Aplikacija za praćenje fizičke aktivnosti

Bošnjak, Josip

Sveučilište Jurja Dobrile u Puli
Odjel za informacijske i komunikacijske tehnologije

Josip Bošnjak

Aplikacija za praćenje fizičke aktivnosti

Diplomski rad

Pula, 2018.godine.

Sveučilište Jurja Dobrile u Puli
Odjel za informacijske i komunikacijske tehnologije

Josip Bošnjak

Sustav za praćenje fizičke aktivnosti

Diplomski rad

JMBAG: 0303038620, redoviti student

Studijski smjer: Informatika

Predmet: Mobilne aplikacije
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc.dr.sc. Siniša Sovilj

Pula, 4.travnja, 2018.godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Josip Bošnjak, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada,te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, 4.Travnja, 2018. godine

IZJAVA
o korištenju autorskog djela

Ja, Josip Bošnjak, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom „Aplikacija za praćenje fizičke aktivnosti“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 4.Travnja, 2018.godine

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
ODJEL ZA INFORMACIJSKO-KOMUNIKACIJSKE TEHNOLOGIJE

Pula, 15. ožujka 2017.

DIPLOMSKI ZADATAK

Pristupnik:	**Josip Bošnjak (0303038620)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Aplikacija za praćenje fizičke aktivnosti**
Naslov (eng.):	Application for monitoring physical activity
Opis zadatka:	Zadatak je razviti mobilnu Android aplikaciju namijenjenu automatskom bilježenju, pohrani i praćenju fizičke aktivnosti korisnika (što je vrlo važno npr. u kontroli dijabetesa). Omogućiti i unos dodatnih podataka poput tjelesne mase te ručnog unosa trajanja nekih specifičnih fizičkih aktivnosti (koje nije moguće automatski registrirati senzorima mobilnog uređaja) i sl. Mobilnu aplikaciju razviti na hrvatskom jeziku te maksimalno pojednostavniti preglednost korisničkog sučelja i ispunjavanja zadanih dnevnih ciljeva. Uz mobilnu razviti i poslužiteljsku Java web aplikaciju korištenjem Java Spring programskog okvira i Apache Tomcat poslužitelja. Nakon autentikacije web aplikacija treba omogućiti dohvat podataka s mobilnog uređaja putem REST API-a, njihovu pohranu u bazu poslužitelja te vizualizaciju podataka unutar internetskog preglednika.

Zadatak uručen pristupniku: 15. ožujka 2017.
Rok za predaju rada: 1. veljače 2018.

Mentor:

Siniša Sovilj

doc.dr.sc. Siniša Sovilj

Sadržaj

Uvod

Danas je fizička neaktivnost moderna bolest suvremenog čovjeka. Lakše je naći izgovore za neaktivnost nego za aktivnost. Tipični izgovori su nemam novaca, nisam baš za sport, nema u mom mjestu nikakve teretane, nisam dovoljno dobar kao i profesionalni sportaš. Neaktivnost uzrokuje pretilost, koja može dovesti do raznih kardiovaskularnih bolesti, koje su vodeće u svijetu po smrtnosti ljudi. Kako to spriječiti? Baviti se bilo kakvom aktivnošću. Trčati, voziti bicikl, šetati, brzo hodati. Postoje razna istraživanja koja pokazuju da se ljudi sve više ohrabruju i postaju svjesni o svome zdravlju, i sve se više bave tjelesnom aktivnošću. Danas gotovo svaka osoba koja se bavi tjelesnom aktivnošću ima mobilni uređaj. Vrlo vjerojatno bi se tjelesne aktivnosti učinile zanimljivijim da recimo ljudi međusobno uspoređuju rezultate koje su postigli na dnevnoj, tjednoj, mjesečnoj ili godišnjoj bazi. Čak bi se i mogli i natjecati. Glavna motivacija za ovu temu je poboljšavanje svijesti ljudi za početak bavljenja aktivnošću umjesto da su cijelo vrijeme u kući i ništa ne rade. Zadatak ovog rada je razviti mobilnu i web aplikaciju koja će korisnicima omogućiti praćenje fizičke aktivnosti, kako bi se recimo uspoređivali rezultati. Naročito je ovo važno kod korisnika s dijabetesom. Mobilna aplikacija bi bilježila vrijeme koliko se koja osoba bavila nekom aktivnošću na dnevnoj bazi, dok bi web aplikacija omogućavala prikaz tih rezultata. Rad se sastoji od šest poglavlja, uključujući i izvorni kod na GitHubu, zajedno sa mobilnom aplikacijom i web aplikacijom,te sql dump bazom. Prvo poglavlje obuhvaća istraživanja koja govore koliki je postotak ljudi u RH koji se bave tjelesnom aktivnošću, zašto se bave i gdje. U drugom poglavlju, može se saznati ukratko povijest Java programskog jezika te *model-view-controller* oblikovnom obrascu softverske arhitekture. Treće poglavlje govori o REST-u, koji su njegovi dijelovi, pojam REST-a te koje metode obuhvaća REST. Četvrto poglavlje govori o Spring programskom okviru te objašnjava neke od dijelova anotacija koje se mogu koristiti, a korištene su i u izradi ovog sustava, pogotovo Spring web aplikacije. Peto poglavlje govori o alatima koji su korišteni pri izradi sustava, i od čega se sastoji pojedini alat (izgled editora, objašnjenje prozora itd.). Zadnje poglavlje se sastoji od opisa čitavog sustava. Uključuje objašnjenje koda, UML dijagrame, prototipove aplikacije, naputke kako koristiti aplikaciju te slike aplikacije izravno u akciji, dok se koristi, a Izvorni kod aplikacije se nalazi na adresi : https://github.com/Josip1234/SustavZaPracenjeFizickeAktivnosti.

1.Važnost tjelesne aktivnosti

U današnje vrijeme tehnologije i ubrzanog života, ljudi često nemaju vremena za sebe i za svoje tijelo. Razlozi mogu biti različiti: previše se provodi vremena na poslu a premalo se bave tjelovježbom ili pak se previše vremena provodi na računalu i tako se zanemaruje vlastito zdravlje. Vrlo je važno tjelesna aktivnost, jer pomaže u rješavanju svakodnevnog stresa, neovisno o prilikama u kojem se taj stres pojavljuje. „Prije više od 10000 godina, pripitomljavanjem životinja, meso i koža se mogla koristiti iz vlastitih izvora, a potreba za lovom je smanjena pa tako i intenzivna fizička aktivnost. U doba industrijske revolucije, 1800-1900 godine, mijenja se stil života. Otkriven je automobil, dio poslova se počelo raditi kod kuće, a automobil je zamijenio pješačenje. Pojavom video rekordera, računala i televizije, fizička aktivnost se drastično smanjila. Tako je industrijska revolucija negativno utjecala na našu aktivnost. “ (Dietpharm n.d.). Danas slobodno vrijeme iskorištavamo na nekvalitetne načine, provodeći vrijeme u kući, na računalu, gledajući televiziju, tipkajući na mobilne uređaje. Tijekom 20-og stoljeća, pojavile su se i nove bolesti suvremenog doba kao što su ospice, gripa, tuberkuloza, nedostatak vitamina i minerala, karidovaskularne bolesti, dijabetes i mnoge druge. Između ostalog, debljina postaje sve veći problem. „Debljina postaje globalni problem između ostalog zahvaća oko 65% muškaraca i 45% žena i čak 22% mladih do 17 godina života. Kod vježbanja od 45-60 minuta dnevno, može se značajno prevenirati nastanak patološke debljine, kod onih s prekomjernom težinom.“ (Dietpharm n.d.). Najčešći izgovori za izostanak fizičke aktivnosti su već spomenuti nedostatak vremena, motivacije, mislimo da smo prestari za sportske aktivnosti ili nedovoljno zdravi, nedostaje nam novaca. Takvi izgovori jednostavno nisu opravdanje jer se vježbati može u bilo kojoj dobi, na različitim mjestima, i na različite načine. Naravno, često se i misli da je za očuvanje zdravlja i poboljšavanje tjelesne aktivnosti potrebno mnogo vremena, prostor i oprema. To ne mora biti nužno tako. Danas se sve više promovira hodanje, tračanje, vožnja biciklom, plivanje ili plesanje kao oblik svakodnevne aktivnosti. Umjesto lifta, možemo jednostavno koristiti stepenice, ili pak ići u dugu šetnju sa svojim psom. Važno je odabrati i aktivnosti koje dozvoljavaju i određenu vrstu kreativnosti. Preporučljivo je barem 30 minuta brzog hoda ili vježbanja dnevno koje može biti podjeljeno i u dva dijela.

1.1. Korisni učinci fizičke aktivnosti

„ Moderno doba za sobom je donijelo i mnoge negativne posljedice. Primjerice, u sredstvima javnog informiranja veća se pozornost pridaje sportu i to s naglaskom na postizanje rezultata, a na taj je način u podređeni položaj dovedena tjelesna vježba koja ima mnoge ciljeve, kao što su prevencija i terapija kod različitih bolesti i slično.“ (Bartoš 2015). Na žalost, u cijelom se svijetu smanjuje potreba za fizičkom aktivnosti, pogotovo u velikim metropolama, gdje gotovo da nema mjesta za fizičku aktivnost ili je preskupo, kao što je spomenuto u prethodnom poglavlju. I za to postoji rješenje. Potrebno je otići na neko mjesto gdje se mogu te aktivnosti provoditi, a izgovor nema se novaca, jednostavno nije prihvatljiv. „Prema podacima Državnog zavoda za statistiku iz 2006. godine, u Hrvatskoj je aktivnih sportaša svega 6% od ukupnog broja stanovništva, a tjelesno je neaktivno 83% muškaraca i 95% žena; pretilo je 48% muškaraca i 34% žena, povišeni kolesterol ima 55% muškaraca i 55% žena, a visoki krvni tlak ima 32% muškaraca i 24% žena.“ (Bartoš 2015). Sve je to posljedica neaktivnosti i predstavlja jednu od mnogih bolesti suvremenog doba. Pozitivni učinci koji se žele postići fizičkom aktivnosti prikazane su na slici.

Slika 1.Pozitivni učinci fizičke aktivnosti, Izvor: Bungić, M.; Barić: Tjelesno vježbanje i neki aspekti psihološkog zdravlja, Hrvatski športskomedicinski vjesnik, Vol. 24 No. 2, Zagreb, 2009., str. 67.

„Tjelesna aktivnost djeluje preventivno na kororarne bolesti, čime dugoročno utječe na usporavanje smanjenog radnog kapaciteta do kojeg dovodi povećanje kronološke dobi. Novija istraživanja povezuju neaktivan način života s pojavom nekih oblika malignih bolesti. “ (Andreja Bogdan 2015). „Također, bitno je spomenuti da tjelesna aktivnost niskog intenziteta nema primjeren utjecaj u jačanju funkcijske sposobnosti, dok se značajna djelotvornost može očitati u aktivnosti umjerenog intenziteta. Tjelesna aktivnost prejakog intenziteta u slučaju nedovoljne pripremljenosti organizma može dovesti do neželjenih zdravstvenih posljedica.“ (Andreja Bogdan 2015). U citiranoj literaturi, postoje i grafovi koliko tko vježba i u kojoj dobi pa čak i zbog čega tko vježba. To istraživanje je provedeno na uzorku od 200 ispitanika starijih od 20 godina, u oba spola. Grupe su podjeljene na mlađe osobe od 20-40 godina, srednju dob od 40-60 godina, i na kasnu odraslu dob 60+ godina. U ovom radu, bitno je prikazati potrebu za aplikaciju koju bi eventualno ti korisnici mogli koristiti te da okvirno pokaže opravdanost izrade takve aplikacije. Zbog nemogućnosti provedbe vlastitog istraživanja, u ovom radu je korišteno istraživanje od Andreje Bogdan i Daniele Babačić[1].

	20-40 godina		40 do 60 godina		60 godina i više	
Zbog zdravlja	15	22,7%	14	46,7%	4	44,4%
Iz hobija	11	16,7%	3	10,0%	2	22,2%
Radi postizanja željene težine	10	15,2%	3	10,0%	0	0,0%
Radi ispunjavanja slobodnog vremena	1	1,5%	0	0,0%	1	11,1%
To je moj stil života	24	36,4%	10	33,3%	2	22,2%
Radi pružanja boljeg primjera djeci	0	0,0%	0	0,0%	0	0,0%
Ostalo	5	7,6%	0	0,0%	0	0,0%

Slika 2.Razlozi sudjelovanja u sportskim aktivnostima, izvor: Andreja Bogdan, Daniela Babačić. »Intrinzična i ekstrinzična motivacija za sport i vježbanje u funkciji dobi.« str. 5.

U skupini od 20-40 godina starosti, trećina ispitanika doživljava sport i tjelovježbu kao stil života. U skupini od 40-60 godina starosti, bavljenjem sportom je još uvijek

[1] Andreja Bogdan, Daniela Babačić. 2015. »Intrinzična i ekstrinzična motivacija za sport i vježbanje u funkciji dobi.« *Stručni rad.* Čakovec: Međimursko veleučilište u Čakovcu, 2. 10.

obilježje stila života. Više se navodi da se također vježba da se očuva tjelesno zdravlje što nije slučaj za mlade. Prema ovom istraživanju, mladi vježbaju i vježbanje im predstavlja stil života, no što godine prolaze, sve se više vježba u svrsi očuvanja zdravlja. Što je s razlozima za ne bavljenje tjelesnom aktivnošću? Sljedeći graf iz istog rada će to prikazati, također je izradio autor tog istraživanja. Sljedeća tablica prikazuje rezultate istraživanja za nesudjelovanje u sportskim aktivnostima.

Ne bavim se sportom zbog:

	20-40 godina		40 do 60 godina		60 godina i više	
Zdravstvenog stanja koje mi to ne dozvoljava	2	5,1%	5	16,7%	7	26,9%
nedostatka motivacije	18	46,2%	6	20,0%	7	26,9%
obiteljskih obaveza	10	25,6%	2	6,7%	0	0,0%
poslovnih obaveza	4	10,3%	6	20,0%	1	3,8%
nedostatka novaca	2	5,1%	3	10,0%	0	0,0%
nedostatne ponude	2	5,1%	1	3,3%	0	0,0%
sport me uopće ne zanima	0	0,0%	4	13,3%	11	42,3%
ostalo	1	2,6%	3	10,0%	0	0,0%

Slika 3.Razlozi nesudjelovanja u aktivnostima, izvor: Andreja Bogdan, Daniela Babačić. »Intrinzična i ekstrinzična motivacija za sport i vježbanje u funkciji dobi.« str.10 .

Prema tablici, glavni razlog zbog čega se ne bave sportom je manjak motivacije ili obiteljskih obaveza. Kod odraslih je pretežito manjak motivacije, poslovne obaveze a tu je negdje i blizu nezainteresiranost za sport. Kod starijih osoba, najveći razlog je nezainteresiranost, a slijede nedostatak zdravlja te nedostatak motivacije. Prema ovom istraživanju na uzorku od 200 osoba, možemo zaključiti našu ciljanu skupinu za koju je potrebno izraditi aplikaciju za praćenje fizičke aktivnosti.Ciljane skupine ljudi koje bi eventualno koristile ovu aplikaciju su mladi ljudi od 20-40 godina i ljude srednje

dobi između 40 i 60 godina.

EU prosjek 42%							
Austrija	27%	Estonija	36%	Litva	46%	Portugal	64%
Belgija	31%	Finska	15%	Luksemburg	29%	Rumunjska	60%
Bugarska	78%	Francuska	42%	Malta	75%	Slovačka	41%
Češka	35%	Grčka	59%	Mađarska	44%	Slovenija	22%
Cipar	54%	Irska	34%	Nizozemska	29%	Španjolska	44%
Hrvatska	29%	Italija	60%	Njemačka	29%	Švedska	9%
Danska	14%	Latvija	39%	Poljska	52%	Ujedinjena Kraljevina	35%

Slika 4. Postotak pučanstava po državama koji se nikada ne bave sportom, Izvor: (Andreja Bogdan 2015) str.15.

„Istraživanje je provedeno na uzorku koji je sačinjavalo 200 ispitanika, od čega je 129 (65%) bilo žena i 71 (35%) muškaraca. Rezultati provedenog anketnog istraživanja na uzorku od 200 osoba različite životne dobi (65% žena i 35% muškaraca) na području RH, sportskom aktivnošću bavi se 52% ispitanih. “ (Andreja Bogdan 2015).

1.2. Hrvatski olipmpijski odbor-istraživanje 2017

„Sportskom, rekreacijskom ili drugim vidom tjelesne aktivnosti bavi se 38% građana Hrvatske, među kojima samo 12% čini tri do četiri puta tjedno, a 9% pet puta tjedno ili češće. Tjelesna aktivnost češće je navika stanovnika velikih gradova, visokoobrazovanih tridesetogodišnjaka na višim pozicijama i s višim prihodima te učenika i studenata u dobi od 15-20 godina. Kad je riječ o spolu, podjednako se ovim aktivnostima bave i žene i muškarci. Navike se međutim mijenjaju sa dobi pa se Hrvati, što su stariji, sve manje bave navedenim tjelesnim aktivnostima. Najmlađa dobna skupina (15-20 godina) značajno se više i češće bave tjelesnim aktivnostima, 72% njih barem jednom tjedno. Najmanje su tjelesno aktivni stariji od 60 godina te se 80% njih nikada ne bave tjelesnom aktivnošću. Vrijeme koje Hrvati odvajaju za bavljenje navedenim tjelesnim aktivnostima u prosjeku je 40 minuta, a najveći dio njih (35%) provede 30 minuta do do sat vremena baveći se nekom od ovih aktivnosti. Žene češće

biraju kraće aktivnosti (do 30 minuta), dok najmlađa dobna skupina češće od prosjeka populacije za rekreaciju izdvoji između 91 i 120 minuta. Otvoreni prostori, parkovi i priroda (46%) Hrvatima su omiljeno mjesto za bavljenjem navedenim aktivnostima, sportski ili fitness klub i centar bira oko 30% građana,dok kod kuće vježba njih 27%. “ (HINA 2017). Dakle, postoji 46% potencijalnih korisnika koji bi koristili aplikaciju za prikupljanjem podataka o svojim dnevnim aktivnostima, i da se pri tome mogu voditi dnevnici te prema potrebi smanjivati tempo vježbanja ili povećavati ih. Najćešće bi se smanjivao tempo tijekom ljeta, kada ljudi idu na godišnje odmore, a najviše bi se povećavali tijekom zime, nevisno o mjestu aktivnosti. „Zdravlje je glavni motiv za 61% građana da se bave sportskim, rekreacijskim ili drugim vidom aktivnosti. Osim radi zdravlja, najviše vježbaju kako bi poboljšali tjelesnu kondiciju (38%), opustili se (37%), bili s prijateljima (29%), zabavili se (27%), poboljšali svoj izgled (25%), kontrolirali težinu (18%) i poboljšali tjelesnu izvedbu. Za većinu stanovništva (61%) oblik športskih i tjelesnih aktivnosti kojima se bave ne zahtijeva nikakve novčane izdatke jer preferiraju aktivnosti na otvorenome. Dobiveni rezultati istraživanja, koje je provedeno na nacionalno reprezentativnom uzorku od 1000 stanovnika RH starijih od 15 godina (52% žena i 48% muškaraca), potvrđuju da građani Hrvatske nisu dovoljno aktivni te njihova svijest o prednostima športa i zdravih navika kretanja još uvijek nije dovoljno razvijena. Potrebno ih je potaknuti na bavljenje bilo kojom vrstom tjelesne aktivnosti. “ (HINA 2017). Prema tome, ciljana publika kojoj je namjen ovaj sustav je isključivo za onu populaciju koja se bavi aktivnostima u prirodi, a to su najčešće aktivnosti hodanja, trčanja i vožnja biciklom. S pomoću senzora, moguće je uzimati podatke kao što su brzina kretanja, broj prehodanih kilometara, lokacije ili najdraže rute.

2. Java i MVC

2.1.O Javi

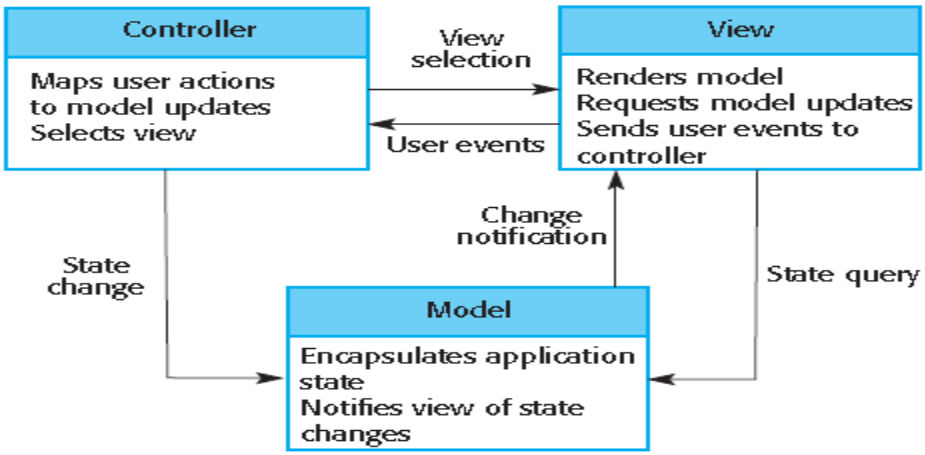
Java se smatra jezikom koji se može upotrijebiti svugdje, za bilo kakvu uporabu. 1991. razvijena je prva Java. Razvila ga je tvrtka Sun Microsistems, a tim koji je razvio prvu verziju jave, predvodio je James Gosling. „Prva namjena jave bila je u perilicama rublja ili za televiziju. 1994. Patrick Naughton i Johnathan Payne u Sun Microsystemsu su razvili web preglednik koji je mogao pokretati Java programe preko interneta. Taj preglednik se zvao HotJava. U jesen 1995, Netscape Incorporated su naprvili svoj web

preglednik koji je imao sposobnost pokretati Java programe. Druge kompanije su ih tužile, i razvile su također programe koje omogućuju pokretanje Jave.“ (Savitch 2016). „Danas je Java vlasništvo Oracle kompanije i primjenjuje se u svakodnevnoj upotrebi, od osobnih računala, do glavnih aplikacija kojim se koriste poduzeća. „Java je objektno orijentirani jezik. Svijet se oko nas sastoji od raznoraznih objekata. Svaki od tih objekata ima određeno ponašanje, i svako ponašanje određenog objekta ima utjecaj na ponašanje drugog objekta. Objektno orijentirano programiranje ima vlastitu terminologiju. Objekti su instance klase, metode su akcije koje neki objekt može izvoditi. Objekti mogu biti sličnog razreda ili klase, ili različitog razreda ili klase. “ (Savitch 2016). U Javi, postoje pojmovi appleti i aplikacije. Aplikacija je jedan običan program. Applet znači mala Java aplikacija. Aplikacija se pokreće sa nečijeg računala, dok se applet pokreće preko web preglednika. Applet se može također pokretati i preko preglednika appleta. Preglednik je bio namjenjen kao okruženje za pronalaženje grešaka u programu, i nije bio namijenjen za konačno okruženje u kojem bi se pokretale aplikacije. Applet uvijek ima prozor, dok se aplikacija može pokretati pomoću konzole.“ (Savitch 2016). Danas se Java može i koristiti i za razvoj mobilnih aplikacija, koristeći objektno orijentirano programiranje te raznoraznih drugih mogućnosti, poput senzora, praćenje lokacija, očitavanja vanjske temperature. Naravno, to je sve počelo od 1991. od kada je prva verzija Java programa pokretala sve ugradbene sustave. Postoji dosta alata preko kojih se može programirati sa Javom, kao što je IntelliJ IDEA, Android Studio, Eclipse, dodatak za Eclipse Spring Tool Suite. Danas popularni programski okvir za programiranje Jave je Spring. Spring se može svugdje pokrenuti i to je veoma bitno svojstvo.

