**SVEUČILIŠTE U MOSTARU**

**FAKULTET PRIRODOSLOVNO-MATEMATIČKIH I ODGOJNIH ZNANOSTI**

**STUDIJ INFORMATIKE**

**MATEJ MADŽAREVIĆ**

**RAZVOJ APLIKACIJE ZA EVIDENCIJU RADNOG VREMENA U JAVI**

**DIPLOMSKI RAD**

**Mostar, 2023.** **SVEUČILIŠTE U MOSTARU**

**FAKULTET PRIRODOSLOVNO-MATEMATIČKIH I ODGOJNIH ZNANOSTI**

**STUDIJ INFORMATIKE**

**RAZVOJ APLIKACIJE ZA EVIDENCIJU RADNOG VREMENA U JAVI**

**DIPLOMSKI RAD**

**Mentor: dr. sc. Tomislav Volarić, izv. prof.**

**Student: Matej Madžarević**

**Mostar, listopad 2023.**

**PODACI O DIPLOMSKOM RADU**

1. **AUTOR**

Ime i prezime: Matej Madžarević

Datum i mjesto rođenja: 09.08.1993. Osijek, Republika Hrvatska

Studij: Informatika

Matični broj:3044

1. **DIPLOMSKI RAD**

Tema:Razvoj aplikacije za evidenciju radnog vremena u Javi

Broj stranica:51

Datum predaje diplomskog rada: 25.09.2023.

Datum obrane diplomskog rada: 12.10.2023.

Mentor: dr.sc. Tomislav Volarić, izv. prof.

**Sastav povjerenstva koje je ocijenilo rad:**

Mentor: dr. sc. Tomislav Volarić, izv. prof.

Predsjednik: dr.sc. Daniel Vasić

Član: mag. Hrvoje Ljubić, asist.

Potpisi članova povjerenstva: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ocjena diplomskog rada i ocjena obrane: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sadržaj

[1. UVOD 1](#_Toc131880866)

[2. PROGRAMSKI JEZIK JAVA 3](#_Toc131880867)

[2.1. Osnovne karakteristike programskog jezika Java 4](#_Toc131880868)

[2.2. Prednosti i nedostaci programskog jezika Java 6](#_Toc131880869)

[2.3. Naredbe u Javi 8](#_Toc131880870)

[2.4. Java platforma 13](#_Toc131880871)

[2.5. Integrirano razvojno okruženje NetBeans 15](#_Toc131880872)

[2.6. Ostala razvojna okruženja za Javu 17](#_Toc131880873)

[2.7. MVC Arhitektura 19](#_Toc131880874)

[3. JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA 21](#_Toc131880875)

[3.1. Opis problema 21](#_Toc131880876)

[3.2. Relacijski model baze podataka 23](#_Toc131880877)

[3.3. Prozori aplikacije 24](#_Toc131880878)

[3.3.1. Prijava 24](#_Toc131880879)

[3.3.2. Pregled radnih vremena - Admin 26](#_Toc131880880)

[3.3.3. Pregled korisnika- Admin 28](#_Toc131880881)

[3.3.4. Dodaj korisnika - Admin 29](#_Toc131880882)

[3.3.5. Izmjeni korisnika - Admin 31](#_Toc131880883)

[3.3.6. Početna – Korisnik 33](#_Toc131880884)

[3.3.7. Povijest radnog vremena- Korisnik 35](#_Toc131880885)

[3.3.8. Postavke - Korisnik 36](#_Toc131880886)

[3.4. MVC arhitektura u aplikaciji 37](#_Toc131880887)

[3.5. Slučajevi korištenja 43](#_Toc131880888)

[3.6. Mogućnosti proširenja aplikacije 44](#_Toc131880889)

[4. ZAKLJUČAK 46](#_Toc131880890)

[5. LITERATURA 47](#_Toc131880891)

**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 razvijena po MVC (Model-Pogled-Upravitelj, eng. Model-View-Controler,) 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 udruženju (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 uz pomoć informacijskih tehnologija 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 stvarnom 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 relativno 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.

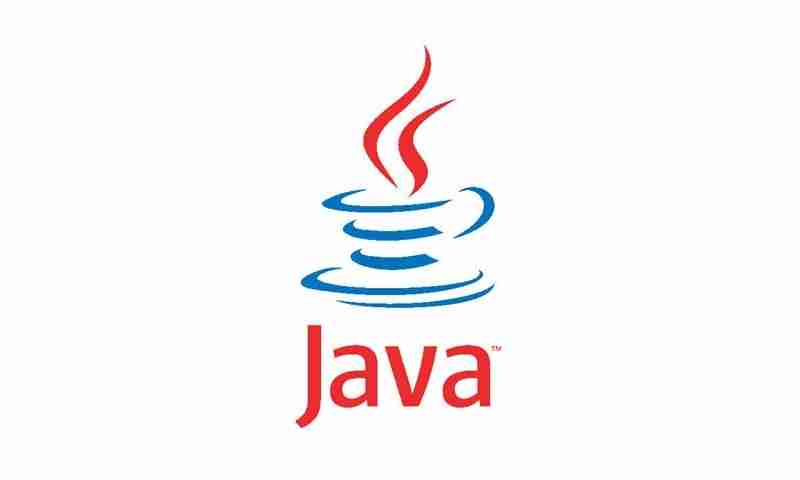
U drugom dijelu rada opisan je problem koji rješava Java desktop aplikacija implementirana u razvojnom okruženju Netbeans, relacijski model baze podataka i prozori koji se pojavljuju prilikom korištenja aplikacije za evidenciju radnog vremena. Također su u posebnom odlomku drugoga dijela navedeni i objašnjeni paketi koji se nalaze unutar same aplikacije, odnosno paketi koji podrazumijevaju modele, poglede i upravitelje. U zasebnom su odlomku uz pomoć dijagrama slučajeva korištenja (eng. 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 (Green 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].

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 (Slika 1.) [2].



Slika 1. Logo Jave

Prvi prgram u Javi 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 Javinih programa unutar web preglednika [1].

Projekt je konačno objavljen u studenom 1995. godine kada je izašla probna verzija programskog jezika (beta verzija) [3].

Početkom 1996. godine izlazi prva službena verzija Java platforme i Java programskog jezika, a iste godine Javu licenciraju važne tvrtke poput NetScape-a, IBM-a, Microsoft-a i drugih [1].

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

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

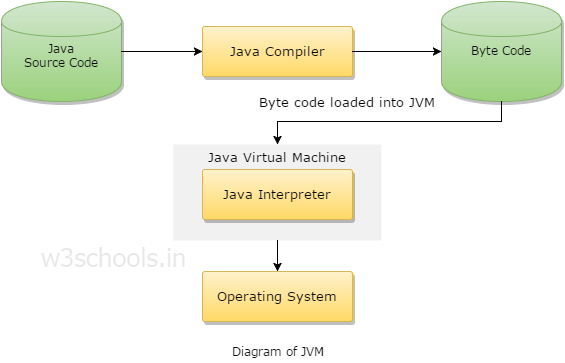
Danas je, prema podacima GitHub-a, Java drugi programski jezik po zastupljenosti među programerskom populacijom i do sada je poboljšavan kroz svojih 20 verzija pa je tako najnovija verzija Jave (Java 20) izašla u ožujku 2023. godine.

## 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. 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 [4].

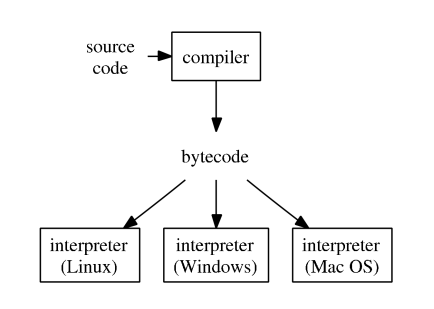
Programski jezik Java je po ovoj podjeli izrazito specifičan iz razloga što se može svrstati u bilo koju od navedenih skupina, točnije za Javu se može reći da je i kompilirani i interpretirani programski jezik. Razlog tome leži u činjenici da je programski jezik Java dizajniran tako da se programi napisani u njemu mogu jednostavno izvoditi kroz različite tipove platformi bez unošenja ikakvih promjena na tim programima. Za ovakvo izvođenje Javinih programa zaslužan je Java virtualni stroj (eng. JVM- Java Virtual Machine), koji može biti instaliran na bilo kojem od postojećih operacijskih sustava [5].

Prilikom prevođenja izvornog koda u strojni kod, Javinih 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 (Slika 2.) [5].



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 Javi 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.) i arhitekturne neutralnosti koja se odnosi na arhitekturu računala ili vrstu procesora na kojoj se izvode programi napisani u Javi [5].



Slika 3. Pokretanje izvornog koda napisanog u Javi 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 Javi piše unutar klasa, što znači da svaki program napisan u Javi mora sadržavati barem jednu (glavnu) klasu.

U usporedbi s ostalim programskim jezicima, sintaksa kao i programska logika jezika Jave najsličnije su onima u C i C++ programskom jeziku koji su postojali prije Jave. Java je upravo od njih zadržala pojedine značajke koje su predstavljale prednosti prilikom pisanja koda, a uklonila je pri tom i određene nedostatke koje 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 Javi ukinuti ili zamijenjeni alternativnim rješenjima su [1]:

* 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 Javi jednako se lako pišu za korištenje u raznim dijelovima svijeta gdje se ljudi ne služe standardnim latiničnim pismom ili engleskim jezikom. To znači da mogu podržavati većinu prirodnih jezika te se automatski prilagoditi okruženju u kojem se izvode.

## 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. Najvažnije od njih su [5]:

* Jednostavnost
* Sigurnost
* Robusnost
* Podrška za paralelno programiranje
* Visoke performanse
* Dinamičnost
* 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 onoj u C-u i C++-u, izrazito je jednostavan za naučiti ukoliko se dobro poznaju spomenuti jezici. Javinoj jednostavnosti pridonosi činjenica da su u Javi uklonjeni eksplicitni pokazivači i preopterećenje operatora [6].

