**Sveučilište u Mostaru**

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

Matej Madžarević

**JAVA DESKTOP APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA**

Diplomski rad

Mostar, 2020.

**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, prosinac 2020.

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](#_Toc57590221)

[2. PROGRAMSKI JEZIK JAVA 3](#_Toc57590222)

[2.1. Osnovne karakteristike Java jezika 4](#_Toc57590223)

[2.2 Prednosti i nedostaci Java jezika 6](#_Toc57590224)

[2.3. Naredbe u Javi 10](#_Toc57590225)

[2.4. Java platforma 22](#_Toc57590226)

[2.5. Razvojno okruženje Netbeans 25](#_Toc57590227)

[2.6. Ostala razvojna okruženja 28](#_Toc57590228)

[2.7. MVC Arhitektura 31](#_Toc57590229)

[3. JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA 33](#_Toc57590230)

[3.1. Opis problema 33](#_Toc57590231)

[3.2. Relacijski model baze podataka 34](#_Toc57590232)

[3.3 Dijagram klasa 35](#_Toc57590233)

[3.4. Prozori aplikacije 37](#_Toc57590234)

[3.4.1.Login 37](#_Toc57590235)

[3.4.2. Evidencija vremena - Admin 39](#_Toc57590236)

[3.4.3. Korisnici - Admin 40](#_Toc57590237)

[3.4.4. Dodaj korisnika - Admin 44](#_Toc57590238)

[3.4.5. Izmjeni korisnika - Admin 45](#_Toc57590239)

[3.4.6. Početna - Korisnik 46](#_Toc57590240)

[3.4.7. Evidencija vremena - Korisnik 50](#_Toc57590241)

[3.4.8. Postavke - Korisnik 51](#_Toc57590242)

[3.5. MVC arhitektura u aplikaciji 51](#_Toc57590243)

[3.6. Slučajevi korištenja 56](#_Toc57590244)

[3.7. Mogućnosti proširenja aplikacije 57](#_Toc57590245)

[4. ZAKLJUČAK 59](#_Toc57590246)

[5. LITERATURA 60](#_Toc57590247)

**SAŽETAK**

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

Cilj ovog rada bio je implementirati desktop aplikaciju za evidenciju radnog vremena koristeći Java programski jezik za implementaciju desktop aplikacije.

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

Bitno je naglasiti kako je aplikacija urađena po MVC (Model-View-Controler, Model-Pogled-Upravitelj) obrascu softwarske arhitekture koji služi kao alat koji uvelike olakšava brzi razvoj računalnog programa (software-a), 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, desktop aplikacija, evidencija radnog vremena, MVC arhitektura

# UVOD

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

Kroz povijest su se zaposlenici susretali s raznim oblicima evidencije radnog vremena, sve od klasičnog potpisivanja olovkom na papir pri dolasku na posao, preko kartica koje se provlače na ulasku u poslovne prostore pa sve do današnjih najmodernijih sustava koje imaju uspješne i velike 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 bio, kao što je već navedeno u sažetku, implementirati jednostavnu Java desktop aplikaciju uz pomoć koje je moguće efikasno evidentirati radna 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 na dugme koje označava početak radnog vremena, a aplikacija će im (u realnom vremenu) odbrojavati vrijeme provedeno na poslu.

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

Naravno, na kraju radnog vremena, zaposlenici bi imali mogućnost jednostavnim klikom na dugme označiti završetak radnog vremena. Nakon toga, svaki zaposlenik bi kroz aplikaciju, imao 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 i zato 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, Java platforma te razvojna okruženja, a u posebnom odlomku detaljnije je objašnjen i MVC arhitektonski obrazac koji olakšava razvoj aplikacija.

U drugom dijelu rada opisana je Java desktop aplikacija razvijena u Netbeans razvojnom okruženju, relacijski model baze podataka, dijagram klasa te prozori aplikacije. Također su u posebnom odlomku drugoga dijela navedeni i objašnjeni paketi koji se nalaze unutar same aplikacije, odnosno paketi koji podrazumijevaju modele, poglede i upravitelje. U zasebnom odlomku su uz pomoć slika i teksta također 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 pa su stoga u posebnom, ujedno i posljednjem odlomku rada, navedene i objašnjene mogućnosti proširenja aplikacije.

# PROGRAMSKI JEZIK JAVA

Java je objektno orijentirani programski jezik, a pokreće se na više od 3 milijarde uređaja. Danas se koristi za razvoj mobilnih aplikacija, web aplikacija, desktop aplikacija, igara i još mnogo toga. Java je računalna platforma i programski jezik visokog nivoa.

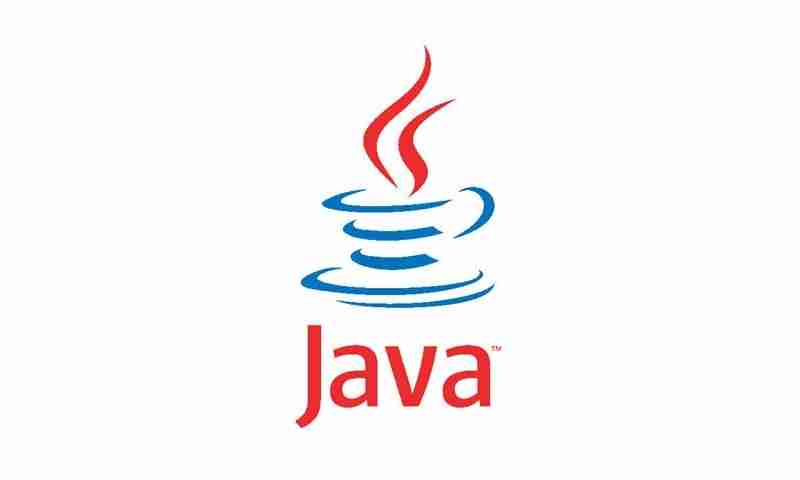
Razvijen je od strane James Gosling-a, Patrick Naughton-a i ostalih inženjera u kompaniji Sun Microsystems koji su 1991. godine počeli raditi na projektu novog programskog jezika (Green project). Cilj projekta je bio razviti programski jezik namijenjen za male uređaje (poput preklopnika za televizor)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

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

Prvi Javin program se zvao \*7 i predstavljao je program za inteligentni daljinski upravljač za televizor. Nakon toga su se nizali različiti programi za usluge kabelske televizije. 1994. godine kompanija Sun razvija web preglednik nazvan HotJava- prvi program koji je imao mogućnost izvršavanja Javinih programa (apleta) unutar web preglednika[[3]](#footnote-3).

Projekt je konačno objavljen u studenom 1995. godine[[4]](#footnote-4) kada je izašla probna verzija programskog jezika (beta verzija). Početkom 1996. godine izlazi službena verzija Java platforme i Java programskog jezika, a iste godine Javu 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 Java programskom jeziku.

Danas je prema podacima Git Hub-a Java treći program po zastupljenosti među programerskom populacijom i dosada je mijenjan kroz svojih 15-ak verzija pa je tako najnovija verzija Jave (Java 15) izašla studenom 2020. godine.



Slika 1 Java logo

Java programi se mogu izvoditi na sljedećim operacijskim sustavima pod uvjetom da je na njima instaliran Java virtualni stroj (engl. Java virtual machine, skraćeno JVM):

* Windows OS
* Linux
* Solaris
* MacOS

## 2.1. Osnovne karakteristike Java jezika

Objektivno rečeno Java je relativno nov i moćan programski jezik. Svoje prednosti je naslijedio od C ++ uklonivši pri tom i neke nedostatke koje je C++ imao.[[5]](#footnote-5) Iz tog razloga su sintaksa i programska logika Jave vrlo slične onoj u C ++. Java je jednostavna, orijentirana na objekte, prenosiva, robusna i višestruka. Ne postoje datoteke zaglavlja, aritmetika pokazivača (ili čak sintaksa pokazivača), strukture, sindikati, preopterećenje operatora i virtualne osnovne klase. Odsustvom pokazivača se u Javi također izbjegavaju sigurnosni problemi.

Glavna razlika između Jave i C++ je u području nasljeđivanja klasa. C++ podržava višestruko nasljeđivanje dok Java podržava samo pojedinačno nasljeđivanje. Neovisna je o platformi (prenosiva je). Jednom kada se Java kôd sastavi u bajtni kôd, pokreće se na svakom operacijskom sustavu koji na sebi ima Java virtualni stroj. Ovo je jedna od najvećih prednosti Jave u odnosu na C ++.

Java kompajler otkriva mnoge probleme tokom prevođenja. Neki od ovih problema otkriveni su za vrijeme pokretanja kod drugih prevoditelja jezika. To Javu čini robusnim (snažnim) jezikom.[[6]](#footnote-6)

Java koristi snažno upravljanje memorijom. U Javi postoji automatsko skupljanje otpadaka (engl. automatic garbage collection). Također, u Javi postoje mehanizmi rukovanja izuzecima (exception handling) i provjere tipa (type checking). Sve ove stavke čine Javu robustnom.[[7]](#footnote-7)

Java bytecode je moguće prenijeti na bilo koju platformu, a također Java je brža od tradicionalne interpretacije pošto je bytecode "blizak" izvornom kodu, ali je ipak nešto sporija od kompajliranih jezika (npr. C++).

Nit (engl. thread) je poput zasebnog programa koji se izvršava paralelno. Moguće je pisati Java programe koji rade na mnogim zadacima odjednom tako što se definiraju višestruke niti. Glavna prednost višenitnosti (engl. multi-threadinga) je dijeljene iste memorije. Niti su važne za multimediju, Web aplikacije itd.[[8]](#footnote-8)

Java je također višestruka. Ova karakteristika Javi pruža bolje ponašanje u stvarnom vremenu.[[9]](#footnote-9)

Java jezik izvorno je bio namijenjen za uporabu u malim, ugrađenim uređajima.[[10]](#footnote-10) Javina je prva uporaba na webu došla u obliku apleta. Apleti su manji dijelovi programskog koda koji se odvijaju pod nadzorom web-preglednika. Donedavno se Java nije koristila u ozbiljnom web poslužiteljskom razvoju. Sada, s poboljšanjima u Java aplikacijskim programskim sučeljima (engl. Application programming interfaces, skraćeno, API) a posebice zahvaljujući Servlet API-u, Java postaje važan alat za web programiranje na strani poslužitelja.

U početku je Java bila vrlo slična C i C ++, ali je pružila mnoge dodatne funkcionalnosti. Dakle, prethodno znanje ovih jezika bit će pozitivna točka tijekom učenja Jave.

Moguće je navesti mnogo karakteristika Jave koje doprinose jednostavnosti i razumljivosti prilikom korištenja. Najvažnije od tih karakteristika su:

* jednostavnost
* objektna orijentiranost
* platformna neovisnost
* sigurnost
* robustnost
* arhitekturna neutralnost
* prenosivost
* dinamičnost
* interpretiranost
* visoke performanse
* višenitnost (Multithread)
* distribuiranost

Prema kompaniji Sun, jezik Java je jednostavan zato što:

* ima sintaksu koja je zasnovana na C++ (pa je programerima lakša za naučiti nakon C++).
* je uklonio mnogo zbunjujućih i/ili rijetko korištenih karakteristika npr. eksplicitne pokazivače, preopterećenje operatora itd.
* nema potrebe za uklanjanjem nereferenciranih objekata jer sadrži tzv. Automatsko sakupljanje smeća (Automatic Garbage Collection).

Java je sigurna zato što:

* nema eksplicitnih pokazivača,
* programi se pokreću unutar tzv. “vreće s pijeskom u virtualnoj mašini” (Virtual Machine Sandbox).

## 2.2 Prednosti i nedostaci Java jezika

Na temelju svojih prednosti programski jezik Java je stekao široku popularnost i izgrađene su višestruke konfiguracije za različite tipove platformi, uključujući Java Standard Edition za Macintosh, Windows i UNIX, Java Micro Edition za mobilne aplikacije i za Java Enterprise Edition za Enterprise aplikacije.

Uz rastuću važnost internetskih i mobilnih aplikacija, Java programski jezik je danas temelj za većinu umreženih aplikacija i smatra se korisnim za skriptiranje, igre i mobilne aplikacije.

