Modeliranje baze podata i izrada mobilne aplikacije za praćenje putnih naloga

Posavec, Dalibor

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic of Međimurje in Čakovec / Međimursko veleučilište u Čakovcu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:110:042921

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-28**

Repository / Repozitorij:

Polytechnic of Međimurje in Čakovec Repository - Polytechnic of Međimurje Undergraduate and Graduate Theses Repository

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

DALIBOR POSAVEC

MODELIRANJE BAZE PODATAKA I IZRADA MOBILNE APLIKACIJE ZA PRAĆENJE PUTNIH NALOGA

ZAVRŠNI RAD

Mentor:
mr. sc. Željko Knok

ČAKOVEC, 2020.

Sažetak

Tema ovoga završnoga rada modeliranje je baze podataka i izrada mobilne aplikacije za praćenje putnih naloga. Putni nalog kreiran je pomoću Android aplikacije od strane korisnika te poslan na provjeru, a administrator je e-mailom obaviješten o novom putnom nalogu za pregled. Putni nalog potvrđen je ili odbijen putem aplikacije. Ako je odbijen, korisnik je obaviješten e-mailom o njegovom neprihvaćanju.

Baza podataka izrađena je u SQL-u (Structured Query Language) korištenjem programa MySQL Workbench. Android aplikacija izrađena je u razvojnom okruženju Android Studija korištenjem Jave kao programskoga jezika. Za spajanje Baze podataka i Android aplikacije potreban je API izrađen u Springboot frameworku baziranom na Javi u intelliJ IDEA razvojnom okruženju. Dvije značajke (engl. features) Springboot-a: Spring Data JPA (Java Persistent API) pri čemu je odabran Hibernate kao njegov JPA davatelja usluge (eng. Provider) i Spring Security korištene su od strane autora.

U teorijskom djelu ovog rada ukratko su objašnjeni programski jezici Java i SQL te razvojna okruženja Android Studio, intelliJ IDEA i Springboot framework.

U praktičnom djelu opisan je postupak izrade Android aplikacije, API-a i SQL baze podataka s isječcima koda iz navedenih razvojnih okolina.

Ključne riječi: ***SQL, MySQL Workbench, Java, Android studio, intelliJ IDEA, Springboot, Spring Data JPA, Hibernate, Spring Security.***

Sadržaj

1. Uvod 1
2. Cilj rada 1
3. Programski alati 1
3.1 Java 2
3.2 SQL (Structured Query Language) 2
3.3 Springboot 3
3.3.1 Spring Data JPA 4
3.3.2 Hibernate 4
3.3.3 Spring Security 4
3.4 MySQL Workbench 5
3.5 IntelliJ IDEA 6
3.6 Android Studio 6
3.7 Modeliranje baze podataka 6
4. Aplikacija 7
4.1 Izrada baze podataka 7
4.2 Instalacija Servera 10
4.3 Izrada android aplikacije 14
5. Zaključak 40
6. Literatura 41
7. Dodatak 42
1. Instalacija MySQL Workbench-a 42
2. Instalacija IntelliJ IDEA 45
3. Instalacija Android studio-a 52

1. Uvod

Pisanje i predaja putnih naloga te uvid u dosad predane i potvrđene putne naloge olakšani su zaposlenicima korištenjem aplikacije. Aplikacijom je i ovlaštenoj osobi olakšano vođenje putnih naloga. Pregled putnih naloga zaposlenika te njihovo potvrđivanje ili odbijanje (ovisno o kriterijima definiranima na početku korištenja ove aplikacije) omogućeni su ovlaštenoj osobi. Prilikom korištenja aplikacije, zaposlenik je registriran, a nakon toga prijavljen u sustav. Odlukom o pisanju putnog naloga njegovi podaci (ime, prezime, zanimanje, radno mjesto) automatski su uzeti iz baze podataka i upisani u putni nalog te nema potrebe za njihovim ručnim unošenjem. Sva slobodna polja ispunjena su od strane zaposlenika kako bi nalog bio poslan na pregled. Nakon uspješnog slanja putnog naloga, ovlaštena osoba je obaviještena o postojanju novog putnog naloga. Putni nalog je pregledan, a zatim potvrđen ili odbijen. Ako je potvrđen, biva spremljen u bazu podataka. Ako je odbijen, ne biva spremljen u bazu podataka, a zaposlenik je e-mailom obaviješten o njegovom neprihvaćanju. Svi potvrđeni putni nalozi prikazani su zaposlenicima u aplikaciji jednako kao što su prikazani i svi kreirani i potvrđeni putni nalozi svih zaposlenika u bazi podataka ovlaštenoj osobi.

2. Cilj rada

Dizajniranje i programiranje Android aplikacije kojom bi slanje HTTP zahtjeva bilo omogućeno te ovisno o tome podaci poslani ili primljeni kao i kreiranje servera putem kojega određen HTTP zahtjev biva prepoznan, a povratna informacija o podacima iz baze podataka poslana, cilj su ovoga rada. Primljeni podaci optimizirani su u aplikaciji tako da je korisniku omogućen dobar vizualni pregled informacija poslanih od strane servera.

3. Programski alati

Programski jezici, razvojne okoline i framework bit će opisani i objašnjeni u ovome dijelu. Logika android aplikacije i API-a pisana je u programskom jeziku Java. Osim Jave kao programskog jezika bit će opisani SQL, zatim razvojne okoline Android studio, intelliJ IDEA, MySQL Workbench te framework Springboot sa svojim značajkama. (engl. *Features*).

3.1 Java

Jedan od najpoznatijih objektno orijentiranih programskih jezika jest Java. Razvoj Jave započet je 1991. godine u sklopu projekta „Green“, a završen projekt objavljen je u studenom 1995. godine. Javom kao objektno orijentiranim programskim jezikom označeno je da su metode razvijanja programske opreme sastavljene poput grupe međusobno povezanih objekata te je ona dizajnirana prvenstveno za izvršavanje u distribuiranom okruženju. Važnija značajka Jave svakako je „neovisnost o platformi“ čime je označeno da programi pisani u njoj mogu bez promjene koda biti izvođeni u bilo kojim operativnim sustavima ako za njih postoji JVM. Kodovi pisani u Javi pretvoreni su u tzv. „bytecode“ koji je pokrenut programom, operacijskim sustavom ili uređajima pomoću JAVA Interpretera. Bytecode nije izvršni kod, već je riječ o uputama dizajniranima za izvršavanje unutar JAVA virtual machine – JVM.

3.2 SQL (Structured Query Language)

Programski jezik namijenjen za rad s relacijskim bazama podataka jest SQL. Izrada i izmjena strukture baze podataka, dodavanje prava korisniku za pristup bazi ili tablici, traženje informacije od baze podataka i mijenjanje sadržaja baze podataka omogućeni su upravo putem SQL-a.

SQL naredbe grupirane su u četiri kategorije ovisno o njihovim funkcijama:

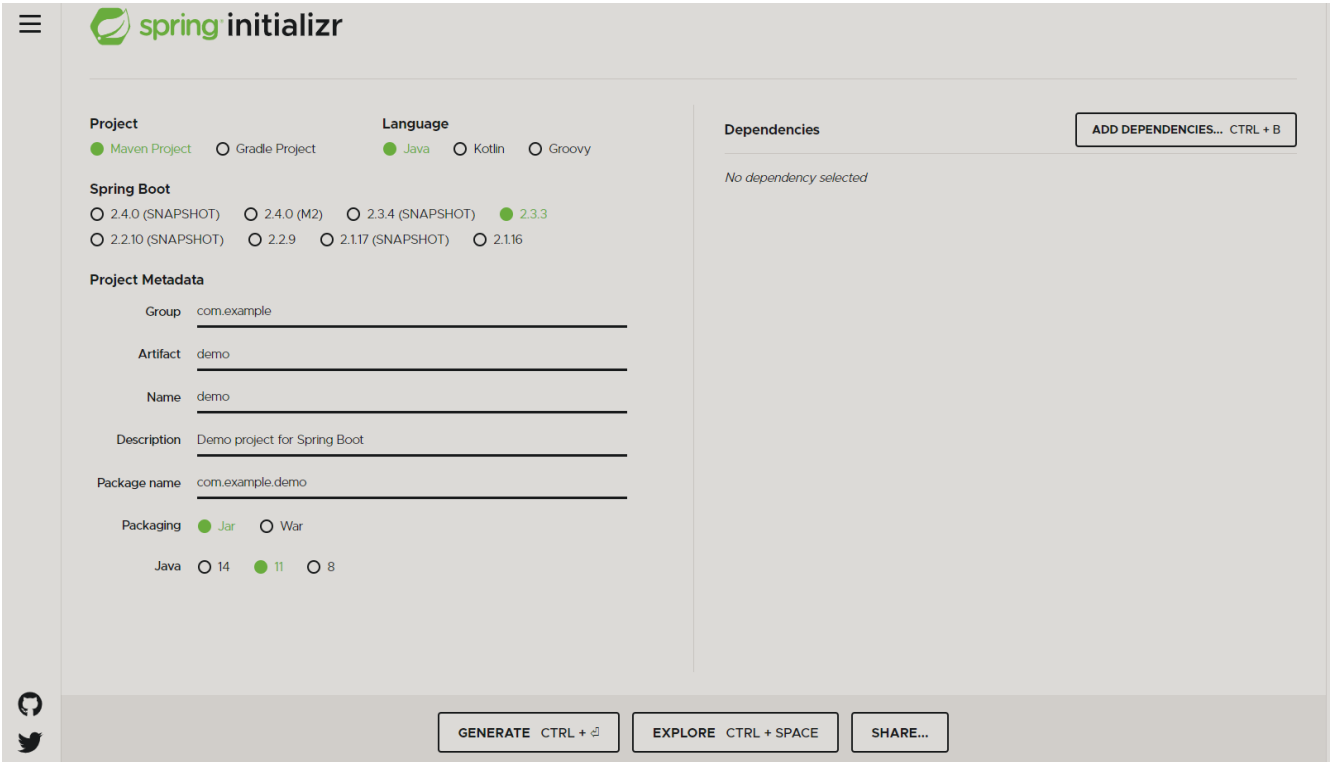
- Data Definition Language – omogućeno je kreiranje, mijenjanje i brisanje podatkovnih struktura baze podataka (tablice, pogledi…) i definiranje shema ili pogleda logičke razine baze, odnosno definiranje podatkovne strukture i veza između njih
- Data Manipulation Language – omogućeno je kretanje po bazi podataka te izvršavanje jednostavnih operacija kao što su upis, čitanje, promjena ili brisanje zapisa određene tablice (CRUD operacije – create, read, update, delete)
- Transaction Control Language - omogućeno je upravljanje transakcijama – nedjeljivim skupom DML naredbi čiji je rezultat pohranjen samo ako je uspješno izvršena svaka naredba iz tog skupa
- Data Control Language – omogućeno je upravljanje pristupom bazi podataka i njezinim objektima

3.3 Springboot

Spring Boot framework je otvorenoga koda (engl. *Open source*) baziran na programskom jeziku Java i razvijen od strane Pivotal Teama. Njegov je cilj kraća duljina koda i pružanje najlakšeg načina razvoja internetske aplikacije. Vrijeme potrebno za razvoj internetske aplikacije skraćeno je konfiguracijom bilješki (engl. *Annotation*) i zadanim kodovima Spring Boota. Stvaranje samostalne aplikacije s manje konfiguracija ili s gotovo nikakvom konfiguracijom olakšano je putem Spring Boota. Njegova posebna značajka (engl. Feauture) jest autokonfiguracija (automatsko konfiguriranje klase na temelju zahtjeva)

Prednosti Spring Boota su:

- fleksibilan način konfiguriranja JavaBeans klasa, XML konfiguriranje i transakcija baze podataka
- automatsko konfiguriranje tj. nije potrebno ručno konfiguriranje
- jednostavnije prilagođavanje i upravljanje
- jednostavnije pokretanje
- ponuda značajki (engl. *Features*) spremnih za proizvodnju
- olakšavanje upravljanja ovisnostima (engl. *Dependency management*)
- pomoć pri izravnoj ugradnji Tomcata, Jettyja ili Undertowa



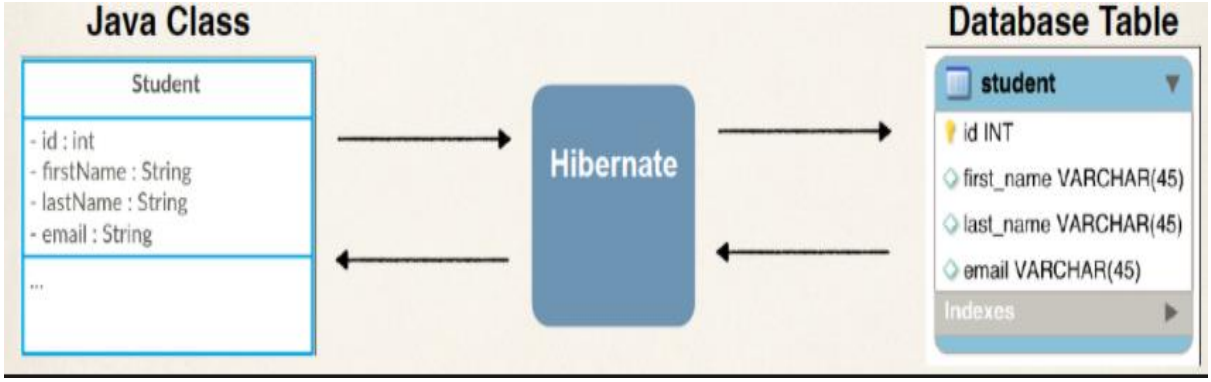
Slika 1. Spring initializr (izvor: Autor)

3.3.1 Spring Data JPA

Jednostavna implementacija spremišta temeljenih na JPA olakšana je putem Spring Data JPA koji je dio Spring Data. Ovaj je modul karakteriziran bavljenjem poboljšanom podrškom za slojeve pristupa podacima temeljenim na JPA i olakšavanjem izgradnje aplikacija na Springu odlikovanom korištenjem tehnologije za pristup podacima. Implementacija sloja pristupa podacima u aplikaciji dugo je vremena bila glomazna, točnije za izvršavanje jednostavnih upita, kao i za paginiranje i reviziju bilo je napisano previše koda. Značajno poboljšanje implementacije slojeva pristupa podacima smanjenjem napora na količinu koja je stvarno potrebna cilj je Spring Data JPA. Sučelja spremišta (engl. *Interface repository*) pisana su od strane programera, uključujući i prilagođene metode pronalaženja, a implementacija je automatski osigurana putem Springa.

3.3.2 Hibernate

Objektno-relacijsko mapiranje ili ORM tehnika je u programiranju za mapiranje objekata iz aplikacije u tablice relacijske baze podataka, a objektno-relacijsko rješenje mapiranja za Javu jest Hibernate. Referentna implementacija Java Persistent API-ja pružena je od strane Hibernatea koji je zbog toga izvrstan izbor kao ORM alat.



Slika 2. Hibernate (izvor: https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1)

3.3.3 Spring Security

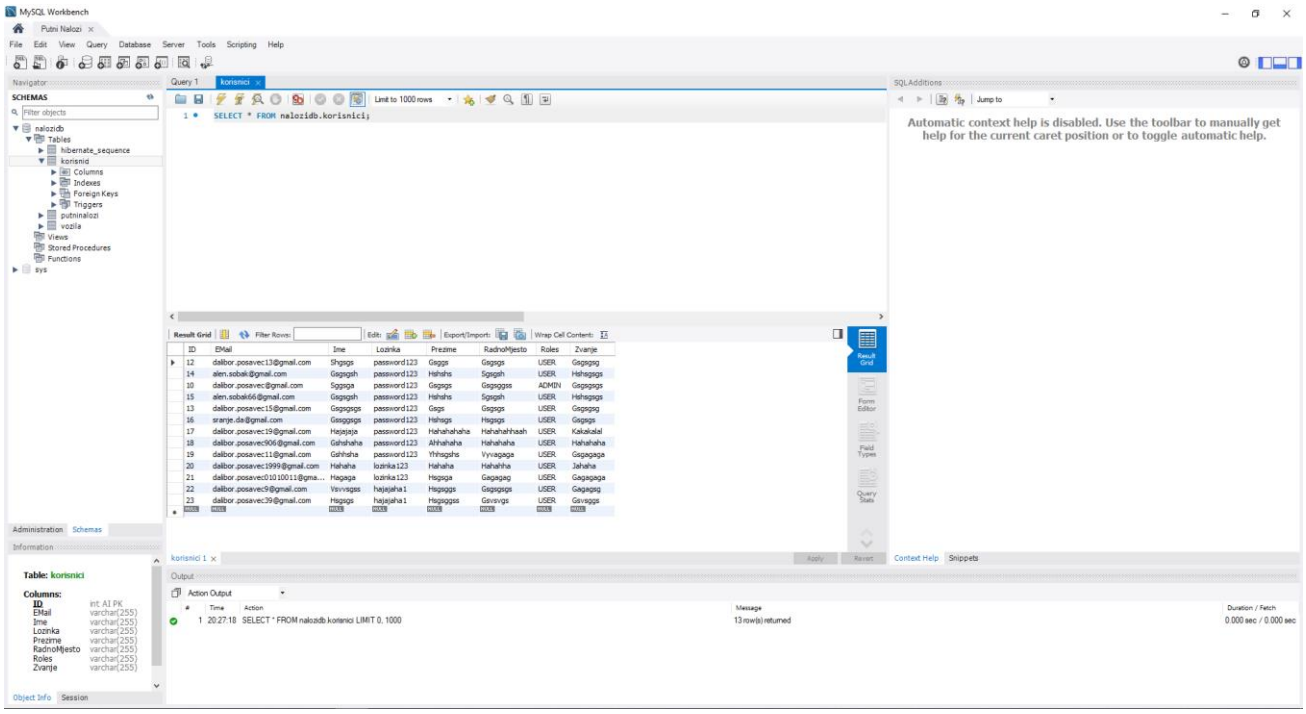
Vrlo prilagodljiv i moćan okvir (engl. Frame) za provjeru autentičnosti i kontrole pristupa jest Spring Security. To je standard za zaštitu aplikacija temeljenih na Springbootu. Spring Security okvir (engl. *Frame*) fokusiran je na provjere autentičnosti i autorizacije Java aplikacija. Stvarna moć Spring Securityja jest njegovo lako proširenje kako bi i prilagođeni zahtjevi programera bili zadovoljeni.

Značajke (engl. Features) Spring Securityja su:

- sveobuhvatna i proširiva podrška za autentifikaciju i autorizaciju
- zaštita od napada kao što su fiksiranje sesija, vraćanje klikova, krivotvorenje zahtjeva za više mjesta, itd.
- integracija API-ja za servlet
- neobavezna integracija sa Spring Web MVC

3.4 MySQL Workbench

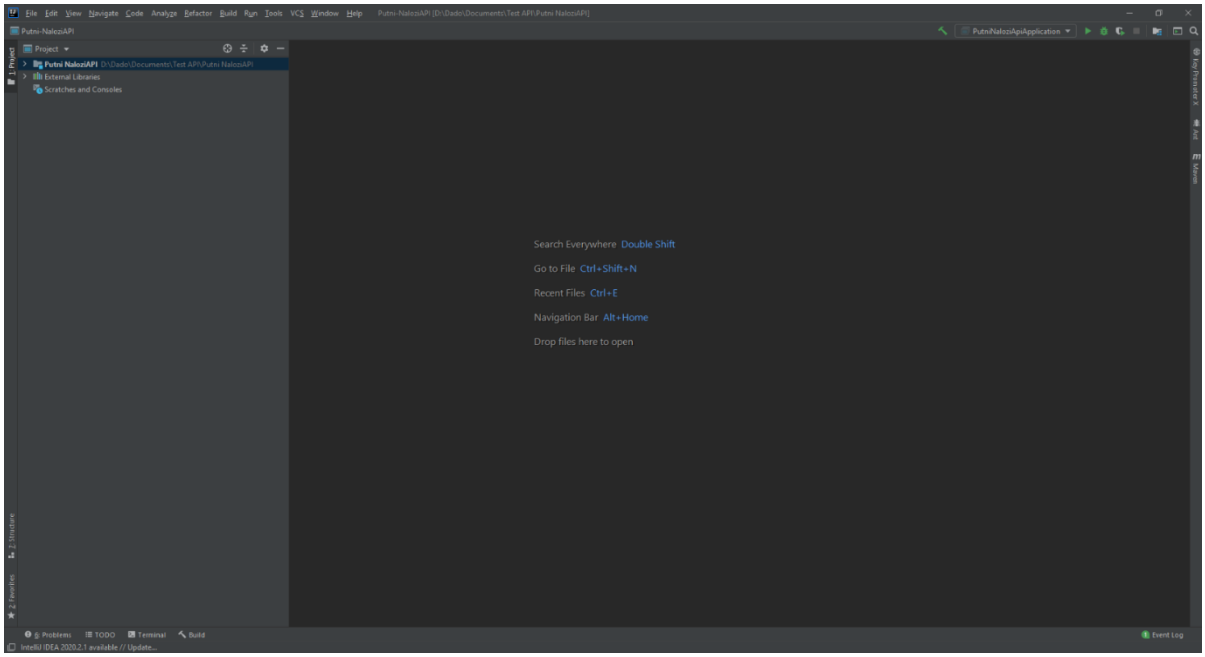
Vizualni alat za dizajn baze podataka pomoću kojega su integrirani razvoj SQL-a, administracija, dizajn baze podataka, stvaranje i održavanje u jedinstveno integrirano razvojno okruženje za MySQL sustav baza podataka jest MySQL Workbench.



Slika 3. MySQL Workbench nakon konekcija na bazu podataka (izvor: Autor)

3.5 IntelliJ IDEA

Razvojno okruženje napisano na Javi za razvoj računalnog softvera jest IntelliJ IDEA. Razvijeno je od strane JetBrainsa te dostupno u dvije varijante: u plaćenom izdanju (Ultimate) i besplatnom izdanju (Community Edition).



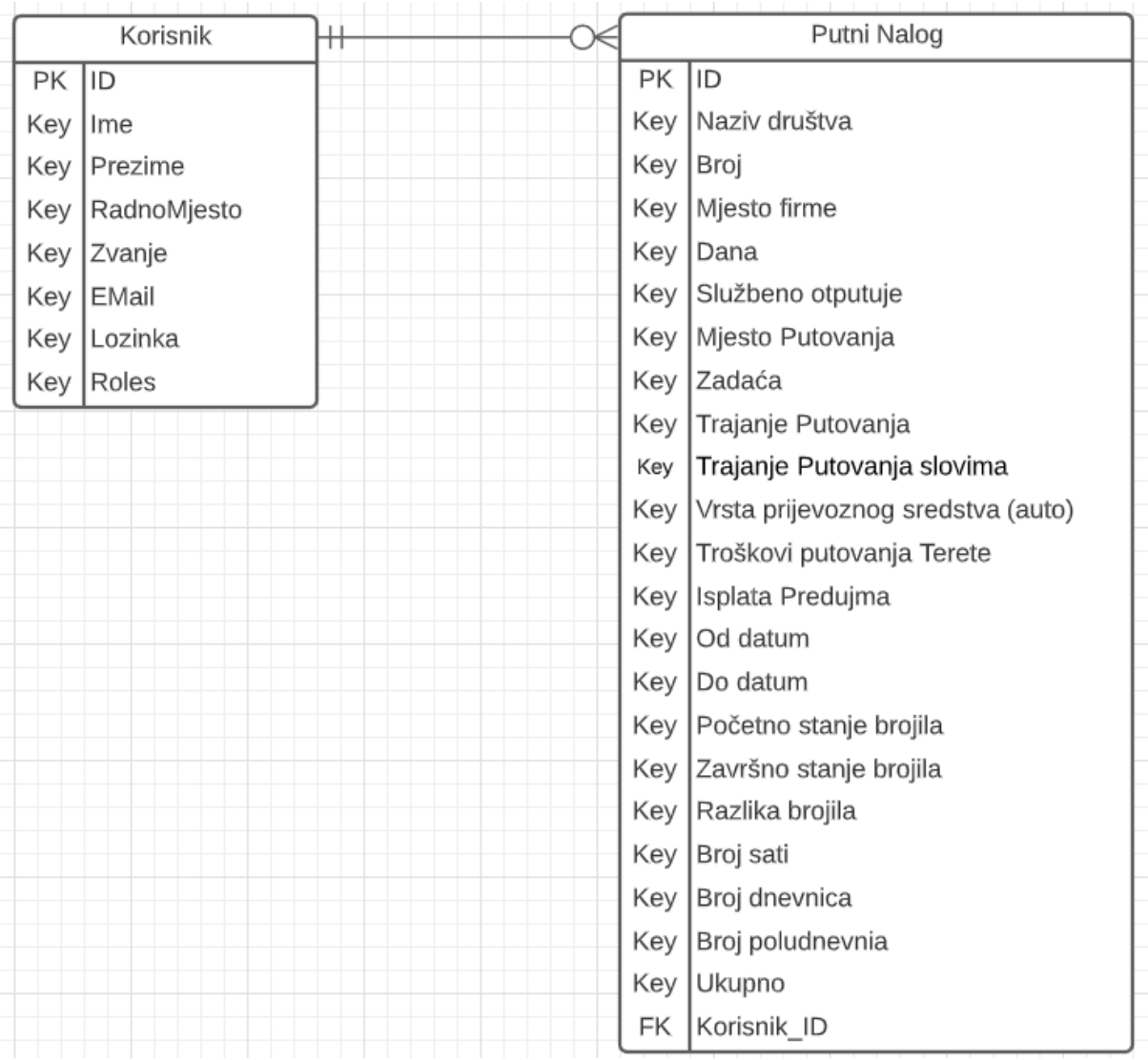
Slika 4. IntelliJ IDEA (izvor: Autor)

3.6 Android Studio

Službeno razvojno okruženje za Googleov operativni sustav Android jest Android Studio koji je izgrađen na JetBrains-ovom IntelliJ IDEA softveru te dizajniran posebno za razvoj Androida.