Java je veoma siguran programski jezik iz dva razloga; Prvo zato što Java interpreter provjerava bytecode prije samog izvršavanja. Ukoliko se ispostavi da je kod nesiguran tada se taj kod neće izvršiti. Drugi razlog je taj što Java ne podržava pisanje koda koji izravno pristupa memoriji lokalnog sustava [6].

Za Javu 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 Jave [1].

Također, u Javi je moguće pisati programe koji rade na mnogim zadacima odjednom tako što se definiraju višestruke niti. Niti su važne za multimediju, Web aplikacije itd. Glavna prednost višenitnosti (eng. multithreading-a) je dijeljene iste memorije. Ova karakteristika pruža programima bolje funkcioniranje u stvarnom vremenu [6].

Java virtualni stroj koristi tzv. točno na vrijeme način prevođenja (eng. Just in time) što u Javi omogućava izvršavanje koda tijekom samog prevođenja i na taj način se ostvaruju visoke performanse samog programskog jezika [6].

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 Javi. 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 i drugih.

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 Javu, ali poznavanje tih nedostataka može pomoći u konstruktivnijem korištenju jave.

Nedostaci Java programskog jezika su sljedeći [7]:

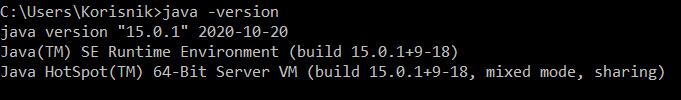
* Java nije dovoljno optimizirana u usporedbi s C i C++ programskim jezicima
* Upravljanje memorijom u Javi je skupo
* Nedovoljna usredotočenost na sigurnosne kopije podataka

## Naredbe u Javi

U programskom jeziku Java, naredba (ključna riječ) je bilo koja od 52 ključnih riječi koje su rezervirane i imaju unaprijed definirano značenje. 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 [8].

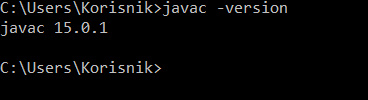
U ovom su poglavlju najprije objašnjene neke od osnovnih naredba u Javi koje je moguće unijeti izravno preko tekstualnog sučelja određenog operacijskog sustava. Nakon tih naredbi su u nastavku objašnjene naredbe i ključne riječi koje se koriste prilikom kodiranja u integriranim razvojnim okruženjima:

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



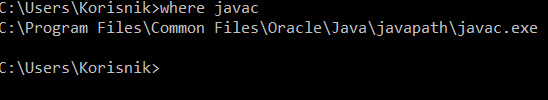
Slika 4. Primjer unošenja naredbe Java –version u terminal

**Javac-version:** Ova naredba se koristi za prikaz verzije prevoditelja koji je odgovoran za sastavljanje izvornog koda. Ovo je također dio Java razvojnog kompleta pod akronimom 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.



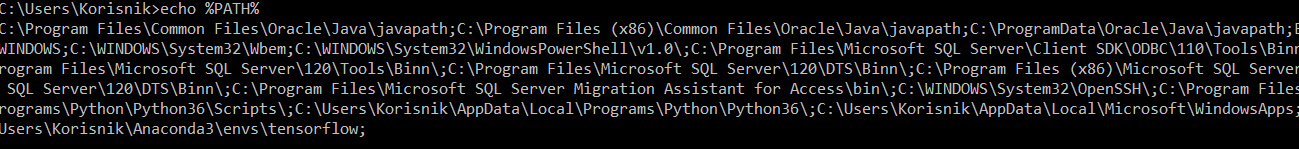
Slika 6. Upotreba naredbe Where u CMD-u

Za izvršavanje programa nakon što je pretvoren u bytecode, koristi se ključna riječ **java** i potom naziv programa:



Slika 7. Izvršavanje programa

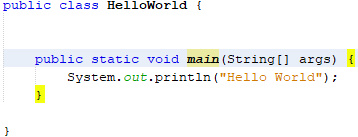
**Echo:** Ova naredba se ne koristi samo u Javi, nego je naredba koju se mora poznavati jer pomaže u prikazivanju podataka određene datoteke. Obično se koristi za provjeru PATH varijabli.



Slika 8. Naredba Echo

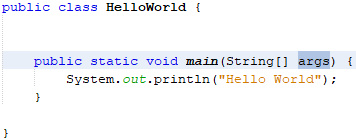
Nakon što su navedene osnovne naredbe koje je moguće unijeti uz pomoć tekstualnog sučelja određenog operacijskog sustava, u daljem tekstu slijede pojedini primjeri kotišenja bitnijih naredbni odnosno ključnih riječi koje se koriste u Javi-i prilikom kodiranja u integriranim razvojnim okruženjima:

**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 Javi može spremiti podatak je korištenjem naredbe pridruživanja vrijednosti. 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:

**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:

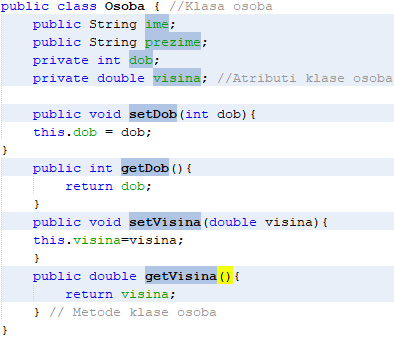
**char** znak=Y;

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** corona\_test= false;

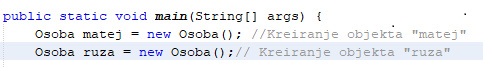
Važno je napomenuti kako svi navedeni tipovi podataka spadaju u skupinu tzv. primitivnih (jednostavnih) tipova podataka. Pored jednostavnih u Javi postoje i složeni tipovi podataka, a to su polja, klase i sučelja. Za deklariranje polja u Javi se koristi ključna riječ **Array**, za deklariranje klasa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

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



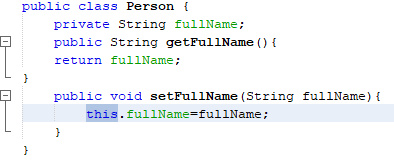
Slika 11. Atributi i metode klase Osoba

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



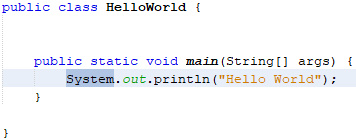
Slika 12. Primjer kreiranja objekata u Javi

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



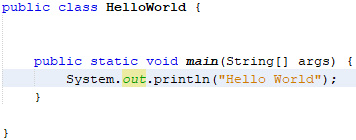
Slika 13. Ključna riječ this

**System:** System je unaprijed izgrađena klasa u Javi koja je prisutna u zadanom paketu lang koji se koristi za rukovanje standardnim ulazima i standardnim izlazima.



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.



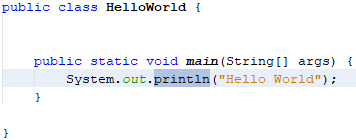
Slika 15. Out- statički član klase System

**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. Javine upravljačke naredbe se mogu svrstati u sljedeće kategorije [9]:

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

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:

* **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**.

Javine 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 Javina osnovna petlja. Ona ponovljeno izvršava naredbu ili blok sve dok je upravljački izraz točan.

Opći oblik petlje while 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.

Opći oblik petlje do-while je:

**do** {

tijelo petlje

} **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.

Na primjer: **for** (int n=10; n>0 ;n--)

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

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 Javi 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.

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.

## 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 te obuhvaća skupinu programa koji omogućuju razvoj, ali i pokretanje programa napisanih u Javi u čijem se središtu nalazi već spomenuti Java virtualni stroj [1].

Razlikuje se od većine ostalih računalnih platformi po tome što je ona software-ski zasnovana platforma koja funkcionira 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:

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.

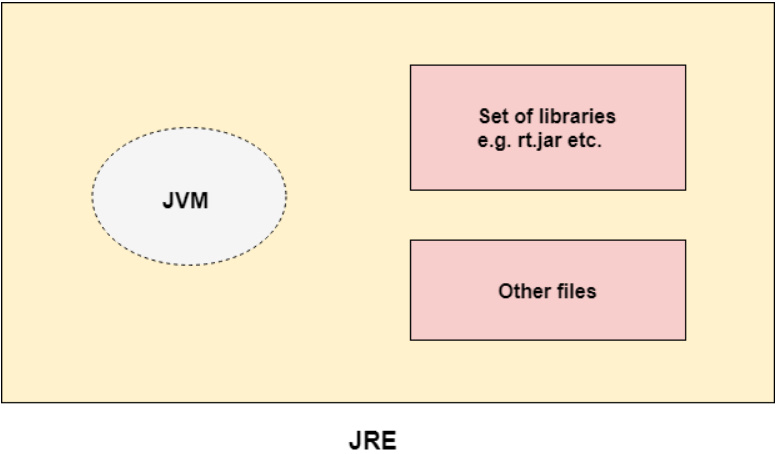
Naime, u središtu platforme se nalazi **JVM**. Naziva se virtualni zato što ne postoji u fizičkom smislu. U ovom radu je opisano kako se programi napisani u Javi se ne prevode (kao što je slučaj kod većine drugih programskih jezika) u izvršni kod koji se izravno izvršava na procesoru računala, nego se 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 [1]. 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 [10]:

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

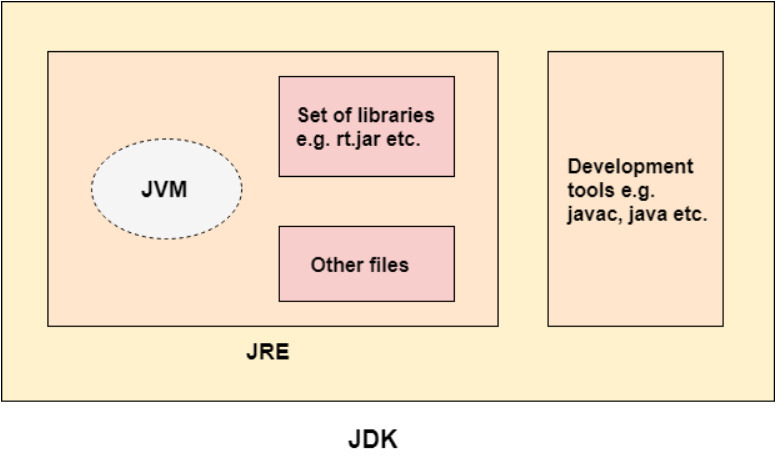
Nadalje, neizostavan dio Java platforme jeste Javino okruženje za izvršavanje odnosno **JRE** (također se rabi skraćenica RTE- Run Time Enviorment).