Prednosti Java programskog jezika su sljedeći:

* Java nudi veću multi-funkcionalnost i bolju prenosivost jer se programi napisani na jednoj platformi mogu izvoditi kroz radne površine, mobitele, ugrađene sustave
* Java je besplatna, jednostavna, objektno orijentirana, distribuirana, podržava višenitnost i nudi multimedijsku i mrežnu podršku
* Budući da je vrlo popularna u poslovnom svijetu, Java ima veliku aktivnu zajednicu korisnika i dostupnu podršku
* Za razliku od C i C ++, Java programi sastavljeni su neovisno o platformi u bytecodeu koji omogućava pokretanje istog programa na bilo kojem stroju na kojem je instaliran JVM
* Java ima moćne razvojne alate poput Eclipse SDK (Software Development Kit-Programski razvojni komplet) i NetBeans koji imaju mogućnost uklanjanja pogrešaka i nude integrirano razvojno okruženje
* Povećavanje jezične raznolikosti, o čemu svjedoči kompatibilnost Jave sa Scale, Groovy, JRuby i Clojure programskim jezikom[[11]](#footnote-11).

Prednost korištenja Jave jeste objektno orijentirano programiranje. Objektna orijentiranost znači da se softver organizira kao kombinacija različitih tipova objekata koji uključuju i podatke i ponašanje. Osnovni koncepti OOP su:

1. Objekt,
2. Klasa,
3. Nasljeđivanje,
4. Polimorfizam,
5. Apstrakcija,
6. Enkapsulacija.

Objektno orijentirano programiranje omogućuje stvaranje modularnih programa i koda za višekratnu upotrebu koji održavaju sustav proširivim i fleksibilnim.

Java nudi API-je za razne aktivnosti kao što su: veza baze podataka, umrežavanje, I / O, XML(Extensible Markup Language- jezik za označavanje podataka) raščlanjivanje, uslužne programe i još mnogo toga.

Veliki broj alata za razvoj otvorenih izvora koji koriste Javu učinili su istu još snažnijom opcijom za programere, tj. Eclipse i Netbeans*.* Izuzetno su jaka razvojna okruženja i efikasno su doprinijela tome da Java postane onakva kakva je danas. Pojednostavljuju kodiranje i pružaju izuzetno moćne mogućnosti za uklanjanje pogrešaka.

Veliki broj knjižnica otvorenog koda i dobra sazrelost s industrijskom podrškom također su osigurali da se Java koristi svuda. Neke od ovih biblioteka doprinijele su mnoge popularne organizacije poput Apache, Google, Yahoo, Facebook, LinkedIn itd., zbog čega je Java još popularnija.

Najvažnija prednost jave jeste što je besplatna. Kada je u pitanju razvoj, danas smo suočeni s intenzivnom konkurencijom. Mnoga mala i srednja poduzeća žele da njihov razvoj softvera i / ili razvoj web aplikacija uradi poslove za njih, ali za to zaista nemaju veliki budžet. Budući da je Java besplatna, čini je isplativom pa se ta poduzeća i odlučuju za Javu.

Konačno, postoji opsežna podrška zajednice koju je Java uspjela prikupiti. To će velikom broju novih Java programera pomoći naučiti umjetnost programiranja, a pomoći će im i da postanu produktivni.

Java, nudi nove značajke kao što su skalabilna i fleksibilna platforma za Internet, manje koda na ploči, nova biblioteka datuma i vremena i API, osvježeni grafički alat, integracija s JavaScript programskim jezikom i drugima.

Uz prednosti, uvijek se nailazi na nedostatke bilo koje platforme. Isto je i u slučaju Jave.

Ti nedostaci su sljedeći:

* Java prevoditelj (eng. compiler) još nije dobro optimiziran u usporedbi s C ++
* Ne postoji odvajanje specifikacija od implementacije
* Upravljanje memorijom u Javi je skupo
* Nedostatak predložaka (eng. template) može ograničiti sposobnost Jave u stvaranju visokokvalitetnih struktura podataka[[12]](#footnote-12)

Ono što treba imati na umu je da su to neki nedostaci koji se nalaze kod većine programskih jezika i nisu jedinstveni samo za Javu. No, informacije o tim nedostacima pomoći će konstruktivnijem korištenju Jave.

Pored nedostataka Java programskog jezika postoji mnogo razloga iz koji bi se trebalo naučiti programski jezik Java. Java programski jezik je izuzetno popularan programski jezik, što samo po sebi povlači činjenicu da osobe koje znaju ovaj programski jezik imaju dosta prilika za posao.

Kada se nauči Java programski jezik lako se može je prijeći na drugi programski jezik kao što su C++ i C. Java je veoma popularan na svim platfomama, zbog svoje portabilnosti. U ranim devedesetim godinama, kada su poslovne aplikacije bile bazirane na C++, koji je ovisio o platformi i koji je prilično kompliciran, Java je došla kao programski jezik koji je uspješno premostio ovu barijeru razvoja poslovnih aplikacija na jednostavniji, objektno-orijentiran i interpretiran način sa podrškom za mutli-treding programiranje.

Za postizanje portabilnosti i nezaovisnosti o platformi, zaslužan je razvoj Java virtualnog stroja (JVM), koji simulira operacijski sustav na virtualnoj razini i na taj način omogućava pokretanje na bilo kojoj platformi. Također, Java uvodi “prikupljač smeća” garabage collection, čime manipulira životnim vijekom objekata.

Sa druge strane, možemo reći da je Java dosta izgubio na popularnosti u poslednjih par godina, usljed uspona drugih programskih jezika, prvenstveno Python-a i JavaScript-a. Veliki minus zajednice i smanjenje populalarnosti Java je stekla nakon sto je Oracle uveo ogromne novčane naknade za licenciranje. Na svu sreću, Java pokušava pratiti Cloud (oblak) tržište kroz GraalVM, ali i kroz razvoj OpenJDK-a, besplatne alternative *Oracle-ovom JDK-ju.* Java je i dalje broj jedan programski jezik kada su u pitanju Enterprise aplikacije.[[13]](#footnote-13)

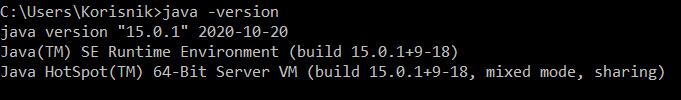
Zaključno, gotovo 20 godina od svog osnutka, Java nastavlja davati značajnu vrijednost svijetu programskog razvoja. [[14]](#footnote-14)

## 2.3. Naredbe u Javi

Program je niz naredbi koje računalo izvršava kako bi obavilo neku zadaću. Da bi računalo moglo izvršavati naredbe, one moraju biti pisane na računalu razumljiv način, tj. u programskom jeziku. Programski jezici se razlikuju od ljudskog po svojoj jasnoći i strogosti što je u programu dozvoljeno, a što nije. Pravila koja određuju što je dozvoljeno zovu se sintaksa jezika. Sintaksna pravila određuju osnovni rječnik jezika i način na koji se programi mogu stvarati koristeći petlje, grananja i potprograme. Sintaksno ispravan program je onaj koji je moguće kompajlirati ili izvršiti. Programi koji sadržavaju sintaksne greške će biti odbačeni uz poruku o grešci. [[15]](#footnote-15)

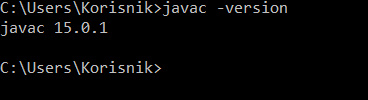
U nastavku su najprije objašnjene neke od osnovnih naredba u Javi koje je moguće unijeti preko terminala. Nakon tih naredbi su u nastavku objašnjene naredbe i ključne riječi koje se obično koriste prilikom kodiranja u integriranom razvojnom okruženju:

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



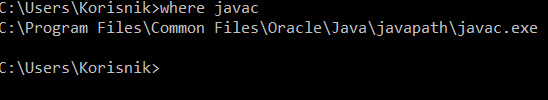
Slika 2 Primjer korištenja naredbe Java –version u terminalu

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



Slika 3 Primjer korištenja naredbe javac –verzija u CMD-u

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



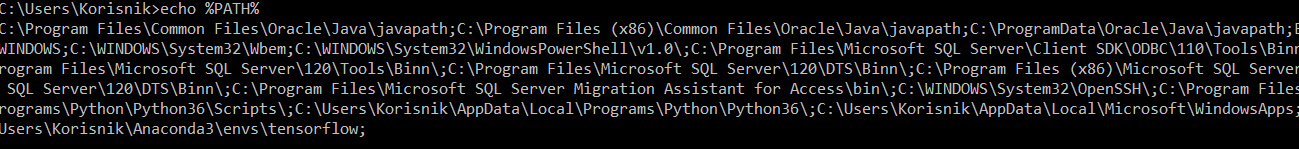
Slika 4 Upotreba naredbe where u CMD-u

Za izvršavanje programa nakon što je pretvoren u bajt kod, koristimo ključnu riječ Java i potom ime programa. U primjeru u nastavku, korišten je program s imenom Hello World.



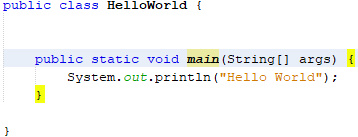
Slika 5 Izvršavanje programa

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



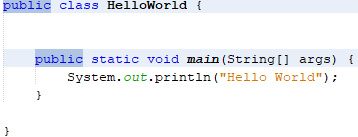
Slika 6 Naredba echo

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



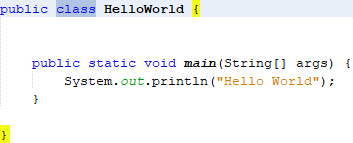
Slika 7 Funkcija main

*Javno*: Ovo je modifikator pristupa u Javi koji se koristi za definiranje nivoa pristupa koji će imati programske kontrole. Javno znači da je klasa javno dostupna.



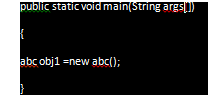
Slika 8 Javni modifikator pristupa

*Class*: Logična je cjelina i za koju se često govori da je nacrt objekta. Sadrži funkcije i ostale članove podataka u jednom entitetu zvanom klasa.



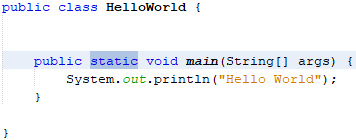
Slika 9 Primjer Klase u Javi

*Object*: Sve u Javi je objekt jer je to jedini fizički entitet od kojeg se sve sastoji. To je kombinacija postupaka i podataka koji rade na već dostupnim podacima.



Slika 10 Objekt u Javi

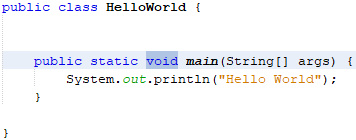
*Static*: Ovo je java ključna riječ koja osigurava da se metoda ili funkcionalni blok može nazvati bez korištenja objekta.



Slika 11 Ključna riječ static

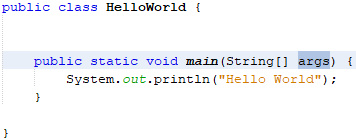
Ostale naredbe uključuju:

*Void*: Ova naredba se koristi za određivanje da neće biti moguće vratiti tip za željenu funkciju.



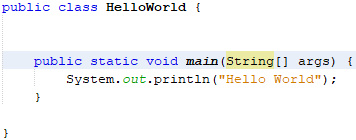
Slika 12 Ključna riječ void

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



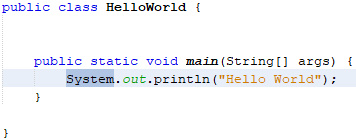
Slika 13 Specificiranje argumenata

*String*: Ovo je unaprijed izgrađena Java klasa koja se koristi za obradu svih povezanih Java nizova i alfabeta.



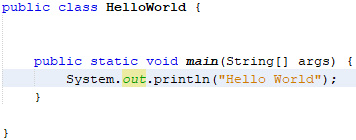
Slika 14 String

*System*: Ovo je unaprijed izgrađena klasa na Javi koja je prisutna u zadanom paketu lang koji se koristi za rukovanje standardnim ulazom, standardnim izlazima i drugim greškama u streamu.



Slika 15 Klasa System u Javi

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



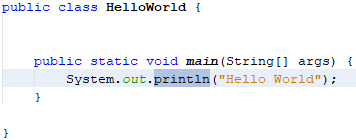
Slika 16 Out, statički član klase System

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



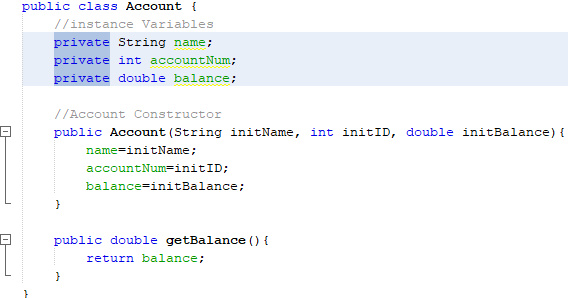
Slika 17 Metoda print

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



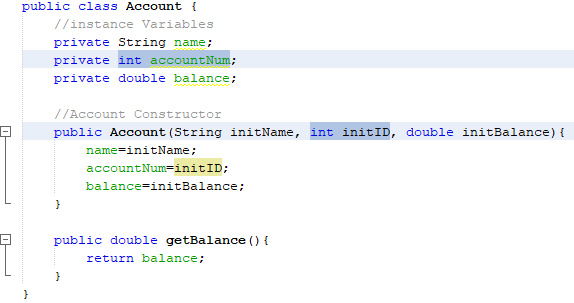
Slika 18 Metoda println

*Private*: Ovo je modifikator pristupa u javi koji se koristi za definiranje opsega metode ili funkcije samo unutar klase.