3.7 Modeliranje baze podataka

Baza podataka modelirana je te kasnije izrađena u MySQL Workbenchu. Baza je sastavljena od dva entiteta – Korisnika i Putnog naloga. Putnih naloga je nula ili više po jednom korisniku, a veza između dva entiteta označena je s „0<“ (nula ili više). Kreirani putni nalog kreiran je od strane jednog i isključivo jednog korisnika te takvu vezu označujemo sa „ | | “ (jedan i isključivo jedan). Atributom PK označen je primarni ključ (engl. *Primary key*), a atributom FK označen je vanjski ključ (engl. *Foreign key*).

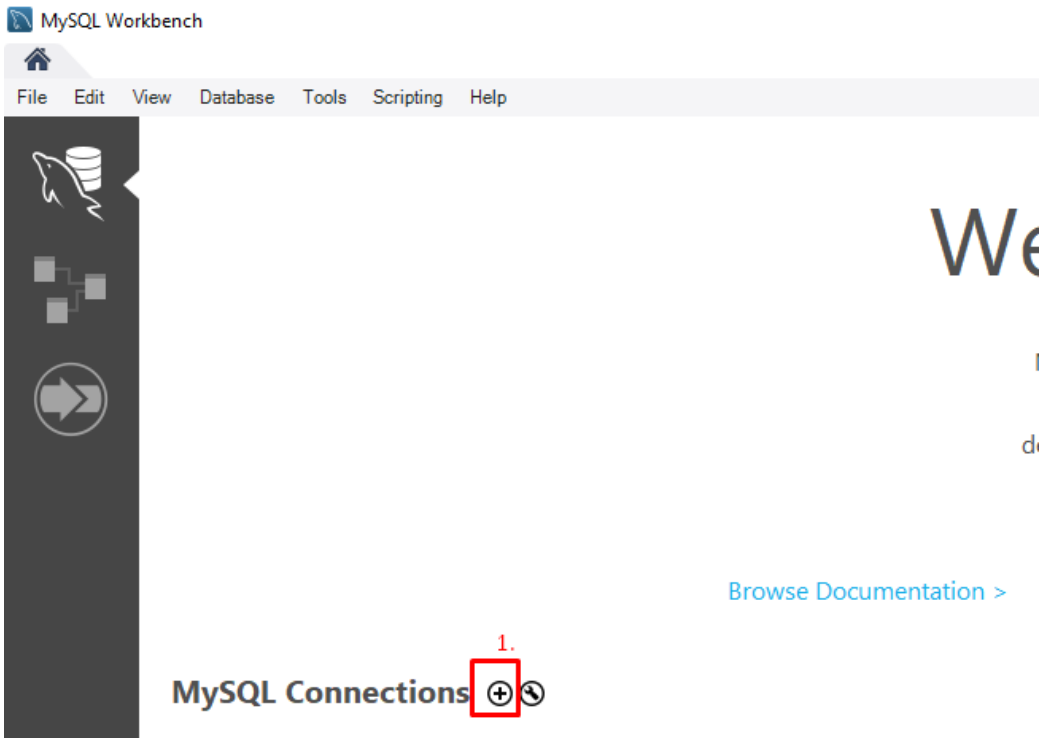


Slika 5. Modeliranje baze podataka (izvor: Autor)

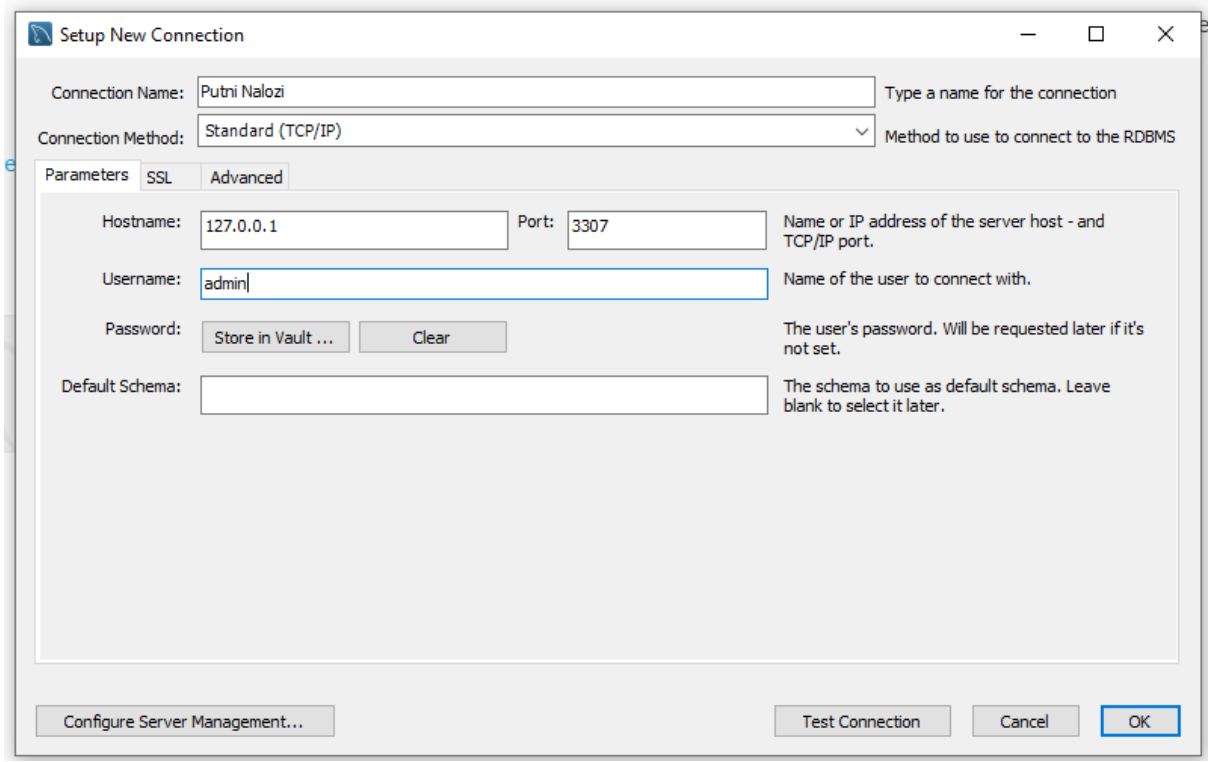
4. Aplikacija

4.1 Izrada baze podataka

Nova konekcija po imenu „Putni Nalozi“ napravljena je u MySQL Workbenchu. Port konekcije je 3307, Username admin, a password password. Username i password korišteni su u Springboot aplikaciji zbog povezanosti s bazom podataka.

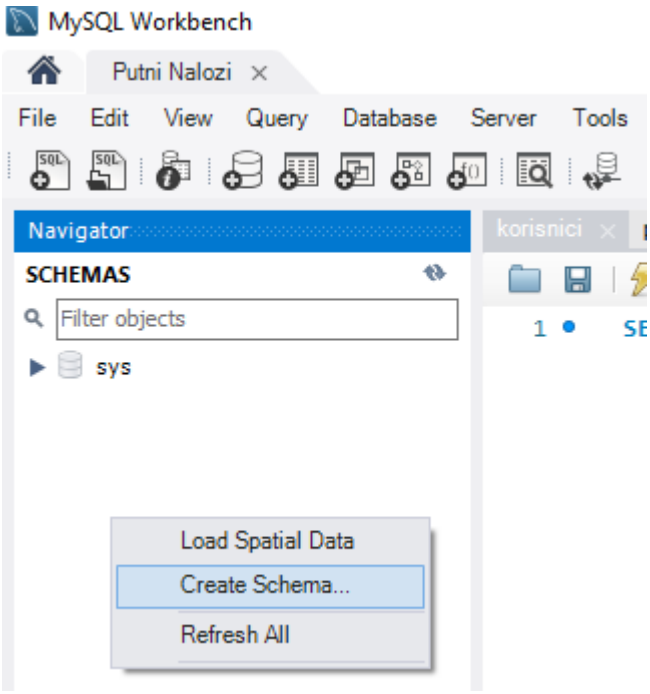


Slika 6. Kreiranje nove MySQL konekcije (izvor: Autor)

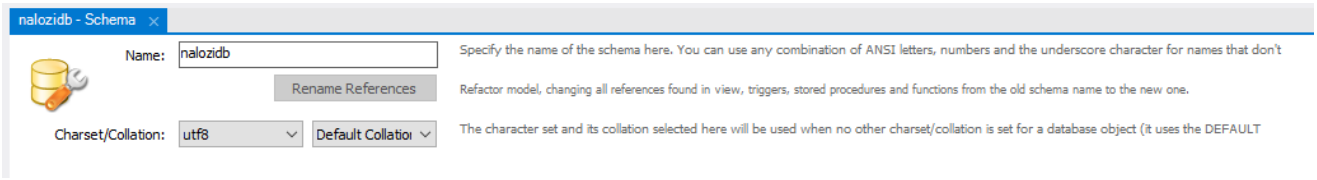


Slika 7. Konfiguracija novo kreirane konekcije (izvor: Autor)

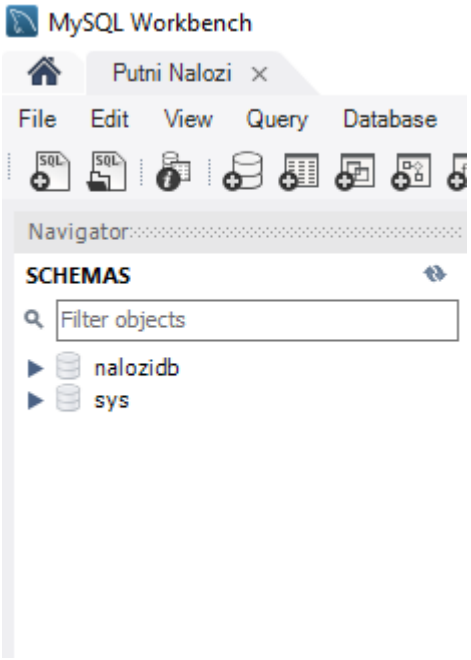
Shema baze podataka kreirana je nakon kreiranja konekcije. U „Schemas“ na prazno mjesto pritiskom na desnu tipka miša odabrano je „Create Schema“, a ta novo nastala shema nazvana je „nalozidb“.



Slika 8. Kreiranje nove sheme (izvor: Autor)



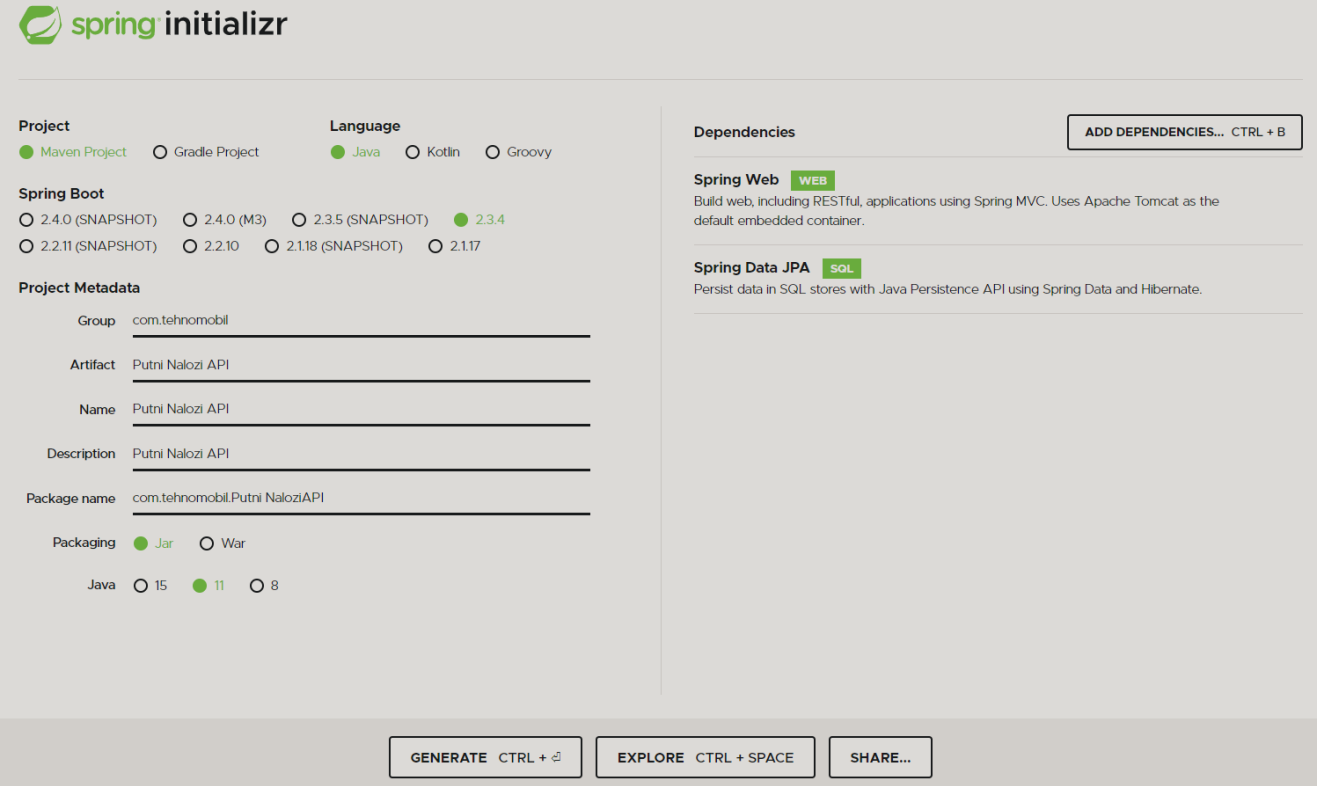
Slika 9. Nakon pritiska opcije „Create Schema“ (izvor: Autor)



Slika 10. Nakon uspješno kreirane sheme (izvor: Autor)

4.2 Instalacija Servera

Server je instaliran putem web-stranice https://start.spring.io/, a Spring inicijaliziran odabirom Maven projekta, Jave kao programskog jezika (verzija Jave 11), stabilne verzije Spring Boota 2.3.4., zatim odabirom „Dependencies“: Spring Web i Spring Data JPA te naposljetku ispunjenjem ostalih detalja vezanih uz projekt (ime, opis, (engl. *Description*) i ostalo).



Slika 11. Spring inicijalizacija projekta (izvor: Autor)

Zatim je projekt preuzet i otvoren u IntelliJ-u. U konkretnome projektu u intelliJ-u pod package name „com.tehnomobil.Putni NaloziAPI“ u resources> aplication properties.

```
server.port=8090

spring.datasource.url=jdbc:mysql://localhost:3307/nalozidb?use
Unicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatet
imeCode=false&serverTimezone=UTC

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.username=admin

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.naming.implicit-
strategy=org.hibernate.boot.model.naming.ImplicitNamingStrateg
yLegacyJpaImpl

spring.jpa.hibernate.naming.physical-
strategy=org.hibernate.boot.model.naming.PhysicalNamingStrateg
yStandardImpl

spring.jpa.hibernate.naming_strategy=org.hibernate.cfg.EJB3Nam
ingStrategy
```

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.
MySQL5Dialect
```

server.port=8090 - pomoću ove linije koda odabran je broj porta servera.
Dijelovi koda u kojima su pisane hibernate automatski su dodani prilikom dodavanja dependenciesa Spring Dana JPA u Spring initializru (Slika 11). Dodavanjem ovih linija koda u properties stvorena je konekcija između baze podataka i servera:

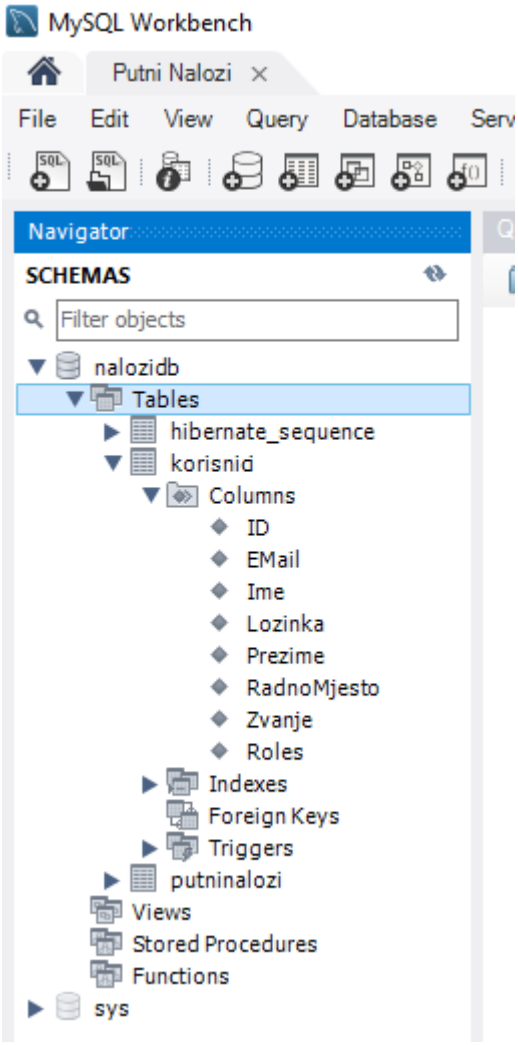
spring.datasource.url=jdbc:mysql://localhost:3307/nalozidb

Ovo je najvažniji dio naredbe jer je ovdje dan URL aplikaciji kojim je vođena do baze podataka. Baza podataka nazvana je „nalozidb“ te smještena na portu 3307. To je zapravo definirano korak prije u MySQL Workbenchu (Slika 9), dok je drugi dio ove naredbe vezan za vremensku zonu.

spring.datasource.username=admin
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update

Aplikacija je spojena na bazu podataka kako bi joj bilo omogućeno čitanje ili pisanje po podacima te zato trebaju biti upisani spring.datasource.username i spring.datasource.password (u ovome slučaju admin i password). (Slika 7).

spring.jpa.hibernate.ddl-auto=update – ovime je označena radnja u kojoj bi baza podataka prilikom novog upisa podatka ili brisanja postojećeg podatka bila ažurirana. Tablica preko koje su izvršeni svi SQL upiti po imenu hibernet_sequence, kreirana je putem Hibernatea, dok su dvije tablice i njihovi atributi kreirani korištenjem Hibernatea od strane programera.



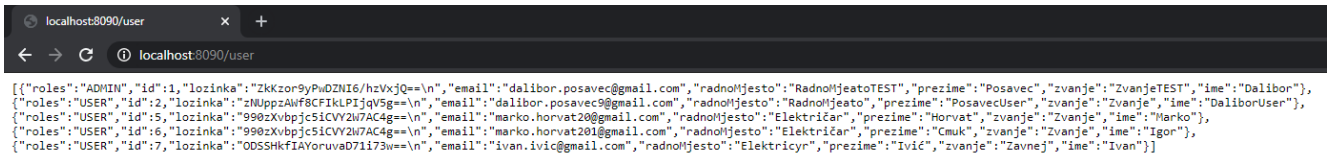
Slika 12. Tablice i atributi „korisnici“ tablice kreirani od Hibernatea nakon uspješne konekcije baze podataka i Servera (izvor: Autor)

Testiranjem HTTP u internetskom pregledniku određene su metode koje će biti izvršene na serveru ovisno o zahtjevu klijenta. U ovome primjeru to je slanje svih registriranih korisnika koji su u bazi podataka u JSON objekt. U klasi kontroler, URL i HTTP zahtjevi upravljani su od strane kontrolera. Metode u toj klasi s oznakom @RequestMapping aktivirane su kada je HTTP zahtjev (engl. Request) poslan od strane korisnika. Više je vrsta HTTP zahtjeva: GET - dobivanje podataka od servera, POST - slanje podataka na server, DELETE – brisanje podataka sa servera, PUT – ažuriranje podataka na serveru. U ovome slučaju riječ je o GET metodi koja je aktivirana unosom URL-a http://localhost:8090/user u internetski preglednik.

```
@RequestMapping("/user")
    public List<User> getAllUsers() {
        return myUserDetailsService.getAllUsers();}
```

U klasi services korištenoj u sloju usluge i kojom su izvođeni servisni zadaci, pomoću Hibernatea rađeni su SQL upiti i izvađeni su podaci iz baze podataka. Iz ove klase određeni podaci proslijeđeni su metodom u klasu kontroler, a JSON objekt poslan je klijentu od strane kontrolera.