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 18.) [10].



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 19.) Predstavlja implementaciju bilo kojeg izdanja Java platforme (Java SE, Java ME ili Java EE) [10].



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 stručnjaci (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:

* 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. 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 Javi, u NetBeans-u je uz odgovarajuća proširenja moguće programirati i u ostalim programskim jezicima, kao što su C, C++, C#, HTML, JavaScript, PHP te brojni drugi programski jezici.

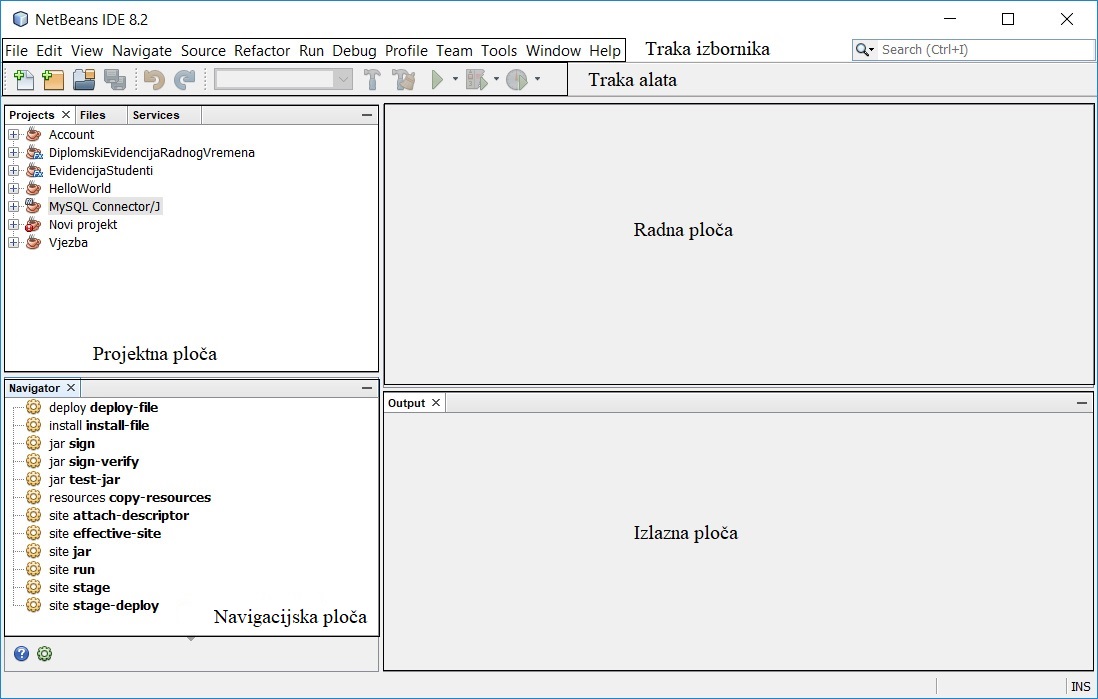
NetBeans korijene vuče iz 1996. kada je započeo kao projekt grupe studenata Fakulteta za matematiku i fiziku na Sveučilištu u Pragu (tadašnja Čehoslovačka, današnja Češka) pod nazivom Xelfi. Cilj projekta bio je stvoriti integrirano razvojno okruženje za Javu, što je grupi studenata i uspjelo 1997. godine, a uspjeh kao takav predstavljao je prvi IDE za Javu 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 arhitekture ovog IDE-a tada 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. Tako je najnovije izdanje NetBeans IDE-a (NetBeans 8.2) izašlo u listopadu 2016. godine [11].

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 praktičnih i učinkovitih alata, a također nudi i predloške koda, savjete za kodiranje i generatore koda.

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 Javu

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 Javu. Pored već opisanog NetBeansa u ovom radu biti će 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 [11]:

* Izdanje zajednice (eng. Community Edition): Dostupno je besplatno i uglavnom ga programeri koriste za Javu i Android.
* Komercijalno izdanje (eng. Ultimate Edition): Ultimate edition se mora platiti, no zato ima najnapredniji skup funkcija za razvoj web i desktop aplikacija.

Najvažnije značajke koje karakteriziraju IntelliJ IDEA su pametno dovršavanje koda, analiza koda prilikom pisanja, napredno refaktoriranje, navigacija i pretraga te alati i okviri za podršku [12].

Osim što podržava Javu i mnoštvo Java okvira, IntelliJ IDEA pruža i podršku za ostale programske jezike temeljene na JVM-u kao što su Clojure, Groovy, Kotlin, Scala.

**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. godine i također se može koristiti na Linux, MacOS, Windows i Solaris platformi. Oba izdanja Eclipse IDE-a su opremljena potrebnim funkcionalnostima pomoću dodataka.

Za Java programere koji žele razviti specifičnu funkcionalnost za Eclipse, dostupan je PDE (Plugin Development Environment). Kako bi 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.

**BlueJ** je također integrirano razvojno okruženje za Javu. 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 [13]:

* 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 Javu.
* 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.
* Inovativnost - BlueJ ima nekoliko funkcija koja ranije nisu viđene u drugim IDE-ovima.

Primarni ekran BlueJ-a prikazuje klasnu strukturu aplikacije koja se u tom trenutku razvija omogućujući interaktivno stvaranje i testiranje objekata. Ostale značajke BlueJ-a prilikom razvoja Java programa 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 [14].

Oracle Jdeveloper se može integrirati s Oracle-ovim okvirom za razvoj aplikacija (Oracle ADF) kako bi se dodatno pojednostavio razvoj aplikacija.

Pored Jave, u Jdeveloper-u se mogu koristi i HTML, JavaScript, PHP, SQL i XML [14].

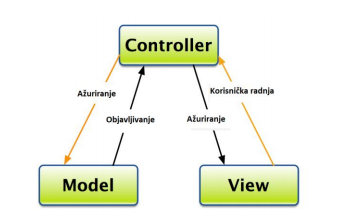
## MVC Arhitektura

MVC (eng. Model-View-Controler tj. Model-prikaz-upravitelj) je softwarski obrazac dizajna koji opisuje specifičan način izrade aplikacija, programa ili sustava, a kao ideja opstala je još od 80-ih godina prošlog stoljeća kada ju je prilikom razvoja programskog jezika SmalTalk80 osmislio norveški znanstvenik (računalnih znanosti) i profesor na Sveučilištu u Oslu- Trigve Reenskaug.

Pored toga što je Reenskaug želio približiti tadašnja računala većem brojem korisnika, također je imao namjeru pojednostaviti razvoj grafičkih sučelja, što je tada predstavljalo poprilično zahtjevan izazov. Međutim, Trigve Reenskaug uočava da se arhitektura programa s grafičkim sučeljem može podijeliti na nekoliko dijelova koji međusobno komuniciraju, ali tako da svaki od njih obavlja različitu zadaću kao npr: Jedan dio može komunicirati s korisnikom putem grafičkog sučelja prezentirajući izlazne, ali i prihvaćajući ulazne podatke od samog korisnika (današnji View), drugi dio može obavljati zadaće vezane za podatke i logiku sustava, programa ili aplikacije (Model), a idealno bi bilo kada bi cjelokupni program sadržavao još jedan dio koji bi služio kao posrednik između ove dvije komponente (današnji Controler) [15].

Iako se MVC arhitekturni obrazac, (kao i programi) značajno razlikuje od MVC-a s kraja prošloga stoljeća, ideja tj. princip je ostao isti; Koristeći se MVC-om proizvođači programske podrške „dijele“ cjelokupan posao aplikacije na tri dijela [15]:

1. Model- sadrži sve podatke potrebne za rad aplikacije, najčešće u obliku baze podataka. Uz podatke, sadrži i samu logiku aplikacije tako što definira što se s podacima može učiniti. U modelu se ne nalazi ulazno-izlazna logika aplikacije.
2. View - Sadrži ulazno-izlaznu logiku aplikacije i uz pomoć View-a su podaci iz modela vidljivi krajnjem korisniku. View po potrebi može sadržavati forme ili polja u koja korisnik može unijeti podatke koji se od njega zatražuju
3. Controler – Njegova zadaća je komunikacija između Modela i Viewa. Controler obavještava Model o promjenama ili događajima koje je korisnik unio/poduzeo preko View-a. Također, ako se u Modelu dogodi određena promjena, Controler će biti zadužen za ažuriranje tih promjena u View-u.



Slika . MVC Arhitektura

Glavne prednosti razvoja aplikacija prema MVC arhitektualnom obrazcu jesu [16]:

1. Olakšano upravljanje razvojem složenih aplikacija zbog mogućnosti posebnog razvoja svake od komponenti aplikacije
2. Olakšano testiranje aplikacije jer je moguće odvojeno testirati pojedine komponente aplikacije i nije uvijek za testiranje potrebno koristiti web poslužitelj (neke komponente se mogu testirati lokalno)
3. 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

Aplikacija za evidenciju radnog vremena razvijena je u programskom jeziku Java korištenjem JavaFX grafičkog sučelja. Aplikacije s grafičkim sučeljem (eng. Graphical User Interface, GUI) omogućavaju korištenje svojih funkcionalnosti putem grafičkih elemenata koji se nazivaju kontrole (eng. Widgets).

JavaFX je skup grafičkih i medijskih paketa koji omogućuju dizajniranje, stvaranje, testiranje te implementaciju aplikacija za radnu površinu. Može biti korišten odvojeno ili integriran unutar razvojnog okruženja. Također, omogućava korištenje CSS-a (eng. Cascading Style Sheet) za oblikovanje, odnosno stiliziranje komponenti grafičkog sučelja.

Aplikacija za evidenciju radnog vremena razvijena je unutar NetBeans integriranog razvojnog okruženja. Grafičko sučelje dizajnirano je korištenjem Scene Builder aplikacije.