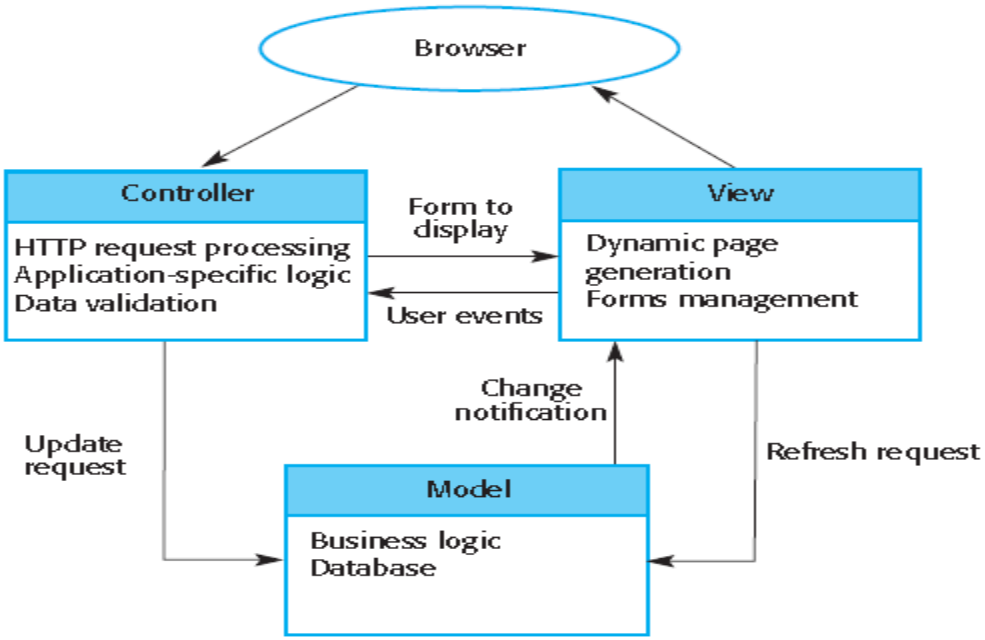
2.2. Pojam model-prikaz-kontroler

„Ideja o modelima kao načinu predstavljanja, dijeljenja i ponovna uporaba znanja o programskim sustavima je prihvaćena u mnogobrojnim područjima softverskog inženjerstva. Glavni okidač za ovo bila je publikacija knjige o objektno-orijentiranom modelu dizajna. Bez odlaganja, razvili su se i drugi modeli kao što su modeli za organizacijski dizajn, modeli iskoristivosti, modeli za kooperacijsku interakciju te modeli za upravljanje konfiguracijama. Model-prikaz-kontroler odvaja prezentaciju i interakciju od sustavnih podataka. Sustav je struktuiran u tri logičke komponente koje su međusobno u interakciji. Komponenta modela upravlja sustavnim podacima i vezanim

operacijama na tim podacima. View komponenta definira i upravlja kako se podaci prikazuju korisniku. Kontroler komponenta upravlja korisničkom interakcijom (npr. pritiskom tipke na tipkovnici, klikom miša itd.) te šalje te interakcije prema prikazu i modelu. Ovaj se model koristi kada postoji puno različitih načina za pregled i interakciju sa podacima. Također se koristi kada se budući zahtijevi za interakciju i prezentaciju nepoznati. Glavne prednosti ovog modela da omogućuju podacima promijenu nevisno o njihovoj reprezentaciji. Podržava prezentaciju istih podataka na različite načine. Nedostatci ovog modela je da zahtijeva dodatno kodiranje i veću složenost programskog koda, dok su modeli podataka i interakcija jednostavni. “ (Sommerville 2016). Slika 5. prikazuje organizacijski model-prikaz-kontrolera. Kontroler mapira korisničke akcije prema ažuriranjima modela i odabire prikaz. Prikaz šalje korisničke događaje prema kontroleru, traži ažuriranje modela i prikazuje model podataka. Model enkapsulira stanje aplikacije i obaviještava prikaz o stanju promijena. Slika 6. prikazuje arhitekturu web aplikacije korištenjem MVC modela. Kontroler obrađuje HTTP zahtjeve, u njemu postoji logika specifična za aplikaciju i omogućuje validaciju podataka. Prikaz generira dinamičke stranice, i upravlja obrascima. Model sadrži bazu podataka i poslovnu logiku. Model prima zahtjeve za ažuriranje, i za osvježavanje od strane kontrolera i prikaza. Prikazu vraća obavijest o promijeni. Prikaz šalje kontroleru korisničke događaje, a od kontrolera dobiva obrazac za prikaz. Na kraju se prikaz prikazuje u web pregledniku. Kada korisnik u generiranom obrascu upiše podatke, taj obrazac se šalje kontroleru gdje kontroler obrađuje HTTP zahtjev.



Slika 5. Organizacija model-prikaz-kontrolera, izvor: Sommerville, Ian. 2016. Software Engineering, str:176.



Slika 6. Arhitektura web aplikacije pri korištenju MVC modela, izvor: Sommerville, Ian. 2016. Software Engineering, str:177.

3.REST

„REST[2] danas predstavlja jedan od najvažnijih i najpopularnijih oblikovnih obrazaca za opisivanje željene web arhitekture i kreiranje aplikacija i web usluga.“ (Kuserbajn 2016). „Rest arhitektura se generalno sastoji od klijenta, poslužitelja, resursa te HTTP operacija poznatijih kao metode zahtijeva koje REST koristi za svoje projektiranje. Klijent šalje zahtijev poslužitelju, a poslužitelj odgovara.“ (Kuserbajn 2016). Sama ideja REST oblikovnog obrasca je u tome da se umjesto kompleksnih mehanizama kao što su RPC ili SOAP, koristi jednostavan HTTP za uspostavljanje veze između uređaja. Primjerice, imamo web aplikaciju napravljena za komuniciranje preko baze u kojoj se obavljaju osnovne operacije poput umetanja, brisanja, ažuriranja ili zamjenjivanja. Putem REST arhitekture, moguće je povezati mobilnu aplikaciju i web aplikaciju mnogo jednostavnije, nego što je to moguće preko drugih arhitektura. „REST je nastao kao spoj nekoliko mrežno temeljenih oblikovnih obrazaca.“ (Kuserbajn 2016). Neki od obrazaca prema kojima se temelji aplikacijski program su obrazac protoka podataka (*Data-Flow styles*), replikacijski obrazac (*Replycation Styles*), hijerarhijski obrazac,

[2] Representational State Transfer- Predstavlja način prikazivanja podataka na internetu

obrazac pokretnog koda (Mobile Code Styles) i obrazac najbližih susjeda (*peer-to-peer styles)*. REST arhitektura sastoji se od primjene REST ograničenja, ta ograničenja predstavljaju ograničenja interakcije između komponenata, konektora i podatkovnih elemenata.

3.1.Dijelovi REST-a

Službena REST ograničenja su klijent-poslužitelj (client-poslužitelj), neovisnost između zahtjeva (stateless), priručna memorija (cache), jedinstveno sučelje (uniform interface), slojeviti sustav (layered system) te kod na zahtjev (code on demand). „Jedinstveno sučelje definira sučelje između klijenta i poslužitelja. Pojednostavljuje arhitekturu u smislu da dopušta svakom dijelu te arhitekture da se razvija neovisno o drugim dijelovima sustava." (Fredrich 2012). Klijenti i poslužitelji imaju svoje određene zadatke. Klijent se ne bavi pohranom podataka, već o tome brine poslužitelj. Poslužitelji se ne brinu o korisničkom sučelju ili korisničkom stanju. Odvajanje klijenta i poslužitelja čine arhitekturu puno jednostavnijom. „REST predstavlja akronim za prezentacijsko stanje transfera ili representational state transfer, gdje je neovisnost između zahtjeva ključ." (Fredrich 2012). „To ograničenje se dodaje u interakciji između klijenta i poslužitelja. Komunikacija se mora odvijati na način da je svaki novi zahtjev neovisan o onom prijašnjem što čini i zahtjev i odgovor neovisnim parom. Svaki poslani zahtjev od klijenta mora sadržavati sve potrebne informacije kako bi ga se razumjelo jer se ne može iskoristiti bilo kakav prošli zahtijev ili kontekst za bolje razumijevanje." (Kuserbajn 2016). Dakle, za svaki zahtjev se kreira novi zahtjev, pri čemu poslužitelj ne mora brinuti o resursima, i pri implementaciji se oslobađaju već zauzeti resursi. Kod priručne memorije, ograničenje postavljeno na tu vrstu memorije zahtijeva da se podaci unutar odgovora na zahtijev implicitno ili eksplicitno označavaju kao podaci koji se spremaju u priručnu memoriju ili kao oni koji se ne spremaju. „Podaci koji se ne spremaju su podaci čija se vrijednost mijenja te ih stoga nema svrhe koristiti. Ako se podatak označi kao onaj koji se sprema, onda ga klijent može ponovno koristiti za kasniji odgovarajući zahtjev." (Kuserbajn 2016). „Jedinstveno sučelje definira sučelje između klijenta i poslužitelja. Pojednostavljuje arhitekturu, i pri tome omogućava da se svaki dio razvija sam za sebe." (Fredrich 2012). Postoje četiri principa sučelja: bazirano na resursima, manipulacija resursa kroz reprezentaciju, samoopisivajuće poruke te hipermedija kao motor aplikacijskog stanja (HATEOAS). „Individualni resursi se

identificiraju u zahtjevima koristeći se URI-ima kao identifikatorima resursa. Resursi su međusobno odvojeni od prezentacijskog sloja koji se vraćaju na stranu klijenta. Primjer: poslužitelj ne šalje svoju bazu podataka, već neki oblik HTML dokumenta, XML dokumenta ili JSON polja koji služe za prezentaciju određenih podataka u enkodiranom UTF-8 obliku, koji ovisi o implementaciji poslužitelja i zahtjeva.“ (Fredrich 2012). „Kod manipulacije resursima, kada klijent drži reprezentaciju resursa, uključujući metapodatke u prilogu, onda ima dovoljno informacija da izmijeni ili izbriše resurse. Kod samopisnih poruka, svaka poruka sadrži dovoljno informacija koji opisuju kako je obraditi.“ (Kuserbajn 2016). Kod hipermedija, klijenti obavljaju stanje prijelaza samo kroz akcije koje su dinamički identificirani unutar hipermedija od strane poslužitelja. „Slojeviti obrazac sustava omogućava arhitekturi da bude sastavljena od hijerarhijskih slojeva tako što ograničava ponašanje komponente na način da svaka komponenta ne može vidjeti dalje od trenutnog sloja s kojim je u interakciji.“ (Kuserbajn 2016). „ Kod koda na zahtjev, REST omogućava proširenje klijentove funkcionalnosti preuzimanjem i izvršavanjem koda u obliku skripte ili appleta (Java appleti ili Java Script). Tako se smanjuje broj značajki koje se moraju provesti. Dopuštanje da se značajke preuzmu nakon implementacije dopušta proširenje sustava, ali i smanjuje vidljivost pa to predstavlja samo još jedno dodatno ograničenje REST-a.“ (Kuserbajn 2016).

3.2.RESTful web usluge

„REST se primarno koristi za razvoj web usluga koje su lake za održavanje i skalabilne. Usluga bazirana na REST arhitekturi naziva se RESTful uslugom. REST je neovisan o bilo kojem protokolu ali skoro svaka REST usluga koristi HTTP kao standardni protokol.“ (Kuserbajn 2016). „Svaki sustav koristi resurse. Ti resursi mogu biti slike, video, web stranice, poslovne informacije itd. Svrha usluge je da pruži prozor ili okvir svojim klijentima tako da oni mogu pristupiti tim resursima. Svaka RESTful usluga bi trebala biti implementirana na način da budu održive, skalabilne i nadogradive.“ (Kuserbajn 2016). „Fokus REST usluga su resursi i kako pružiti pristup tim resursima. Jedan resurs se može sastojati od više drugih resursa. Prilikom projektiranja sustava, prvo je potrebno identificirati resurse i utvrditi kako se oni međusobno povezuju. Nakon identifikacije resursa, potrebno je naći način za reprezentaciju ili prikaz tih resursa. Dozvoljeno je korištenje bilo kojeg formata za

reprezentaciju jer REST na njih ne stavlja ograničenja. Ovisno o zahtjevu, može se koristiti XML ili JSON. “ (Kuserbajn 2016). Resursi mogu biti dokument, slika, privremene usluge, kolekcije drugih resursa itd. „Klijent i poslužitelj razgovaraju putem poruka. Klijent pošalje zahtjev, a poslužitelj na njega reagira. Osim podataka te poruke sadrže i metapodatke. Kako bi se projektirala RESTful usluga, potrebno je poznavati i HTTP format za zahtjeve i odgovore. REST zahtjeva da svaki resurs ima barem jedan URI[3]. RESTful sustavi bi trebali imati jedinstveno sučelje. HTTP pruža skup metoda nazvanih riječima. Najpoznatije ključne riječi su GET, PUT, POST, DELETE, OPTIONS, HEAD itd. REST preporuča jedinstveno sučelje dok ga HTTP pruža. U REST obrazcu interakcija je neovisna, što znači da se ne podržava pohrana podataka za nijednog klijenta. Ne postoji ovisnost novog zahtjeva o prošlom zahtjevu i svaki se novi zahtjev tretira kao novi. Reprezentacija resursa može sadržavati veze do drugih resursa kao što i web stranica pruža vezu do druge stranice. Te veze se nalaze u HTTP dokumentima.“ (Kuserbajn 2016). Kod privremenog skladištenja podataka, koji predstavlja koncept memoiranja generiranih rezultata i korištenja tih rezultata umjesto da se oni iznova generiraju, omogućuje se poboljšavanje korisničkog sadržaja, a ukoliko se ne koristi primjereno, može doći do toga da poslužitelj koristi zastarjele podatke. Ti podaci se moraju ažurirati. Privremeno skladištenje se može napraviti na klijentu, poslužitelju ili nekoj drugoj komponenti između njih. Postoje ove REST komponente: Korisnički agent (*User agent*), izvorni poslužitelj (origin poslužitelj), posrednik (*proxy*) te poveznik (*gateway*). Kontrola podataka se može vršiti pomoću sljedećih HTTP zaglavlja: *Date*, *Last modified*, *Cache-control*, *Age*, *Expires*.

3.3.HTTP metode koje koristi REST

HTTP predstavlja aplikacijski protokol. Temelj je komunikacije podataka za www[4]. Funkcionira kao protokol zahtjev-odgovor. „HTTP definira metode koje ukazuju na željene akcije koje bi se trebale izvršiti nad identificiranim resursom.“ (Kuserbajn 2016). HTTP 1.0 definira GET, POST i HEAD metode, dok HTTP 1.1. dodaje OPTIONS, PUT, DELETE, TRACE I CONNECT. Najčešće se koriste POST, GET, PUT, PATCH, DELETE. „GET metoda zahtijeva prikaz nekog resursa. Zahtijevi koji koriste GET trebali bi samo dohvatiti podatke, bez bilo kakvih drugih učinaka. PUT metoda - traži

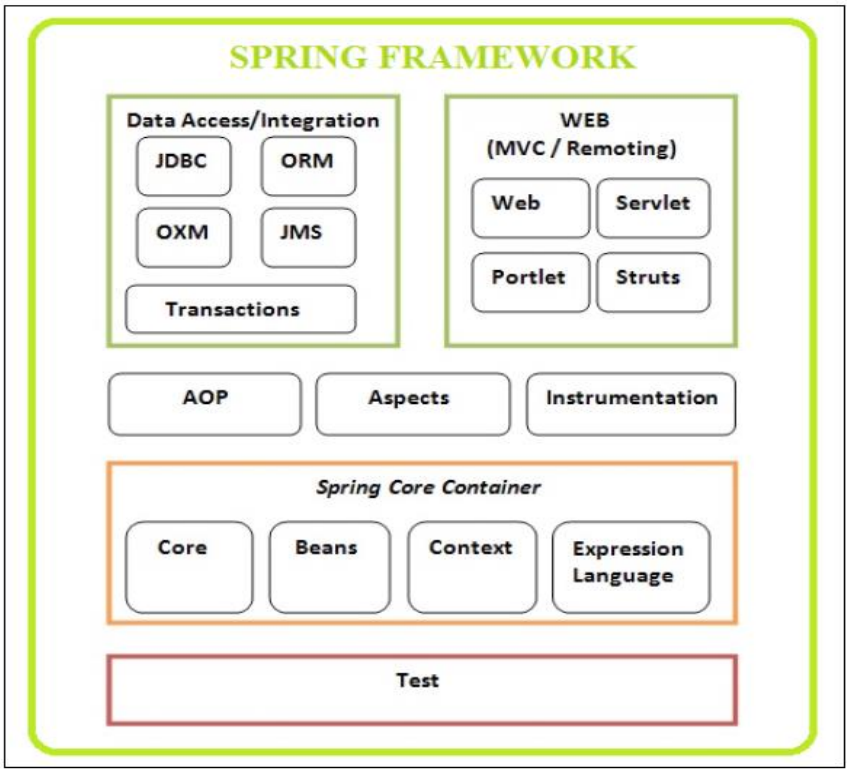
[3] Niz znakova koji se koriste za identifikaciju resursa ili skupa resursa
[4] Oznaka za World Wide Web

da se entitet u prilogu skladišti samo u pruženom URI-u. Ako se URI odnosi na već postojeći resurs onda je on modificiran, a ako se ne odnosi na postojeći resurs onda poslužitelj može kreirati novi resurs s tim URI. POST metoda zahtijeva da poslužitelj prihvati entitet zatvoren u zahtijevu kao novi podređeni entitet web resursa identificiran od strane URI. Podaci koji su poslani preko POST metode mogu biti poruke za oglasnu ploču, označavanje postojećih resursa, interesne grupe itd. A DELETE metoda briše specificirani resurs.“ (Kuserbajn 2016). Glavne metode za obavljanje CRUD operacija su za stvaranje HTTP PUT, za čitanje HTTP GET, za ažuriranje HTTP POST te za brisanje HTTP DELETE.

4.Spring razvojni okvir

„Spring predstavlja programski okvir i kontejner za aspektno orjentirano programiranje. Spring je kreiran kako bi se ublažilo kompleksnost nekadašnjeg J2EE standarda, dajući alternativni model. Bilo koja Java EE aplikacija može imati koristi od Spring razvojnog okvira, kroz jednostavnost, labavo spajanje i kroz sposobnost za testiranje. I dalje ostaje popularan zbog jednostavnosti izgradnje aplikacija. Također nudi dosljedan programerski model za različite vrste tehnologija, kao što je recimo pristup podacima ili slanje poruka pojedinoj infrastrukturi. Ovo razvojni okvir omogućuje korisniku da odredi određene probleme i izradi rješenja za njih. Spring razvojni okvir omogućuje opsežnu podršku za razvoj Java EE aplikacija, gdje Spring upravlja infrastrukturom, dok se razvojni programer koncentrira samo na razvoj aplikacije. Spring omogućuje dobru programsku praksu u obliku kodiranja sučelja i omogućavanja Java EE jednostavnije za korištenje. To pravi pomoću omogućavanja takozvanog Plain Old Java Object (POJO) modela, koji se može omogućiti u širokom rasponu okruženja za razvoj aplikacija. POJO predstavlja običan objekt koji ne bi trebao implementirati specificirano sučelje ili nasljediti specificiranu klasu ili sadržavati bilo kakve anotacije. Spring je modularan, dopušta korištenje određenih dijelova koje će sadržavati određena aplikacija, bez bespotrebnih komplikacija. Spring se može koristiti za sve slojeve implementacija ili za razvoj samo pojedinačnog sloja aplikacije. Glavne mogućnosti koje Spring pruža su: laki razvojni okvir, nenametljivost (neovisnost s razvojim okvirom), inverzija kontrole (IoC) koji uključuje Spring beanove i njihov životni ciklus, aspektno orjentirano programiranje (ASP), JDBC upravljanje iznimkama,

Spring model-prikaz-kontroler oblikovni obrazac (MVC), Spring sigurnost. Druge mogućnosti koje Spring ima su: Spring Web Usluge koje omogućuju prvi ugovorni model za razvoj web usluga, gdje se usluge pišu da bi zadovoljavali uvjete ugovora, Spring batch - korisno kada se trebaju obavljati operacije nad podacima, Spring social - socijalno umrežavanje te Spring mobile - podržava razvoj za mobilne web aplikacije.“ (Ravi Kant Soni 2017).



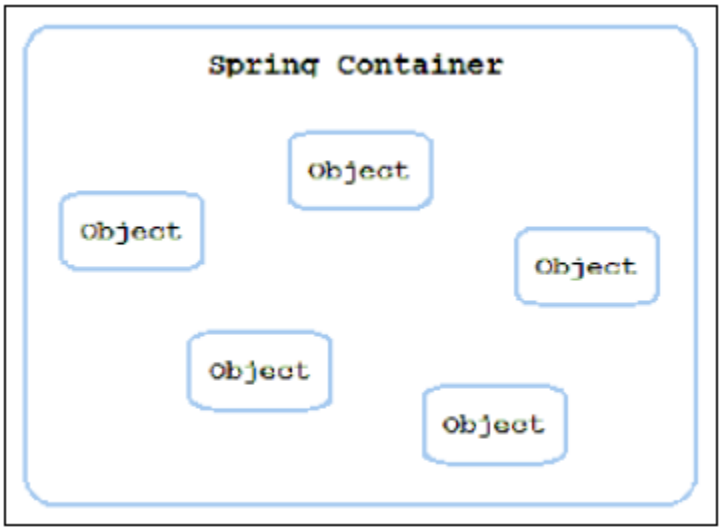
Slika 7.Arhitektura Springa, izvor: (Ravi Kant Soni 2017).

Slika prikazuje arhitekturu Spring razvojnog okvira, u ovom projektu, korišteni su samo pojedini dijelovi. Osnovni pojmovi su objašnjeni u sljedećim potpoglavljima. U razvoju Spring web aplikacije, korišteni su beansi, JDBC, servlet i web. Spajanje na bazu podataka izvodi se klasičnim putem, kroz Java programiranje, bez upotrebe Springa, ali koristi se MVC kako bi se aktivirao spoj na bazu i služi recimo za unos korisnika. Spring JDBC se koristi za unos podataka sa mobilne aplikacije, ažuriranja korisničkog profila i brisanja korisničkog profila. Spring JDBC se zapravo mnogo razlikuje od klasičnog spajanja na bazu, jer zahtijeva mnogo manje kodiranja, nego sa klasičnom Java vezom prema bazi. Nema pisanja iznimki, Spring JDBC zapravo sam

upravlja iznimkama, što je još jedno korisno svojstvo.

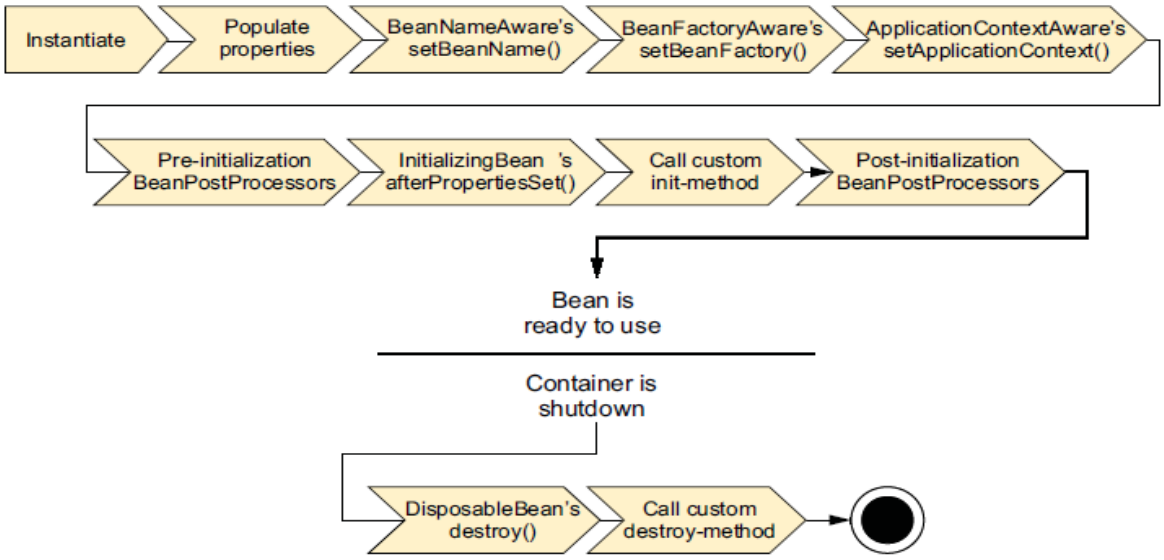
4.1.Spring bean

„Beanovi predstavljaju ponovno iskoristive softverske komponente kojima upravlja Spring IoC kontejner. Oni sadržavaju svojstva, getter i setter metode od klase. U programskom inženjerstvu IoC predstavlja određenu programsku tehniku u kojem se objekti spajaju tijekom izvođenja od strane asembler objekta i uobičajeno nije poznata tijekom vremena prevođenja koristeći statističku analizu. IoC može biti klasa, klijent ili samo neka vrsta IoC spremnika. Riječ spremnik opisuje bilo koju komponentu koja može sadržavati druge komponente unutar njih. Spring spremnik je središnja komponenta Springa. Spring kontejner upravlja životnim ciklusom pojedinog beana koji traje unutar spremnika. “ (Ravi Kant Soni 2017).



Slika 8.Spring kontejner, izvor: (Ravi Kant Soni 2017).

Spring beanovi se mogu definirati i konfiguirati na dva načina: putem XML-a i putem Java konfiguracije. U projektu se koristila Java konfiguracija beana. Verzija Apache Tomcata od 7.0 na dalje prihvaća definiranje beana putem java klase, dok se na ranijim verzijama poslužitelja i dalje mora definirati putem XML-a.



Slika 9.Životni ciklus beana, izvor: (Walls 2015)

„Prvo Spring napravi instancu beana. Zatim injektira vrijednosti i reference beana u svojstva beana. Ako bean implementira BeanNameAware, Spring prosljeđuje id od beana do setBeanName() metode. Ako bean implementira BeanFactoryAware, Spring poziva setBeanFactory() metodu, prosljeđujući se u bean factory. Ako bean implementira ApplicationContextAware, Spring poziva setApplicationContext() metodu prosljeđujući referencu na zatvarajući kontekst aplikacije. Ako bean implementira BeanPostProcessor sučelje, Spring poziva postProcessBeforeInitialization() metodu. Ako bean implementira InitializingBean sučelje, Spring poziva after metodu PropertySet(). Ako je bean slučajno deklariran sa inicijalizacijskom metodom, tada se poziva specificirana inicijalizacijska metoda. Ako bean implementira BeanPostProcessor, Spring poziva postProcessAfterInitialization() metodu. Sada se bean može koristiti od strane aplikacije, i ostaje u aplikaciji sve dok se kontekst ne uništi. Ako bean implementira DisposableBean sučelje, Spring poziva destroy() metodu. Slično tome, ako je bean deklariran sa bean-destroy metodom, poziva se specificirana metoda. “ (Walls 2015). Sljedeća tablica prikazuje popis anotacija i opisi koji se koriste prilikom definicije ili testiranja beana u Spring okruženju.