```
public List<User> getAllUsers() {
        List<User> users = new ArrayList<>();
        userRepository.findAll().forEach(users::add);
        return users;
    }
```



Slika 13. Slanje JSON objekta svim korisnicima koji su u bazi podataka
(izvor: Autor)

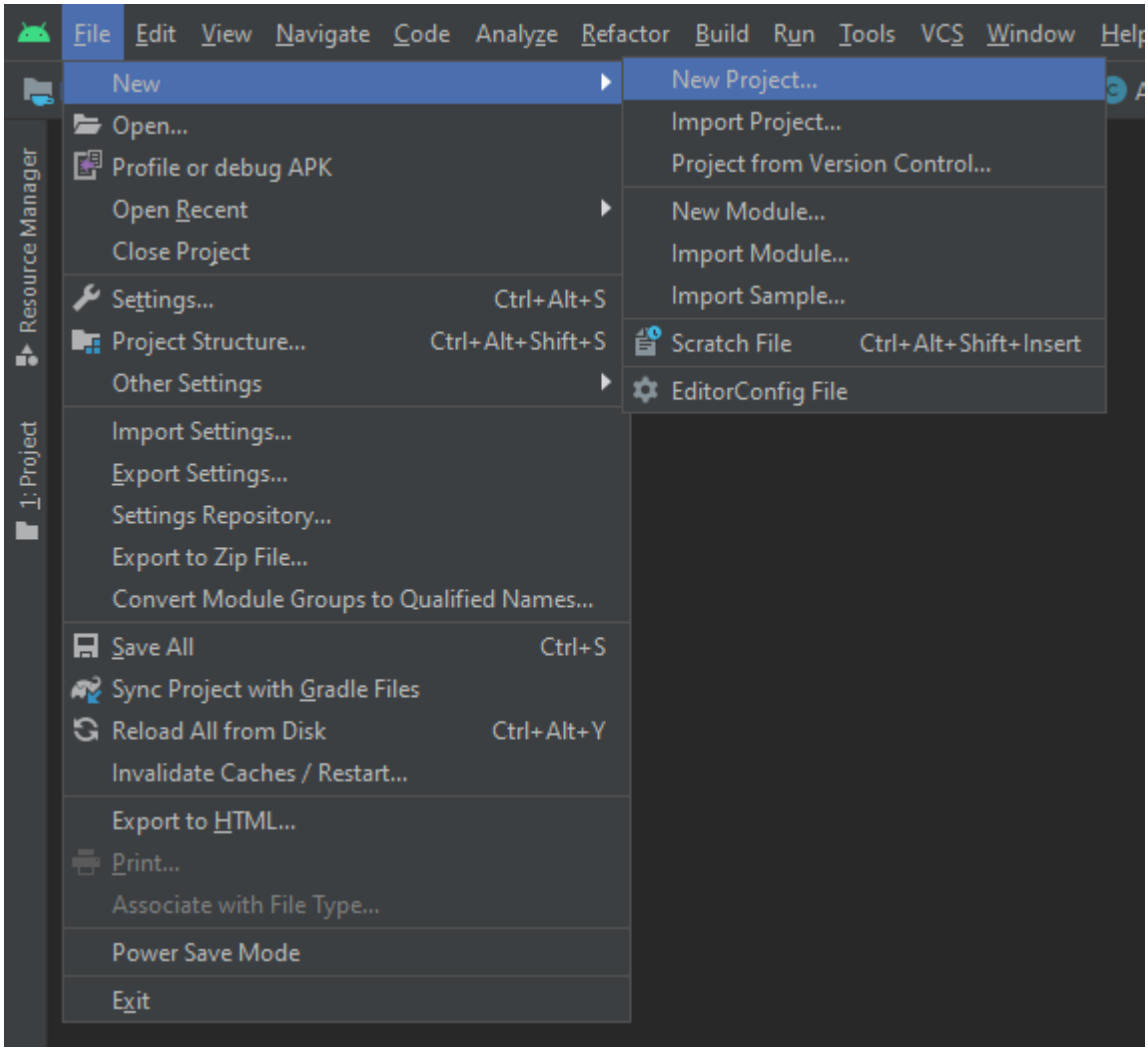
4.3 Izrada android aplikacije

Novi projekt pod imenom „Putni Nalozi“ napravljen je u programu Android Studio. Prava za povezivanje na Internet dana su aplikaciji jer je riječ o internetskoj aplikaciji. Za davanje prava potreban je ulazak u manifests > AndroidManifest.xml gdje će biti dodana

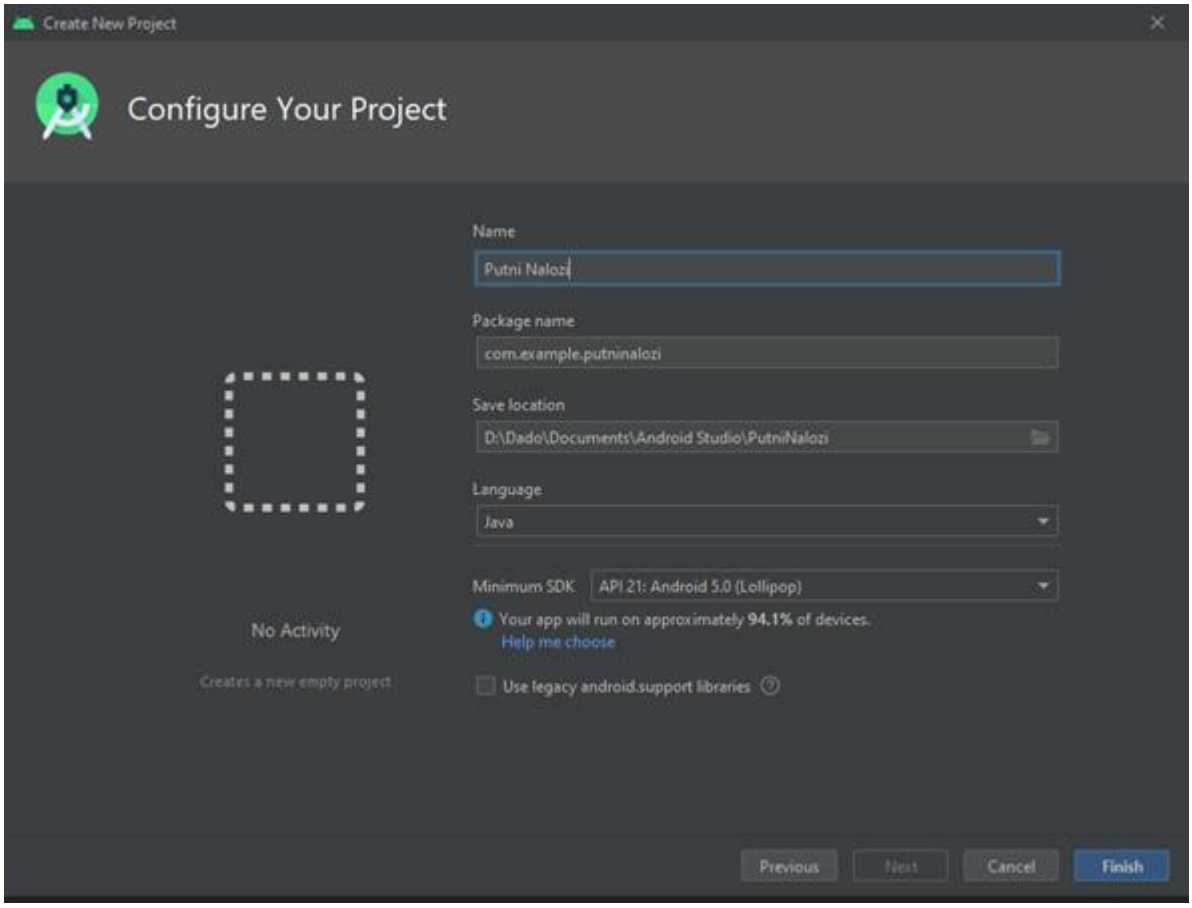
```
<uses-permission android:name="android.permission.INTERNET" />
```

linija koda.

Retrofit korišten je za slanje i primanje HTTP zahtjeva preko aplikacije. Njime su omogućeni brzo i jednostavno pretvaranje HTTP API u Java Interface te brzo i lako slanje ili primanje HTTP zahtjeva kroz aplikaciju. Retrofit mora biti implementiran u aplikaciju za što su potrebni ulazak u Gradle Scripts > build.gradle(Module: app) te u „dependencies“ stavljena „ implementation 'com.squareup.retrofit2:retrofit:2.9.0' “ linija koda.



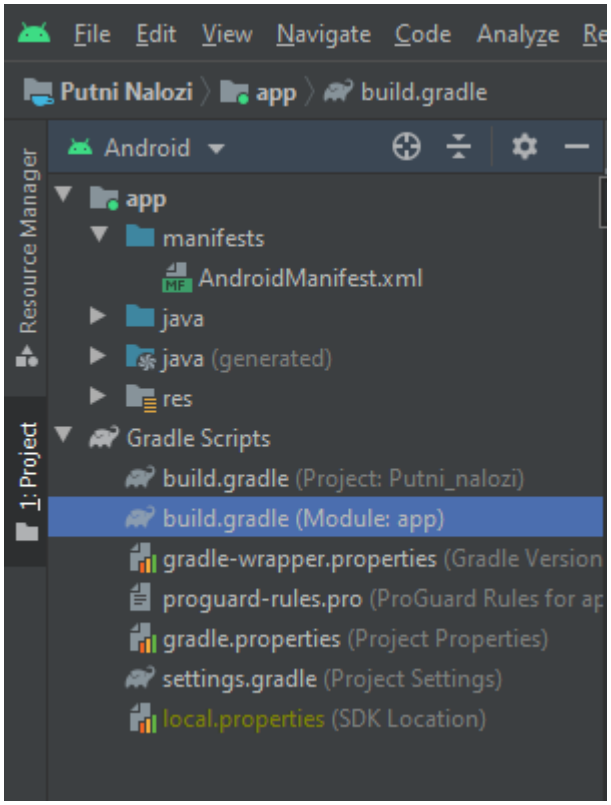
Slika 14. Kreiranje novoga projekta u Android studiju (izvor: Autor)



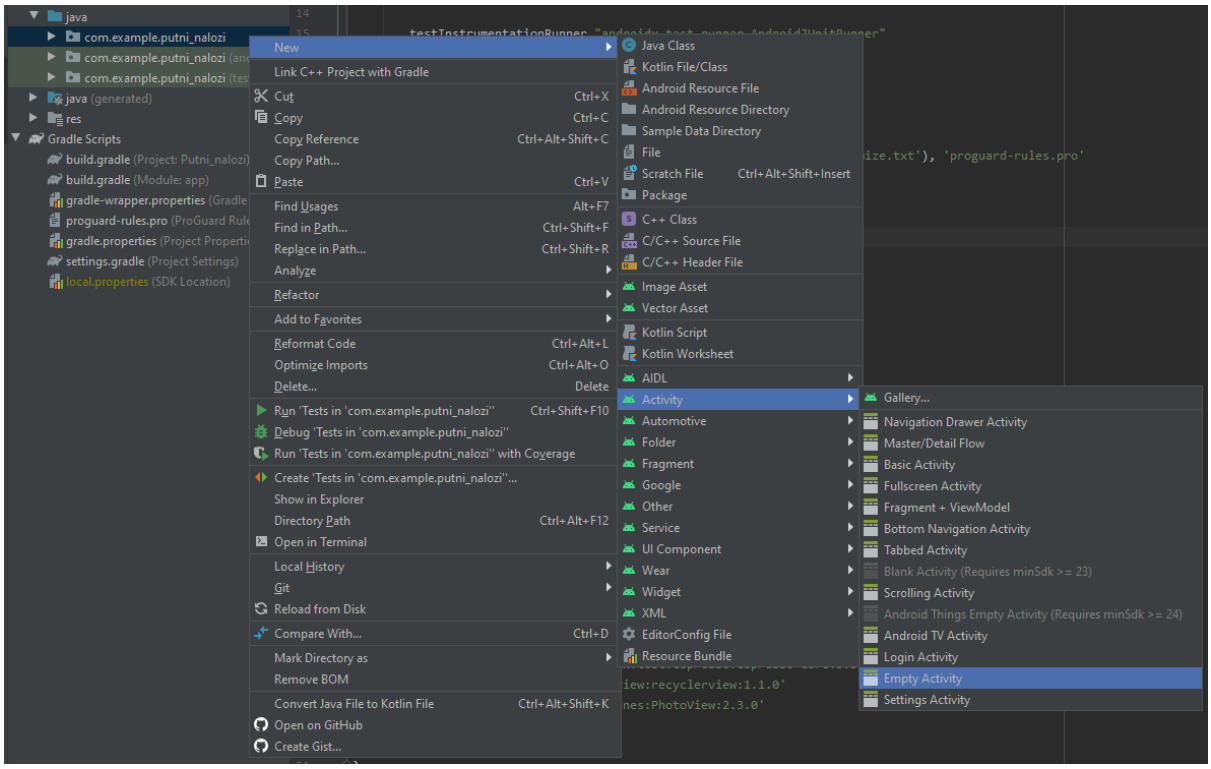
Slika 15. Konfiguriranje novonastalog projekta (izvor: Autor)

Odabran je „Minimum SDK“ Android 5.0 čime je određen rad aplikacije na svakom uređaju s minimalnim Android operativnim sustavom 5.0 ili više. Odabran je „No Activity“ čime je zapravo označen prazni projekt. Activityji su kreirani od strane programera u mapi Java pritiskom na desnu tipku miša New > Activity > Empty Activity. Activity je temeljni pojam Android Studija. To je događaj koji je prikazan nakon korisnikova klika na aplikaciju ili kad je prebačen iz jednog djela aplikacije na drugi ili na potpunu drugu aplikaciju. Activity je sastavljen od 2 dijela: XML-a – dizajnerskog dijela activityja zaduženog za izgled određenog activityja i Java klase. Logika u pozadini upravljana je Java klasom kojom je dan život određenim XML komponentama.

Primjer. Poslije pritiska određene tipke izvršena je određena akcija

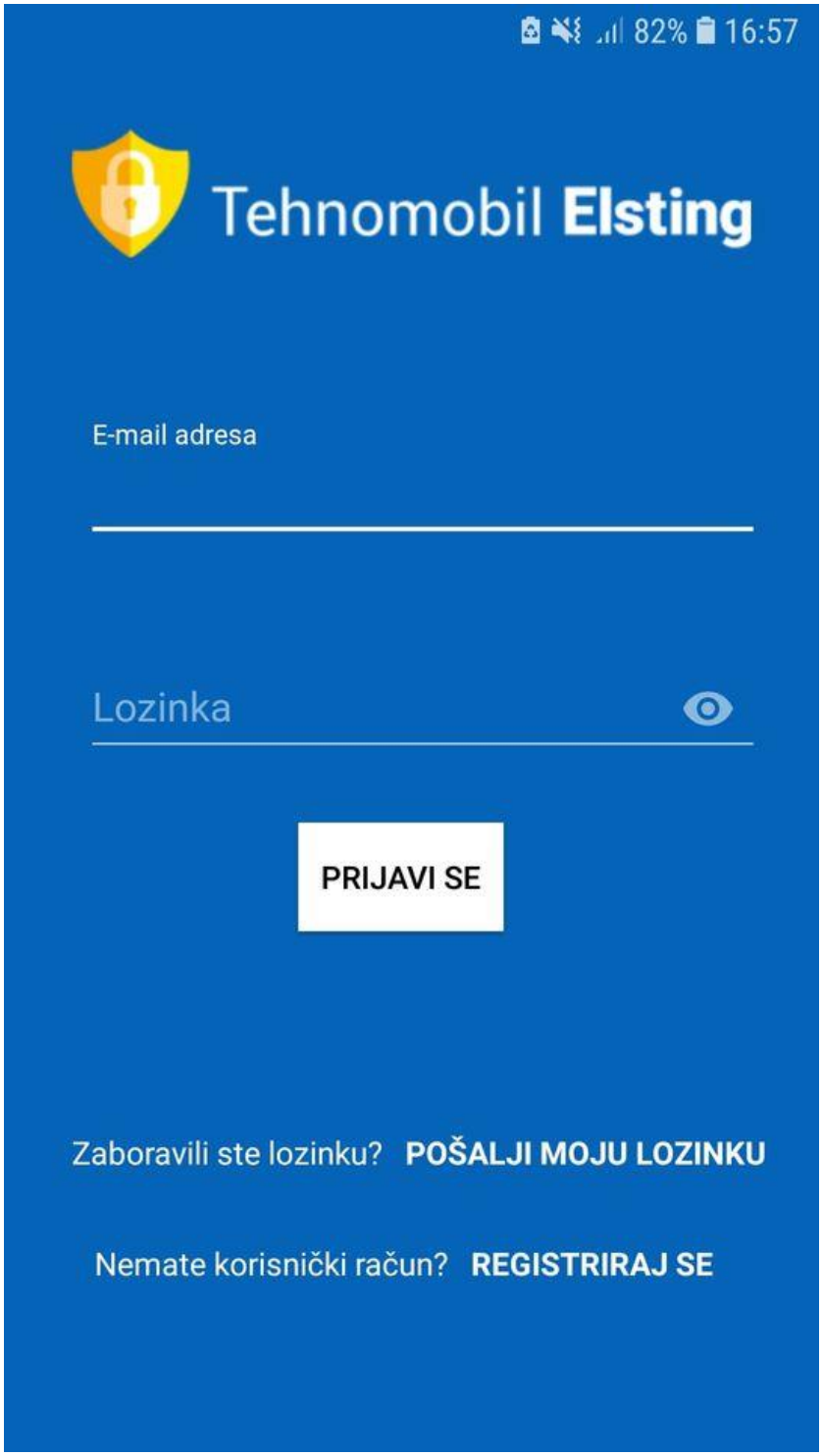


Slika 16. AndroidManifest i build.gradle(Module: app) generirani su nakon što je projekt kreiran (izvor: Autor)

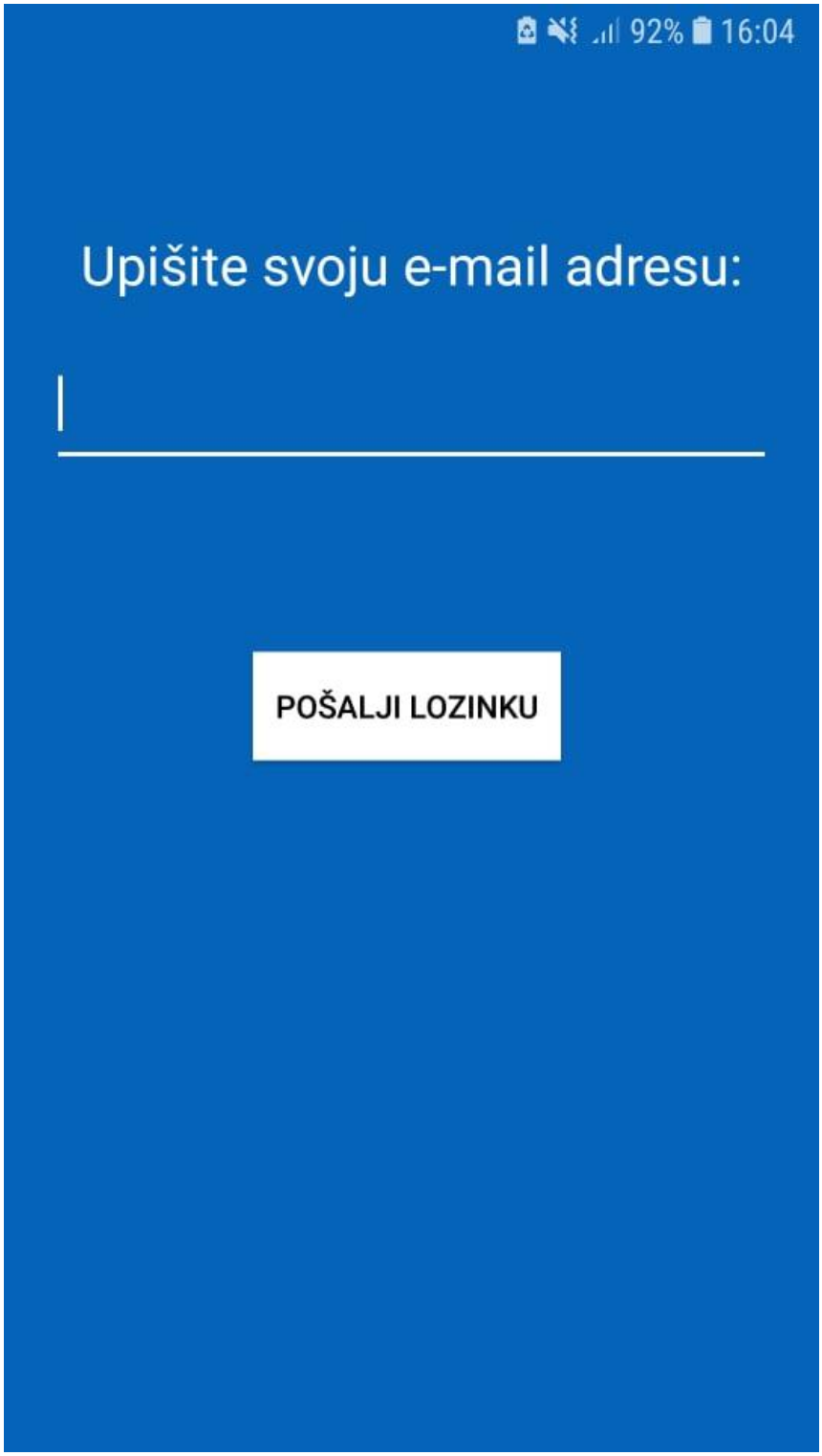


Slika 17. Kreiranje novog praznog Activityja (izvor: Autor)

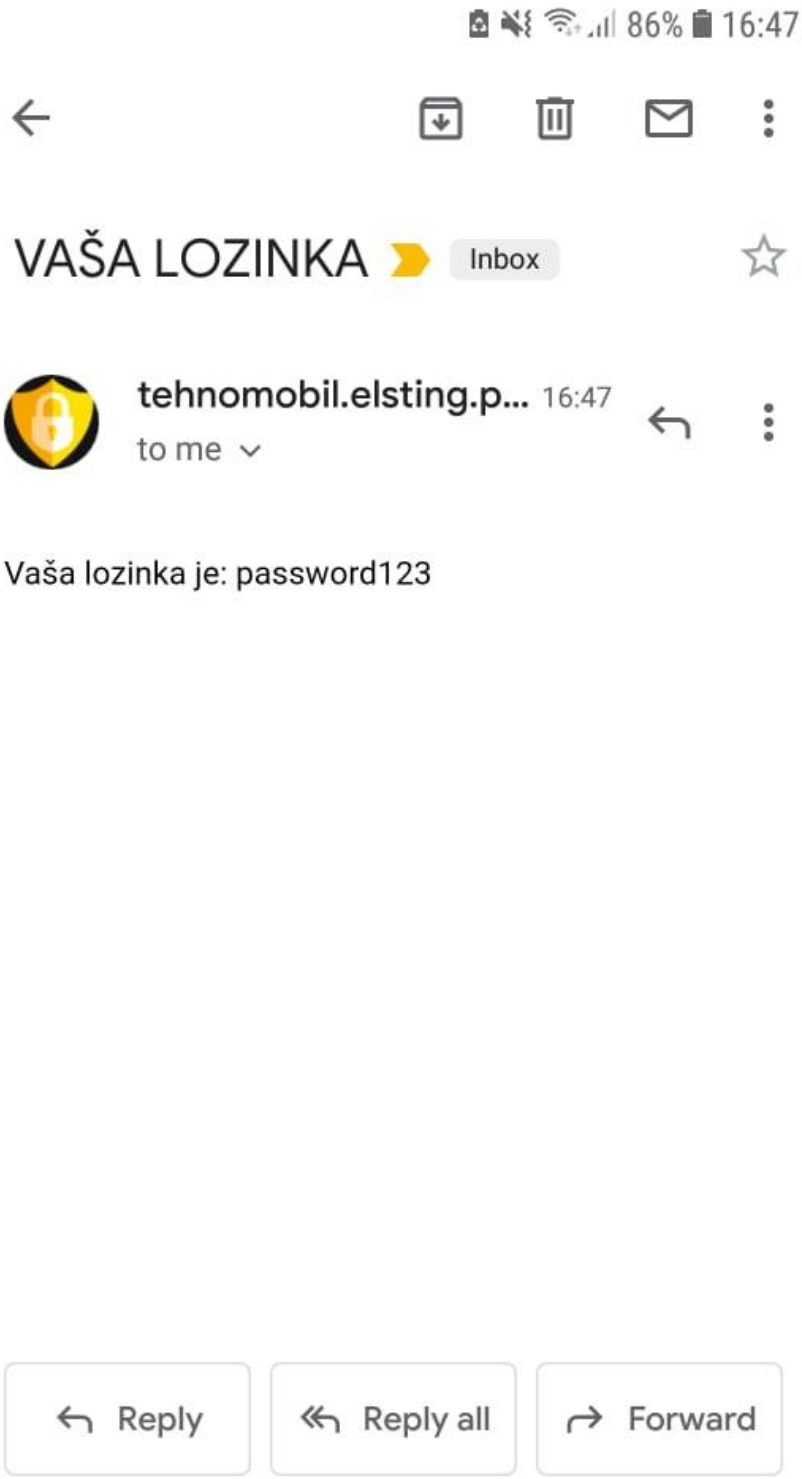
Početni zaslon nakon pokretanja aplikacije na Android uređaju



