.URL;
11. **import** java.sql.Timestamp;
12. **import** java.time.Instant;
13. **import** java.util.ResourceBundle;
14. **import** javafx.fxml.FXML;
15. **import** javafx.fxml.Initializable;
16. **import** javafx.scene.control.Button;
17. **import** javafx.scene.control.Label;
19. /\*\*
20. \* FXML Controller class
21. \*
22. \* @author owner
23. \*/
24. **public** **class** UserDashboardController **implements** Initializable {
25. **public** **long** UserID = LoginService.getUser().getID();
26. **public** Status currentStatus = Status.getUserCurrentStatus(UserID);
28. @FXML
29. Button btnLogout, btnEvidencija, btnKorisnici, btnPostavke;
31. @FXML
32. Button btnPregledRadnogVremena;
34. @FXML
35. Button btnProfil;
37. @FXML
38. Button btnDolazak;
40. @FXML
41. Button btnPauza;
43. @FXML
44. Button btnKrajPauze;
46. @FXML
47. Button btnOdlazak;
49. @FXML
50. Label lblTrenutniStatus;

53. @Override
54. **public** **void** initialize(URL url, ResourceBundle rb) {
55. setButtonStates(currentStatus);
56. }

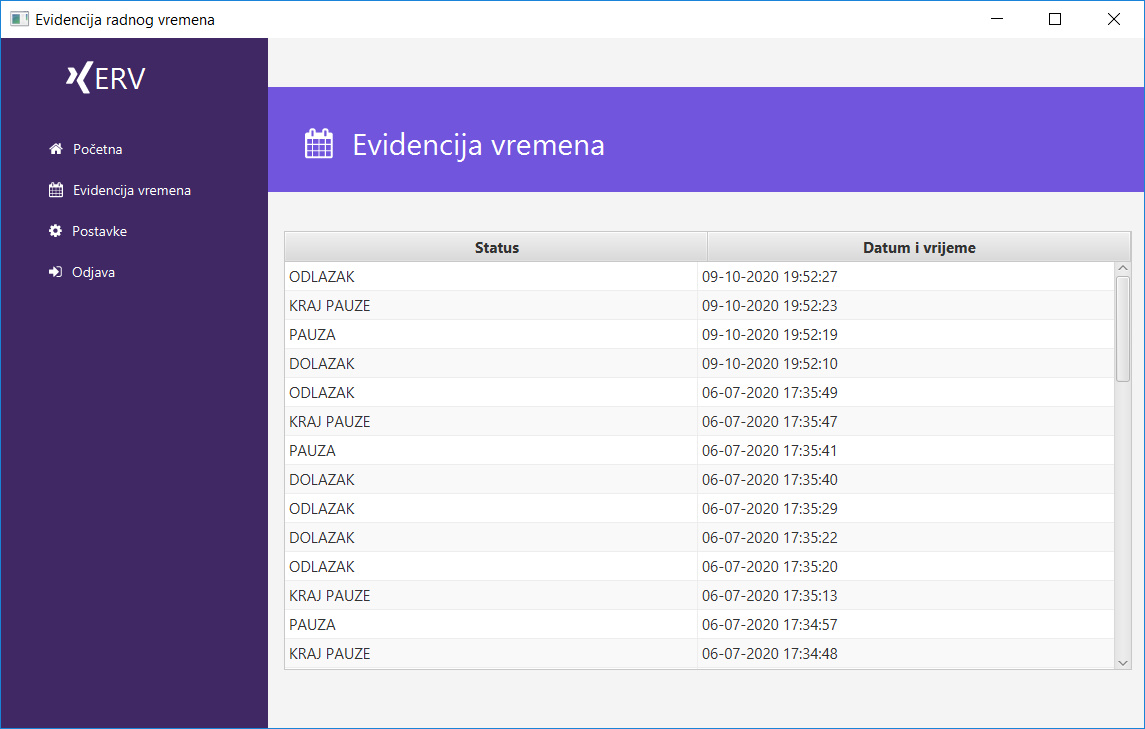


61. **public** **void** btnEvidencijaOnClick(){
62. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Evidencija radnog vremena", btnLogout.getParent(), "UserPregledRadnogVremena");
63. }
64. **public** **void** btnKorisniciOnClick(){
65. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Početna", btnLogout.getParent(), "UserDashboard");
66. }
67. **public** **void** btnPostavkeOnClick(){
68. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Postavke", btnLogout.getParent(), "UserProfil");
69. }
70. **public** **void** btnLogoutOnClick(){
71. LoginService.logout(btnLogout.getParent());
72. }
74. **public**  **void** btnPregledRadnogVremenaOnAction() {
75. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Pregled radnog vremena", btnDolazak.getParent(), "UserPregledRadnogVremena");
76. }
78. **public**  **void** btnProfilOnAction() {
79. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Korisnički profil", btnDolazak.getParent(), "UserProfil");
80. }
82. **public**  **void** btnDolazakOnAction() {
83. Status status = **new** Status(0, "DOLAZAK", UserID, Timestamp.from(Instant.now()));
84. status.insert();
85. currentStatus = status;
86. setButtonStates(status);
87. }
89. **public**  **void** btnPauzaOnAction() {
90. Status status = **new** Status(0, "PAUZA", UserID, Timestamp.from(Instant.now()));
91. status.insert();
92. currentStatus = status;
93. setButtonStates(status);
94. }
96. **public**  **void** btnKrajPauzeOnAction() {
97. Status status = **new** Status(0, "KRAJ PAUZE", UserID, Timestamp.from(Instant.now()));
98. status.insert();
99. currentStatus = status;
100. setButtonStates(status);
101. }
103. **public**  **void** btnOdlazakOnAction() {
104. Status status = **new** Status(0, "ODLAZAK", UserID, Timestamp.from(Instant.now()));
105. status.insert();
106. currentStatus = status;
107. setButtonStates(status);
108. }