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

Baza podataka korištena pri izradi aplikacije je MySQL baza podataka. Navedena baza je otvorenog koda (eng. Open source) te pripada sustavima za upravljanje relacijskim bazama podataka (eng. Relational Database 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 nekom zamišljenom poduzeću. S navedenom problematikom brojna su se poduzeća susretala dugi niz godina, ponajviše one s velikim brojem zaposlenika. U takvim poduzećima, koja 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 dragocjenog 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 nekom zamišljenom poduzeću, 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 koja je određenom korisniku dodijeljena. Aplikacija ima dva tipa korisnika, a to su Admin i User. Korisnik s ulogom Admin ima 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 vlastitih radnih statusa | - | + |
| 4. | Pregled svih radnih statusa | + | - |
| 5. | Unos korisnika | + | - |
| 6. | Brisanje korisnika | + | - |
| 7. | Uređivanje korisnika | + | - |
| 8. | Postavke | - | + |
| 9. | Odjava iz aplikacije | + | + |

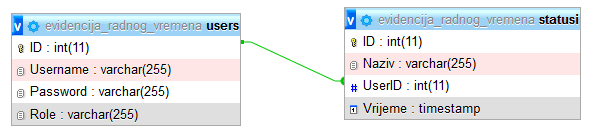
**Tablica 1. Radnje koje korisnici mogu vršiti ovisno o ulozi koju imaju**

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 *Users* povezana je s tablicom *Statusi* relacijom jedan na više preko primarnog ključa tablice *Users* i stranog ključa tablice *Statusi*. To znači da se svaki od statusa u bazi podataka može odnositi isključivo za jednog korisnika, dok istovremeno svaki od korisnika može imati više statusa. Popis tablica, uključujući atirbute i njihove integritete te kratke opise tih atibuta, 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 |

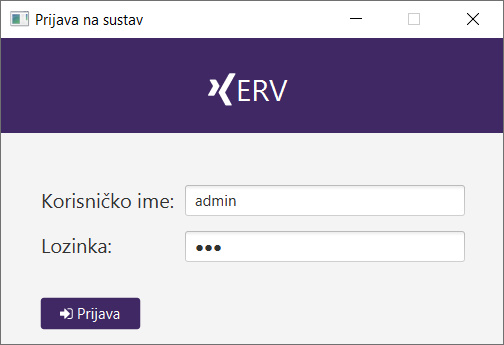
**Tablica 3. - Opis tablice Statusi**

## Prozori aplikacije

U ovom poglavlju su prikazani i objašnjeni svi prozori koji se pojavljuju u aplikaciji opisani kroz pogled prijavljenog korisnika s ulogom Admin, kao i kroz pogled korisnika s ulogom Korisnik. Uz tekstualni opis prozora, prikazane su slike zaslona, 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 23.



Slika 23. Prozor za prijavu korisnika

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

U slučaju neispravnog unosa, na zaslonu 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.

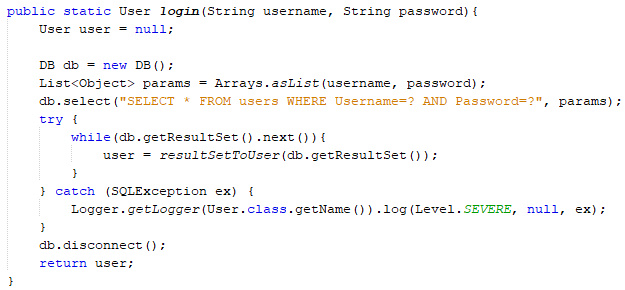
Prilikom ispravnog unosa korisničkog imena i lozinke, provjerava se korisnikova uloga te se temeljem dobivenog podatka, otvara odgovarajući prozor. Za korisnika s ulogom Admin otvara se prozor *Pregled Radnih Vremena-Admin*, a za korisnika s ulogom User otvara se prozor *Početna-Korisnik*.

Kod koji se izvršava klikom na gumb *Prijava* prikazan je na slici 24.



Slika 24. 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 25. Ostvaruje se konekcija nad bazom podataka te se izvršava jednostavan SELECT upit za dohvat korisnika. Rezultat dohvata sprema se u klasu *User*.



Slika 25. Dohvaćanje korisnika iz baze podataka

### Pregled radnih vremena - Admin

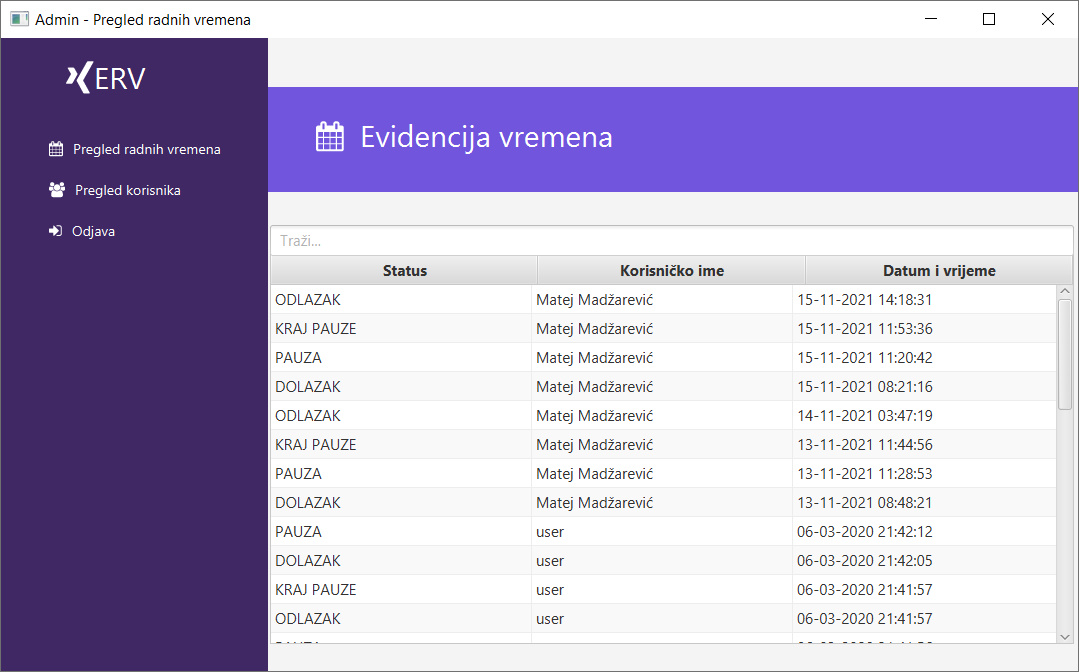
Nakon uspješne prijave korisnika s ulogom Admin, otvara se prozor u kojem su, kroz tablični prikaz, vidljiva sva radna vremena koja su korisnici unijeli u sustav.

U prozoru za pregled radnih vremena prikazana su tri stupca: status, korisničko ime te datum i vrijeme. Stupac *Status* prikazuje status kojeg je pojedini korisnik unio u sustav, stupac *Korisničko ime* prikazuje ime korisnika za kojeg se prikazuje status, a stupac *Datum i vrijeme* prikazuje datum i vrijeme kada je status unesen u sustav.

S lijeve strane nalazi se navigacijski izbornik s tri opcije:

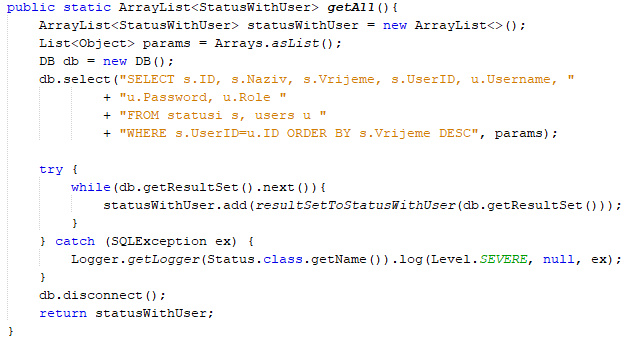
* *Pregled radnih vremena* – trenutni prozor u kojem su prikazana sva radna vremena evidentirana u aplikaciji
* *Pregled korisnika* – otvara se prozor u kojem Admin može pregledati sve korisnike u aplikaciji uz mogućnost izmjene i brisanja postojećih ili dodavanja novih korisnika
* *Odjava* – odjava iz aplikacije i povratak na prozor za prijavu

Prozor, *Pregled radnih vremena*, koji se otvara prilikom uspješne prijave u aplikaciju s Admin ulogom prikazan je na slici 26.



Slika 26. Pregled radnih vremena- Admin

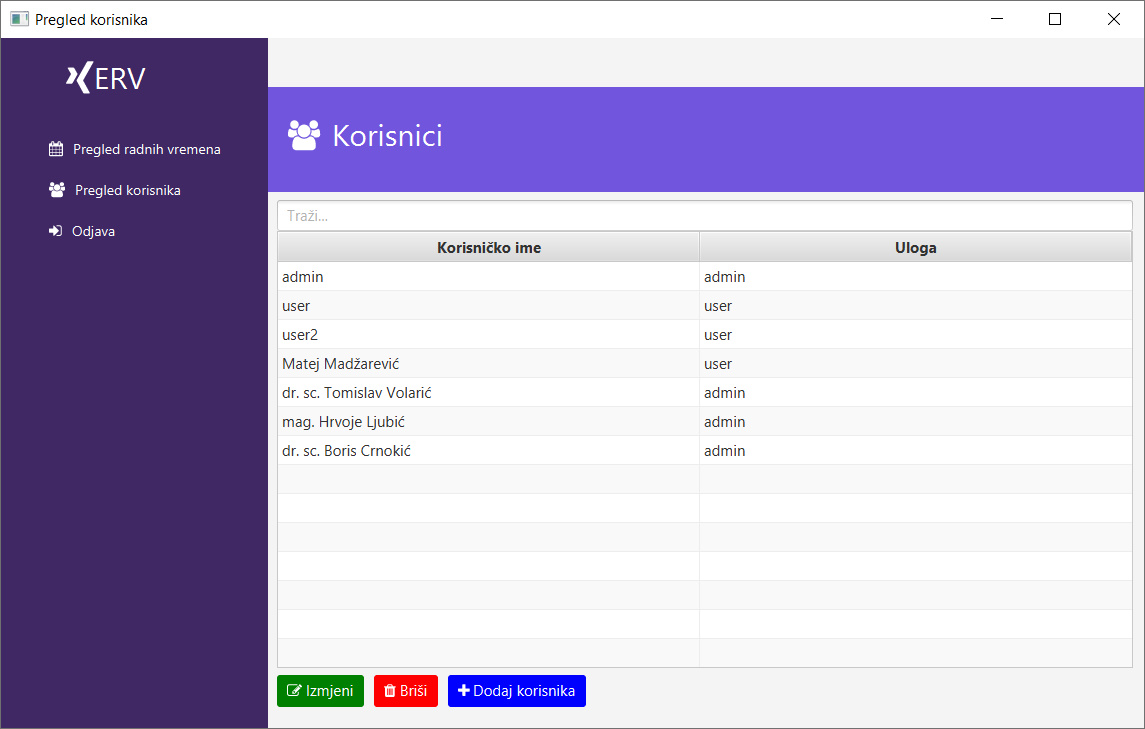
Isječak koda za dohvat svih korisnika u bazi podataka prikazan je na slici 27. Svi dohvaćeni korisnisi spremaju se u listu.