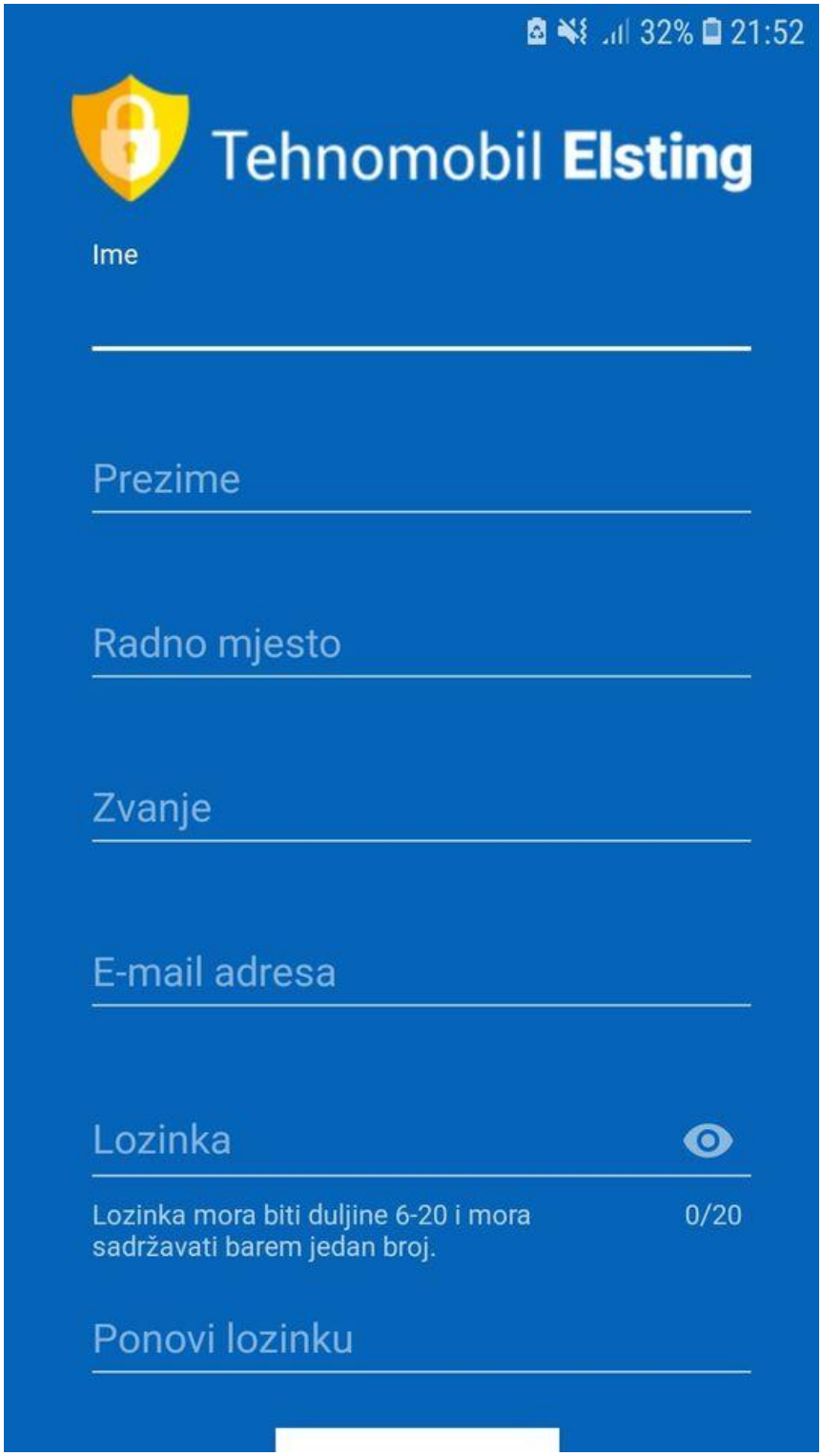
Slika 18. Dizajn Prijava activityja na mobitelu (izvor: Autor)



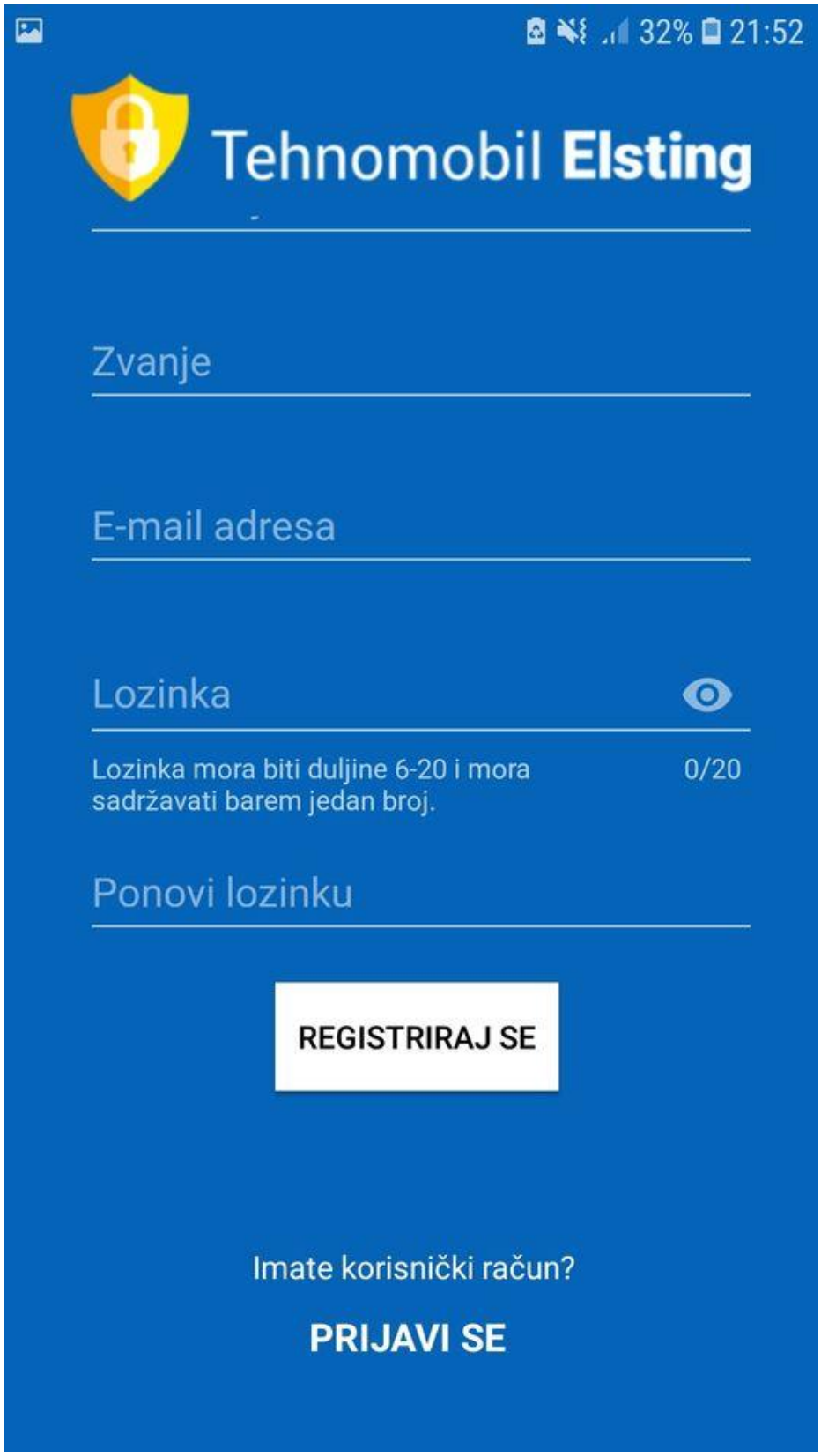
Slika 19. Dizajn Pošalji moju Lozinku activity na mobitelu (izvor: Autor)



Slika 20. Pristigla pošta na e-mailu nakon pritiska na gumb „Pošalji Lozinku“ (izvor: Autor)



Slika 21. Dizajn Registracije activity na mobitelu – 1.dio (izvor: Autor)



Slika 22. Dizajn Registracije activity na mobitelu – 2.dio (izvor: Autor)

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	EMail	Ime	Lozinka	Prezime	RadnoMjesto	Zvanje	Roles
1	dalibor.posavec@gmail.com	Dalibor	ZkKzor9yPwDZNI6/hzVxjQ==	Posavec	RadnoMjeatoTEST	ZvanjeTEST	ADMIN
2	dalibor.posavec9@gmail.com	DaliborUser	zNUppzAWf8CFIkLPIjqV5g==	PosavecUser	RadnoMjeato	Zvanje	USER
5	marko.horvat20@gmail.com	Marko	990zXvbpjc5iCVY2W7AC4g==	Horvat	Električar	Zvanje	USER
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Slika 25. Spremljeni podaci u bazi podataka nakon registracije u tablici Korisnici (izvor: Autor)

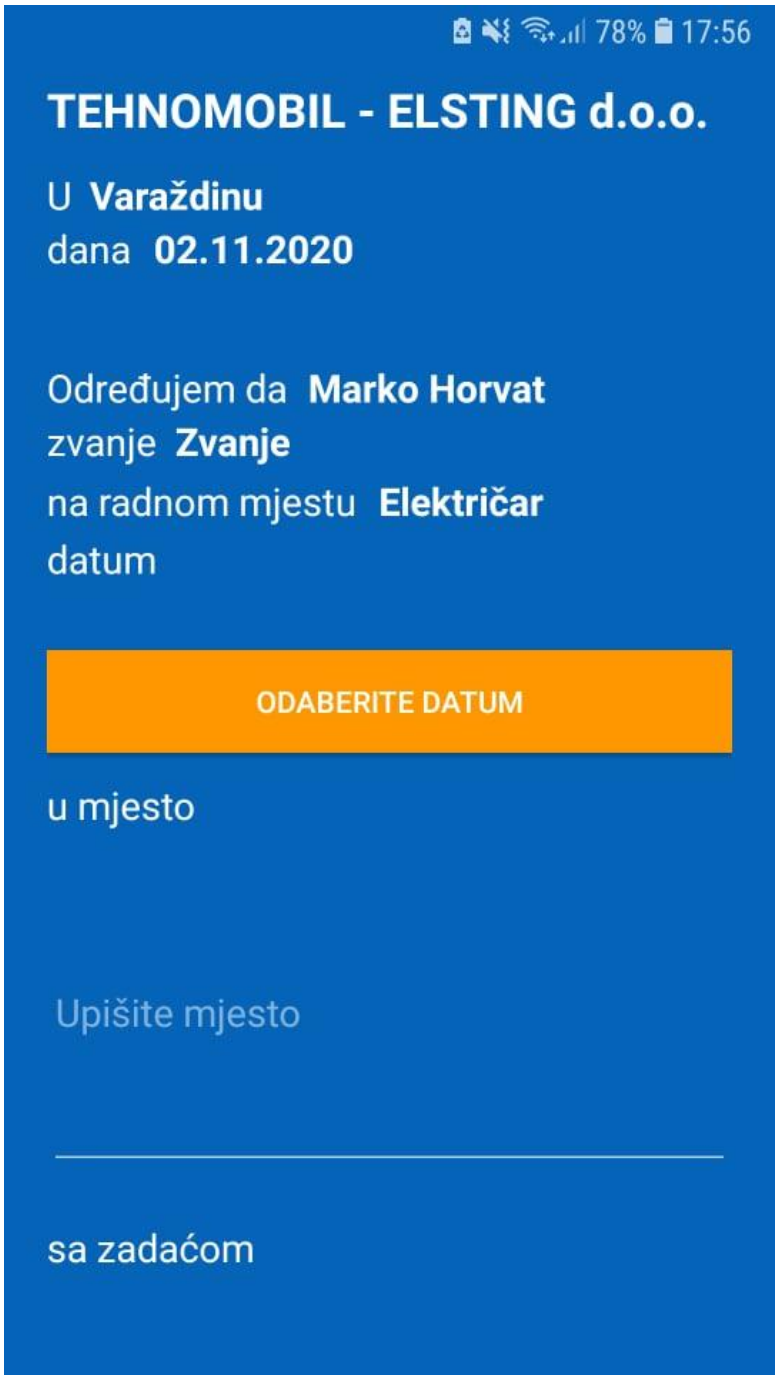
Korisnici su preusmjereni na Korisnik početni zaslon ili Administrator početni zaslon ovisno o ulozi.

Slika 23. Dizajn Korisnik početni zaslon activity na mobitelu (izvor: Autor)

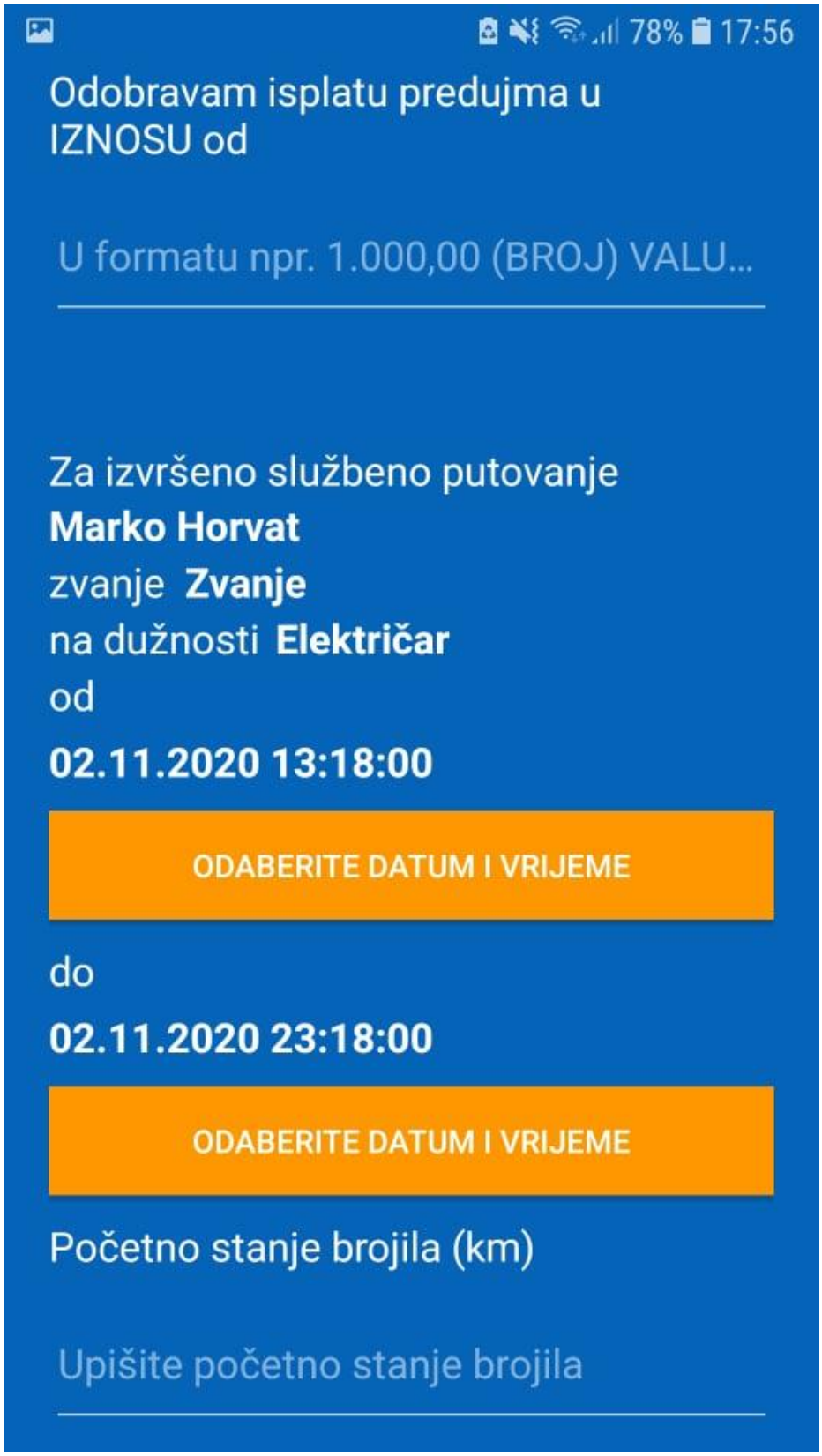
Slika 24. Dizajn Administrator početni zaslon activity na mobitelu (izvor: Autor)

Activityjem „Novi putni nalog“ korisniku je omogućeno popunjavanje svih detalja oko putnog naloga. Neke su stvari unesene automatski, ovisno o kojem prijavljenom korisniku je riječ, dok neki detalji nisu uneseni zbog neposjedovanja prava administratora. Dobivena je validacija da polja unosa ne mogu biti prazna prije nego što je nalog poslan na server. Omogućeno je računanje satnice ovisno o odabranim danima. Algoritam je takav da ukoliko je manje od 8 sati rada dobivena je satnica od 0 kuna, za rad između 8 i 12 sati dobivena je satnica od 85 kuna i to je poludnevnica, za rad između 12 i 24 sati dobivena je satnica od 175 kuna i to je dnevnica. Više od 24 sata gledano je na drugačiji način: jedna

poludnevnica iznosi 12 sati (npr. 36 sati je jedna dnevnica i jedna poludnevnica itd.). Naposljetku iznad gumba „pošalji putni nalog“ prikazana je opcija „pregledaj putni nalog“ u kojem su uneseni podaci iz activityja i prikazani u klasičnom prikazu, kao da je riječ o fizičkom papiru.



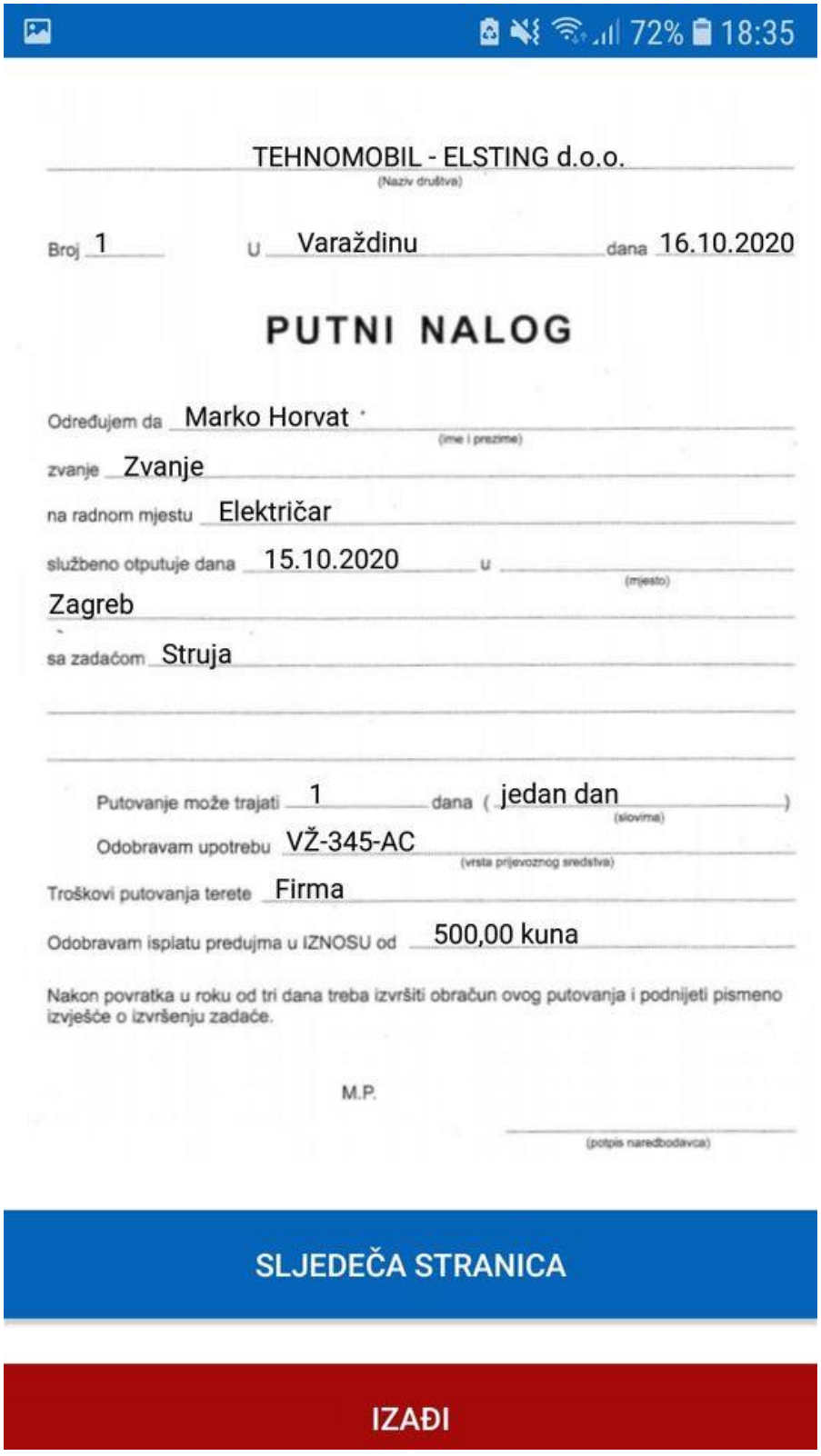
Slika 25. Dizajn „Novi putni nalog“ activity na mobitelu – podebljana slova automatski su unesena - 1.dio (izvor: Autor)



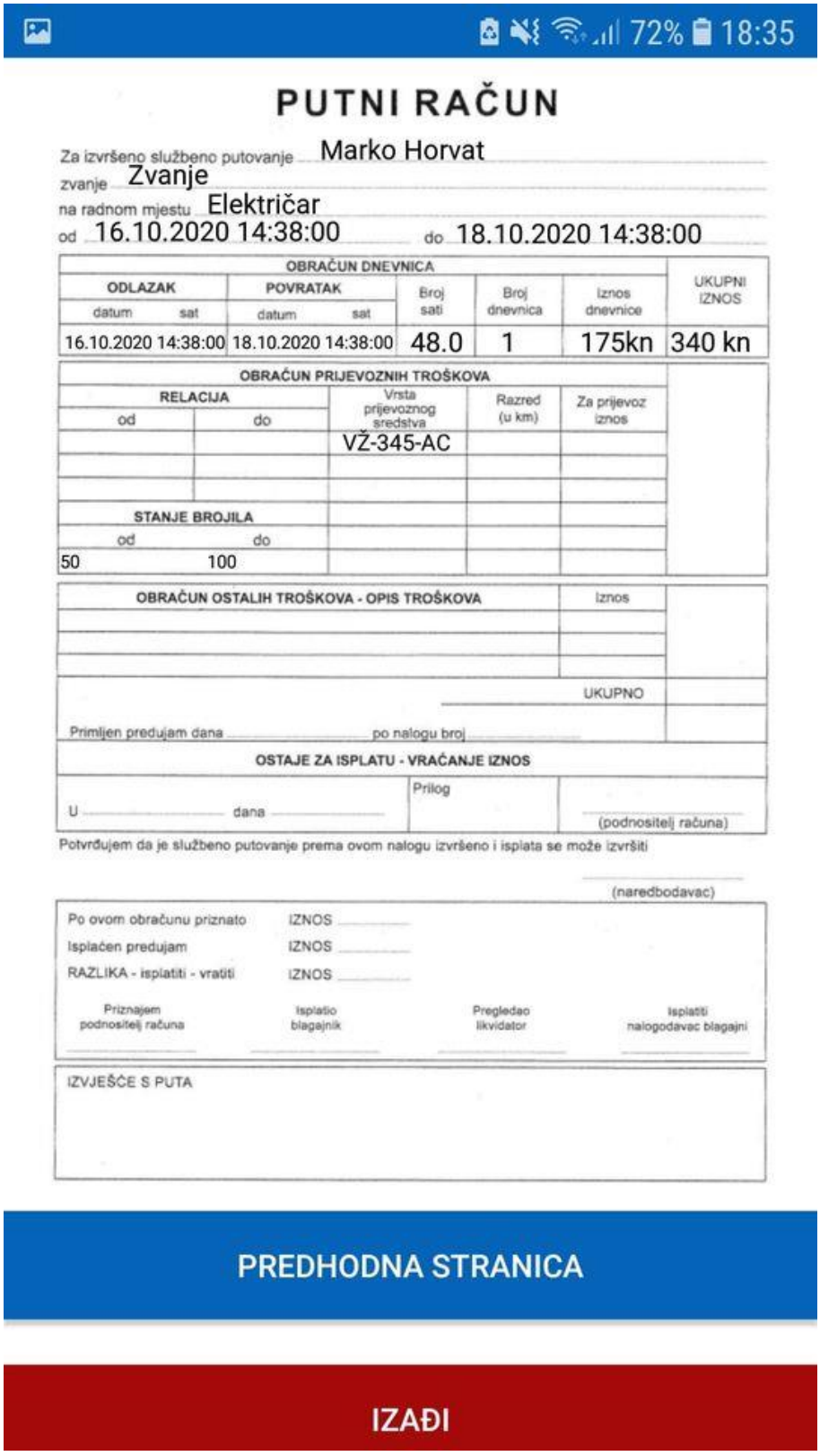
