**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, listopad 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](#_Toc69250367)

[2. PROGRAMSKI JEZIK JAVA 3](#_Toc69250368)

[2.1. Osnovne karakteristike programskog jezika Java 4](#_Toc69250369)

[2.2. Prednosti i nedostaci programskog jezika Java 7](#_Toc69250370)

[2.3. Naredbe u Java-i 9](#_Toc69250371)

[2.4. Java platforma 18](#_Toc69250372)

[2.5. Integrirano razvojno okruženje NetBeans 20](#_Toc69250373)

[2.6. Ostala razvojna okruženja za Java-u 23](#_Toc69250374)

[2.7. MVC Arhitektura 25](#_Toc69250375)

[3. JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA 27](#_Toc69250376)

[3.1. Opis problema 28](#_Toc69250377)

[3.2. Relacijski model baze podataka 29](#_Toc69250378)

[3.3. Dijagram klasa 30](#_Toc69250379)

[3.4. Prozori aplikacije 32](#_Toc69250380)

[3.4.1. Prijava 32](#_Toc69250381)

[3.4.2. Evidencija vremena - Administrator 35](#_Toc69250382)

[3.4.3. Pregled korisnika- Administrator 37](#_Toc69250383)

[3.4.4. Dodaj korisnika - Administrator 37](#_Toc69250384)

[3.4.5. Izmjeni korisnika - Administrator 40](#_Toc69250385)

[3.4.6. Početna – Korisnik 42](#_Toc69250386)

[3.4.7. Evidencija vremena - Korisnik 45](#_Toc69250387)

[3.4.8. Postavke - Korisnik 45](#_Toc69250388)

[3.5. MVC arhitektura u aplikaciji 47](#_Toc69250389)

[3.6. Slučajevi korištenja 53](#_Toc69250390)

[3.7. Mogućnosti proširenja aplikacije 55](#_Toc69250391)

[4. ZAKLJUČAK 57](#_Toc69250392)

[5. LITERATURA 58](#_Toc69250393)

**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 je implementacija funkcionalne desktop aplikacije za evidenciju radnog vremena uz pomoć programskog jezika Java u integriranom razvojnom okruženju Netbeans.

Za kreiranje i oblikovanje dizajna aplikacije korištena je aplikacija JavaFX Scene Builder 2.0 koja 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 (engl. Model-View-Controler, Model-Pogled-Upravitelj) obrascu software-ske arhitekture (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, provjere dolaska zaposlenih, provjere intenziteta i obujma posla i sl.

Kroz povijest su se zaposlenici susretali s različitim 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 na gumb koji označava početak radnog vremena,a aplikacija će im (u realnom vremenu) odbrojavati vrijeme provedeno na poslu.

Također, zamišljeno je 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.

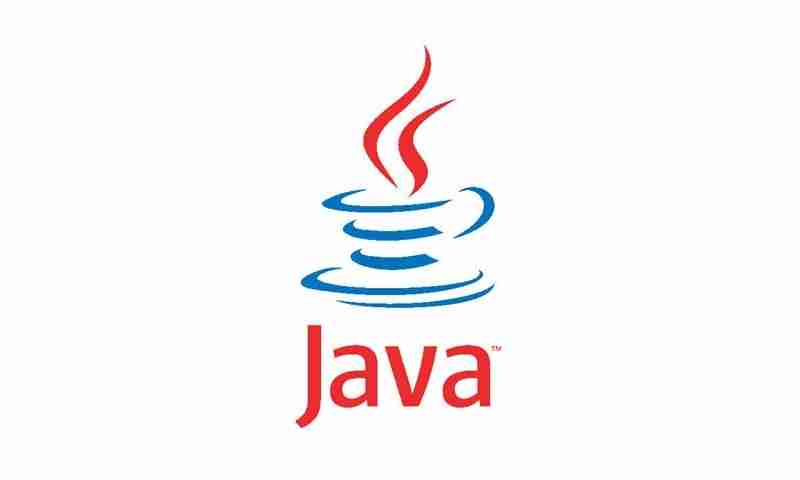
U drugom dijelu rada opisan je problem koji rješava Java desktop aplikacija razvijena u razvojnom okruženju Netbeans, relacijski model baze podataka, dijagram klasa te prozori aplikacije. Također su u posebnom odlomku drugoga dijela navedeni i objašnjeni paketi koji se nalaze unutar same aplikacije, odnosno paketi koji podrazumijevaju modele, poglede i upravitelje. U zasebnom su odlomku uz pomoć dijagrama slučajeva korištenja (engl. Use case diagrams) 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 jedan je 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 razvijen je 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 kasnije je 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. Logo Java-e

Prvi se Java-in program 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.

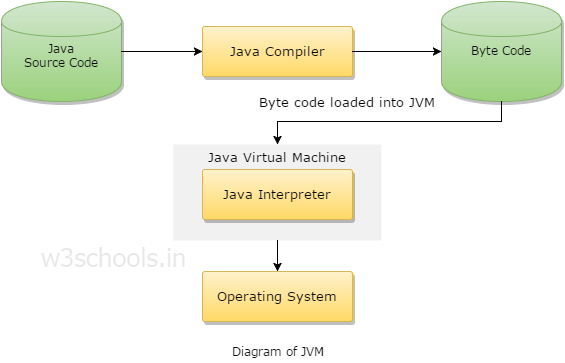
Zanimljiva činjenica je kako se donedavno Java nije toliko koristila u razvoju Web poslužitelja. Međutim, s poboljšanjima u sučeljima za programiranje aplikacija (engl. Application programming interface, skraćeno API), a posebice Servlet API-ju, Java postaje gotovo neizostavan alat u Web programiranju.

Danas je, prema podacima GitHub-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 17) izašla u rujnu 2021. godine.

## Osnovne karakteristike programskog jezika Java

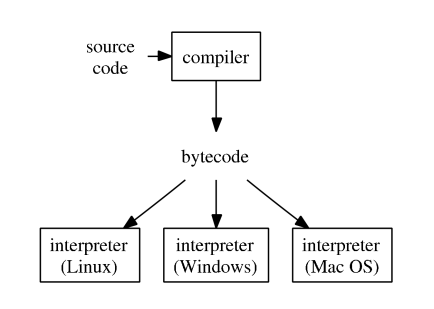
Kako bi se izvorni kod nekog programa (bilo koji niz izraza napisan u nekom od programskih jezika) mogao izvršiti na odgovarajućoj računalnoj platformi potrebno ga je najprije prevesti u jedini jezik kojeg računala mogu razumjeti- strojni programski jezik, odnosno jezik nula i jedinica. Takvo transformiranje 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 se jezici mogu podijeliti u dvije skupine; prevedene (kompilirane) i intepretirane programske jezike. Programski jezik Java 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 svoju najznačajniju 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 izvornog koda napisanog u Java-i 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 se jezik Java temelji na klasama i u njemu su obavezni objektno orijentirani principi. Sav izvorni kod se u Java-i 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 najsličnije su 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 izvršavaju.[[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. Pored navedenih postoji još 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 reći kako je 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 prilikom korištenja Java-e.[[13]](#footnote-13)

Također, u Java-i je moguće 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 u Java-i omogućava izvršavanje koda tijekom samog prevođenja i na taj način se ostvaruju visoke performanse samog programskog jezika.[[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 (engl. 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-e. 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.

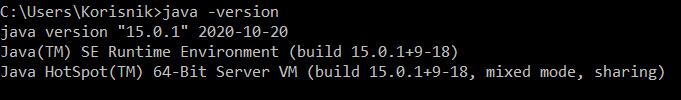
## Naredbe u Java-i

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 biti će odbačeni uz poruku o grešci.[[20]](#footnote-20)

U programskom jeziku Java, naredba (ključna riječ) je bilo koja od 52 ključnih riječi koje su rezervirane i imaju unaprijed definirano značenje.[[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)

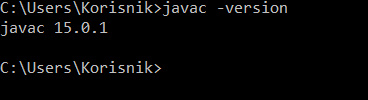
U ovom su poglavlju najprije objašnjene neke od osnovnih naredba u Java-i koje je moguće unijeti i uz pomoć tekstualnog sučelja. Nakon tih naredbi su u nastavku objašnjene naredbe i ključne riječi koje se koriste prilikom kodiranja u integriranim razvojnim okruženjima:

**Java –version** (verzija): Ovo je jedna od najosnovnijih naredbi u Java-i 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.



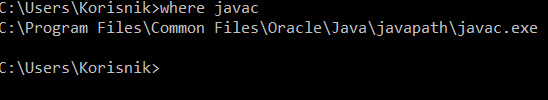
Slika 4. Primjer unošenja naredbe Java –version u terminal

**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.



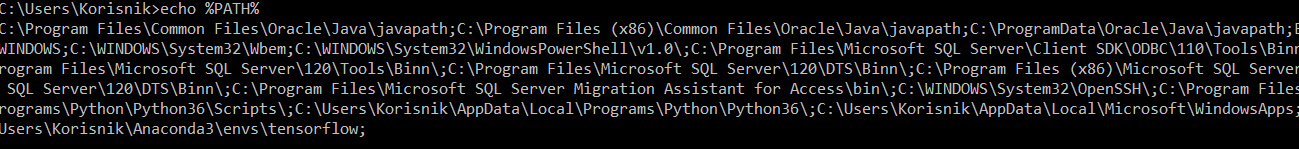
Slika 6.Upotreba naredbe Where u CMD-u

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



Slika 7. Izvršavanje programa

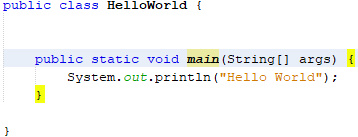
**Echo:** Ova naredba se ne koristi samo u Java-i, nego je naredba koju se mora poznavati jer pomaže u prikazivanju podataka određene datoteke. Obično se koristi za provjeru PATH varijabli.



Slika 8. Naredba Echo

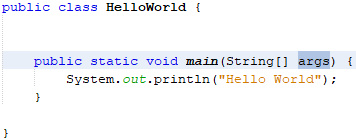
Nakon što su navedene osnovne naredbe koje je moguće unijeti uz pomoć tekstualnog sučelja, slijede primjeri i objašnjenja ostalih naredbi, odnosno ključnih riječi koje se koriste u integriranim razvojnim okruženjima prilikom kodiranja:

**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 koriste se sljedeće naredbe:

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

intbrojac=0; Varijabli brojacpridružujemo 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:

floatprosjek\_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:

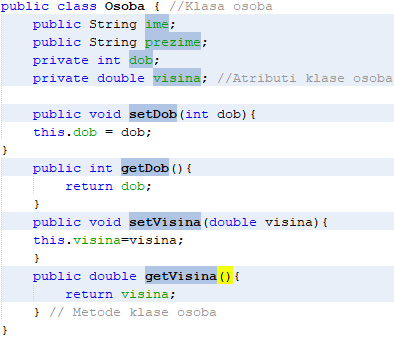
charznak=Y;

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:

booleancorona\_test= false;

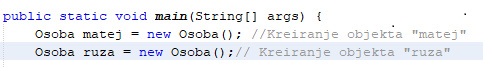
Važno je napomenuti kako svi navedeni tipovi podatakaspadaju u skupinu tzv. primitivnih (jednostavnih) tipova podataka. Pored jednostavnih u Java-i postoje i složeni tipovi podataka, a to su polja, klase i sučelja. Za deklariranje polja u Java-i se koristi ključna riječ **Array**, za deklariranje klasa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



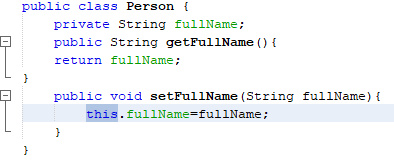
Slika 11. Atributi i metode klase Osoba

**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.



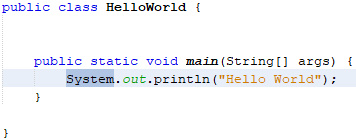
Slika 12. Primjer kreiranja objekata u Java-i

**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.



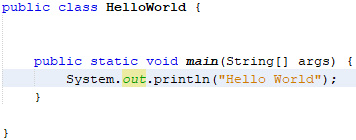
Slika 13. Ključna riječ this

**System:**System 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. Pozivanje metode println

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



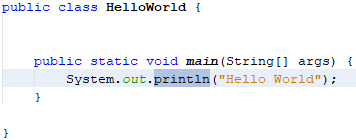
Slika 15. Out- statički član klase System