111. **public** **void** setButtonStates(Status status) {
112. **if**(status != **null**){
113. **if**(status.getNaziv().equals("DOLAZAK")){
114. lblTrenutniStatus.setText(String.valueOf("Trenutni status: NA POSLU..."));
115. btnDolazak.setDisable(**true**);
116. btnOdlazak.setDisable(**false**);
117. btnPauza.setDisable(**false**);
118. btnKrajPauze.setDisable(**true**);
119. }
120. **else** **if**(status.getNaziv().equals("ODLAZAK")){
121. lblTrenutniStatus.setText(String.valueOf("Trenutni status: ZAVRŠENO RADNO VRIJEME..."));
122. btnDolazak.setDisable(**false**);
123. btnOdlazak.setDisable(**true**);
124. btnPauza.setDisable(**true**);
125. btnKrajPauze.setDisable(**true**);
126. }
127. **else** **if**(status.getNaziv().equals("PAUZA")){
128. lblTrenutniStatus.setText(String.valueOf("Trenutni status: NA PAUZI..."));
129. btnDolazak.setDisable(**true**);
130. btnOdlazak.setDisable(**true**);
131. btnPauza.setDisable(**true**);
132. btnKrajPauze.setDisable(**false**);
133. }
134. **else** **if**(status.getNaziv().equals("KRAJ PAUZE")){
135. lblTrenutniStatus.setText(String.valueOf("Trenutni status: NA POSLU..."));
136. btnDolazak.setDisable(**true**);
137. btnOdlazak.setDisable(**false**);
138. btnPauza.setDisable(**false**);
139. btnKrajPauze.setDisable(**true**);
140. }
141. }
142. **else**{
143. lblTrenutniStatus.setText(String.valueOf("Trenutni status: ZAVRŠENO RADNO VRIJEME..."));
144. btnDolazak.setDisable(**false**);
145. btnOdlazak.setDisable(**true**);
146. btnPauza.setDisable(**true**);
147. btnKrajPauze.setDisable(**true**);
148. }
149. }
151. }

### Evidencija vremena - Korisnik

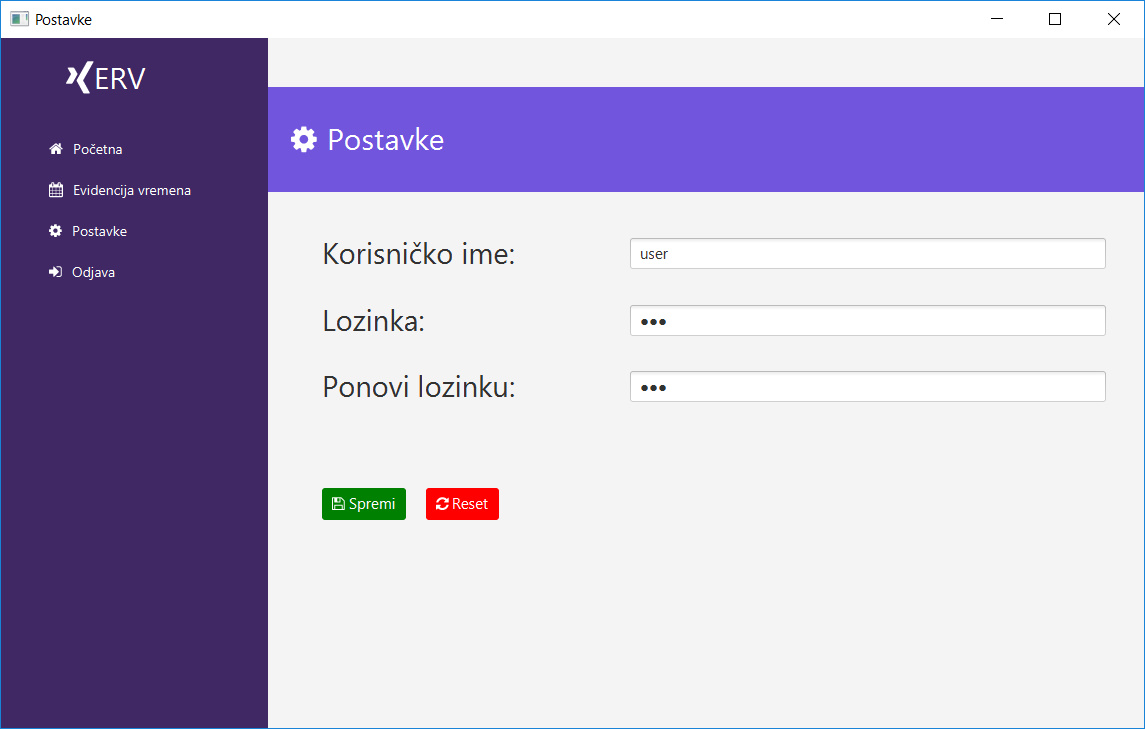
Ukoliko je u izborniku odabrana opcija *Evidencija vremena,* prikazuje se nešto drugačija tablica od one koja se prikazuje administratoru. Naime, ne prikazuje se evidencija vremena za sve korisnike, već samo prikaz vremena onog korisnika koji je trenutno prijavljen pa je stoga i sama tablica nešto drugačija. Tablica sadrži dvije kolone *Status* i *Datum i vrijeme,* koje predstavljaju akciju koja se odigrala i vremenski period u kom se ta akcija odigrala.