Slika 26. Dizajn „Novi putni naloga“ activity na mobitelu – 2.dio (izvor: Autor)

Slika 27. Dizajn „Novi putni nalog“ activity na mobitelu – 3.dio (izvor: Autor)

Slika 28. Dizajn „Novi putni nalog“ activity na mobitelu – 4.dio (izvor: Autor)



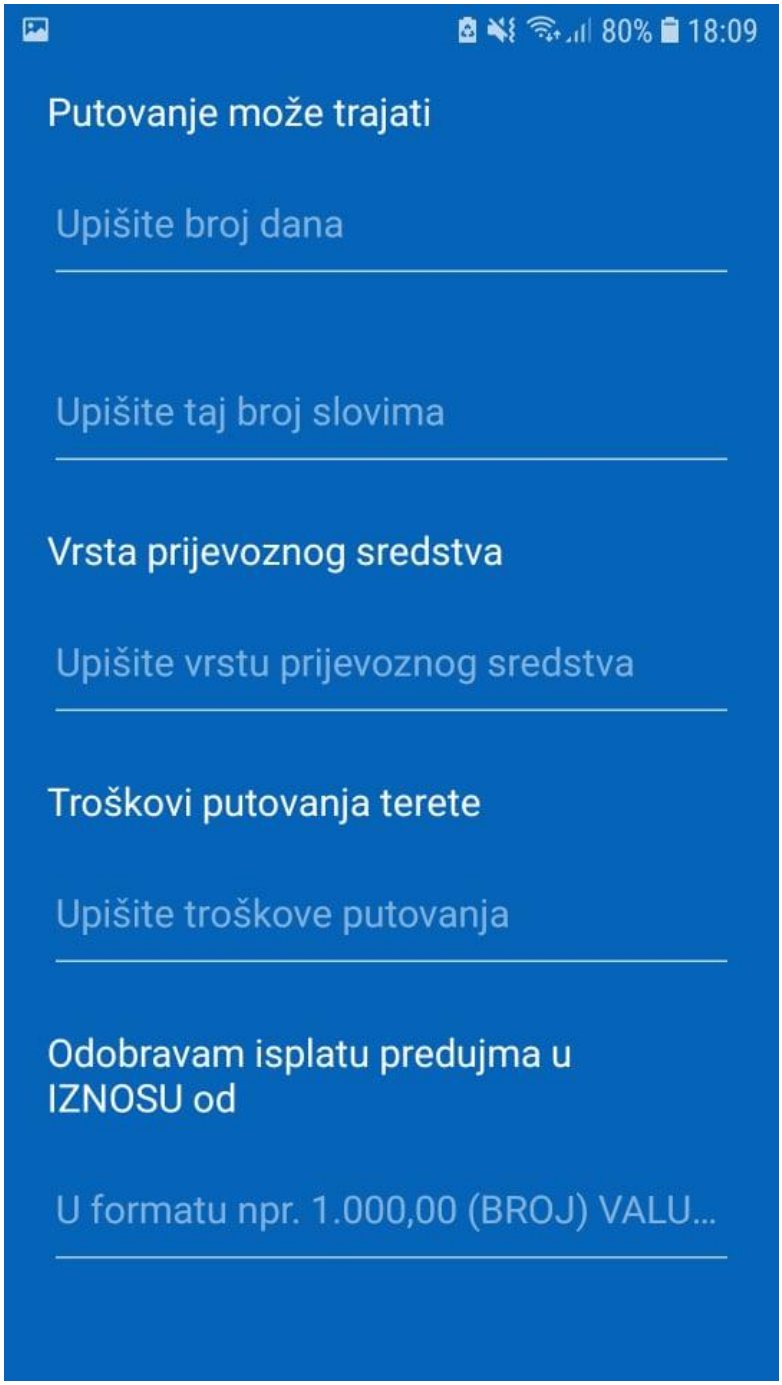
Slika 29. Nakon pritiska na gumb „Pregledaj putni nalog“ na mobitelu stranica jedan (izvor: Autor)



Slika 30. Nakon pritiska na gumb „Pregledaj putni nalog“ na mobitelu stranica dva (izvor: Autor)

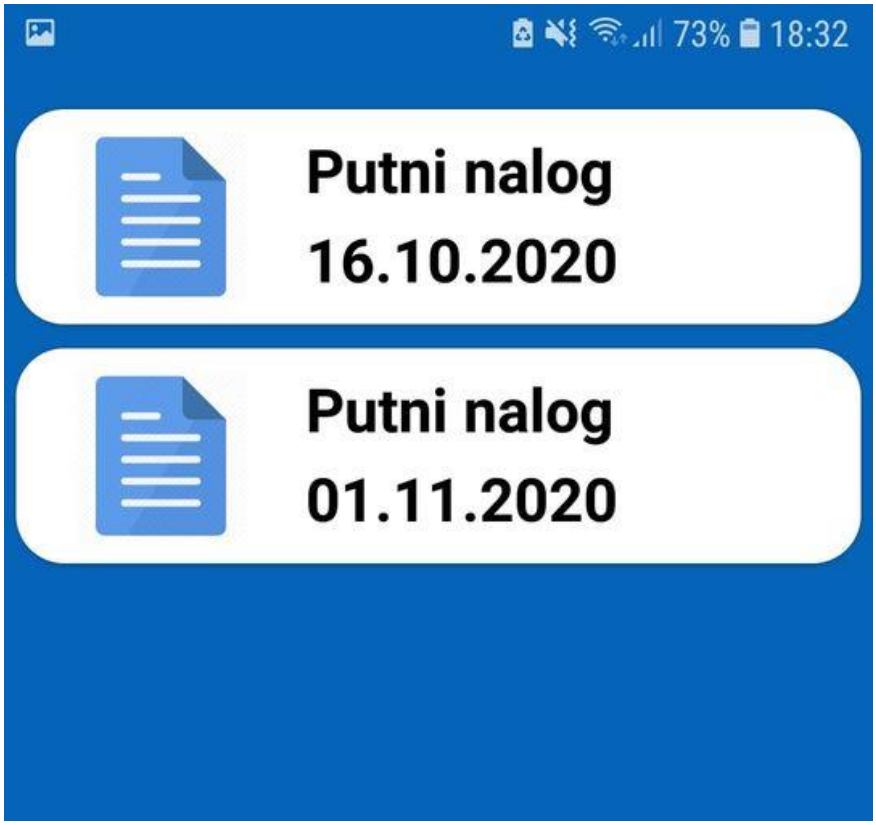
ID	NazivDrustva	auto	broj	brojDnevnica	brojPoludnevnica	brojSati	dana	doDatum	isplataPredujma	mjestoFirme	mjestoF
1	TEHNOMOBIL - ELSTING d.o.o.	AUTO	1	1	2	48	16.10.2020	18.10.2020 14:38:00	500,00 kuna	Varaždinu	Zagreb

Slika 31. Nakon uspješne validacije i slanja putnog naloga od strane korisnika na server u bazi podataka (izvor: Autor)

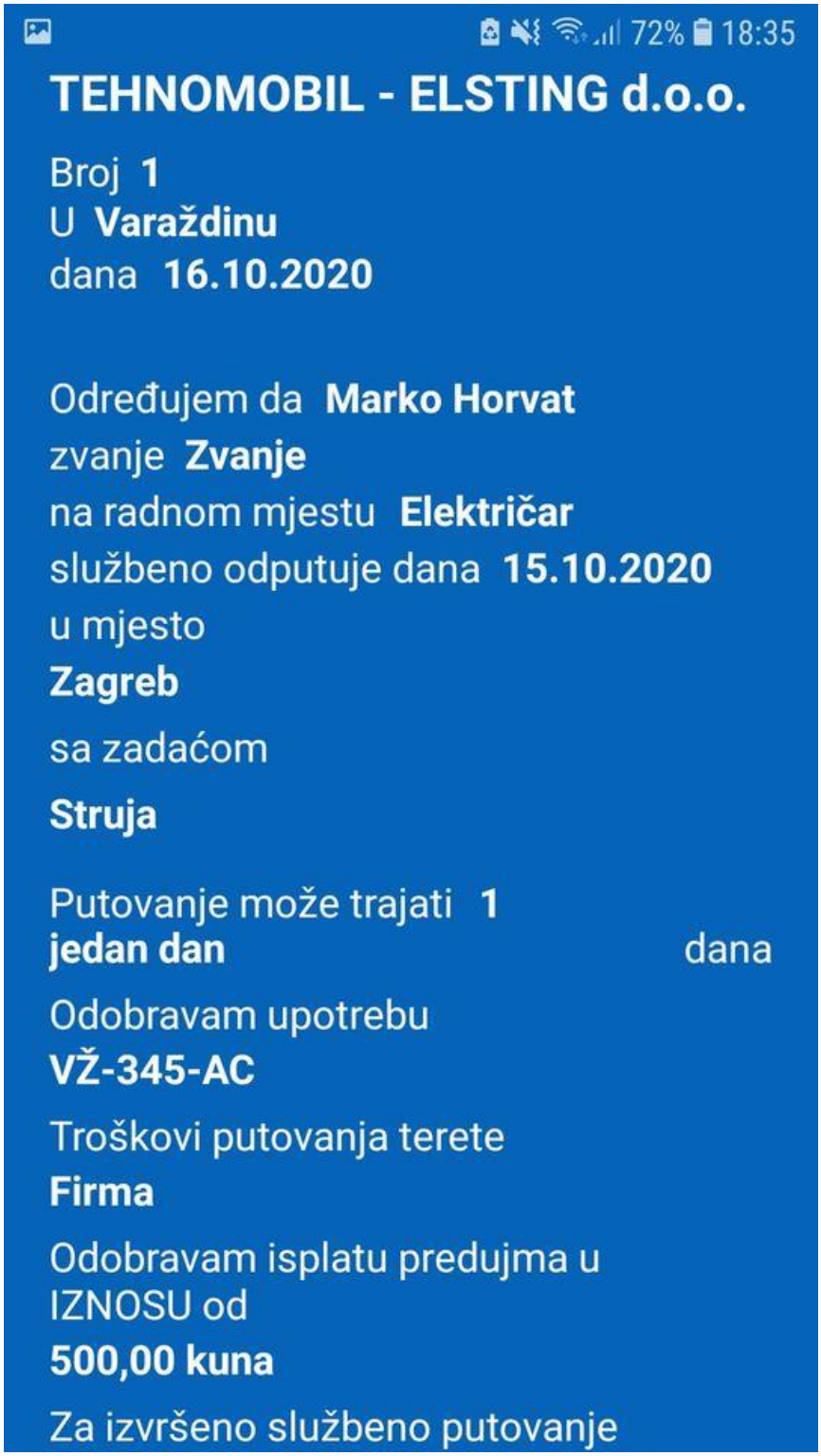


Slika 32. Dodatna opcija „Vrsta prijevoznog sredstva“ za administratora (izvor: Autor)

U nastavku su slike activityja „Moji putni nalozi“ dostupnog Korisniku i Administratoru.



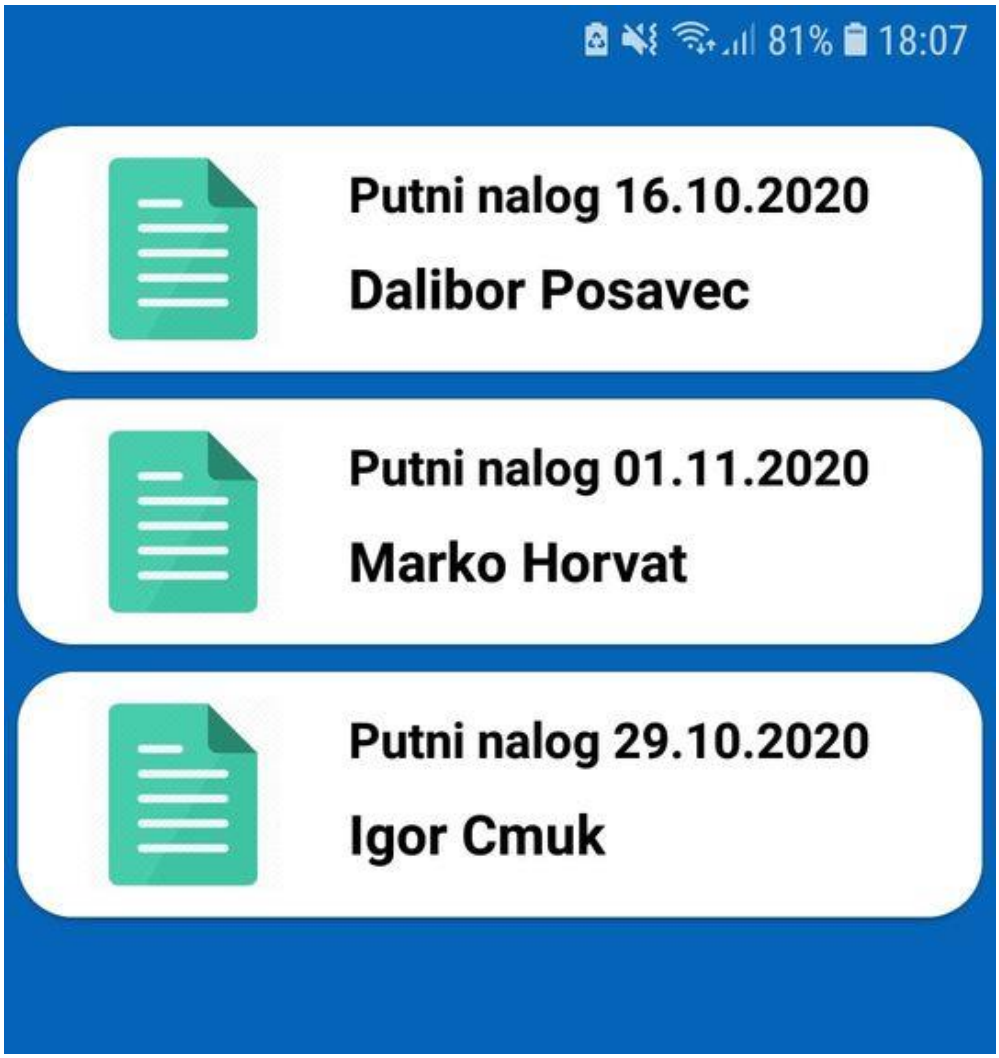
Slika 33. Activity ako su u bazi podataka putni nalozi od prijavljenog korisnika na mobitelu prikazan u obliku liste (izvor: Autor)



Slika 34. Detalji o odabranome putnom nalogu na mobitelu iz liste – 1.dio (izvor: Autor)

Slika 35. Detalji o odabranome putnom nalogu na mobitelu iz liste – 2.dio (izvor: Autor)

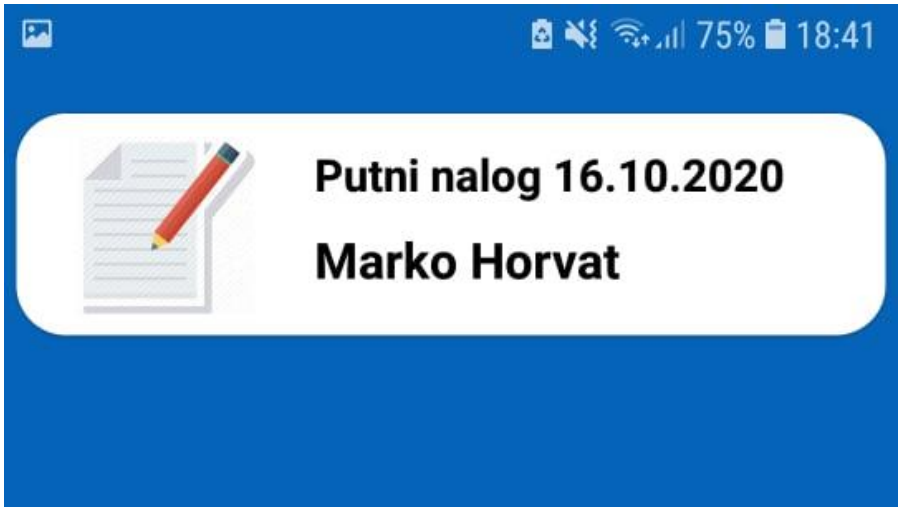
U nastavku su slike activityja „Pogledaj sve putne naloge“, dostupnog samo Administratoru.



Slika 36. Lista putnih naloga u bazi podataka (izvor: Autor)

U nastavku su slike activityja „Pregledaj putne naloge“ preko kojega su kreirani putni nalozi pregledani te potvrđeni ili odbijeni od strane administratora koji je emailom obaviješten o putnim nalozima. Ako je putni nalog odbijen, korisnik je e-mailom o tome obaviješten.

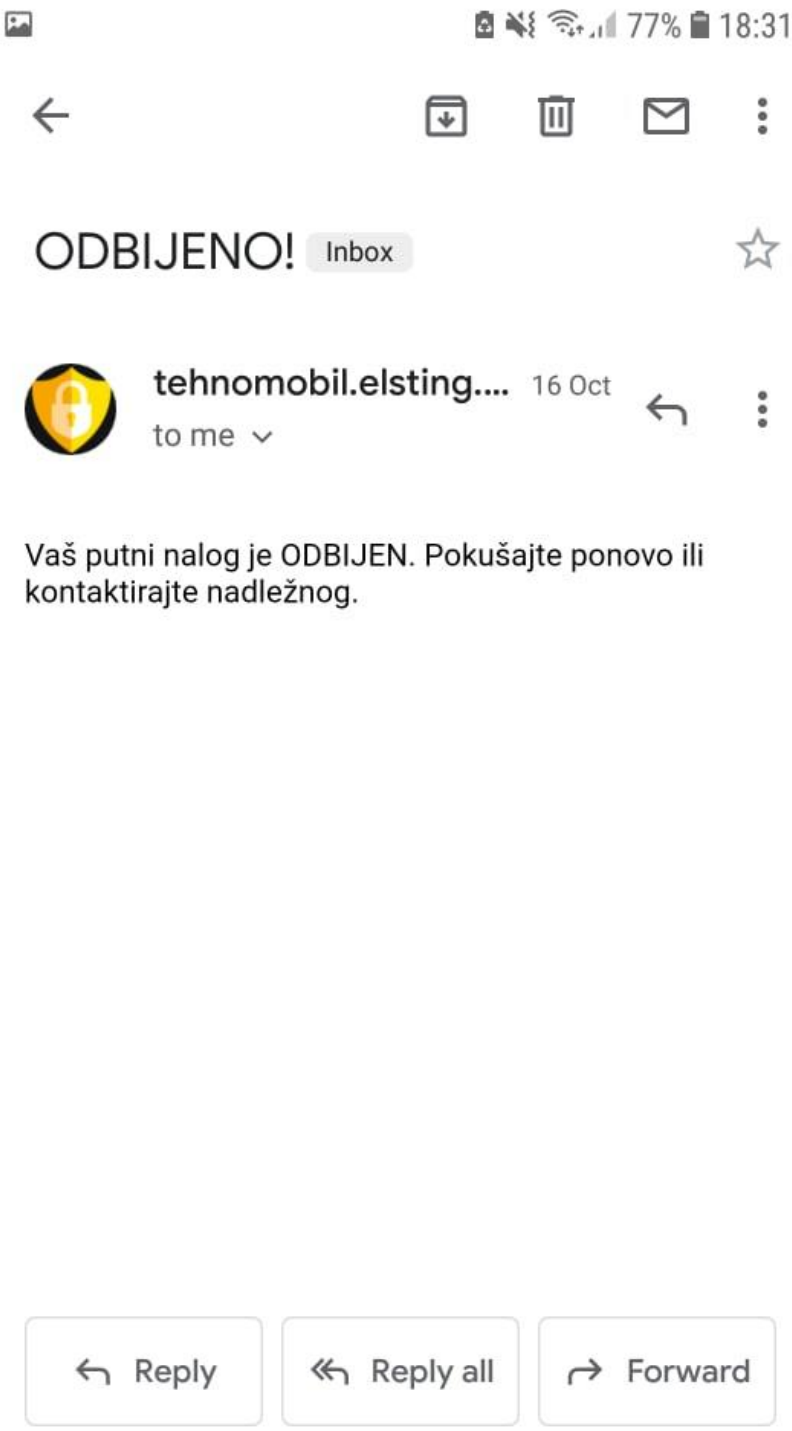
Slika 37. E-mail kojim je administrator obaviješten o postojanju novog putnog naloga za pregled (izvor: Autor)



Slika 38. Lista o postojanju putnog naloga za pregled (izvor: Autor)

Promijenjen je dizajn administratorovim pritiskom na određeni putni nalog za ažuriranje te su dodana još dva gumba „Potvrdi putni nalog“ i „Odbaci putni nalog“.

Slika 39. Dva dodatna gumba nakon administratorova pritiska na provjeru putnog naloga (izvor: Autor)



Slika 40. Administratorovim pritiskom na „odbaci putni nalog“ korisniku je na njegovu e-mail adresu poslan ovakav format e-maila. (izvor: Autor)

5. Zaključak