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



Slika 16. Metoda print

**Println:**Metoda println 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 i do-while)
  3. naredbe skoka (break, continue ireturn)

Java podržava dvije naredbe uvjetovanja i to **if** i **switch**. Naredbom if program se 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:

intbrojac=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 s radom, 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 čineJava platformu jesu:[[32]](#footnote-32)

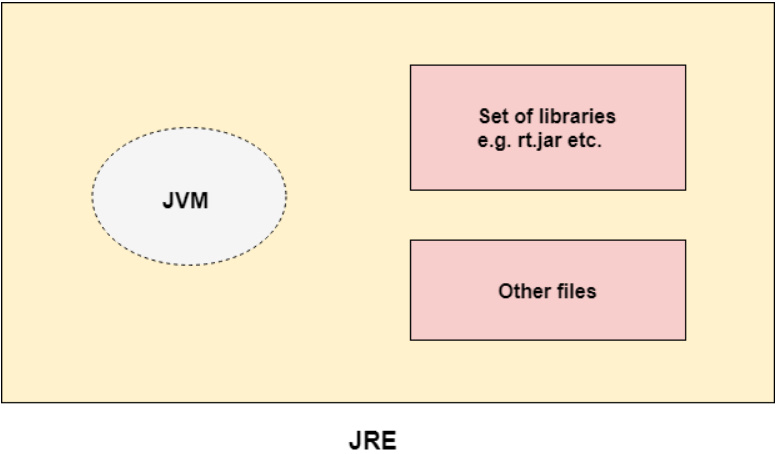
1. Java programski jezik
2. Java razvojni komplet (JDK – Java Development Kit)
3. Java okruženje za izvršavanje (JRE- Java RuntimeEnviorment)
4. Java prevoditelj
5. Java virtualni stroj (JVM- Java VirtualMachine)

Kako bi se stekla jasnija slika o cjelokupnoj Java platformiod 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 kakose 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)

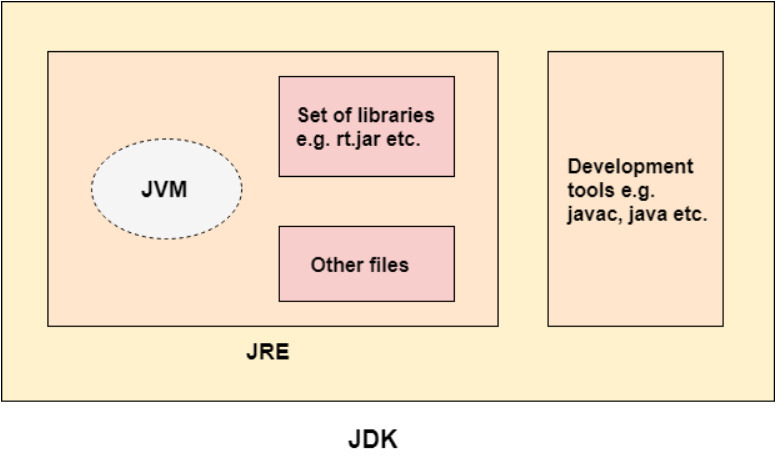
1. Učitavanje koda
2. Verifikacija koda
3. Izvršavanje koda
4. Pružanjeokruženja za izvođenje

Nadalje, neizostavan dio Java platforme jeste Java-ino okruženje za izvršavanje odnosno **JRE**(također se rabi skraćenica RTE-RunTimeEnviorment).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 RuntimeEnviorment (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 dovoljan je 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 nazivaju se 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)

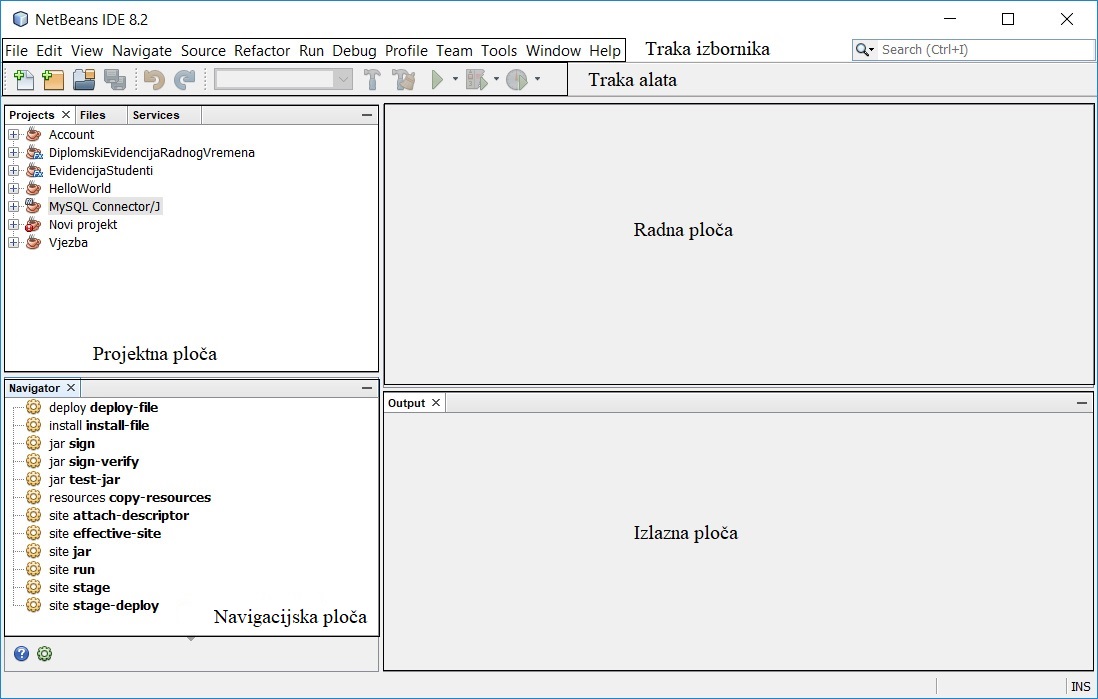
* Uređivanje koda, završavanje koda skraćenicama i prečacima, sintaksno naglašavanje, pomoć (hints)
* Pretraživanje i refaktoriranje koda
* Prevođenje i izvršavanje koda
* Čarobnjaci (Wizards) koji olakšavaju komplicirane zadatke
* Kreiranje izvršnih i instalacijskih paketa

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 NetBeansmože 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 bio je 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-atada je 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 NetBeansIDE. 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 uređivačuvlači 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 naraspolaganju 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 biti ćenavedeni i opisanijoš 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)

* CommunityEdition (Izdanje zajednice): Dostupno je besplatno i uglavnom ga programeri koriste za Java-u i Android.
* UltimateEdition (Komercijalno izdanje): Ultimateedition 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 (ContextsandDepedencyInjection), 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 se može 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**jetakođer jedan od tri velika Java IDE-a. Moderno integrirano razvojno okruženje dostupno je u desktop i cloud (oblak) izdanju. Oblak izdanjeEclipse-a, nazvanoEclipseChe, 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-asu opremljena potrebnom / dodatnom funkcionalnošću pomoću dodataka teEclipse imamnoštvo 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-iniprogrameri 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- BlueJnamjerno 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 sBlueJ-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 upravoBlueJ-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)

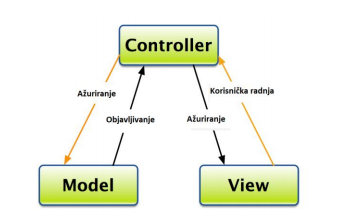
Oracle JDeveloperse može integrirati s Oracle-ovim okvirom za razvoj aplikacija (Oracle ADF) kako bi se dodatno pojednostavio razvoj aplikacija. Pored Java-e, JDeveloper se može koristiti i za razvoj aplikacija u GTML-u, JavaScript-u, PHP-u, SQL-u i XML-u.[[47]](#footnote-47)

## MVC Arhitektura

Model-prikaz-upravitelj (obično poznat kao MVC)je software-skiobrazac 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 ( engl. GraphicalUserInterface, GUI) 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 21).[[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 razvojasvake 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 (engl. Test-Driven Development - TDD), jer je mogućeodvojeno testirati pojedine komponente aplikacije i nije uvijek za testiranje potrebno koristiti web poslužitelj (neke komponente se mogu testirati lokalno)
* Olakšano procesiranje zahtjeva koje šalje korisnik, jer se za tu svrhu koristi tzv.FrontController (prednji upravitelj) koji kroz jedan upravitelj šalje sve zahtjeve korisnika.

# JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA

Aplikacija za evidenciju radnog vremena razvijena je u programskom jeziku Java korištenjem JavaFX grafičkog sučelja. Aplikacije s grafičkim sučeljem (engl. GraphicalUserInterface, GUI) omogućavaju korištenje svojih funkcionalnosti putem grafičkih elemenata koji se nazivaju kontrole (engl. Widgets). JavaFX omogućava dizajniranje grafičkog sučelja aplikacije koji može biti korišten odvojeno ili integriran unutar razvojnog okruženja. Također, omogućava korištenje CSS-a (engl. CascadingStyleSheet) za oblikovanje, odnosno stiliziranje komponenti grafičkog sučelja. Aplikacija za evidenciju radnog vremena razvijena je unutar NetBeans razvojnog okruženja. Grafičko sučelje dizajnirano je korištenjem Scene Builder aplikacije.

Aplikacijom Scene Builder moguće je dizajnirati grafičko sučelje korištenjem posebno oblikovane XML (engl. eXtensibleMarkupLanguage) datoteke s ekstenzijom „.fxml“ (FXMarkupLanguage). Korištenjem navedene aplikacije moguće je odvojiti prezentacijsku od programske logike unutar JavaFX aplikacija.

Baza podataka korištena u izradi aplikacije je MySQL baza podataka. Navedena baza je otvorenog koda (engl. Open source) te pripada sustavima za upravljanje relacijskim bazama podataka (engl. RelationalDatabase Management System, RDBMS).

Operacijski sustav na kojem je razvijena aplikacija za evidenciju radnog vremena je Windows 10. Aplikaciju je moguće pokrenuti na bilo kojem uređaju koji ima instaliran Java virtualni stroj.

## Opis problema

Problem koji aplikacija za evidenciju radnog vremena rješava može se zaključiti iz naziva same aplikacije – evidentiranje radnog vremena zaposlenika u nekoj zamišljenoj organizaciji. S navedenom problematikom brojne su se tvrtke susretale dugi niz godina, ponajviše one s velikim brojem zaposlenika. U takvim tvrtkama, koje broje i po nekoliko stotina zaposlenika u poslovnim prostorima, gotovo je nemoguće voditi evidenciju dolaska i odlaska pojedinog zaposlenika s posla, kao i njegovo kretanje unutar radnog vremena. Upisivanje evidencije na papirnati medij bilo bi neučinkovito te bi oduzimalo mnogo vremena zaposlenicima i njihovim nadređenima. Također, korištenje papirnatog medija šteti svijetu na ekološkoj razini. Danas, gotovo da ne postoji organizacija s većim brojem zaposlenika koja nije na sličan način riješila problem evidencije zaposlenika. Jedna od takvih institucija je i Sabor Republike Hrvatske gdje je implementiran sustav za provjeru dolaska zaposlenika.

Aplikacija koja je predmet istraživanja ovog diplomskog rada rješava navedene i opisane probleme evidencije radnih vremena zaposlenika u nekoj zamišljenoj organizaciji, instituciji ili udruženju. Aplikacija omogućava zaposlenicima jednostavno unošenje podataka o radnom vremenu korištenjem računala zaposlenika.