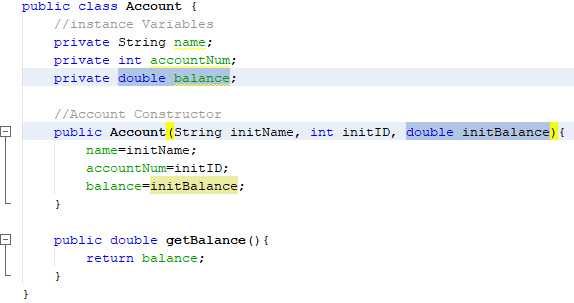
Slika 19 Privatni modifikator pristupa

*Int*: Ovo je cjelobrojni podatkovni tip koji se koristi za obradu svih vrijednosti cjelobrojnih tipova i stoga se varijabla mora inicijalizirati s int tipom podataka.



Slika 20 Cjelobrojni tip podatka

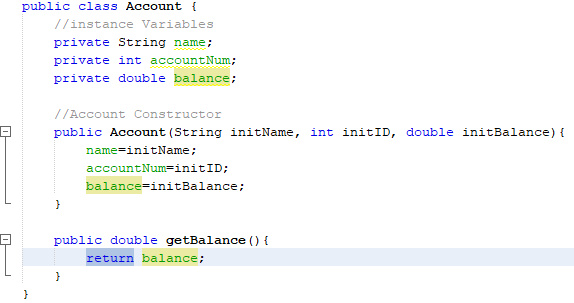
*Double*: Još jedan tip podataka koji se koristi za definiranje varijabli s decimalnim vrijednostima. Dolazi pod kategoriju Java primitivnih tipova podataka koji se koriste za pohranjivanje vrijednosti dvostruke preciznosti. To ima dugi raspon u usporedbi s float varijablom koja se također koristi za definiranje i deklariranje varijabli s decimalnom slikom.



Slika 21 Tip podatka Double

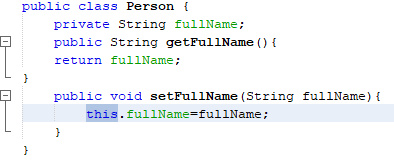
Napredne naredbe su objašnjene u nastavku:

*Return*: Ovo je naredba koja se koristi na kraju programa i posebno vraća samo jednu vrijednost. Također je rezervirana ključna riječ, što znači da se ne može koristiti kao identifikator u Javi. Povratna ključna riječ može se koristiti za izlazak iz metode bilo sa vrijednošću ili bez vrijednosti.



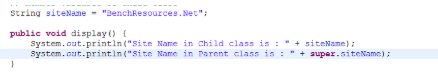
Slika 22 Naredba return

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



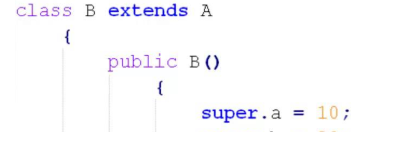
Slika 23 Ključna riječ this

*Super* ključna riječ: To je varijabla koja se koristi za referencu predmeta roditeljske klase ili za pozivanje metoda roditeljske klase.



Slika 24 Super

*Extends*: Ovo je ključna riječ koja se koristi u slučaju nasljeđivanja, a to je Java objektno orijentirani koncept programiranja gdje druge komponente atribute jedne komponente (klase ili sučelja) nazivaju.



Slika 25 Extends

Ovo su neke od Java naredbi i 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:

1. naredbe uvjetovanja (if i switch)

2. naredbe ciklusa (for, while, do-while)

3. naredbe skoka (breal, continue, return).

Java podržava dvije naredbe uvjetovanja i to if i switch. Naredbom if se program može usmjeriti duž jednog od dva različita puta. Opći oblik naredbe if:

* If (uvjet) naredba 1;
* Else naredba 2;[[16]](#footnote-16)

Ako postoji više uvjeta može se koristiti if-else-if naredba .Međutim kako bi se izbjeglo 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.[[17]](#footnote-17) Njen opći oblik je:

While (uvjet) {

// tijelo petlje

}

Na primjer:

int brojac=0;

while(brojac<10) {

f=x-brojac;

Brojac ++;

}

Petlja do-while uvijek izvršava tijelo petlje barem jednom, jer se uvjetni izraz nalazi na njenom kraju.[[18]](#footnote-18) Njen opći oblik je:

Do {

// tijelo

} while (uvjet);

Na primjer:

int k=0;

do {

System.out.println(”k= ”+k);

k++;

} while (k<=10);

Petlja for radi na sljedeći način: kada petlja započne rad, izvršava se inicijalizacija koja djeluje kao brojač ciklusa petlje. Zatim se ispituje uvjet koji uspoređuje upravljačku varijablu petlje sa zadatom ciljnom vrijednošću. Ako je izraz točan, izvršava se tijelo petlje. Ako je netočan, petlja se završava. Zatim se izvršava iteracija.[[19]](#footnote-19) 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 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.[[20]](#footnote-20)

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 nasredba je return, koja se koristi za bezuvjetan izlazak iz metode.[[21]](#footnote-21)

## 2.4. Java platforma

Platforma je hardversko ili softversko okruženje u kojem program radi. Dakle, postoje dva tipa platformi: softverski zasnovane i hardverski zasnovane. Java osigurava softverski zasnovanu platformu. Java platforma se razlikuje od većine drugih platformi u smislu da je to softverski zasnovana platforma koja radi na drugim hardverski zasnovanim platformama.

Java platforma naziv je računalne platforme Oracle koja pomaže korisnicima pokretati te razvijati Java aplikacije. Platforma ne omogućuje korisniku samo pokretanje i razvoj Java aplikacije, već nudi i širok izbor alata koji programerima mogu pomoći u efikasnom radu s Java programskim jezikom.

Platforma se sastoji od dva bitna dijela softvera:

* Java Runtime Environment (JRE), potreban za pokretanje Java aplikacija i apleta; i
* Java Development Kit (JDK), koji je potreban za razvoj tih Java aplikacija i apleta. [[22]](#footnote-22)

Bilo koji dio koda napisan u programskom jeziku Java može se pokrenuti na bilo kojem operacijskom sustavu, platformi ili arhitekturi - u stvari, može se pokrenuti na bilo kojem uređaju koji podržava Java platformu. Prije Jave, ovu količinu sveprisutnosti bilo je jako teško postići. Ako je softver napisan za sustav zasnovan na Unixu, bilo je nemoguće pokrenuti istu aplikaciju na Windows sustavu - u ovom slučaju aplikacija je bila izvorna samo za sustave koji se bave Unixom.

Glavna prekretnica u razvoju Java programskog jezika bila je razvijanje posebnog okruženja za izvršavanje koje bi izvršilo bilo koju Java aplikaciju neovisno od operacijskog sustava, platforme ili arhitekture računara.

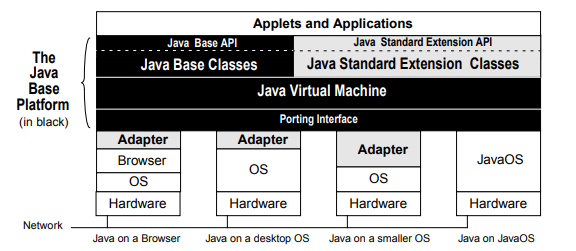
Java Runtime Environment (JRE) nalazi se na vrhu operacijskog sustava, platforme i arhitekture stroja. Ako i kada se pokrene Java aplikacija, JRE djeluje kao veza između osnovne platforme i te aplikacije. On interpretira Java aplikaciju da se pokreće u skladu s podlogom platforme, tako da nakon pokretanja aplikacije izgleda i ponaša se kao izvorna aplikacija. Dio JRE koji postiže ovaj složen sporazum o vezi zove se Java virtualni stroj (JVM).[[23]](#footnote-23)

JRE se brine za pokretanje Java koda na više platformi, međutim programere zanima pisanje čistog koda u Javi koji se zatim može pretvoriti u Java bajt kôd za masovno postavljanje. Programeri ne trebaju pisati Java bajt kod; radije pišu kôd na programskom jeziku Java (što je prilično slično pisanju C ili C ++ kôda).

Nakon preuzimanja JDK, programer osigurava da njihov sustav posjeduje odgovarajući JRE i dodatne alate za pomoć u razvoju aplikacija na Java programskom jeziku.

Java platforma ima dva glavna dijela, Java virtualni stroj i Java API, kao što je prikazano na slici 26.

* Java Virtual Machine -može se implementirati u softver ili hardver. To je apstraktni stroj dizajniran za uvođenje preko postojećih procesora.
* Java API -formira standardno sučelje za aplete i aplikacije, bez obzira na osnovni operacijski sustav. Java API je suštinski okvir za razvoj aplikacija.
* Java Base API pruža osnovni jezik, korisnost, I / O, mrežne, GUI i aplet usluge; OS kompanije koje su licencirale Java ugovorili su da će ih uključiti u bilo koju Java platformu koju instaliraju.
* Java Standard Extension API proširuje mogućnosti Jave uz API Java Base. Neka od tih proširenja će se s vremenom premjestiti na API Java Base. Ostale API-je za nestandardno proširenje mogu pružiti applet, aplikaciju ili temeljni operacijski sustav. [[24]](#footnote-24)



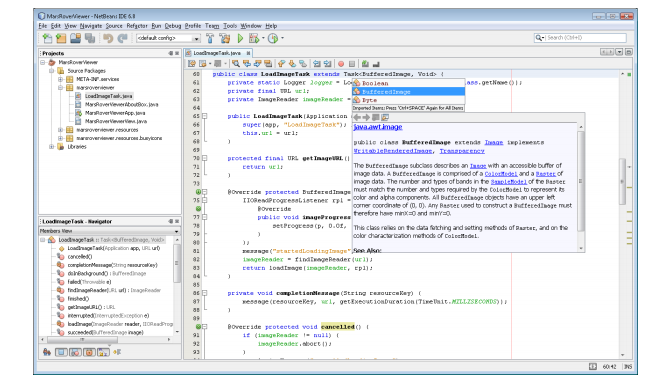
Slika 26 Java Platforma

Izvor: Kramer, 1996

Na ovoj slici, *Java Base* platforma je dio prikazan crnom bojom, uključujući blokove s oznakom Adapter. Java API uključuje i API Java Base i API standardnog proširenja Java. Klase su implementacija tog API-ja. *Java Virtual Machine* je u srži platforme. Sučelje za prijenos je između Java Virtual Machine i operacijskog sustava ili pretraživača. Ovo prijenosno sučelje ima dio koji je neovisan o platformi (prikazan crnom bojom) i dio koji ovisi o platformi, prikazan kao adapteri. OS i JavaOS pružaju funkciju prozora, arhiviranja i mreže. Mrežom se mogu povezati različiti strojevi, kao što je prikazano. Okvir Java API je otvoren i proširiv. Specifikacije za svako sučelje razvijaju stručnjaci širom industrije za svako područje. Predstojeće specifikacije bit će objavljene i otvorene za pregled industrije. Implementacija API specifikacija bit će dostupna od programske kompanije JavaSoft, ali i drugih u cijeloj industriji. U današnjem okruženju brzih inovacija, Java API okvir omogućuje toj inovaciji da lako postoji kao proširenja na Java platformu. API se organizira po grupama ili skupovima. Svaki od API seta može se implementirati kao jedan ili više paketa (imenskih prostora). Svaki paket grupira skup klasa i sučelja koji definiraju skup povezanih polja, konstruktora i metoda.

