SERVIS ZA AUTOMATIZACIJU OBRADE PODATAKA POMOĆU VIZUALNOGA PROGRAMSKOG JEZIKA

Jug, Darian

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

SERVIS ZA AUTOMATIZACIJU OBRADE PODATAKA POMOĆU VIZUALNOGA PROGRAMSKOG JEZIKA

Darian Jug

Zagreb, veljača 2019.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, 26. veljače 2019..

Predgovor

Zahvaljujem svom mentoru, dipl. ing. Aleksandru Radovanu, na iskazanom povjerenju i svesrdnoj pomoći pri izradi ovog završnog rada.

Zahvaljujem i Karlu Crnekoviću na pomoći i savjetima tijekom izrade ovog završnog rada te Igoru Mandiću i Silviji Radelić na pomoći pri testiranju sustava za ovaj završni rad.

Od srca zahvaljujem svojoj obitelji na pruženoj potpori i motivaciji tijekom studiranja.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Aplikacija *bitsmth* je mobilna aplikacija za vizualno programiranje. Ona služi kao potpora servisu za automatizaciju obrade podataka. Cilj *bitsmtha* je da korisniku omogući izradu kompleksnih tijekova rada *(engl. workflows)* koji u više koraka i iz više izvora procesiraju podatke, a bez potrebe za eksplicitnim naredbama za izvršavanje.

Ključne riječi: Mobilna aplikacija, vizualno programiranje, tijek rada, obrada podataka, programski jezik

Sadržaj

1. Uvod

Ovaj završni rad bavi se temom pronalaska najefikasnijeg rješenja automatizacije obrade podataka pomoću vizualnog programiranja i vizualnih programskih jezika. Vizualni programski jezici su jezici IV. generacije koji za programiranje koriste vizualne elemente poput blokova i linija, za razliku od programskih jezika III. generacije koji koriste sintaksu ključnih riječi.

U drugom poglavlju obrađuju se problemi s postojećim rješenjima u tom području. U trećem poglavlju obrađuje se princip implementacije platforme, mobilne aplikacije za *iOS* i dizajn korisničkog iskustva i korisničkog sučelja. Četvrto poglavlje bavi se implementacijom potporne serverske aplikacije i arhitekturama korištenim u tu svrhu. Peto poglavlje opisuje rezultate testiranja *iOS* prototipa te usporedbu s postojećim rješenjima.

Svrha *bitsmth* aplikacije je istražiti najefikasnije implementacije povezivanja više IT sustava, automatizirati ponavljajuće procese i optimizirati svakodnevno korištenje.

2. Problem i cilj

2.1. Izbor problema za završni rad

Ideja za razvojem vizualnoga programskog jezika i mobilne aplikacije za automatizaciju obrade podataka proizlazi iz potrebe za sučeljem koje jednostavnije i intuitivnije omogućava svakodnevnu interakciju s IT sustavima. Tu interakciju je u današnje vrijeme gotovo nemoguće izbjeći - internetsko i mobilno bankarstvo postali su najjednostavniji i najbrži način obavljanja transakcijskih usluga; društvene mreže postale su dominantan način dolaska do informacija o tome gdje izaći, koji film pogledati ili što čestitati prijateljima ili članovima obitelji; aplikacije za navigaciju postale su gotovo jedini siguran način za pronalazak rute do odredišta zbog njihove točnosti, dostupnosti i jednostavnosti korištenja. Takvi sustavi, mobilne aplikacije i servisi razvijeni su kako bi se svakodnevni zadaci i potrebe ubrzale i pojednostavile, a sve s ciljem što veće automatizacije i smanjenja većine dnevnih zadataka modernog čovjeka. No, ako su ti sustavi međusobno nepovezani, stvara se nova grupa zadataka koja potencijalno neutralizira pozitivne efekte modernizacije i internalizacije društva.

Često se na pametnim telefonima nalazi mnoštvo aplikacija koje gotovo izvršavaju isti zadatak. Korisnik koji ima više aplikacija iz kategorije financija, npr. aplikaciju za praćenje potrošnje i aplikaciju za izvršavanje transakcija, često jednu te istu stvar unosi u obje aplikacije, kako bi na kraju autorizirao transakciju aplikacijom za autorizaciju internet bankarstva. Pristup informacijama smatra se srži interneta, a privatne informacije lagano se maskiraju nestandardiziranim formatom podataka i aplikacijama zatvorenog koda.