Sučelje aplikacije za evidenciju radnog vremena razlikuje se ovisno o ulozi korisnika, odnosno dodijeljenoj roli. Aplikacija ima dva tipa korisnika, a to su Admin i User. Korisnik tipa Adminima najviše prava na desktop aplikaciji. Radnje koje mogu vršiti korisnici ovisno o tipu, odnosno dodijeljenoj ulozi, prikazane su u tablici 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Redni broj.** | **Radnja** | **Admin** | **User** |
| 1. | Prijava | + | + |
| 2. | Unos radnog statusa | - | + |
| 3. | Pregled svojih radnih statusa | - | + |
| 4. | Pregled svih radnih statusa | + | - |
| 5. | Unos korisnika | + | - |
| 6. | Brisanje korisnika | + | - |
| 7. | Uređivanje korisnika | + | - |
| 8. | Postavke | - | + |
| 9. | Odjava iz aplikacije | + | + |

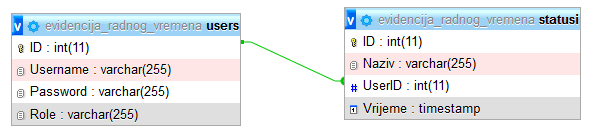
**Tablica 1. Radnje koje korisnici mogu vršiti ovisno o ulozi koju imaju**

Korisnik s ulogom Admin ima mogućnost prijave i odjave iz aplikacije, pregleda, unosa, izmjene i brisanja korisnika, odnosno zaposlenika. Također, ima mogućnost pregleda radnih statusa svih zaposlenika.

Korisnik s ulogom User ima mogućnost prijave i odjave iz aplikacije, unosa radnog statusa te pregled vlastite evidencije radnog vremena.

## Relacijski model baze podataka

Prilikom implementacije korištene su dvije tablice unutar baze podataka – tablica pod nazivom *Users* i tablica *Statusi*. Tablica *Users* sadrži podatke o svim korisnicima unutar sustava, dok tablica *Statusi* služi za spremanje statusa i vremena pojedinog korisnika. Relacijski model baze podataka prikazan je na slici 22.



Slika 22. Relacijski model baze podataka

Tablica *Statusi* povezana je s tablicom *Users* relacijom jedan-na-više pomoću stranog ključa. Popis tablica, uključujući primarne i strane ključeve prikazan je u tablicama 2 i 3.

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet** | **Kratki opis atributa** |
| ID | PK, NOT NULL | Identifikator korisnika |
| Username | NOT NULL | Korisničko ime |
| Password | NOT NULL | Lozinka korisnika |
| Role | NOT NULL | Uloga korisnika |

Tablica 2. - Opis tablice Users

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet** | **Kratki opis atributa** |
| ID | PK, NOT NULL | Identifikator statusa |
| Naziv | NOT NULL | Naziv statusa |
| UserID | FK, NOT NULL | Identifikator korisnika |
| Vrijeme | NOT NULL | Datum i vrijeme statusa |

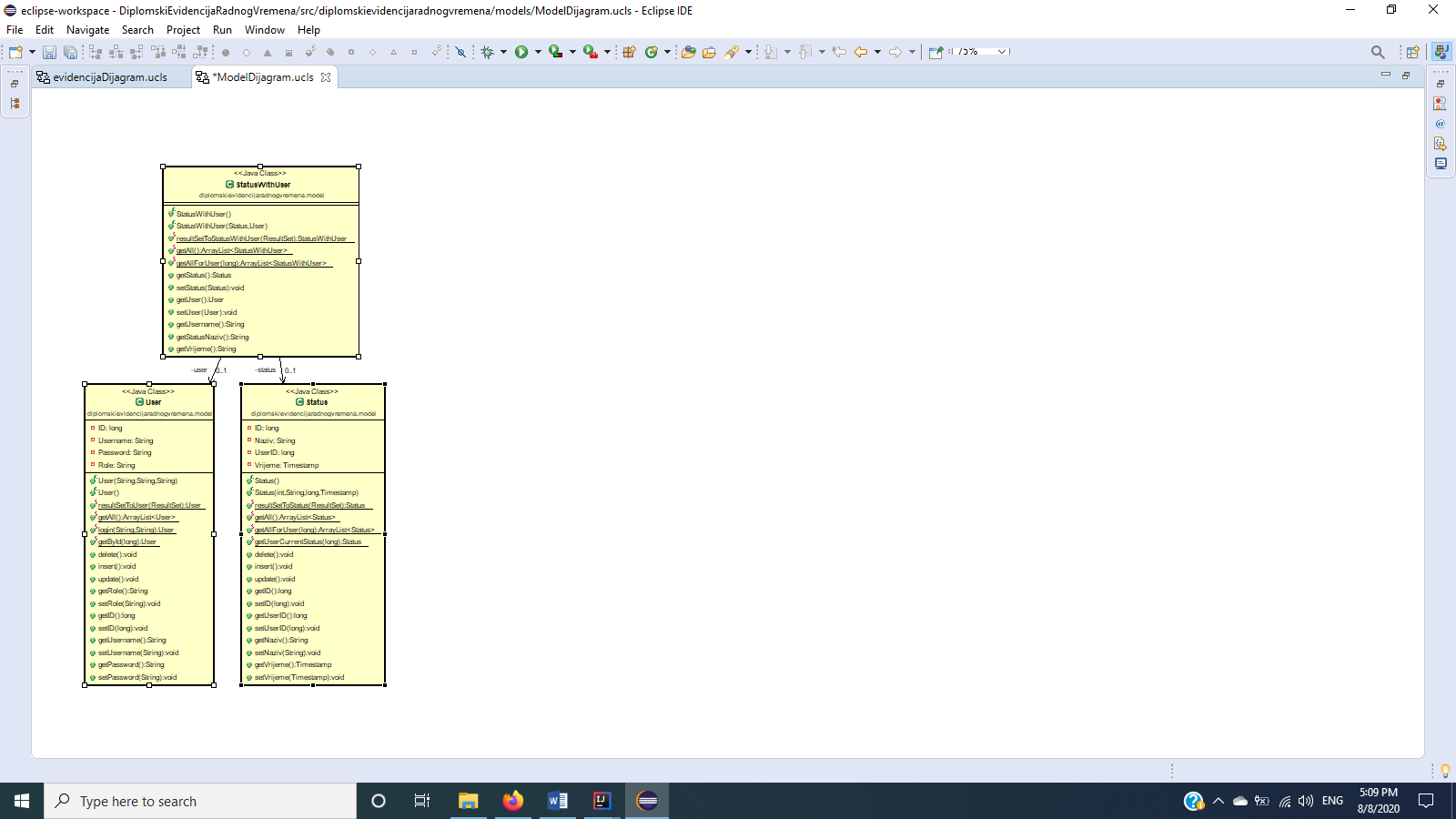
**Tablica 3. - Opis tablice Statusi**

## Dijagram klasa

Dijagram klasa vrsta je strukturnog dijagrama koji prikazuje strukturu sustava kroz klase, objašnjavajući njihove atribute i međusobne odnose.

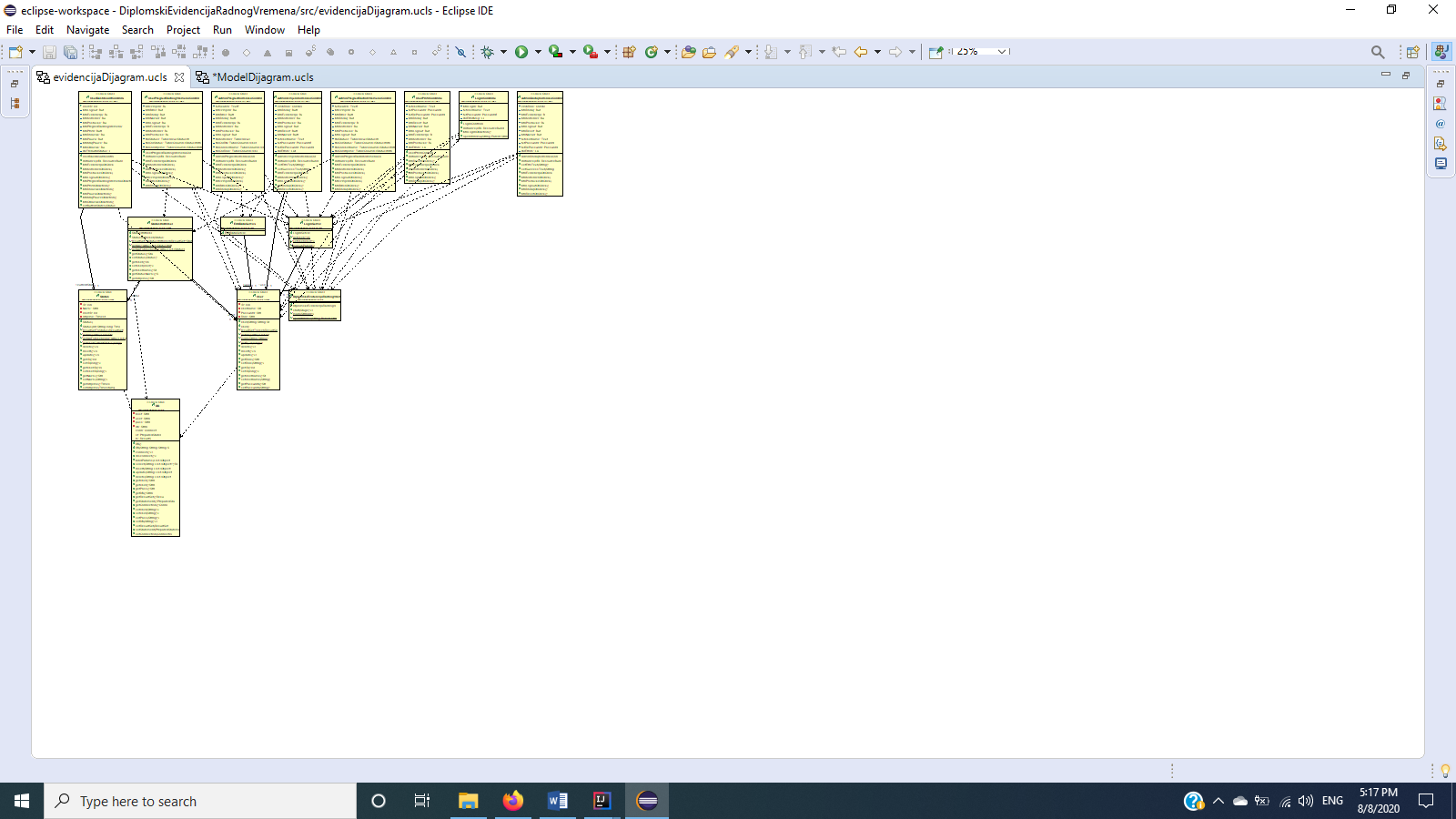
Dijagram klasa iz paketamodels posebno je izdvojen jer on predstavlja kostur same aplikacije. Na slici 23. prikazan je dijagram klasa koje se nalaze u paketu models. Sastoji se od tri klase:

* *User* – služi za upravljanje korisnicima sustava, te za njihovu interakciju s bazom podataka
* *Status* – služi za upravljanje radnim statusima korisnika sustava, te za njihovu interakciju s bazom podataka.
* *StatusWithUser* – predstavlja spoj klasa *User* i *Status*.



Slika 23. Dijagram klasa iz models paketa

Dijagram klasa cijele aplikacije nije prikazan najjasnije zbog njegove veličine koja nije jasno vidljiva na papiru formata A4. U dijagramu se nalaze sve klase korištene u ovoj aplikaciji, dakle, sve datoteke s .java ekstenzijom. U prvom redu prikazani su upravitelji, odnosno klase iz paketa controllers, dolje lijevo nalaze se klase iz paketa models, a dolje desno klase iz paketa services.



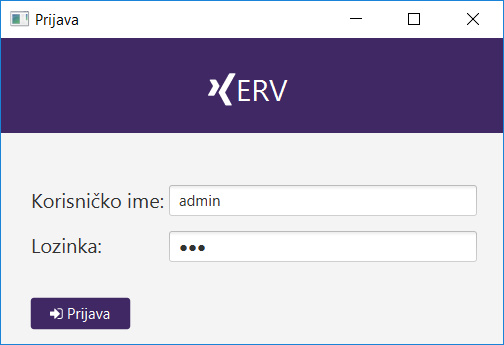
Slika 24. Dijagram klasa kompletne aplikacije

## Prozori aplikacije

U ovom poglavlju su prikazani i objašnjeni svi prozori koji se pojavljuju u aplikaciji opisani kroz pogled prijavljenog korisnika s ulogomAdmin, kao i kroz pogled korisnika s ulogomUser. Uz tekstualni opis prozora, prikazane su slike ekrana, kao i isječci koda koji su važni u radu same aplikacije.

### Prijava

Pokretanjem aplikacije otvara se prozor za prijavu u sustav. Prozor za prijavu sadrži dva polja, polje *Korisničko ime* i polje *Lozinka*. Korisnik mora unijeti korisničko ime odnosno lozinku kako bi se prijavio. Prikaz prozora za prijavu korisnika prikazan je na slici 25.