Efikasnost i dizajn aplikaciju čine jednostavnom za korištenje. Brzina pisanja putnih naloga vidno je smanjena zahvaljujući aplikaciji. Osobni podaci (npr. ime, prezime, zanimanje) neće trebati biti zapisani pa je vrijeme za pisanje novih putnih naloge uveliko smanjeno. Kalkulator satnice precizan je i učinkovit te je novac koji treba biti dobiven za određen broj sati rada izračunat u realnom vremenu. Slanje putnih naloga efikasno je, a cijeli je sustav transparentan (svi korisnici obaviješteni su e-mailom o pregledu i potvrdi putnoga naloga ili o njegovom odbijanju ukoliko prvobitno doneseni kriteriji nisu zadovoljeni). Osim toga, svi su zaposlenici tvrtke na dobitku jer je riječ o pouzdanome sustavu preko kojeg je plaćanje radnika ažurno i učinkovito.

Springboot jednostavan je i moćan alat za kreiranje servera koji ima korisne i unaprijed programirane dependenciese koji lako mogu biti uključeni u ovakav projekt.

Sve potrebne datoteke bivaju automatski generirane pomoću ovoga alata, a početni server odmah konfiguriran bez potrebe za daljnjim konfiguriranjem što je veliki plus za junior, ali i senior programere jer nema trošenja vremena na konfiguraciju, već na kreiranje metoda koje će biti korištene u budućnosti. SQL upiti automatski su kreirani pomoću metoda u Javi kojima se koristi Hibernate, jedan od boljih alata u Springbootu, a na taj su način određeni podaci u bazi filtrirani ili obrisani bez potrebe za kreiranjem dodatnih metoda vezanih za SQL upite. Android aplikacija lako je izrađena pomoću Android Studija jer je ponuđeno mnogo odličnih alata te „Step by step“ debug funkcija koja je odlična zbog laganog korištenja i mogućnosti da u realnom vremenu bude vidljivo sve u vezi aplikacije, ali i kakve su vrijednosti u varijablama.

Cjelokupno istraživanje i sama realizacija ideje za autora rada jedno su veliko informativno iskustvo, a osnovni rad internetskih aplikacija, servera te njihova komunikacija u potpunosti su mu razjašnjeni. Razjašnjen mu je takav način rada koji zapravo koriste i sve velike tvrtke i njihove aplikacije koje su korištene svakodnevno (npr. Facebook, Instagram, WhatsApp, Viber itd.).

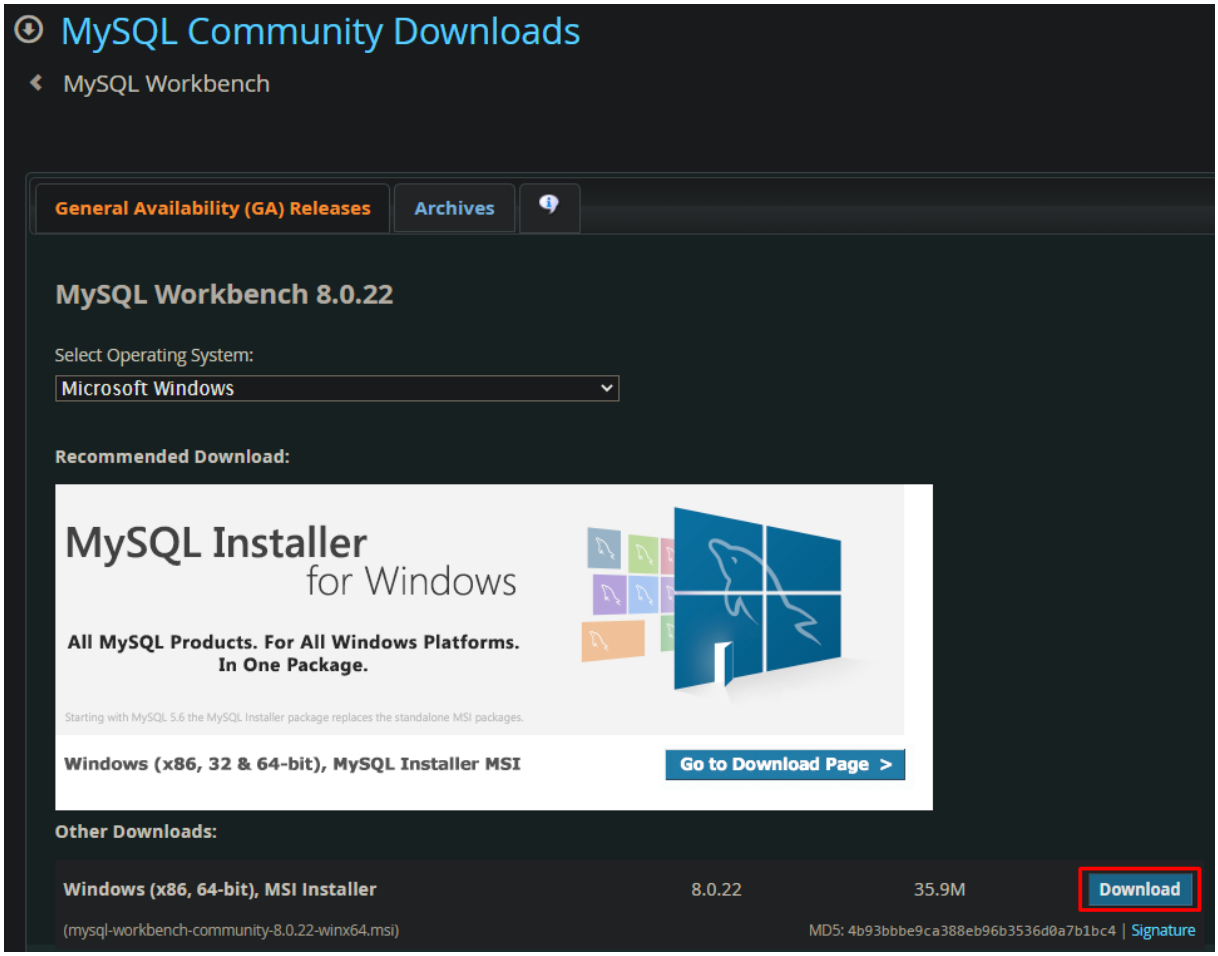
6. Literatura

1. Java, (https://www.w3schools.com/java/java_intro.asp)
2. Java, (https://java.com/en/download/faq/whatis_java.xml)
3. SQL, (https://www.w3schools.com/sql/sql_intro.asp)
4. SQL, (https://www.tutorialspoint.com/sql/sql-overview.htm)
5. Springboot, (https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm)
6. Springboot, (https://dzone.com/articles/understanding-the-basics-of-spring-vs-spring-boot)