Slika 27. Spremanje dohvaćenih korisnika u listu

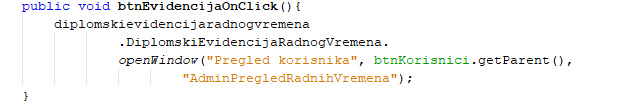
### Pregled korisnika- Admin

Odabirom opcije *Pregled korisnika* u navigacijskom izborniku, otvara se prozor *Pregled Korisnika-Admin*. Na desnom dijelu prozora prikazana su korisnička imena i uloge svih korisnika koji su pohranjeni u bazi podataka. Na donjem dijelu prozora se nalaze tri gumba *Izmjeni*, *Briši* i *Dodaj korisnika* pomoću kojih korisnik s ulogom Admin može uređivati i brisati postojeće korisnike ili dodavati nove korisnike. Izgled prozora *Korisnici* prikazan je na slici 28.



Slika 28. Pregled korisnika- Admin

Kod koji se izvršava klikom na opciju *Korisnici* prikazan je na slici 29.



Slika 29. Otvaranje prozora "Pregled korisnika"

### Dodaj korisnika - Admin

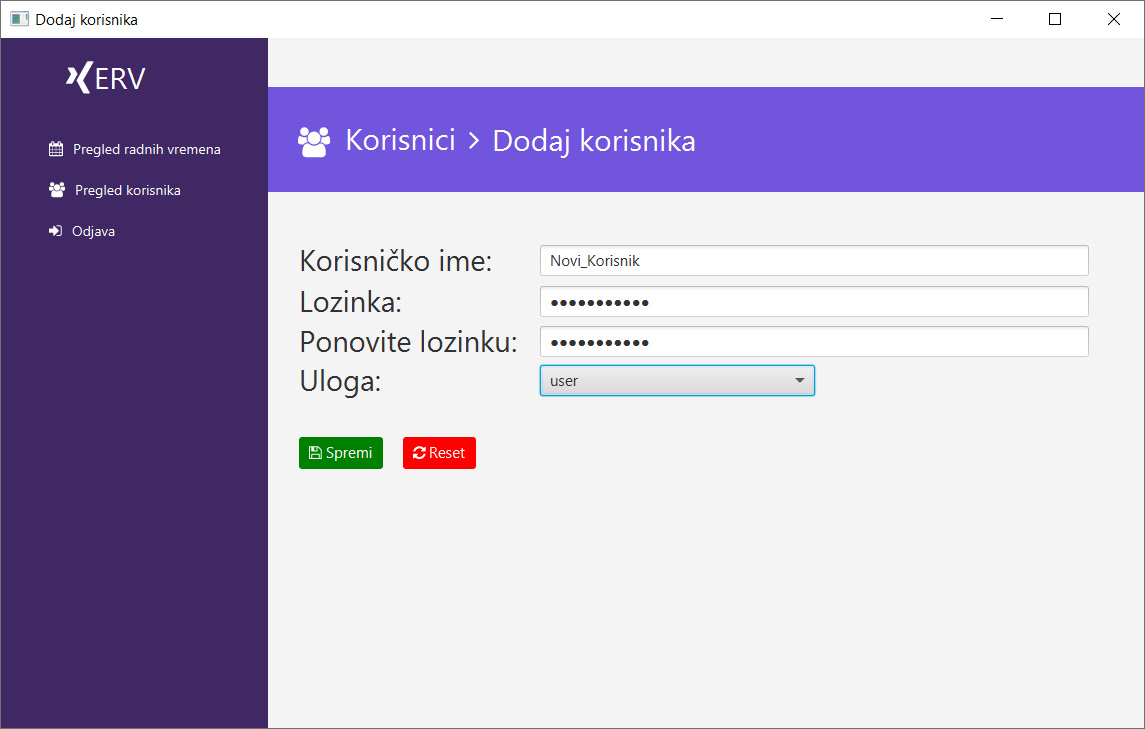
Pritiskom na gumb *Dodaj korisnika* koji se nalazi na prozoru *Pregled korisnika- Admin*, otvara se novi prozor za unos podataka o novom korisniku. Potrebno je unijeti korisničko ime, lozinku i ponovljenu lozinku te s padajućeg izbornika odabrati ulogu novog korisnika, koja može biti Admin ili User. 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 zaslonu 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 su podaci pravilno uneseni, na zaslonu će se pojaviti poruka *Korisnik uspješno unesen!*.

Prozor za dodavanje novog korisnika prikazan je na slici 30.



Slika 30. Dodaj korisnika- Admin

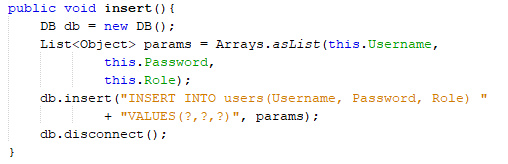
Kod koji se izvršava klikom na gumb *Spremi* prikazan je na slici 31.



Slika 31. Spremanje korisnika

Na slici 32. prikazana je metoda za unos korisnika u bazu podataka. Unutar metode kreira se SQL (eng. Structured Query Language) upit za unos podataka u bazu – INSERT INTO. Koristi se objekt klase PreparedStatement kako se upit ne bi direktno kreirao s korisničkim unosima. Također, kreirani upit se ne prevodi 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 32. Spremanje korisnika u bazu podataka

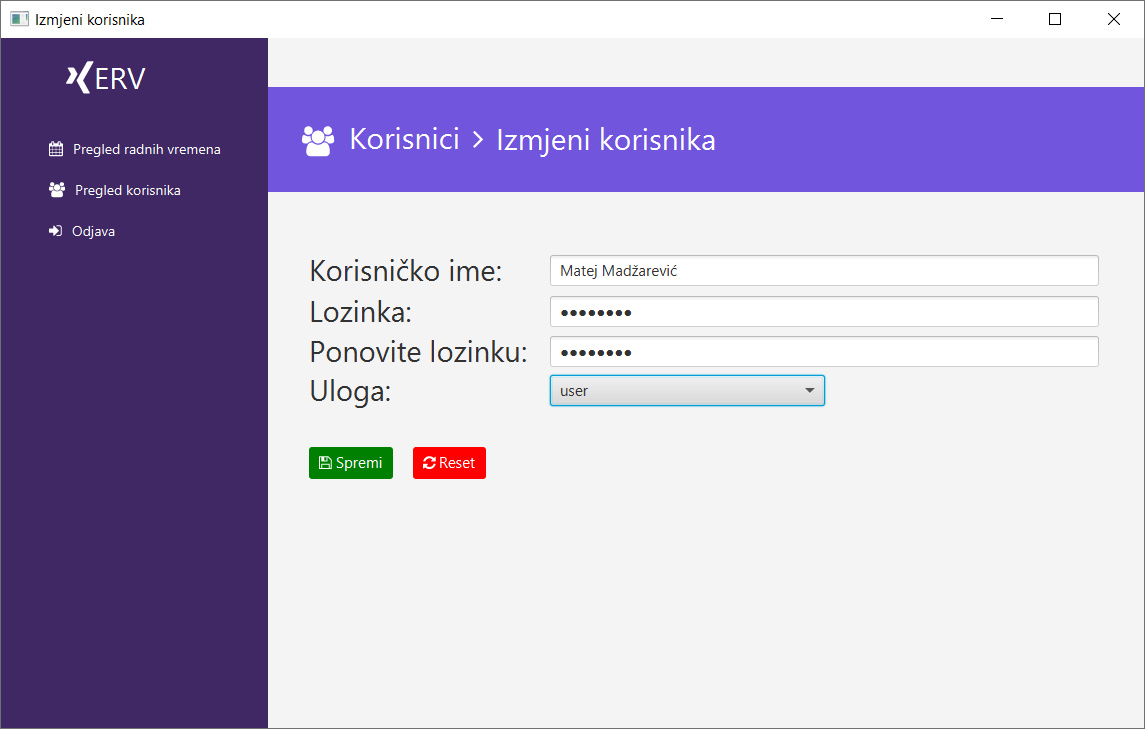
### Izmjeni korisnika - Admin

Podatke o postojećim korisnicima moguće je izmijeniti pritiskom na gumb *Izmjeni korisnika* koji se nalazi u prozoru *Pregled korisnika-Admin*. Važno je napomenuti da Admin prvo mora iz tablice s korisnicima, odabrati korisnika čije podatke želi izmijeniti. Klikom na gumb *Izmijeni* otvara se novi prozor koji sadrži podatke o korisničkom 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 zaslonu 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 zaslonu će se pojaviti poruka *Izmjene uspješno spremljene!*. Izgled prozora za izmjenu korisnika prikazan je na slici 33.