Slika 25. Prozor za prijavu korisnika

Klikom na gumb *Prijava* poziva se metoda za btnLoginOnAction().

U slučaju neispravnog unosa, na ekranu će se pojaviti jedna od sljedećih poruka:

* *Unesite korisničko ime* – ako nije ispunjena vrijednost polja *Korisničko ime*
* *Unesite lozinku* – ako nije ispunjena vrijednost polja *Lozinka*
* *Greška prilikom prijave* – ako u bazi podataka nije pronađen korisnik s unesenim korisničkim imenom i lozinkom.

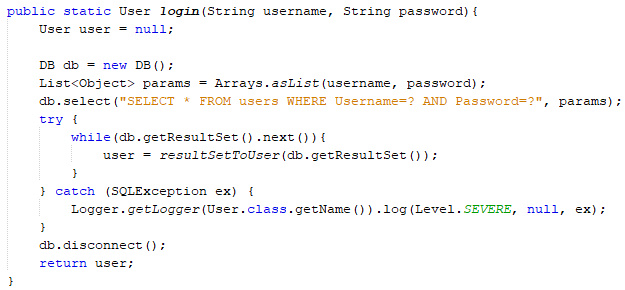
Prilikom ispravnog unosa korisničkog imena i lozinke, provjerava se korisnikova uloga, te se, temeljem dobivenog podatka, otvara odgovarajući prozor. Za korisnika s ulogom Adminotvara se prozor *AdminPregledRadnihVremena*, a za korisnika s ulogom User otvara se prozor *UserDashboard*.

Kod koji se izvršava klikom na gumb *Prijava* prikazan je na slici 26.

Slika 26. Prijava u sustav



Upit za dohvaćanje korisnika iz baze podataka izvršava se pozivom metode User.login(username, password), u kojoj se, kao argumenti, šalju korisničko ime i lozinka, uneseni od strane korisnika kroz sučelje aplikacije. Kod koji se izvršava pozivom navedene metode prikazan je na slici 27. Ostvaruje se konekcija nad bazom podataka te se izvršava jednostavan SELECT upit za dohvat korisnika. Rezultat dohvata sprema se u klasu *User*.



Slika 27. Dohvaćanje korisnika iz baze podataka

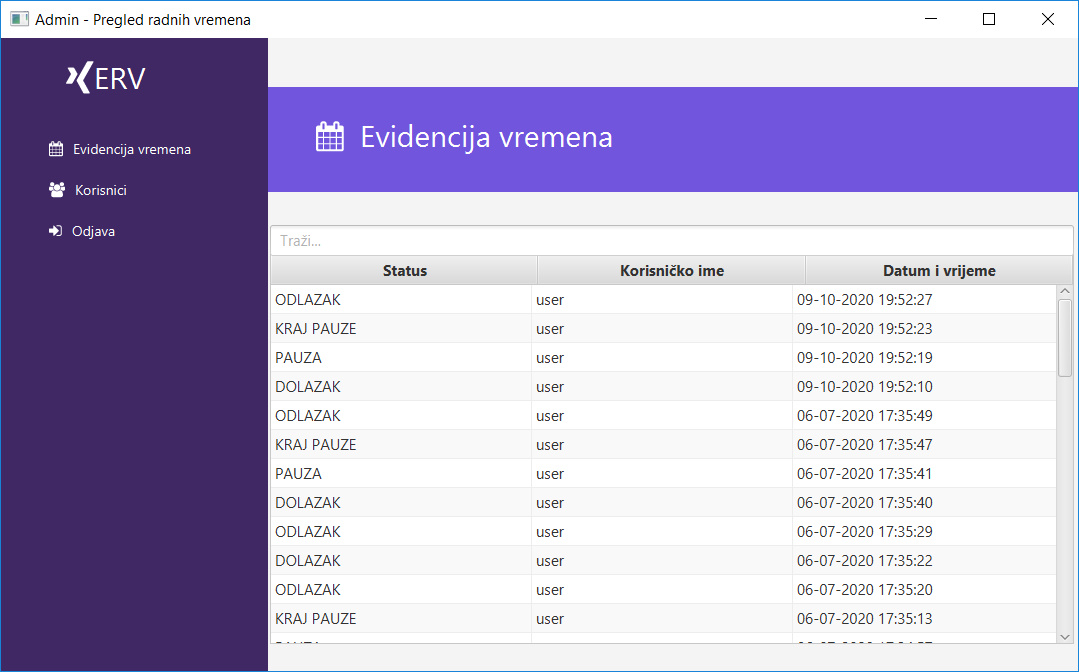
### Evidencija vremena - Administrator

Nakon uspješne prijave korisnika s Admin rolom, otvara se ekran na kojem su, kroz tablični prikaz, vidljive evidencije vremena svih korisnika u sustavu. Na ekranu za evidenciju vremena prikazana su tri stupca: status, korisničko ime te datum i vrijeme. Stupac *Status* prikazuje aktivnost koji je korisnik unio, *Korisničko ime* prikazuje ime korisnika koji za kojeg se prikazuje status, a stupac *Datum i vrijeme* prikazuje datum i vrijeme kada je aktivnost unesena u sustav.

S lijeve strane nalazi se navigacijski izbornik s tri opcije:

* *Evidencija vremena* – ekran za prikaz evidencije vremena svih korisnika
* *Korisnici* – ekran za prikaz svih korisnika uz mogućnost izmjene, brisanja ili dodavanja novih korisnika
* *Odjava* – odjava iz aplikacije

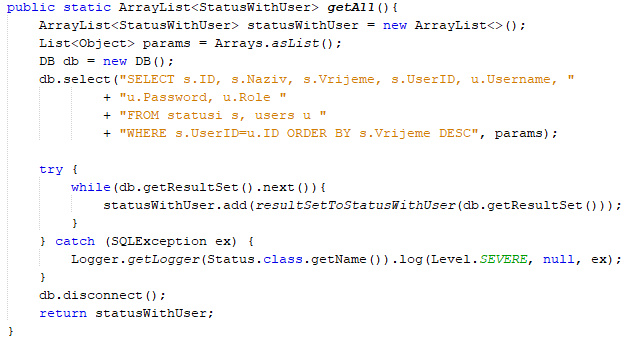
Ekran, *Evidencija vremena*, koji se otvara prilikom uspješne prijave u aplikaciju s Admin rolom prikazan je na slici 28.



Slika 28. ADMIN - Evidencija vremena

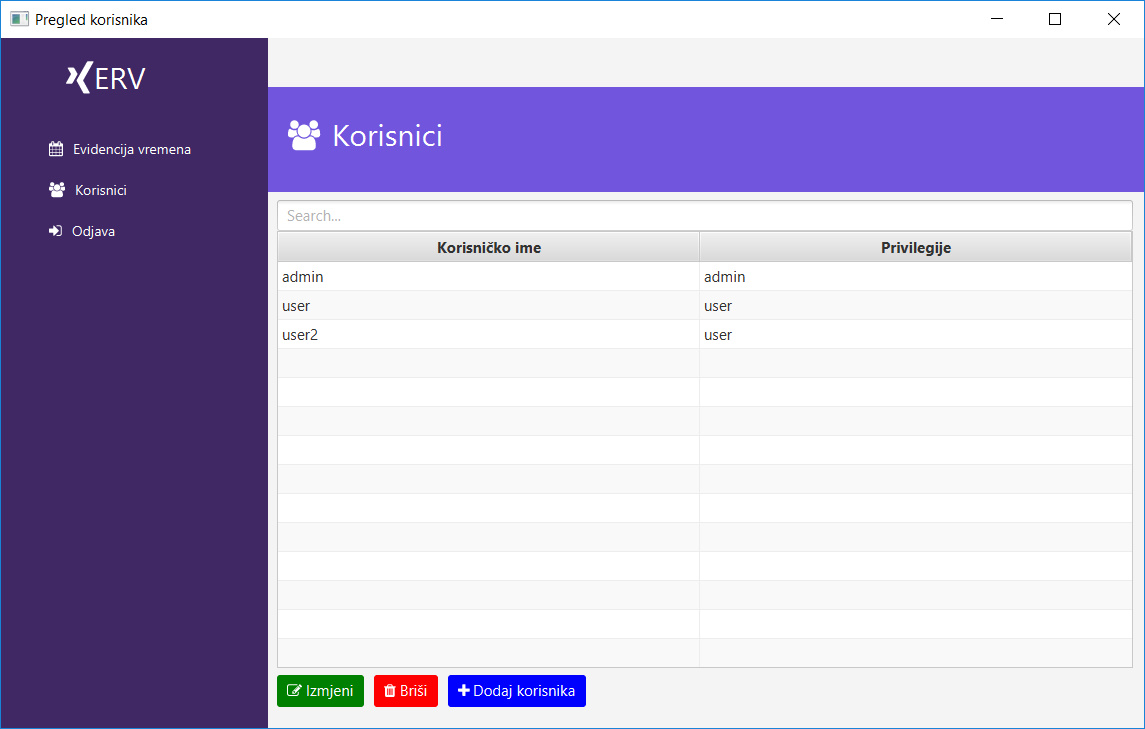
Isječak koda za dohvat svih korisnika u bazi podataka prikazan je na slici 29. Svi dohvaćeni korisnisi spremaju se u listu.

Slika 29. Spremanje dohvaćenih korisnika u listu



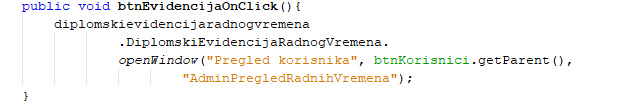
### Pregled korisnika- Administrator

Odabirom opcije *Korisnici* u navigacijskom izborniku, otvara se prozor *AdminPregledKorisnika*. Na desnom dijelu ekrana prikazana su korisnička imena i uloge svih korisnika koji su pohranjeni u bazi podataka. Na ekranu se nalaze tri gumba *Izmjeni*, *Briši* i *Dodaj korisnika* pomoću kojih korisnik s ulogomAdmin može uređivati, brisati postojeće ili dodavati nove korisnike. Izgled ekrana *Korisnici* prikazan je na slici 30.



Slika 30. ADMIN - Korisnici

Kod koji se izvršava klikom na opciju *Korisnici* prikazan je na slici 31.



Slika 31. Otvaranje ekrana "Pregled korisnika"

### Dodaj korisnika - Administrator

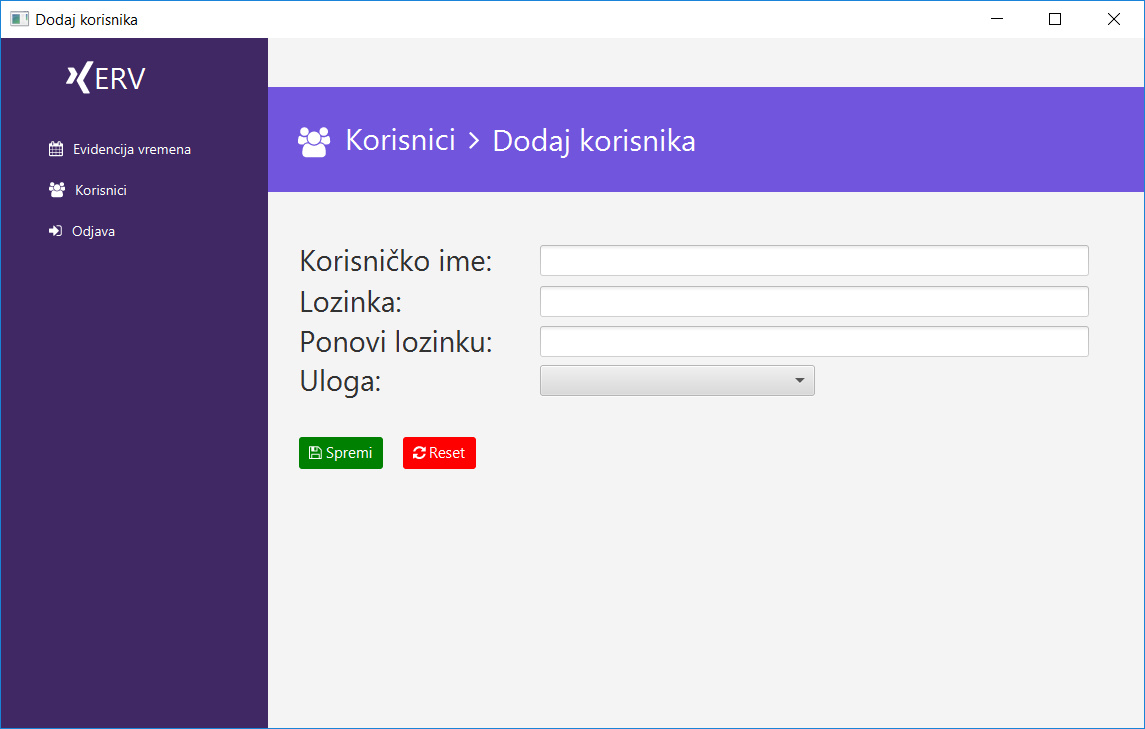
Pritiskom na gumb *Dodaj korisnika* koji se nalazi na ekranu *Korisnici*, otvara se novi prozor za unos podataka o novom korisniku. Potrebno je unijeti korisničko ime, lozinku i ponovljenu lozinku te odabrati korisnikovu ulogu, koja može biti Admin ili User. Klikom na gumb *Spremi*, podaci o unesenom korisniku spremaju se u bazu podataka. Klikom na gumb *Reset* brišu se unesene vrijednosti iz svih polja.