7. Spring Data JPA, (https://spring.io/projects/spring-data-jpa)
8. Spring Security, (https://spring.io/projects/spring-security)
9. Hibernate (https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1)
10. MySQL Workbench, (https://www.mysql.com/products/workbench/)
11. IntelliJ IDEA, (https://www.jetbrains.com/idea/)
12. Android Studio, (https://developer.android.com/studio/intro)
13. Spring @Service, (https://stackoverflow.com/questions/15922991/is-spring-annotation-controller-same-as-service)
14. Spring @Controller, (https://www.baeldung.com/spring-controller-vs-restcontroller)
15. Retrofit, (https://square.github.io/retrofit/)
16. JSON, (https://www.json.org/json-en.html)

7. Dodatak

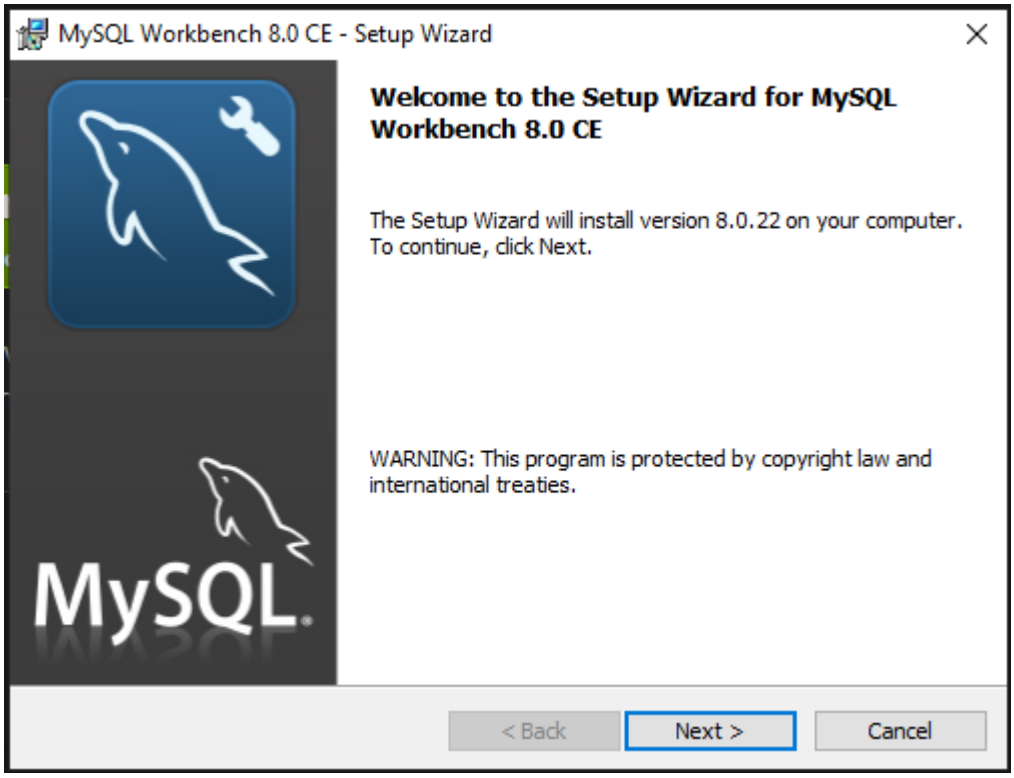
1. Instalacija MySQL Workbench-a

Instalirana je besplatna verzija MySQL Workbench po nazivu MySQL Workbench Community Edition, a s ove poveznice preuzet je alat: https://dev.mysql.com/downloads/workbench/.

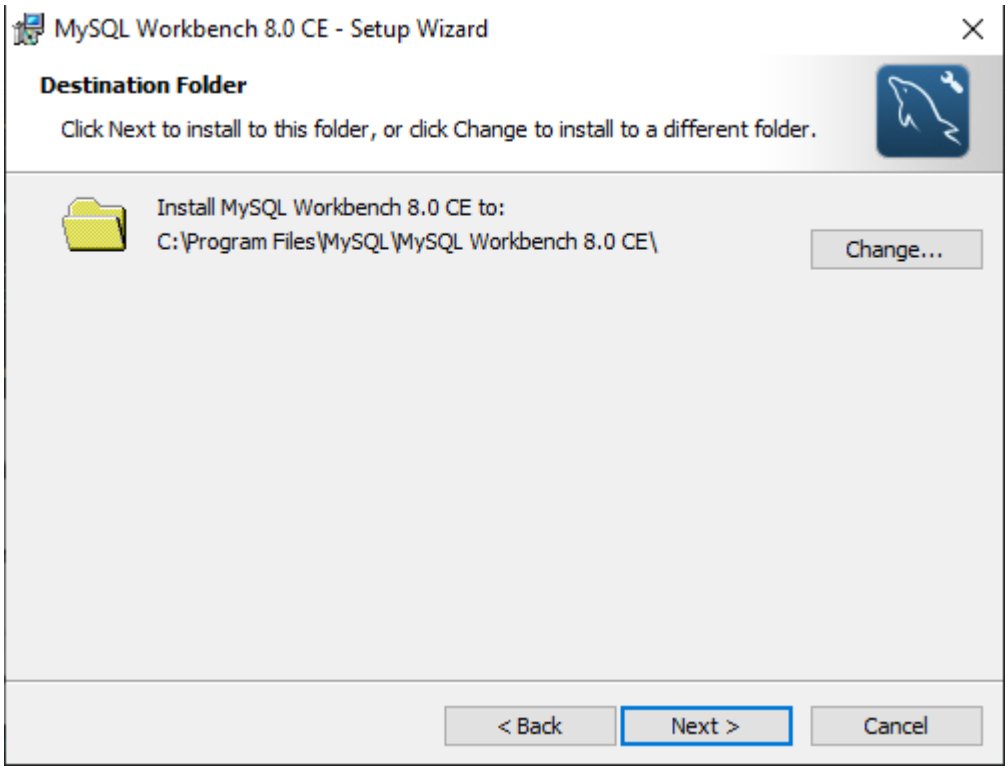


Slika 1. Preuzimanje alata (Izvor: Autor).

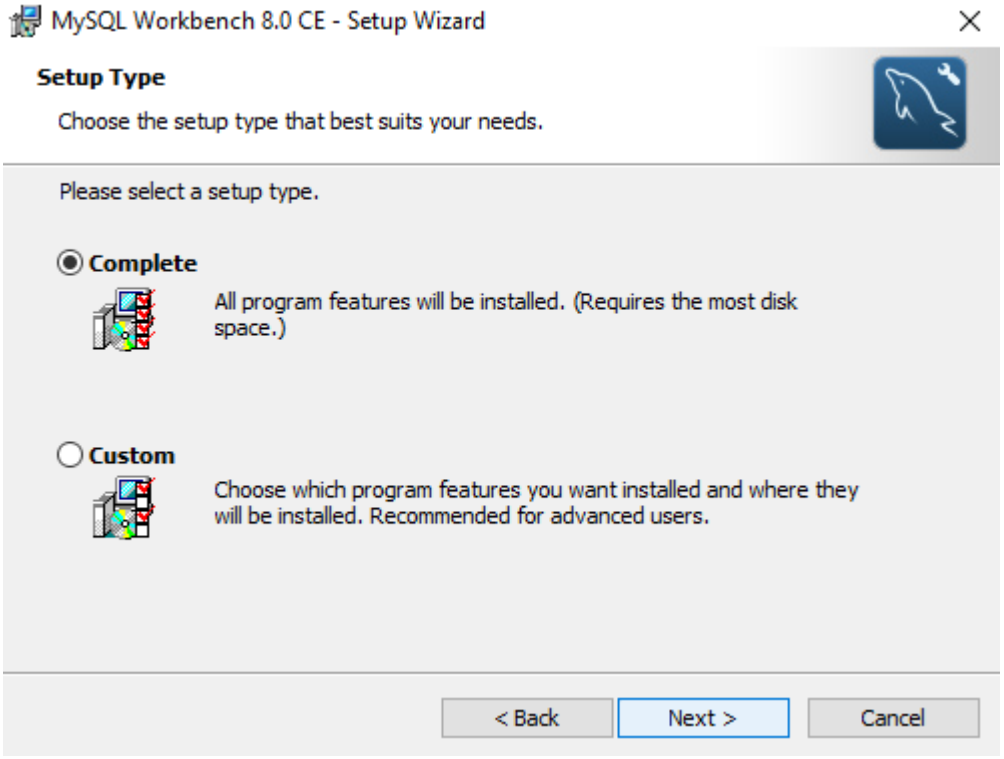
Nakon preuzimanja alata slijedi instalacijski postupak.



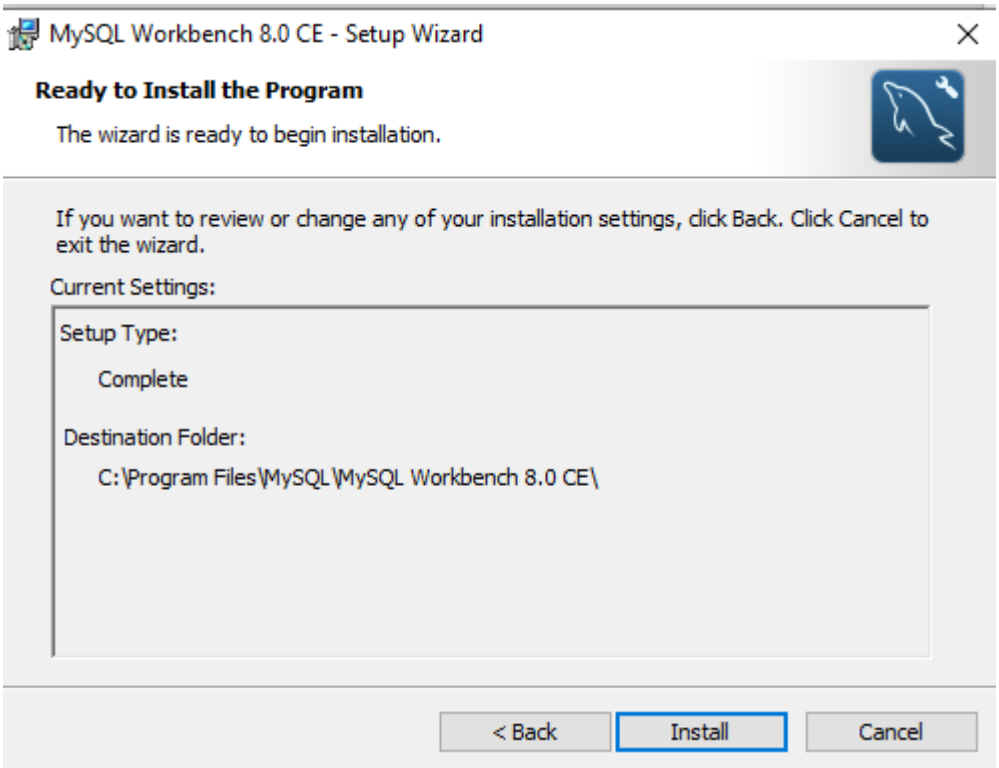
Slika 2. „Setup Wizard“ za instalaciju alata (izvor: Autor)



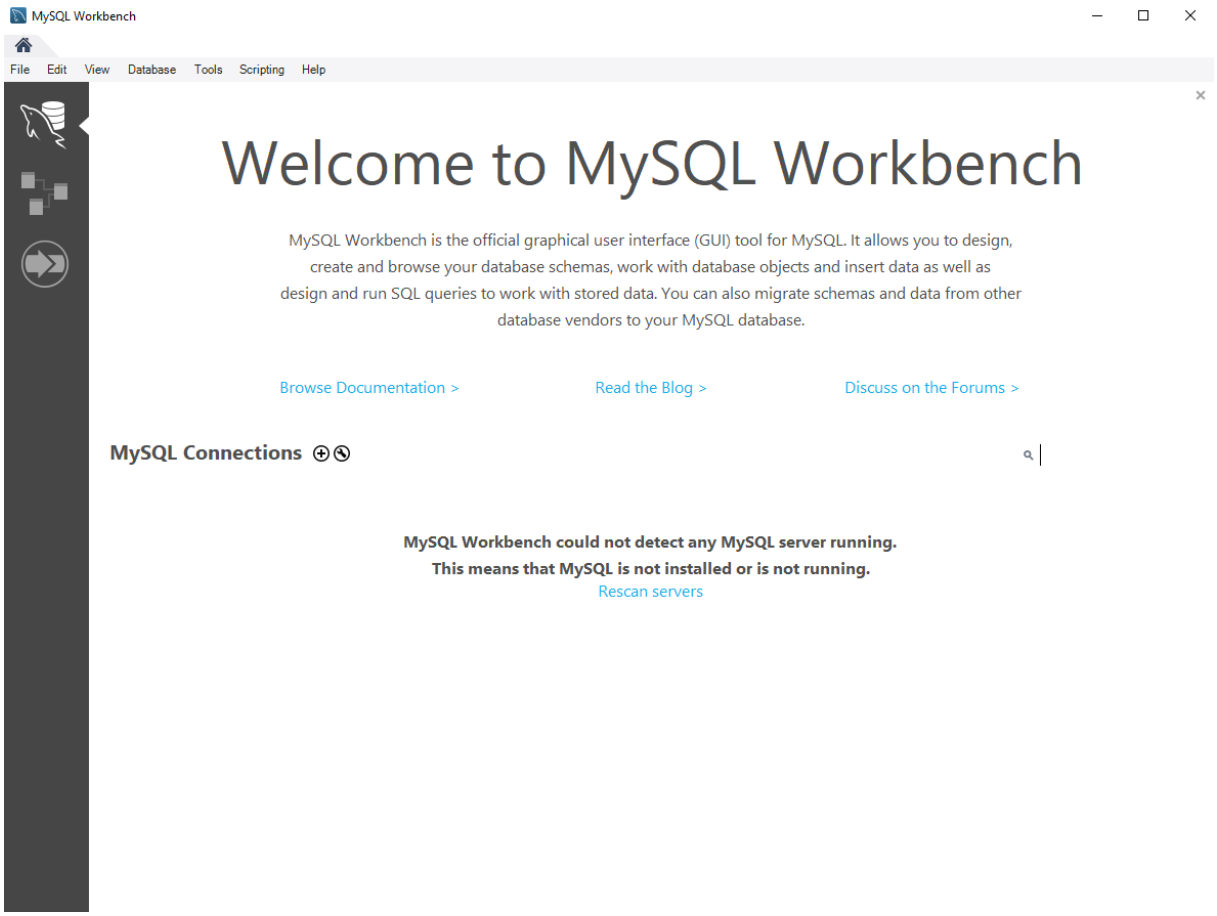
Slika 3. Odabir mjesta na računalu gdje će alat biti instaliran (izvor: Autor)



Slika 4. Odabir „Setup Type“ opcije kojom će prilikom instalacije alata biti instalirane i sve njegove značajke (eng. features) (izvor: Autor)



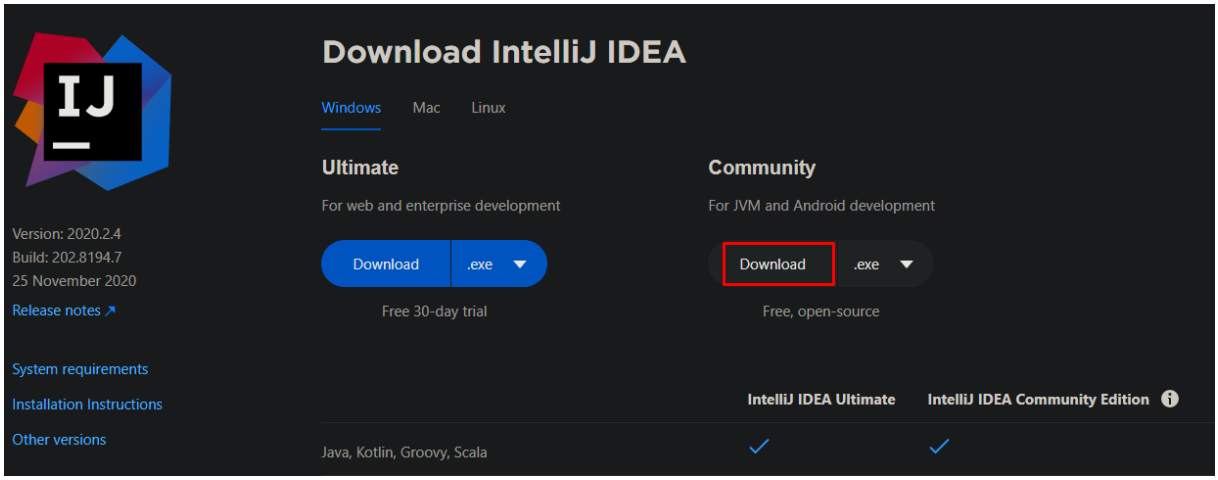
Slika 5. Pritisak na „Install“ gumb te započinjanje instalacije alata (izvor: Autor)



Slika 6. Pokretanje alata nakon uspješne instalacije (izvor: Autor)

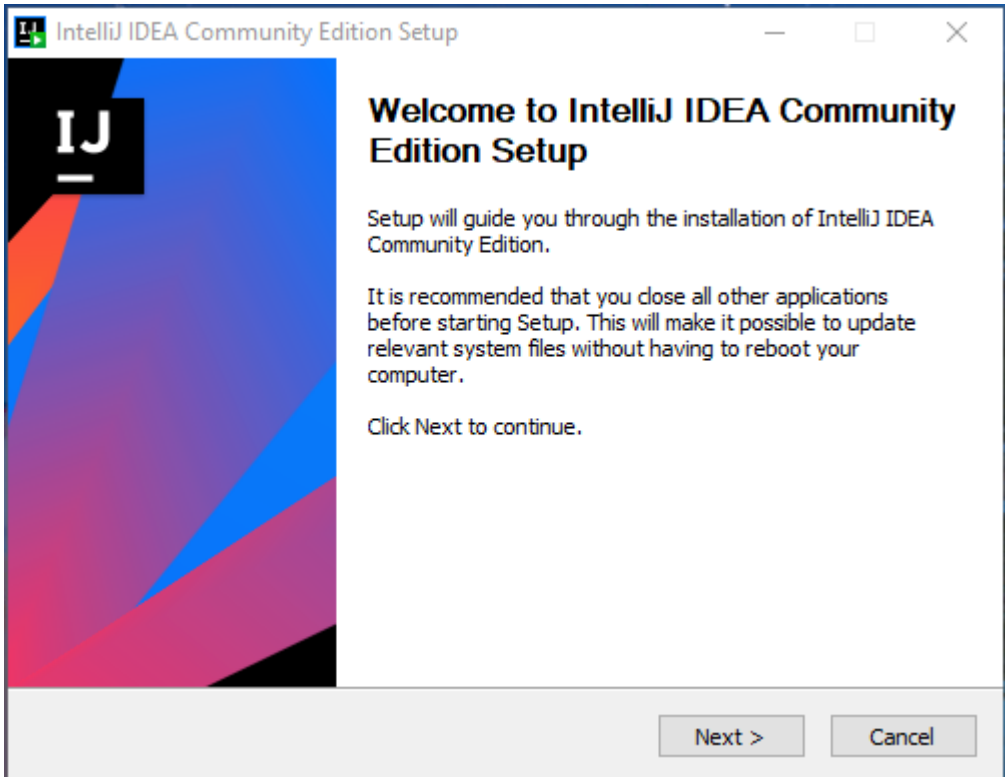
2. Instalacija IntelliJ IDEA

Instalirana je besplatna verzija IntelliJ IDEA po nazivu IntelliJ IDEA Community Edition, a s ove poveznice preuzet je alat: https://www.jetbrains.com/idea/download/#section=windows.

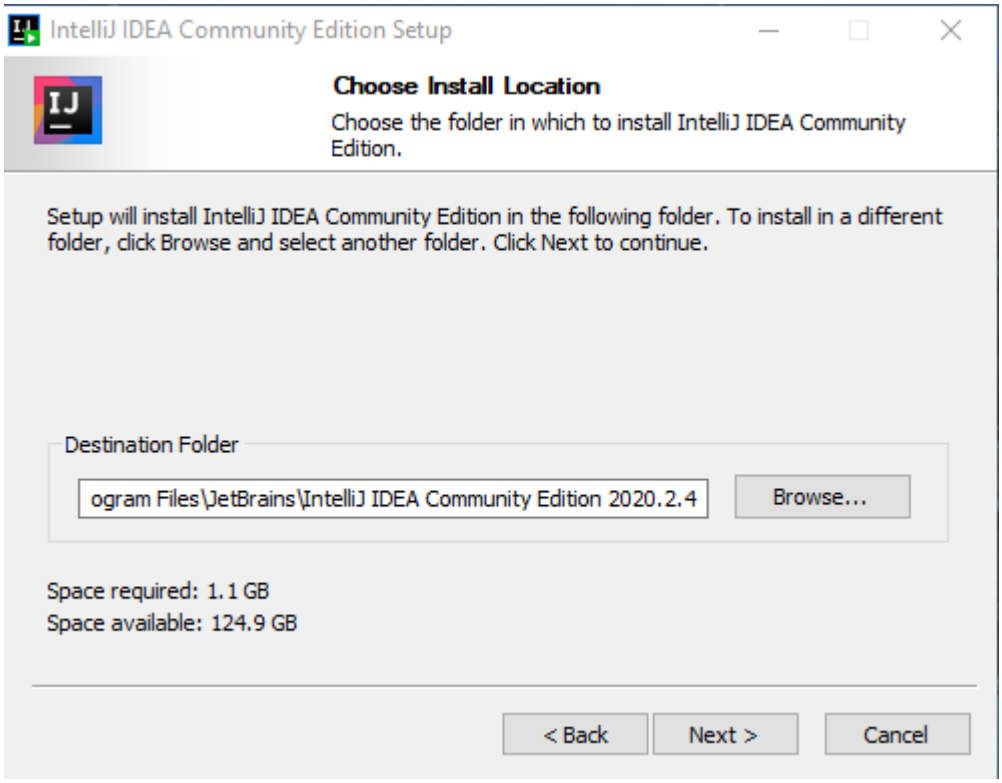


Slika 7. Preuzimanje alata, odabir „Download“ gumba u Community kategoriji (Izvor: Autor).

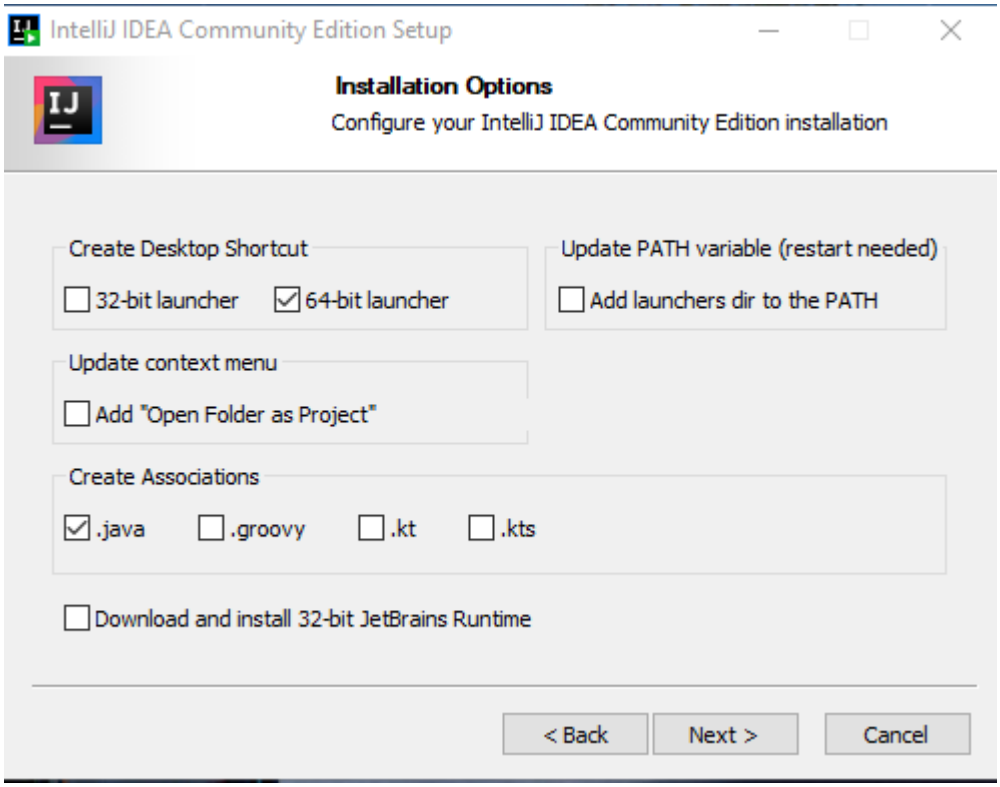
Nakon što je alat preuzet započet je instalacijski postupak.



Slika 8. „Setup Wizard“ za instalaciju alata (izvor: Autor)

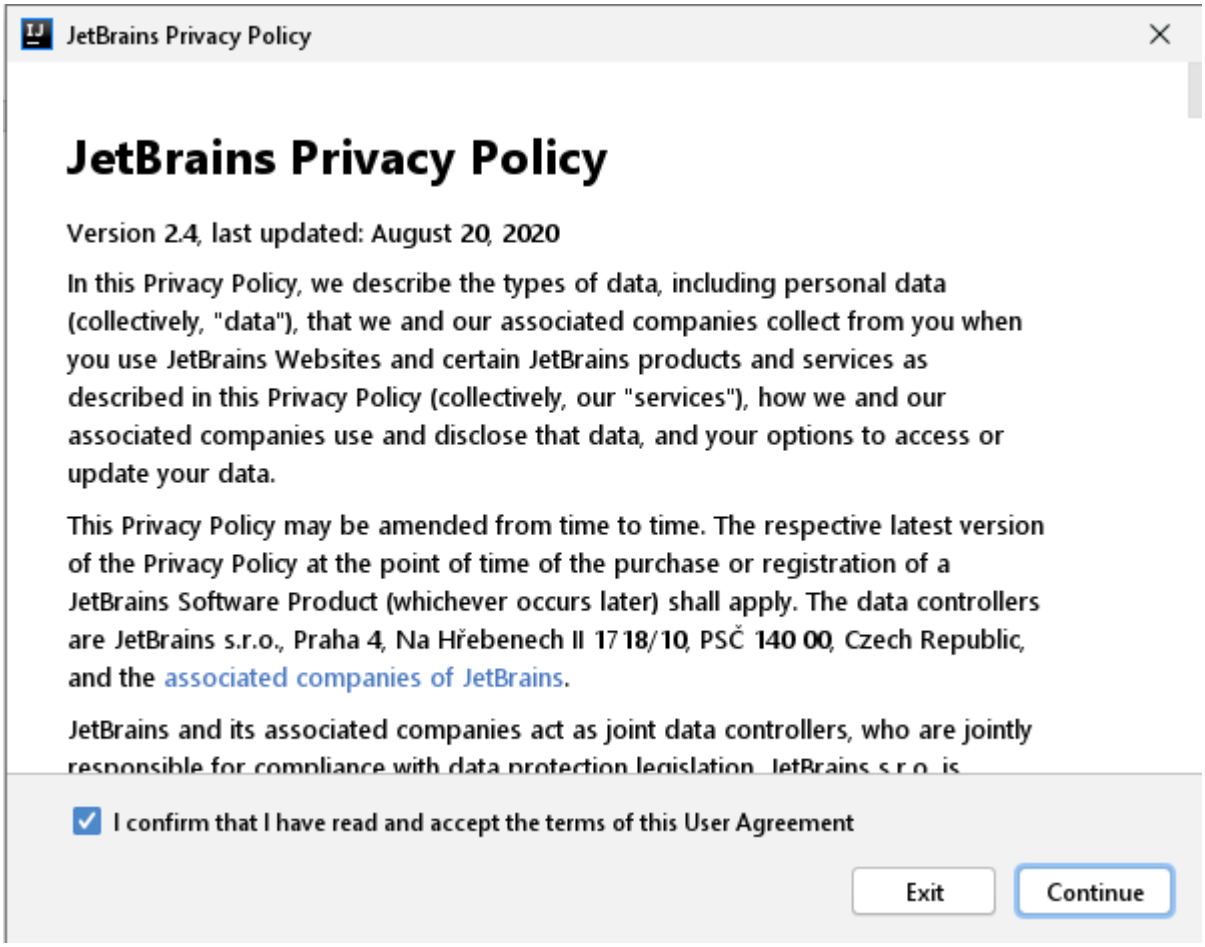


Slika 9. Odabir mjesta na računalu gdje će biti instaliran alat (izvor: Autor)

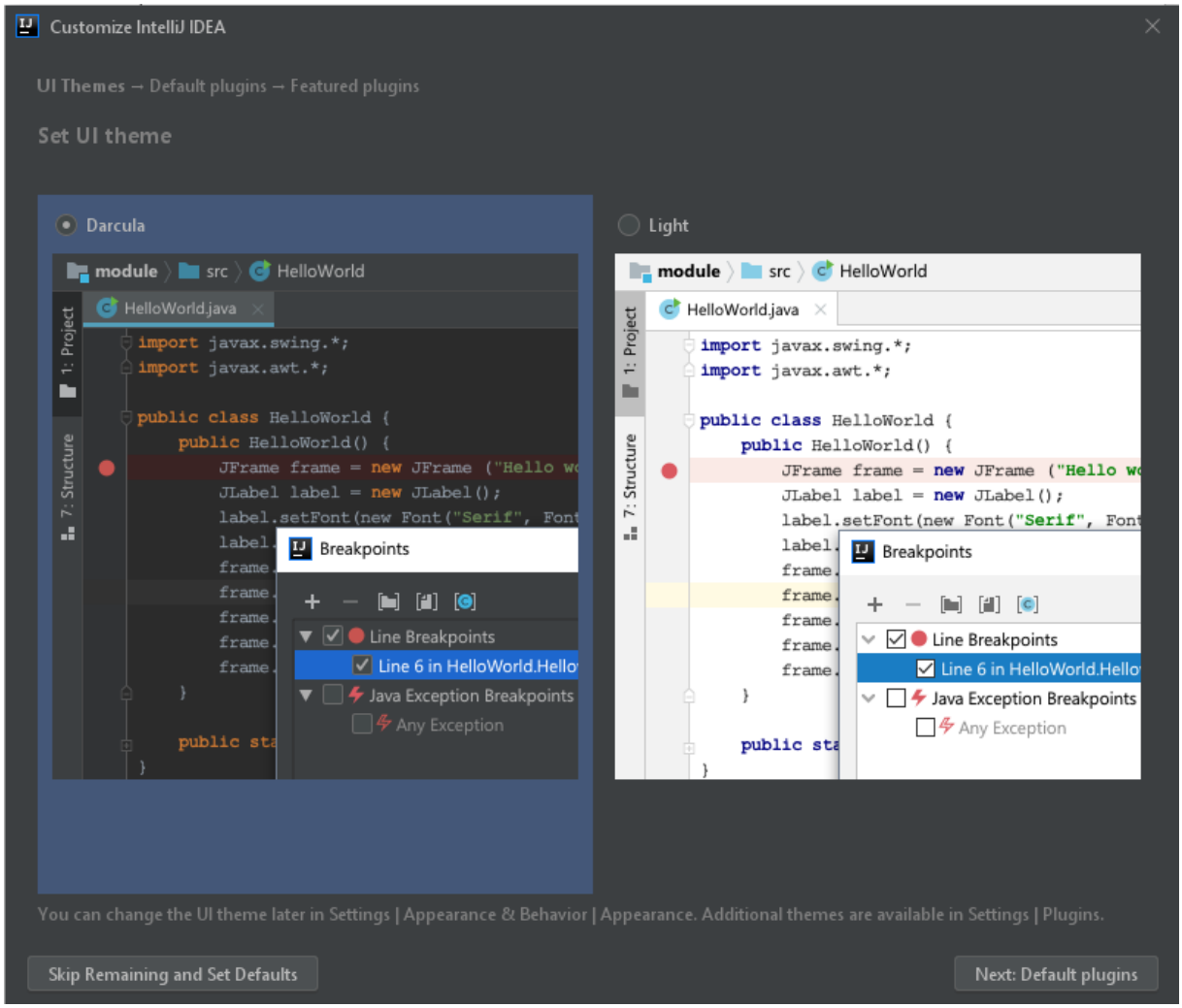


Slika 10. Konfiguriranje instalacije alata (izvor: Autor)

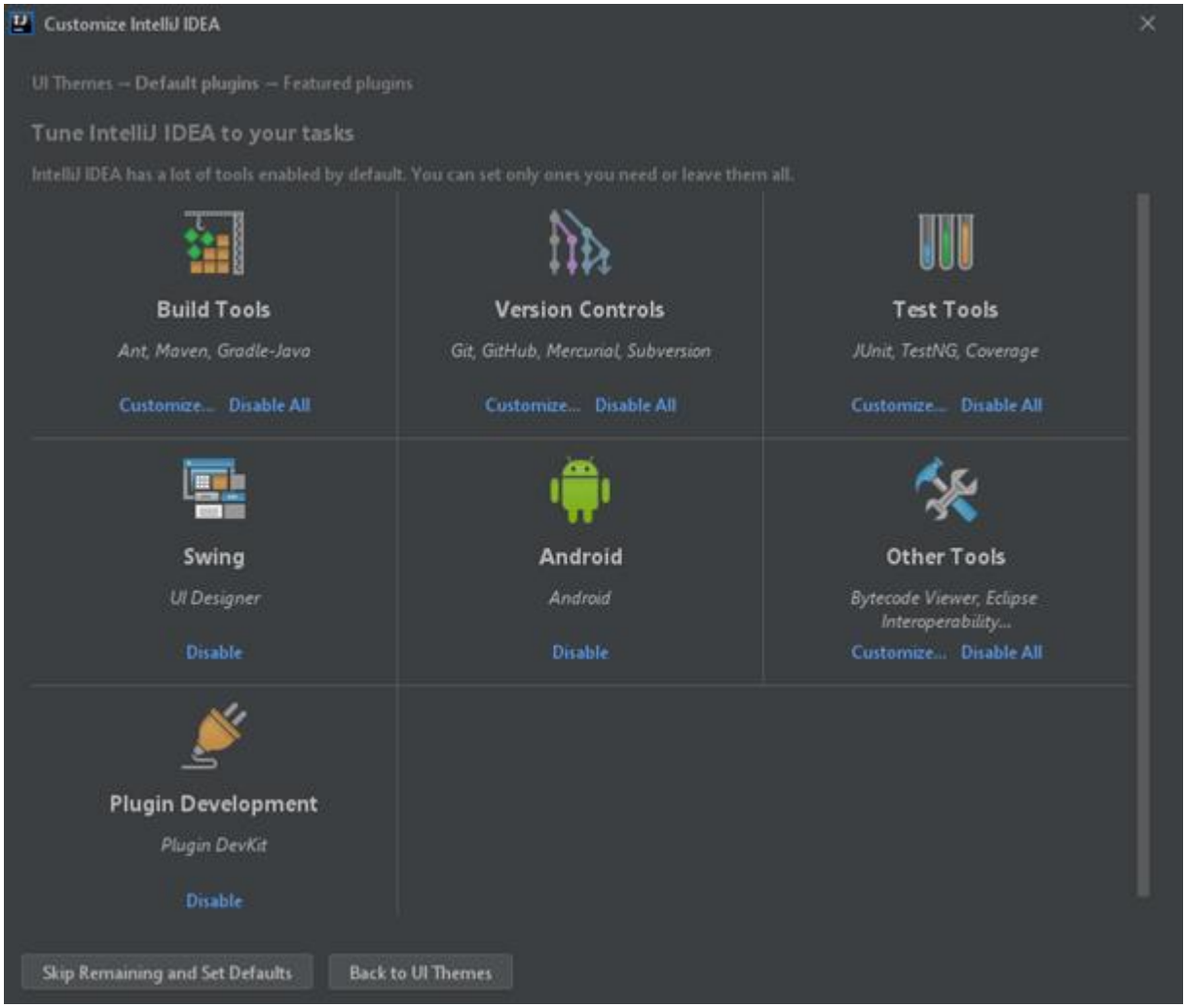
JetBrains politika privatnosti mora biti prihvaćena nakon instalacije alata kako bi on bio korišten.



