Principi programskog inženjerstva na primjeru aplikacije za pronalazak potencijalnih suigrača

Makar, Tomislav

Undergraduate thesis / Završni rad

2022

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:110:661820

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-12-25**

Repository / Repozitorij:

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

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

TOMISLAV MAKAR

PRINCIPI PROGRAMSKOG INŽENJERSTVA NA PRIMJERU APLIKACIJE ZA PRONALAZAK POTENCIJALNIH SUIGRAČA

ZAVRŠNI RAD

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

TOMISLAV MAKAR

PRINCIPI PROGRAMSKOG INŽENJERSTVA NA PRIMJERU APLIKACIJE ZA PRONALAZAK POTENCIJALNIH SUIGRAČA

PRINCIPLES OF SOFTWARE ENGINEERING ON THE EXAMPLE OF APPLICATION FOR FINDING POTENTIAL PLAYERS

ZAVRŠNI RAD

Mentor:

dr. sc. Josip Nađ, pred.

ČAKOVEC, 2022.

ZAHVALA

Želim se zahvaliti dragom mentoru dr.sc. Josipu Nađu na strpljenju, pomoći i vođenju tijekom pisanja rada, čime je omogućio da temu ovog završnog rada kvalitetno razložim i sročim u strukturiranu i koherentnu cjelinu.

Tomislav Makar

SAŽETAK

Razvitak interneta donio je golem napredak u protoku informacija i tehnologije koja je postala svakodnevica u današnjem svijetu. Samim time povezanost ljudi nikada nije bila veća. Kako bi se ta povezanost iskoristila na što bolji način, u ovom će radu biti osmišljena jednostavna aplikacija. Završni rad pod nazivom „Principi programskog inženjerstva na primjeru aplikacije za pronalazak potencijalnih suigrača“, u primjeru izrade jednostavne aplikacije, prikazuje najvažnije te neophodne korake koje se koriste tijekom planiranja i izrade programske podrške. Izrada jednostavne aplikacije podrazumijeva da autor ne prikazuje detaljno sve elemente programskog inženjerstva, nego ukazivanje na opće smjernice te bitnu ulogu kojom se postiže kvalitetna programska podrška. Rad se može podijeliti na tri bitna djela:

1. Uvod u osnove programskog inženjerstva,
2. Primjena istih na osmišljavanju plana kreiranja programske podrške i
3. Realizacija korisničkih zahtjeva.

Rad se na početku osvrće na kratkoj povijesti nastanka discipline i počinje uvodom u opće smjernice programskog inženjerstva kojima upoznaje čitatelja s glavnim alatima i atributima za postizanje kvalitetne programske podrške. Kada je čitatelj upoznat s glavnim elementima uspješne izrade softvera, rad će te iste moći usporediti na konkretnom primjeru osmišljavanja i izrade aplikacije pronalaskom potencijalnih suigrača u sportu. Kada se radi na projektu izrade softvera, ključ uspjeha leži na samom početku projektnog stvaranja. Kako tijekom faze stvaranja ne bi došlo do zabune, potrebno je korisničke zahtjeve detaljno razložiti te opisati. Posljednja je faza - faza stvaranja. Nakon što su korisnički zahtjevi napisani te dijagrami procesa nacrtani, u radu će se autor osvrnuti na realizaciju istih. Izradi programske podrške pristupit će se agilnim modelom koji je danas najzastupljeniji model programskog inženjerstva. Glavni je aspekt agilnog pristupa sprint, nakon kojeg postoji potencijalno razvijen program koji može krenuti u izvedbu. U ovom završnom radu bit će predstavljena prva od dvije iteracije te će se na konkretnom primjeru realizirati korisnički zahtjevi koji će kasnije biti provjereni i testirani.

Ključne riječi: *programsko inženjerstvo, programska podrška, modeliranje sustava, baza podataka, aplikacija.*

Sadržaj

1. Uvod

Kako se i sam svijet razvija nepobitno je da ga tehnologija prati. Svjedoci smo tomu da je internet doveo do ogromnog oblikovanja svijeta u kojem trenutno živimo. Samim napretkom, zahtjevi za softverskom potporom rastu. Kada je u pitanju konkurencija, presudni su kvaliteta i pouzdanost. Kako bi se razvio efikasan softver, bitne su dobre metode i alati.

Stvaranje softvera mlada je disciplina koja se razvija svakim danom. Metode i principi programskog inženjerstva, koje prije nisu bile poznate, rezultirale su propadanjem projekta, i to zbog prekoračenja vremena ili budžeta. Sukladno tomu, razvija se disciplina koja tijekom aktivnog strukturiranja, provjere, dizajniranja, testiranja i održavanja danas zovemo programsko inženjerstvo.

U ovom radu prikazat će se primarni principi programskog inženjerstva na praktičnom primjeru izrade aplikacije za pronalazak potencijalnih suigrača. Kako bi prikazao temelj i glavne aspekte programskog inženjerstva, rad će se fokusirati na osnovu.

Kako bi dao smjernice za bolje percipiranje same discipline, u drugom poglavlju rada autor će se osvrnuti na glavne pojmove softverskog inženjerstva te kratkoj povijesti nastanka discipline. Nakon kratke povijesti i ključnih pojmova rad će upoznati čitatelja s glavnim alatima i atributima koji se koriste za kvalitetnu izradu programske podrške. Na kraju drugog poglavlja naglasak je na planiranju i testiranju koji predstavljaju temelj uspješne izrade softvera.

Naglasak je u trećem, četvrtom i petom poglavlju na konkretnom primjeru izrade programske podrške praktičnim primjerom izrade aplikacije za pronalazak potencijalnih suigrača u kojim se primjenjuju metode i alati opisani u drugom poglavlju. Rad se u ovim poglavljima razrađuje teorijski, a u uvodu govori o kreiranju same aplikacije.

Posljednje šesto i sedmo poglavlje razrađuje definirane korisničke zahtjeve, navedene u poglavljima prije, i to u izradi jednostavnog korisničkog sučelja. Korisničko sučelje pružat će korisnicima jednostavno i lako korištenje funkcionalnosti aplikacije. Na kraju rada vrše se testovi kako bi se uvjerili da nije došlo do pogreške.

2. Programsko inženjerstvo

U ovom poglavlju autor rada osvrnut će se na teorijsku osnovu programskog inženjerstva te pobliže opisati njezine principe i svrhu nastanka.

2.1. Programska potpora

Kako bi se definiralo programsko inženjerstvo te istaknula njegova svrha, neophodno je objasniti pojam programske podrške. Programska potpora ili softver (engl. software) skup je programa koji nema fizikalnih dimenzija te je neophodan za izvođenje računalnih naredbi. Konkretnije, programska je potpora, kao što i sama riječ pojašnjava, skup objekata u jedinstvenoj kombinaciji koju čine podaci, dokumentacija i računalni programi, potrebni za uspješno rješavanje računskih procesa.

2.2. Povijest programskog inženjerstva i njena definicija

Pojam *programsko inženjerstvo* (engl. Software Engineering) pojavljuje se 1968. i 1969. na konferenciji NATO-ova saveza kako bi se razriješili nagomilani problemi programske podrške u velikim poslovnim sustavima [1]. Na konferenciji je utvrđeno da je zbog „krize programske podrške“ (engl. Software crisis) [1] potrebna primjena inženjerskog pristupa radi unapređenja kvalitete i efikasnosti programske potpore i smanjenja troškova razvoja.

Samostalne, jednokorisničke aplikacije pojavljuju se početkom devedesetih, nakon čega je uslijedio razvoj poslužiteljskih aplikacija nad bazom podataka. Sve bržim razvojem računalnih sadržaja povećava se tendencija povećanja kvalitete, kao što su točnost, održivost, pouzdanost te smanjenja troškova razvoja korisnijih softvera. Samim time, krajem devedesetih uslijedio je razvoj klijentskih aplikacija s odvojenom bazom podataka na serveru. Usavršavanjem dolazi do razvoja internetskih aplikacija, nakon kojeg se softver proširuje na distribuirane i mobilne aplikacije.

Potrebno je naglasiti kako programsko i računalno inženjerstvo nije jednako. U računalnom inženjerstvu naglasak je na teorijske osnove dok je programsko inženjerstvo

disciplina koja se služi metodama i alatima za profesionalno dizajniranje i proizvodnju softvera.

Drugim riječima, programsko se inženjerstvo definira kao inženjerska disciplina kojoj je primarni cilj evolucija programskog proizvoda, pri čemu se koriste određene metode: izrada specifikacije (analizom zahtjeva), njegove implementacije i potvrđivanje valjanosti kojom se unapređuje programski proizvod.

2.3. Aktivnosti, faze programske podrške

Poznato je kako postoje razni procesni modeli postupanja razvoja programske podrške, a temelj tih modela trebao bi biti zasnovan na sljedećim pet aktivnosti:

- specifikacija: prikupljanjem zahtjeva od klijenta, uz detaljnu analizu formuliraju se specifikacije koje sadrže upute za rad programske podrške,
- oblikovanje: faza dizajniranja sučelja uz definiranje programske podrške,
- implementacija: faza programiranja, pomoću programskih alata i jezika kojoj je glavni cilj slijediti izvedenu specifikaciju te ju prenijeti u programski kod,
- verifikacija i validacija: testiranje programske podrške: uspoređuje se radi li programska podrška kako je definirana specifikacijom te
- održavanje ili evolucija programske potpore: vezano je za sve nove zahtjeve te prilagodbe istih.

2.4. Svojstva programske podrške i njezini atributi

Svaki bi softver trebao težiti postizanju konkurentne prednosti te na zahtjeve okoline odgovarati kvalitetno, brzo i redovito, a to se postiže kontinuiranim poboljšanjem kvalitete atributa programske podrške; neki od tih atributa prikazani su na slici 1, koja prikazuje bitna svojstva za kvalitetu programske podrške. Ključni su atributi: [1]:

- mogućnost održavanja: programska podrška treba se modificirati, drugim riječima, treba se prilagoditi zahtjevima poslovnih procedura,
- pouzdanost i sigurnost: programska podrška mora biti predvidiva te ne smije izazvati nikakvu štetu, bila ona fizikalna ili ekonomska,

- efikasnost: označava da programska podrška koristi resurse na štedljiv i uravnotežen način, sukladan performansama softvera,
- upotrebljivost: efektivna primjena postojećih resursa. Drugim riječima, ako već postoji softver, treba ga modificirati i prilagoditi, ako je to prikladno.

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Slika 1: Atributi kvalitete [1]

Ubrzanim rastom tehnologije raste i konkurencija koja u velikoj mjeri pritišće proizvođače softvera te to utječe na krajnji proizvod. Bitno je istaknuti kako nijedan sustav nije idealan te ne može sadržavati (slika 1) sve navedene atribute. Možemo reći kako je stvaranje programske podrške u velikoj korelaciji s kreativnošću, a tijekom same izrade iskustvo je bitan i poželjan faktor. Svaki kupac definira softver na drugačiji način, prikladan performansama koje želi da softver sadrži. Zato je presudno da krajnji korisnik točno zna što želi, kako se zahtjevi mogu što uspješnije kreirati, i na temelju kojih će programska podrška raditi.

Prema [2] ključna su i međusobno ovisna tri faktora: kvaliteta, cijena i vrijeme. Postoji mogućnost da dva od tri faktora budu zadovoljavajuća, ali ne postoji mogućnost da se napravi sustav sa sva tri.

Slika 2: Omjer kvalitete, cijene i vremena [2]

Iz slike 2 jasno je vidljiva veza među tri faktora te će se korisnik, kojemu je bitan kvalitetan softver, morati odreći brzine ili cijene izrade. Za klijenta, kojemu je bitno da programska potpora bude povoljnija, morat će žrtvovati brzinu ili kvalitetu. Primatelj koji želi brže napravljen softver morat će odlučiti želi li skupo platiti softver, te tako zadržati faktor kvalitete ili će se željeti odreći kvalitete, što će rezultirati jeftinijem softverom [2].