U slučaju neispravnog unosa, na ekranu će se pojaviti jedna od sljedećih poruka:

* *Unesite korisničko ime* – kada nije ispunjena vrijednost polja *Korisničko ime*
* *Unesite lozinku* – kada nije ispunjena vrijednost polja *Lozinka*
* *Ponovite lozinku* – kada nije ispunjena vrijednost polja *Ponovi lozinku*
* *Lozinke se ne podudaraju* – ako vrijednosti polja *Lozinka* i *Ponovi lozinku* nisu jednake

Ako je korisnik uspješno usnen, na ekranu će se pojaviti poruka *Korisnik uspješno unesen!*.

Ekran za dodavanje novog korisnika prikazan je na slici 32.



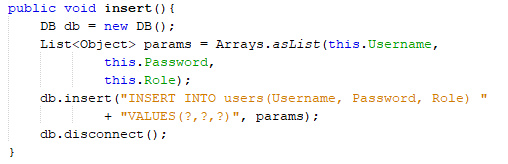
Slika 32. ADMIN - Dodaj korisnika

Kod koji se izvršava klikom na gumb *Spremi* prikazan je na slici 33.

Slika 33. Spremanje korisnika



Metoda za unos korisnika u bazu podataka. Unutar metode kreira se SQL (engl. Structured Query Language) upit za unos podataka u bazu – INSERTINTO. Koristi se objekt klase PreparedStatement kako upit ne bi direktno kreirao s korisničkim unosima. Također, kreirani upit ne prevodi se više puta već samo jednom. Moguće je korištenje promjenjivih parametara koji se označavaju znakom *?* i na taj način omogućava se višestruko izvođenje istog upita s drugim parametrima.



Slika 34. Spremanje korisnika u bazu podataka

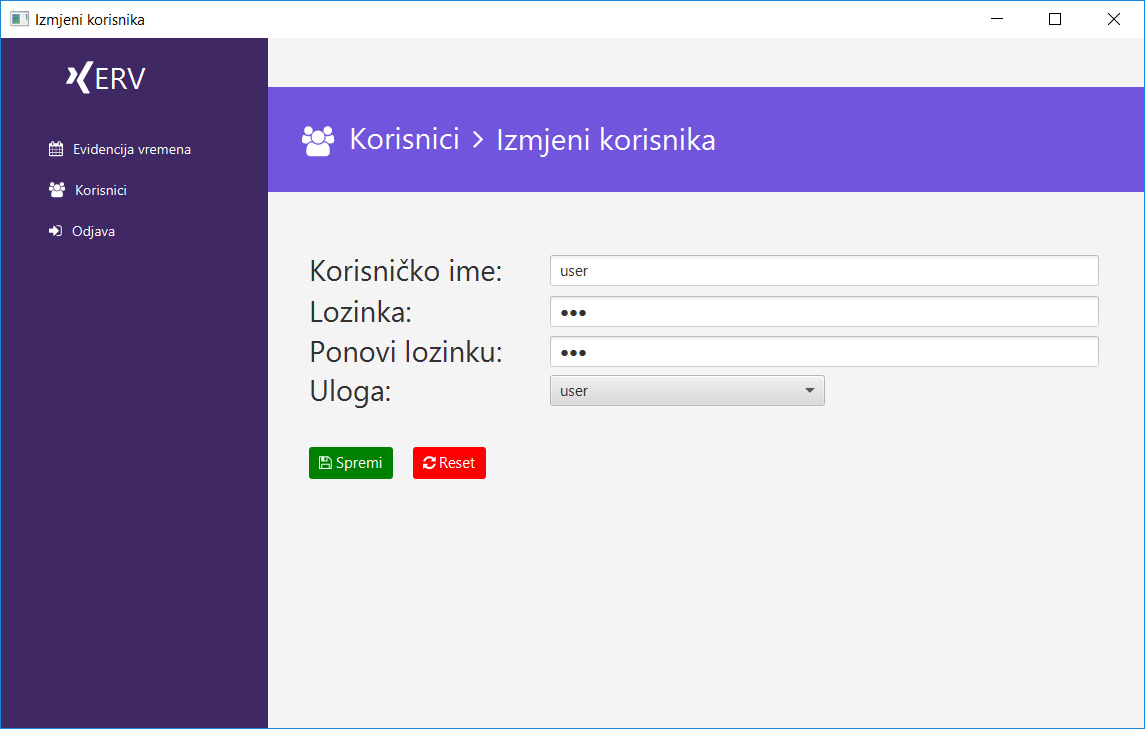
### Izmjeni korisnika - Administrator

Podatke o postojećim korisnicima moguće je izmijeniti pritiskom na gumb *Izmjeni korisnika* koji se nalazi na ekranu *Korisnici*. Važno je napomenuti da korisnik prvo mora, iz tablice s korisnicima, odabrati korisnika čije podatke želi izmijeniti. Klikom na gumb *Izmijeni* otvara se novi prozor koji sadrži podatke o korisničko imenu, lozinki i ulozi korisnika kojem se žele izmijeniti podaci. Klikom na gumb *Spremi* poziva se osluškivač događaja dodijeljen za akciju pritiska na tom gumbu. Klikom na gumb *Reset* uklanjaju se svi podaci iz polja za unos.

U slučaju neispravnog unosa, na ekranu će se pojaviti jedna od sljedećih poruka:

* *Unesite korisničko ime* – ako nije ispunjena vrijednost polja *Korisničko ime*
* *Unesite lozinku* – ako nije ispunjena vrijednost polja *Lozinka*
* *Ponovite lozinku* – ako nije ispunjena vrijednost polja *Ponovi lozinku*
* *Lozinke se ne podudaraju* – ako vrijednosti polja *Lozinka* i *Ponovi lozinku* nisu jednake

Ako je korisnik uspješno izmijenjen, na ekranu će se pojaviti poruka *Izmjene uspješno spremljene!*. Izgled ekrana za izmjenu korisnika prikazan je na slici ispod.



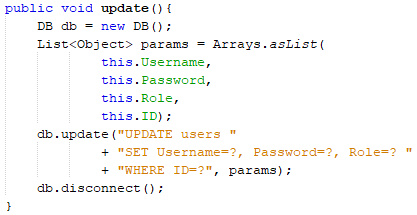
Slika 35. ADMIN - Izmjeni korisnika

Kod koji se izvršava klikom na gumb *Spremi* prikazan je u na slici 36.



Slika 36. Izmjena korisnika

Nakon što su sva unesena polja uspješno provjerena, poziva se metoda update() koja se nalazi u klasi *User* te pomoću koje se ažuriraju podaci o korisniku. Kod koji se izvršava pozivom update() metode prikazan je ispod. Unutar metode kreira se SQL upit za ažuriranje koristeći ključnu riječ UPDATE.



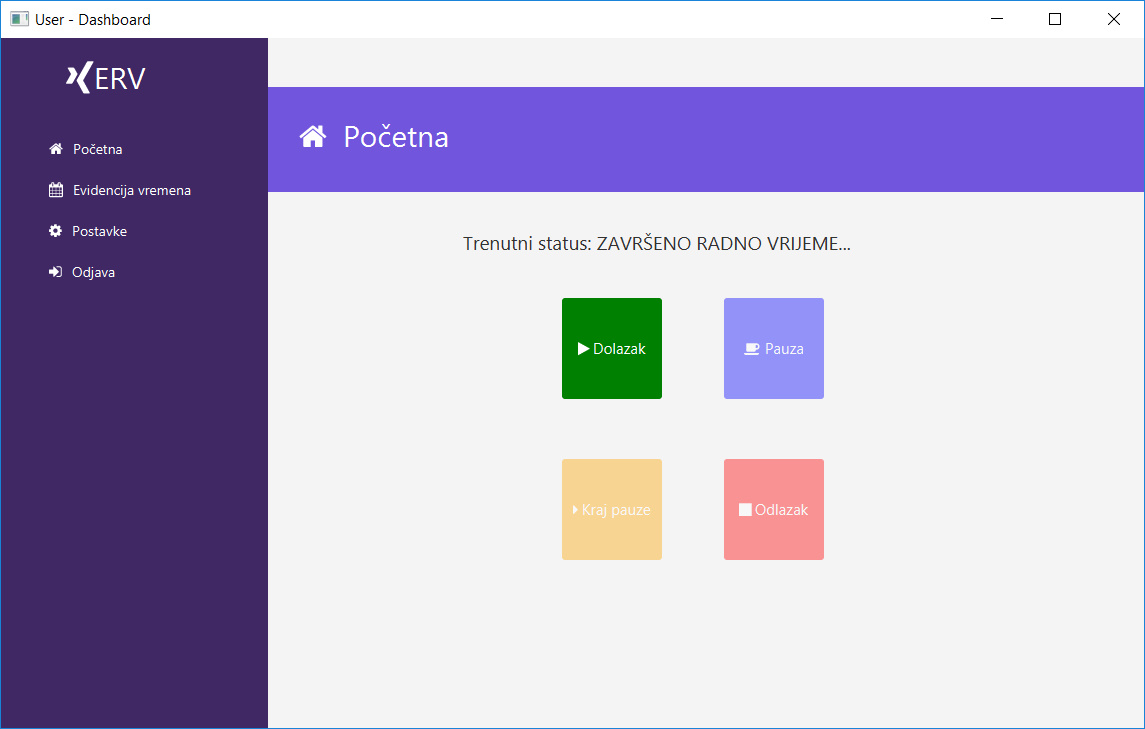
Slika 37. Izmjena korisnika u bazi podataka

### Početna – Korisnik

Prijavom korisnika s ulogom User, otvara se početna stranica koja se razlikuje od početne stranice korisnika s ulogom Admin. Na početnom ekranu prikazana su četiri gumba za evidenciju vremena: *Dolazak*, *Pauza*, *Kraj pauze*, *Odlazak*. Početni ekran koji se pojavljuje nakon prijave korisnika s rolom USER prikazan je na slici ispod. Klikom na gumb *Dolazak*, korisnik unosi aktivnost koja označava dolazak na posao. Klikom na gumb *Pauza* korisnik unosi aktivnost početka korištenja pauze. Klikom na gumb *Kraj pauze* korisnik unosi vrijeme završetka pauze i povratka na posao. Gumb *Odlazak* potrebno je pritisnuti pri završetku radnog vremena, odnosno pri odlasku s posla. Važno je napomenuti kako se gumbovi međusobno isključuju, primjerice, nije moguće kliknuti na gumb *Dolazak* prije nego je kliknut gumb *Odlazak*, nakon klika na gumb *Pauza* moguće je jedino pritisnuti gumb *Kraj pauze*.