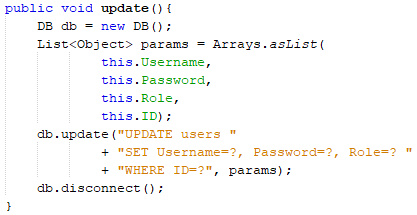
Slika 33. Izmjeni korisnika- Admin

Kod koji se izvršava klikom na gumb *Spremi* prikazan je u na slici 34:



Slika 34. 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 na slici 35. Unutar metode kreira se SQL upit za ažuriranje koristeći ključnu riječ UPDATE.



Slika 35. 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 zaslonu prikazana su četiri gumba za evidenciju vremena: *Dolazak*, *Pauza*, *Kraj pauze* i *Odlazak*. Početni zaslon koji se pojavljuje nakon prijave korisnika s ulogom User prikazan je na slici 36.

Klikom na gumb *Dolazak*, korisnik u bazu podataka unosi status koji označava dolazak na posao tj. početak radnog vremena.

Klikom na gumb *Pauza* korisnik unosi status koji označava početak korištenja pauze.

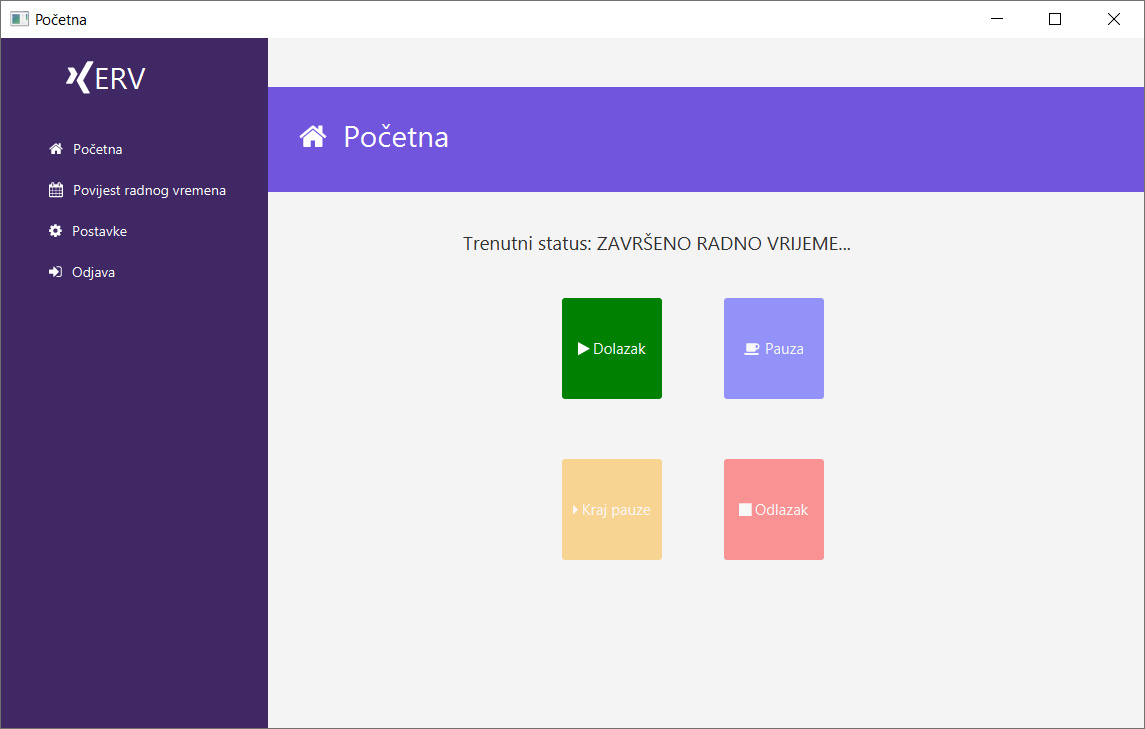
Klikom na gumb *Kraj pauze* korisnik unosi vrijeme završetka pauze i povratka na posao.

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* – trenutni prozor za unos statusa u bazu podataka
* *Povijest radnog vremna* – prozor za prikaz evidencije vremena za prijavljenog korisnika
* *Postavke* – prozor za izmjenu korisničkog imena ili lozinke
* *Odjava* – odjava iz aplikacije



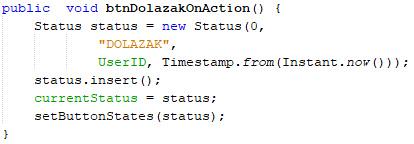
Slika 36. Početna- Korisnik

Kod za isključivanje pojedinog gumba prikazan je na slici 37. 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…*



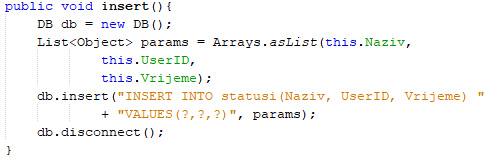
Slika 37. Izmjena radnog statusa

Primjer koda koji se izvršava klikom na gumb „Dolazak“ prikazan je na slici 38.



Slika 38. Pritisak gumba "Dolazak"

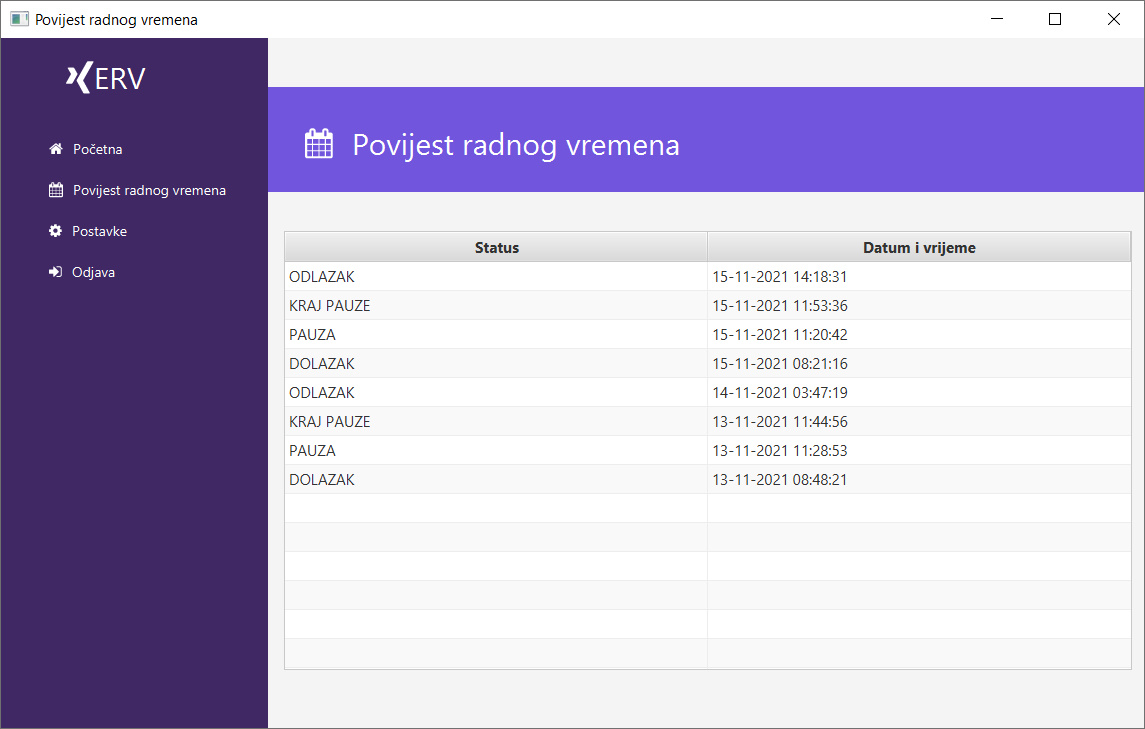
Unutar gore navedene metode poziva se metoda insert() smještena unutar klase *Status* pomoću koje se status unosi u bazu podataka. Kod koji se izvršava pozivom metode insert() prikazan je na slici 39.



Slika 39. Unos statusa u bazu podataka

### Povijest radnog vremena- Korisnik

Odabirom opcije *Povijest radnog vremena* prikazuje se, u tabličnom prikazu, povijest evidencije radnog vremena onog korisnika koji je trenutno prijavljen u aplikaciji. Tablica sadrži dva stupca, *Status* i *Datum i vrijeme,* koje predstavljaju unesene statuse i vrijeme kada su ti statusi uneseni. Prikaz prozora povijest radnog vremena prikazan je na slici 40.



Slika 40. Povijest radnog vremena-korisnik

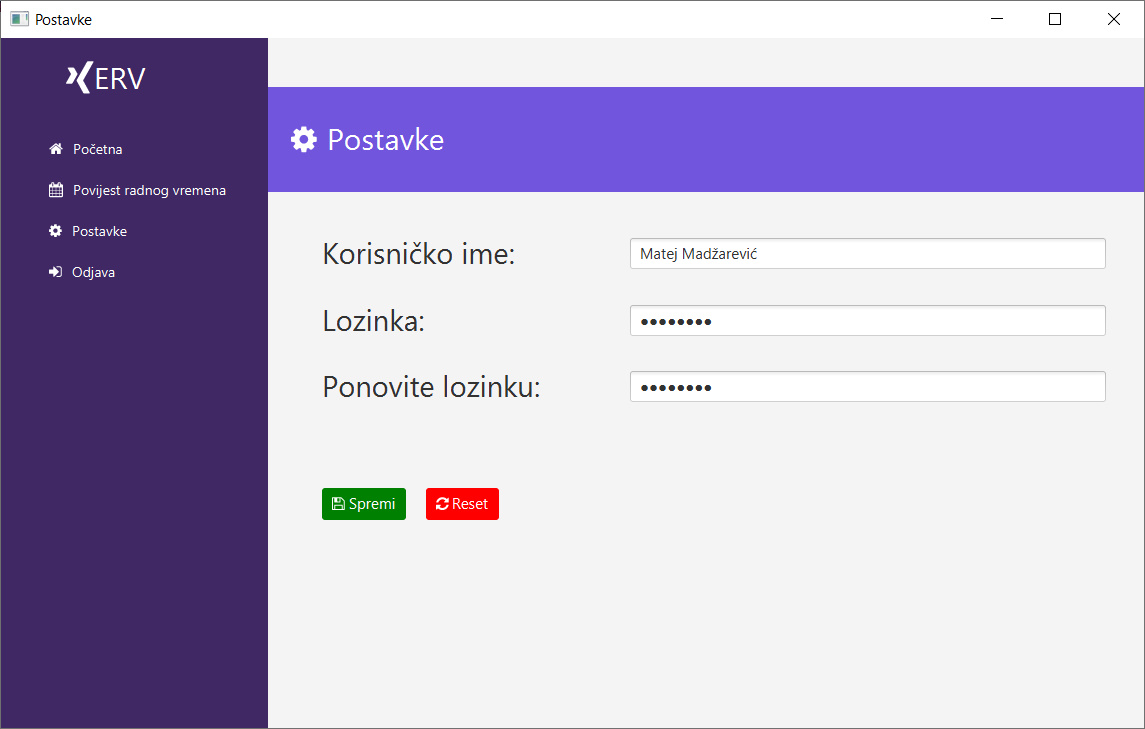
### Postavke - Korisnik

Pritiskom na opciju *Postavke* otvara se prozor 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 zaslonu 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 prozora koji se otvara klikom na opciju *Postavke* prikazan je na slici 41.