Tablica 1.Opis Anotacija i njihovo značenje, izradio: autor

Anotacija	Značenje
@Component	Identificira klasu kao komponentnu klasu i poslužuje ju kao trag Springu da se bean treba kreirati za tu klasu
@ComponentScan	Anotacija koja dopušta skeniranje komponenata u Springu
@Configuration	Anotacija koja identificira klasu kao konfiguracijsku klasu
@Autowire	Spring automatski otkriva ovisnosti beanova
@Bean	Eksplicitno stvaranje beana

4.2. Aspektno-orijentirano programiranje (AOP)

„Predstavlja programsku paradigmu koja izolira podržavajuće funkcije iz poslovne logike od glavnog programa. Omogućava razvojnom programeru da gradi jezgrenu funkcionalnost nekog sustava bez da bude svjestan o m dodatnim zahtjevima. AOP se koristi u Springu omogućavanje deklarativnih aspekata kao što su transakcije i sigurnost. Aplikacijski objekti izvode poslovnu logiku i nisu odgovorni za ostale brige kao što je vođenje dnevnika, sigurnosti, vođenje audit bilježaka, zaključavanja i upravljanja događajima. AOP predstavlja metodu s kojom se mogu omogućiti middleware-i kao što su sigurnosni servisi te upravitelji transakcija na Spring aplikacijama. Aspekti se mogu dodavati ili micati kada je potrebno bez mjenjanja vlastitog koda. Spring aspekti mogu biti konfiguirani preko vlastitog IoC kontenjera. Spring AOP uključuje aspekte koji se sastoje od savjeta i tzv. Točke rezanja ili pointcuta. “ (Ravi Kant Soni 2017). Sljedeća tablica objašnjava pojmove AOP-a.

Tablica 2. Objašnjenje dijelova AOP-a u Springu, izradio: autor

Dio AOP-a	Objašnjenje
Savjet	Definira što i kada aspekt treba nešto napraviti. Opisuje posao Koji će aspekt obavljati, tj. adresira pitanje kada se treba izvesti koji posao. Postoji 5 savjeta: before, after, after returning, after-throwinng, around
Točka rezanja (pointcut)	Pomažu pri sužavanju točki spajanja savjetodavanih od strane aspekta.
Točka spajanja	Točka u izvršavanju aplikacije gdje aspekt može biti priključen.
Aspekti	Spoj savjeta i točke rezanja. Zajedno definiraju sve što je potrebno znati o aspektu - što radi i gdje i kada to radi.
Predstavljanje	Omogućava dodjeljivanje novih metoda ili atributa u postojeće klase. Može se dodati auditor koji može čuvati stanje o modificiranom objektu, tj. vrijeme modificiranja.
Tkanje	Proces postavljanja aspekta u ciljani objekt za kreiranje novog proxy objekta. Aspekti se pletu u ciljani objekt na specifičnoj točki spajanja, Tkanje se radi tijekom kompajliranja, tijekom vremena otvaranja klase i tijekom izvođenja.

„Spring podržava AOP u četiri obrazca: klasični Spring-proxy bazirani AOP, čisti POJO aspekti, @ASPECTJ anotacijsko vođeni aspekti te injektivan AspectJAspekti koji su dostupni u svim verzijama Springa.“ (Walls 2015). Sljedeća tablica objašnjava savjete.

Tablica 3.Savjeti i objašnjenja, izradio: autor

Savjet	Objašnjenje
@Before	Izvršava se prije poziva savjetodavne metode
@After	Izvršava se tijekom izvođenja savjetodavne metode, neovisno o ishodu
@After-returning	izvršava se nakon što se savjetodavna metoda uspješno izvrši
@After-throwing	Izvršava se kada savjetodavna metoda izbaci iznimku
@Around	Zapakira savjetodavnu metodu pružajući neku funkcionalnost prije i nakon što je savjetodavna metoda pozvana

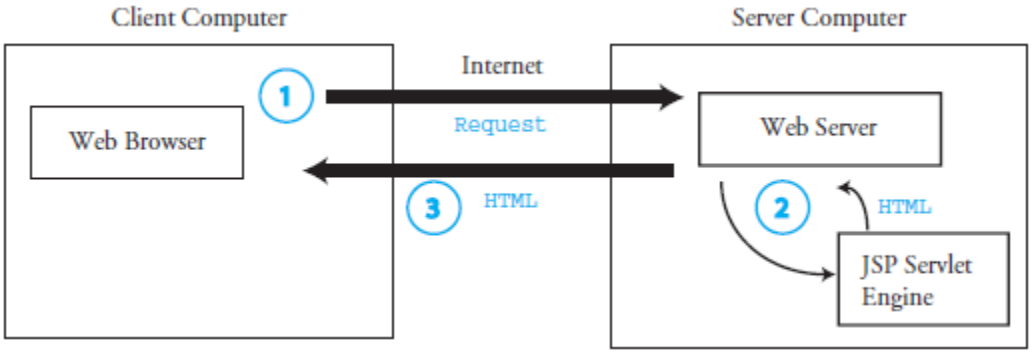
Prije izvršavanja savjeta, postoji funkcija execution() koja sadrži razne regularne izraze i koja sadrži ime klase i metodu koja će se izvršiti.

4.3. Spring MVC

Za izradu Spring web aplikacija koristi se Spring model-prikaz-kontroler, s pomoću kojeg se još može dodati jedan sloj sigurnosti. Kao generator Java stranica može se koristiti standardni HTML jezik plus neki određeni dodatci u obliku jstl biblioteka koji se mogu koristiti u izradi tzv. JSP pogleda. Da bi se JSP mogao prikazati na stranici potreban je određeni servlet koji će aktivirati tu stranicu i omogućiti da umjesto koda piše sadržaj stranice.

4.3.1. Osnovni pojmovi JSP, JSP View, Servlet

„Web aplikacije koje se programiraju pomoću Java programskog jezika uključuju Java Applete, Java Servlete te Java Poslužitelj Pages ili JSP. “ (Savitch 2016). Java applet koristi JavaScript, stoga nije osobito važno pojašnjenje pojma applet. „Servlet se mora kompajlirati prije pokretanja, dok JSP predstavlja određenu dinamičku web stranicu, a servlet predstavlja statičku. JSP ima određene HTML oznake, i kompajlira se na leteći na servlet kada dolazi zahtjev.“ (Savitch 2016). Sljedeća slika prikazuje kako se pokreće JSP stranica preko servleta.



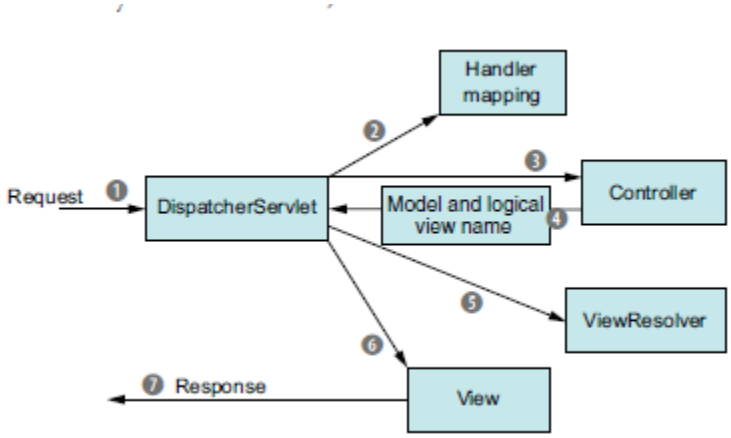
Slika 10. Pokretanje JSP stranice, izvor: (Savitch 2016).

„Klijent pošalje zahtjev poslužitelju za stranicu koja sadržava JSP kod. JSP servlet dinamički kompajlira JSP source kod u Java servlet ako trenutna kompajlirana verzija servleta ne postoji. Servlet pokreće i ispisuje HTML koji se vraća prema web poslužitelju. Zatim web poslužitelj šalje servletov HTML prema klijentu za prikaz u web pregledniku. “ (Savitch 2016). U ovom projektu, koristio se Apache Tomcat poslužitelj

verzija 7.0. Spring olakšava što manje korištenja izvornog koda na stranici. Preko kontrolera se prebacuje na JSP samo vrijednost, i koristi se određena oznaka da se dohvati ta vrijednost. Pri tome, nema nikakve deklaracije koda na stranici, što povećava sigurnost.

4.3.2. MVC

„Za razvojnog programera u Javi, aplikacije bazirane na web stranicama predstavlja glavni fokus. Upravljanje stanjem, tijek podataka i validacija su jedne od glavnih stvari na koje se treba pripaziti. Spring je projektiran posebno da pomogne u sređivanju tih stvari. Baziran je prema model- prikaz-kontroler uzorku. Svaki put kad korisnik klikne na link ili ispuni obrazac koji pošalje, stvara se zahtjev. Zahtjev predstavlja određenog dostavljača, i glavni zadatak mu je prijenos informacija od jednog do drugog mjesta. Od vremena kada ode iz preglednika pa sve dok se ne vrati sa odgovorom, napravi nekoliko zaustavljanja, svaki put ispusti dio informacija i prikuplja više informacija.“ (Walls 2015). Sljedeća slika prikazuje životni ciklus request-a.



Slika 11. Životni ciklus request-a,izvor: (Walls 2015).

„Kada request ode iz preglednika (broj 1) prenosi informaciju koju je korisnik zatražio. To može biti neki URL ili neki podaci koje je korisnik poslao sa nekog obrazca. Prva postaja gdje se request zaustavlja je servlet. Njegov glavni zahtjev je da prenese request u Spring MVC Controller. Kontroler obrađuje zahtjeve. Tipična aplikacija ima više kontrolera pa servlet treba pomoć pri odlučivanju na koji će se kontroler poslati

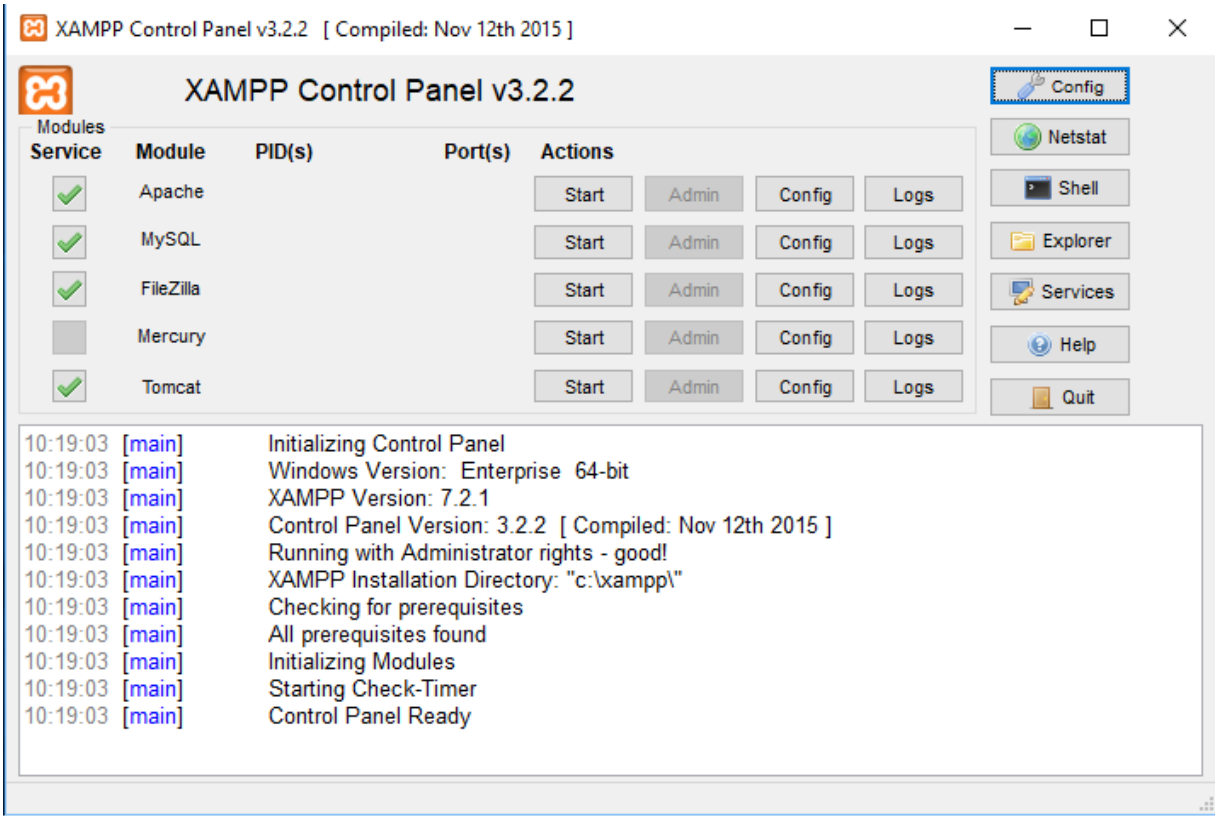
zahtjev. Servletu trebaju određeni putevi da shvati koja je sljedeća postaja zahtjeva. Ti putevi su unaprijed određeni, putem @RequestMappinga, koji put prosljeđuju servletu. Upravitelj mapiranja sadrži određene string vrijednosti koje predstavljaju put, tj. URL kojega nosi zahtjev. Servlet je saznao put, zna o kojem se kontroleru radi. Kada je pravi kontroler izabran, servlet šalje zahtjev prema izabranom kontroleru. Na kontroleru, ispušta se informacija, i čeka se da kontroler obradi tu informaciju. Kada se obradi informacija, zapakira se model podataka, i identificira se ime prikaza, i vraća se prema servletu. Kako bi se spriječilo zapetljavanje, ne prenosi se direktno ime prikaza, već samo prenosi ime koje će se koristiti za ispis vrijednosti. Servlet tada pita resolver, da mapira to prosljeđeno logičko ime prema specifičnoj implementaciji prikaza, koji ne mora biti JSP. Servlet zna rezultat koji će biti na izlazu prikaza te dostavlja model podataka. Zahtjev je obrađen. Korisniku se vraća rezultat.“ (Walls 2015).

5.Opis alata potrebnih za izradu sustava

5.1. Xampp i Spring Tools Suite

„XAMPP je Apache distribucija koja sadržava najćešće korištene alate za razvoj web stranica u jednom paketu. Studentima je idealan alat za razvoj i testiranje aplikacija u PHP-u i u MySql-u. “(Dvorski 2007). Xampp sadrži i ostale komponente, kao što je Tomcat 7.0, na kojem se mogu pokretati Java web stranice te isto tako kao što se mogu lako testirati PHP aplikacije, tako se lako mogu i testirati i Java web aplikacije. „Xampp je besplatni paket dostupan za slobodno skidanje sa interneta i korištenje ra raznorazne web zadatke. Svi Xampp paketi i dodatni dodaci se distribuiraju preko Apache Friends web stranice[5].“ (Dvorski 2007). Xampp ima čitav niz dodataka, čiji se dodaci mogu skinuti sa njihovih stranica. Wordpress, Moodle, Magneto, Joomla su samo neki od njih. Naravno, za neke je dodatke potrebna i registracija i prijava, kako bi se mogli skinuti, poput Joomle,Magneta ili Moodle-a.

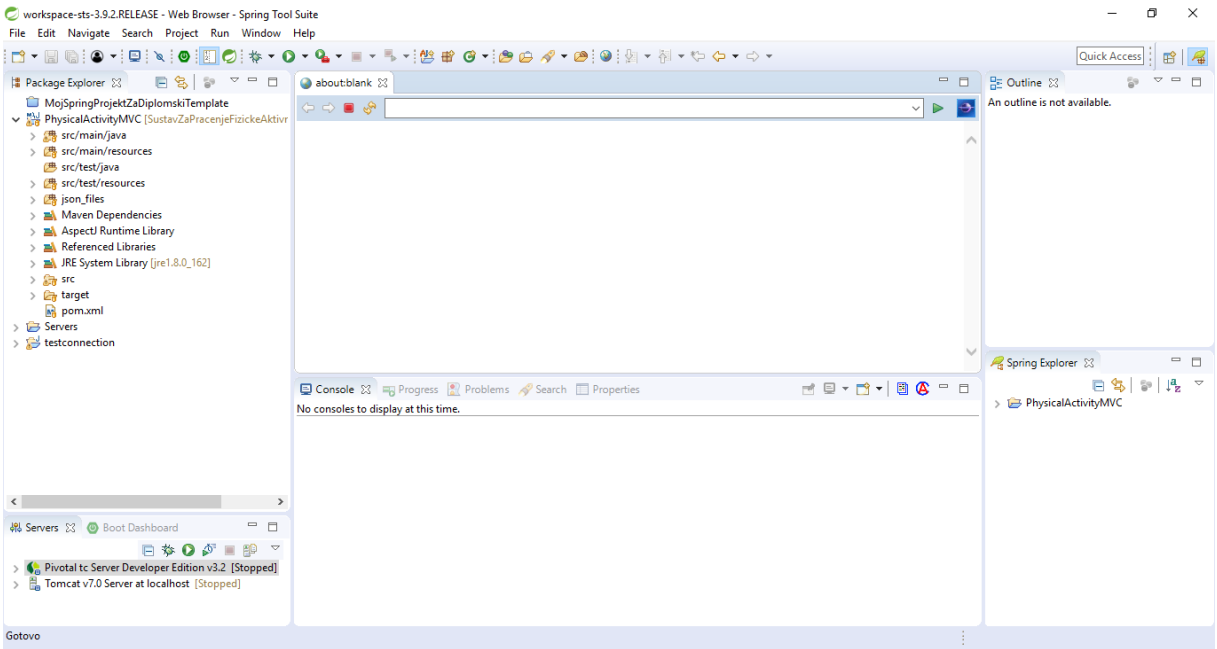
[5] Adresa: http://www.apachefriends.org/



Slika 12. Xampp upravljačka ploča, izradio: autor

Slika 12. prikazuje upravljačku ploču Xampp alata za razvoj Java i PHP web stranica. Pri tome se mogu vidjeti i dnevnici koji izlistavaju aktivnosti pojedinog dijela, ukoliko dođe do problema, lako se pojedini dio može deinstalirati i ponovno instalirati. Također, mogu se otvarati datoteke za konfiguraciju, mogu se pregledavati statistike portova te resursa koje ih koriste. Postoji mogućnost da se za neke stvari koristi i naredbeni redak, u slučaju da se neke poteškoće ne mogu riješiti putem instalacije ili deinstalacije te ukoliko se žele neke dodatne stvari instalirati ili pokrenuti, a pri tome se ne može koristiti *control panel*, već naredbeni redak. U praktičnom radu, korištene su ove komponente: Apache, MySql te Tomcat verzija 7.0, zbog potrebe da se Java web stranica pokrene na nekom poslužitelju, a da se ne stavlja direktno na poslužitelj. Naravno, nije bilo potrebe koristiti Xampp, međutim, zbog korisničkog sučelja koji Xampp ima, pogotovo MySql-a, koje je potrebno za lakše projektiranje baze podataka potrebne za aplikaciju, Xampp je alat savršen za to. Xampp je zapravo virtualizacija poslužitelja, koja oponaša klijenta i poslužitelja, a da pri tome u stavrnosti fizički ne postoji.

Spring Tools Suite predstavlja dodatak za Eclipse. Ovaj dodatak je bilo potrebno koristiti zbog potrebe praktičnog dijela diplomskog rada. Bilo je potrebno izraditi Spring web aplikaciju, koja će primati podatke sa Android mobilnog uređaja,te ju spremati u sql bazu koristeći Spring razvojni okvir. Ovaj dodatak se može pokretati i izvan Eclipse alata pa instaliranje Eclipse alata nije bilo potrebno. Također, nudi testiranje aplikacija izravno u editoru pa čak nije ni potreban Xampp. Jednostavno je potreban Apache Tomcat poslužitelj ili neki sličan poslužitelj preko kojeg se može pokretati Java web stranica. Spring Tools Suite omogućuje i korištenje raznih template projekata, ima ugrađen debugger, a Spring Tools Suite template koji je korišten u praktičnom dijelu je SpringLegacyMVC. Zašto legacy? Spring mvc se može konfiguirati putem XML-a ili jave, a može biti i kombinacija između njih. Razvojni programer može izabrati. Stariji Apache Tomcat poslužitelji , manji od verzije 7.0, koriste XML definicije, stoga je potrebno odabrati legacy template. Novije verzije koriste Java definiciju, stoga se lako može zakomentirati XML konfiguracija i koristiti Java konfiguracija. Izgled Spring Tools Suite editora je prikazan na sljedećoj slici.



Slika 13.Struktura Spring Tools Suite alata, izradio: autor

Spring Tools Suite također ima i ugrađen web preglednik, što znači da nije potrebno otvarati preglednik koji se koristi u općenite svrhe. Ovo je još jedno svojstvo alata koje je jako korisno ukoliko ne želimo otvarati web preglednik izvan razvojnog okvira. Prozor

ispod preglednika, sadrži konzolu, i dodatne prozore koji služe za dnevnik aktivnosti, ispis grešaka na aplikaciji, ukoliko ih ima, popis problema definiranih u svojstvima projekta (greške ili upozorenja). Na lijevoj strani se nalaze popisi polsužitelja,te struktura projekta i njegovih klasa. Na desnoj donjem dijelu, nalazi se Spring explorer, koji pokazuje popis klasa, beanova,te korištenih anotacija u projektu. U alatu se još nalazi i dio za projektiranje web stranica, kojima se mogu vući komponente iz prozora na web pogled, a Spring sadrži i određene Jstl biblioteke koje omogućuju korištenje naredbi za ispis iz raznih struktura podataka, čiji se podaci šalju prema web pregledniku preko tog istog pogleda.

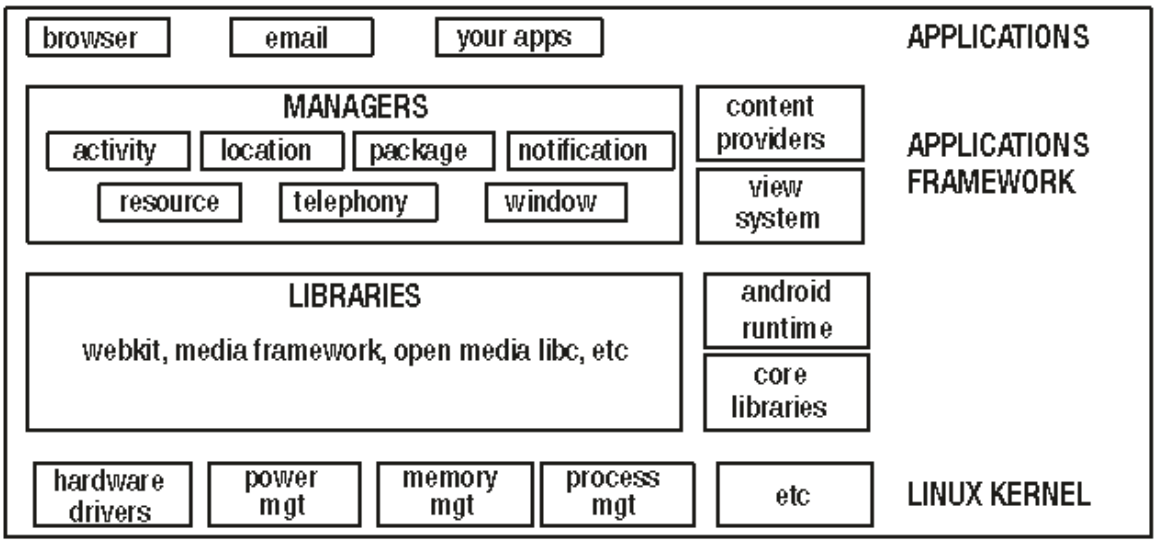
5.2. Android studio

Android studio je alat za razvoj Java mobilnih aplikacija. Sadrži i vlastiti operacijski sustav Android, i uključuje mnogo stvari, poput SDK-a, raznoraznih gotovih biblioteka, emulatora itd. Osim za mobilne aplikacije, Android operacijski sustav se uporebljava i za televiziju, igraće konzole, satove, e-čitače, automobilske sustave itd. „Android ima veoma zanimljivu povijest. Prvi put je oživio u 2003. godini, od tvrtke pod nazivom Android Inc, osnovane od strane Andy Rubin-a. Android Inc je dobivala potporu od strane Google-a, ali je još u to vrijeme nisu imali. 2005. godine, Google je kupio ovu firmu za 50+ milijuna dolara. 2007. godine, Android operacijski sustav je postao open source. Međutim, Android još uvijek nije imao svoju prvu verziju, prva verzija Androida je izašla 2008. godine. 2009 i 2010. godine, stigle su prve verzije kojima su davali imena. Cupcake, Donut, Froyo, eclair i Gingerbread su izašli.“ (Hagos 2018).