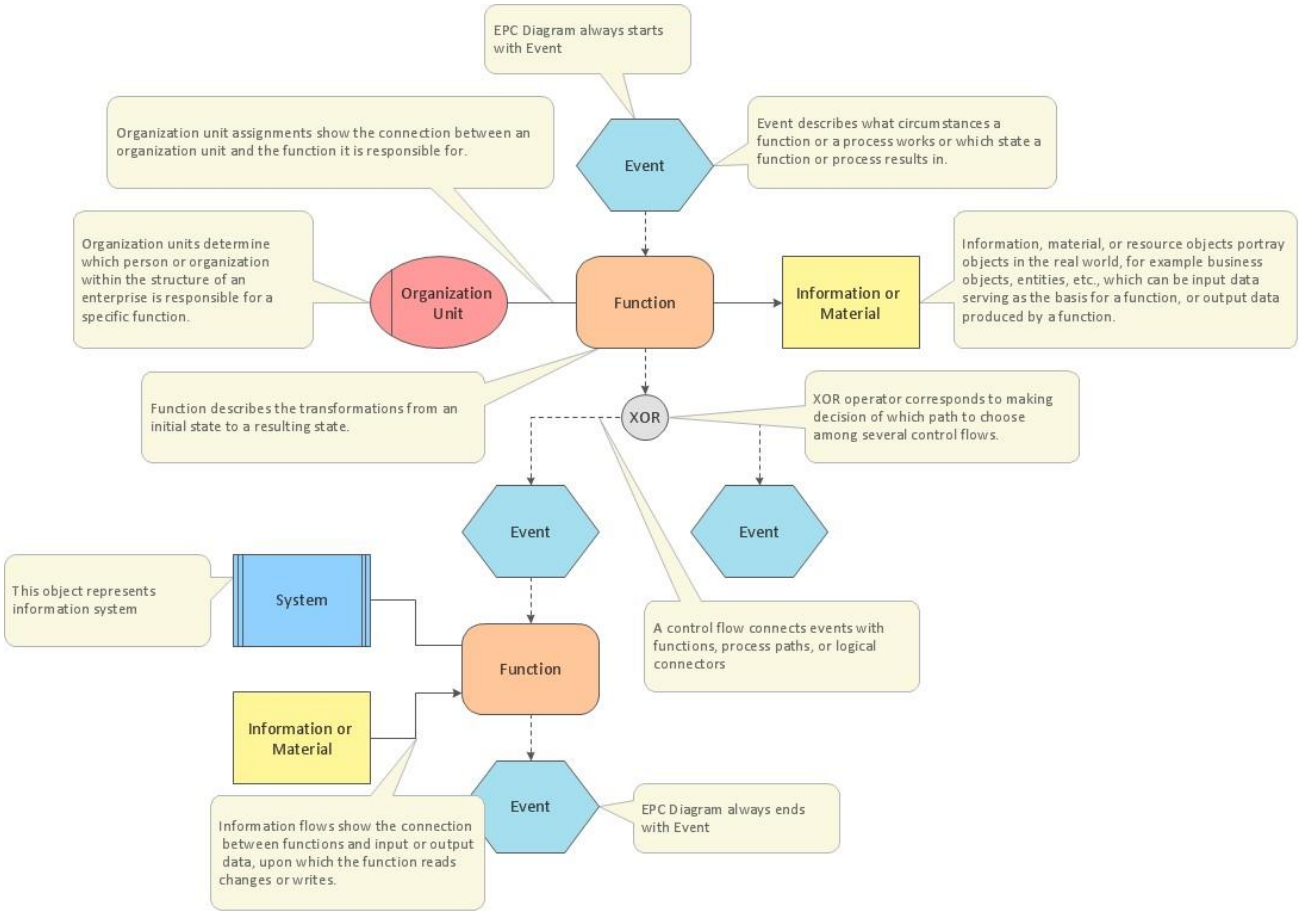
2.5. Modeliranje UML i EPC dijagramima

Razvoj složenih softverskih sustava kompleksan je proces koji ponekad potraje godinama. Ako bi kvalitetu softvera provjeravali tek na kraju razvojnog procesa, to bi najvjerojatnije bilo prekasno. Pod pretpostavkom da softver ne zadovoljava planirane zahtjeve kvalitete, cijene ili vremena, gubici mogu biti veliki. Postoji način kako izbjeći ovaj problem - pravilnim rukovođenjem i provjeravanjem svake faze razvojnog procesa. Tako sprječavamo pogreške koje smo napravili s ciljem da ih prosljeđujemo u naredne faze. Realizacija različitih procedura tehnika i alata ključ je za proizvodnju kvalitetne programske podrške [3].

Nakon što su prikupljeni zahtjevi te posjedujemo sve ključne podatke potrebne za razvoj programske podrške, potreban je pouzdan plan. Postoje mnoge tehnike i metode uspješnog planiranja, a u ovom će se radu proučiti grafički prikaz UML i EPC dijagrama.

EPC dijagram (engl. *Event Driven Process Chain*) jedan je od najučestalijih grafičkih modela koji pruža logički pregled dinamičnih lančanih događaja uvjetovanim predodređenim zahtjevima. Slika 3 prikazuje EPC dijagram koji je vođen događajima za poboljšanje u cijeloj organizaciji. Pomoću njega se izrađuje model koji smanjuje vrijeme

potrebno za izradu modela poslovnog procesa. Slika pobliže opisuje elemente koji se koriste za izradu EPC dijagrama.



Slika 3: EPC dijagram [4]

2.6. Modeli izrade softvera

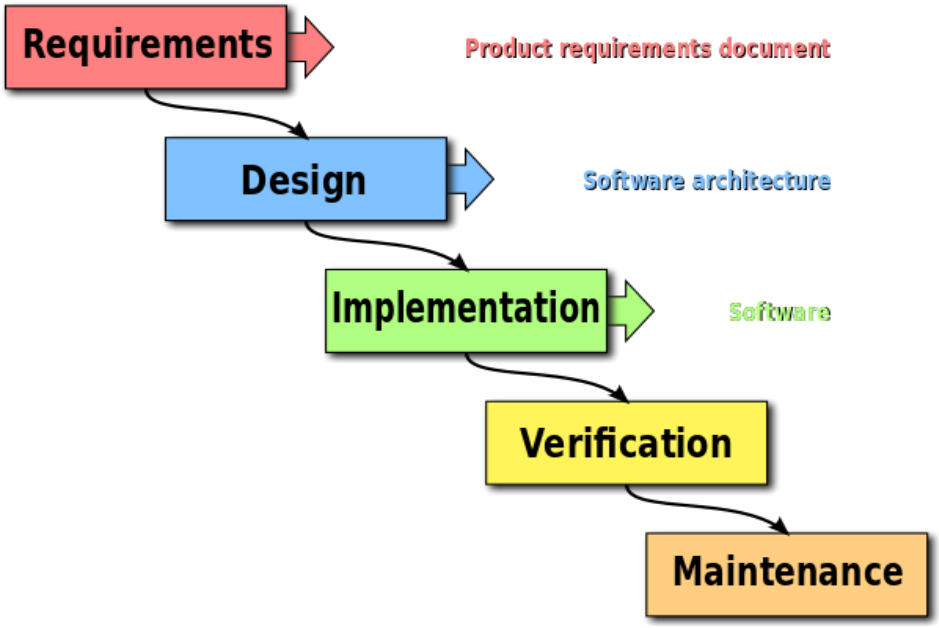
Modeli procesa softvera potrebni su kako bi počela proizvodnja programske podrške na najučinkovitiji način. Uvedeni su na osnovama saznanja razvojnih timova prioritiziranjem odvijanja projektnih aktivnosti. Samim time inženjeri su tijekom vremena razvili brojne modele, a u ovom dijelu bit će opisana tri: vodopadni, spiralni i agilni koji će se kasnije koristiti za usporedbu procesa izrade aplikacije.

2.6.1 Vodopadni model

Model vodopad (engl. waterfall) specifičnog je lančanog oblika u kojemu su faze samostalne, odnosno koraci slijede jedan za drugim. Nova faza započinje tek kada je prijašnja gotova, a može se promatrati sljedeće nezavisne faze [5]:

- izlučivanje zahtjeva,
- oblikovanje,
- implementacija,
- integracija,
- održavanje sustava.

Iako je metoda vodopad lagana za razlučivanje i korištenje, zbog njegove sekvencijalnosti, model ne dozvoljava izvršavanje više faza istovremeno. Posljednja je faza - faza testiranja te, ako se otkriju problemi sa softverom, dolazi do gubitka vremena i resursa, zato što isti nisu detektirani u prijašnjim razvojnim fazama. Zbog njezine neprilagodljivosti dodatnim korisničkim zahtjevima [6] nije idealna za veće i kompliciranije programske podrške.

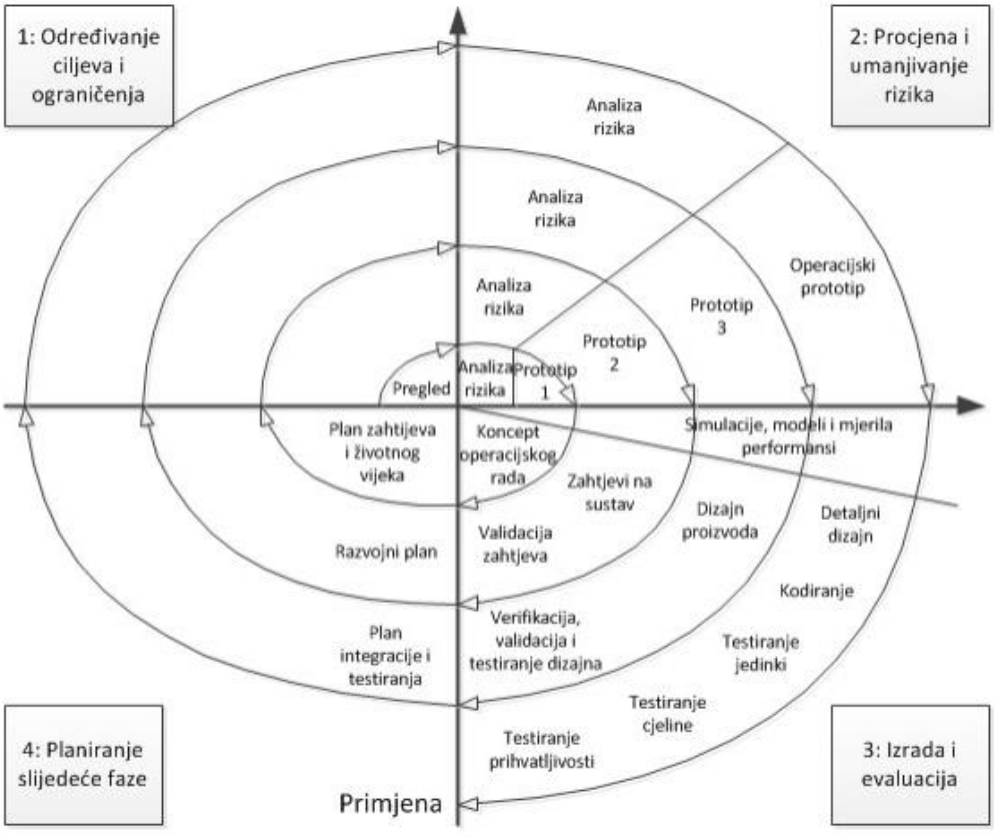


Slika 4: Model vodopada [7]

2.6.2 Spiralni model

Možemo reći da je spiralni model apsolutna suprotnost vodopadu. Ovaj model podijeljen je u četiri faze od kojih je ključna faza analize rizika. Programska podrška neprekidno prolazi kroz sve četiri faze u iteracijama nazvane spirale [8]. Opis spiralnih faza prema [2]:

- faza planiranja - određivanje zahtjeva softvera,
- analiza rizika – ispitivanje rizika vezanih za projekt te pronalazak alternativnih rješenja. Potrebno je napraviti prototip kako bi se uspješno postigli ciljevi,
- faza inženjerstva – prototip kreiran u prijašnjoj fazi koristi se za evaluaciju rješenja te dolazi do razvoja i testiranja softvera,
- faza evaluacije – odobravanje projekta, propitkivanje zadovoljstva krajnjih korisnika dosadašnjim napretkom. Prelazak u drugi dio spirale gdje prolazi kroz gore navedene faze.