Slika 19 Korisnik- Evidencija radnog vremena

### Postavke - Korisnik

Treba napomenuti da u aplikaciji postoji još jedna dodatna opcija a to je *Postavke,* koja služi za podešavanje naloga, odnosno za promjenu korisničkog imena i lozinke. Tekstualna polja su već popunjena sa trenutnim vrijednostima koja se nalaze u bazi, a sve što korisnik treba učiniti da bi uradio izmjenu je da validno prepravi željena polja i klikne na dugme *Spremi.* Ako je sve popunjeno kako valja izvršit će se izmjene u bazi kroz klasu *User*.

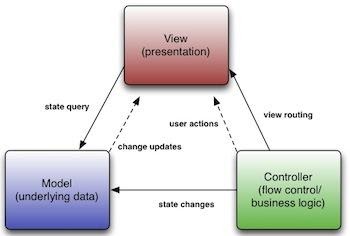


Slika 20 Postavke profila- korisnik

## MVC arhitektura u aplikaciji

U ovom dijelu rada će biti pobliže objašnjen MVC arhitektonski obrazac koji je implementiran unutar aplikacije za evidenciju radnog vremena. MVC arhitektonski obrazac koji je dugo postojao u programskom inženjerstvu. Većinom svi jezici koriste MVC s malim odstupanjima, ali konceptualno ostaje isti.

Kao što je već objašnjeno u prvom dijelu rada MVC znači Model, View i Controller te MVC razdvaja aplikaciju na tri komponente - Model, View i Controller.



Slika 21 Tok aplikacije koristeći MVC arhitektonski obrazac

Model: Model predstavlja oblik podataka i poslovnu logiku. Održava podatke aplikacije. Objekti modela dohvaćaju i spremaju stanje modela u bazu podataka.

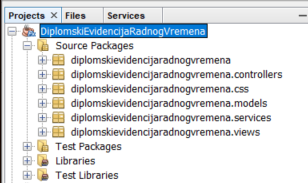
Pogled: Pogled je korisničko sučelje. Pregledajte prikazne podatke pomoću modela korisniku i također im omogućuje izmjenu podataka.

Upravitelj: Upravitelj obrađuje korisnički zahtjev. Korisnik obično komunicira s Pogledom, koji zauzvrat podiže odgovarajući zahtjev za URL-om, tim će zahtjevom upravljati upravitelj. Upravitelj daje odgovarajući prikaz s podacima modela kao odgovorom.

Unutar aplikacije, kreirana su 4 java paketa:

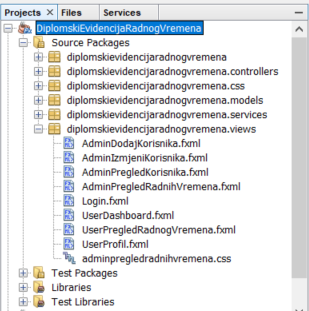
**diplomskievidencijaradnogvremena.controllers, diplomskievidencijaradnogvremena.models, diplomskievidencijaradnogvremena.views i**