2.2. Cilj i zadaci završnog rada

Cilj ovog rada je analizirati postojeća rješenja i istražiti kako na efikasniji i intuitivniji način poboljšati načine povezivanja različitih IT sustava, mobilnih i web aplikacija te optimizirati njihovo svakodnevno korištenje. Na tržištu trenutno postoje dva rješenja koja djelomično rješavaju problematiku i opseg problema razmatranog u ovom radu: *IFTTT* i *Apple Shortcuts*.

2.2.1. *IFTTT*

Trenutno najpoznatije rješenje koje zadovoljava potrebe za izradom tijekova rada povezivanja više sustava je *IFTTT (skr. engl. If This Then That)* [17]. *IFTTT* omogućava korisnicima izradu jednostavnih tijekova rada prema načelu ”ako se dogodi X, napravi Y”. ”Ako se dogodi X” dio naziva se okidač, a ”napravi Y” definira radnju (akciju) y koja će se dogoditi ako se ostvari uvjet x definiran u okidaču [16]. *IFTTT* svoju primjenu nalazi u rješavanju radnih tijekova u automatizaciji doma, na društvenim mrežama i servisima poput *Dropboxa, Google Drivea* i *GMaila* [16]. Monetizacija platforme razrađena je tako da kreatori okidača i radnji plaćaju implementaciju tijekova rada u platformu. *IFTTT* nudi i besplatnu inačicu u kojoj hobistima omogućavaju izradu tijekova rada, koji se ne smiju komercijalno ili javno publicirati [16].

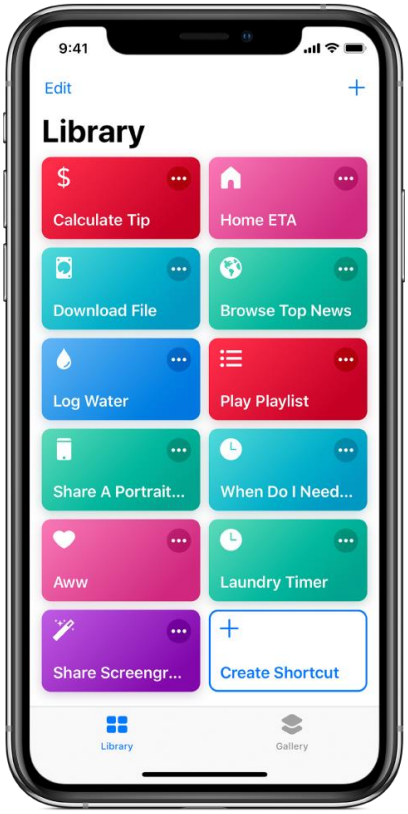
Korisnik *IFTTT* limitiran je na proces okidač-akcija. Ne postoji mogućnost izrade višestrukih okidača i/ili akcija za pojedini radni tijek te je korisničko sučele limitirano na odabir okidača i radnji koje taj okidač potiče. Također nije moguće izraditi višeslojne obrade podataka ni skupljati podatke iz više izvora odjednom. Izvedba pojedinog radnog tijeka ograničena je na vremenski prozor od 15 minuta, što znači da platforma svakih 15 minuta provjerava da li je okidač ispunjen da bi izvršila akciju [26]. Tijekovi rada se izvršavaju u pozadini na serverima *IFTTT-a* te ne postoji upit u korisnikovu interakciju s radnim tijekom.

2.2.2. *Apple Shortcuts*

Apple Shortcuts (skr. Shortcuts) predstavljen je u svibnju 2018. godine na *Apple World Wide Developer Conferenceu* kao dodatak novoj verziji mobilnog operacijskog sustava *iOS* 12 u obliku zasebne aplikacije dostupne besplatno na *Apple App Storeu* koja je aplikacija za preuzimanje aplikacija na operacijskom sustavu [7]. *Shortcuts* omogućava korisnicima izradu prečaca za izvedbu određenih radnji koristeći vizualnog urednika. Korisnik može izraditi jednostavne radne tijekove koji se integriraju u pojedine aspekte *iOS* operacijskog sustava, npr. pozivanje, slanje tekstualnih i *iMessage* poruka ili izvedbu radnji za koje korisnik obično koristi *Appleovu* virtualnu pomoćnicu *Siri* [7]. Radni tijekovi u aplikaciji *Shortcuts* predstavljeni su radnjama koje su prikazane u vertikalnom nizu. Prva izvedena radnja svoj *output* šalje onoj ispod nje i tako do kraja radnog tijeka. Radnja ne mora izdati *output* i ne mora prihvatiti *input* akcije iznad nje. Korisnik može, na primjer, izraditi tijek rada koji započinje odbrojavanje/štopericu i radni zapis te podsjetnik da zaključi radni zapis

kada napusti lokaciju. Drugi primjer radnog tijeka je podsjetnik za čaj koji nasumično bira vrstu čaja koju korisnik može popiti. Korisnik može preuzeti radne tijekove s Interneta ili integrirane galerije i modificirati ih prema svojim potrebama. *Shortcuts* nudi brojne radnje koje se integriraju s radnjama i funkcionalnostima sustava *iOS*.