2003	Android Inc., founded by Andy Rubin and backed by Google, was born
2005	Google bought Android Inc.
2007	Android was officially open sourced. Google turned over its ownership to the Open Handset Alliance (OHA)
2008	version 1.0 was released
2009	versions 1.1, 1.5 (Cupcake), 1.6 (Donut), and 2.0 (Eclair) were released
2010	versions 2.2 (Froyo) and 2.3 (Gingerbread) were released
2011	3.0 (Honeycomb) and 4.0 (Ice Cream Sandwich) were released
2012	version 4.1 (Jellybean)
2013	version 4.4 (KitKat)
2014	versions 5.0–5.1 (Lollipop); Android became 64-bit
2015	version 6.0 (Marshmallow)
2016	version 7.0-7.1.2 (Nougat)
2017	version 8 (Oreo)

Slika 14. Popis Android verzija, izvor: Hagos,Ted.2018. Learn Android studio 3 Efficient Android App Development.Manilla: Apress, str. 2

„Postoji 7.2 milijarde Android uređaja. Već je prekoračen ukupan broj ljudi koja živi na zemlji. 3 desetljeća je trebalo mobilnim uređajima da dosegnu broj od nule do 7.2 milijarde. 2,617 je broj puta koliko pojedini korisnik dodirne ekran mobitela. Postoji 2 milijarde aktivnih korisnika na mjesečnoj bazi. Najvidljiviji dio Androida, predstavlja operacijski sustav. Operacijski sustav je složena stvar, i većinu vremena, stoji između korisnika i sklopovlja.“ (Hagos 2018). Sljedeća slika pokazuje pojednostavljenu arhitekturu Android operacijskog sustava, zbog toga što ne sadrži sve aplikacije, komponente i bibilioteke koje se nalaze na njemu.

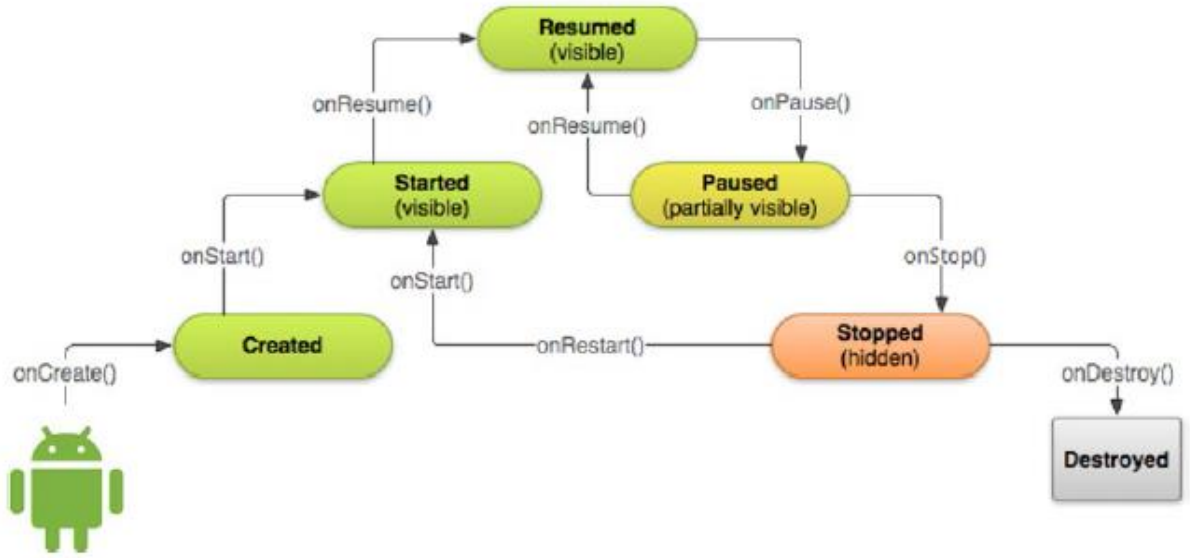


Slika 15. Platformska arhitektura, izvor: (Hagos 2018) ,str.3.

„Najdonji nivo na dijagramu je onaj koji je odgovoran za vezu sa sklopovljem,te raznim servisima poput upraviteljem memorije i izvršiteljem procesa. Dio operacijskog sustava je i Linux. Iznad Linuxa, se nalaze biblioteke kao što su SQLite,OpenGL itd. Ove biblioteke nisu dio Linuxa, i predstavljaju aplikacije niskog nivoa koje su napisane u C++ programskom jeziku. Na istom nivou se nalaze Android runtime (Android klasne biblioteke + Dalvik virtualni stroj), gdje se zapravo Android aplikacije i pokreću. Za razliku od ostalih Java programa, Android izvršne datoteke nemaju ekstenziju .class, već .dex datoteke. Dex datoteke se ne pokreću na tipičnom Java virtualnom stroju (JVM), već se pokreću na virtualnom stroju koji je optimiziran za niskoenergetske prijenosne uređaje. Kompilacijski ciklus ide ovako: .Java datoteke(izvorni kodovi) > Java kompajler (.class) > dex kompajler (.dex) > paket aplikacije (.apk). Sljedeći nivo

je sloj aplikacijskog razvojnog okruženja. Nalazi se na vrhu između biblioteka niskog nivoa i Android vremena izvođenja, jer mu je potrebno oboje za normalno funkcioniranje. Ovaj nivo sadrži sve potrebne biblioteke koje su potrebne za razvoj aplikacija i s kojim developeri imaju interakciju. Na najgornjem sloju nalazi se aplikacijski sloj. Ovdje se aplikacije izvršavaju, dakle sve aplikacije koje napišemo i one aplikacije koje su došle zajedno sa operacijskim sustavom. Aplikacije koje su došle sa operacijskim sustavom nemaju nikakve drugačije privilegije ili ovlasti, od onih koji se pišu ili razvijaju. Ako nam se recimo ne sviđa email aplikacija koja je trenutno na telefonu, možemo napraviti svoju i koristiti tu svoju aplikaciju.“ (Hagos 2018). Android studio ima svoj Java editor, mogućnost projektiranja aplikacije putem XML-a, moguće je komponentu napisati izravno u XML-u ili vući komponente na XML pa poslije uređivati određene identifikatore koje se kasnije inicijaliziraju u Java programskom kodu i određuje se ponašanje u emulatoru. Emulator oponaša stvarno izvođenje aplikacije na stvarnom uređaju, iako ima svoje nedostatke. Android ima neke svoje biblioteke, poput lokacije, kojima je potrebna internet veza kako bi recimo dohvaćali adresu trenutne lokacije ili bi mjerile brzinu kretanja. Za mapiranje puta, koristi se Google API, gdje je potrebno imati generirani ključ koji se može generirati na Google API stranici. Taj ključ je potrebno umetnuti na za to predviđena mjesta u Androidu,te mu dopustiti određene dozvole u aplikaciji, putem Android manifesta. U Android manifestu, registriraju se aktivnosti aplikacije. Aktivnosti se pokreću unutar emulatora i imaju svoj životni ciklus. Implementacijom životnog ciklusa, dobiva se željeno ponašanje aplikacije. „Mobilne aplikacije nisu samo aplikacije za računala koji se pokreću na manjim ekranima. Ne koristimo mobilne aplikacije na isti način kao i one na računalima. Kada korisimo aplikacije na računalu, one u međuvremenu ostaju otvorene,te aktivne tijekom nekog vremena. Mobilne aplikacije imaju nešto kraći životni ciklus. Najčešće ih izvadimo iz džepova, nešto napravimo na brzinu i zatim ih spremimo. Nekada se čak dogode i neki prekidi u aplikaciji od drugih aplikacija, poput telefonskog poziva,te bi se zbog toga izvorna aplikacija prekinula. Kao developer, developer ne kontroliran životni ciklus aplikacije, već korisnik. Ne može se preedvidjeti da će netko nazvati korisnika i u tom trenutku će korisnik biti prekinut u korištenju te aplikacije. Postoji mogućnost da će sustav izaći iz aplikacije, a da pritom podaci ne budu spremljeni. Potrebno je razmisliti kada bi se to moglo dogoditi. “(Hagos 2018). Iduća slika predstavlja životni ciklus aktivnosti. Osim od aktivnosti, Android se sastoji od još h koomponenata, a te komponente su prikazane na slici 16. Neke od važnijih

komponenata su aktivnosti, servisi, prijemnik za emitiranje te poslužitelj sadržaja. Aktivnost drži, prikazuje elemente korisničkog sučelja. Primjer: pregledavanje pojedinog email-a, pisanje bilješki. Servisi pokreću procese u pozadini. Primjer: skidanje velike slike sa interneta. Prijemnici za emitiranje prohvaćaju poruke od strane drugih aplikacija ili Android sustava. Primjer: prikaz poruke upozoranja kada baterija ima manje od 10% života. Poslužitelj sadržaja sprema i prima podatke, kao baza podataka. Primjer: kontaktira ili adresira telefonski imenik na telefonu, aplikacija može dodavati nove kontakte na telefon.



Slika 16.Životni ciklus aktivnosti, izvor: (Hagos 2018),str.121.

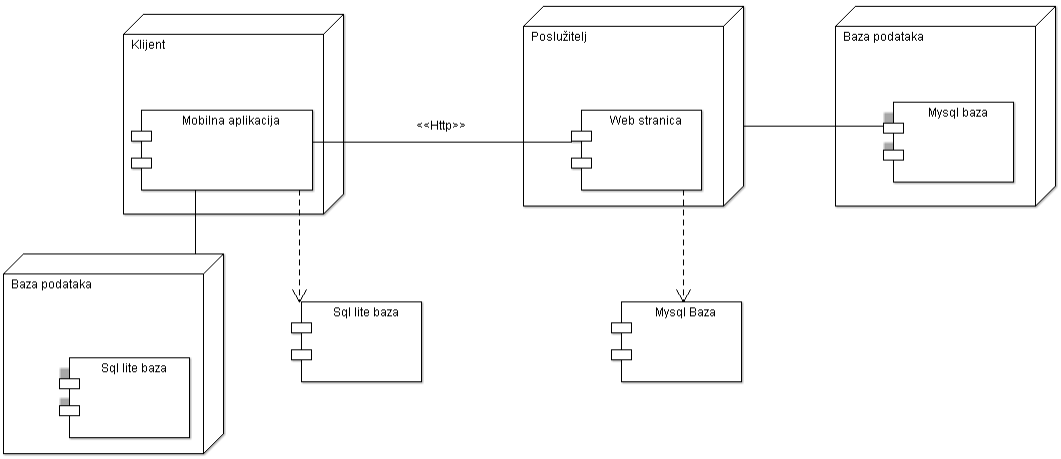
U životnom ciklusu aktivnosti, postoje metode koje se nadjačavaju, a s time i pozivaju tijekom vremena izvođenja, kada korisnik koristi aplikaciju. Te metode su onCreate, onRestart, onStart, onResume, onPause, onStop i onDestroy. „Metoda onCreate se poziva kada se aktivnost prvi put napravi. Najčešće se stavljaju inicijalizacijske varijable. Metoda onRestart se poziva kada se aktivnost stopira i resetira ponovno. Sljedi je metoda onStart. Metoda onStart se poziva kada aktivnost postane vidljiva korisniku. Metoda onResume se poziva kada je aktivnost spremna za interakciju sa korisnikom, na ovoj točki aktinvost je na vrhu stoga aktivnosti,te zauzima cijeli zaslon. Metoda onPause se poziva kada se aktivnost sprema da postane pozadinski proces. Ovo se može dogoditi najčešće kada druga aktivnost bude u fokusu. Metoda onStop

se poziva kada aktivnost više nije vidljiva korisniku. I zadnja metoda, onDestroy, poziva se kada je aktivnost uništena. Da bi se aplikacija vratila, treba se stvoriti ponovno.“(Hagos 2018). Implementacijom ovih metoda, može se odrediti ponašanje čitave aplikacije i kao što je već spomenuto, potrebno je razmisliti o prekidima unutar aplikacije. U tome je i bit životnog ciklusa, da pomogne ukoliko netko nazove korisnika dok koristi tu aplikaciju, a da pri tome njegovi podaci što se nalaze na toj aplikaciji budu sačuvani i da se može vratiti u aplikaciju kad korisnik želi. Naravno, nije potrebno implenentirati te funkcije, sve se može napraviti u onCreate. Druga opcija su fragmenti.Fragmenti isto imaju svoj životni ciklus. Koriste se za mape, izbornike itd. Služe da se postavljaju elementi u aktivnost tijekom vremena izvođenja aplikacije.

6.Dokumentacija sustava

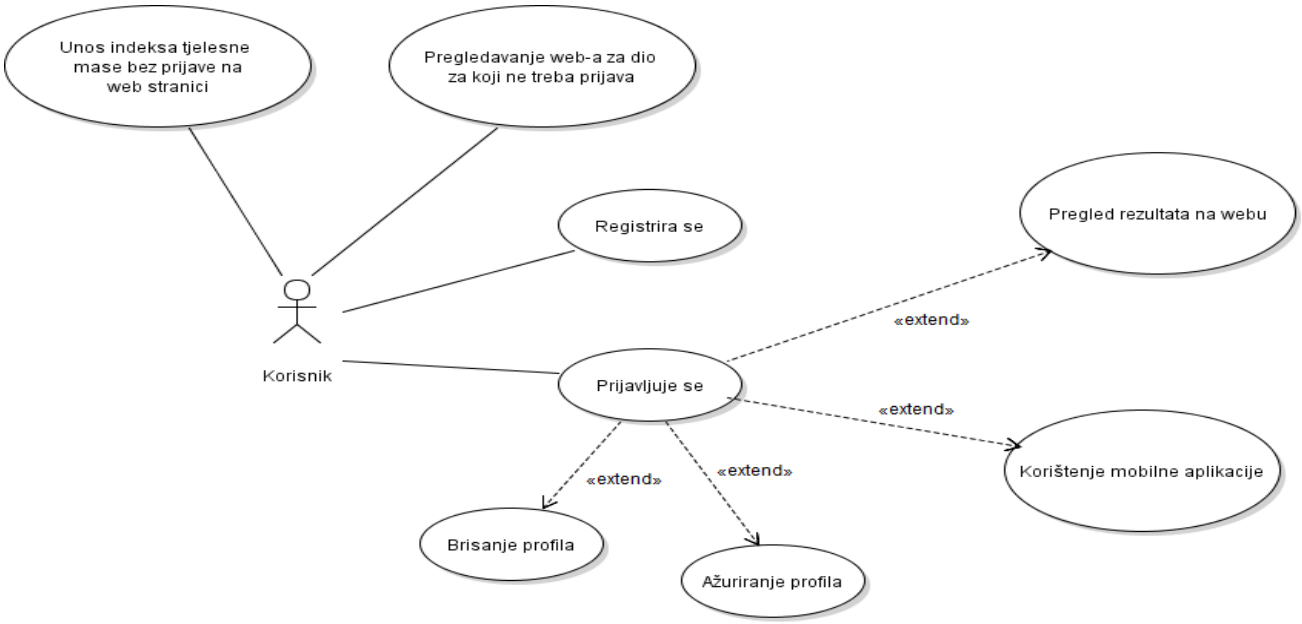
6.1. Opis sustava

Sustav se sastoji od klijenta i poslužitelja. Na klijentu se nalazi mobilna aplikacija, koja sadrži i vlastitu lokalnu bazu SQLite. Na poslužitelju se nalazi web aplikacija, koja se spaja na MySql bazu podataka. Mobilna aplikacija i web aplikacija zajedno komuniciraju preko HTTP-a, šaljući i primajući međusobno podatke, od poslužitelja do klijenta i obrnuto. Sljedeći dijagram razmještaja prikazuje arhitekturu sustava. Web aplikacija kada zaprimi podatke od klijenta, podatke sprema u MySql bazu podataka. Mobilna aplikacija, kada zaprimi podatke od web aplikacije, podatke pohranjuje u lokalnu SQLite bazu podataka. Sljedeća slika prikazuje dijagram razmještaja koji opisuje grafički ovaj sustav. Mobilna i web aplikacija komuniciraju putem posebne Spring biblioteke koja se zove Spring REST Template. Ta biblioteka sadrži metode koje komuniciraju prema web poslužitelju. Metode su delete, exchange, execute, getForEntity, getForObject, headForHeaders, optionsForAllow, postForEntity, postForLocation, postForObject, put. U praktičnom radu su korištene metode exchange i getForObject. Metoda exchange izvršava URL i pri tome hvata odgovor sa web poslužitelja preko tijela odgovora. Ostvaruje se dvostruka komunikacija od klijenta prema poslužitelju i obrnuto. Metoda getForObject ima samo jednosmjernu komunikaciju, glavni joj je zadatak da dohvati podatke sa poslužitelja. Kako bi se ostvarila komunikacija, potrebni su određeni converteri kako bi REST prepoznao o kojem se tipu podataka radi.



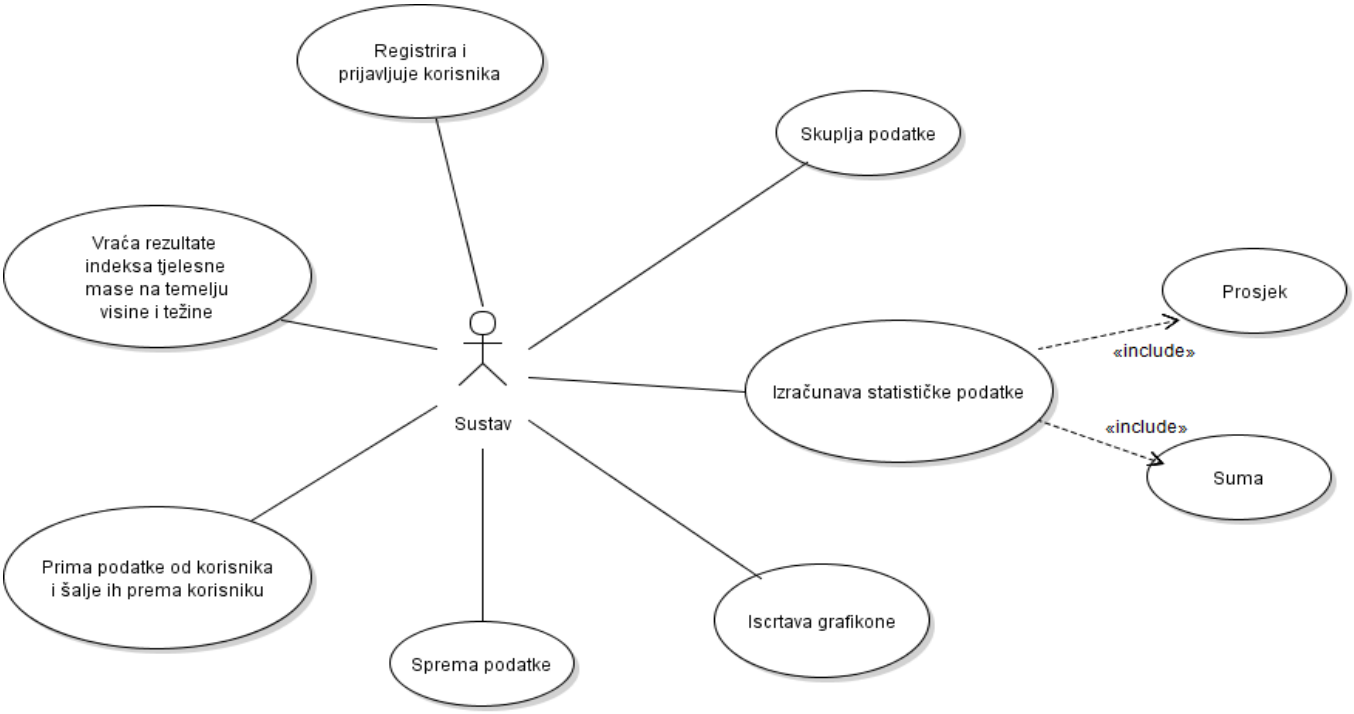
Slika 17. Dijagram razmještaja, izradio: autor

Mobilna aplikacija je napravljena u Android studiu a web aplikacija u Spring razvojnom okviru koristeći dodatak za Eclipse Spring Tools Suitete arhitekturu model-prikaz-kontroler.Verzija Jave koja je korištena za izradu projekta je 1.8, a dodatni alati su Xampp koji uključuje MySql bazu te Apache Tomcat 7.0. Sustav omogućuje registraciju korisnika na web stranici, prijavu korisnika na web stranicu, prijava korisnika na web aplikaciju, prihvaćanje podataka sa poslužitelja, slanje podataka na poslužitelj, prikupljanje podataka pomoću senzora u Androidu kao što su broj koraka, brzina, datum, adresa na kojoj se korisnik tek nalazi, kalkulacije prosječne brzine, ukupne udaljenosti, prosječne prijeđene udaljenosti, prikaz povijesti aplikacije na web stranici, koje uključuje pregledavanje liste podataka prema nekom datumu, broj potrošenih kalorija. Naravno, web aplikacija omogućuje i prikaz grafikona, kao i tablični prikaz povijesti. Mobilna aplikacija služi čisto za prikupljanje jednostavnih podataka,te slanje tih podataka na web gdje se ti podaci mogu pregledavati. Korisnik, naravno, ima i opciju ažuriranja profila,te brisanja profila ako to želi. U slučaju brisanja, svi podaci korisnika će nestati. To je riješeno u bazi dodavanjem stranog ključa prema glavnoj tablici i osiguravanje da ako se profil izbriše, brišu se i njegovi podaci. Korisnikove opcije, što on može napraviti na aplikaciji su prikazane u use case dijagramu, a struktura baze na er dijagramu.



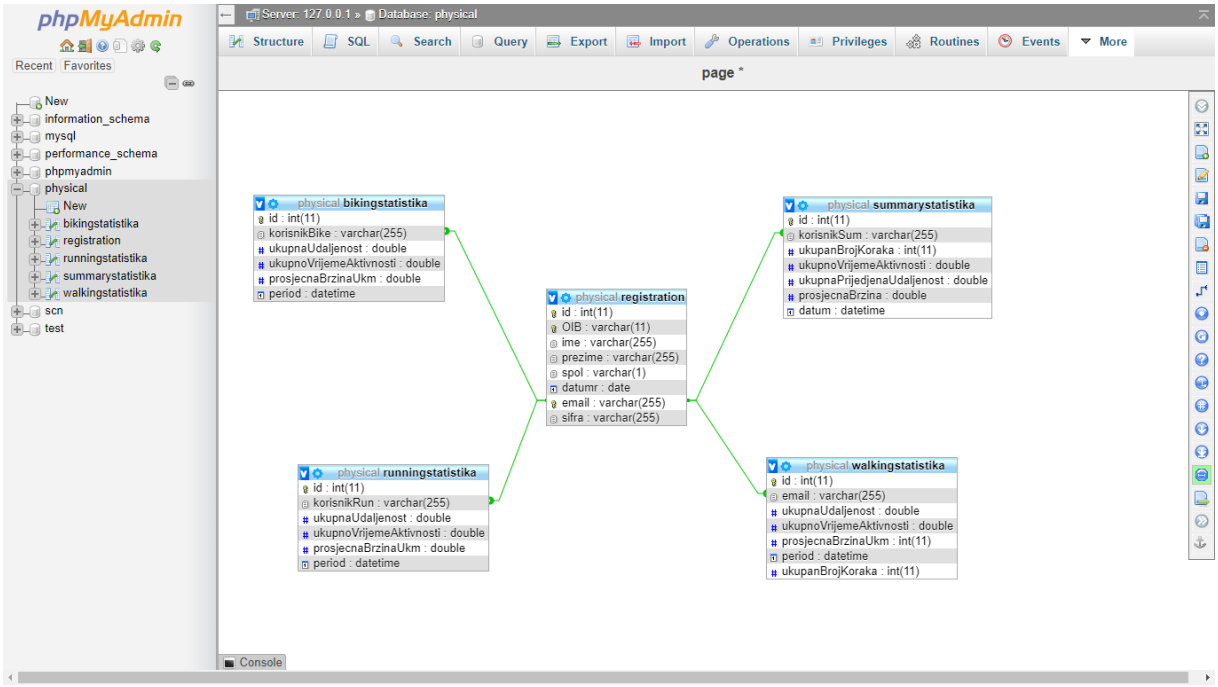
Slika 18. Use case dijagram korisnika, izradio: autor

Korisnik ima neke ovlasti bez da se registira na sustav, kao što je recimo unos tjelsnog indeksa i vraćanja rezultata na temelju unešene visine i težine. Može pregledavati i određeni dio web stranice za koje nije potrebna prijava i registracija, poput početne stranice i indeksa tjelesne mase. Za sve ostalo, potrebna je registracija i prijava u sustav. Korisničko ime, koje je u stvari email adresa, služi za izlistavanje rezultata po pojedinom korisniku, a da nema prijave i registracije, izlistavali bi se svi rezultati. Također, korisnik može i ažurirati dijelove profila, sve osim datuma rođenja i spola. Za promijenu toga, potrebna je nova registracija. Korisnik ima mogućnost i brisanja svog profila, i kad se njegov profil izbriše iz sustava, brišu se i svi njegovi rezultati. To je korisno, jer sustavu ne treba milijun rezultata korisnika koji ne postoji da zauzima prostor kad se taj prostor može ponovno iskoristiti.