## 2.5. Razvojno okruženje Netbeans

NetBeans je integrirano razvojno okruženje (IDE) za Javu. NetBeans omogućava razvoj aplikacija iz skupa modularnih softverskih komponenti koje se nazivaju moduli. NetBeans radi na Windowsima, MacOS-u, Linuxu i Solarisu. Pored Java razvoja, on ima proširenja za druge jezike poput PHP, C, C ++, HTML5 i JavaScript.[[25]](#footnote-25)



Slika 27 Prikaz NetBeans IDE sučelja

Izvor: <https://zir.nsk.hr/islandora/object/etfos%3A2203/datastream/PDF/view>

Aplikacije utemeljene na NetBeans-u, uključujući NetBeans IDE, programeri mogu proširiti u skladu sa svojim potrebama. IDE je više od samo uređivača teksta: to je skup integriranih alata koji pomažu u tipičnom razvojnom toku rada kodiranja, testiranja i uklanjanja pogrešaka, profiliranja, kompiliranja, pokretanja i raspoređivanja aplikacija. Pomoću NetBeans IDE-a lako je učinkovito pisati svoj kôd, upravljati i graditi velike softverske projekte i održavati kontrolu nad višestrukim revizijama datoteka u timskom okruženju.

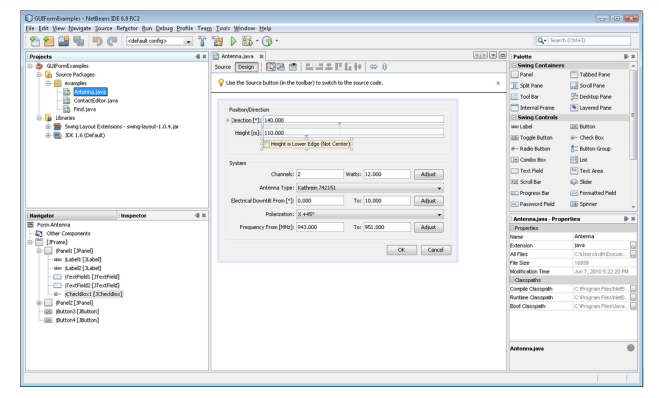
NetBeans je započeo 1996. godine kao Xelfi (igra riječi na Delphiju), projekt Java IDE pod vodstvom Fakulteta za matematiku i fiziku na Sveučilištu Charles u Pragu. Roman Staněk je 1997. osnovao tvrtku oko projekta i proizveo komercijalne verzije NetBeans IDE-a dok ga Sun Microsystems nije kupio 1999. Sun je pokrenuo NetBeans IDE u lipnju sljedeće godine. Od tada, NetBeans zajednica nastavlja rasti. Kupivši tvrtku Sun Microsystem, Oracle Corporation je 2010. godine kupio i NetBeans.[[26]](#footnote-26)

U sklopu Oracle-a, NetBeans se natjecao s JDeveloperom, besplatnim IDE-om koji je povijesno bio proizvod tvrtke. Oracle je u rujnu 2016. podnio prijedlog za donaciju projekta NetBeans softverskoj fondaciji Apache, navodeći da se "otvara model upravljanja NetBeans upravljačima kako bi se sastavnicama NetBeansa dalo veći glas u smjeru projekta i budući uspjeh kroz tada nadolazeće izdanje Java 9". Potez je podržao Javin tvorac James Gosling.

NetBeans platforma je okvir za pojednostavljenje razvoja Swing desktop aplikacija. NetBeans IDE paket za Java SE sadrži ono što je potrebno za početak razvoja dodataka NetBeans i aplikacija utemeljenih na NetBeans platformi gdje nije potreban dodatni SDK. Aplikacije mogu dinamički instalirati module. Svaka aplikacija može uključivati modul Update Center (centar za ažuriranje) koji omogućava korisnicima aplikacije da preuzimaju digitalno potpisanu nadogradnju i nove značajke izravno u pokrenutu aplikaciju. Ponovna instalacija nadogradnje ili novog izdanja ne prisiljava korisnike da ponovno preuzmu cijelu aplikaciju.

Među glavnim značajkama platforme su :

* Upravljanje korisničkim sučeljem (izbornici i alatne trake) ,
* Upravljanje korisničkim postavkama,
* Upravljanje skladištenjem ,
* Upravljanje prozorima ,
* Framework čarobnjak ,
* NetBeans vizualna knjižnica ,
* Integrirani razvojni alati (GUI alat prikazan na slici 28).[[27]](#footnote-27)



Slika 28 Prikaz NetBeans GUI alata za dizajn aplikacije

Izvor: <https://zir.nsk.hr/islandora/object/etfos%3A2203/datastream/PDF/view>

NetBeansIDE je službeni IDE za Java programski jezik. Pomoću njegovih uređivača, analizatora koda i pretvarača mogu se brzo i glatko nadograditi vlastite aplikacije za korištenje novih konstrukcija na jeziku Java, poput lambda, funkcionalnih operacija i referenci metoda.

Dostupni su skupni analizatori i pretvarači za pretraživanje kroz više aplikacija istovremeno, podudaranje obrazaca za pretvorbu u nove konstrukcije jezika Java.

Zahvaljujući svom stalnom unapređivanju Java Editor-a, mnogim bogatim značajkama i širokom rasponom alata, predložaka i uzoraka, NetBeans IDE postavlja standard za razvoj sa vrhunskim tehnologijama.

IDE je mnogo više od tekst editora. NetBeans Editor uvlači retke, podudara se s riječima i zagradama te naglašava izvorni kod sintaktički i semantički. Omogućuje jednostavno vraćanje koda refaktora s nizom zgodnih i moćnih alata, a također nudi i predloške koda, savjete za kodiranje i generatore koda.

Uređivač (engl. editor) podržava mnoge jezike od Java, C / C ++, XML i HTML, do PHP, Groovy, Javadoc, JavaScript i JSP. Budući da je uređivač moguće proširivati, moguće je uključiti podršku za mnoge druge jezike.

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.

## 2.6. Ostala razvojna okruženja

Java je jedan od vodećih programskih jezika. Široko se koristi za razvoj snažnih i sigurnih aplikacija za radne površine, poslovne aplikacije, mobilne aplikacije i web aplikacije.

Na raspolaganju je mnoštvo razvojnih okruženja za rad s Javom. Iako su Eclipse, IntelliJ IDEA i NetBeans velika trojka, postoje brojni drugi IDE-ovi za razvoj aplikacija koje učinkovito koriste Javu.

Pored NetBeansau ovom radu će biti navedeni i opisani još neki od njih.

**IntelliJIDEA** jedan je 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.

IntelliJ Idea dostupna je u dva izdanja:

* Community Edition: Dostupno je besplatno i uglavnom se koristi programerima Java i Android. Podržava većinu jezika kao što su Java, Kotlin, Groovy, Clojure, Scala i drugi.
* Ultimate Edition: Ultimate edition ima najnapredniji skup funkcija za razvoj web i desktop aplikacija. Podržava integraciju proljetnog okvira, okvira za web razvoj poput Node.js, Angular i React, Java EE podršku poput JSF, JAX-RS, JPA, CDI, itd.[[28]](#footnote-28)

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

**Eclipse** je jedan od tri velika Java IDE-a. Moderno integrirano razvojno okruženje dostupno je u desktop i cloud izdanjima. Oblak izdanje Eclipse, nazvan Eclipse Che, omogućava programerima razvijanje aplikacija putem web-preglednika. Eclipse se pojavio 2001. prve godine i može se korisiti na Linux, macOs, Windows i Solaris platofrmi. Oba izdanja Eclipse IDE opremljena su potrebnom / dodatnom funkcionalnošću pomoću dodataka. Eclipse ima niz dodataka na raspolaganju. Da bi se olakšala inkrementalna kompilacija Java koda, Eclipse dolazi s prilagođenim sastavljačem.[[29]](#footnote-29)

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

Eclipse podržava razvoj aplikacija u nekoliko programskih jezika putem dodataka. C, C ++, Clojure, Groovy, Haskell, JavaScript, Julia, Perl, PHP, Ruby, Rust i Scala neki su od različitih programskih jezika koje podržava Eclipse.

**BlueJ** je integrirano razvojno okruženje za Javu. Iako je uglavnom dizajniran s obrazovnim namjerama, BlueJ je prikladan za razvoj softvera malih razmjera. Unatoč tome što su razvijeni kao gotovi IDE za početnike, Java veterani uvelike preferiraju ovaj Java IDE. Prvi put se pojavio još 1999. godine.

Njegove glavne karakteristike su:

* Simple BlueJ ima namjerno manje i jednostavnije sučelje od profesionalnih okruženja poput NetBeans ili Eclipse.
* Dizajniran za podučavanje - Postoji popularni udžbenik dizajniran za podučavanje uvodnih sveučilišnih / fakultetskih tečajeva sa BlueJ-om i web mjesto prepuno nastavnih sredstava.
* Interaktivni BlueJ omogućava interakciju s objektima. Možete pregledati njihovu vrijednost, pozivati ​​metode na njih, proslijediti ih kao parametre i još mnogo toga. Java izraze možete i direktno pozivati ​​bez kompiliranja. Tako je BlueJ moćna grafička ljuska / REPL za Javu.
* Prijenosni BlueJ radi na Windows, Mac OS X, Linux i drugim platformama na kojima radi Java. Može se izvoditi i bez instalacije s USB sticka.
* Zreli BlueJ je stariji od petnaest godina, ali nastavlja da ga ažurira i podržava tim koji je stalno zaposlen. Želi odgovoriti na sve zahtjeve za tehničku podršku u roku od jednog radnog dana.
* Inovativni BlueJ ima nekoliko funkcija koja ranije nisu viđena u drugim IDE. Objektna klupa, šifra jastuka i obojenje opsega bile su sve originalne značajke BlueJ-a.[[30]](#footnote-30)

Primarni ekran BlueJ-a prikazuje klasnu strukturu aplikacije koja se razvija. Omogućuje i interaktivno stvaranje i testiranje objekata. Multi-platforma Java IDE omogućava programerima laku komunikaciju s objektima. Programeri mogu pozvati metode na objekte, pregledati vrijednosti objekata i proslijediti objekte kao parametre pomoću BlueJ Java IDE. Ostale moćne značajke razvoja Java programa koje nudi BlueJ uključuju stvaranje dinamičkih objekata i vizualno skeniranje koda.

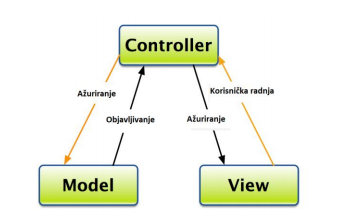
**JDeveloper** je besplatni program Java IDE tvrtke Oracle. Nudi integrirani razvojni okvir s mnoštvom značajki i nekoliko alata za vizualni razvoj. JDeveloper pokriva cjelokupni životni ciklus razvoja, kodiranje, dizajniranje, uklanjanje pogrešaka, optimizaciju, profiliranje i implementaciju.

Oracle JDeveloper može se integrirati s okvirom za razvoj aplikacija Oracle (Oracle ADF) kako bi se dodatno pojednostavio razvoj aplikacija. Pored Jave, JDeveloper se može koristiti i za razvoj aplikacija u GTML-u, JavaScript-u, PHP-u, SQL-u i XML-u.[[31]](#footnote-31)

## 2.7. MVC Arhitektura

Model-prikaz-upravitelj (obično poznat kao MVC) je softverski obrazac dizajna (engl. design pattern) koji se obično koristi za razvoj korisničkih sučelja koji dijeli srodnu programsku logiku na tri međusobno povezana elementa. To se radi da 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 korišten za grafička korisnička sučelja za stolna računala (GUI), ovaj je obrazac postao popularan 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 znači Model, View i Controller*.* MVC razdvaja aplikaciju na tri komponente - Model, View i Controller.