Slika 41. Postavke- korisnik

Kod koji se izvršava klikom na gumb spremi prikazan je na slici 42. Ako su korisnički podaci uspješno izmijenjeni, prikazat će se tekst zelene boje, u protivnom će se ispisati tekst crvene boje.



Slika 42. Spremanje postavki

## MVC arhitektura u aplikaciji

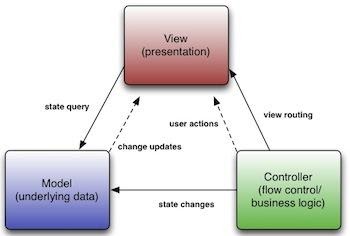
JavaFX aplikacije temelje se na MVC arhitekturi. Navedena arhitektura odvaja slojeve aplikacije koji predstavljaju podatke (eng. Model), grafičko sučelje (eng. View) i samu programsku logiku aplikacije (eng. Controller). Takva arhitektura koristi se i u ostalim programskim jezicima uz mala odstupanja.

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.

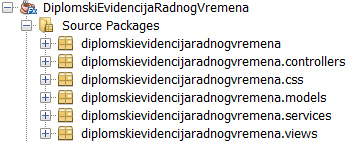
Programska logika aplikacije sadrži klase u kojima se povezuju elementi grafičkog sučelja, primjerice, gumba, s metodama koje sadrže logiku koja se mora izvršiti tijekom interakcije korisnika s tim elementima (primjerice, pritisak gumba).

Izgled MVC arhitekture prikazan je na slici 43.

****

Slika 43. MVC arhitektura

Postoje dva načina strukturiranja aplikacije – prema slojevima (eng. Package by layer) i prema značajkama (eng. Package by 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 44.



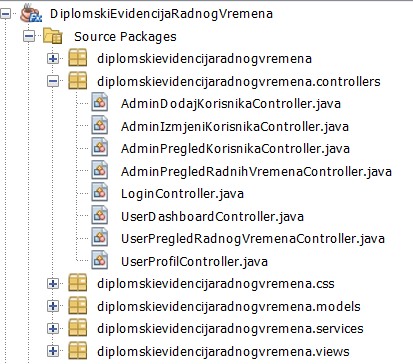
Slika 44.Struktura aplikacije

Kao što je vidljivo na slici 44. unutar aplikacije kreirana su pet paketa:

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

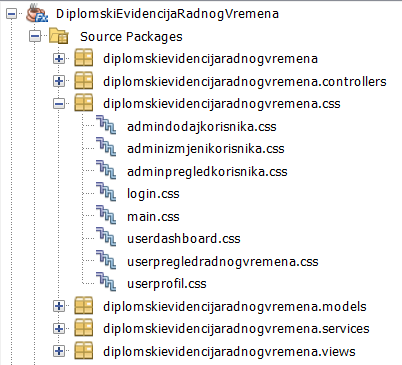
Postoji i vršni paket **diplomskievidencijaradnogvremena** u kojem se nalazi klasa DiplomskiEvidencijaRadnogVremena.java koja predstavlja početnu točku aplikacije, te klasa DB.java koja sadrži logiku za otvaranje i zatvaranje konekcije nad bazom podataka.

Unutar paketa **diplomskievidencijaradnogvremena.controllers** smještene su sve controller klase u kojima se nalazi programska logika sustava. Datoteke unutar controllers paketa prikazane su na slici 45.



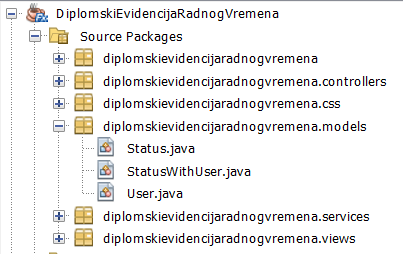
Slika 45. Datoteke unutar controllers paketa

Unutar paketa **diplomskievidencijaradnogvremena.css**nalaze se CSS (eng. Cascading Style Sheets) 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 css prikazane su na slici 46.



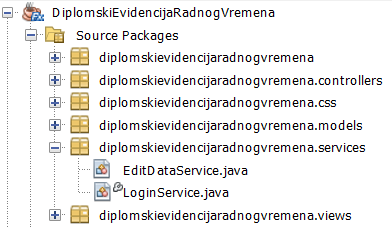
Slika 46. Datoteke unutar css paketa

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 47.



Slika 47. Datoteke unutar models paketa

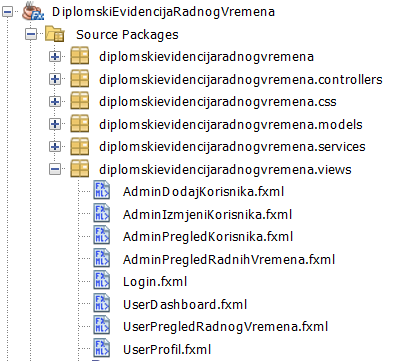
Unutar paketa **diplomskievidencijaradnogvremena.services** nalaze se klase za ažuriranje postojećeg korisnika i za odjavu iz aplikacije. Datoteke koje se nalaze unutar paketa services prikazane su na slici 48.



Slika 48. Datoteke unutar services paketa

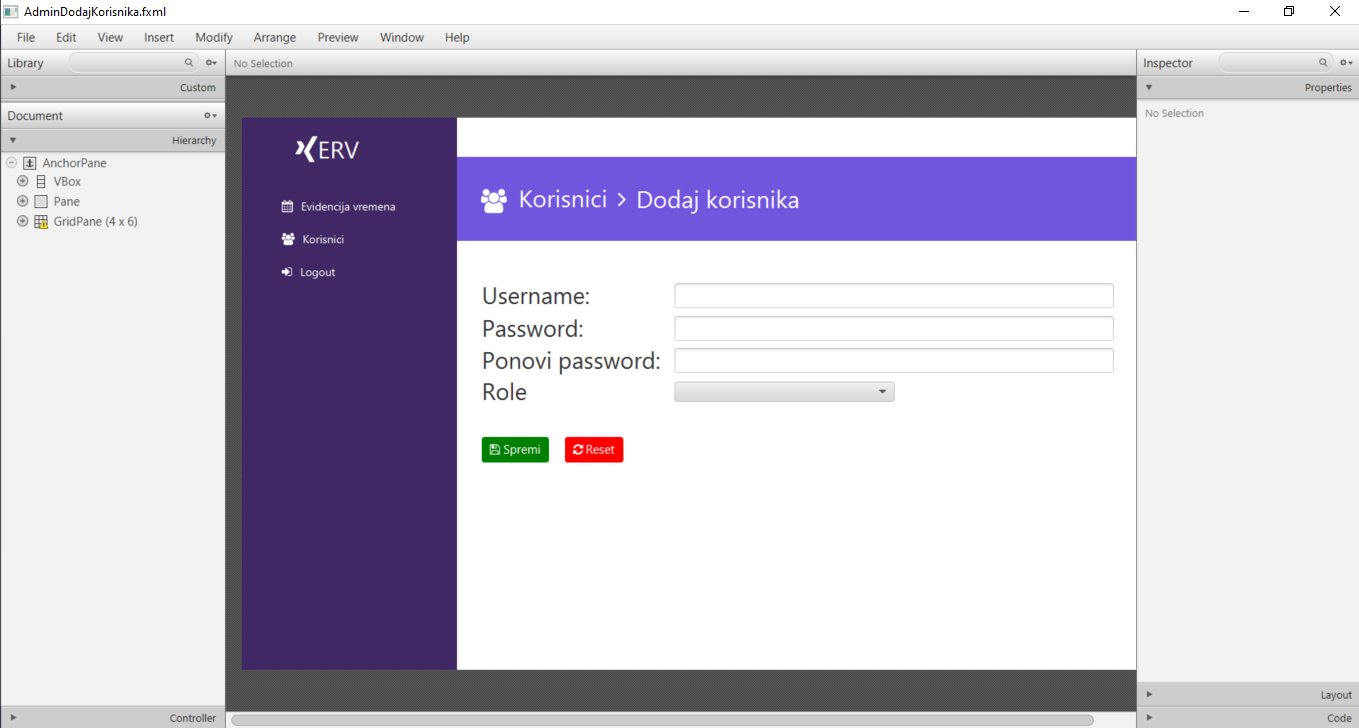
Unutar paketa **diplomskievidencijaradnogvremena.views** nalaze se datoteke s ekstenzijom „.fxml“. Svaka datoteke unutar paketa views ima dodijeljenu CSS datoteku kojom su komponente stilizirane te kontroler klasu koja sadrži programsku logiku. Primjerice, datoteka pod nazivom AdminDodajKorisnika.fxml ima dodijeljenu CSS datoteku pod nazivom admindodajkorisnika.css te kontroler klasu AdminDodajKorisnika.controller.

Sadržaj paketa views prikazan je na slici 49.