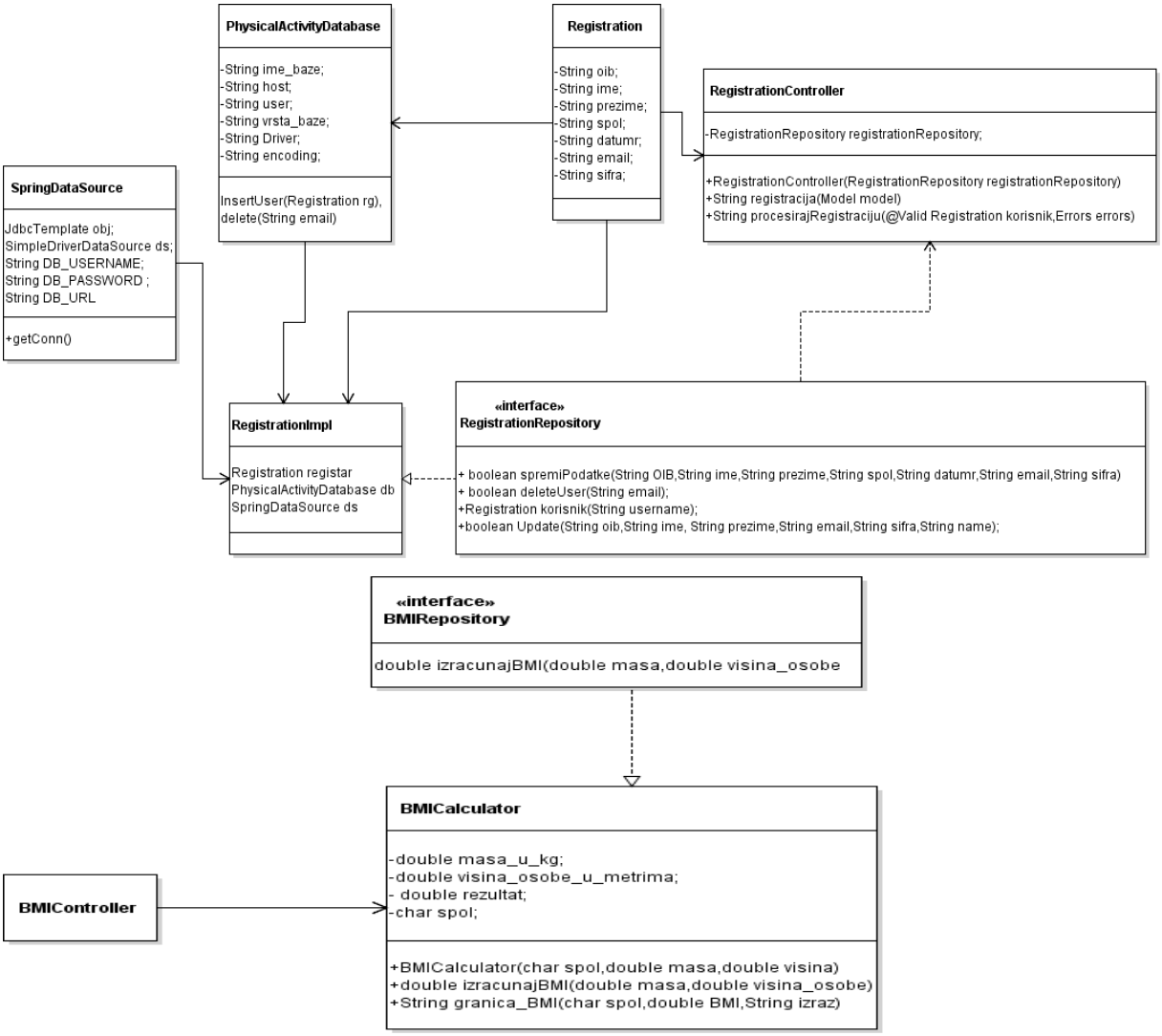
Slika 19. Scenarij sustava, izradio: autor

Dakle, prije nego što korisnik može koristiti mobilnu aplikaciju, prvo se mora prijaviti na web putem registracijskog obrasca. Zatim, nakon što se registrira, u bazu se spremaju njegovi podaci. Nakon toga, korisnik se može prijaviti na web stranicu da pregleda svoj profil, kojeg može uređivati ili obrisati te naravno, pregledavati svoje rezultate na web stranici. Kod mobilne aplikacije, korisnik se prvo prijavljuje, ispunjava formu za prijavu i čeka da se podaci sa weba spreme u lokalnu bazu na Androidu. Nakon toga, korisnik može koristiti aplikaciju, odabrati na listi hodanje trčanje ili bicikliranje i nakon što odabere aktivnost, njegovi se podaci mogu početi pratiti. Nakon što korisnik završi sa trenutnom aktivnošću, vjerojatno će kliknuti na rezultate. Na rezultatima su prikazani rezultati prema pojedinim aktivnostima,te sveukupni rezultati. Sljedeća slika prikazuje tablice i njihove relacije u phpMyAdminu. Naziv baze na webu je physical. Baza podataka je uređena na način da postoji jedna centralna tablica registracije korisnika, koja sadrži korisničke podatke i prema toj tablici se na web stranici ispisuju podaci od korisnika i postoji mogućnost uređivanja podataka. Korisnik može i obrisati svoj profil, a pri tome se brišu i svi njegovi podaci. Tablice koje imaju vezu prema centralnoj tablici, imaju indeks i na tom indeksu je dodan strani ključ i označen je kao CASCADE, što znači da se svi podaci brišu ukoliko se obriše korisnički profil.



Slika 20. Struktura baze, izradio: autor

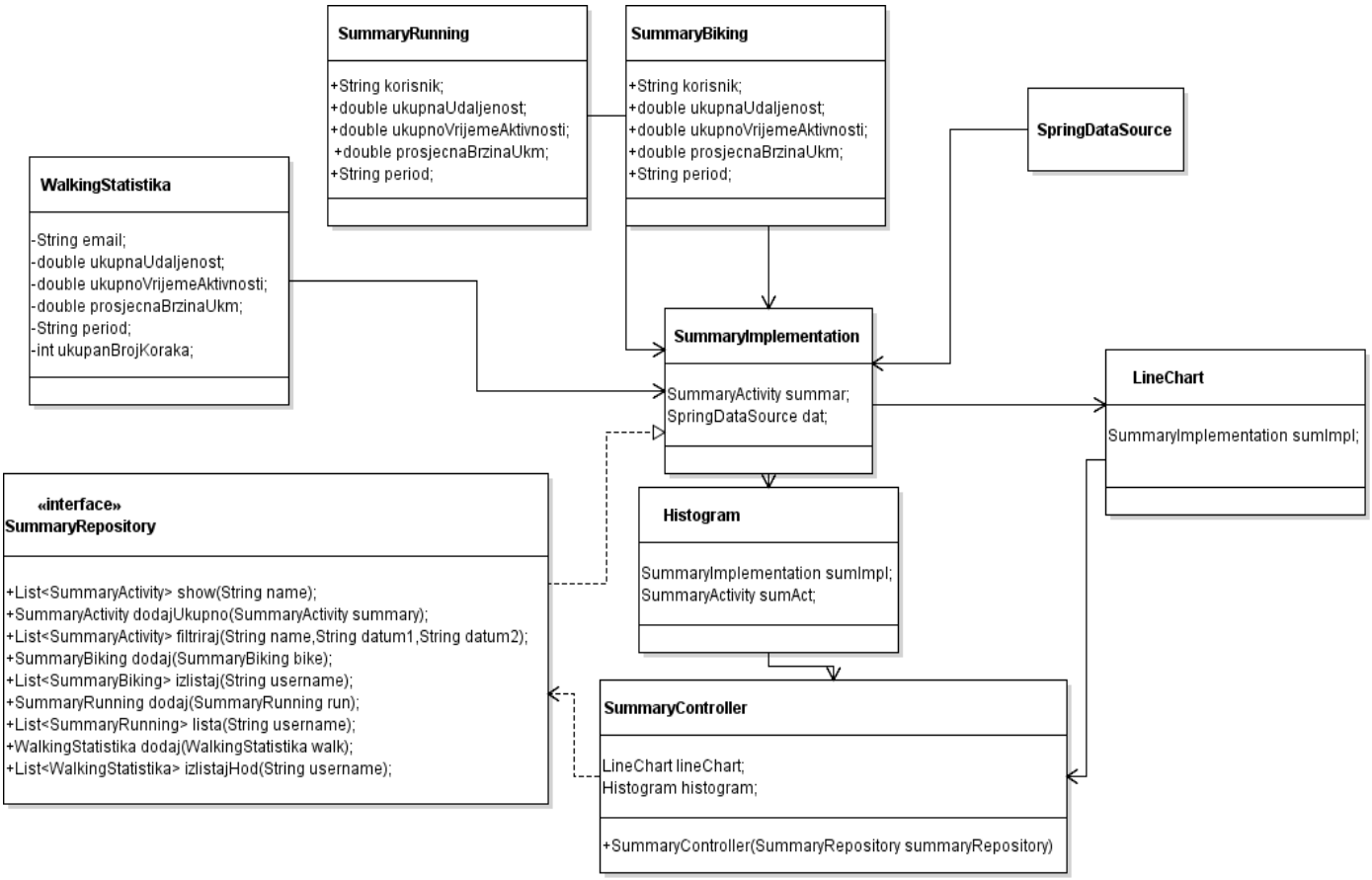
Centralna tablica baze physical je registration, koja služi za spremanje korisničkih podataka. Prvotna ideja je bila da se preko OIB-a spaja sve, no zbog jednostavnosti i smanjenja upita na bazu, strani ključ koji se koristi je email, jer predstavlja korisničko ime preko kojeg se korisnik prijavljuje na sustav. U realnom sustavu, šifra se ne bi trebala spremati u bazu, ili bi trebala biti šifrirana. Zbog jednostavnosti sustava, šifra će se u ovom sustavu spremati u obliku originala. Ako se sustav bude stavljao u funkciju, svaki unos šifre će se šifrirati ugrađenim kriptografskim algoritmom. Tablice za prikupljanje statistike će dobivati podatke o ukupnim rezultatima po pojedinoj aktivnosti, da bi na kraju, konačna tablica sa statistikama primila konačne podatke o aktivnosti tijekom tog jednog perioda kroz koji se koristi mobilna aplikacija. Modeli podataka koji se nalaze u bazi slični su i u aplikaciji, no ipak u pojedinim dijelovima postoji razlika, jer je deklarirana drugačije varijabla nego u bazi, primjerice za datum. U modelu na aplikaciji, datum je string, koji je na kraju u obliku datetime u bazi. S obzirom da je datum i vrijeme koji se ubacuju u bazu ispravan. Kad bi se izvodile računske operacije u bazi, bilo bi bitno onda deklairati i u modelu aplikacije kao datum, što u ovom sustavu za to nema potrebe. Sljedeći dijagrami klase opisuju modele podataka i njihove metode te tipove podataka.



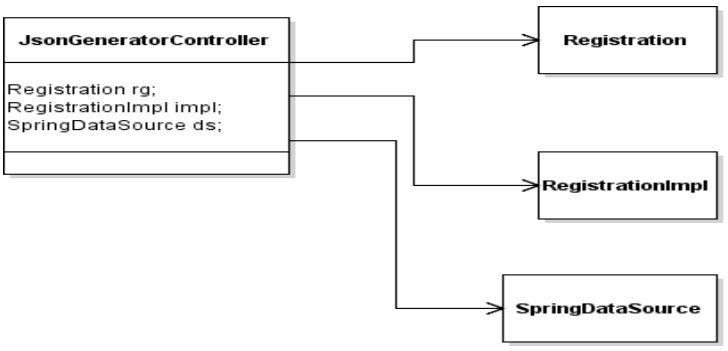
Slika 21. Dijagram klase web stranice broj 1, izradio: autor

Dijagram predstavlja prvi dio klasa koje se nalaze na webu. Zbog jednostavnosti i bolje preglednosti, konstruktore je bolje izostaviti kao i pripadajuće gettere i settere. Model registracija je povezana sa `physical activity database` klasom i `registrationImpl`, kao i sa kontrolerom. Physical activity database ima vezu prema `registrationimpl`.Druga verzija baze, tj. Spring baza je povezana sa `registration impl`.`RegistrationRepository` se implementira u `registrationImpl`, o čemu ovisi i kontroler. Korisnik se registrira na sljedeći način. Upisuje svoje podatke u obrazac za registraciju. Zatim se ti podaci validiraju, a nakon validacije tih podataka, ukoliko je sve u redu, korisnik se vraća na početnu stranicu odakle može dalje prema želji otići na druge stranice koje zahtjevaju prijavu. Korisnik može koristiti indeks tjelesne mase bez

registracije, jer se podaci ne spremaju. Indeks tjelesne mase služi korisniku da si sam može odrediti tempo rekreacije. Indeks tjelesne mase nije prikazan na dijagramu, ali također ima i svoj kontroler. Prvi dio kontrolera, otvara stranicu. Drugi dio preko POST metode zahtijeva unešene podatke, preko @RequestParam anotacije, i pri tome prima određene parametre. Te parametre prosljedi klasi koja sadrži i metodu za izračun indeksa i ispis rezultata. Na kraju korisniku vraća rezultat na ekran.

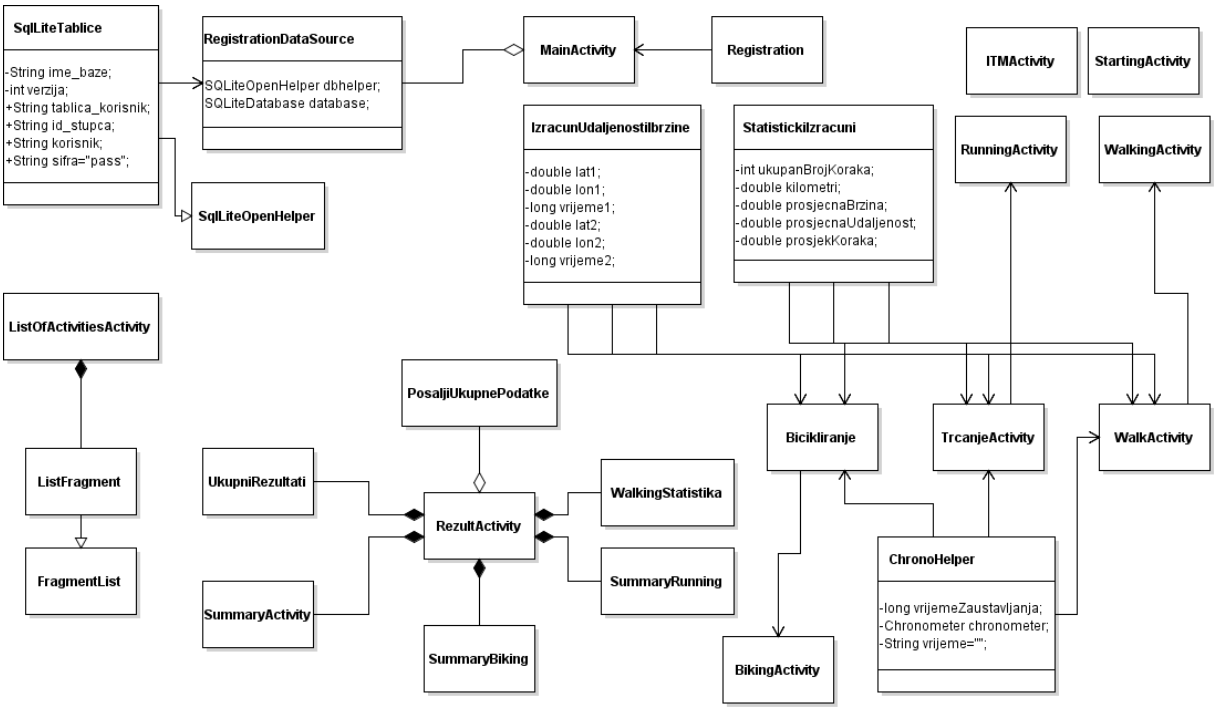


Slika 22. Dijagram klase web stranice broj 2, izradio: autor



Slika 23. Dijagram klase web stranice broj 3, izradio: autor

`SummaryController` ima svoj repositorij podataka koji sadrži operacije poput dodavanja podataka u bazu te ispisivanje podataka iz baze i vraćanje lista podataka koji se prikazuju na prikazu ili se preko njih iscrtavaju grafikoni i ti grafikoni se prikazuju na web stranici. `SummaryImplementation` implementira metode u repositoriju. Može se reći da kontroler ovisi o repositoriju, jer se preko konstruktora, poziva objekt i preko tog objekta se spremaju podaci. `JsonGeneratorController` predstavlja REST kontroler koji je također povezan sa ovim klasama, taj kontroler prihvaća podatke sa mobilne aplikacije i zatim se zove implementacija repositorija kako bi se ti podaci spremili u bazu. `SummaryController` dohvaća podatke iz liste i iscrtava grafikone te ih ispisuje na ekranu. Zadatak drugog kontrolera je primanje podataka sa mobilne aplikacije i spremanje u poslužiteljsku bazu podataka. Sljedeći dijagram klasa prikazuje strukturu mobilne aplikacije, sa pomoćnim i glavnim klasama. Zbog bolje preglednosti, varijable koje se nalaze u aktivnostima se ne prikazuju.



Slika 24.Dijagram klase mobilne aplikacije, izradio: autor

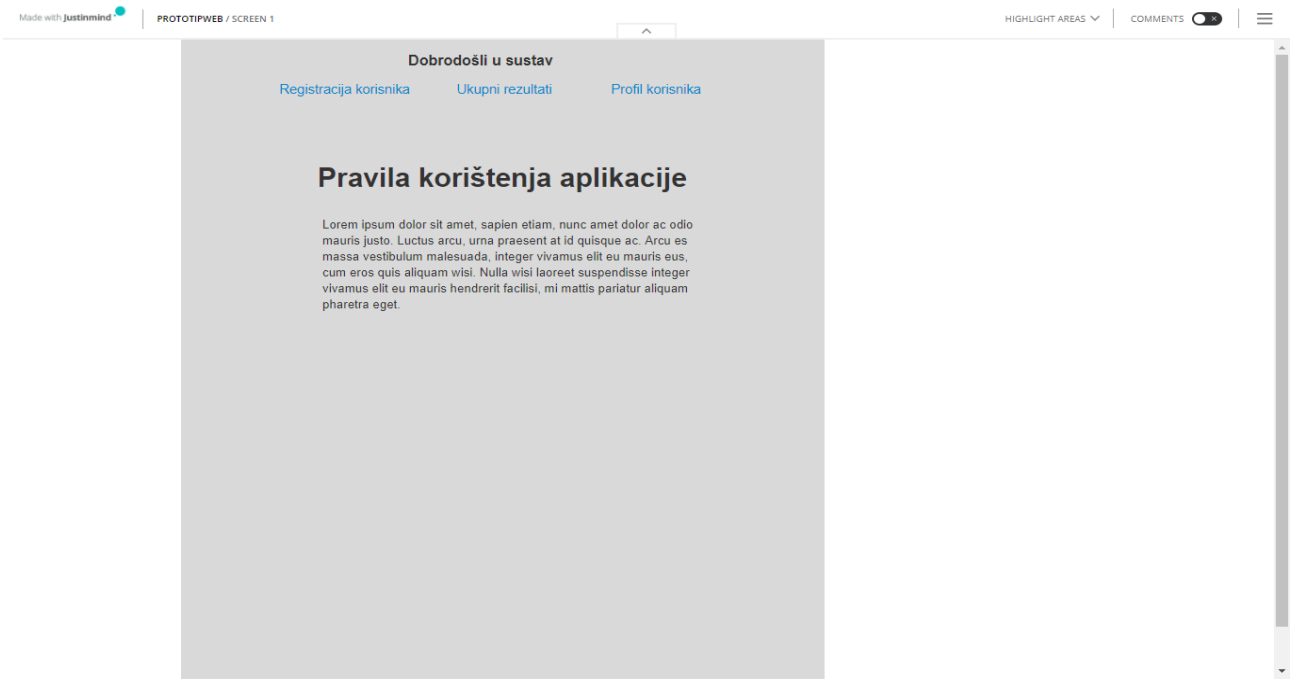
`MainActivity` služi za prijavu korisnika u mobilnu aplikaciju. Povezan je sa modelom podataka za registraciju i mehanizmom agregacije se spaja na bazu podataka gdje se spremaju isti podaci u tu bazu. Kasnije se vrši autentifikacija i

autorizacija kako bi korisnik mogao koristiti tu aplikaciju. `StartingActivity` i `ITMActivity` su samostalne klase, prva služi za pokretanje druge aktivnosti i sadrži i uvjete korištenja aplikacije, dok se druga spaja na web stranicu, preko web pogled Android widgeta,te se na taj način taj dio ponaša kao i web dio. Klasa `ListOfActivitiesActivity` predstavlja glavni izbornik, tj. listu gdje korisnik može kliknuti i ići na sljedeće aktivnosti. Za stvaranje izbornika koristi se list fragment. Aktivnosti gdje se prikupljaju podaci su povezane sa svojim modelima,te sa klasama poput `ChronoHelpera`, `StatistickiIzracuna` te `IzracunaUdaljenostiIBrzine`. To su pomoćne klase, koje izračunavaju udaljenost, vrijeme kronometra koje služi kao mjerno vrijeme aktivnosti,teza izračun ukupnih rezultata po aktivnostima. Rezultati sadrže kompoziciju prema ukupnim rezultatima, sadrži i objekte koje sadrže ukupne rezultate po pojedinim aktivnostima i na kraju, klasu za slanje ukupnih podataka, koja šalje sveukupne podatke, zajedno sa pojedinim podacima izračunatim po aktivnostima, na web poslužitelj i trajno ih pohranjuje u bazu podataka na web poslužitelj.

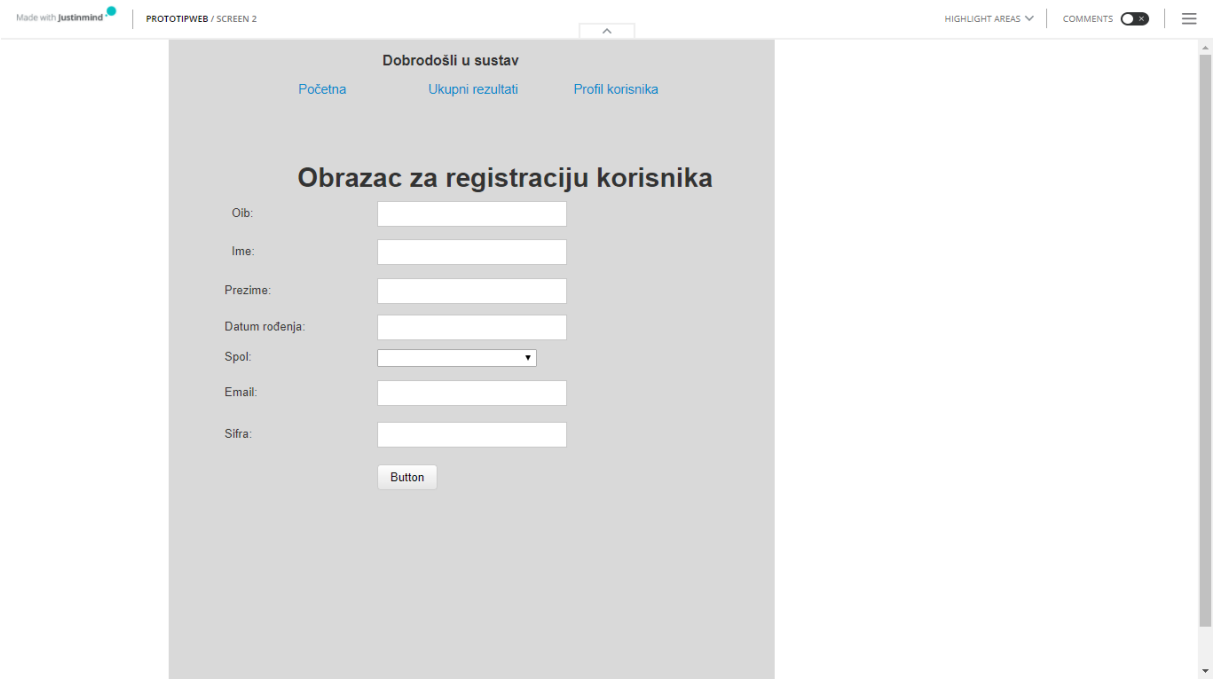
6.2. Prototipovi sučelja

Izrada prototipa sučelja je korisna kada je projekt u početku osmišljavanja ili dodavanja nove funkcionalnosti ili novog dijela korisničkog sučelja. Pri razvoju novog proizvoda, ne moramo sve dijelove iskoristiti od prototipa. Moguće je iskoristiti samo neke od dijelova. Tipičan primjer je automobilska industrija. Automobilska firma izradi jedan prototip automobila, koji sadrži sve ideje kakav bi novi model automobila iste firme trebao biti. Uzmu dio od njega i ugrade ga u završni proizvod. Slično je i u informatičkoj industriji. Napravi se prvo nekekav konceptualni model, pogotovo kada korisnik ugovara recimo izgled web stranice ili mobilne aplikacije, zatim se krene sa implementacijsom i ovisno o mogućnosti, naprave se neki dijelovi koji su slični prototipu ili su bolji nego što je na prototipu. Po promijeni zahtjeva, najbolje je onda prvo načiniti izmjenu na prototipu pa da se vidi ponašanje kako bi to trebalo biti, a tek onda krenuti sa implementacijom istoga. Ako je neki zahtijev prevelik, normalno da će se taj dio odbaciti i neće se koristiti, već će se naći zamijena.