Slika 5: Spiralni model razvoja [9]

2.6.3. Agilni model

Ovaj model utemeljen je na agilnom manifestu. Dizajniran je na uočenim problemima tradicionalnih modela poput vodopada. Utemeljen je na komunikaciji proizvođača i korisnika od rane faze početka razvojnog ciklusa. Ključni je element agilnog pristupa testiranje koje počinje na samome početku razvoja softvera te se nastavlja sve do samoga kraja [10].

Agilne su metode formirane kao odgovor na formalne procese u kojem je fokus na što ranijem dostavljanju radnog softvera korisniku koji može predlagati i mijenjati zahtjeve. Ti zahtjevi bit će dodani u nadolazećim iteracijama sustava. „Expect the System Requirements to Change and so Design the System to Accommodate these Changes.“ [1].

	Agile Methods	Heavy Methods
Approach	Adaptive	Predictive
Success Measurement	Business Value	Conformation to plan
Project size	Small	Large
Management Style	Decentralized	Autocratic
Perspective to Change	Change Adaptability	Change Sustainability
Culture	Leadership-Collaboration	Command-Control
Documentation	Low	Heavy
Emphasis	People-Oriented	Process-Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Upfront Planning	Minimal	Comprehensive
Return on Investment	Early in Project	End of Project
Team Size	Small/Creative	Large

Slika 6: Razlike u agilnoj i tradicionalnoj metodologiji [11]

2.7. Testiranje

Razvojem tehnologije tržište zahtjeva sve složenije, modernije i besprijekorne softvere, a samim time klijenti podrazumijevaju da oni budu bez greške. Kada govorimo o pogreškama, bez obzira jesu li male ili velike, one znatno mogu utjecati na cijeli projekt. Ako je pogreška detektirana prekasno, dolazi do prekoračenja proračuna ili vremenskih rokova. Kako bi se to spriječilo i na vrijeme reagiralo, koristi se testiranje. Prema [5] pogreška dio je stanja sustava koje je izazvalo ponašanje koje nije bilo u skladu sa željenim.

Kako bi se ispitala točnost, efikasnost i kvaliteta samoga softvera, u izradi programske podrške, testiranje softvera jedan je od ključnih procesa. Testiranje omogućuje provedbu prvobitno zacrtanih zahtjeva detektiranjem pogreške, čime se postiže isporuka kvalitetnog proizvoda.

Kako bi testiranje softvera pružalo adekvatan produkt, potrebno je formulirati pravilnu strategiju testiranja te na temelju dobivenih ishoda prilagoditi dobivene rezultate očekivanim. Temelj testiranja provodi se sastavljanjem scenarija koji sadrže što se treba testirati. Kako bismo izabrali adekvatnu strategiju prema [6] trebamo slijediti sedam načela testiranja:

- iscrpno testiranje je nemoguće; to je testiranja svih mogućih funkcionalnosti ulaznih podataka. Samim time gubi se na vremenu i resursima. Zbog toga je potrebno uračunati prihvatljiv rizik koji može nastati u radu softvera.
- klasteri podrška – većina grešaka javlja se u 20% sektora te predstavlja 80% pogrešaka što i opisuje Paretovo načelo. Osoba koja vrši testiranje na temelju iskustva može predvidjeti gdje će se pogreška pojaviti.
- paradoks pesticida – ako softver testiramo sličnim testovima, on će postati otporan te se detekcija pogrešaka neće pojavljivati, a pogreške postoje. Kako bi se to izbjeglo, potrebno je softver testirati različitim testovima.
- testiranje ukazuje na nedostatke – ako se tijekom testiranja pogreške ne javljaju, pogrešaka nema.
- izostanak pogreške - zabluda - ako se softver testira za krivu svrhu, postoji mogućnost da i dalje neće raditi, iako je većina pogrešaka detektirana.
- rano testiranje - rano testiranje omogućuje brži razvoj i uštedu resursa.
- testiranje ovisi o kontekstu - bitna je namjena softvera te se testiranje mora tome prilagoditi.

S tradicionalnog gledišta testiranje se može podijeliti u tri glavne faze:

- jedinično testiranje,
- integracijsko testiranje,
- sistemsko testiranje.

2.7.1. Jedinično testiranje

Osnova jediničnog testiranje provjera je ispravnosti pojedinačnih dijelova sustava. Kod jediničnog testiranja, testiranje obavlja implementator. Može testirati liniju koda pa sve do nekoliko stotina ili tisuća te se gleda može li samostalna jedinica funkcionirati i pozvati druge samostalne jedinice. Velika je prednost ove vrste testiranja to što će se pronađena pogreška odnositi samo na jedinicu. Takvu pogrešku puno je lakše ispraviti [2].

Prema [2] ono je zasnovano na jednostavnom principu: Ako jedinica, koja se testira zadovoljava testni scenarij, onda možemo reći da ta ista jedinica, postavljena u veći skup za koji važe iste ili slične karakteristike, djeluje na jednak način.

2.7.2. Integracijsko testiranje

Nakon što je završeno jedinično testiranje prelazi se na spajanje jedinica u veću cjelinu. Integracijsko testiranje ima za cilj otkrivanje pogrešaka koje su nastale spajanjem pojedinih dijelova. Iako jedinice zasebno rade i zadovoljavaju prijašnje testove, postoji močenost dok se spoje u zajednički skup kako neće raditi.

Prema [2] integracijsko testiranje fokusira se na četiri stvari:

- ispitivanje o tome egzistiraju li iznenadne i suvišne interakcije među komponentama u nekom sustavu,
- ispitivanje koje otkriva zadovoljavaju li komponente korisnikova očekivanja,
- ispitivanje sigurnosti sustava te reakcija na različita okruženja u kojima će se nalaziti,
- ispitivanje odzivnosti, propusnosti, sigurnosti i ostalih atributa kvalitete sustava.

2.7.3. Sistemsko testiranje

Nakon uspješno odrađenog integracijskog testiranja slijedi sistemsko. Sistemsko testiranje sadrži ispitivanje sustava kao cjeline s glavnim ciljem zadovoljavanja korisničkih zahtjeva. Drugim riječima, u ovom dijelu testiranja pokušavaju se otkriti

pogreške koje do sada nisu bile vidljive, testira se cijeli softver te se provjerava kako sve komponente sustava rade zajedno.

Sistemskim testiranjem testiraju se performanse, efikasnost, točnost, otpornost i učinkovitost, kako bi se osiguralo da sve komponente softvera rade besprijekorno ono za što su predviđene korisničkim zahtjevima, a prijašnja testiranjima to nisu mogla dokazati [12].

2.7.4. Standardi

Standardi su definirani propisi koje odobravaju ovlaštene organizacije za kontrolu i usklađivanje procesa, a samim time značajni su i u procesu testiranja. Oni omogućavaju naputke koje služe kao smjernice za kreiranje kvalitetnog proizvoda. Međunarodne standarde donose sljedeće organizacije [6]:

- Međunarodna organizacija za standardizaciju (engl. International Standardization Organization, ISO),
- Institut inženjera elektrike i elektrotehnike (engl. Institute od Electrical and Electronic Engineers, IEEE),
- Međunarodna elektrotehnička komisija (engl. Commission électrotechnique internationale, IEC).

Prema [6] za testiranje softvera koristi se ISO/IEC/IEEE 29119. „Primjenom pojedinog standarda proizvod je ujednačen s ostalim takvim proizvodima proizvedenim bilo gdje u svijetu, u skladu s istim standardom. Ako je riječ o testiranju softvera, usuglašenost s međunarodnim standardom jamči da je on testiran pod jednakim uvjetima. ISO/IEC 29119 standard je za testiranje softvera i odnosi se na sve faze razvojnog ciklusa softvera koju provodi bilo koja organizacija.“ Prema [6] ISO/IEC 29119 standard se može podijeliti na sljedeće potkategorije:

- ISO/IEC 29119 -1:2013 Koncepti i definicije,
- ISO/IEC 29119-2:2013 Postupci testiranja,
- ISO/IEC 29119-3:2013 Dokumentacija o testiranju,
- ISO/IEC 29119-4:2015 Tehnike testiranja,
- ISO/IEC 29119-5:2016 Testiranje pogonjeno ključnim riječima.

Neophodno je spomenuti ISO/IEC 9126 standard koji se oslanja na programsko inženjerstvo i kvalitetu proizvoda. On se sastoji od [13]:

- modela kvalitete,
- eksternih mjerenja,
- internih mjerenja,
- mjerenja kvalitete u upotrebi.

3. Strukturiranje programske podrške

Nakon što su temeljne osnove programskog inženjerstva objašnjene, rad će se osvrnuti na primjenu istih na konkretnom primjeru. Od ovog poglavlja, nadalje, bit će prikazani osnovni principi programskog inženjerstva na konkretnom primjeru konstruiranja aplikacije za pronalazak potencijalnih suigrača.

3.1. Podloga za ideju

Razvoj interneta doveo je do ogromnog napretka u protoku informacija. Samim razvojem interneta, došli smo do povezivanja svijeta u dosad nedostižnim razmjerima te možemo reći kako ljudi nisu nikada bili toliko povezani kao danas, i to zahvaljujući tehnologiji. Kako se svijet mijenja, drugim riječima „raste“, nepobitno je da tehnologija poveliko utječe na okolinu i društvo u kojem trenutno živimo.

Svatko od nas upravlja svojim vremenom i o nama ovisi kako ćemo iskoristiti ta dvadeset i četiri sata. Mnogi su se zbog razvoja tehnologije zatvorili u virtualni svijet i oni sami odlučuju koliko ga žele konzumirati. Kako bi pojedinac i grupa imali različite mogućnosti provođenja slobodnog vremena tijekom dvadeset i četiri sata, osmišljena je idejna aplikacija koja pruža pronalazak potencijalni suigrača u sportu.

Drugim riječima, ako je osoba koja se rekreativno ili profesionalno bavi sportom u situaciji da mu je potreban suigrač, koji dijeli slične ili iste interese, on će zahvaljujući aplikaciji upoznati takvu osobu.

3.2. Opis projekta

Svi projekti i radovi započinju idejom i ambicijom za poboljšanjem, stvaranjem i željom za promjenom nečeg novog. Osmišljavanje dobre ideje još uvijek nije projekt. Osim ideje, bitno je razviti projekt u kojemu je jasno definirano što želimo a što ne želimo stvoriti.

Temelj ovog projekta je kreirati sustav koji će služiti korisnicima za jednostavno i brzo povezivanje, dogovaranje i rekreaciju u njihovim sportovima i hobijima.

Otvaranjem početnog korisničkog sučelja, korisnik ima pravo koristiti aplikaciju na dva moguća načina. Prva mogućnost je koristiti aplikaciju kao neregistrirani korisnik ili gost; u tom slučaju, nije u mogućnosti koristiti potpune funkcionalnosti softverskog proizvoda, nego ima samo sposobnost pregleda karakteristika aplikacije. Kako bi u potpunosti imao pristup svim specifikacijama programa, korisnik mora ispuniti kratku i jednostavnu registraciju.

Tijekom završetka unosa svih potrebnih podataka, program provjerava ispravnost, te u slučaju krivog unosa javlja pogrešku. Ako nije došlo do pogreške, aplikacija dozvoljava korisniku pristup svim karakteristikama i funkcionalnostima koju ona pruža. Ako je registracija bila uspješna, upisani podatci automatski se zapisuju u bazu podataka u kojoj su spremljeni. Korisnik drugim dolaskom ne treba ispuniti obrazac za registraciju, već obrazac za prijavu s podatcima koje je unio tijekom registracije. Aplikacija, uspoređivanjem istih s podatcima pohranjenima u bazi podataka, automatski provjerava istinitost podataka,

Aplikacija će biti podijeljena u tri glavna dijela. Prvi dio bit će sučelje za kreiranje događaja koji će raditi po jednostavnom principu. Korisnik će trebati ispuniti određena polja podatcima kojima opisuje o kakvom je događaju riječ. Nakon što su obavezna polja ispunjena te korisnik kreira događaj, aplikacija provjerava podatke te, ako je došlo do kakve pogreške, obavještava korisnika da ponovi unos. Ako nije došlo do pogreške, aplikacija unesene podatke sprema u bazu podataka. Kada su podatci spremljeni u bazu podataka, program automatski ispunjava tablicu sportova i hobija te je sada vidljiva svim korisnicima aplikacije.