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 5).[[32]](#footnote-32)



Slika 29 MVC okvir

Izvor:<http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf>

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

Komponenta View koristi se za svu UI 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 View 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 (Microsoft, 2013):

* Olakšano upravljanje razvojem složenih aplikacija zbog mogućnosti posebnog razvoja svake od komponenti aplikacije, čime se olakšava organizacija posla razvoja aplikacije, a također i nadogradnja novim verzijama (ponekad je češće potrebno unaprijediti poglede i upravitelje dok Model ostaje isti)
* Olakšano testiranje aplikacije (tzv. Test-Driven Development - TDD), jer je moguće odvojeno testirati pojedine komponente aplikacije i nije uvijek za testiranje potrebno koristiti web server (neke komponente se mogu testirati lokalno)
* Olakšano procesiranje zahtjeva koje šalje korisnik, jer se za tu svrhu koristi *Front Controller* koji kroz jedan upravitelj šalje sve zahtjeve korisnika.[[33]](#footnote-33)

# JAVA APLIKACIJA ZA EVIDENCIJU RADNOG VREMENA

## 3.1. Opis problema

*Cilj ovog rada bio je implementirati desktop aplikaciju za evidenciju radnog vremena(engl. „from-scratch“), koristeći Java programski jezik za implementaciju poslovne logike (engl. „business logic“), te JavaFX Scene Builder 2.0 za kreiranje dizajna aplikacije. Za bazu podataka korištena je MySQL baza podataka iz razloga što je najkompaktnija – najlakša za preuzimanje, instalaciju i korištenje. JavaFX korištena je iz razloga što pruža znatno veće mogućnosti od Swing frameworka. Jedna od prednosti je korištenje CSS-a za oblikovanje izgleda aplikacije. Sustav korišten za implementaciju aplikacije je Windows 10, ali aplikaciju je moguće pokrenuti i na bilo kojem drugom uređaju koji ima instaliranu Java podršku.*

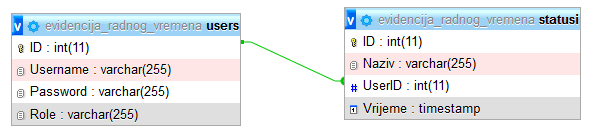
Sustav razlikuje 2 vrste korisnika:

1. Korisnik
2. Admin

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

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

## 3.2. Relacijski model baze podataka

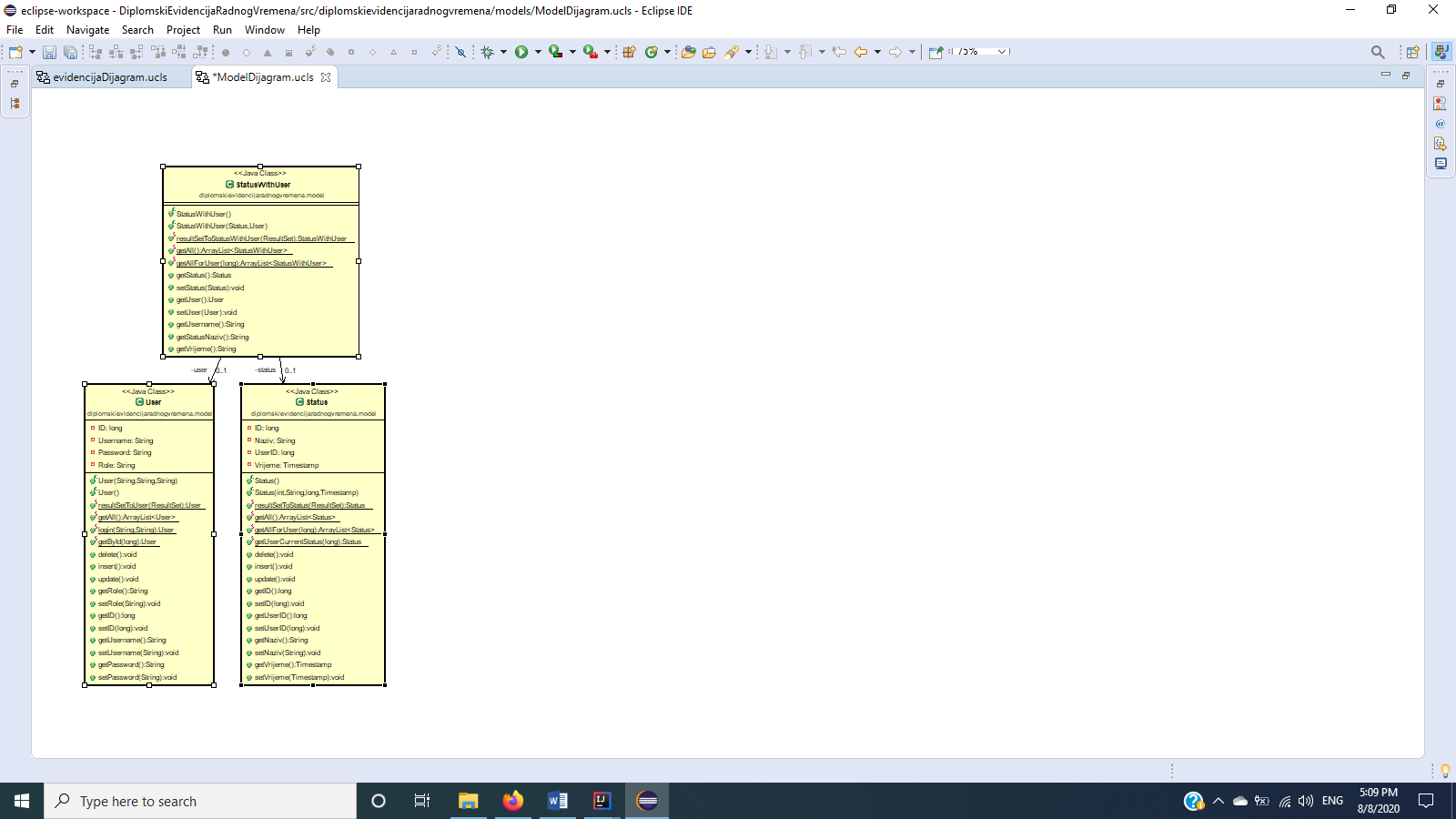


Slika 30 Relacijski model baze podataka

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

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

## 3.3 Dijagram klasa

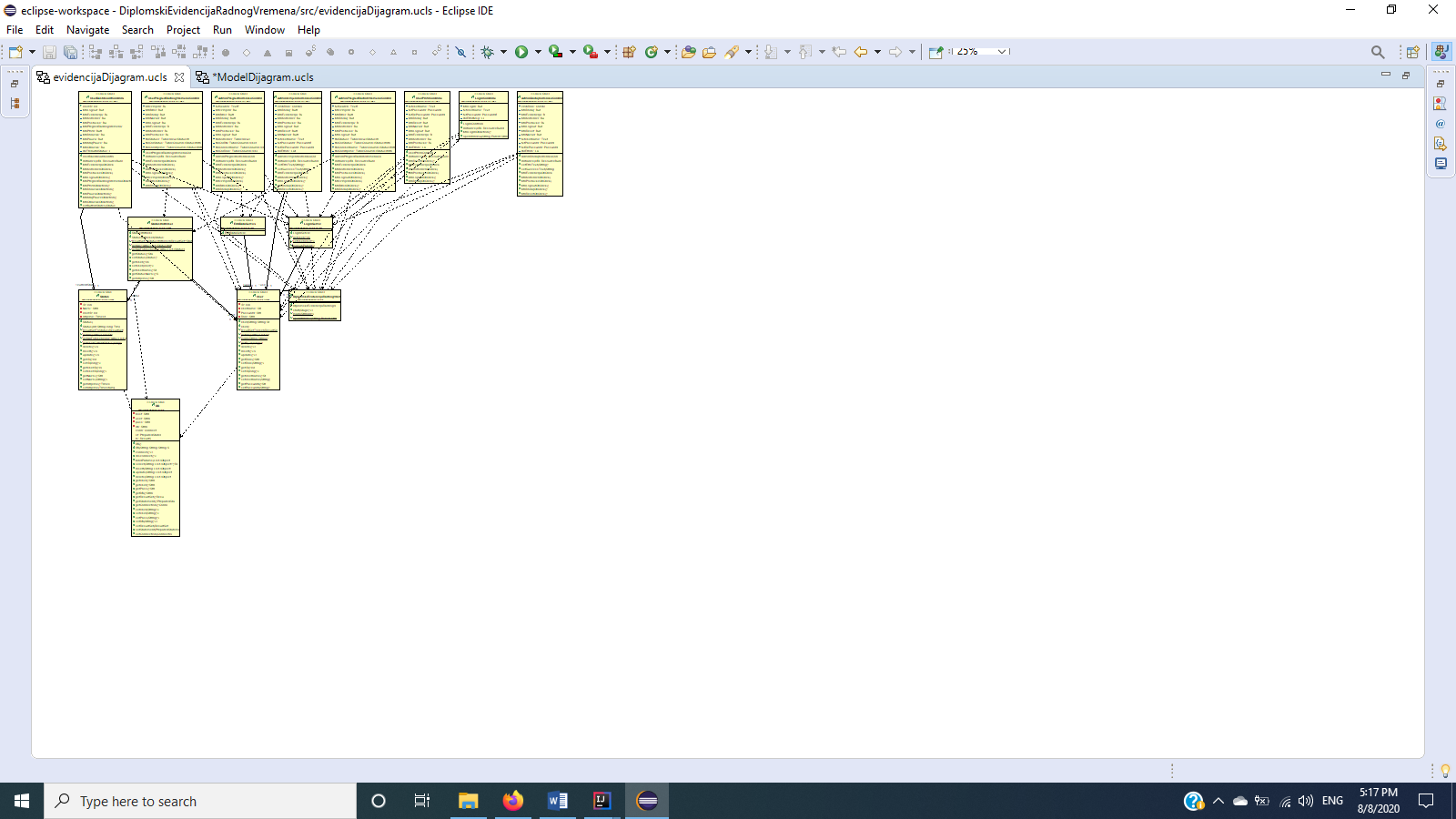


Slika 31 Dijagram klasa za paket model

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

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

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



Slika 32 Dijagram klasa kompletne aplikacije

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

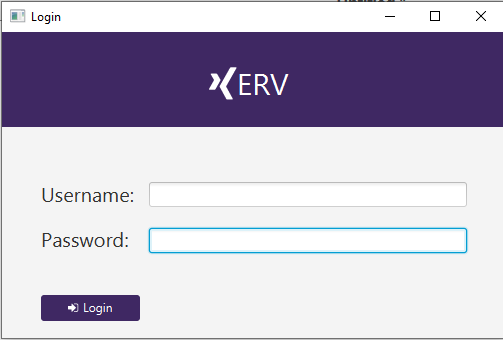
Prvi red na prethodnoj slici su upravitelji, odnosno klase iz paketa *controller*, dole lijevo su klase paketa *models,* a dole desno paket *services.*