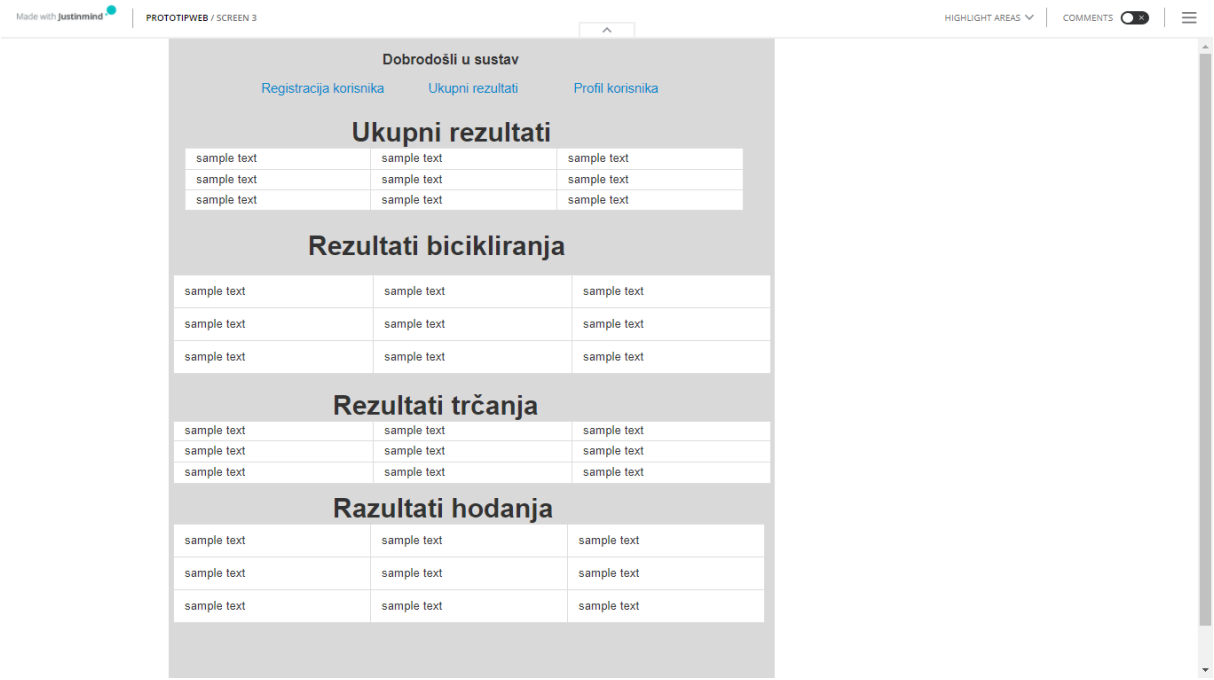
U ovom radu, izrađen je prototip web stranice i mobilne aplikacije. Prototip je prilično jednostavan, a mobilna aplikacija i web aplikacija se i ne razlikuju puno od prototipa sučelja. Alat koji se koristio za izradu prototipa je Justinmind Prototyper. Alat je poprilično jednostavan. Izabire se prvo prototip koji se želi izrađivati, poput web stranice, prototip Iphone aplikacije, Android aplikacije te se taj isti i otvori. Alat sadrži i neke primjere pa ukoliko se ne želi skroz otpočetka graditi prototip, moguće je samo promijeniti malo izgled i prototip bude veoma brzo gotov. Elementi se vuku sa alatne trake i na kraju se dobije element sličan elementu na stranici. Alat također nudi i simuliranje ponašanja na webu. Postoji besplatna i komercijalna verzija alata, a besplatna sadrži neka ograničenja, za razliku od komercijalne. Sljedeće slike prikazuju prototipove sučelja web i mobilne aplikacije sustava za praćenje fizičke aktivnosti. Prototip sadržava stranice za registraciju, početnu stranicu te stranicu za rezultate i naravno, stranica profila korisnika. Prototip sučelja mobilne aplikacije sadrži početnu aktivnost, za prijavu, za izbornik te jednu aktivnost za prikupljanje rezultata. Nije potrebno prikazati potpun prototip u radu, već samo neke od najvažnijih dijelova. Cijelokupan prototip je dostupan skupa sa izvornim kodom projekta.



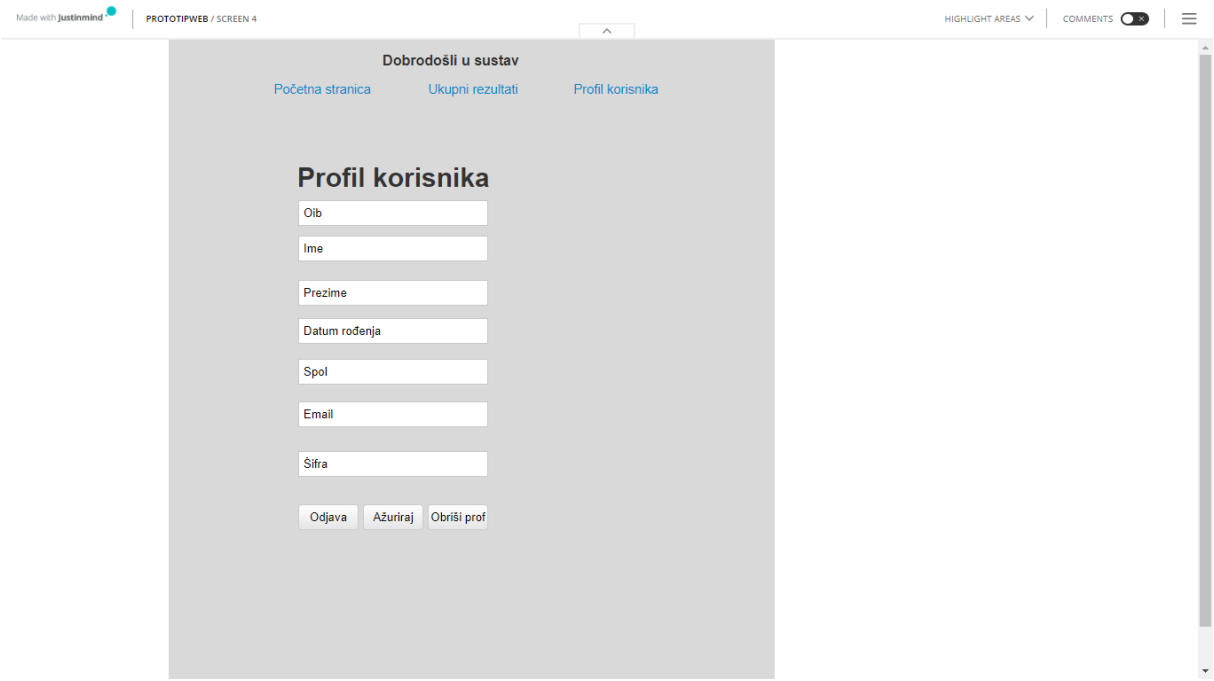
Slika 25. Prototip sučelja početne web stranice, izradio: autor



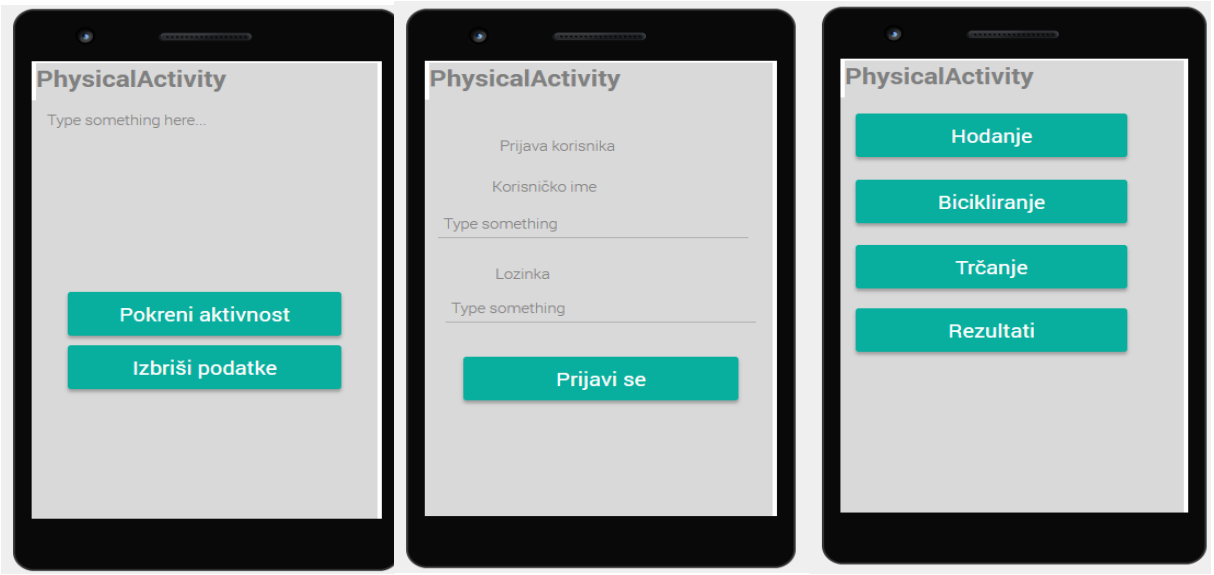
Slika 26. Sučelje za registraciju, izradio: autor



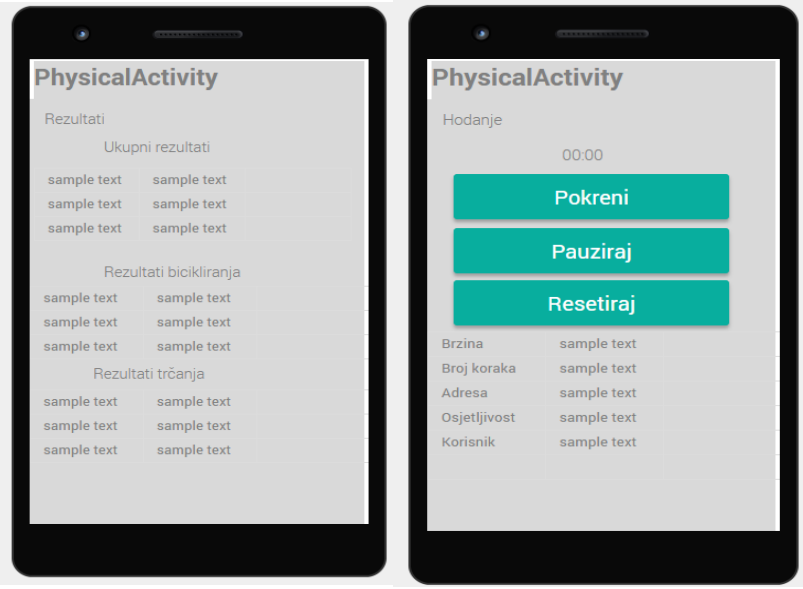
Slika 27. Sučelje za rezultate, izradio: autor



Slika 28. Sučelje profila, izradio: autor



Slika 29. Prototip sučelja mobilne 1, izradio: autor



Slika 30. Prototip sučelja mobilne aplikacije 2, izradio: autor

S pomoću ovog prototipa dalje se može razvijati sustav, dodavati novi elementi, nove funkcionalnosti, a završni proizvod može ali i ne mora biti drugačiji od prototipa. No, to se ovisi o zahtjevima. Korisnički zahtijevi se brzo mogu promijeniti pa je velika vjerojatnost da će i završeni sustav biti potpuno različit od prototipa sustava. Sustav prema ovom prototipu web sučelja i mobilnog sučelja, sadrži elemente koje bi u cjelini trebao sadržavati. Početna stranica, registracija korisnika, profil korisnika, rezultati korisnika, početna aktivnost u mobilnoj aplikaciji, prijava u mobilnu aplikaciju, izbornik u mobilnoj aplikaciji, rezultati korisničke aktivnosti te aktivnost gdje se prikupljaju podaci korisnika. Prema prototipu, korisnik ima kontrolu nad prikupljanjem podataka, pogotovo sa vremenom i brojem koraka. Nema kontrolu nad osjetljivosti i određivanja tempa izračuna koraka, čak nije niti prikazana osjetljivost, već bi osjetljivost bila zadana, nema kontrolu nad lokacijom i mjerenjem brzine. Korisnik ima kontrolu nad prijavom, dok upisuje svoje podatke, nad registracijom, isto kad upisuje podatke te kontrolu nad svojim osobnim podacima. Može ih uređivati i brisati. Kod mobilne aplikacije korisnik ima i kontrolu nad izbornikom, može izabrati koju god aktivnost. Ovakva vrsta prototipa uvelike olakšava izradu konačnog sučelja koje će korisnik koristiti.

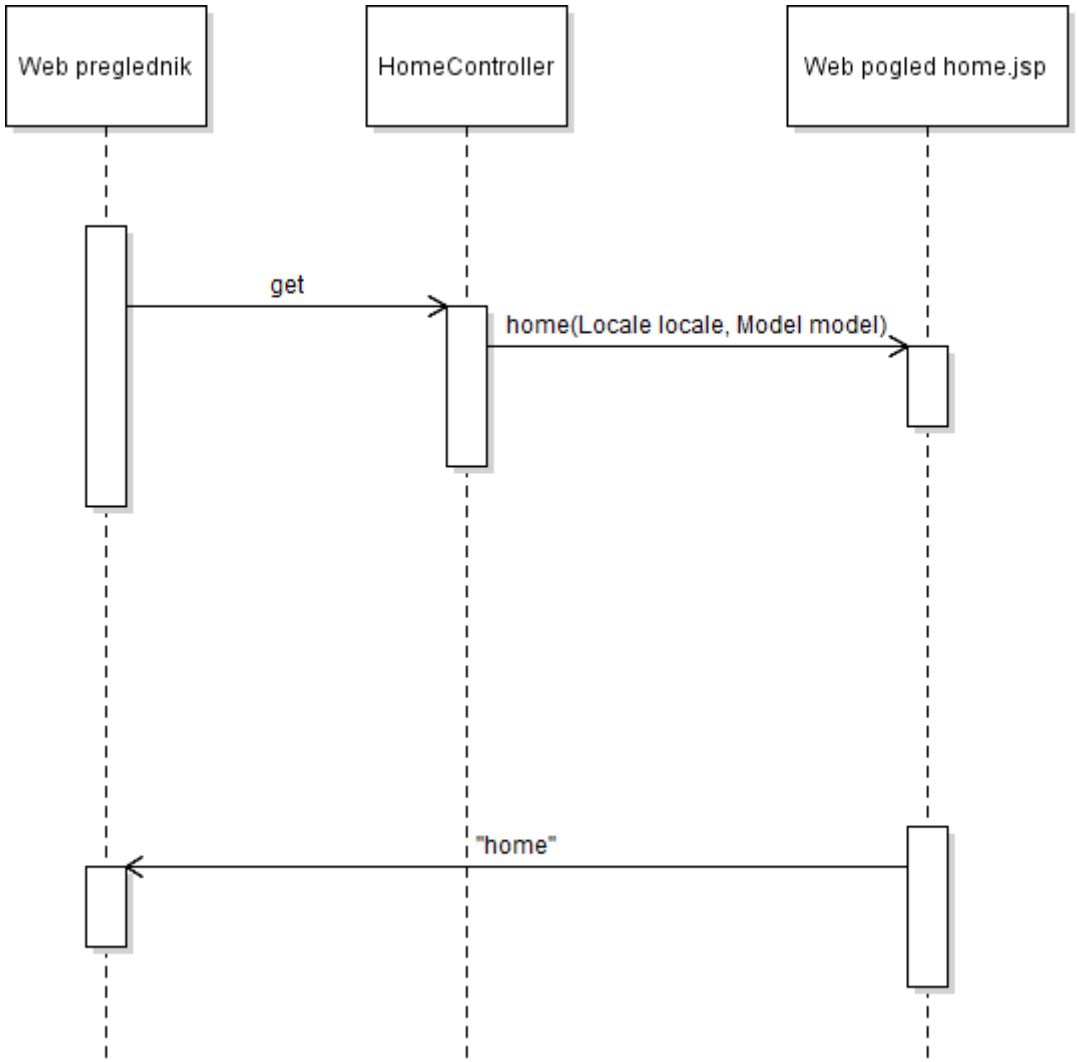
6.3. Funkcionalnost sustava i korisničke upute za korištenje

Kako bi korisnik mogao početi koristiti sustav, potrebna je registracija i prijava. Prije instalacije mobilne aplikacije na svom uređaju, korisnika je potrebno upoznati sa sustavom, kako da se registrira na sustav i kako da se prijavi na sustav kako bi pratio svoje dnevne rekreativne aktivnosti. Sljedeća slika pokazuje početnu stranicu web aplikacije, na koju korisnik mora doći kako bi mogao otići na dio za registraciju.

Slika 31. Početna stranica, izradio: autor

Početna web stranica na koju korisnik prvi put dolazi kada se treba eventualno registrirati ili pregledati svoje podatke objašnjava kako koristiti web i mobilnu aplikaciju. Također, sadrži i navigaciju prema ostalim stranicama. Navigacija je relativno jednostavna i korisniku je lako uočiti linkove prema drugim stranicama, koje bi on mogao posjetiti. Scenarij je sljedeći. Korisnik dolazi na web stranicu. Pri tome se koristi get metoda, kako bi se generirao put gdje se korisnik nalazi. Kontroler kaže web pogledu da generira taj put, a početni put je definiran u servlet-u. Taj put može biti /physical/ ili physical/home. Putem servleta je definirana početna stranica, a kontroler putem @RequestMapping anotacije, kaže koji je taj put. Put je definiran i kod funkcije za generiranje puta, ali i na početnom dijelu kontrolera, gdje se nalaze polja sa

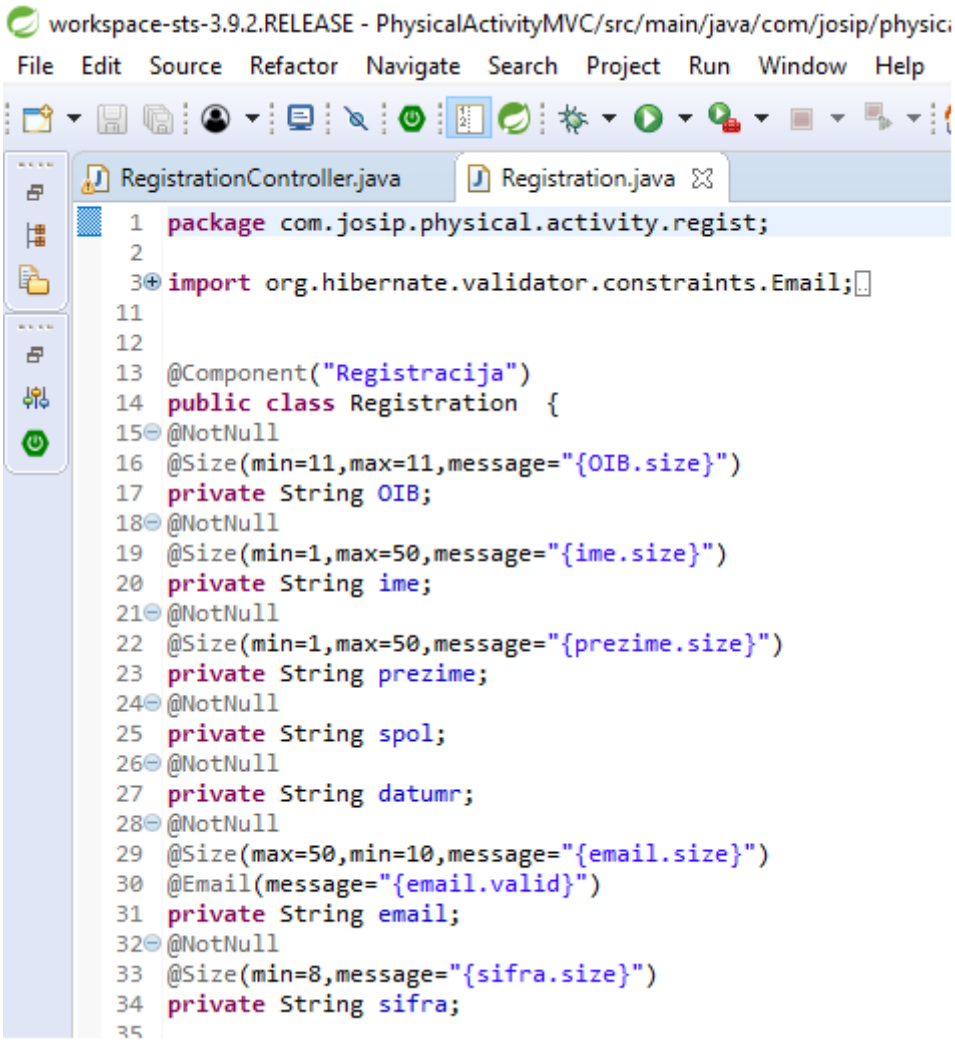
definiranim putevima za ostale stranice za koje postoji veza jednog kontrolera prema drugim stranicama. Tako dakle servlet prima sve puteve i koristi logički ili da generira svaki put koji se nalazi u polju ili kontrolera. Ako neki put nije definiran u tom polju ili kontroleru, servlet će javiti grešku i korisnik neće moći taj dio koristiti. Također, kontroler može sadržavati model koji put model.addAttribute metode prosljeđuje podatke prema web pogledu, koje web pogled generira i stavlja na stranicu, a putem oznake ${lista.naziv} na stranici, generira se objekt sa liste i daje se korisniku na pregled. Sljedeći dijagram pokazuje kako kontroler generira put prema stranici.



Slika 32. Komunikacija između preglednika, kontrolera i pogleda, izradio: autor

Model-prikaz-kontroler je već objašnjen u jednom poglavlju, ali tek toliko da se korisnik ne vraća na to poglavlje, napravljen je ovaj sekvencijski dijagram kako se

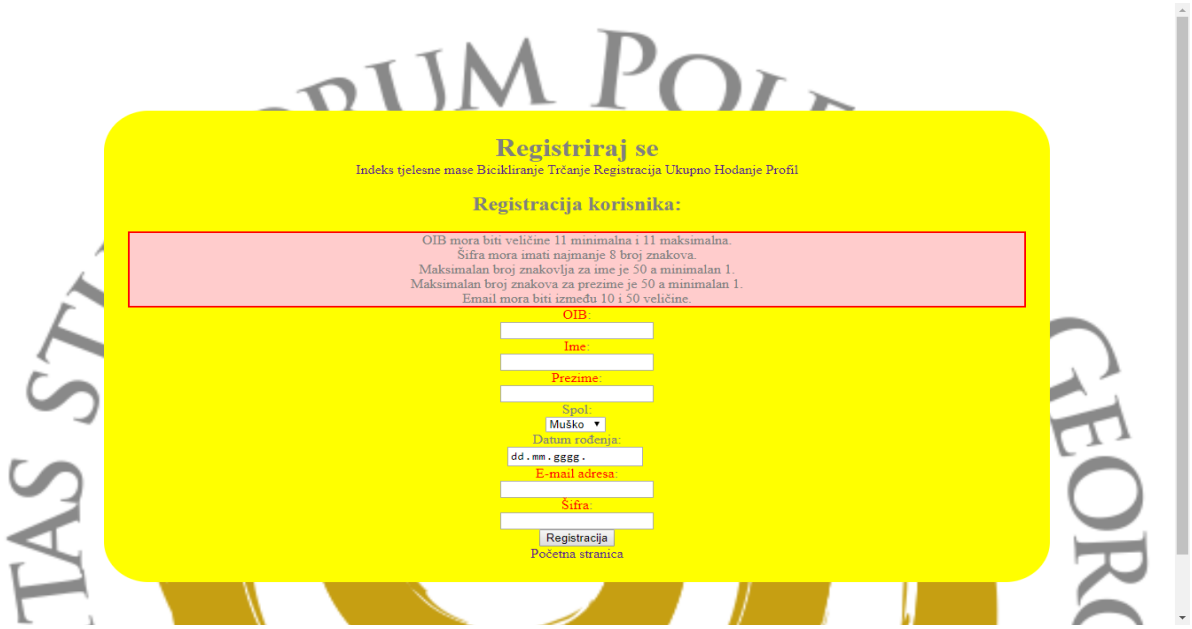
ponaša kontroler, web preglednik i pogled ali bez modela. Dakle, preko web preglednika kad korisnik dođe na web, preglednik šalje kontroleru zahtjev. Kontroler zatim dodaje model, tj. generira obrazac koji će se prikazivati na pregledniku, šalje ga pogledu, a pogled bi vratio put pregledniku gdje bi se otvorila web stranica. Nakon što je korisnik došao na početnu stranicu, pročitao upute za korištenje web stranice, korisnik ide na registraciju. Kada korisnik upiše svoje podatke, pošalje post zahtjev. Pri tome, njegovi podaci se provjeravaju. Postoji validacija na nivou entiteta. Javax validacija omogućuje postavljanje na entitet određene anotacije kao što je npr @NotNull. Sljedeća slika pokazuje dio modela koji validira korisnikove podatke.