Drugi će dio aplikacije bit namijenjen trenerima i organizacijama. Princip rada ovog korisničkog sučelja velikim se dijelom podudara s prvim. Treneri i organizacije trebat će navesti dodatne informacije o uslugama koje pružaju, od kojih će to, na primjer, biti cijena usluge i kontakt.

Treći dio aplikacije je za korisnike koji se žele pridružiti već kreiranim događajima, treninzima ili organizacijama. U tom korisničkom sučelju nalazi se tablica svih događaja koja se automatski popunjava podatcima iz baze podataka. Korisnik odabire željeni događaj te u nekoliko klikova može pristupiti kreiranom događaju. Kada je pristupio,

kako bi dogovorio pojedinosti i detalje odvijanja, dobiva obavijest o mogućem korištenju razgovora sa svim korisnicima koji su pristupili istom eventu.

Ocjenjivanje je zadnja stavka koju će pružati aplikacija. Bio to rekreativan korisnik, trener ili organizacija, na svojim će profilima imat prosjek svih danih ocjena. Nakon što svaki događaj završi, korisnici će imati pravo ocijeniti druge korisnike. Ocjenjivanjem se nastoji potaknuti korisnike da budu prijateljski osviješteni i da skupljaju bolje prosjeke.

3.3. Svrha projekta

Svrha je ovog projekta prikazati principe programskog inženjerstva, i to konstruiranjem i osmišljavanjem programske podrške praktičnim primjerom izrade aplikacije za pronalazak potencijalnih suigrača u sportu.

Sama aplikacija bit će osmišljena s ciljem da pomogne ljudima kojima treba suigrač u sportu. Oni će biti u stanju sami odlučiti kako žele koristiti aplikaciju: za pristupanje događaju za pronalazak potencijalnih suigrača i trenera ili za kreiranje događaja. Danas živimo u svijetu u kojem je internet svuda oko nas i postao je svakodnevnica. Glavna je svrha ovog projekta potaknuti ljude diljem svijeta na zdrav život kroz sport i njihove hobije.

3.4. Projektni ciljevi

Za razliku od svrhe, projektni ciljevi moraju biti potpuno konkretni i jasni. Ciljevi su u konstruiranju programske podrške nulta točka za gotovo svaki pothvat i izvršeni posao. Ako se dogode pogreške te se odmakne od projekta, ciljevi će dovesti projektante na pravi put. Projektni ciljevi vode projekt u proces planiranja te će pomoći pri realizaciji projekta. Kako bi se to postiglo, vrlo je bitno na pravilan način definirati ciljeve.

Najistaknutija i jedna od najuspješnijih metoda pristupa postavljanju ciljeva naziva se SMART. Svako slovo predstavlja pridjev koji je akronim na engleskom jeziku te kaže kako cilj mora imati i mora biti:

- S specifičan
- M mjerljiv

- A dostupan i ostvariv
- R realan
- T vremenski prihvatljiv

Ako to primijenimo na projekt, to će izgledati ovako:

- Specifičan → kreirati aplikaciju za pronalazak potencijalnih suigrača u sportu. Pristupiti već kreiranom događaju.
- Mjerljiv → u prvoj iteraciji kreiranja aplikacije zadovoljiti četiri od devet funkcionalnosti koje pruža aplikacija
- Ostvarivo i realno → u ovom djelu bitno ne postaviti previsoke ciljeve. Odnosno, aplikacija za pronalazak potencijalnih suigrača u sportu bit će jednostavna za krajnje korisnike te će oni u samo par klikova potpuno moći koristiti funkcionalnosti aplikacije.
- Vremenski prihvatljiv → projekt će biti podijeljen na dvije iteracije koje će trajati po mjesec dana. Nakon prve iteracije korisnici će moći koristiti aplikaciju u početnoj verziji.

4. Definiranje zahtjeva

Definiranje zahtjeva jedna je od temeljnih procesa dizajniranja i osmišljavanja programske potpore. Možemo reći kako je prikupljanje zahtjeva, tijekom konstruiranja softvera, prva komponenta aktivnosti. Oni percipiraju što će programski sustav trebati raditi, a ujedno su opisana i ograničenja koja treba izbjeći. Zahtjevi ovise o veličini projekta te o kompliciranosti elemenata koji ga čine. Prema [2] karakteristike dobro definiranih zahtjeva su: jasnoća, sažetost, laka razumljivost, razumnost, povjerljivost, međusobno međudjelovanje.

U skladu s navedenim, autor će se u radu u ovom poglavlju posvetiti prvo prikupljanju i opisivanju zahtjeva konkretnim slučajevima primjene, a potom će iste detaljnije razraditi i opisati raznim detaljnim slučajevima. Naposljetku poglavlja rad će se osvrnuti na korisnike aplikacije i na koje će skupine biti podijeljeni. Razlika između pojedinih skupina utjecat će na dozvolu i upotrebu funkcionalnosti koje će korisnik aplikacije moći koristiti.

4.1. Slučajevi primjene

Tijekom prikupljanja zahtjeva, zahtjeve možemo podijeliti u pet kategorija: poslovne, korisničke, funkcionalne, nefunkcionalne i implementacijske. U ovom radu razradit će se tri slučaja primjene od kojih će se koristiti dvije od gore navedenih pet kategorija, a to su korisničke i funkcionalne. Slučajevi primjene činit će temelj razrade i konstruiranja programske podrške, a kasnije će raznim dijagramima i modelima činit punu sliku, tj. programsku podršku.

4.1.1. Slučaj primjene 1

Prvi slučaj primjene koristio bi krajnjim korisnicima u pronalasku potencijalnih suigrača ili trenera i organizacija. Ovisno o želji i potrebi korisnika, odabirom kategorije potencijalnih suigrača ili kategorije, treneri i organizacije aplikacija ispisuju sve događaje zapisane u bazi podataka u tablicu koja je vidljiva svim korisnicima aplikacije. Korisnik ima i mogućnost filtriranja dobivenih podataka. Odabirom željene aktivnosti, korisnik označuje događaj te se istom pridružuje. Kada je korisnik član tog događaja, kako bi

saznao detalje odvijanja događaja, dobiva mogućnost korištenja razgovora tekstualnim putem.

4.1.2. Slučaj primjene 2

Drugi slučaj primjene koristio bi krajnjim korisnicima u kreiranju događaja. Korisnik klikom na *kreiraj događaj* otvara sučelje. Sučelje sadrži pet tekstualnih polja koja korisnik mora ispuniti kako bi kreirao događaj. Tekstualna polja sadrže sljedeće podatke: naziv sporta, mjesto održavanja, broj osoba koje se mogu prijaviti, vrijeme održavanja i datum održavanja događaja. U slučaju da korisnik krivo unese podatke, aplikacija javlja pogrešku te je potrebno ponoviti unos. Uspješnim unosom svih podataka korisnik kreira događaj te se podatci automatski spremaju u bazu podataka. Kada su podaci uspješno spremljeni, aplikacija samostalno osvježava i generira novonastali događaj koji je vidljiv u tablici svih događaja.

4.1.3. Slučaj primjene 3

Treći slučaj primjene koristio bi trenerima i organizacijama za kreiranje događaja. Korisnik klikom na *kreiraj događaj* otvara sučelje. Ono sadrži pet tekstualnih polja koja korisnik mora popuniti kako bi kreirao događaj.

Tekstualna polja sadrže sljedeće podatke:

- naziv sporta,
- mjesto održavanja,
- vrijeme održavanja,
- kontakt trenera ili organizacije,
- cijenu usluge.

U slučaju da korisnik krivo unese podatke, aplikacija javlja pogrešku te je potrebno ponoviti unos. Uspješnim unosom svih podataka korisnik kreira događaj te se podatci automatski spremaju u bazu podataka. Kada su podaci uspješno spremljeni, aplikacija samostalno osvježava i generira novonastali događaj koji je vidljiv u tablici svih događaja.

4.2. Detaljni slučajevi primjene

U ovom odlomku tri navedena slučaja primjene bit će detaljno opisana. Detaljan opis podijelit će se na sljedeće potkategorije i načine rada:

- što korisnik treba učiniti kako bi pokrenuo slučaj primjene,
- preduvjeti koje mora ispuniti,
- osnovni put kojim će to postići,
- rezultat koji bi trebao dobiti,
- što će se dogoditi ako dođe do pogreške.

4.2.1. Slučaj primjene 1: Za pronalazak potencijalnih suigrača ili trenera i organizacija

Nakon uspješno obavljene prijave, korisnik klikom na link *potvrdi* započinje rad u aplikaciji. Za ovaj slučaj primjene potrebno je izabrati jednu od dvije opcije. Opcija jedan, za pronalazak potencijalnih suigrača ili opcija dva, za pronalazak trenera i organizacija.

4.2.1.1. Preduvjeti

Korisnik za prijavu u sustav mora imati važeće ime i lozinku. Moguće je korištenje sustava i bez prijave, ali time korisniku neće biti dostupne sve funkcionalnosti i opcije koje nudi aplikacija.

4.2.1.2. Osnovni put

Ovisno o željenom rezultatu, postoje dvije mogućnosti:

- početna stranica sustava → odabir opcije *pristupi događaju* → popunjavanje tekstualnih polja potrebna za filtraciju podataka → pregled aktivnih događaja u tablici događaja → označavanje i pristupanje događaju
- početna stranica sustava → odabir opcije *pristupi treningu* → popunjavanje tekstualnih polja potrebna za filtraciju podataka → pregled aktivnih događaja u tablici događaja → označavanje i pristupanje događaju

4.2.1.3. Rezultat

Na temelju ispunjenih tekstualnih polja informacijama, sustav filtrira sve „potencijalne suigrače“ ili „potencijalne trenere i organizacije“ koji zadovoljavaju kriterije te korisniku na sučelju ispisuje filtrirane podatke

4.2.1.4. Alternativno

U slučaju pogrešnog unosa ili nepotpunog popunjavanja tekstualnih polja, korisnik dobiva rezultate sukladne njegovim odgovorima u kojima postoji mogućnost nepotpunih rezultata. Korisniku je omogućen ponovni unos podataka te filtriranje istih.

4.2.2. Slučaj primjene dva: Za kreiranje događaja

Nakon uspješne obavljene prijave korisnik klikom na link *potvrdi* započinje rad u aplikaciji. Za ovaj slučaj primjene potrebno je izabrati opciju *kreiraj događaj*. Korisniku se otvara sučelje kreiranja događaja te popunjava tekstualna polja kako bi postavio uvjete događaja.

4.2.2.1. Preduvjeti

Korisnik za prijavu u sustav mora imati važeće ime i lozinku. Moguće je korištenje sustava i bez prijave, ali time korisniku neće biti dostupne sve funkcionalnosti i opcije koje nudi aplikacija.

4.2.2.2. Osnovni put

Početno korisničko sučelje → odabir opcije *kreiraj događaj* → popunjavanje tekstualnih polja potrebna za postavljanje uvjeta događaja koji se kreira → pregled aktivnih događaja u tablici događaja → Mogućnost uređivanja, brisanja i ponovnog unosa događaja.

4.2.2.3. Rezultat

Na temelju unesenih tekstualnih informacija o događaju, sustav provjerava korektnost i točnost unesenih informacija. Aplikacija kreira uneseni događaj koji je vidljiv svim korisnicima aplikacije u tablici s popisom svih događaja.

4.2.2.4. Alternativno

U slučaju pogrešnog unosa ili nepotpunog popunjavanja tekstualnih polja, sustav obavještava korisnika kako je došlo do pogreške te je potrebno ponoviti unos.

4.2.3. Slučaj primjene tri: Za trenere i organizacije koje kreiraju događaj

Korisnik klikom na link *potvrdi* započinje rad u aplikaciji nakon uspješno obavljene prijave. Za ovaj slučaj primjene potrebno je izabrati opciju *kreiraj trening*. Korisniku se otvara sučelje kreiranja događaja te popunjava tekstualna polja kako bi postavio uvjete događaja.