**Diplomskievidencijaradnogvremena.css**

****

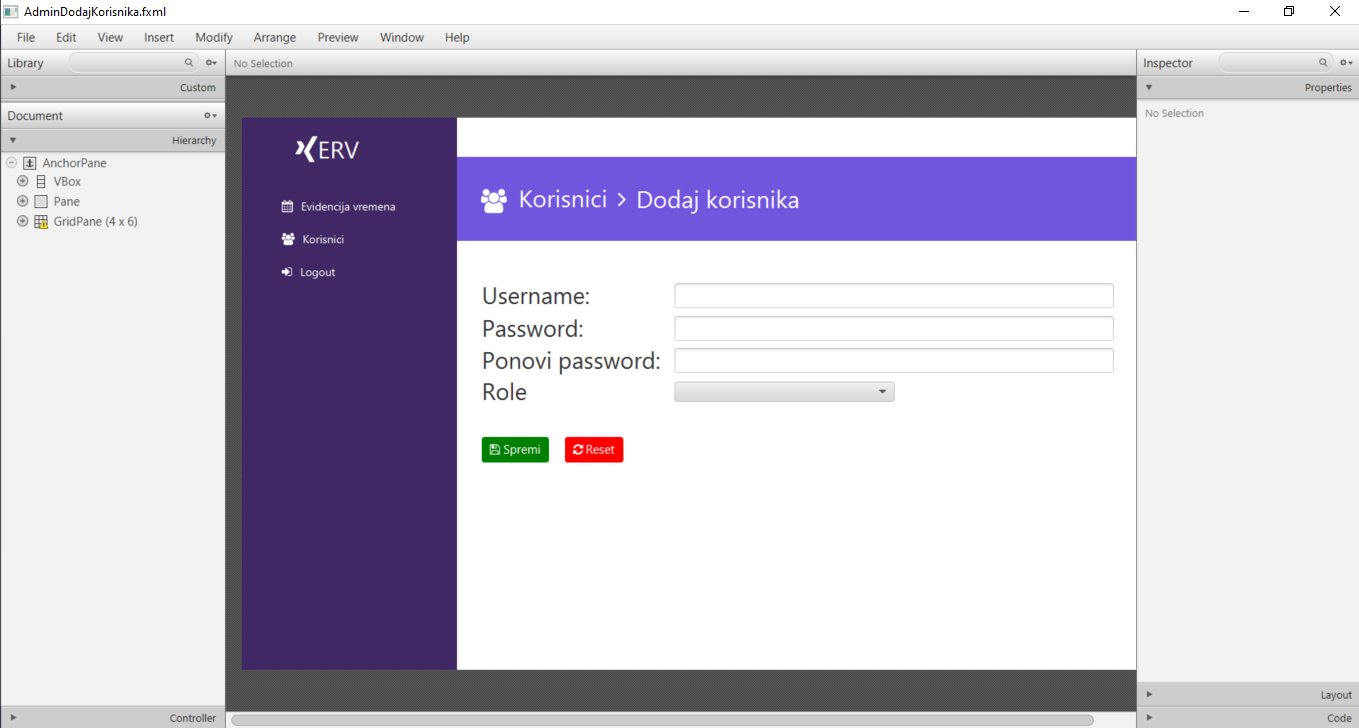
Slika 22 Prikaz svih java paketa aplikacije

U paketu **diplomskievidencijaradnogvremena.views** nalaze se pogledi (engl. „*Views“)* aplikacije, to su ustvari prozori aplikacije pisani u .fxml-u. Svaki pogled (engl. *„View“)*  ima i svoj css file te upravitelja koji njime upravlja. Npr. **AdminDodajKorisnika.fxml** pogled, ima upravitelja koji njime upravlja unutar controllers paketa, koji se zove **AdminDodajKorisnikaController.java**, te ima css fajl za oblikovanje izgleda unutar css paketa **admindodajkorisnika.css.**

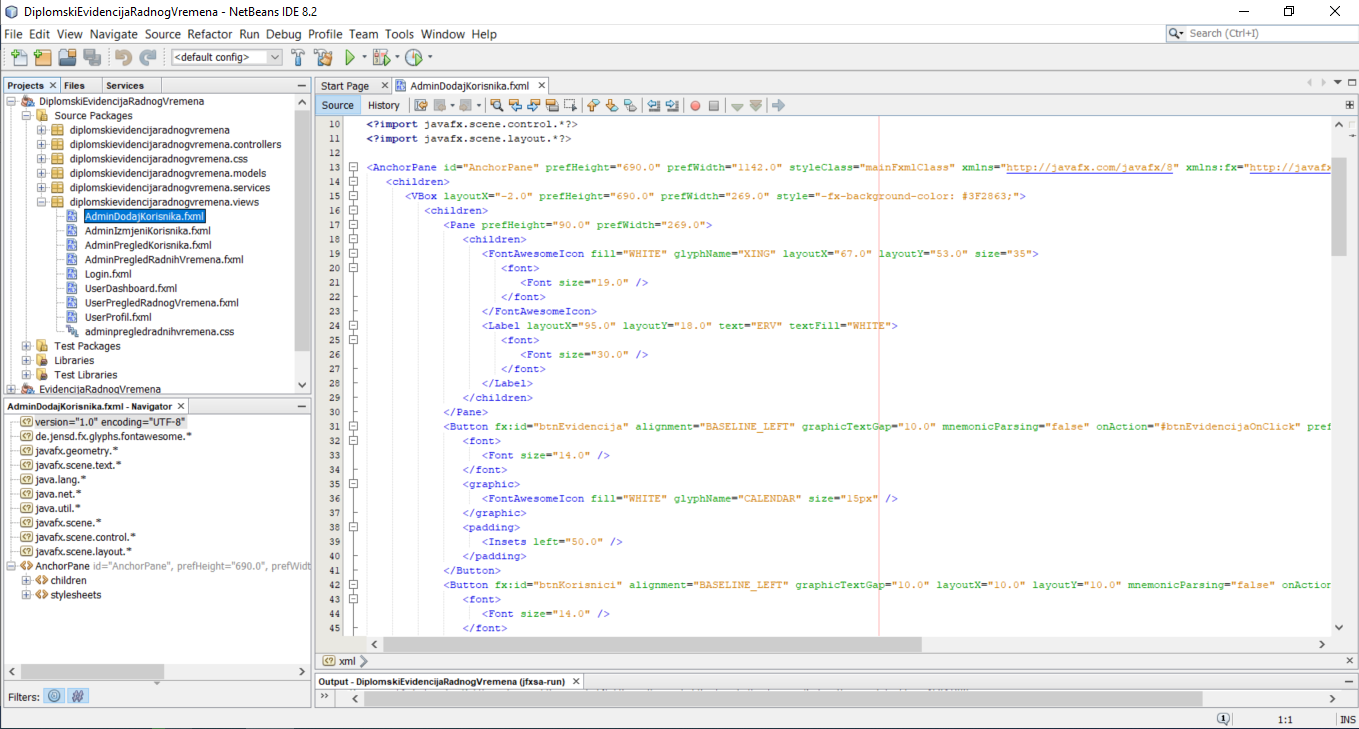


Slika 23 Pregled datoteka unutar views paketa

Poglede je moguće uređivati izravno pomoću fxml koda ili koristeći JavaFX Scene Builder 2.0.