## Prozori aplikacije

### 3.4.1.Login

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

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



Slika 33 Prikaz Login prozora aplikacije

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

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

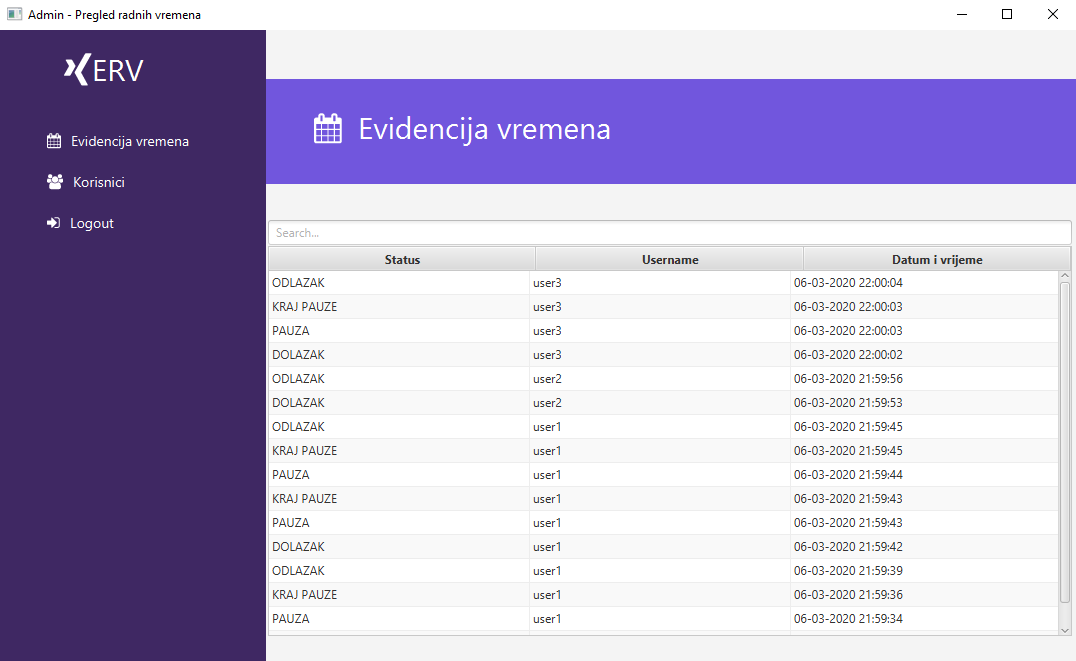
### 3.4.2. Evidencija vremena - Admin

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

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

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

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

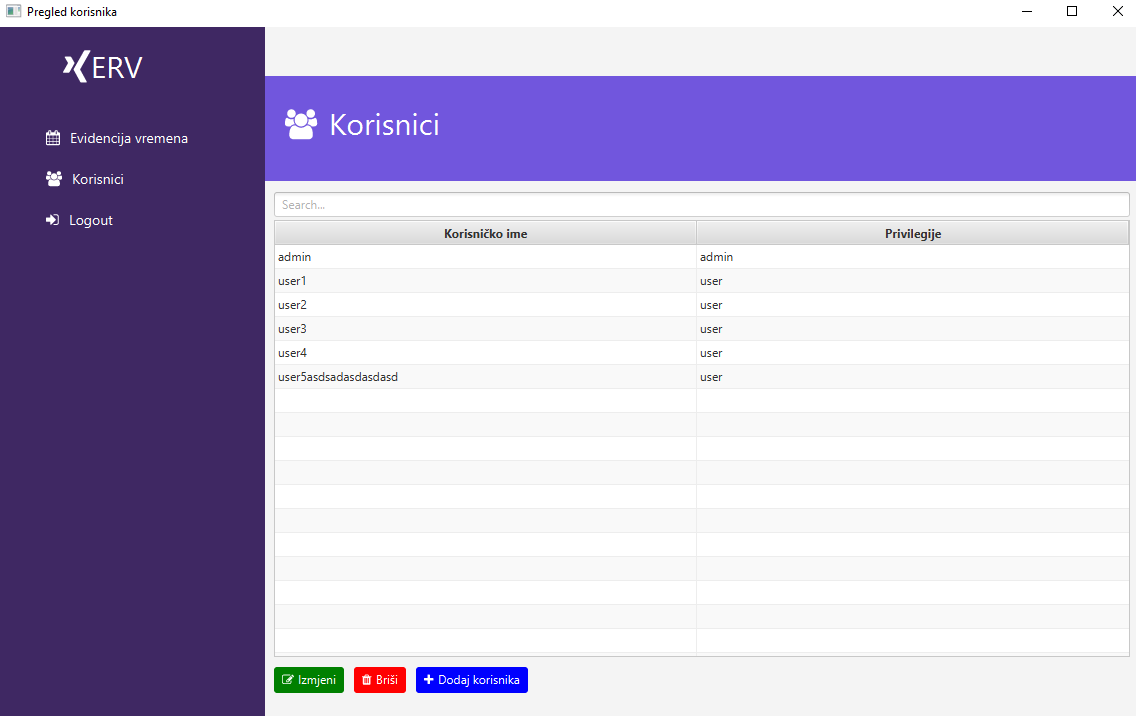


Slika 34 Admin Evidencija vremena

### Korisnici - Admin

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

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



Slika 35 Admin Pregled korisnika

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

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

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

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



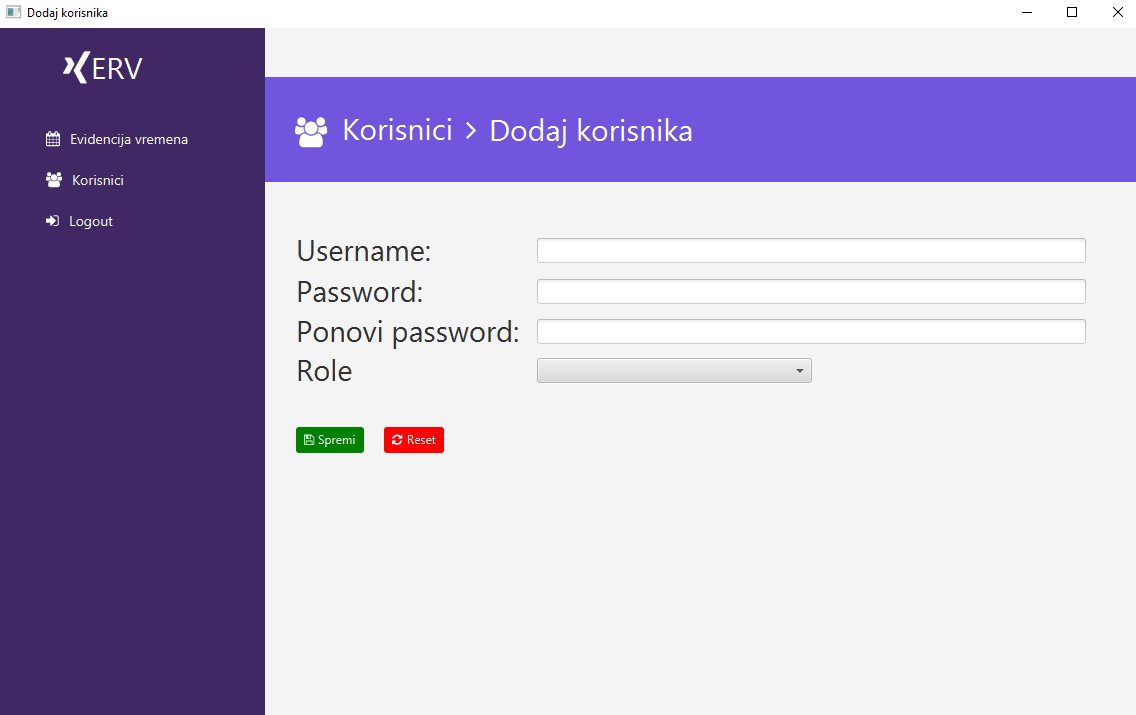
75. **public** **static** User login(String username, String password){
76. User user = **null**;
78. DB db = **new** DB();
79. List<Object> params = Arrays.asList(username, password);
80. db.select("SELECT \* FROM users WHERE Username=? AND Password=?", params);
81. **try** {
82. **while**(db.getResultSet().next()){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

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

### Dodaj korisnika - Admin

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

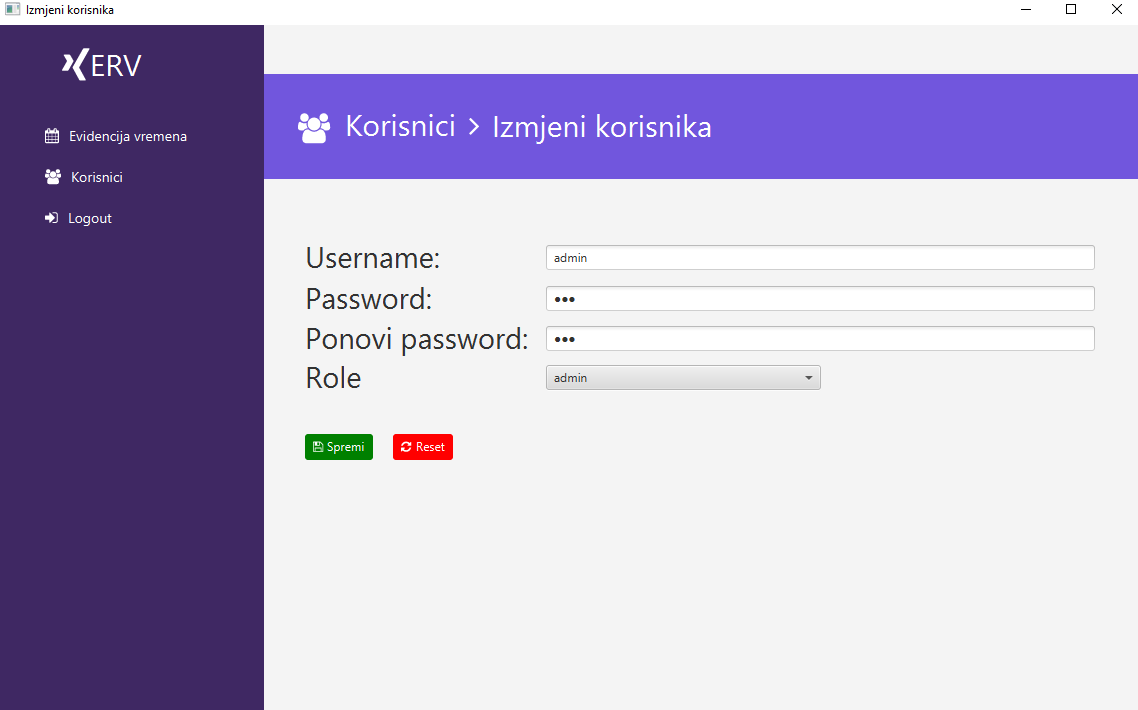
Nakon što su popunjena sva polja potrebno je kliknuti na dugme *Spremi* koje će se izvršiti u *AdminDodajKorisnikaController* odakle će novi korisnik preko klase *User* biti upisan u *MySQL* bazu podataka.



Slika 36 Admin Dodaj Korisnika

### Izmjeni korisnika - Admin

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



Slika 37 Admin Izmjeni Korisnika

### Početna - Korisnik

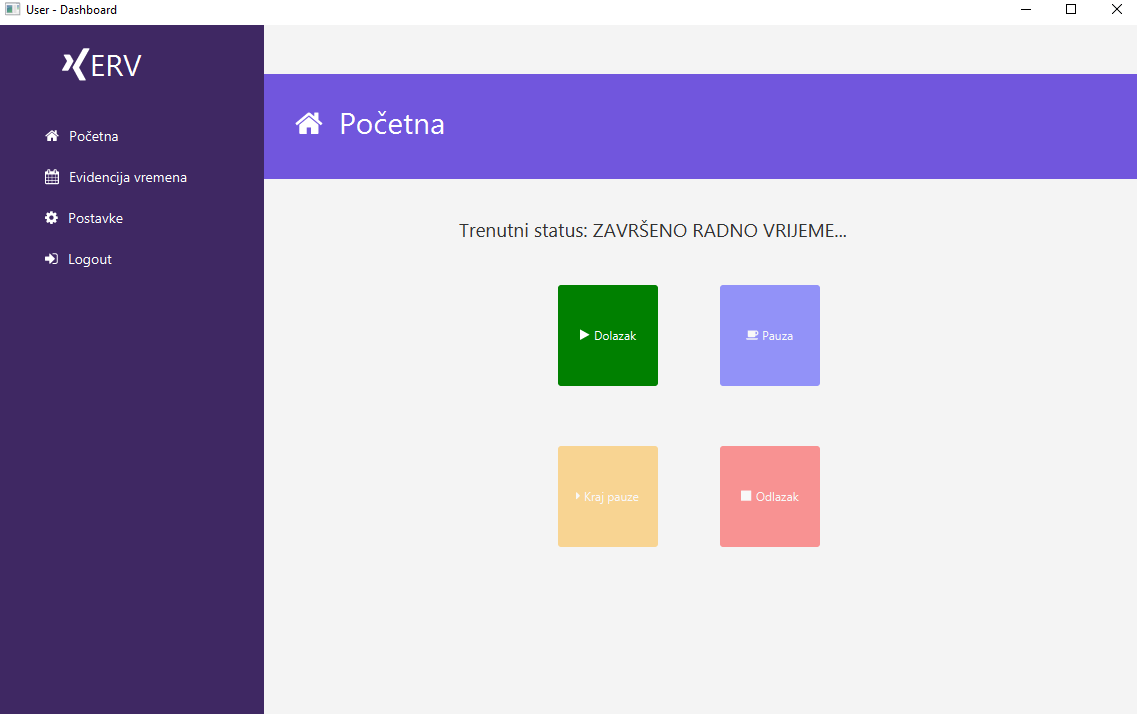
Ukoliko je prijava izvršena s podacima računa koji ima ulogu *Korisnik,* a ne *Admin,* otvara se prozor koji ima nešto drugačije korisničko sulčelje. Prije svega treba primjetiti da imamo drugačije opcije u izborniku s lijeve strane:

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

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

Aplikacija ima četiri dugmeta (buttona):