4.2.3.1. Preduvjeti

Korisnik za prijavu u sustav mora imati važeće ime i lozinku. Moguće je korištenje sustava i bez prijave, ali time korisniku neće biti dostupne sve funkcionalnosti i opcije koje nudi aplikacija.

4.2.3.2. Osnovni put

Početno korisničko sučelje → odabir opcije *kreiraj događaj* → popunjavanje tekstualnih polja potrebna za postavljanje uvjeta događaja koji se kreira → pregled aktivnih događaja u tablici događaja → Mogućnost uređivanja, brisanja i ponovnog unosa događaja.

4.2.3.3. Rezultat

Na temelju unesenih tekstualnih informacija o događaju, sustav provjerava korektnost i točnost unesenih informacija.

Aplikacija kreira uneseni događaj koji je vidljiv svim korisnicima aplikacije u tablici s popisom svih događaja.

4.2.3.4. Alternativno

U slučaju pogrešnog unosa ili nepotpunog popunjavanja tekstualnih polja, sustav obavještava korisnika da je došlo do pogreške te da je potrebno ponoviti unos.

4.3. Korisnici aplikacije

Tijekom definiranja zahtjeva bitna su stavka korisnici koji će se služiti aplikacijom. Prijeko je potrebno naglasiti koje će funkcionalnosti koja vrsta korisnika imati. U ovom poglavlju rad će pobliže opisati i definirati dvije kategorije korisnika - neregistrirane korisnike i registrirane.

4.3.1. Neregistrirani korisnik

Neregistrirani korisnik ili gost ima mogućnost pregleda aplikacije i njezinih funkcionalnosti, ali nije u mogućnosti koristiti se istima. Kako bi u potpunosti koristio aplikaciju te imao mogućnost kreiranja i pristupanja događajima, mora obaviti jednostavnu registraciju. Samim time postiže se veći nivo sigurnosti i jednostavnija kontrola tijeka podataka.

4.3.2. Registrirani korisnik

Registrirani korisnik obavlja registraciju jedanput te svakim sljedećim dolaskom prijavljuje se u aplikaciju. Taj korisnik ima potpuni pristup pregleda i korištenja svih opcija i funkcionalnosti koje pruža aplikacija, a njih možemo podijeliti u sljedeće tri potkategorije:

- korisnik koji kreira događaj - takav korisnik mora ispuniti kratki upitnik o vrsti događaja kojeg kreira,

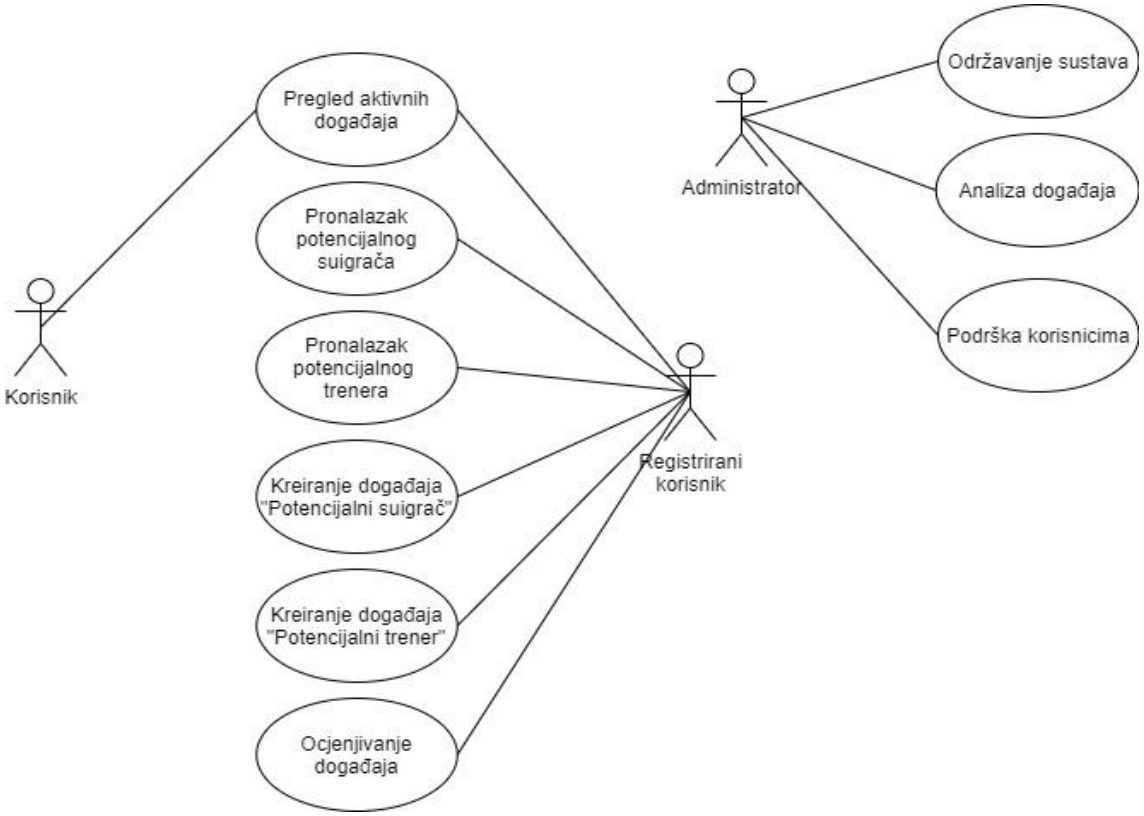
- korisnik koji se želi pridružiti događaju - takav korisnik mora znati što ga interesira te sukladno tome odabire kategoriju pridruživanja sportu, organizaciji ili odabiru osobnog trenera,
- korisnik koji je trener i želi kreirati događaj - takav korisnik popunjava kratki upitnik o vrsti njegovih usluga.

5. Dijagram procesa

Po završetku definiranja zahtjeva te uspješno prikupljenih korisničkih podataka prva faza je završila. Kako bi se moglo krenuti u sljedeću ključnu fazu, fazu realizacije, potrebno je proizvesti kvalitetan i pouzdan plan. U ovome dijelu rad će se osvrnuti na grafički prikaz koji će pobliže opisati veze između procesa. Grafički prikaz olakšava komunikaciju s krajnjim korisnikom te mu na pojednostavljeni način opisuje kako će aplikacija raditi.

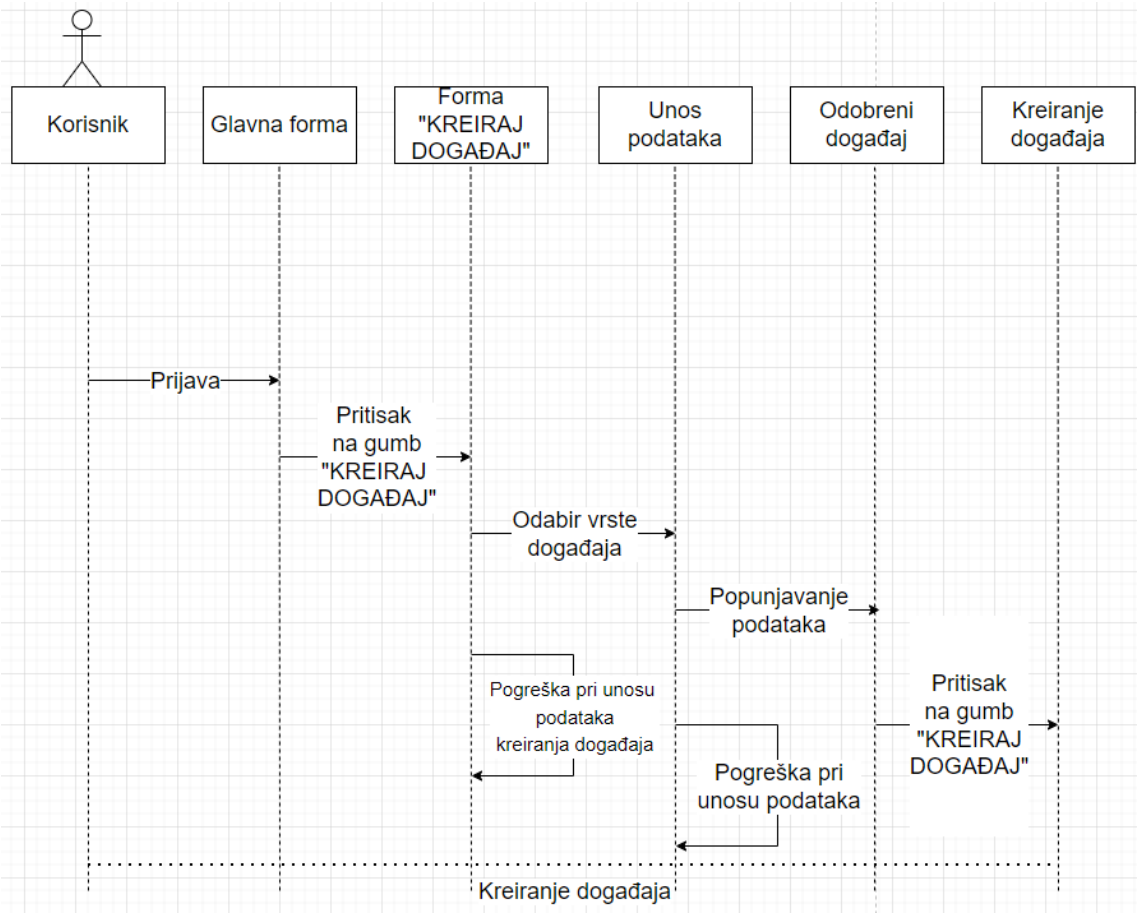
5.1. UML-dijagram

Za grafički prikaz sustava koristit će se dva od mnogih mogućih dijagrama: dijagram slučajeva i dijagram slijeda. Dijagram slučajeva, kao što se vidi na slici sedam, opisuje korisnikovu i administratorovu vezu sa sustavom.



Slika 7: Dijagram slučajeva

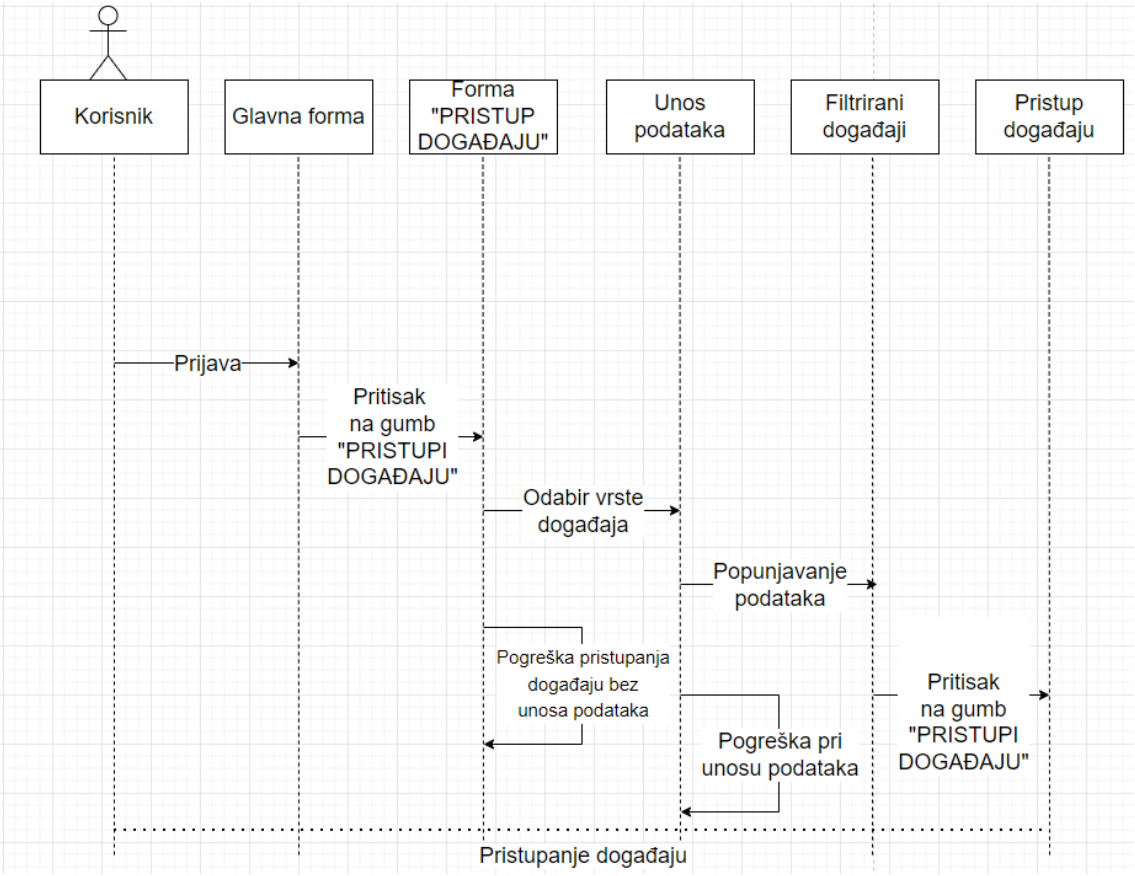
Slika 8 prikazuje dijagram slijeda operacija koje se manifestiraju tijekom kreiranja događaja te pobliže opisuje kako doći do krajnjeg rezultata. Prijavom u sustav korisnik dobiva pravo korištenja svih funkcionalnosti koje aplikacija nudi, te mu se otvara glavna forma aplikacije. Pritiskom na gumb *kreiraj događaj* korisniku se otvara sučelje kreiranja događaja te mora ispuniti sva tekstualna polja potrebna za njegovo kreiranje. U slučaju krivog unosa, korisnik mora ponoviti unos. Ako do pogreške nije došlo, događaj je uspješno kreiran.