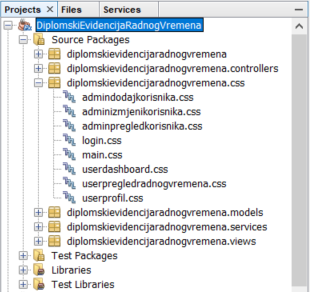


Slika 24 AdminDodajKorisnika.fxml fajl pokrenut unutar JavaFX Scene Builder 2.0



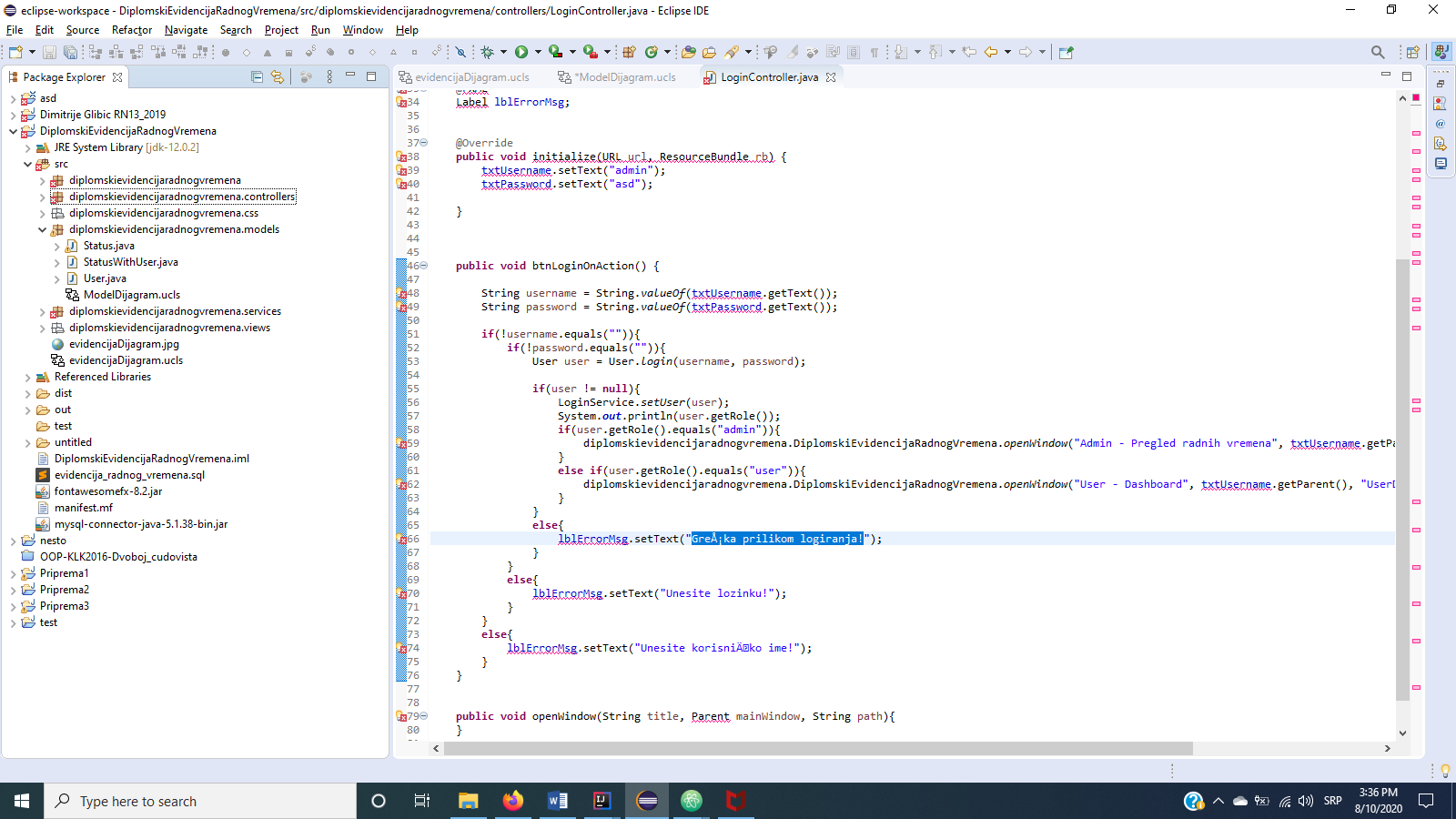
Slika 25 AdminDodajKorisnika.fxml prikaz koda unutar Netbeans-a

U paketu **diplomskievidencijaradnogvremena.css** nalaze se css (engl. „*Cascading StyleSheets“)*  koji služe za oblikovanje pogleda, isto kao i css kod web dizajna, možemo korisiti css klase i atribute za oblikovanje elemenata.