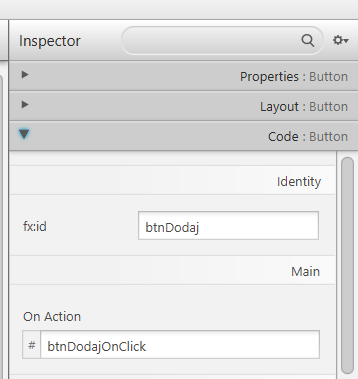
Slika 49. Datoteke unutar views paketa

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 50.

****

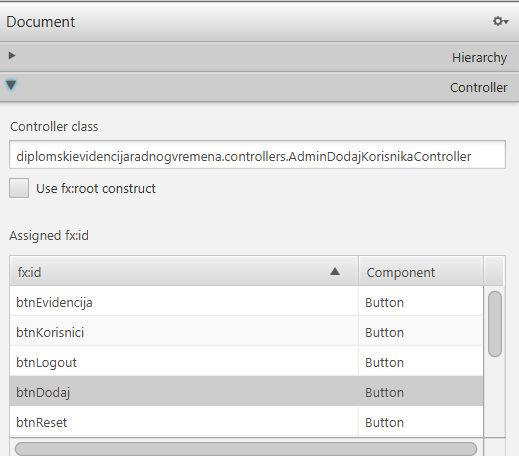
Slika 50. AdminDodajKorisnika.fxml datoteke unutar Scene Builder aplikacije

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 51.



Slika 51. 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 otvorenoj fxml datoteci. Primjer dodijeljivanja klase za datoteku AdminDodajKorisnika.fxml prikazan je na slici 52.

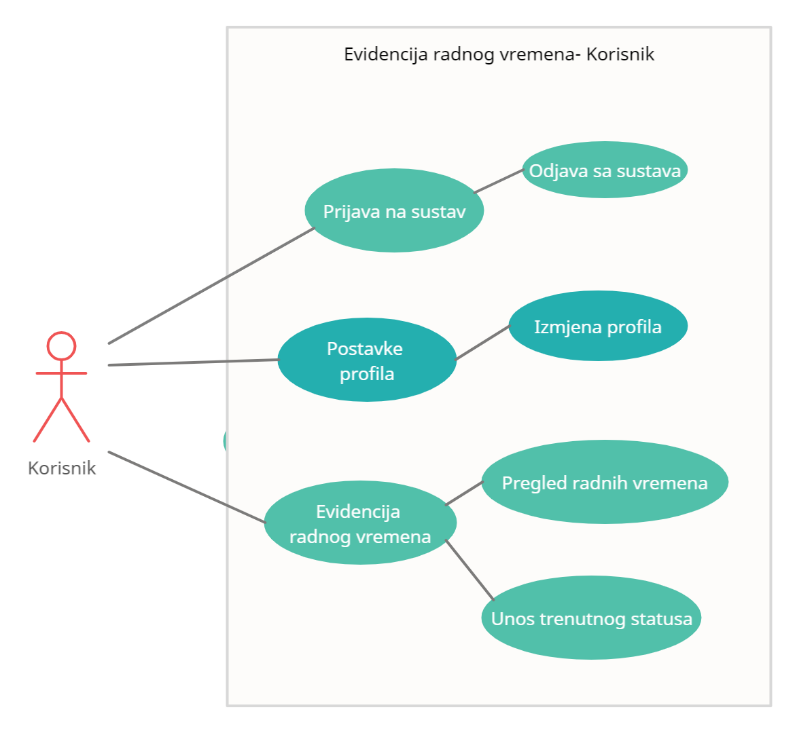


Slika 52. Dodjeljivanje controller klase

## Slučajevi korištenja

Java desktop aplikacija za evidenciju radnog vremena razlikuje dvije razine korisničkog sučelja – korisničko sučelje za korisnike s ulogom Admin 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 53. 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, postavki profila koje podrazumijevaju izmjenu korisničkog imena i lozinke profila te funkcionalnost evidencije radnog vremena. Funkcionalnost evidencije radnog vremena uključuje pregledavanje svih dosadašnjih evidencija radnog vremena i unošenje trenutnog statusa na poslu, odnosno dolaska, odlaska i pauze na poslu.

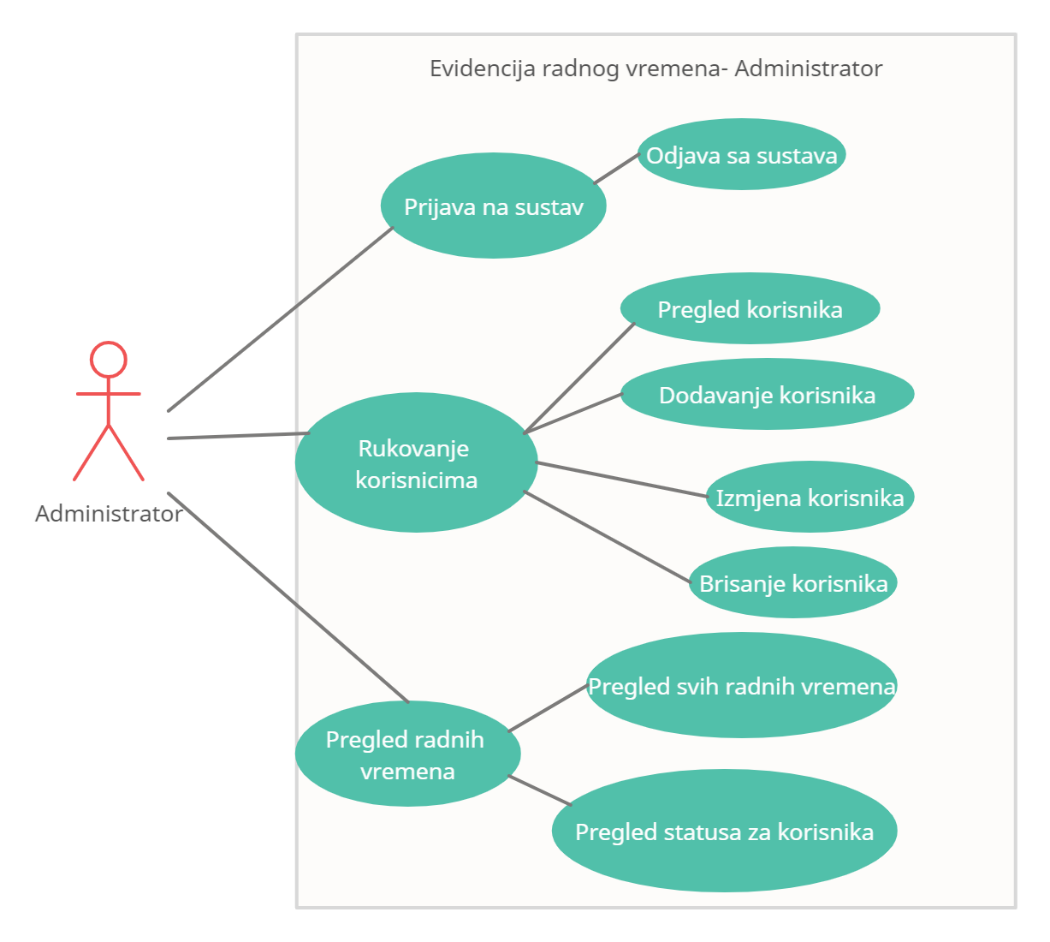


Slika 53. Slučajevi korištenja - Korisnik

Na slici 54. se nalaze slučajevi korištenja za korisnike s ulogom Admin. Kao i kod korisnika s ulogom user i administratori imaju funkcionalnost prijave u sustav, a samim time i odjave iz sustava.

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.

Admina funkcionalnost pregleda radnih vremena podrazumijeva pregled cjelokupne evidencije u aplikaciji i pregled evidencije za pojedinog korisnika.



Slika 54. 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 četiri pred-definirane vrijednosti.

Trenutno implementirana desktop aplikacija zamišljena je da bude na ulazu u tvrtku, 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 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 (eng. Representational state transfer*)* arhitekturu. REST arhitektura koristi HTTP (eng. Hypertext Transfer Protocol), a podaci se šalju i primaju u JSON (eng. Java Script Object Notation) 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 (eng. 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 Jave poput jednostavnosti, sigurnosti, besplatnosti itd.

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

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 nekom zamišljenom poduzeću. 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] M. Topolnik and M. Kušek, “Uvod u programski jezik Java,” 2008.

[2] Holmes J. Barry and Joyce T Daniel, *Object-oriented Programming with Java*. Jones and Bartlett, 2001.

[3] J. Caron, “Java: Status Report and Language Overview,” 1995.

[4] Minnick Chris, *Coding All-in-One For Dummies*. Wiley, 2022.

[5] Campione Mary, Walrath Kathy, and Huml Alison, *The Java Tutorial*, Treće izdanje. Addison-Wesley, 2001.

[6] Gupta Vikas, *Comdex Computer Programming Course Kit*. Dreamtech Press, 2002.

[7] S. Khoirom, S. Moirangthem, B. Laikhuram, J. Laishram, and S. Davidson, “Comparative Analysis of Python and Java for Beginners,” *International Research Journal of Engineering and Technology*, 2020, [Online]. Available: www.irjet.net

[8] Lowe Doug, *Java All-in-One For Dummies*. 2017.

[9] Sarin Ashish and Sharm J, *Getting started with Java programming language*. CreateSpace Independent Publishing Platform, 2017.

[10] Kumar Niranjan, *Java from Plinth to Paramount*. 2019.

[11] Samoylov Nick, *Learn Java 17 Programming*. Packt Publishing, 2022.

[12] Orsine Assumpção Hudson, *Getting Started with IntelliJ IDEA*. Birmingham: Packt Publishing Ltd, 2013.

[13] Banerjee Kunal, *ICSE Simplified Java With Bluej for Class X*. Educreation Publishing, 2017.

[14] Coxall Malcolm, *Oracle Quick Guides - Part 3 - Coding in Oracle: SQL and PL/SQL*. 2013.

[15] R. Conery, Hanselman Scott, Haack Phil, and Guthrie Scott, *Professional ASP.NET MVC 1.0*. Wiley, 2009.

[16] Chiaretta Simone and Nayyeri Keyvan, *Beginning ASP.NET MVC 1.0*. Wiley, 2009.