Slika 8: Dijagram slijeda kreiranja događaja

S druge strane, slika 9 prikazuje dijagram slijeda, i to kada je događaj kreiran te prikazuje kako mu pristupiti. Korisnik prijavom u sustav dobiva pravo korištenja svih funkcionalnosti koje aplikacija nudi te mu se otvara glavna forma aplikacije. Pritiskom na gumb *pristup događaju*, korisniku se otvara sučelje pristupanja događaju. Korisnik

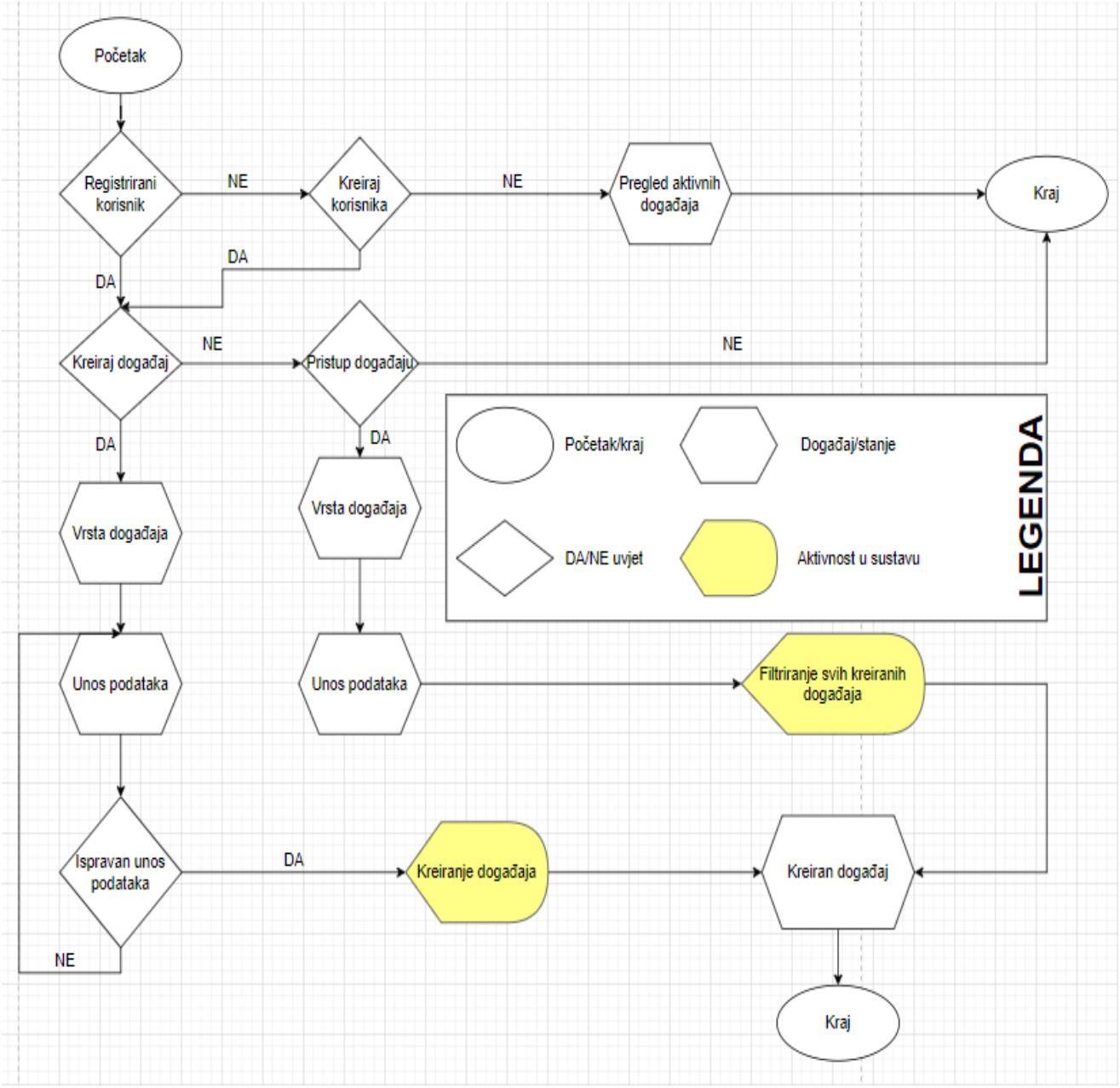
odabire željeni događaj te podnosi zahtjev za pristupanje. Ako je došlo do pogreške korisnik mora ponoviti unos.



Slika 9: Dijagram slijeda pristupanja događaju

5.2. EPC-dijagram

Ova vrsta dijagrama na jednostavan i znakovit način prati procese koji će se odvijati u aplikaciji. Prednost ove vrste dijagrama lagana je čitljivost i razumljivost te je izrazito korisna za kreiranje plana realizacije. Slika 10 prikazuje pojednostavljeni EPC dijagram koji prikazuje jednostavan i znakovit način na koji će sama aplikacija raditi.



Slika 10: EPC-dijagram

6. Model programskog inženjerstva i oblikovanje korisničkog sučelja

6.1. Agilni model

Osnovni cilj agilnog pristupa, tj. metodologije je da nakon svakog sprinta postoji potencijalno razvijen program koji može krenuti u izvedbu. Sprint predstavlja kratak period koji traje otprilike jedan tjedan pa sve do i nekoliko mjeseci te se može opisati kao mali projekt. Nakon što je sprint odrađen i testiran, klijenti analiziraju najnoviju iteraciju te nude prijedloge i zahtjeve za korekciju. Sukladno zahtjevima i prijedlozima, programeri klijenata rade prilagodbe u sljedećoj iteraciji.

U ovom završnom radu realizirat će se, oblikovanjem korisničkog sučelja, zahtjevi prve iteracije. Konkretno, na primjeru izrade aplikacije za pronalazak potencijalnih suigrača u sportu, definirani su zahtjevi u poglavlju četiri te poglavlje pet predstavlja dijagram procesa koji pobliže opisuje kako bi aplikacija trebala raditi.

Programeri, sukladno zahtjevima i dijagramima, pobliže shvaćaju što aplikacija treba raditi te samoorganizacijom odlučuju primarne i sekundarne zahtjeve te koliko će se vremena potrošiti na svaki. Korisnici bi nakon ove iteracije već mogli koristiti aplikaciju u početnoj verziji. Sukladno tome, aplikacija bi trebala imati zadovoljene sljedeće zahtjeve:

- povezivanje s bazom podataka u kojoj se spremaju podaci,
- dohvat podataka iz baze podataka,
- pronalazak događaja i potencijalnih suigrača,
- kreiranje događaja.

Nakon završetka prve iteracije gore navedeni zahtjevi uspješno su realizirani. Aplikacija odlazi klijentu na analizu te nude prijedloge i zahtjeve za korekciju. Korisnici aplikacije mogu koristiti aplikaciju u probnoj fazi. U drugoj iteraciji potrebno je poboljšati i nadograditi početnu kreiranu programsku podršku. Sljedeći zahtjevi trebaju biti završeni u drugoj iteraciji:

- kreiranje treninga,
- pronalazak trenera i organizacija,
- pristupanje razgovoru nakon uspješnog pristupanja događaju ili treningu,

- funkcionalnost ocjenjivanja,
- prosjek ocjena prikazan na profilu korisnika.

6.2. Oblikovanje korisničkog sučelja

Korisničko sučelje definira vezu između računala, tj. softvera i korisnika. Bitno je naglasiti da korisničko sučelje u pravilu sadrži dvije glavne komponente: ulaz i izlaz. Pod ulazne funkcije spadaju komponente kojima govorimo računalu što bi trebalo raditi, a najčešće se koriste: tipkovnica, miš, mikrofon. Izlaz predstavlja povratne informacije koje računalo daje korisniku te ovisi o ulaznim parametrima koje je korisnik unio.

U ovom odlomku rad će se fokusirati na izradu jednostavnog korisničkog sučelja, koji će biti dio java swing aplikacije za pronalazak potencijalnih suigrača u sportu. Izrada sučelja bit će podijeljena u tri komponente:

1. komponenta izrade sučelja za prijavu i registraciju korisnika,
2. komponenta izrade sučelja za kreiranje događaja,
3. komponenta izrade sučelja za pristupanje događaju.

Slijedom navedenog prikazat će se ključna rješenja korisničkih sučelja aplikacije za ostvarivanje i ispunjavanje najvažnijih korisničkih zahtjeva.

6.2.1. Baza podataka

Baza podataka pojavljuje se kao bitan dio neke pojedine aplikacije ili kao pojedini element koji pruža potporu raznim aplikacijama. U današnje vrijeme postale su nužne za organizaciju rada, njezinu analizu i manipulaciju. Razvojem tehnologije razvija se koncept stvaranja kolekcije podataka koje nisu više na datotekama na disku, nego su fizički odvojene na vanjskoj memoriji računala. Samim time, postignuta je učinkovitost, efikasnost i pouzdanost. Prema [14], baza podataka skup je međusobno povezanih podataka, pohranjenih u vanjskoj memoriji računala. Podaci su istovremeno dostupni raznim korisnicima i aplikacijskim programima.

U ovom radu kreirat će se dvije osnovne baze podataka koje će pružati temelj kreiranja i rada korisničkih sučelja. Prva baza podataka bit će vezana uz registraciju i prijavu korisnika, odnosno služiti će za pohranu osnovnih osobnih podataka.

Kada korisnik prvi puta izvrši registraciju i upiše sve potrebne podatke, oni će se automatski kopirati i zapisati u bazu podataka u kojoj će biti pohranjeni. Tijekom sljedećeg dolaska, tj. ulaska u sustav, korisnik popunjava polja za prijavu. Upisani podatci automatski se uspoređuju s podatcima već pohranjenim u bazi podataka. Ako su podatci istiniti i nalaze se u bazi podataka, korisniku će biti odobren pristup i rad s funkcijama aplikacije. Međutim, ako je došlo do krivog unosa i nije došlo do podudaranja unesenih podataka s baznim, aplikacija javlja pogrešku te je potrebno ponoviti upis.