Slika 33. Javax validacija, izradio: autor

Dakle, na modelu se validiraju podaci kada se pošalju. Na kontroleru postoji @Valid i Errors objekt. Oni služe za komunikaciju između grešaka. @Valid je dio Javax

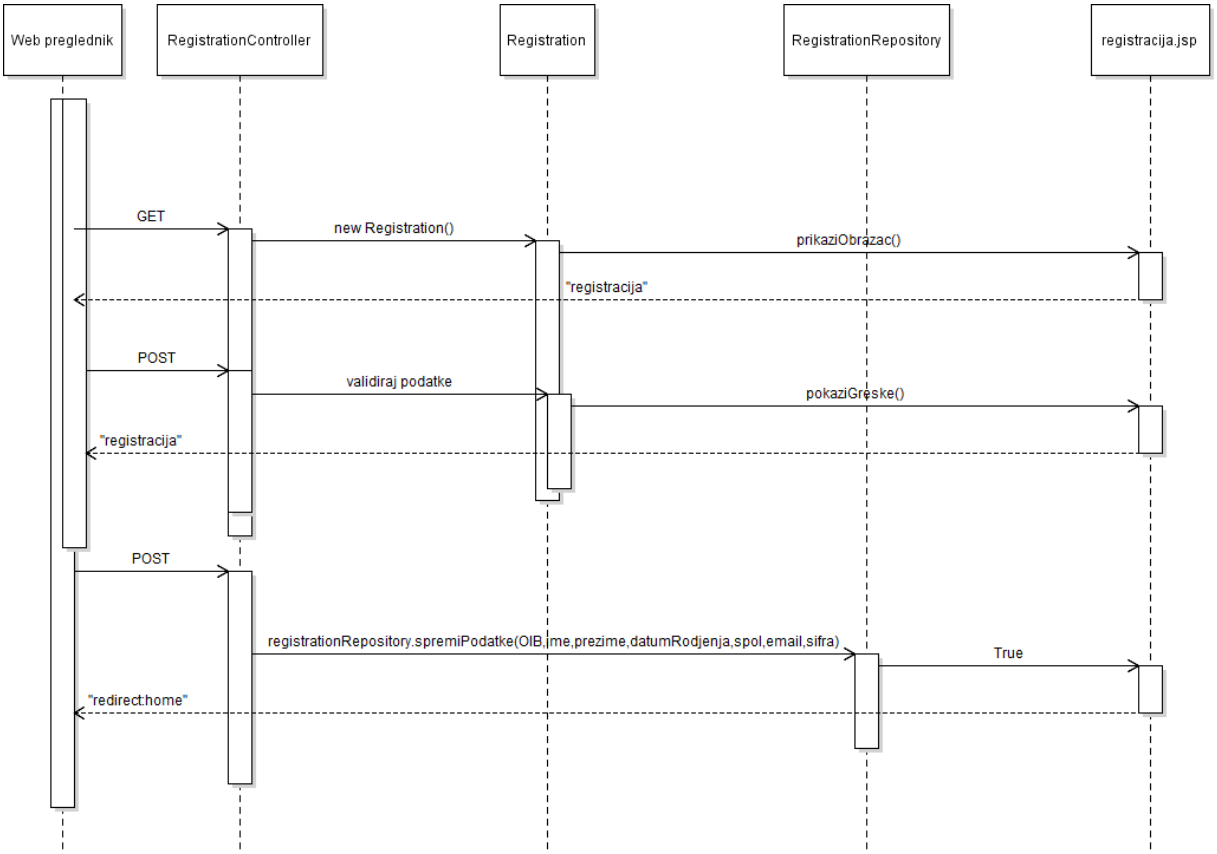
validacije dok je error dio Spring razvojnog okvira. U objekt errors spremaju se greške od Javax validacije i ako ima grešaka, vraća se obrazac, ako nema, korisnik se preusmjerava prema početnoj stranici. Greške se također ispisuju na ekranu, a u zagradi u modelu podataka se vidi ključ-vrijednost poruka. Poruka se dohvaća iz ValidationMessages.poperties i prikazuju se na pregledniku. Na registration.JSP prikazu, koristila se posebna Jstl biblioteka s kojom su se greške ispisale odmah iznad obrasca, a podaci koji su se poslali su se bindali pomoću objekta. Sljedeća slika prikazuje stranicu registracija sa pripadajućim greškama.



Slika 34.Registracija korisnika, izradio: autor

Sljedeći sekvencijalni dijagram prikazuje proces registracije. Kada korisnik dođe na web stranicu, preko get metode dohvaća obrazac za registraciju. Ukoliko postoji bilo kakav problem sa vezom, servlet će javiti grešku da se taj obrazac ne može prikazati. Korisnik je ispunio podatke, kliknuo na gumb za slanje podataka, tada se poziva validacija na nivou modela podataka ili entiteta. Ako ima grešaka, korisnika se vraća na registraciju i pogled prikazuje greške na ekranu,te greške su prikazane na prethodnoj slici. Upisane vrijednosti ne smiju biti null, svaka vrijednost ima određen broj mininmalan i maksimalan broj znakova koji treba zadovoljiti. Email se također validira. Šifra ne smije imati manji broj znakova od osam, kako se ne bi brute force metodom pogodila nečija šifra na temelju korisničkog imena. Nakon što korisnik ispuni svoje podatke i validacija ne javi nikakvu grešku, korisnik može poslati svoje podatke,

tj. spremiti ga u sučelje RegistrationRepository, koje ima svoju implementaciju za spremanje podataka u bazu. Za spremanje podataka registriranih korisnika, korištena je klasična Java veza prema bazi. Primjerice, za REST povezivanje i spremanje podataka, korišten je Spring izvor podataka, koji omogućuje jednostavniji unos i manji kod.



Slika 35. Sekvencijalni dijagram registracije, izradio: autor

Korisnik je sada registriran i može koristiti mobilnu aplikaciju. Sljedeće slike, uključujući i dijagrame, prikazuju kako funkcionira komunikacija između pojedinih dijelova mobilne aplikacije i web aplikacije, uključujući i prijavu preko mobilnog uređaja. Na webu se također nalazi i dodatni kontroler, REST kontroler, koji služi za zaprimanje podataka sa mobilne aplikacije, spremanje tih podataka u bazu. REST kontroler je povezan sa implementacijama klasa, koje se preko SummaryRepositorija, stavljaju na web stranicu u obliku liste i dobijaju se rezultati preko pogleda. Posebne klase za crtanje grafikona se također preko istog kontrolera, poziva i pri tome se crtaju grafovi na temelju podataka koji se dobivaju preko liste. Na webu je također napravljen i filter za

filtriranje podataka po datumu, koji je koristan ukoliko postoji dosta podataka pa korisnik može filtrirati i te podatke proučavati kroz tablični prikaz ili grafički prikaz. Preko mobilnog uređaja, prikupljaju se podaci, koji se privremeno spremaju u json datoteku. Podaci se čitaju i izračunavaju se prosječne i ukupne vrijednosti. Nakon što je sve izračunato, spremljeno, u objekt ukupnih rezultata se stavljaju izračunate vrijednosti i preko klase pošalji ukupne podatke, podaci se šalju na web stranicu. REST kontroler prihvaća te podatke i sprema ih u bazu.

Slika 36. Početne aktivnosti mobilne aplikacije, izradio: autor

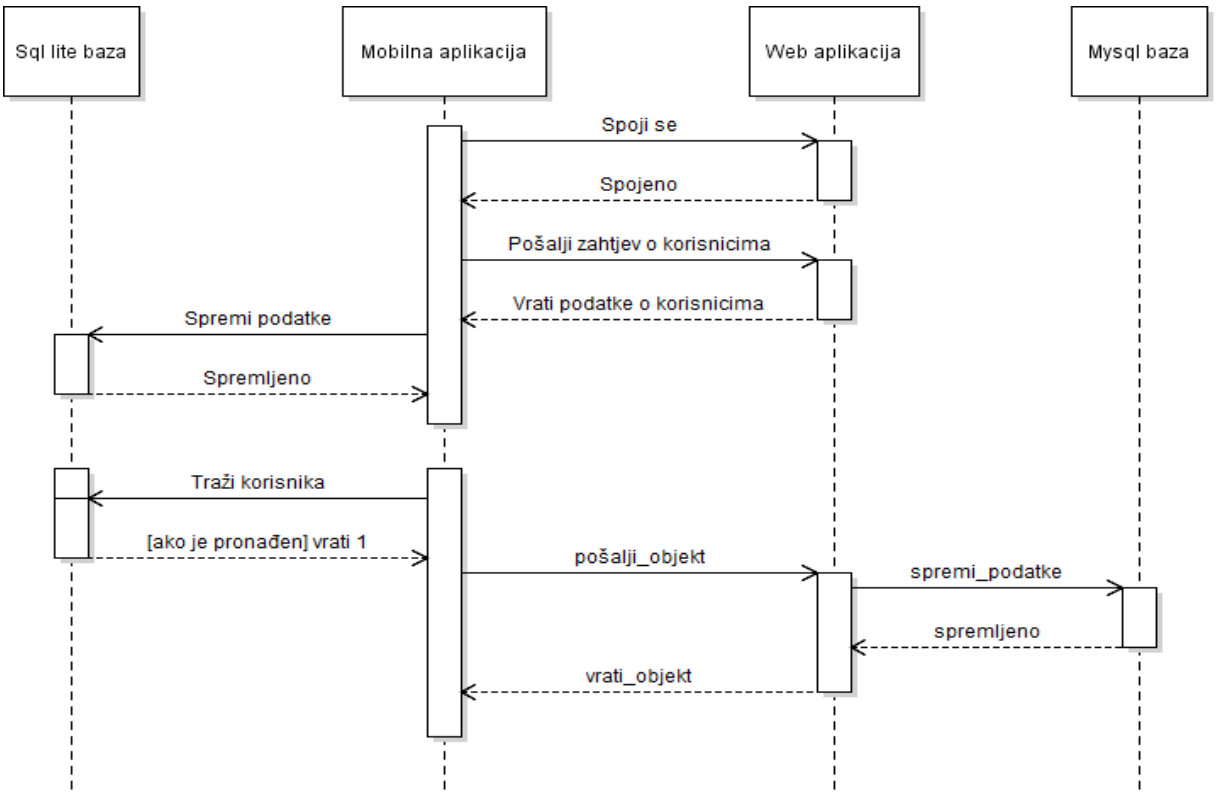
Prva aktivnost predstavlja početnu aktivnost aplikacije. Ako se aplikacija koristi prvi put, pojavljuje se i pomoć. Gumb za brisanje podataka iz prethodne sesije je skriven, jer se aplikacija koristi prvi put i nema potrebe da se prikazuje. Gumb se pojavi tek kad se aplikacija koristi više puta pa ako korisnik ne želi da mu se podaci prate od prethodne sesije, korisnik klikne na brisanje podataka. Tada se brišu podaci spremljeni u json datoteku. Kada korisnik klikne na gumb pokreni, putem namjere se otvori sljedeća aktivnost. Na toj aktivnosti, je obrazac za prijavu. Kao i u prethodnoj aktivnosti, ako se aplikacija koristi prvi put, pojave se instrukcije kako se prijaviti. Korisnik upisuje svoje podatke, no u međuvremenu se gumb sakrije, jer se u trenutku stvaranja aktivnosti poziva asinkroni zadatak koji dohvaća podatke od korisnika sa poslužitelja i sprema je u SQLite bazu, da bi se poslije korisnik pretraživao u bazi. Ako se nalazi u

bazi, korisnik se stavlja u JSON i zatim se otvara lista fragmenata. Ukoliko korisnik nije spojen na internet i ne može se dohvatiti korisnički podaci sa poslužitelja, kao što je prikazano na slici, javlja se greška. Ako je korisnik prijavljen, otvara se izbornik u obliku liste. Na toj listi fragmenata, kada korisnik klikne na bilo koju aktivnost, otvara se nova aktivnost i korisnik može započeti prikupljati svoje podatke.

Slika 37. Mobilna aplikacija u akciji, izradio: autor

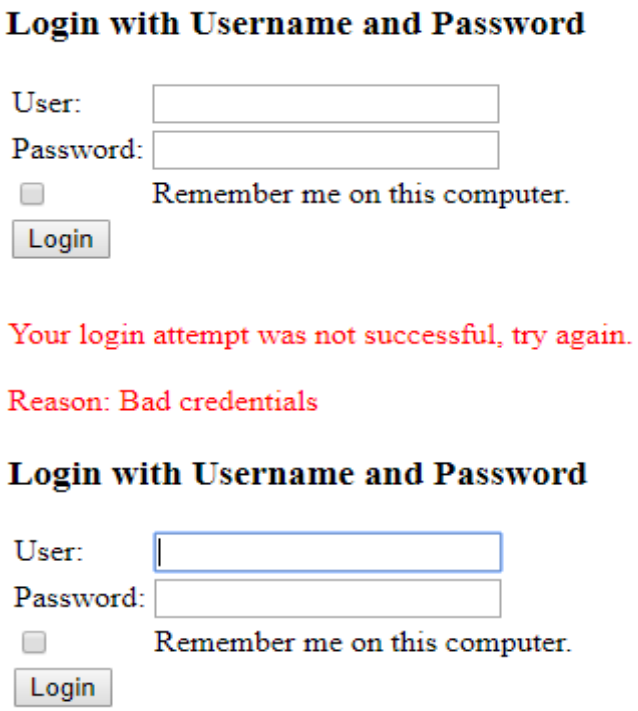
Ako se aplikacija koristi prvi put, pojave se instrukcije ako korisnik nije upoznat sa aplikacijom. Za svaku aktivnost, bicikliranje, trčanje i hodanje postoji kronometar. Kronometar postoji za mjerenje vremena aktivnosti. Korisnik ima mogućnost stopirati

kronometar, pauzirati ili resetirati ukoliko nije nešto u redu sa vremenom. Aktivnosti za bicikliranje i trčanje su slične, a one sadrže samo podatke koji su pročitani od strane gps-a. Kod hodanja, postoji i dodatni senzor, acelerometar, koji mjeri korakena temelju osjetljivosti. Nakon što se pomakne mobilni uređaj, korak se mjeri ovisno o osjetljivosti. Što je osjetljivost manja, koraci se mjere brže. To se podešava putem klizača. Nakon što korisnik izađe iz aktivnosti, mjeri se prosječna brzina i ukupno vrijeme aktivnosti te ukupni prijeđeni kilometri. Korisnik može i resetirati korake. Rezultati se spremaju u JSON. Udaljenost i brzina se spremaju u listu, a ukupno vrijeme se dohvaća kada se zaustavi kronometar i s time se pozove objekt od chrono helpera i s time se dohvati vrijeme u milisekundama. Sveukupni rezultati se čitaju iz jsona u rezultatima, stavlja se korisniku na ekran, a korisnik kada izađe iz rezultata, podaci se šalju na web stranicu. REST kontroler prima podatke sa mobilne aplikacije i sprema ju u bazu. Podaci od prethodne sesije korisnika, mogu se izbrisati na početku, ukoliko korisnik to želi. Samo se u json stavljaju null podaci. Sljedeći dijagram opisuje komunikaciju mobilne aplikacije i web stranice.



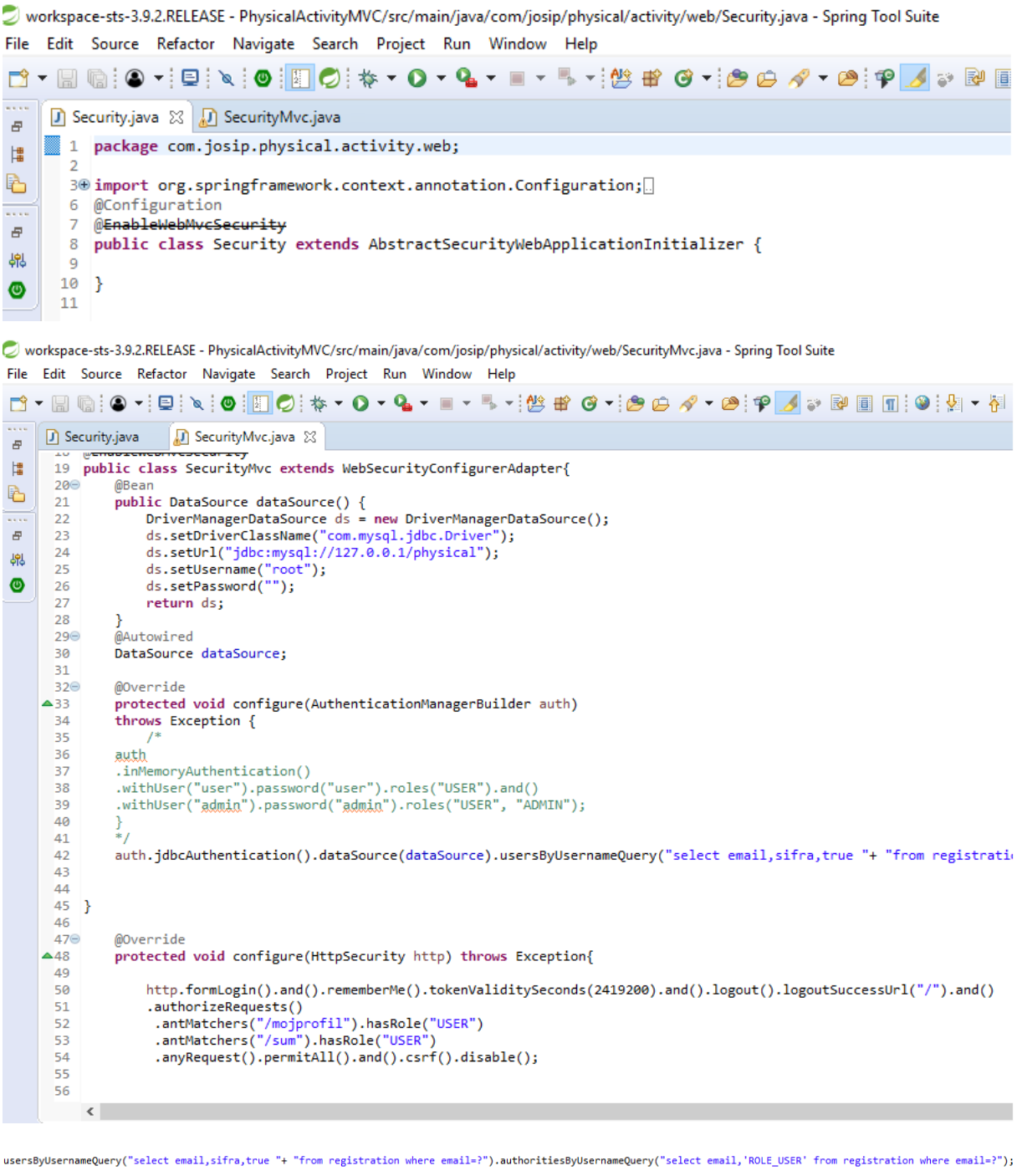
Slika 38.Komunikacija mobilne i web apllikacije,izradio:autor

Kad korisnik upali aplikaciju, pojavi se početna aktivnost. Korisnik ima dva gumba. Jedan gumb se pojavi kasnije, osim ako se aplikacija ne koristi prvi put. Ako korisnik klikne na gumb pokreni, pokreće se namjera i korisnik ide prema aktivnosti za prijavu na mobilnu aplikaciju. Tada, mobilna aplikacija pri kreiranju zahtjeva od web aplikacije podatke od korisnika. Ako je sve u redu, aplikacija zaprimi podatke. Ako ne, korisniku će se, kad upiše svoje korisničko ime i lozinku, javljati greška. Nakon što se korisnik prijavio, ide u izbornik i odabire jednu od ponuđenih aktivnosti. Nakon što skupi podatke, podaci se spremaju u json datoteku. Podaci se kupe pomoću liste i json datoteka. Kada su podaci spremljeni i korisnik ide na rezultate, korisniku će se prikazati rezultati iz json datoteke. Pojedini rezultati su izračunati iz posebne pomoćne klase. Kada se korisnik vraća iz rezultata, tada se šalju objekti rezultata, pojedinačno i sveukupno i tada web aplikacija dobiva podatke i sprema ih u bazu podataka. Na temelju tih podataka generiraju se grafikoni i prikazuju tablično na web stranici.Stranica za pregledavanje rezultata je ograničena od strane Spring web sigurnosti, a podaci za prijavu, pogotovo korisničko ime, služi za izlistavanje podataka od trenutnog korisnika. Prijava putem Spring sigurnosti prikazana je na sljedećoj slici. Također, ukoliko korisnički podaci ne valjaju, po defaultu će se javiti greška.



Slika 39. Spring sigurnost, izradio: autor

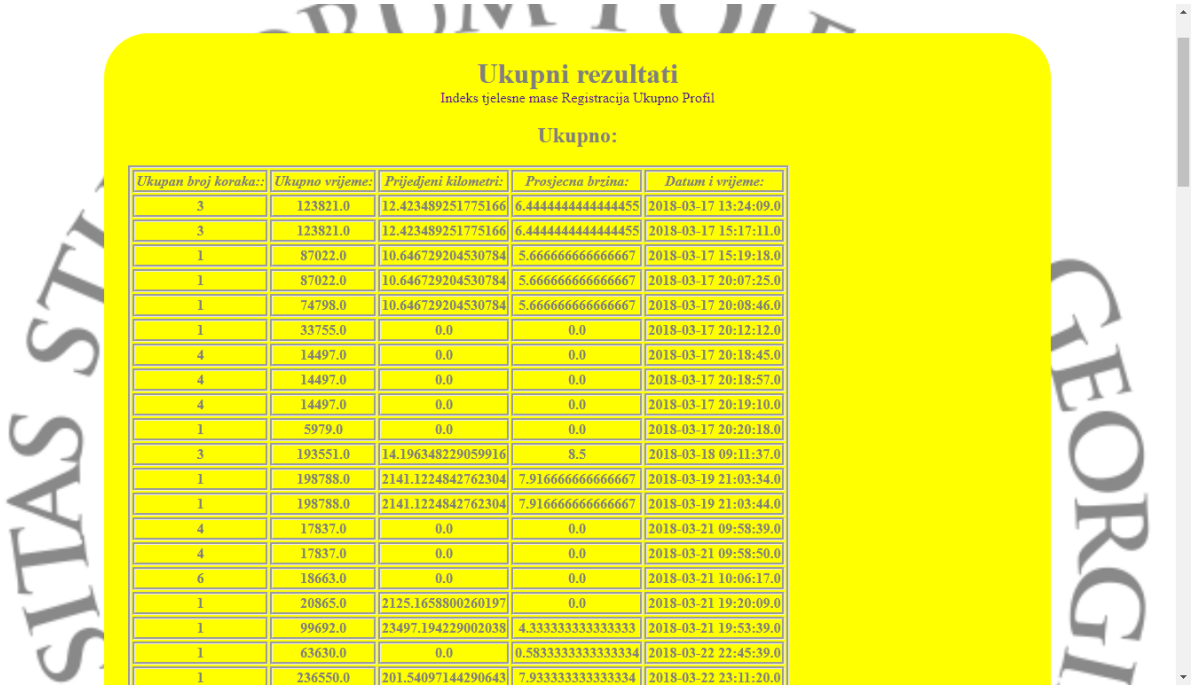
Preporučljivo je napraviti svoju vlastitu implementaciju prijavnice, pogotovo što ovakva prijavnica narušava izgled koji je definiran preko css-a. Također, ukoliko se želi stranica koja je potpuna na drugom jeziku osim engleskog, isto tako je preporučljivo da se napravi vlastita prijavnica. Zbog mnogobrojnih poteškoća, trenutna verzija koristi default prijavnicu. Prijavnica je napravljena preko Spring sigurnosti. Postoji nekoliko klasa koje definiraju Spring sigurnost.



Slika 40. Web sigurnost, izradio: autor

Klasa `Security` implementira filter koji sadržava druge filtere za sigurnost. Bilo koja klasa koja nasljeđuje konfiguracijski priključak od spring web sigurnosti, sada može pridonjeti sigurnosti. U klasi koja implementira adapter, nalazi se izbor podataka. To je baza podataka na koju se spaja i traži korisnike sa korisničkim imenom i lozinkom. Nadjačava se funkcija za autentifikaciju i piše se vlastita implementacija. Sljedeće nadjačavanje je nadjačavanje Http sigurnosti. Određuju se koje stranice će se zaštititi te koliko će pojedini token trajati, i također se određuje da se u određenom trenutku prikaže prijavnica za korisnika za te stranice koje su ograničene za pristup. Ograničene stranice su profil korisnika i rezultati korisnika. Klasa se spaja na bazu, i traži korisnike sa pripadajućim korisničkim imenom. Ukoliko su korisničko ime i lozinka jednaki, korisnik se autentificira i autorizira mu se pristup aplikaciji. Ima privilegije običnog korisnika kao i svi prijavljeni korisnici.

Dakle, nakon što je korisnik prijavljen, otvara mu se stranica i njegovi podaci se, prema Spring sigurnosti, stavljaju u listu i prikazuju na ekranu. Na stranici su prikazani tablični rezultati, ukupni rezultati po aktivnostima i ukupni rezultati uopće. Tako je i na ostalim stranicama gdje se prikazuju tablični prikazi podataka. Dodan je i filter, kako bi korisnik mogao filtrirati rezultate u slučaju da ih ima previše na stranici.