Slika 26 Pregled datoteka unutar css paketa

U paketu **diplomskievidencijaradnogvremena.models** nalaze se modeli koji služe za lakše rukovanje podacima iz baze. Svaka tablica unutar baze ima prikladni model unutar paketa models. Model klase sadrže metode za dohvaćanje, izmjenu, pregled i brisanje podataka iz baze.



Slika 27 Pregled datoteka unutar models paketa

## Slučajevi korištenja

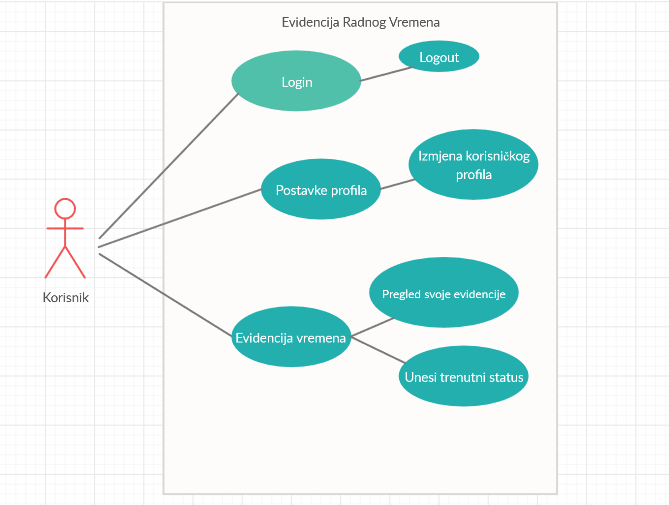
Kao što je već navedeno, implementirana Java aplikacija razlikuje dvije razine korisničkog sučelja; korisničko sučelje za administratora te korisničko sučelje za korisnike koji evidentiraju svoje radno vrijeme. Samim time, u aplikaciji postoje dvije razine slučajeva korištenja- slučajevi korištenja za obične korisnike te slučajevi korištenja za administratore.

Na slici 48 se nalaze slučajevi korištenja za običnog korisnika, koji se služi aplikacijom kako bi evidentirao vlastito radno vrijeme provedeno na poslu. Ti slučajevi korištenja podrazumijevaju funkcionalnost login-a (prijave na sustav), postavljanja profila i funkcionalnost evidentiranja radnog vremena.

Funkcionalnost prijavljivanja na sustav podrazumijeva i mogućnost odjave sa sustava, nakon završetka korištenja aplikacije.

Funkcionalnost postavljanja profila uključuje mogućnost izmjena postavki profila kao što su korisničko ime i lozinka.

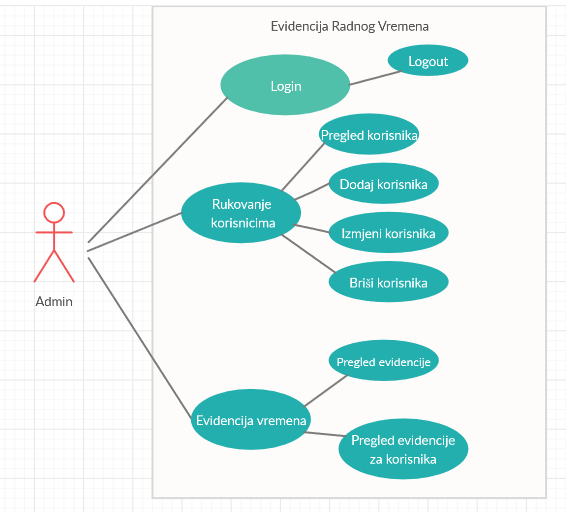
Korisnikova funkcionalnost evidencije vremena uključuje pregledavanje svih dosadašnjih evidencija radnog vremena i unošenje trenutnog statusa na poslu, odnosno dolaska, odlaska i pauze na poslu.



Slika 28 Prikaz slučajeva korištenja za korisnika

Na slici 49 se nalaze slučajevi korištenja za administratora sustava. Kao i kod običnih korisnika i administrator ima funkcionalnost prijave na sustav i evidentiranja radnog vremena, ali za razliku od korisnika administrator ima i funkcionalnost rukovanja svim ostalim korisnicima. Ova funkcionalnost omogućuje administratoru pregled, izmjenu i brisanje postojećih korisnika, ali i dodavanje novih korisnika.

Također, administrator ne evidentira vlastito radno vrijeme, nego se pod funkcionalnosti evidentiranja vremena podrazumijeva pregled cjelokupne evidencije i pregled evidencije za pojedinog korisnika.



Slika 29 Prikaz slučajeva korištenja za administratora

## Mogućnosti proširenja aplikacije

Aplikacija je izrađena prvenstveno u svrhe učenja i vježbanja programskoga jezika Java, te stoga se sa aplikacijom nije išlo u krajnosti i detalje. Cilj je bio samo napraviti najosnovniju aplikaciju u kojoj je moguće vidjeti od kada do kada je netko bio na poslu.

Ako bi se išlo u detalje, moguće je za svakog korisnika pratiti njegove osobne podatke, ime, prezime, email, jmbg, mjesto rođenja itd. Osim toga može se i omogućiti unos datoteka za svakoga korisnika, pa bi se mogle pratiti i kopije dokumenata korisnika npr. Osobne iskaznice, prometne, vozačke, certifikati itd..