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



Slika 38 Početna stranica- Korisnik

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

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

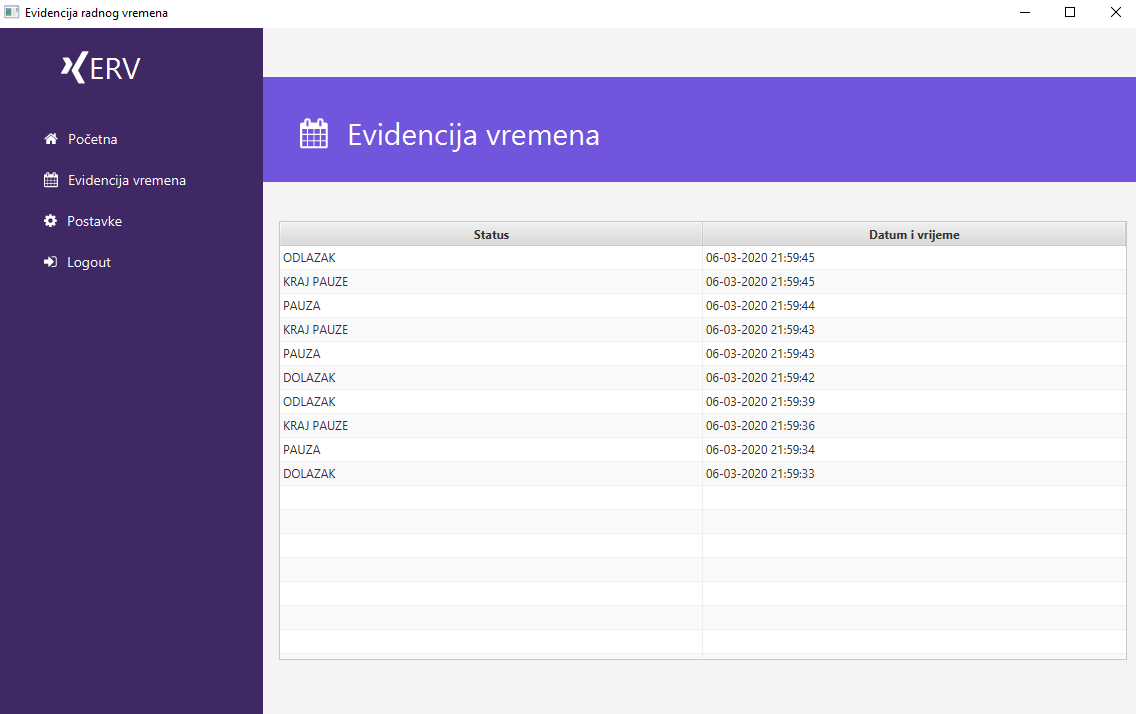


61. **public** **void** btnEvidencijaOnClick(){
62. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Evidencija radnog vremena", btnLogout.getParent(), "UserPregledRadnogVremena");
63. }
64. **public** **void** btnKorisniciOnClick(){
65. diplomskievidencijaradnogvremena.DiplomskiEvidencijaRadnogVremena.openWindow("Početna",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

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

### Evidencija vremena - Korisnik

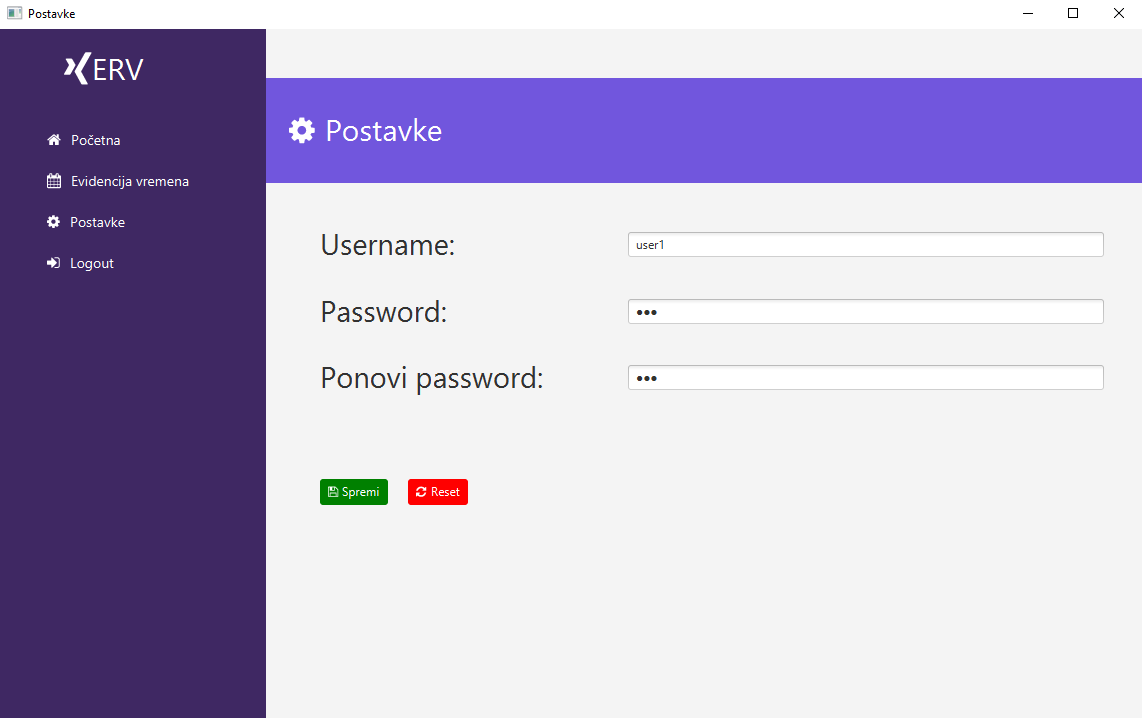
Ukoliko je u izborniku odabrana opcija *Evidencija vremena,* prikazuje se nešto drugačija tablica od one koja se prikazuje administratoru. Naime, ne prikazuje se evidencija vremena za sve korisnike, već samo prikaz vremena onog korisnika koji je trenutno prijavljen pa je stoga i sama tablica nešto drugačija.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



Slika 39 Evidencija vremena korisnika

### Postavke - Korisnik

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

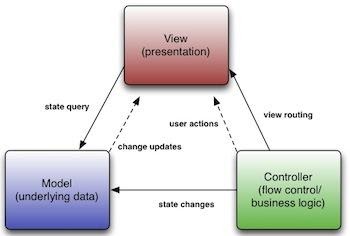


Slika 40 Postavke profila- korisnik

## 3.5. MVC arhitektura u aplikaciji

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

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



Slika 41 Tok aplikacije koristeći MVC arhitektonski obrazac

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

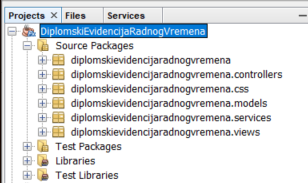
Pogled: Pogled je korisničko sučelje. Pregledajte prikazne podatke pomoću modela korisniku i također im omogućuje izmjenu podataka.

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

Unutar aplikacije, kreirana su 4 java paketa:

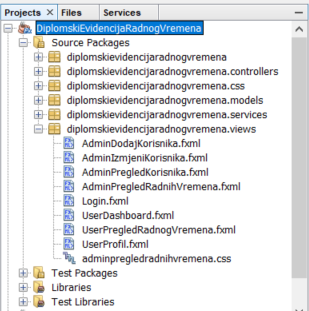
**diplomskievidencijaradnogvremena.controllers, diplomskievidencijaradnogvremena.models, diplomskievidencijaradnogvremena.views i**

**Diplomskievidencijaradnogvremena.css**

****

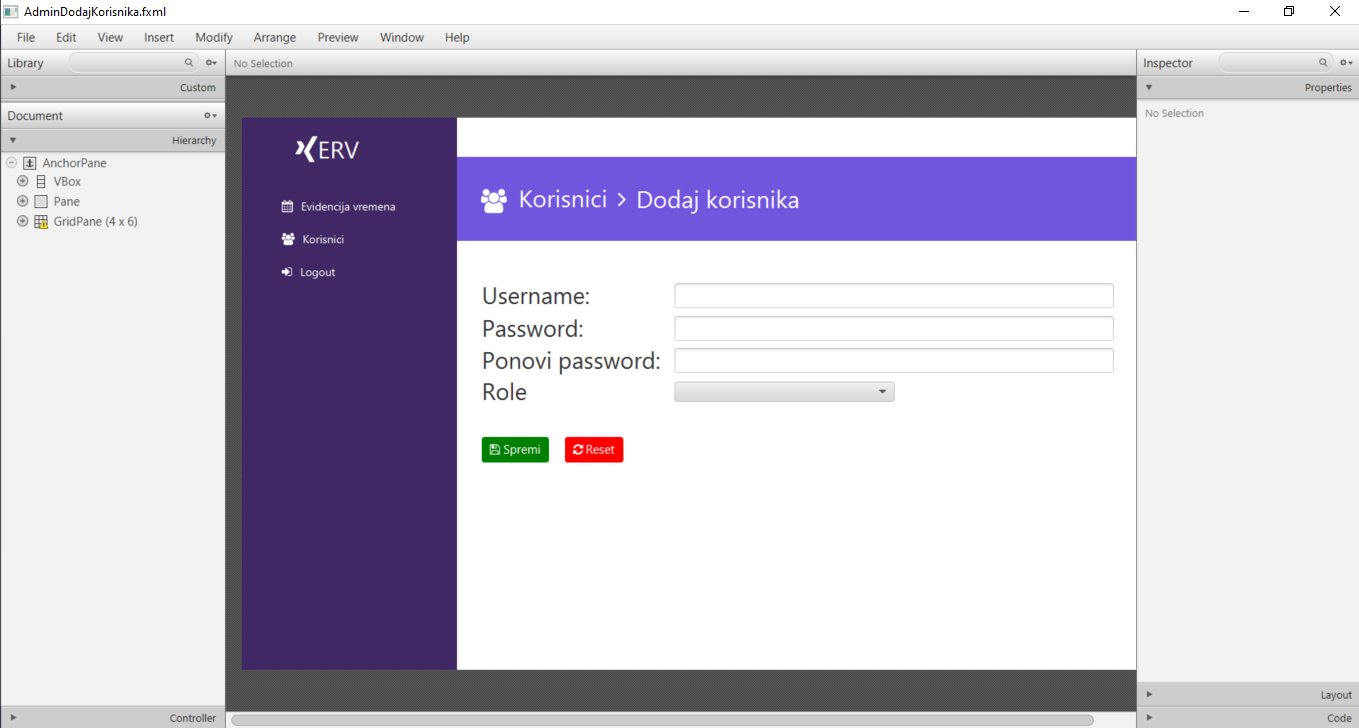
Slika 42 Prikaz svih java paketa aplikacije