S lijeve strane nalazi se navigacijski izbornik s četiri opcije:

* *Početna* – ekran za unos evidencije vremena u aplikaciju
* *Evidencija vremena* – ekran za prikaz evidencije vremena za prijavljenog korisnika
* *Postavke* – ekran za izmjenu korisničkog imena ili lozinke
* *Odjava* – odjava iz aplikacije



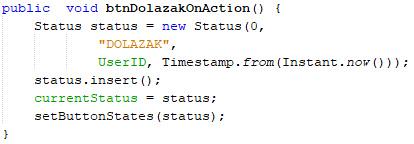
Slika 38. Korisnik – Početna stranica

Kod za isključivanje pojedinog gumba prikazan je na slici 39. Osim što se, kroz metodu, gumbovi međusobno isključuju, postavljaju se i trenutni statusi korisnika koji se ispisuju na ekranu. Na gornjoj slici, trenutni status predstavljen je tekstom *Trenutni status: ZAVRŠENO RADNO VRIJEME…*



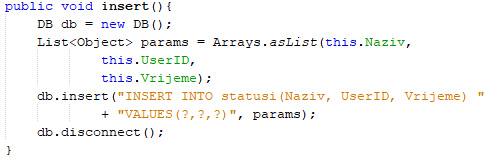
Slika 39. Izmjena radnog statusa

Primjer koda koji se izvršava klikom na gumb „Dolazak“ prikazan je na slici 40.



Slika 40. Pritisak gumba "Dolazak"

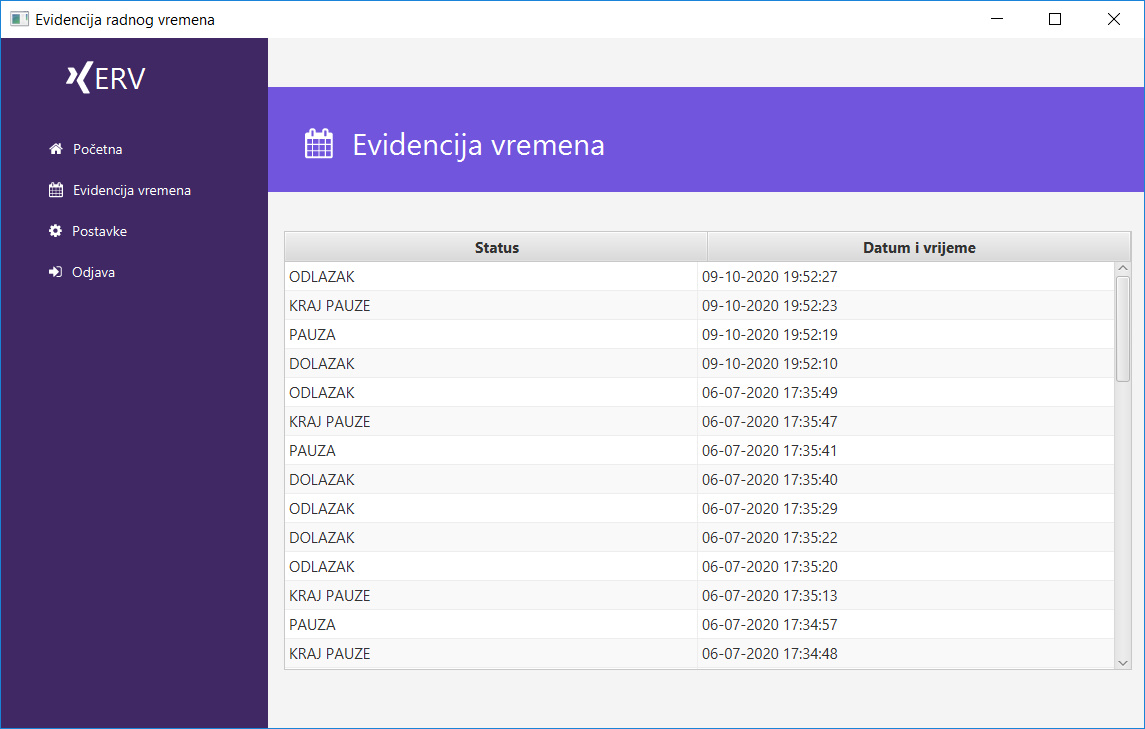
Unutar gore navedene metode poziva se metoda insert() smještena unutar klase *Status* pomoću koje se unosi status u bazu podataka. Kod koji se izvršava pozivom metode insert() prikazan je na slici 41:



Slika 41. Unos statusa u bazu podataka

### Evidencija vremena - Korisnik

Odabirom opcije *Evidencija vremena* prikazuje se, u tabličnom prikazu, evidencija vremena onog korisnika koji je trenutno prijavljen u aplikaciju. Tablica sadrži dvije kolone *Status* i *Datum i vrijeme,*koje predstavljaju unesene statuse i vremena kada su statusi uneseni. Prikaz ekrana za evidenciju vremena prikazan je na slici ispod.



Slika 42. Korisnik - evidencija vremena

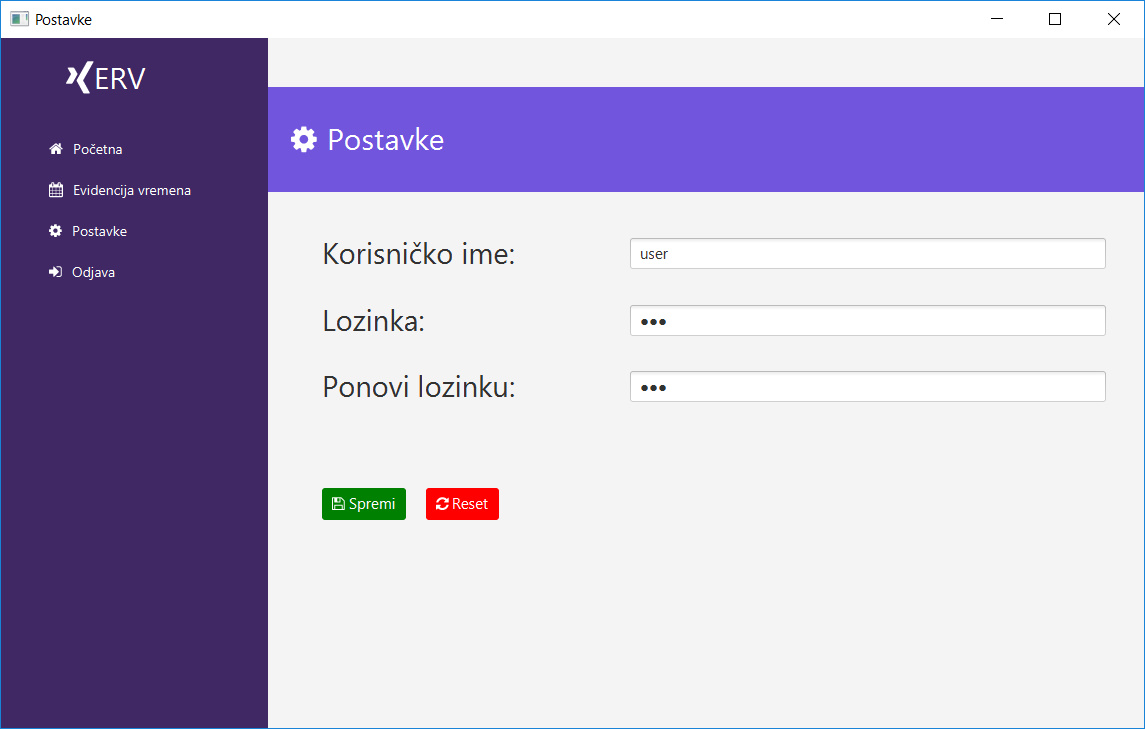
### Postavke - Korisnik

Pritiskom na opciju *Postavke* na ekranu se prikazuje ekran pomoću kojeg trenutno prijavljeni korisnik može izmijeniti svoje podatke. Korisnik može izmijeniti svoje korisničko ime i lozinku. Polja su popunjena vrijednostima dohvaćenim iz baze podataka. Pritiskom na gumb *Spremi* podaci se ažuriraju u bazi podataka. Preduvjet za uspješno ažuriranje je ispravan unos podataka u polja.

U slučaju neispavanog unosa, na ekranu će se prikazati jedna od sljedećih poruka:

* *Unesite korisničko ime* – ako nije ispunjena vrijednost polja *Korisničko ime*
* *Unesite lozinku* – ako nije ispunjena vrijednost polja *Lozinka*
* *Ponovite lozinku* – ako nije ispunjena vrijednost polja *Ponovi lozinku*
* *Lozinke se ne podudaraju* – ako vrijednosti polja *Lozinka* i *Ponovi lozinku* nisu jednake

Klikom na gumb *Reset* obrisat će se sve vrijednosti unutar polja za unos. Prikaz ekrana koji se otvara klikom na opciju *Postavke* prikazan je na slici ispod.



Slika 43. Korisnik – Postavke

Kod koji se izvršava klikom na gumb spremi prikazan je na slici 44. Ako su korisnički podaci uspješno izmijenjeni, prikazat će se tekst zelene boje, u protivnom će se ispisati tekst crvene boje.



Slika 44. Spremi postavke

## MVC arhitektura u aplikaciji

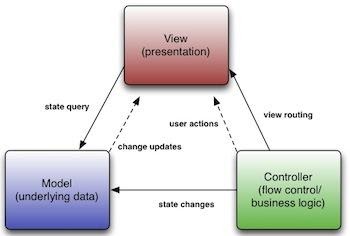
JavaFX aplikacije temelje se na MVC arhitekturi. Navedena arhitektura odvaja slojeve aplikacije koji predstavljaju podatke (engl. Model), grafičko sučelje (engl. View) i samu poslovnu logiku aplikacije (engl. Controller). Takva arhitektura koristi se i u ostalim programskim jezicima uz mala odstupanja.

Unutar modela podataka nalaze se klase koje predstavljaju domenske podatke, odnosno podatke iz baze podataka.

Grafičko sučelje temelji se na datotekama koje imaju „.fxml“ ekstenzije u kojima se definira izgled grafičkog sučelja. Osim navedenog, grafičko sučelje može sadržavati klase pomoću kojih se dizajnira grafičko sučelje pomoću naredbi kroz kod napisan u Java programskom jeziku.

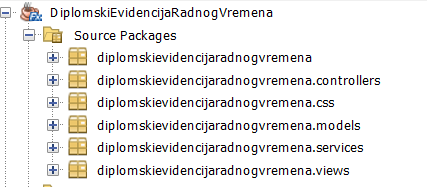
Poslovna logika aplikacije sadrži klase u kojima se povezuju elementi grafičkog sučelja, primjerice, gumba, s metodama koje sadrže logiku koja se mora izvršiti tijekom interakcije korisnika s tim elementima (primjerice, pritisak gumba).

Izgled MVC arhitekture prikazan je na slici 45.

****

Slika 45. MVC arhitektura

Postoje dva načina strukturiranja aplikacije – prema slojevima (engl. Package by layer) i prema značajkama (engl. Packageby feature). Strukturiranje paketa prema slojevima podrazumijeva kreiranje paketa koji će sadržavati sve klase iste funkcionalnosti. Primjerice, unutar paketa naziva controller, nalazit će se sve Controller klase. Svaki paket sadrži komponente koje uglavnom nisu usko povezane. Strukturiranje prema značajkama označava kreiranje strukture prema značajkama sustava. Za svaku značajku sustava kreiran je dedicirani paket. Paketi su međusobno slabo povezani, a sve komponente koje međusobno komuniciraju smještene su u istom paketu. Aplikacija za evidenciju radnog vremena strukturirana je prema slojevima kako bi bila preglednija. Struktura aplikacije prikazana je na slici ispod.



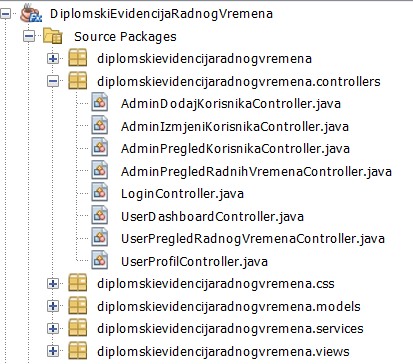
Slika 46.Struktura aplikacije

Kao što je vidljivo na slici 46. unutar aplikacije kreirana su pet paketa:

* diplomskievidencijaradnogvremena.controllers,
* diplomskievidencijaradnogvremena.css,
* diplomskievidencijaradnogvremena.models
* diplomskievidencijaradnogvremena.services
* diplomskievidencijaradnogvremena.views

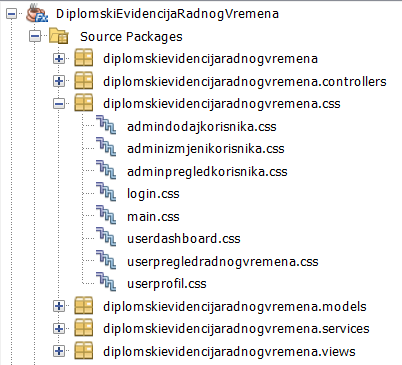
Postoji i vršni paket **diplomskievidencijaradnogvremena**u kojem se nalazi klasa DiplomskiEvidencijaRadnogVremena.java koja predstavlja početnu točku aplikacije, te klasa DB.java koja sadrži logiku za otvaranje i zatvaranje konekcije nad bazom podataka.