Bilo bi lijepo kada bi se moglo pratiti koliko dana bolovanja je koji korisnik iskoristio tokom godine, te koliko ih se prenosi u iduću godinu. Također je moguće dinamički implementirati radne statuse, koji su trenutno hardkodirani sa vrijednostima: DOLAZAK, ODLAZAK, PAUZA i KRAJ PAUZE.

Trenutno implementirana desktop aplikacija zamišljena je da bude na ulazu u firmu, te radnici kako bi ušli u firmu moraju se prijaviti i unijeti da su došli na posao i od tog trenutka počinje brojanje radnog vremena. Moguće proširenje ove aplikacije bila bi integracija mobilne Android aplikacije. Na ovaj način klijenti bi mogli unijeti radni status izravno sa svojih mobilnih uređaja. Svi podaci bili bi dostupni za pregled i preko mobilnih aplikacija i preko desktop aplikacije. Kod ovakvog pristupa, za pristup bazi podataka bilo bi dobro implementirati tzv. REST API (engl. „*Representational State Transfer“),* koji bi odašiljao podatke preko http protokola u JSON formatu. Na ovaj način, postojao bi samo jedan backend neovisno o tome pristupamo li podacima iz baze preko mobilne android, ios, desktop, web aplikacije i sl. Svaka aplikacija bi podatke dohvaćala preko HTTP protokola u JSON (engl. *„Javascript Object Notation“)* obliku, i zatim bi ovakve podatke pretvarala u kod u kojemu je pisana aplikacija (npr. Java Android, Swift3 za iOS, ili c# web aplikacija...). REST API moguće je implementirati koristeći Spring Boot MVC framework unutar Jave, ali to je trenutno ostavljeno za moguća proširenja kako se ne bi išlo previše u detalje s aplikacijom.

# ZAKLJUČAK

U ovom diplomskom radu, bez velikih poteškoća, implementirana je jednostavna aplikacija za evidenciju radnog vremena. Korišten je Java programski jezik te arhitektonski obrazac MVC. Potrebno je poznavati Java-u, MySQL i CSS. Također je potrebno poznavati MVC i MVVM programske konstrukcijske dizajne. Čitava aplikacija je prvobitno implementirana koristeći samo Swing framework, ali je zbog brzine i dodatnih funkcionalnosti, aplikacija prepisana u JavFx-u.

Poznavanje JavaFX-a i CSS je dovoljno za implementaciju modernih, brzih i izgledom bogatih desktop aplikacija i sučelja, koje se mogu pokretati na svim operacijskim sustavima. Smatra se da će Java kao programski jezik za razvoj aplikacija biti relevantna i široko upotrijebljena barem narednih 10 godina.

Napisati o mogućnostima proširenja bar 2/3 stranice………

# LITERATURA

1. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf>
2. [javatpoint.com/features-of-java](file:///C:\Users\Korisnik\Desktop\javatpoint.com\features-of-java)
3. <https://hr.wikipedia.org/wiki/Java_(programski_jezik)>
4. <http://docplayer.rs/184548646-Programiranje-nastava-prof-dr-sc-dra%C5%BEena-ga%C5%A1par-datum.html>
5. <https://medium.com/@rahul77349/difference-between-compiler-and-interpreter-with-respect-to-jvm-java-virtual-machine-and-pvm-22fc77ae0eb7>
6. <https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf>

<https://www.topperskills.com/tutorials/java/features-properties-java-language.html>

1. Jaworski, J. (1998): Java 1.2: Unleashed. 4th Edition. Macmillan Computer Publishing, Indianapolis, IN, 1998
2. Hunter, J. i Crawford W. (1998). Java Servlet Programming. O'Reilly and Associates, Inc., Sebastopol, CA.

<https://www.weblineindia.com/blog/java-development-advantages-disadvantages/>

1. <http://laris.fesb.hr/java/osnovni_program.htm>

[https://docs.oracle.com/en/java/javase/15/docs/specs/sealed-classes-jls.html#jls-3.9](https://docs.oracle.com/en/java/javase/15/docs/specs/sealed-classes-jls.html%23jls-3.9)

1. [https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html#jls-3.9](https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html%23jls-3.9)

<http://laris.fesb.hr/java/varijable.htm>

1. <https://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/3.%20Klase_i_objekti.pdf>
2. <https://javaprogramiranje.wordpress.com/2009/11/07/upravljacke-naredbe/>
3. Burd B.(2017): Beginning Programming with Java For Dummies. IDG Books Worldwide, Inc. Foster City, California.
4. <https://www.softwaretestinghelp.com/java-components-java-platform-jdk/>
5. <https://www.javatpoint.com/difference-between-jdk-jre-and-jvm>

<http://mdita.metropolitan.ac.rs/qdita-temp/KT103/L01/KT103-L01-pptlc4.html>

1. <https://sr.wikipedia.org/wiki/NetBeans>
2. <https://netbeans.org/about/history.html>
3. [https://www.javatpoint.com/intellij-vs-eclipse.](https://www.javatpoint.com/intellij-vs%20eclipse#:~:text=The%20more%20plugins%20installed%20in,smoother%20as%20compared%20to%20Eclipse.)
4. <https://hackr.io/blog/best-java-ides>
5. <https://www.bluej.org/about.html>
6. <https://www.oracle.com/application-development/technologies/jdeveloper.html>
7. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf>

1. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> [↑](#footnote-ref-1)
2. <javatpoint.com/features-of-java> [↑](#footnote-ref-2)
3. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> [↑](#footnote-ref-3)
4. <https://hr.wikipedia.org/wiki/Java_(programski_jezik)> [↑](#footnote-ref-4)
5. <http://docplayer.rs/184548646-Programiranje-nastava-prof-dr-sc-dra%C5%BEena-ga%C5%A1par-datum.html> [↑](#footnote-ref-5)
6. <https://medium.com/@rahul77349/difference-between-compiler-and-interpreter-with-respect-to-jvm-java-virtual-machine-and-pvm-22fc77ae0eb7> [↑](#footnote-ref-6)
7. <https://medium.com/@rahul77349/difference-between-compiler-and-interpreter-with-respect-to-jvm-java-virtual-machine-and-pvm-22fc77ae0eb7> [↑](#footnote-ref-7)
8. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> [↑](#footnote-ref-8)
9. <https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf> [↑](#footnote-ref-9)
10. <https://www.javatpoint.com/features-of-java> [↑](#footnote-ref-10)
11. <https://www.topperskills.com/tutorials/java/features-properties-java-language.html> [↑](#footnote-ref-11)
12. <https://www.javatpoint.com/features-of-java> [↑](#footnote-ref-12)
13. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> [↑](#footnote-ref-13)
14. Jaworski, J. (1998): Java 1.2: Unleashed. 4th Edition. Macmillan Computer Publishing, Indianapolis, IN, 1998 [↑](#footnote-ref-14)
15. Hunter, J. i Crawford W. (1998). Java Servlet Programming. O'Reilly and Associates, Inc., Sebastopol, CA. [↑](#footnote-ref-15)
16. <https://www.topperskills.com/tutorials/java/features-properties-java-language.html> [↑](#footnote-ref-16)
17. <https://www.weblineindia.com/blog/java-development-advantages-disadvantages/> [↑](#footnote-ref-17)
18. <https://www.weblineindia.com/blog/java-development-advantages-disadvantages/> [↑](#footnote-ref-18)
19. <https://www.weblineindia.com/blog/java-development-advantages-disadvantages/> [↑](#footnote-ref-19)
20. <http://laris.fesb.hr/java/osnovni_program.htm> [↑](#footnote-ref-20)
21. [https://docs.oracle.com/en/java/javase/15/docs/specs/sealed-classes-jls.html#jls-3.9](https://docs.oracle.com/en/java/javase/15/docs/specs/sealed-classes-jls.html%23jls-3.9) [↑](#footnote-ref-21)
22. [https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html#jls-3.9](https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html%23jls-3.9) [↑](#footnote-ref-22)
23. <http://laris.fesb.hr/java/varijable.htm> [↑](#footnote-ref-23)
24. <https://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/3.%20Klase_i_objekti.pdf> [↑](#footnote-ref-24)
25. <https://javaprogramiranje.wordpress.com/2009/11/07/upravljacke-naredbe/> [↑](#footnote-ref-25)
26. Ibid [↑](#footnote-ref-26)
27. Ibid [↑](#footnote-ref-27)
28. Ibid [↑](#footnote-ref-28)
29. Ibid [↑](#footnote-ref-29)
30. Burd B.(2017): Beginning Programming with Java For Dummies. IDG Books Worldwide, Inc. Foster City, California. [↑](#footnote-ref-30)
31. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> [↑](#footnote-ref-31)
32. <https://www.softwaretestinghelp.com/java-components-java-platform-jdk/> [↑](#footnote-ref-32)
33. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> [↑](#footnote-ref-33)
34. <https://www.javatpoint.com/difference-between-jdk-jre-and-jvm> [↑](#footnote-ref-34)
35. Ibid [↑](#footnote-ref-35)
36. <http://mdita.metropolitan.ac.rs/qdita-temp/KT103/L01/KT103-L01-pptlc4.html> [↑](#footnote-ref-36)
37. <https://sr.wikipedia.org/wiki/NetBeans> [↑](#footnote-ref-37)
38. <https://netbeans.org/about/history.html> [↑](#footnote-ref-38)
39. <https://hackr.io/blog/best-java-ides> [↑](#footnote-ref-39)
40. https://www.javatpoint.com/intellij-vs-eclipse [↑](#footnote-ref-40)
41. https://www.javatpoint.com/intellij-vs-eclipse [↑](#footnote-ref-41)
42. <https://hackr.io/blog/best-java-ides> [↑](#footnote-ref-42)
43. Ibid [↑](#footnote-ref-43)
44. Ibid [↑](#footnote-ref-44)
45. <https://www.bluej.org/about.html> [↑](#footnote-ref-45)
46. <https://hackr.io/blog/best-java-ides> [↑](#footnote-ref-46)
47. <https://www.oracle.com/application-development/technologies/jdeveloper.html> [↑](#footnote-ref-47)
48. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf> [↑](#footnote-ref-48)
49. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf> [↑](#footnote-ref-49)