Ukupni rezultati

Indeks tjelesne mase Registracija Ukupno Profil

Ukupno:

Ukupan broj koraka::	*Ukupno vrijeme:*	*Prijeđeni kilometri:*	*Prosjecna brzina:*	*Datum i vrijeme:*
3	123821.0	12.4234892517751 66	6.4444444444444455	2018-03-17 13:24:09.0
3	123821.0	12.4234892517751 66	6.4444444444444455	2018-03-17 15:17:11.0
1	87022.0	10.646729204530784	5.666666666666667	2018-03-17 15:19:18.0
1	87022.0	10.646729204530784	5.666666666666667	2018-03-17 20:07:25.0
1	74798.0	10.646729204530784	5.666666666666667	2018-03-17 20:08:46.0
1	33755.0	0.0	0.0	2018-03-17 20:12:12.0
4	14497.0	0.0	0.0	2018-03-17 20:18:45.0
4	14497.0	0.0	0.0	2018-03-17 20:18:57.0
4	14497.0	0.0	0.0	2018-03-17 20:19:10.0
1	5979.0	0.0	0.0	2018-03-17 20:20:18.0
3	193551.0	14.196348229059916	8.5	2018-03-18 09:11:37.0
1	198788.0	2141.1224842762304	7.916666666666667	2018-03-19 21:03:34.0
1	198788.0	2141.1224842762304	7.916666666666667	2018-03-19 21:03:44.0
4	17837.0	0.0	0.0	2018-03-21 09:58:39.0
4	17837.0	0.0	0.0	2018-03-21 09:58:50.0
6	18663.0	0.0	0.0	2018-03-21 10:06:17.0
1	20865.0	2125.1658800260197	0.0	2018-03-21 19:20:09.0
1	99692.0	23497.194229002038	4.333333333333333	2018-03-21 19:53:39.0
1	63630.0	0.0	0.5833333333333334	2018-03-22 22:45:39.0
1	236550.0	201.54097144290643	7.933333333333334	2018-03-22 23:11:20.0

Slika 41. Ukupni rezultati, izradio:autor

3	72695.0	0.0	1.4444444444444444	2018-03-26 13:39:48.0
6	62483.0	102.43094799433939	1.0	2018-03-26 13:53:32.0

Ukupno po bicikliranju

Ukupna udaljenost:	*Ukupno vrijeme aktivnosti:*	*Prosjecna brzina*	*Datum i vrijeme*
7.096578215918581	34979.0	11.0	2018-03-17 08:15:23.0
7.096578215918581	34979.0	11.0	2018-03-17 08:15:23.0
7.096578215918581	34979.0	11.0	2018-03-17 08:15:23.0
7.096578215918581	34979.0	11.0	2018-03-17 08:15:23.0
7.096578215918581	34979.0	11.0	2018-03-17 08:15:23.0
8.869614514564777	90363.0	8.0	2018-03-18 09:11:36.0
8.869614514564777	90363.0	8.0	2018-03-18 09:11:36.0
7832.401595125167	25335.0	3.5	2018-03-21 19:53:37.0
105.8123440535569	33723.0	10.0	2018-03-22 23:11:19.0
630.5780114631734	129643.0	10.25	2018-03-26 13:43:03.0

Ukupno trcanja

Ukupna udaljenost:	*Ukupno vrijeme aktivnosti:*	*Prosjecna brzina*	*Datum i vrijeme*
3.550150988612203	32949.0	6.0	2018-03-17 08:14:45.0
3.550150988612203	32949.0	6.0	2018-03-17 08:14:45.0
3.550150988612203	32949.0	6.0	2018-03-17 08:14:45.0
3.550150988612203	32949.0	6.0	2018-03-17 08:14:45.0
0.0	3501.0	0.0	2018-03-17 20:12:59.0
3.549973667250756	41423.0	13.5	2018-03-18 09:09:58.0
3.549973667250756	41423.0	13.5	2018-03-18 09:09:58.0
3.549973667250756	41423.0	13.5	2018-03-18 09:09:58.0

Slika 42.Ukupni rezultati, izradio: autor

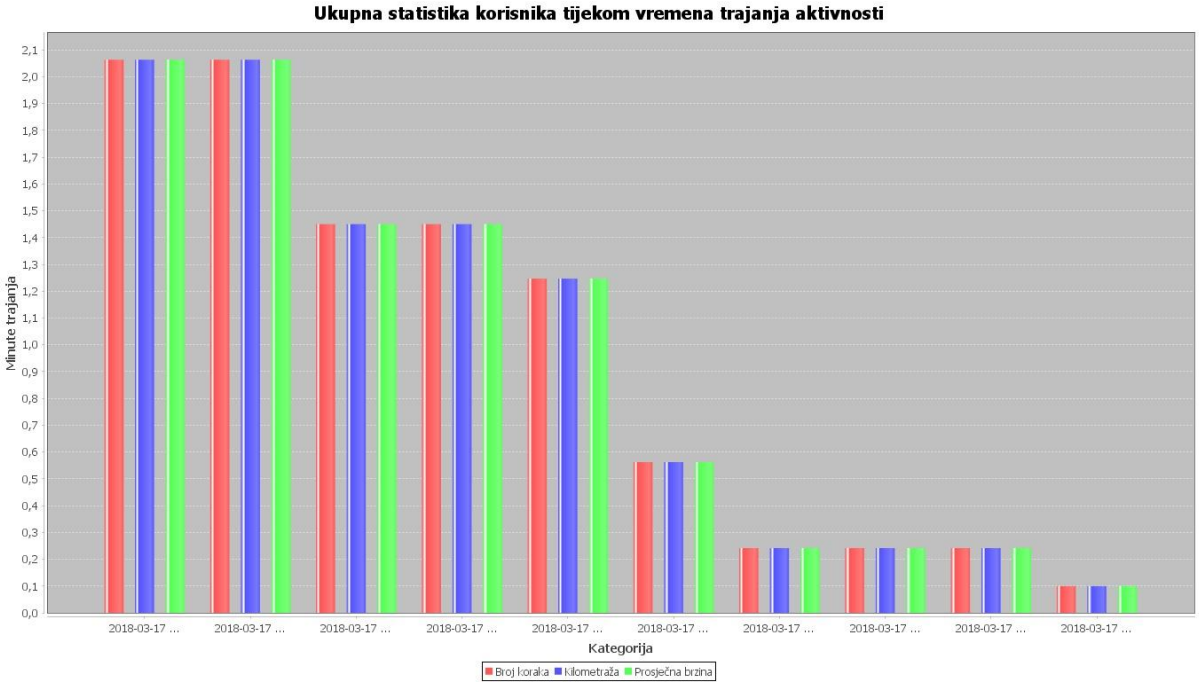
7832.391018024694	36981.0	8.5	2018-03-21 19:52:59.0
0.0	143894.0	9.8	2018-03-22 23:10:45.0
0.0	110169.0	10.5	2018-03-22 23:14:04.0
1795.541904218688	1761.0	14.666666666666666	2018-03-22 23:23:33.0
103.87863135147919	42386.0	2.0	2018-03-26 13:40:38.0
103.87863135147919	42386.0	2.0	2018-03-26 13:40:38.0

Ukupno hodanja:

Ukupna udaljenost:	*Ukupno vrijeme aktivnosti:*	*Prosjecna brzina*	*Datum i vrijeme*	*Ukupan broj koraka*
1.7767600472443834	55893.0	2.0	2018-03-17 08:14:10.0	3
0.0	19094.0	0.0	2018-03-17 15:19:16.0	1
0.0	19094.0	0.0	2018-03-17 15:19:16.0	1
0.0	6870.0	0.0	2018-03-17 20:08:45.0	1
0.0	33755.0	0.0	2018-03-17 20:12:00.0	1
0.0	33755.0	0.0	2018-03-17 20:12:00.0	1
0.0	4161.0	0.0	2018-03-17 20:17:31.0	1
0.0	14497.0	0.0	2018-03-17 20:18:44.0	4
0.0	14497.0	0.0	2018-03-17 20:18:44.0	4
0.0	14497.0	0.0	2018-03-17 20:18:44.0	4
0.0	9027.0	0.0	2018-03-17 20:20:06.0	7
0.0	5979.0	0.0	2018-03-17 20:20:17.0	1
1.7767600472443834	61765.0	4.0	2018-03-18 09:09:14.0	3
2128.7028960944144	67002.0	2.0	2018-03-19 21:03:33.0	1
2128.7028960944144	67002.0	2.0	2018-03-19 21:03:33.0	1
0.0	17837.0	0.0	2018-03-21 09:58:36.0	4
0.0	17837.0	0.0	2018-03-21 09:58:36.0	4
0.0	18663.0	0.0	2018-03-21 10:06:16.0	6

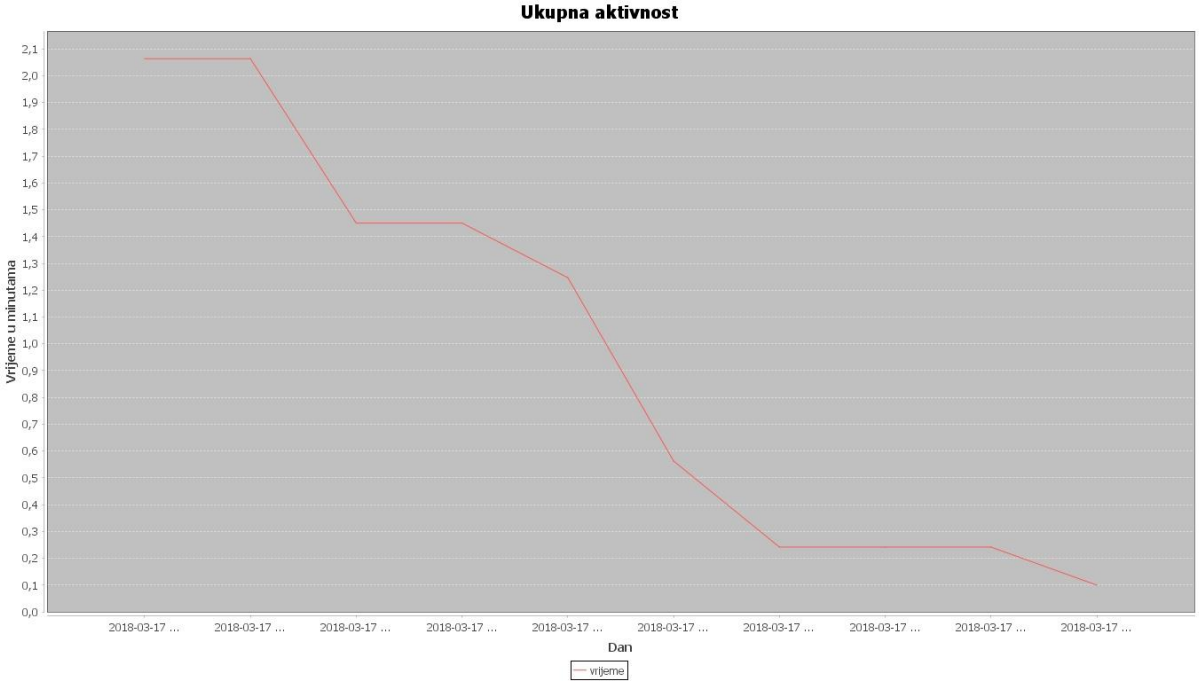
Slika 43. Ukupni rezultati, izradio: autor

Slika 44. Ukupni rezultati, izradio: autor



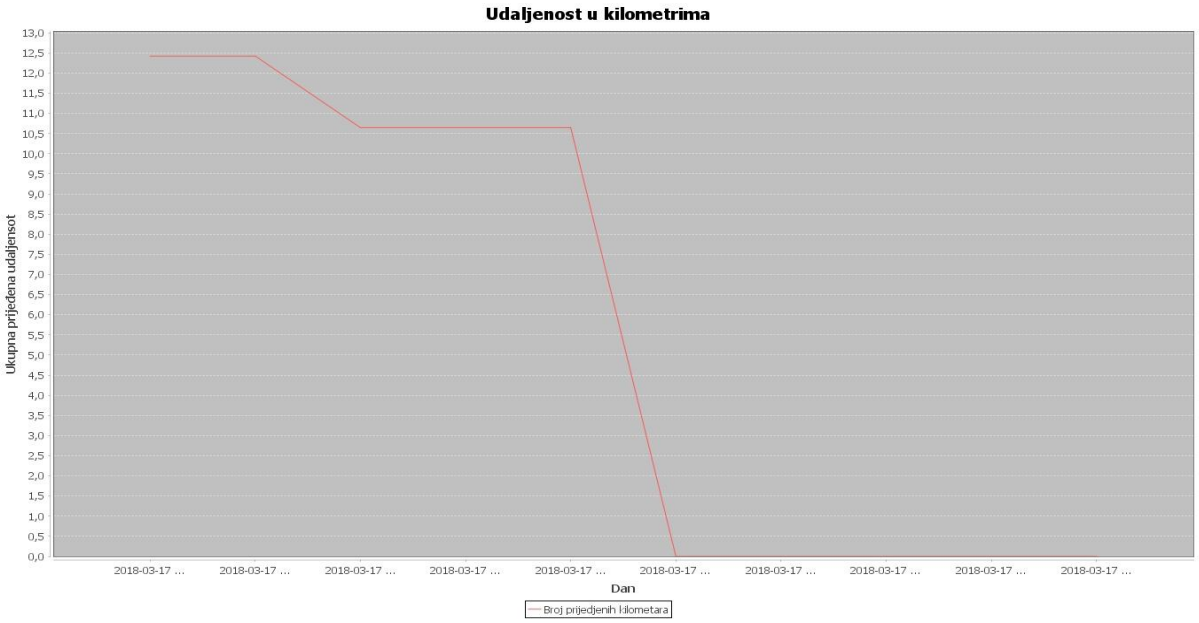
Slika 45. Histogram, izradio: autor

Histogram uspoređuje ukupne rezultate po pojedinoj aktivnosti. Na y koordinati se nalazi ukupan broj minuta tijekom jednog perioda. Na x u se nalazi datum kada je korisnik bio aktivan. Na grafikonu se prikazuje ukupan broj koraka, kilometraža te prosječna brzina tijekom vremena aktivnosti po periodu.



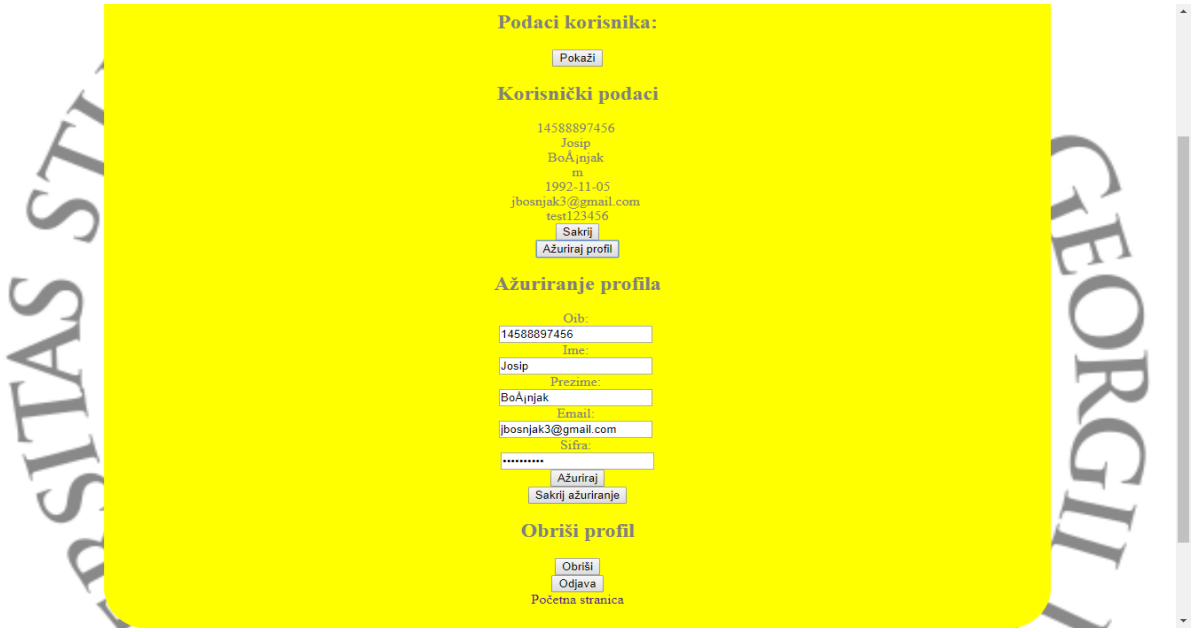
Slika 46.Linijski grafikon, izradio: autor

Ovaj linijski grafikon prikazuje ovisnost vremena o periodu, tj. pokazuje kolika je korisnikova aktivnost bila tijekom određenog perioda. Na grafu se jasno vidi da korisnik smanjuje aktivnost tijekom određenog perioda.



Slika 47. Linijski grafikon, izradio: autor

Ovaj grafikon pokazuje koliko je korisnik prešao kilometara tijekom određenog perioda. Vidi se da kilometraža pada nakon određenog vremena.



Slika 48.Stranica profila mojprofil.JSP, izradio: autor

Na prethodnoj slici prikazana je stranica za pregled podataka od korisnika, ažuriranje podataka od korisnika te brisanje podataka od korisnika. Ažurirati se mogu samo podaci poput oiba, imena, prezimena, email-a i šifre, dok se podaci poput datuma rođenja i spola ne mogu mijenjati. Ukoliko je potrebno da se mijenjaju ti podaci, tada korisnik treba obrisati svoj profil i kreirati novi profil. Tada će izgubiti i sve svoje podatke koje je prikupio tijekom korištenja mobilne aplikacije. Naravno, stranica sadrži i gumb za odjavu, ukoliko se korisnik želi odjaviti. Prilikom odjave korisnika, korisnik se vraća na početnu stranicu. Ukoliko korisnik želi obrisati profil, klikne na obriši i pri tome kontroler aktivira funkciju za brisanje, koja se nalazi u repositoriju i zatim se korisnik obriše iz baze. Nakon brisanja, korisniku se prikazuje početna stranica. Stranicu je potrebno osvježiti, jer zbog nekog razloga, nestanu neke HTML oznake na pogledu. Sljedeća slika prikazuje indeks tjelesne mase.

Slika 49. Stranica za unos indeksa tjelesne mase, izradio: autor

Preko mobilne aplikacije se također može pristupiti ovoj web stranici. Aplikacija koristi widget web pogled koji se preko Java klase spaja na web i prikazuje web stranicu koja se inače prikazuje u web pregledniku direktno u aplikaciji. Pri tome je potrebna internet veza, inače će servlet javiti grešku da stranica nije dostupna. Indeks tjelesne mase može biti pokazatelj dali korisnik ima pretjeranu tjelesnu težinu u odnosu na svoju visinu. Neke granične vrijednosti nisu iste za žene i muškarce pa je dodan i spol kako bi se što realnije prikazao indeks tjelesne mase, na temelju kojeg korisnik može određivati svoj tempo rekreacije.

6.4. Slične aplikacije

Neke od sličnih aplikacija za praćenje fizičke aktivnosti kao što su Runkeeper, Endomondo i Fitbit imaju neke veće mogućnosti nego što ima trenutna aplikacija. Trenutno napravljena aplikacija koncentrira se na prikupljanje osnovnih podataka, poput udaljenosti između točaka, brzine u kilometrima, vremena trajanja aktivnosti, broju prehodanih koraka tako da postoji dobar dio podataka koji se mogu vizualizirati na web stranici i mogu se u nekim grafikonima prikazivati te se uspoređivati sa drugim rezultatima. Runkeeper koristi oko 15 milijuna ljudi, koji vole trčati. Aplikacija mjeri formu dok se trči, a korisnik može pratiti rezultate, kreirati rute i planove za iduća trčanja. Na njoj se može pratiti i druge vrste vježbanja,te se može i integrirati i sa drugim spravama. Dostupna je za Android i Ios. Endomondo je aplikacija koja prati trčanje, vožnju biciklom i ostale aktivnosti a slovi kao jedna od najboljih aplikacija svoje vrste. Uz praćenje podataka, aplikacija omogućuje i snimanje vlastitih motivacijskih govora za svoje prijatelje,te ih se tako potiče na vježbanje i održavanje forme. Ova aplikacija je ocijenjena izvrsnom ocjenom, a dostupna je za Android i Ios. Fitbit je aplikacija koja koristi pametnu narukvicu za prikupljanje podataka. Automatski mjeri korake, trčanje, kalorije i težinu,te iznosi i druge zdravstvene statistike. Za fitbit nije potrebna narukvica jer je dovoljan i mobilni uređaj. Dostupna je i za Android i za Ios.

Zaključak

Zadatak ovog diplomskog rada je izraditi rješenje koje bi moglo motivirati ljude za bavljenje određenom fizičkom aktivnošću. Izrađen je sustav koji prikuplja osnovne informacije o aktivnosti korisnika, kao što su vrijeme aktivnosti, prijeđena brzina, broj koraka koliko je korisnik prohodao te udaljenost po pojedinoj aktivnosti koji je korisnik prešao. Omogućena je i vizualizacija ukupnih podataka pa korisnik može interaktivno vidjeti koliku je aktivnost tijekom dana korisnik imao,te koliko je kilometara prešao tijekom tog istog dana,te je napravljen grafikon sa usporedbom broja koraka, prosječne brzine te prijeđene udaljenosti pa se podaci lakše mogu uspoređivati sa podacima iz drugih dana. Kako bi grafikon bio uredniji, napravljen je i filtar u kojem korisnik filtrira podatke po datumu.

Veoma je važna svakodnevna tjelesna aktivnost kako bi se čovjek osjećao i sretnije i zadovoljnije svojim životom. U današnjem sjedilačkim načinom života, čovjek bi se trebao što više baviti nekim rekreativnim sportom, poput trčanja ili bicikliranja, ali tu i šetanje dolazi u obzir. Tjelesna aktivnost smanjuje rizik od kardiovaskularnih bolesti i povećava kvalitetu života. Aplikacije za praćenje fizičke aktivnosti tu mogu doprinjeti da rekreativne aktivnosti ne budu više dosadne i mogu postati vrlo zabavne. Korisnici si međusobom mogu djeliti podatke, rengirati ih i međusobno se natjecati. To dovodi do kvalitetnijeg načina života.

Sustav je napravljen do razine prototipa i daje sliku sustava kako bi on u produkciji trebao izgledati i koje bi probleme trebao riješavati. Problem prikupljanja statističkih podataka, motiviranje ljudi za bavljenje fizičkom aktivnošću, vizualizacija podataka tijekom određenog perioda, poboljšanje kvalitete života.

Literatura

Andreja Bogdan, Daniela Babačić. 2015. »Intrinzična i ekstrinzična motivacija za sport i vježbanje u funkciji dobi.« *Stručni rad.* Čakovec: Međimursko veleučilište u Čakovcu, 2. 10.

Bartoš, Allen. 2015. »Zdravlje i tjelesna aktivnost civilizacijska potreba modernog čovjeka.« *Media, culture, and public relations*, 15. 2: 68-78.

n.d. *Dietpharm.* Pokušaj pristupa 3. 5 2017. http://www.dietpharm.hr/vaznost-tjelesne-aktivnosti-za-prevenciju-i-nastanak-bolesti-a90.

Dvorski, Dalibor D. 2007. »Installing, configuring, and developing with XAMPP.« *Ontario Skills Competition* 1-10.

Fredrich, Todd. 2012. »RESTful Service Best Practices.« Pearson eCollege, 29. 05.

Hagos,ted. 2018. *Learn Android Studio 3 Efficient Android App Development.* Manilla: Apress.

HINA. 2017. *HOO.* 12. 7. Pokušaj pristupa 10. 2 2018. http://www.hoo.hr/hr/olimpizam/olimpijske-vijesti/4848-cak-62-hrvata-nije-tjelesno-aktivno-u-nedovoljnom-kretanju-prednjace-stariji.

Kuserbajn, Sara. 2016. »Rest Arhitektura.« *Završni rad.* Šibenik: Veleučilište u Šibeniku, Kolovoz.

Ravi Kant Soni, Amuthan Ganeshan Rajesh RV. 2017. *Spring: Developing Java Applications for the Enterprise.* Birmingham: Packt.

Savitch, Walter. 2016. *Absolute Java.* London: Pearson.

Sommerville, Ian. 2016. *Software Engineering.* Harlow,Essex: Pearson Education.

Walls, Craig. 2015. *Spring in Action.* New York: Manning.

Popis slika

Popis tablica

Sažetak

Glavni cilj ovog diplomskog rada je bio izraditi sustav za praćenje fizičke aktivnosti. Sustav uključuje autentifikaciju i autorizaciju korisnika, prikupljanje podataka korisnika preko mobilne aplikacije, slanje podataka na web poslužitelj, spremanje tih podataka u bazu podataka poslužitelja,te prikaza rezultata na ekranu mobilne aplikacije te vizualizacija grafičkim prikazom na web stranici. Rad se temelji i na teorijskom i praktičnom dijelu. U teorijskom dijelu, objašnjena je jezgra Springa, Android operacijski sustav, daje se kratki pregled povijesti Java programskog jezika. Objašnjen je ukratko i REST te razvojnog okvira model-prikaz-kontrolera. U praktičnom radu, izrađena je mobilna aplikacija i web aplikacija. Mobilna aplikacija služi za prikupljanje podataka korisnika i slanje na web poslužitelj, dok je zadatak web poslužitelja da spremi te podatke i prikazuje ih na web stranici.

Ključne riječi: Spring, Java, sustav za praćenje fizičke aktivnosti, hodanje, trčanje, bicikliranje, model-prikaz-kontroler, mobilna aplikacija, web aplikacija

Summary

Main goal of this paper is to develop application for monitoring physical activity. App includes authentification and authorization of users, collecting data from users by mobile application, sending data to web poslužitelj, storing data on poslužitelj 's database, showing results on mobile app and graphical visualization of results. This paper contains theoretical and pratcical part. In practical part, it is developed mobile and web application. Mobile app collects data from user. Web poslužitelj stores data to database and helps to visualize data on web application. In theoretical part, there is an explaination of Spring core, some Java history, model view controller architecture and REST.

Key words: Spring, Java, application for monitoring physical activity, walking, running, biking, model view controller, mobile app, web app