Unutar paketa **diplomskievidencijaradnogvremena.controllers** smještene su sve kontroler klase u kojima se nalazi poslovna logika sustava. Datoteke unutar controllers paketa prikazane su na slici 47.



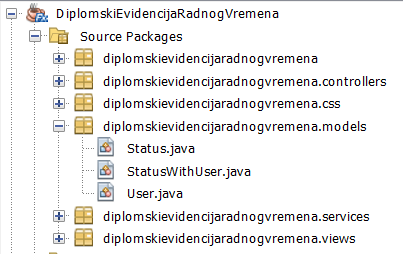
Slika 47. Datoteke unutar controllers paketa

Unutar paketa **diplomskievidencijaradnogvremena.css**nalaze se CSS (engl. CascadingStyleSheets) datoteke koje služe za oblikovanje datoteka smještenih unutar views paketa. Kao i kod stiliziranja web sučelja, moguće je koristiti CSS klase i atribute za oblikovanje elemenata unutar desktop aplikacije. CSS datoteke smještene unutar paketa naziva cssprikazane su na slici 48.



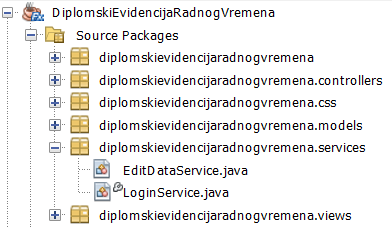
Slika 48. Datoteke unutar css paketa

U paketu diplomskievidencijaradnogvremena.models nalaze se modeli koji predstavljaju domenske klase. Pomoću njih ostvaruje se jednostavnije upravljanje podacima iz baze podataka. Svaka tablica unutar baze podataka ima dodijeljenu klasu unutar paketa models. Klase unutar navedenog paketa sadrže metode za dohvaćanje, izmjenu, pregled i brisanje podataka iz baze podataka. Prikaz datoteka unutar models paketa dan je na slici 49.



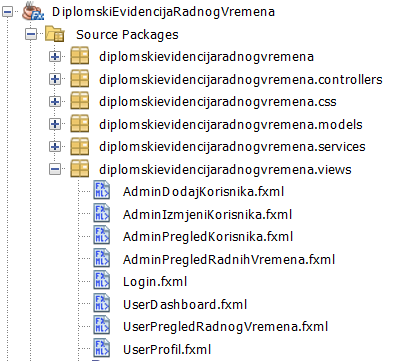
Slika 49. Datoteke unutar models paketa

Unutar paketa **diplomskievidencijaradnogvremena.services**nalaze se klase za ažuriranje postojećeg korisnika i za odjavu iz aplikacije. Datoteke koje se nalaze unutar paketa services prikazane su na slici.



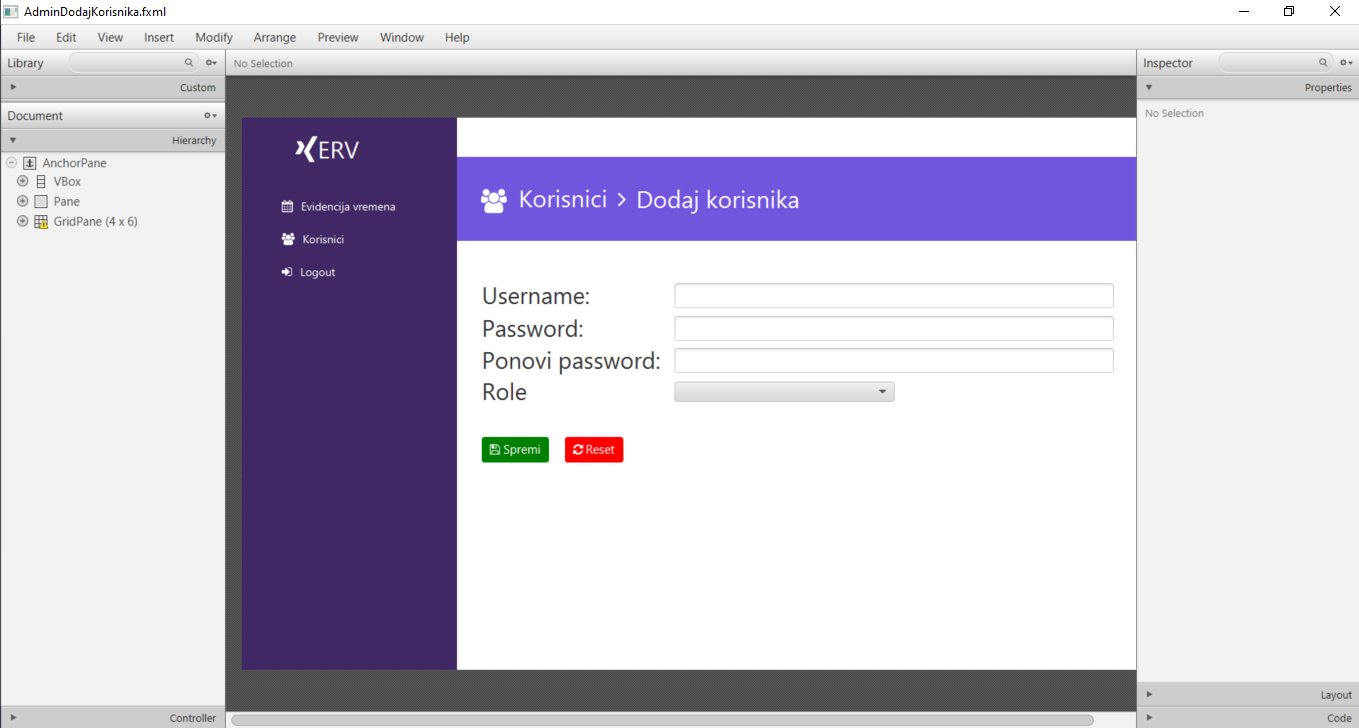
Slika 50. Datoteke unutar services paketa

Unutar paketa **diplomskievidencijaradnogvremena.views**nalaze se datoteke s ekstenzijom „.fxml“. Svaka datoteke unutar paketa views ima dodijeljenu CSS datoteku kojom su komponente stilizirane te kontroler klasu koja sadrži poslovnu logiku. Primjerice, datoteka naziva AdminDodajKorisnika.fxml ima dodijeljenu CSS datoteku naziva admindodajkorisnika.css te kontroler klasu AdminDodajKorisnika.java. Sadržaj paketa viewsprikazan je na slici ispod.



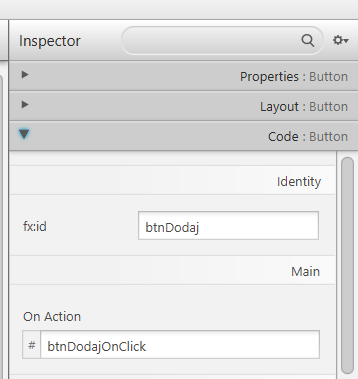
Slika 51. Datoteke unutar views paketa

Poglede, odnosno .fxml datoteke moguće je kreirati i uređivati izravno koristeći aplikaciju JavaFX Scene Builder 2.0. Izgled datoteke AdminDodajKorisnika.fxml pokrenut pomoću Scene Builder aplikacije prikazan je na slici 52.

****

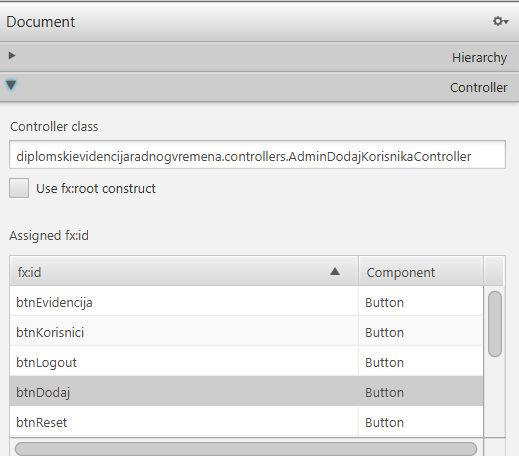
Slika 52. AdminDodajKorisnika.fxml datoteke unutar Scene Builder aplikacije

S desne strane sučelja, moguće je definirati koja akcija će se izvršiti, primjerice, klikom na gumb *Spremi*. Unutar polja naziva *On Action* potrebno je upisati metodu koja će se pozvati. Postavljanje akcije za gumb *Spremi* prikazano je na slici 53.



Slika 53. Postavljanje akcije nad gumbom "Spremi"

U izborniku s lijeve strane aplikacije Scene Builder moguće je određenom pregledu dodijeliti kontroler klasu. Potrebno je upisati pakete i naziv klase koju želimo pridružiti otvorenojfxml datoteci. Primjer definiranja klase za datoteku Admin.DodajKorisnika.fxml prikazan je na slici 54.

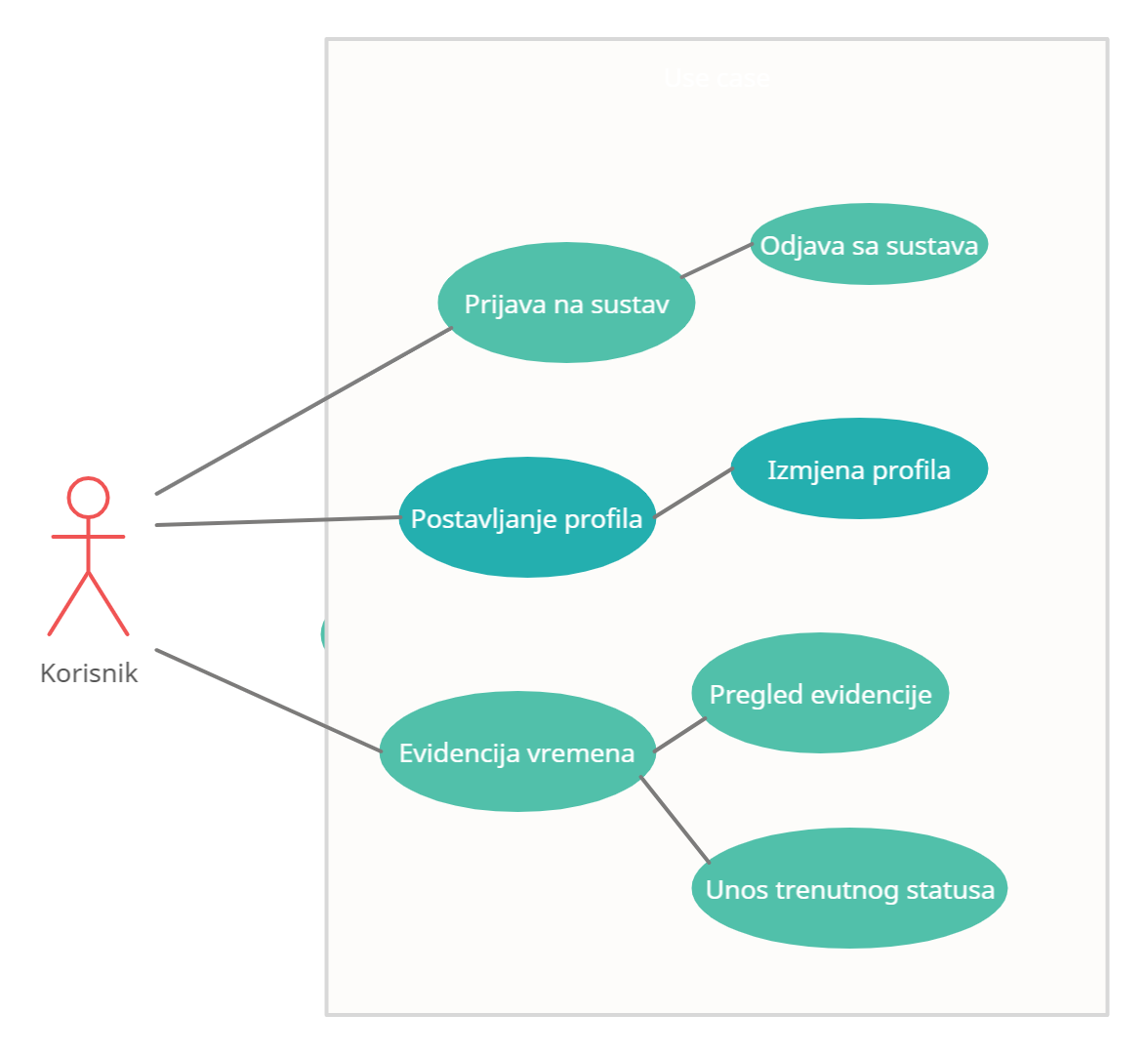


Slika 54. Dodjeljivanje controller klase

## Slučajevi korištenja

Java desktop aplikacija koja je predmet ovog istraživanja razlikuje dvije razine korisničkog sučelja – korisničko sučelje za korisnike s ulogomAdmin i korisničko sučelje za korisnike s ulogom User. Sukladno navedenoj razdiobi, u aplikaciji postoje dvije razine slučajeva korištenja – slučajevi korištenja za administratore te slučajevi korištenja za obične korisnike.