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

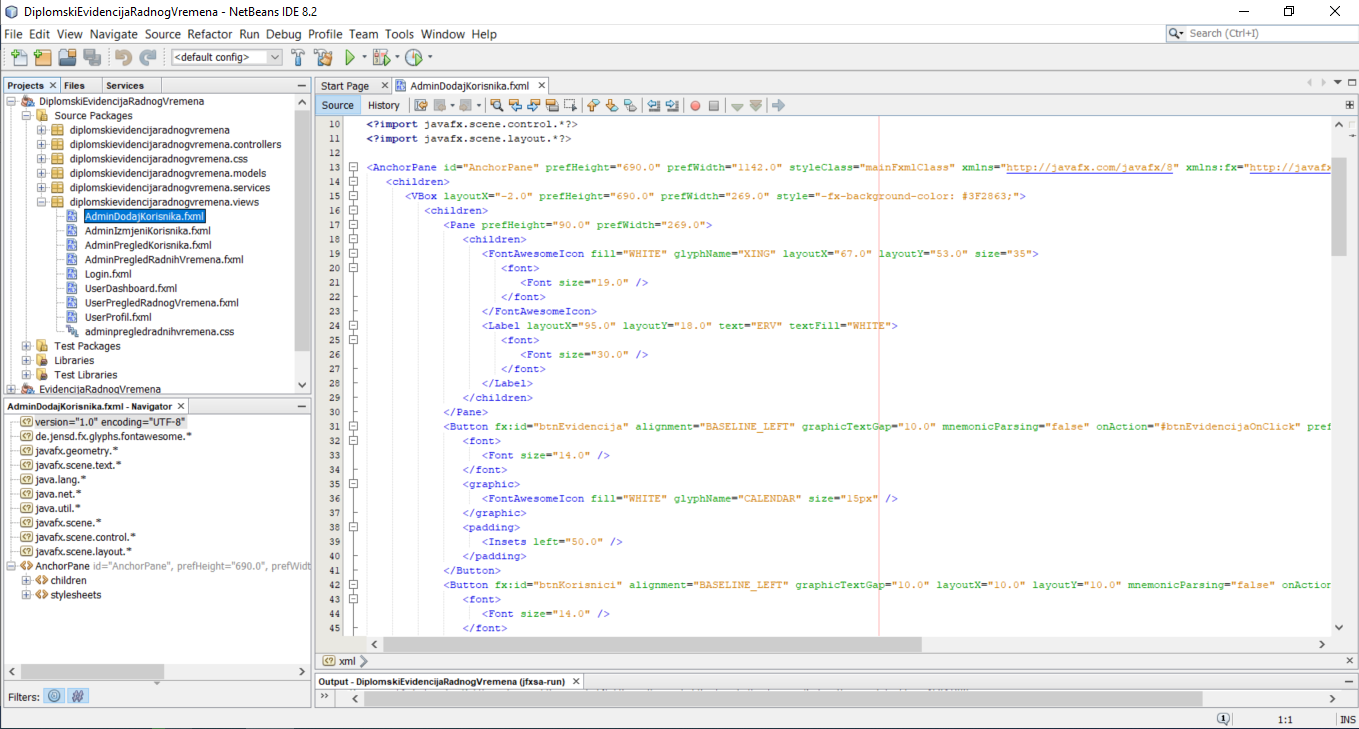


Slika 43 Pregled datoteka unutar views paketa

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

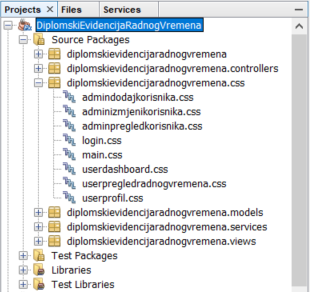


Slika 44 AdminDodajKorisnika.fxml fajl pokrenut unutar JavaFX Scene Builder 2.0



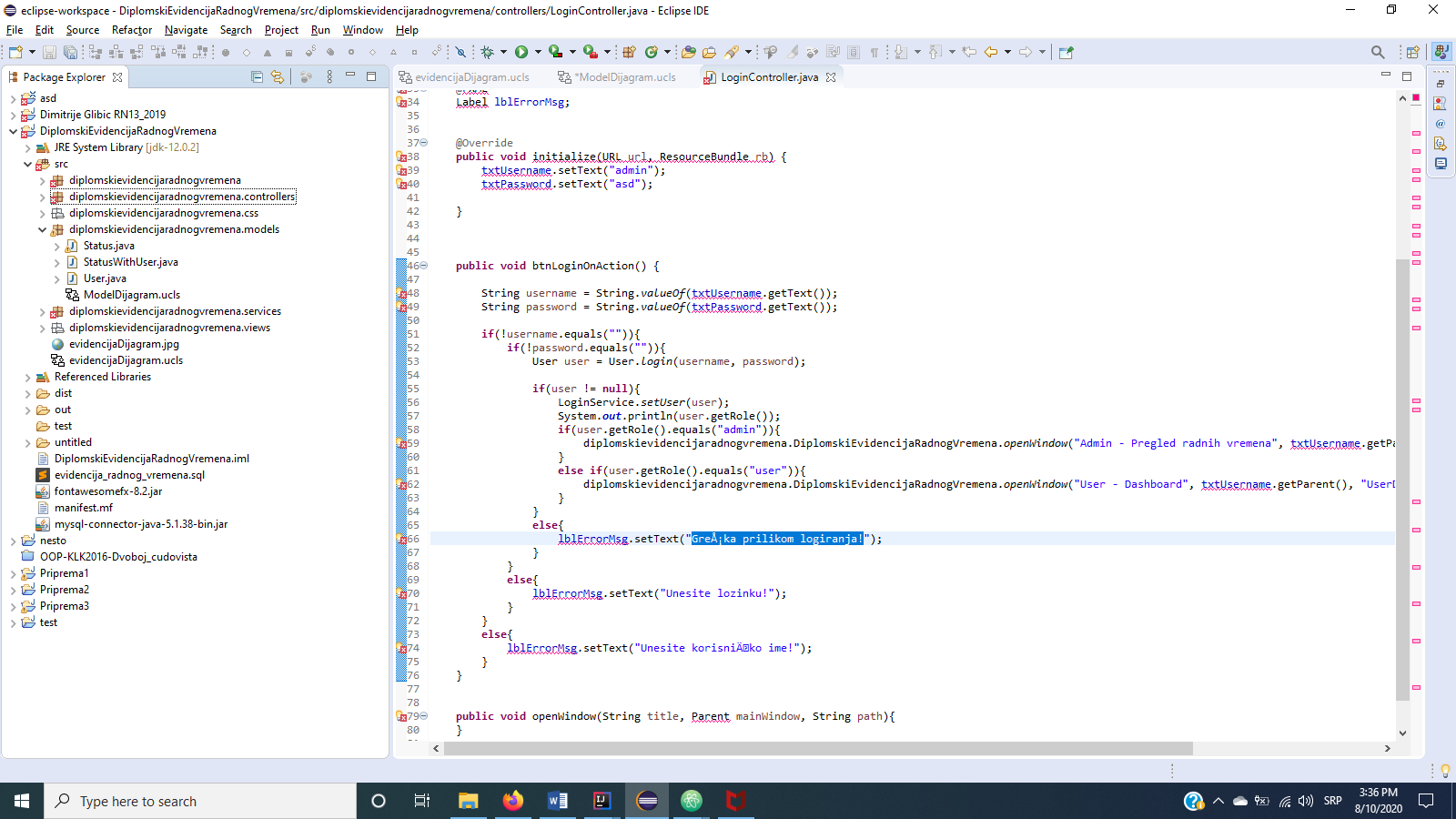
Slika 45 AdminDodajKorisnika.fxml prikaz koda unutar Netbeans-a

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



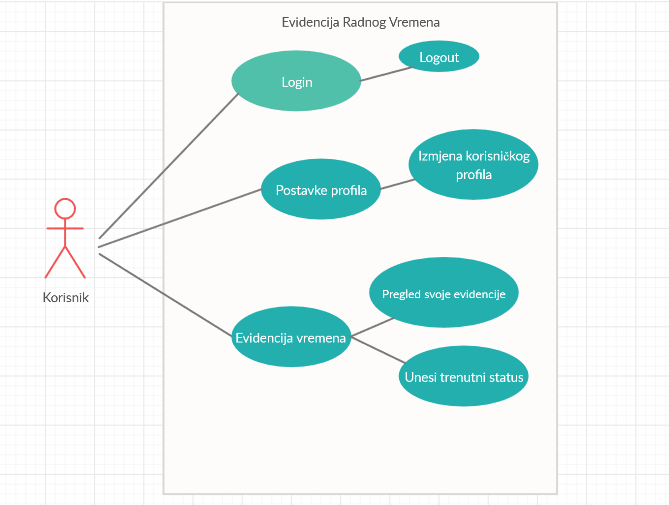
Slika 46 Pregled datoteka unutar css paketa

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

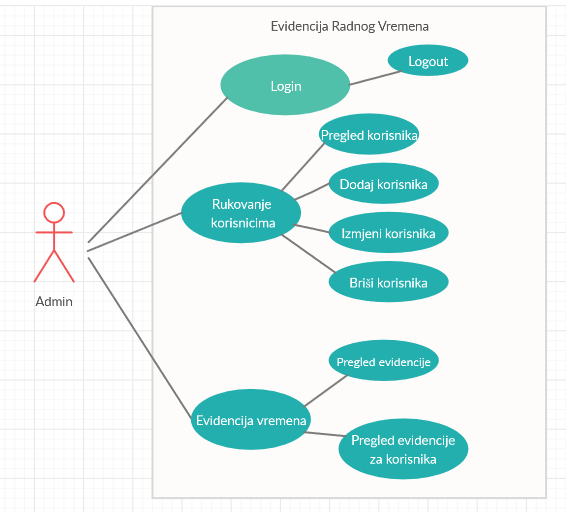


Slika 47 Pregled datoteka unutar models paketa

## 3.6. Slučajevi korištenja



Slika 48 Prikaz slučajeva korištenja za korisnika



Slika 49 Prikaz slučajeva korištenja za administratora

## Mogućnosti proširenja aplikacije

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

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

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

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

# ZAKLJUČAK

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

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

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

# LITERATURA

1. Burd B.(2017): Beginning Programming with Java For Dummies. IDG Books
2. Budimlic, Z., & Kennedy, K. (1997). Optimizing Java: theory and practice. *Concurrency: Practice and Experience*, *9*(6), 445-463.
3. Dibble, P. (2002). *Real-time Java platform programming*. Prentice Hall Professional.
4. Hunter, J. i Crawford W. (1998). Java Servlet Programming. O'Reilly and Associates, Inc., Sebastopol, CA
5. Eckel, B. (1998). Thinking in Java: A Definitive Introduction to Object-Oriented Programming in the Language of the World Wide Web. Prentice Hall, Upper Saddle River, NJ. Worldwide, Inc. Foster City, California.
6. Kramer, D. (1996). The Java Platform. Mountain View, California.

Jaworski, J. (1998). Java 1.2: Unleashed. 4th Edition. Macmillan Computer Publishing, Indianapolis, IN, 1998

1. Službena stranica NetBeansa-a, <https://netbeans.org/>
2. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf>

<https://www.oracle.com/application-development/technologies/jdeveloper.html>

<https://www.bluej.org/about.html>

1. [https://www.javatpoint.com/intellij-vs eclipse#:~:text=The%20more%20plugins%20installed%20in,smoother%20as%20compared%20to%20Eclipse.](https://www.javatpoint.com/intellij-vs%20eclipse#:~:text=The%20more%20plugins%20installed%20in,smoother%20as%20compared%20to%20Eclipse.)

<https://www.javatpoint.com/features-of-java>

1. <https://zir.nsk.hr/islandora/object/etfos%3A2203/datastream/PDF/view>
2. <https://javaprogramiranje.wordpress.com/2009/11/07/upravljacke-naredbe/>

<https://programiranje.ba/blog/koje-programske-jezike-uciti-u-2020-ultimativni-vodic/>

1. <http://laris.fesb.hr/java/osnovni_program.htm>
2. <https://www.geeksforgeeks.org/the-complete-history-of-java-programming-language/>

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. Jaworski, J. (1998): Java 1.2: Unleashed. 4th Edition. Macmillan Computer Publishing, Indianapolis, IN, 1998 [↑](#footnote-ref-5)
6. Ibid [↑](#footnote-ref-6)
7. <https://www.javatpoint.com/features-of-java> [↑](#footnote-ref-7)
8. Ibid [↑](#footnote-ref-8)
9. Hunter, J. i Crawford W. (1998). Java Servlet Programming. O'Reilly and Associates, Inc., Sebastopol, CA. [↑](#footnote-ref-9)
10. Eckel, B. (1998). Thinking in Java: A Definitive Introduction to Object-Oriented Programming in the Language of the World Wide Web. Prentice Hall, Upper Saddle River, NJ. [↑](#footnote-ref-10)
11. Budimlic, Z., & Kennedy, K. (1997). Optimizing Java: theory and practice. *Concurrency: Practice and Experience*, *9*(6), 445-463. [↑](#footnote-ref-11)
12. Ibid [↑](#footnote-ref-12)
13. <https://programiranje.ba/blog/koje-programske-jezike-uciti-u-2020-ultimativni-vodic/> [↑](#footnote-ref-13)
14. [↑](#footnote-ref-14)
15. <http://laris.fesb.hr/java/osnovni_program.htm> [↑](#footnote-ref-15)
16. <https://javaprogramiranje.wordpress.com/2009/11/07/upravljacke-naredbe/> [↑](#footnote-ref-16)
17. Ibid [↑](#footnote-ref-17)
18. Ibid [↑](#footnote-ref-18)
19. Ibid [↑](#footnote-ref-19)
20. Ibid [↑](#footnote-ref-20)
21. Burd B.(2017): Beginning Programming with Java For Dummies. IDG Books Worldwide, Inc. Foster City, California. [↑](#footnote-ref-21)
22. Dibble, P. (2002). *Real-time Java platform programming*. Prentice Hall Professional. [↑](#footnote-ref-22)
23. Ibid [↑](#footnote-ref-23)
24. Kramer, D. (1996). The Java Platform. Mountain View, California. [↑](#footnote-ref-24)
25. Službena stranica NetBeansa-a, <https://netbeans.org/> [↑](#footnote-ref-25)
26. <https://zir.nsk.hr/islandora/object/etfos%3A2203/datastream/PDF/view> [↑](#footnote-ref-26)
27. <https://zir.nsk.hr/islandora/object/etfos%3A2203/datastream/PDF/view> [↑](#footnote-ref-27)
28. [https://www.javatpoint.com/intellij-vs eclipse#:~:text=The%20more%20plugins%20installed%20in,smoother%20as%20compared%20to%20Eclipse.](https://www.javatpoint.com/intellij-vs%20eclipse#:~:text=The%20more%20plugins%20installed%20in,smoother%20as%20compared%20to%20Eclipse.) [↑](#footnote-ref-28)
29. Ibid [↑](#footnote-ref-29)
30. <https://www.bluej.org/about.html> [↑](#footnote-ref-30)
31. <https://www.oracle.com/application-development/technologies/jdeveloper.html> [↑](#footnote-ref-31)
32. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf> [↑](#footnote-ref-32)
33. <http://www.efos.unios.hr/upravljanje-marketingom/wp-content/uploads/sites/228/2013/04/RPA_P4_MVC-web-aplikacije.pdf> [↑](#footnote-ref-33)