Slika 11. prikazuje četiri podatkovna tipa koje čine, uoči registracije i prijave, glavnu strukturu baze podataka.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	Ime	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
☐	2	Prezime	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
☐	3	Lozinka	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
☐	4	Mail	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More

Slika 11: Struktura glavne baze podataka

Druga baza podataka vezana je za pristupanje i kreiranje događaja. Kada korisnik kreira događaj, popunjava obavezna tekstualna polja. Nakon što je upisao potrebne podatke, isti se kopiraju i zapisuju u bazu podataka. Kada neki drugi korisnik želi upotrijebiti funkciju aplikacije pristupa događaju, povlače se iz baze podataka svi događaji te su mu na raspolaganju. Slika 12 označava pet podatkovnih tipova vezanih za ključnu strukturu baze podataka, tijekom kreiranja i pristupanja slučaju.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	NazivSporta	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
☐	2	Mjesto	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
☐	3	BrojOsoba	int(20)			No	None			Change Drop More
☐	4	VrijemeOdrzavanja	time			No	None			Change Drop More
☐	5	Datum	date			No	None			Change Drop More

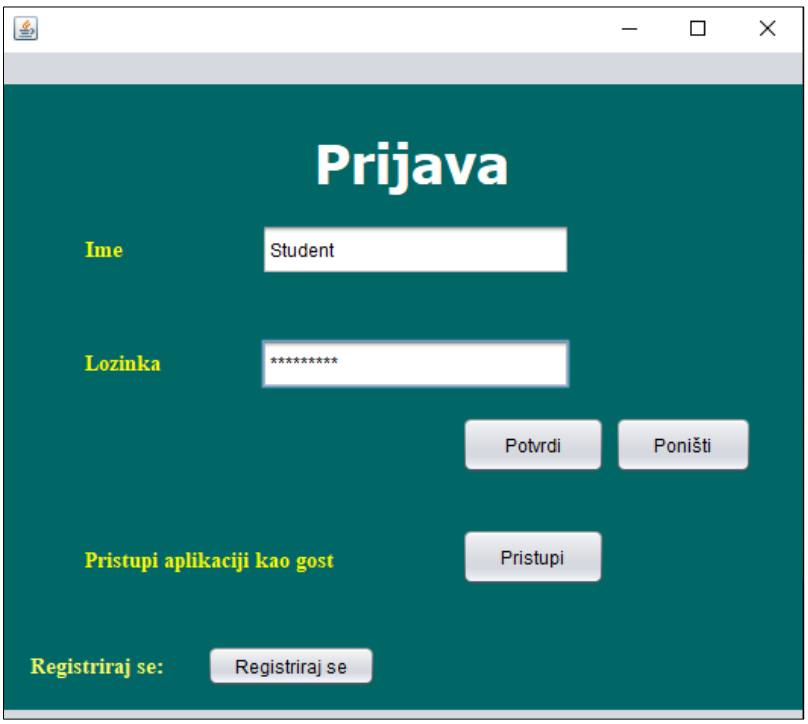
Slika 12: Struktura druge baze podataka

6.2.2. Sučelje za registraciju i prijavu korisnika u aplikaciju

Forma za prijavu u aplikaciju napravljena je na vrlo jednostavan način te sadrži osnovne elemente koji su potrebni za rad aplikacije.

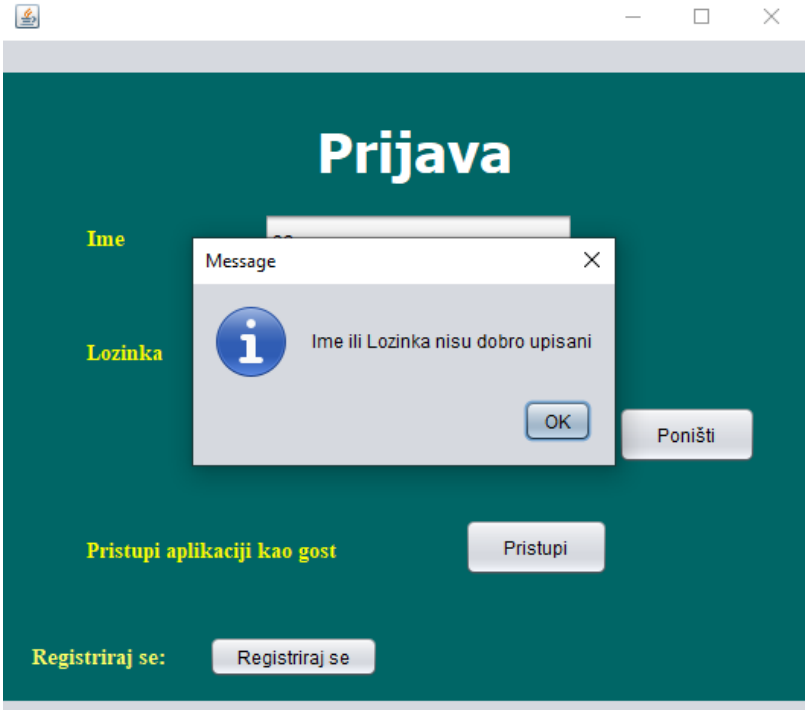
Login sučelje sadrži dva polja od kojih je jedno ime, a drugo lozinka. Zaporka treba poštivati osnovno pravilo sigurnosti, odnosno prekrivanje unesenog podatka prikazuje se zvjezdicama (*). Nakon što korisnik upiše korisničko ime i lozinku, mora potvrditi svoj unos te aplikacija automatski provjerava postoji li upisano ime u bazi podataka. Korisnik ima i treću mogućnost - pristup aplikaciji kao gost, koji nema sposobnost kreiranja i pristupanja događaju, već samo pregled mogućih funkcionalnosti aplikacije.

Slika 13. prikazuje formu za prijavu u aplikaciju.



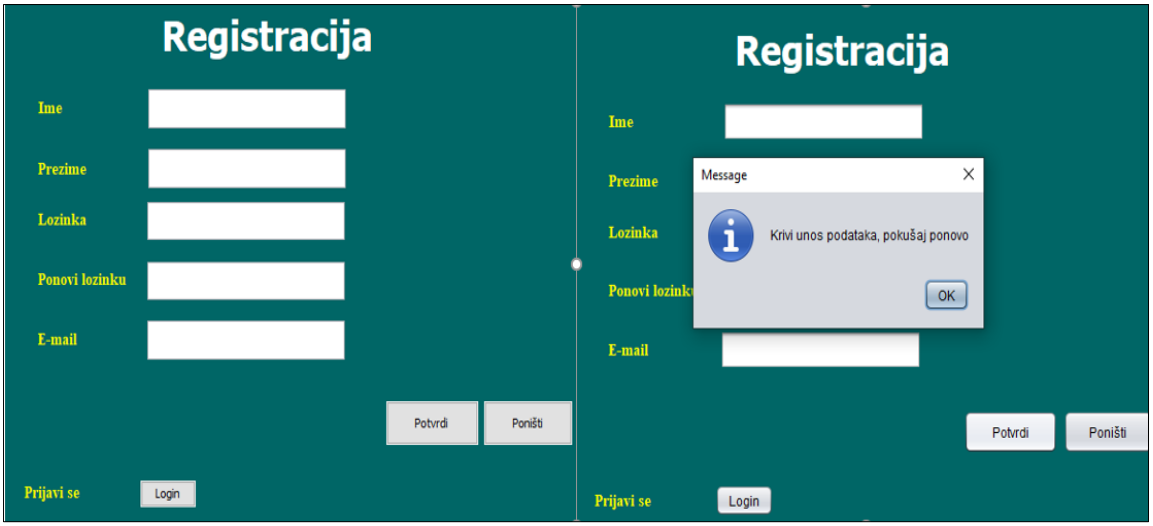
Slika 13: Primjer forme za ulazak u aplikaciju

U slučaju krivog unosa, korisnik nije u mogućnosti pristupiti aplikaciji te je potrebno, kako bi pristupio funkcionalnostima aplikacije, ponovno upisati ime i lozinku. Samim time postiže se viši nivo sigurnosti. Slika 14 prikazuje moguće pogreške tijekom prijave u aplikaciju.



Slika 14: Primjer moguće pogreške

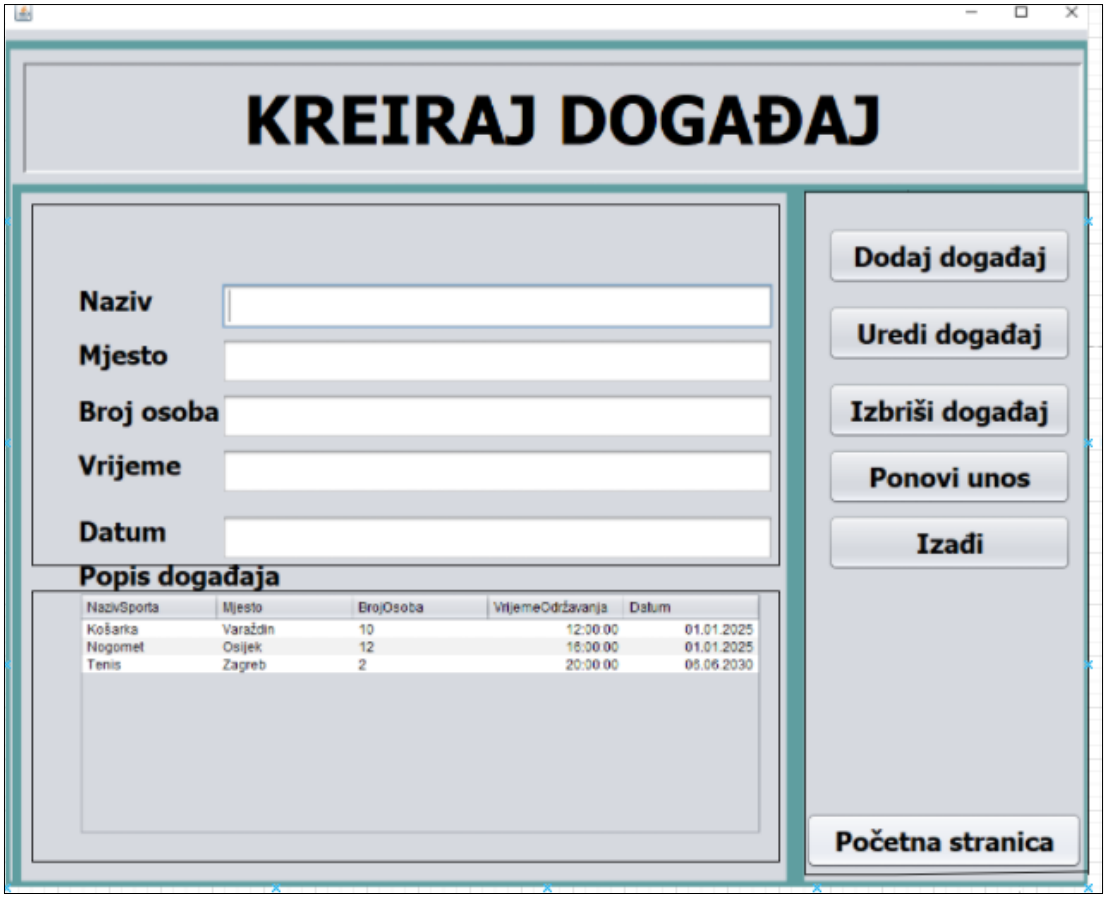
Ako korisnik prvi put koristi aplikaciju, od njega se zahtijeva da prođe kroz jednostavnu registraciju u kojoj mora popuniti pet ključnih polja: ime, prezime, lozinku, ponovni unos lozinke te e-mail. Kada su sva polja popunjena, korisnik pritiskom na gumb *potvrdi* završava svoju registraciju te je automatski upisan u bazu podataka i može aktivno sudjelovati u funkcijama koje pruža aplikacija. Ako korisnik nije popunio sva polja ili je došlo do pogreške, aplikacija obavještava korisnika kako su podaci krivo uneseni te da je potrebno ponoviti unos, što je vidljivo na slici 15.



Slika 15: Primjer sučelja za registraciju

6.2.3. Sučelje kreiranja događaja

Slika 16 prikazuje sučelje kreiranja događaja. Nakon što se korisnik uspješno registrirao i prijavio u aplikaciju, ponuđene su mu dvije moguće opcije aplikacije. Prva opcija je kreiranje događaja, a druga je pristupanje događaju. Slika 16 prikazuje opciju jedan. Korisniku se otvara sučelje s pet polja koje mora popuniti kako bi uspješno kreirao događaj. U slučaju da korisnik nije popunio sva navedena polja, nije u mogućnosti kreirati isti. Aplikacija korisniku također pruža jednostavne funkcionalnosti: *uredi događaj*, *izbriši događaj*, *ponovni unos*. Kada je korisnik, klikom na gumb *dodaj događaj*, popunio sva navedena polja, podaci su automatski spremljeni u bazu podataka te su sada vidljivi svima koji koriste aplikaciju (slika 17).