Na slici 55. prikazani su slučajevi korištenja kada je riječ o korisniku s ulogom User. Slučajevi korištenja podrazumijevaju funkcionalnosti prijave u sustav, a samim time i odjave iz sustava, postavljanja profila i funkcionalnost evidentiranja radnog vremena. 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.

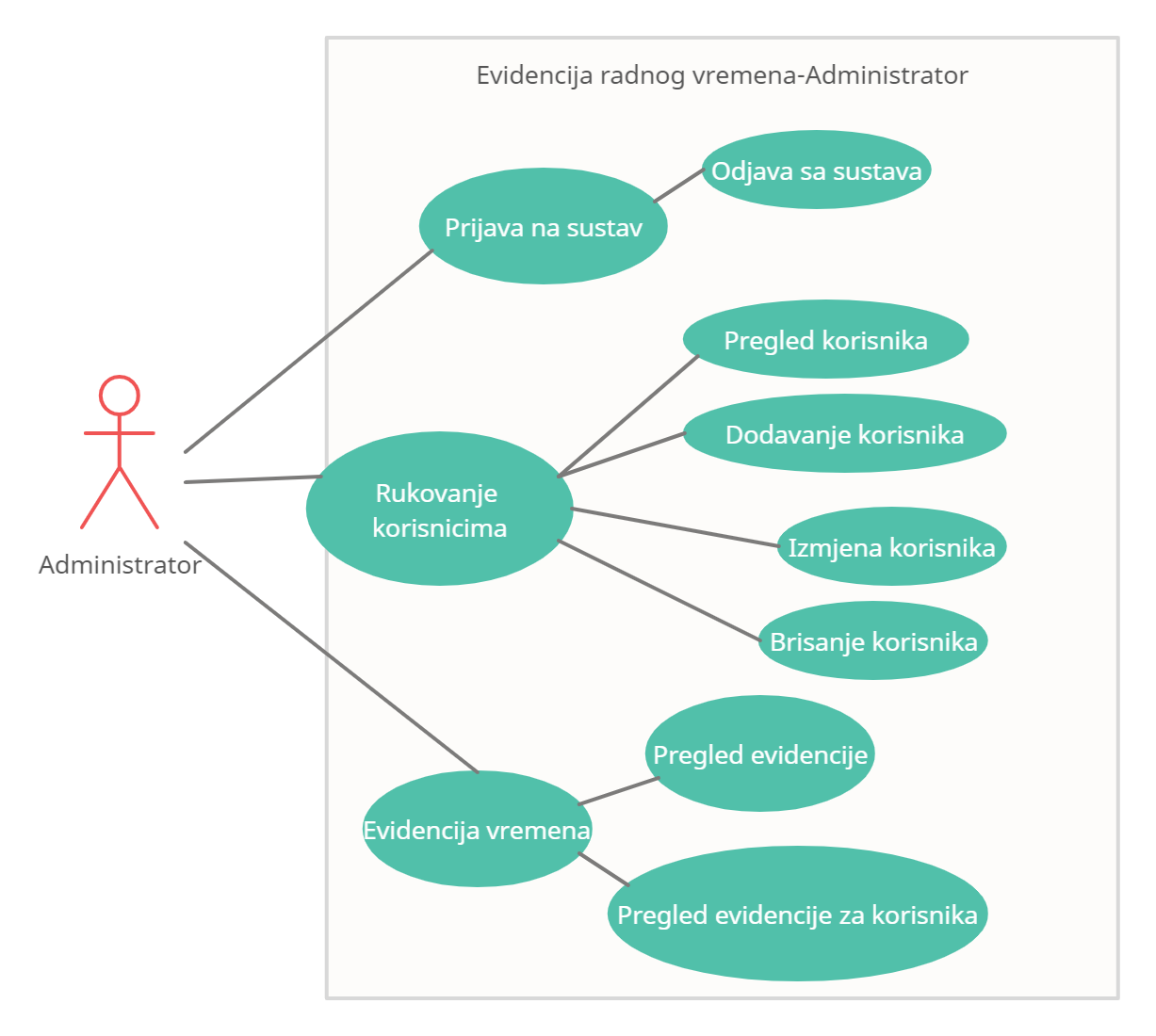


Slika 55. Slučajevi korištenja - Korisnik

Na slici 56. se nalaze slučajevi korištenja za korisnike s ulogomAdmin.Kao i kod korisnika s ulogom User, i navedeni korisnici imaju funkcionalnost prijave u sustav, a samim time i odjave iz sustava. Također, imaju funkcionalnost pregleda evidencije radnih vremena.

U odnosu na korisnike s ulogom User, korisnici s ulogom Admin imaju funkcionalnost upravljanja svim korisnicima unutar sustava. Ova funkcionalnost omogućava 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 56. Slučajevi korištenja – Administator

## Mogućnosti proširenja aplikacije

Aplikacija je izrađena prvenstveno u svrhe učenja i vježbanja programskoga jezika Java te stoga se s aplikacijom nije išlo u krajnosti i detalje. Cilj ovog rada bio je napraviti aplikaciju koja ima osnovne funkcionalnosti evidentiranja radnog vremena.

Aplikaciju je moguće proširiti na više načina. Moguće je pohraniti korisničke podatke u bazu, poput imena, prezimena, datuma i mjesta rođenja, e-maila, JMBG-a. Također, moguće je proširiti aplikaciju na način da se omogući unos datoteka. Na taj način bi se mogli pohranjivati važni podaci, poput kopije osobne iskaznice, vozačke dozvole, certifikata. Uz navedeno, mogli bi se pohranjivati i ugovori o radu, kao i dodaci o ugovoru.

Moguće proširenje je i dodavanje kalendara gdje bi korisnik mogao unositi dane kada je na bolovanju ili na godišnjem. Na taj način bi se, na jednostavan način, mogli izvući podaci koliko je zaposleniku ostalo dana godišnjeg odmora ili koliko dugo je koristio bolovanje.

Također, moguće je izmijeniti postojeću funkcionalnost radnih statusa na način da se statusi mogu dinamički upisivati od strane korisnika, kako korisnik ne bi bio ograničen na predefiniranih četiri vrijednosti.

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 komunikaciju između Android i Java aplikacije trebalo bi koristiti REST (engl.Representationalstate transfer*)* arhitekturu. REST arhitektura koristi HTTP (engl.Hypertext Transfer Protocol), a podaci se šalju i primaju u JSON (engl. JavaScriptObjectNotation) obliku. Na taj način, objavom metoda preko API sučelja, aplikaciji bi se moglo pristupiti koristeći više tehnologija, primjerice, putem WEB sučelja, Android aplikacije, iOS aplikacije, desktop aplikacije i slično. Sve navedene aplikacije pozivale bi iste objavljene metode slanjem zahtjeva (engl. request) u JSON obliku koji bi se deserijalizirao u objekt zadane klase kojeg bi zatim bilo moguće koristiti dalje u Java aplikaciji.

# ZAKLJUČAK

U ovom diplomskom radu je najprije u kratkim crtama opisan programski jezik Java; Navedene su neke njegove osnovne karakteristike od kojih su posebno istaknute prenosivost kao i objektna orijentiranost programskog jezika, a istaknute su i još neke karakteristike koje ujedno spadaju i u prednosti korištenja Java-e poput jednostavnosti, sigurnosti, besplatnosti itd.

Od 52 ključne riječi koje se koriste za naredbe izdvojene su i u kratkim crticama su objašnjene one koje se često koriste prilikom programiranja u Java-i.

Osim što je Java programski jezik visoke razine također predstavlja neovisnu računalnu platformu na kojoj se mogu razvijati i izvršavati programi. Java programski jezik se može koristiti u više integriranih razvojnih okruženja, a najpoznatija od njih su NetBeans, IntelliJIdea, Eclipse, BlueJ i JDeveloper.

U praktičnom dijelu rada je uz pomoć NetBeans-a implementirana jednostavna aplikacija za evidenciju radnog vremena. Korišten je Java programski jezik te arhitektonski obrazac MVC.Aplikacija je razvijena u svrhu učenja Java programskog jezika, ali uz nju je moguće efikasno evidentirati radna vremena zaposlenika u nekoj zamišljenoj tvrtci.Također, razvijenu aplikacije je moguće proširiti primjerice s integracijom mobilne android ili iOS aplikacije, gdje bi korisnici mogli unositi statuse direktno sa svojih mobilnih uređaja.

Zaključno, poznavanje Java programskog jezika, Java FX-a, MySQL-a i MVC-a je dovoljno za implementaciju jednostavnih, brzih i modernih desktop aplikacija bogatih izgledom koje mogu raditi na svim operacijskim sustavima.

# LITERATURA

1. <https://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf> (Siječanj, 2021.)
2. [javatpoint.com/features-of-java](file:///C:\Users\Korisnik\Desktop\javatpoint.com\features-of-java)(Siječanj, 2021.)
3. <https://hr.wikipedia.org/wiki/Java_(programski_jezik)>(Siječanj, 2021.)
4. <http://docplayer.rs/184548646-Programiranje-nastava-prof-dr-sc-dra%C5%BEena-ga%C5%A1par-datum.html>(Siječanj, 2021.)
5. <https://medium.com/@rahul77349/difference-between-compiler-and-interpreter-with-respect-to-jvm-java-virtual-machine-and-pvm-22fc77ae0eb7>

(Siječanj, 2021.)

1. <https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf>

(Siječanj, 2021.)

<https://www.topperskills.com/tutorials/java/features-properties-java-language.html>(Siječanj, 2021)

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

<https://www.weblineindia.com/blog/java-development-advantages-disadvantages/>(Siječanj, 2021.)

1. <http://laris.fesb.hr/java/osnovni_program.htm>(Siječanj, 2021.)

[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)(Siječanj, 2021)

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>(Siječanj, 2021.)

1. <https://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/3.%20Klase_i_objekti.pdf>(Siječanj, 2021.)
2. <https://javaprogramiranje.wordpress.com/2009/11/07/upravljacke-naredbe/> (Siječanj, 2021.)
3. Burd B.(2017): BeginningProgrammingwith Java For Dummies. IDG Books Worldwide, Inc. Foster City, California.
4. <https://www.softwaretestinghelp.com/java-components-java-platform-jdk/> (Siječanj, 2021.)
5. <https://www.javatpoint.com/difference-between-jdk-jre-and-jvm> (Siječanj, 2021)

<http://mdita.metropolitan.ac.rs/qdita-temp/KT103/L01/KT103-L01-pptlc4.html>(Veljača, 2021)

1. <https://sr.wikipedia.org/wiki/NetBeans>(Veljača, 2021.)
2. <https://netbeans.org/about/history.html>(Veljača, 2021.)
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.)(Veljača,2021.)
4. <https://hackr.io/blog/best-java-ides>(Veljača,2021.)
5. <https://www.bluej.org/about.html>(Veljača, 2021.)
6. <https://www.oracle.com/application-development/technologies/jdeveloper.html>(Veljača, 2021.)
7. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf>

(Veljača,2021.)

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 ServletProgramming. O'ReillyandAssociates, 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): BeginningProgrammingwith 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)