Slika 11. Prihvaćanje politike privatnosti (izvor: Autor)

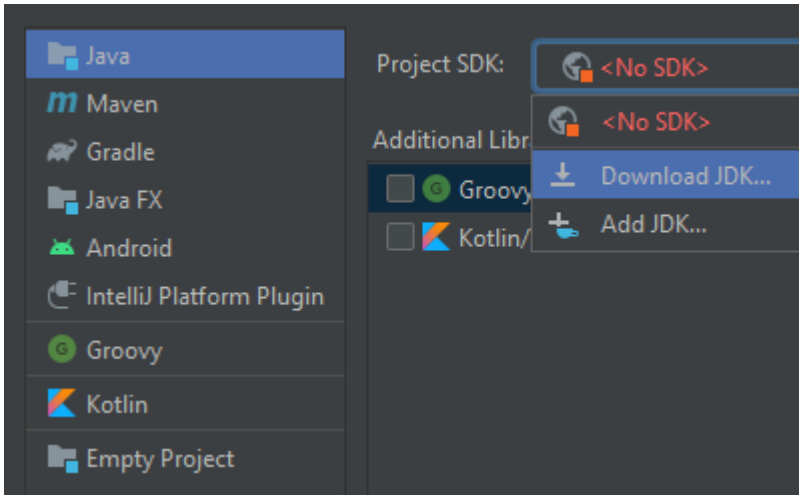


Slika 12. Odabir teme alata (izvor: Autor)

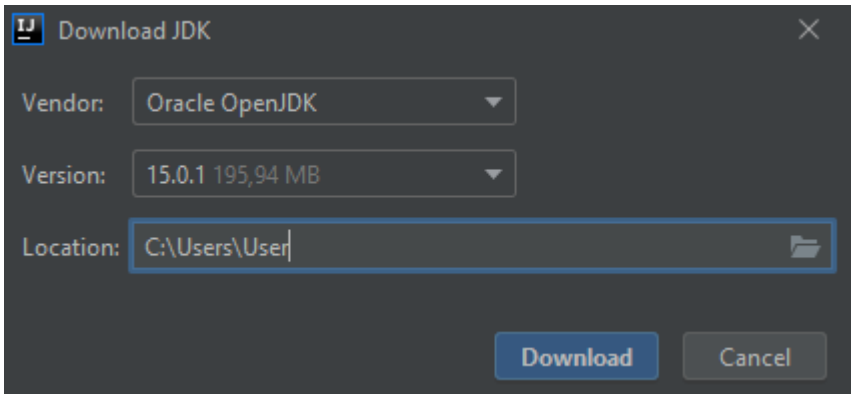


Slika 13. Instalacija dodatnih alata u IntelliJ IDEA (izvor: Autor)

Ovdje mogu biti instalirani dodatni alati. U ovome slučaju sve je odabrano po zadanim postavkama (engl. *Default settings*) i nema potrebe za dodatnim instaliranjem Java JDK verzija mora biti instalirana da bi bilo omogućeno programiranje u Javi. Dobra stvar ovoga alata je ako i nije instaliran nikakav JDK, alat sam preuzme i instalira odabranu JDK verziju.



Slika 14. Početak preuzimanja JDK verzije u alatu (izvor: Autor)

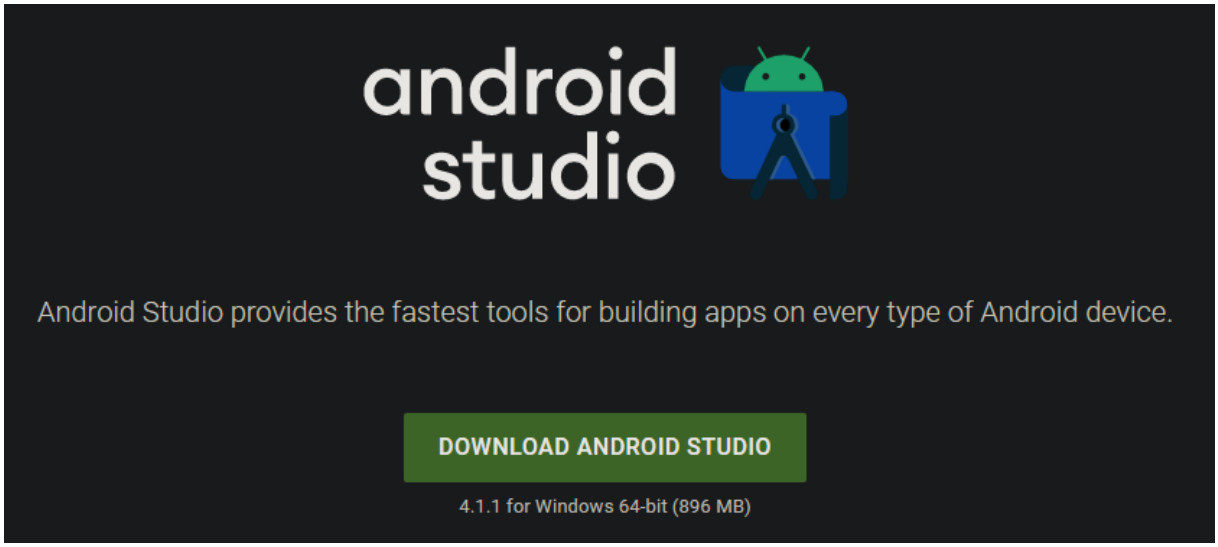


Slika 15. Odabir željene JDK verziju za preuzimanje i mjesta preuzimanja (izvor: Autor)

Slika 16. Pokretanje alata nakon uspješne instalacije (izvor: Autor)

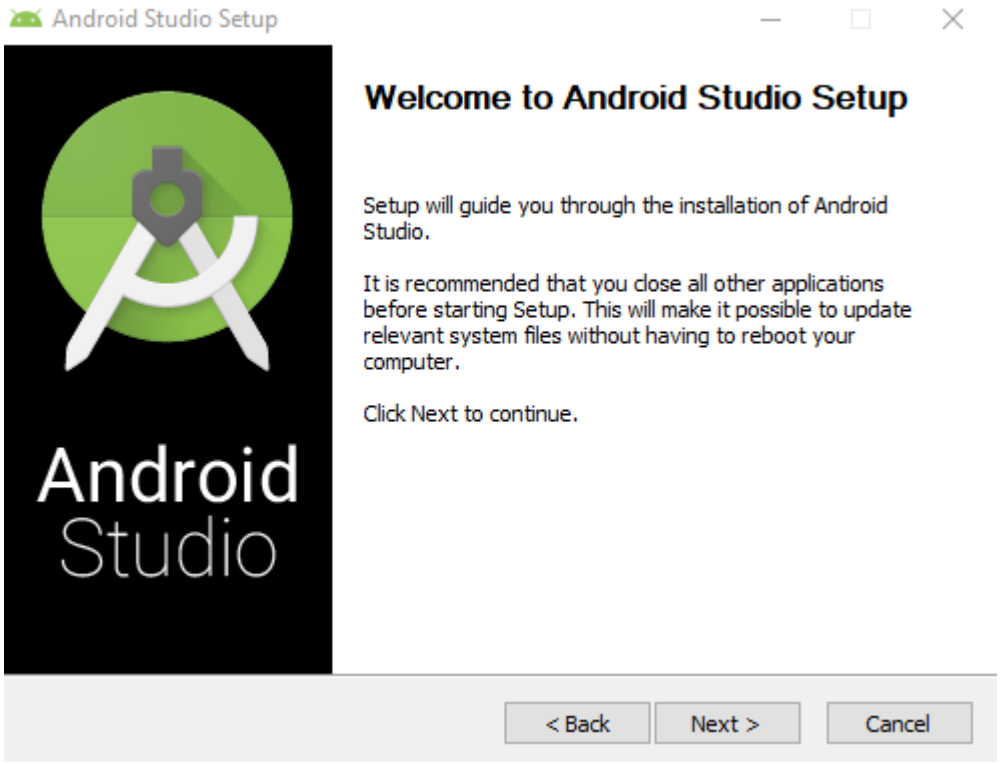
3. Instalacija Android studio-a

Alat je preuzet s ove poveznice: https://developer.android.com/studio/.

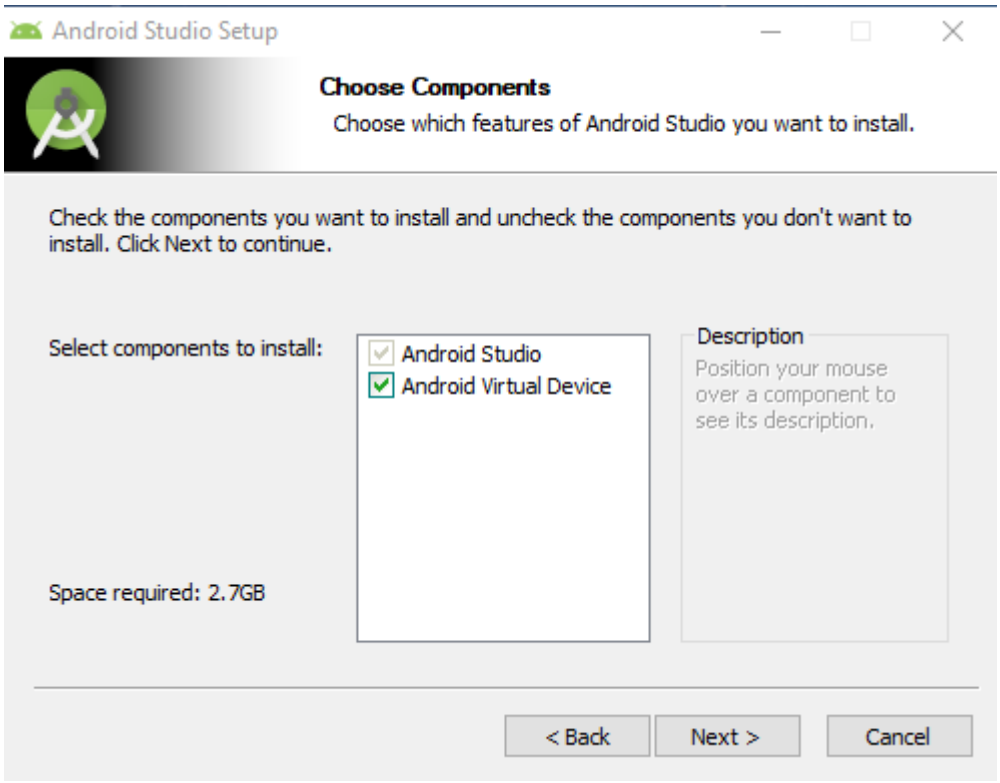


Slika 17. Preuzimanje alata (Izvor: Autor).

Nakon što je alat preuzet započeta je njegova instalacija.

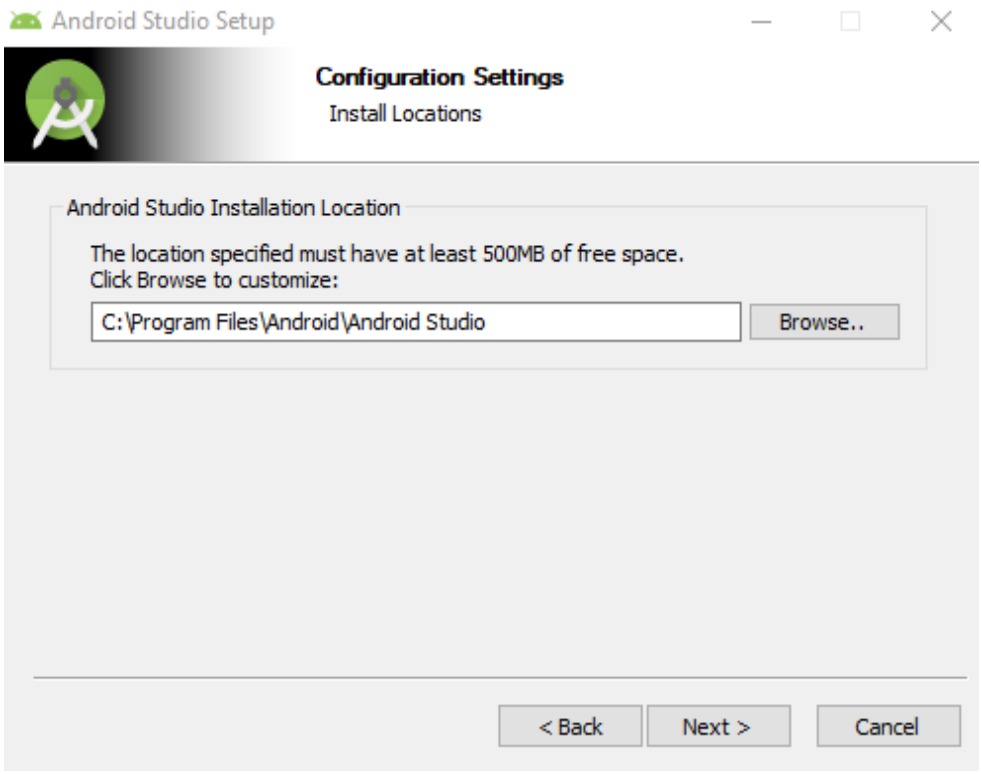


Slika 18. „Setup Wizard“ za instalaciju alata (izvor: Autor)

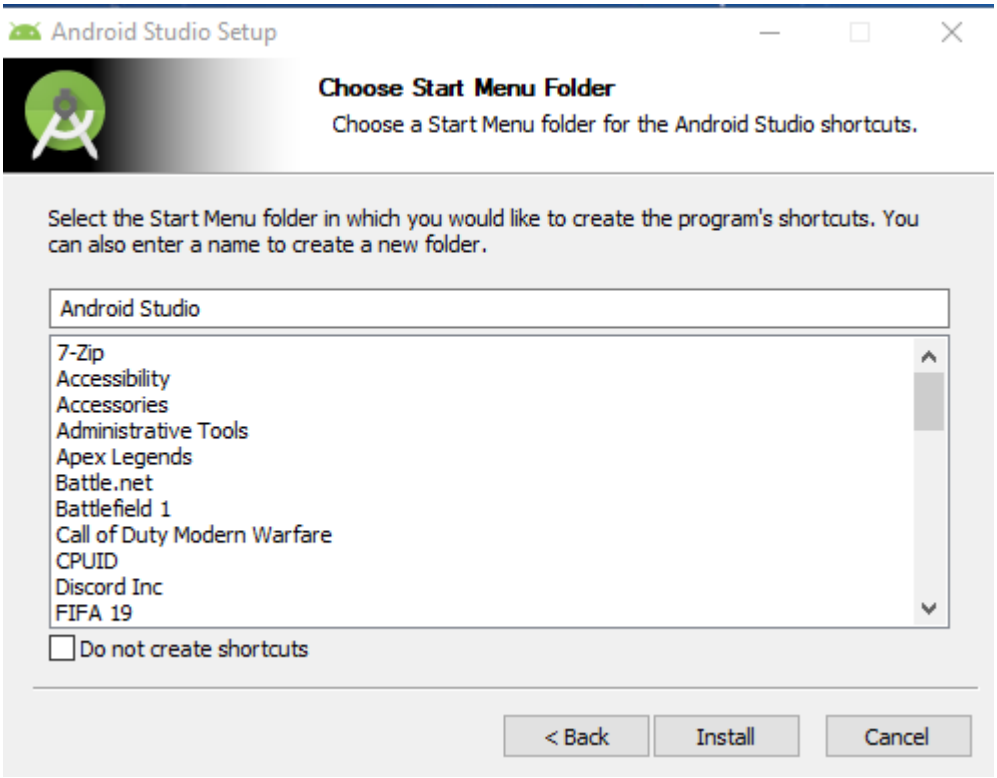


Slika 19. Odabir komponente „Android Virtual Device“ (izvor: Autor)

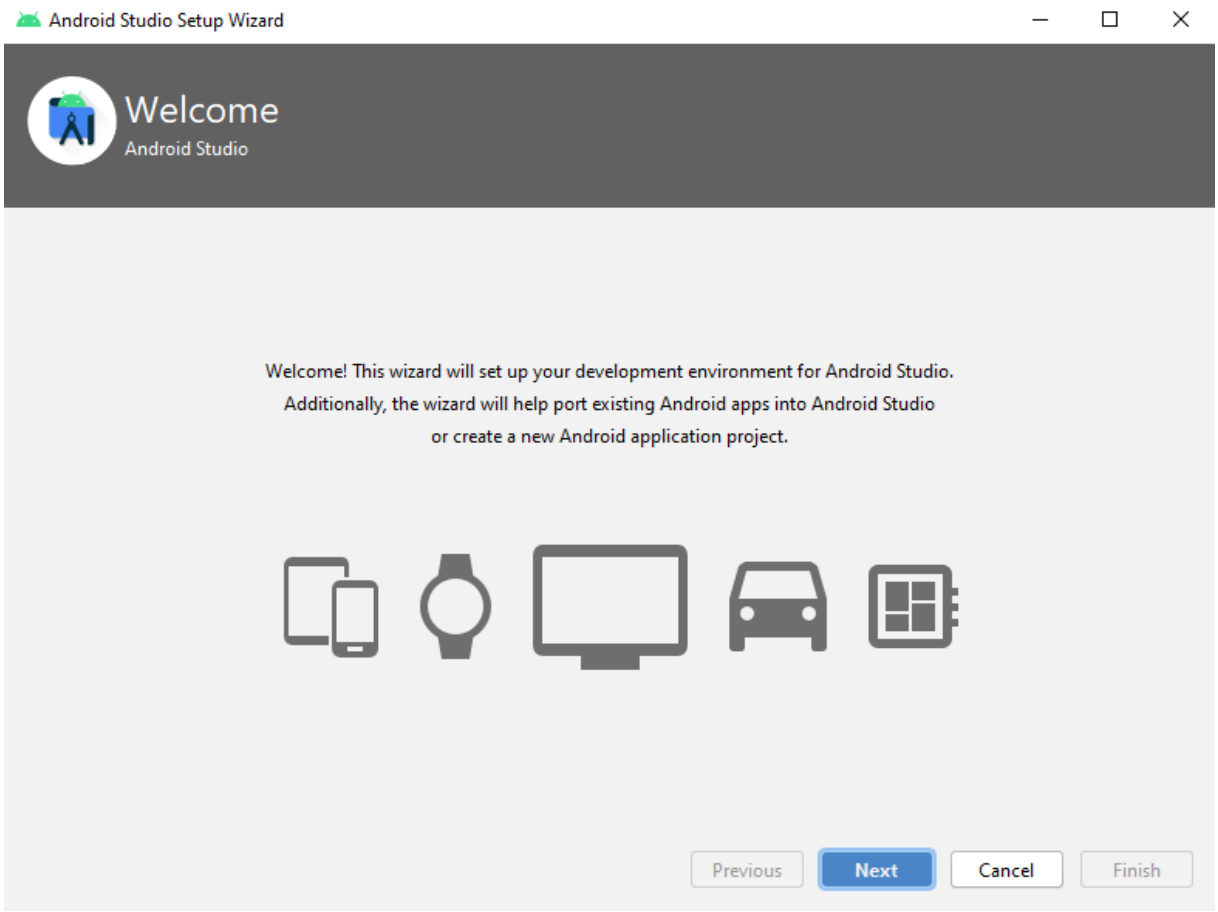
Pomoću ove značajke (eng. features) bez korištenja fizičkog uređaja, a s postojanjem želja za testiranjem aplikacije, emulator biva pokrenut pomoću Android studija i preko njega Android uređaj emuliran s instaliranom aplikaciju koja je u izradi. Time je programeru omogućen vizualni doživljaj rada aplikacije na uređaju.



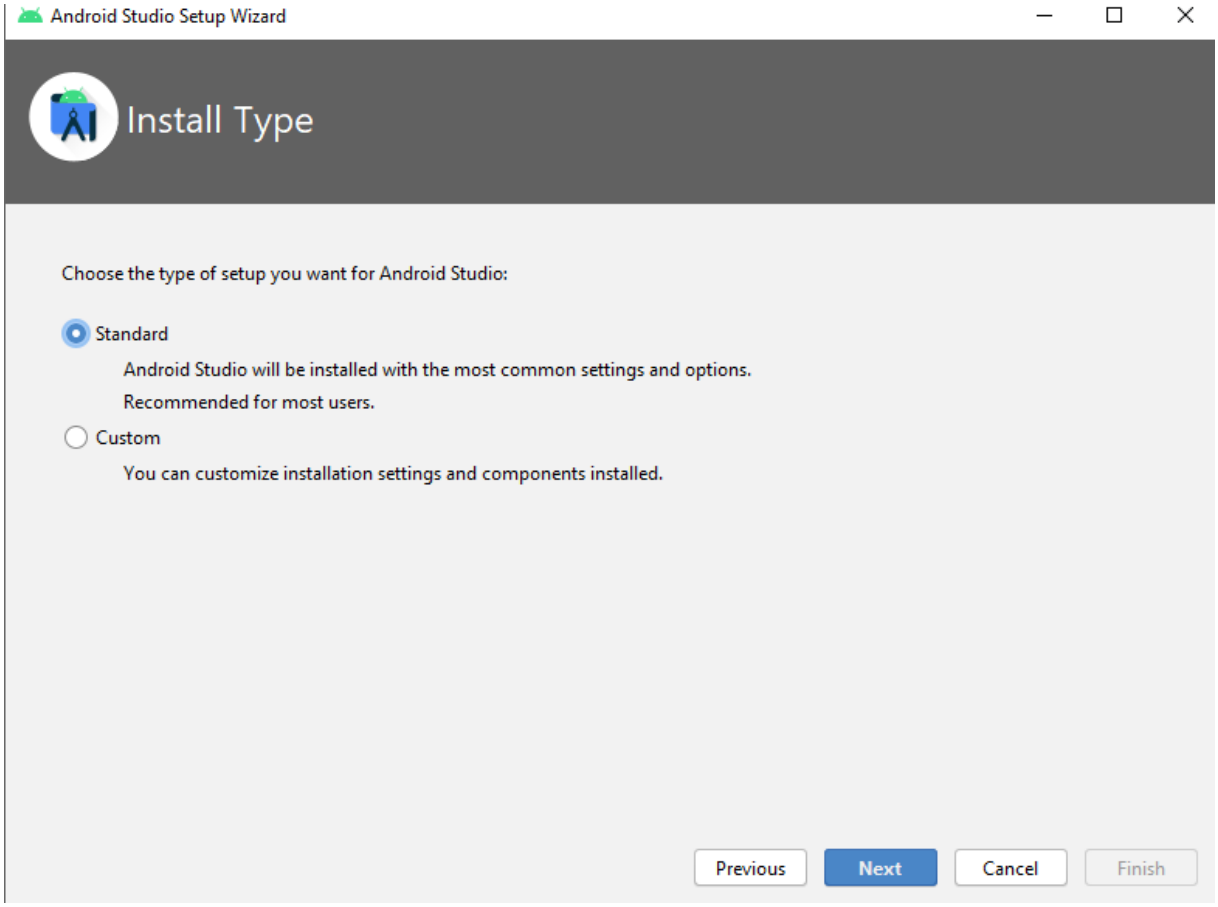
Slika 20. Odabir mjesta na računalu gdje će alat biti instaliran (izvor: Autor)



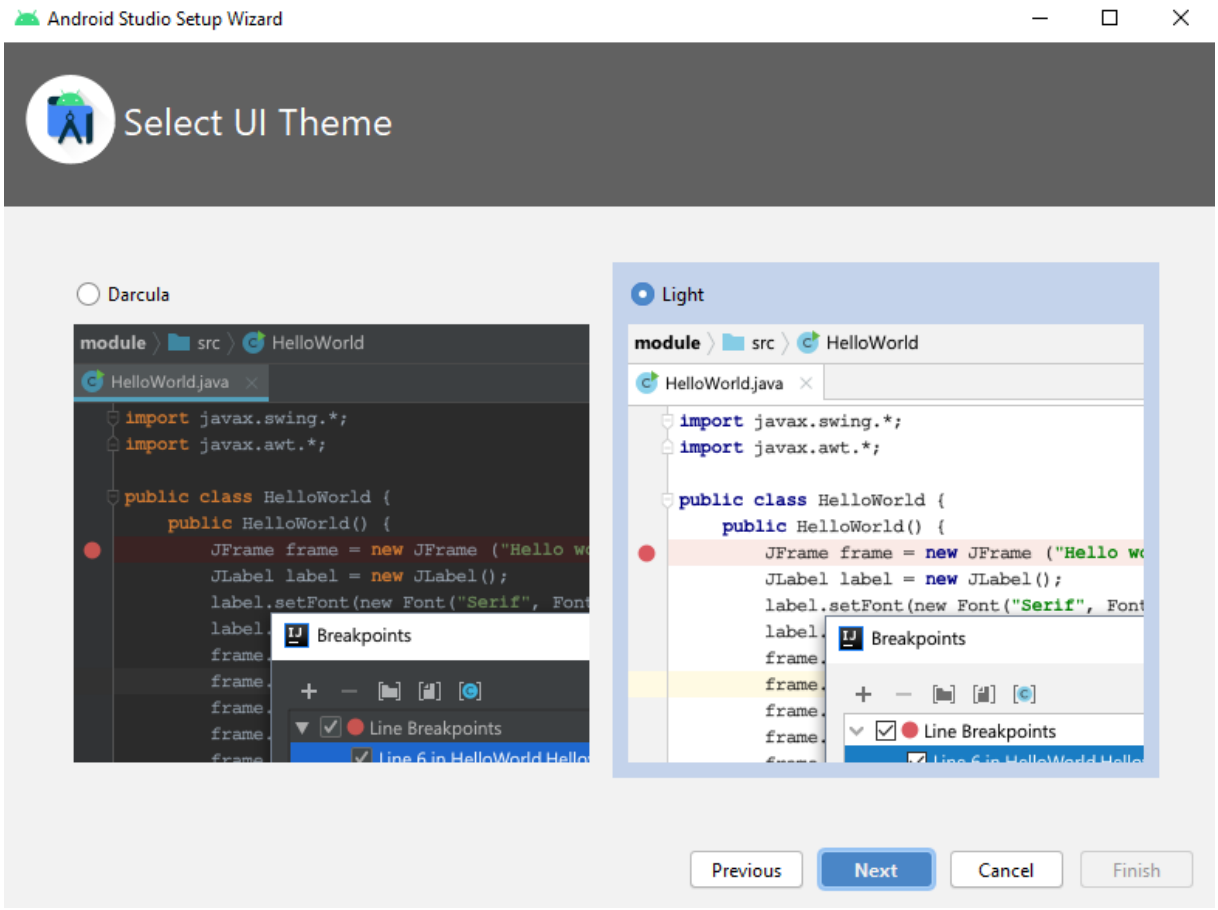
Slika 21. Pritisak na „Install“ gumb i početak instalacije alata (izvor: Autor)



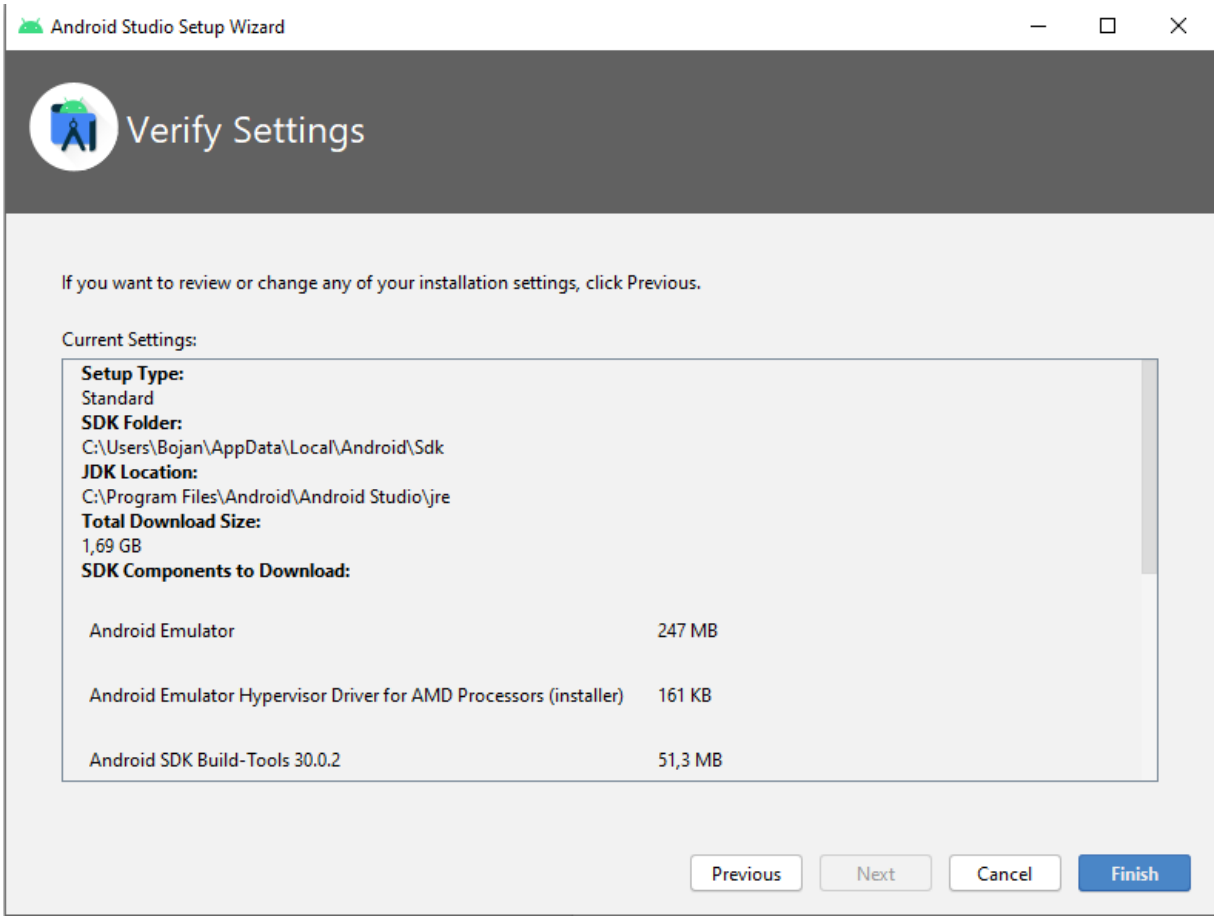
Slika 21. „Setup Wizard“ nakon prvog pokretanja alata (izvor: Autor)



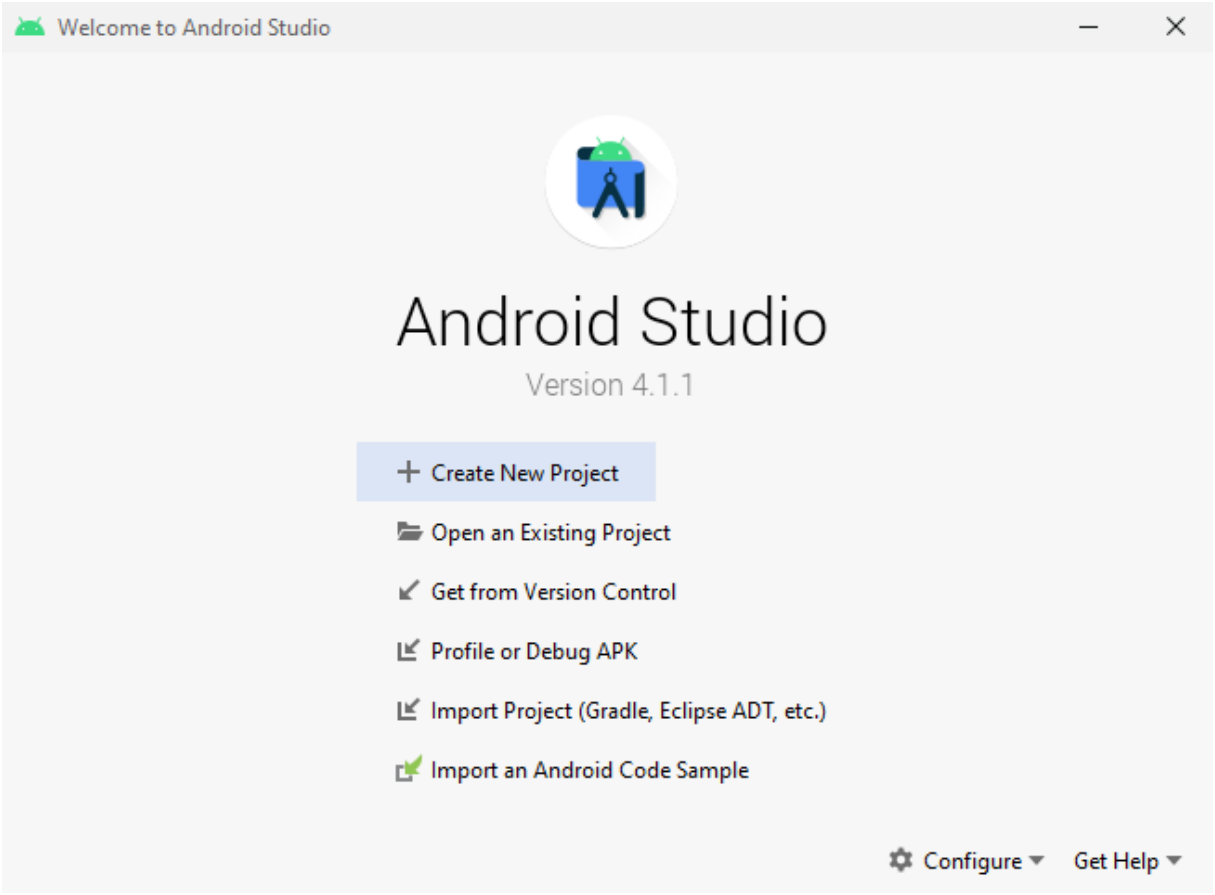
Slika 22. Odabir opcije „Standard“ pomoću koje najčešće postavke i opcije bivaju instalirane, što je i preporučeno (izvor: Autor)



Slika 24. Odabir teme alata (izvor: Autor)



Slika 25. Preuzimanje i instalacija svih dodatnih alata (SDK) koje su potrebne za funkcioniranje Studija (izvor: Autor)



Slika 26. Pokretanje alata nakon uspješne instalacije (izvor: Autor)