Slika 16: Sučelje kreiranja događaja

+ Options

NazivSporta	Mjesto	BrojOsoba	VrijemeOdrzavanja	Datum
Košarka	Varaždin	10	12:00:00	01.01.2025
Nogomet	Osijek	12	16:00:00	01.01.2025
Tenis	Zagreb	2	20:00:00	06.06.2030

Slika 17: Popis događaja u bazi podataka

6.2.4. Sučelje pristupanja događaju

Nakon što se korisnik uspješno registrirao i prijavio u aplikaciju, ponuđene su mu dvije moguće opcije aplikacije. Prva opcija je kreiranje događaja, a druga je pristupanje događaju. Slika 18 prikazuje opciju dva. Korisniku se otvara sučelje s pet polja i tablica svih kreiranih događaja. Korisnik ga klikom na događaj u tablicu označuje te pritiskom na tipku *pristupi* pridružuje se istom.

Ako je u tablici previše podataka, aplikacija ima ugrađeni mehanizam filtriranja. Filtriranje radi po principu unosa početnih slova u bilo koji od pet tekstualnih polja. Vidljivo je na slici 18 da je korisnik u mogućnosti filtrirati podatke u sljedećim kategorijama: naziv sporta, mjesto održavanja, broj osoba, vrijeme održavanja i datum održavanja.

Slika 18: Filtracija tablice sa svim događajima

7. Testiranje

Testiranje je nužno i ne smije biti odbačeno iz razvoja programskog modela. U većini industrija i disciplina, koje pružaju proizvod ili uslugu, treba postojati proces koji testira performanse, koliko je precizan, koje su mu granice, koje radnje dovode do mijenjanja performansi i slično.

Kada se govori o testiranju u svijetu programiranja, u većini slučajeva kod će sadržavati pogrešku, a krajnji softver neće odgovarati 100 % korisnikovim zahtjevima. Konstatiramo da uz kod i program, koji mora raditi savršeno, testiranjem olakšavamo održavanje samog softvera. Znatan broj proizvoda nije napravljen za kratkotrajnu upotrebu, već je namijenjen za dugotrajnu. U tom se ciklusu na projektu može promijeniti i do nekoliko različitih ljudi. Kako bi se novodolazeći programeri upoznali s kodom i njegovim funkcionalnostima, koje proizvod pruža na brz i efikasan način, koristi se testovi. Testiranje odlično funkcionira za održavanje softvera. Bitnu ulogu u održavanju koda predstavlja dokumentiranje testova te se tako postiže dugotrajna korist za programski proizvod: spremanjem i čuvanjem za buduće nadogradnje i izmjene. S obzirom na to da se tehnologija brzo razvija, promjene su neophodne. U tom slučaju programeri koriste testove kako bi im mijenjanje koda bilo olakšano [15].

7.1. Plan testiranja

Plan testiranja provodit će se u tri faze:

- jedinično testiranje,
- integracijsko testiranje,
- korisničko testiranje.

7.1.1. Jedinično testiranje

Prva faza testiranja odnosit će se na jedinično testiranje u kojem se provjerava ispravnost samostalnih jedinica sustava. Tako provjeravamo prisutnost grešaka u pojedinim dijelovima koda i na temelju toga može se brže i efikasnije ispraviti pronađena pogreška.

7.1.2. Integracijsko testiranje

U prvoj smo fazi testirali svaku jedinicu zasebno, a sada trebamo provjeriti funkcionalnost cjelokupnog sustava. Potrebno je ispitati je li je sustav prikladan za predviđenu svrhu. Moramo provjeriti kako među pojedinom funkcionalnosti sustava ne postoje neočekivane interakcije

Plan testiranja:

1. Prijava i registracija

Provjeravamo ispravnost unesenih podataka. Također trebamo provjeriti spremaju li se podaci registriranih korisnika u bazu podataka, a kod prijavljenih moramo osigurati pravilno dohvaćanje istih.

2. Koristi li korisnik aplikaciju kao registrirani ili neregistrirani korisnik

Odluči li korisnik ostati anoniman, nije u mogućnosti koristiti potpune funkcionalnosti aplikacije, nego ih može samo pregledavati. Kako bi imao mogućnost potpunog korištenja, potrebno je izvršiti registraciju, a svaki sljedeći puta prijavu u sustav.

3. Pohrana i dohvat podataka iz baze podataka

Provjerava se točnost dohvata i upisa podataka u bazu podataka.

4. Kreiranje događaja

Provjera ispravnosti funkcionalnosti koje su potrebne za kreiranje događaja. Dođe li do pogreške, aplikacija od korisnika zahtjeva ponovni unos.

5. Filtriranje i pristupanje događaju

Radi lakšeg pronalaska i organizacije podataka u tablicu svih događaja testiramo pristupanje događaju, kao i funkciju filtriranja.

7.1.3. Korisničko testiranje

Posljednja faza odnosit će se na ponašanje sustava u odnosu na zahtjeve korisnika. Aplikacija mora biti svakom korisniku pregledna i jednostavna za korištenje. Testiranje će se provodit kroz početničku beta verziju aplikacije. Korisnici šalju svoje mišljenje, reakcije i probleme programerima koji zatim ispravljaju uočene pogreške.

8. Rasprava

Poglavlje *Rasprava* obuhvatit će sveukupne dobivene rezultate te način na koji ih je potrebno interpretirati. U usporedbi s drugim znanstvenim profesijama, znanost programsko inženjerstvo novija je disciplina, ali to ne znači da je manje bitna. Budući da su u današnje vrijeme projekti sve veći i kompliciraniji, primjenjuju se ove bitne znanosti. Kako bi projekti bili konkurentni na tržištu, potrebno je detaljno planiranje i razrađivanje svakog detalja, bilo to s financijskog, vremenskog ili kvalitetnog gledišta.

Završni rad nakon poglavlja *Programsko inženjerstvo* prelazi na praktičan dio rada. Praktičan dio može se podijeliti na dva glavna dijela. Prvi dio *Osmišljavanje aplikacije* i *Prikupljanje korisničkih zahtjeva* te drugi dio u kojem se ti isti korisnički zahtjevi realiziraju jednostavnom aplikacijom. Kako bi aplikacija sadržavala sve željene funkcionalnosti i elemente, korisnički zahtjevi trebaju biti detaljno opisani i razrađeni do najmanjeg detalja, te se tek onda prelazi na drugi dio – realizaciju i programiranje.

Ideja ovog završnog rada je jednostavnom aplikacijom približiti korisnicima sport i rekreaciju kako bi se pronašli potencijalni suigrači u sportu. Korisnici aplikacije na vrlo jednostavan način kreiraju događaje na koje se drugi korisnici u samo par klikova mogu prijaviti i sudjelovati.

Kao što je navedeno u poglavlju *Model programskog inženjerstva* i *oblikovanje sučelja* koristi se agilni pristup te je u ovom završnom radu razrađena prva iteracija, tj. sprint. Kako bi aplikacija bila u potpunosti spremna za korištenje, trebat će prijeći barem još jednu, a moguće i više iteracija. Mogući pravci razvoja aplikacije:

- kreiranje treninga,
- pronalazak trenera i organizacija,
- pristupanje razgovoru nakon uspješnog pristupanja događaju ili treningu,
- funkcionalnost ocjenjivanja,
- prosjek ocjena prikazan na profilu korisnika,
- baza podataka pohranjena na internetu,
- rad aplikacije na mobitelu,
- grafičko oblikovanje i uređivanje aplikacije.

9. Zaključak

Uvjet za postizanje i razvoj čak i najmanjeg softvera, koji je kvalitetan i učinkovit, adekvatno je vođenje. Kako bi proizvod bio prihvatljiv na konkurentnom tržištu, za adekvatno vođenje potrebno je poznavati različite metode i alate programskog inženjerstva.

Samoj konstrukciji najčešće se pristupa s tehničkog aspekta te se naglašava planiranje i upravljanje. U ovom radu jasno se otkriva kroz koje dijelove prolazi softver te su u svakoj fazi opisani ključni pojmovi kako bi se ta faza završila i započela druga. Samim radom može se zaključiti kako ne postoje najbitnije faze, te se samo dobrom organizacijom i kohezijom svih elemenata postiže korelacija između svih faza.

Rad do poglavlja *Strukturiranje programske podrške* obrazlaže specifične metodologije i principe programskog inženjerstva, korelaciju terminologije, kratak povijesni prikaz i porijeklo programskog inženjerstva. Rad objašnjava metode, alate i osnovne procese programskog inženjerstva i manipuliranje procesima, te je u njemu naglasak na poznavanju i shvaćanju načina kojim se poslovne regulative iz svijeta poslovanja dodiruju s procedurama programskog inženjerstva.

Od poglavlja *Strukturiranje programske podrške,* rad nastavlja detaljnije objašnjavati osnovne principe razvoja programske podrške, i to na primjeru izrade aplikacije za pronalazak potencijalnih suigrača u sportu. Rad se do poglavlja *Model programskog inženjerstva i oblikovanje korisničkog sustava,* detaljnim opisivanjem i strukturiranjem programske podrške, fokusira na teorijsku realizaciju i planiranje programske podrške. Sukladno tome, u radu se definiraju korisnički zahtjevi opisani dijagramima procesa, koji će biti temelj pri izradi korisničkog sučelja, za pronalazak potencijalnih suigrača u sportu.

Rad završava dvama važnim poglavljima programskog inženjerstva: *Model programskog inženjerstva i oblikovanje korisničkog sustava* i *Testiranje*. U ovom dijelu, izradom jednostavne aplikacije, realiziraju se prije nastali i definirani korisnički zahtjevi, za pronalazak potencijalnih suigrača u sportu.

10. Literatura

1. Sommerville I.: Software Engineering, 9th ed., Addison-Wesley, Harlow, England, 2011.
2. Nađ J.: Programsko inženjerstvo i informacijski sustavi, Međimursko veleučilište u Čakovcu, 2020.
3. Frančić M., Pogarčić I. (2010): Kvaliteta modela poslovanja razvijenog UML-om, https://www.semanticscholar.org/paper/A-Brief-Essay-on-Software-Testing-Bertolino-Marchetti/b7e3cf5b5bae2349eb0bf8284db6bb881de524ea 27.08.2021.
4. ConceptDraw: Dostupno na: https://www.conceptdraw.com/How-To-Guide/epc-for-configuring-an-enterprise-resource-planning, 27.08.2021.
5. Jović A., Frid N., Ivošević D.: Procesi programskog inženjerstva, Sveučilište u Zagrebu, skripta, 2019.
6. Kovačević Ž., Krusha E.: Programsko inženjerstvo i informacijski sustavi, Tehničko veleučilište u Zagrebu, priručnik, 2020.
7. Škola Koda: Model vodopada: Dostupno na: https://skolakoda.github.io/model-vodopada, 27.08.2021.
8. Strephonsays: Razlika između vodopadnog i spiralnog modela: Dostupno na : https://hr.strephonsays.com/waterfall-and-vs-spiral-model-4079 , 27.08.2021.
9. Drašković N.: Informacijski sustav za evidenciju komunikacijskih uređaja na brodovima, Spiralni model razvoja, Sveučilište u Dubrovniku, 2012. str.7. Dostupno na: https://www.bib.irb.hr/593801, 27.08.2021.
10. hr.differencevs: Razlika između agilne i tradicionalne metodologije razvoja. Dostupno na https://hr.differencevs.com/6861206-difference-between-agile-and-traditional-software-development-methodology, 27.08.2021.
11. Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. University of Western Australia, 35.
12. Bertolino A., Marchetti E.: A Brief Essay on Software Testing, str. 1-14
13. ISO/IEC, International Standard ISO/IEC 9126, ISO/IEC, 2019.
14. Manger R., Osnove projektiranja baze podataka, Sveučilišni računski centar u Zagrebu, priručnik, 2010.
15. Myers G. J., Badgett T. I Sandler C.: The art of software testing – Third edition, New Jersey: Hoboken, N.J., USA : John Wiley & Sons, 2012.

Popis slika