Jedan od problema *Shortcuts* aplikacije je upravo vertikalni način izrade pri definiranju kompleksnijih tijekova rada. Tijek rada postane veoma nepregledan već nakon dvije ili tri radnje i često je komplicirano i zbunjujuće prelaziti i pregledavati kompletni tijek rada i pojedine radnje koje su međusobno povezane. Nadalje, svi tijekovi rada izvršavaju se isključivo na korisnikovom uređaju, što znači da uređaj mora biti konstantno uključen i pri ruci. Također, izvršavanje svakog tijeka rada zahtijeva korisničku interakciju i ne postoji mogućnost automatskog pokretanja tijeka rada. Ako radnja zahtijeva modifikacije sistemskih aplikacija, korisnik mora dati dodatnu privolu za izvršavanje svake radnje kako bi se tijek rada nastavio i izvršio do kraja, što može postati zamorno ako korisnik ima više takvih radnji.



Slika 1. *Shortcuts* aplikacija. Izvor: Apple Shortcuts App Guide: https://support.apple.com/guide/shortcuts/welcome/ios, siječanj 2019.

Na tržištu trenutno ne postoje aplikacije za *iOS* mobilni operacijski sustav koje bi korisnicima omogućile izradu kompleksnih tijekova rada koji bi mogli procesirati podatke na udaljenim serverima u više koraka i iz više izvora kada je neki okidač zadovoljen.

Od prethodno opisanih primjera, jedino je *Shortcuts* primjer vizualnog programiranja uz pomoć grafičkih elemenata umjesto tekstualnih ključnih riječi i simbola. Potrebno je osmisliti i izvesti rješenje koje će korisniku omogućiti izradu kompleksnih i višeslojnih tijekova, a kroz koje se istovremeno može jednostavno i pregledno upravljati.

Također, ograničenja aplikacije *Shortcuts* leže u tome što izrada kompliciranih radnih tijekova nije jednostavno izvediva jer svi elementi leže na istoj osi. Korisnici *iOS* sustava upoznati su i navikli na geste poput štipanja za uvećavanje ili povlačenja po ekranu za pomicanje sadržaja. Te geste mogu se iskoristiti za lakše upravljanje kroz radni prostor programskog jezika, što bi na kraju omogućilo korisnicima jednostavno pronalaženje željenog elementa na radnom prostoru.

3. Implementacija rješenja

Rješenje je mobilna aplikacija koji omogućuje stvaranje kompliciranih tijekova rada i izvršavanje istih putem servisa. Takav programski jezik je takozvani jezik IV. generacije [23], jezik koji se temelji na III. generaciji, ali je više orijentiran na lakoću pisanja i razumijevanje od strane programera. Primjer takvog jezika je SQL, koji je više orijentiran za razumijevanje nego efikasnost pisanja koda. Temeljni element vizualnog programskog jezika je element u obliku pravokutnika. On predstavlja metodu i najmanja je jedinica u izvršavanju tijeka rada. Taj element sadrži kružiće koji označavaju ulaze i izlaze za podatke. Kružići koji se nalaze na lijevoj strani su ulazni, a oni koji se nalaze na desnoj su izlazni. Boja kružića označava tip podataka koji se prenosi.

Slijed izvršavanja radnji se određuje povezivanjem kružića elementa koji označavaju metode. Sličan tip programiranja je definiran pod standardom IEC 61499 [21]. Prema kojem su definirani elementi koji predstavljaju funkcije koje se sastoje od ulaza i izlaza. Ulazi i izlazi se dijele na podatkovne i na one koji označavaju završetak nekog određenog događaja. Standard se koristi za kontrolu i programiranje industrijskih strojeva.

3.1. Opis implementacije platforme

3.1.1. Analiza mogućih implementacija i dostupnih arhitektura

3.1.1.1. Mobilna aplikacija

Za optimalno izvršenje navedenih ciljeva najveći fokus treba staviti na korisničko iskustvo programiranja i manipulacije elementima vizualnog programskog jezika. Kao što je spomenuto u prethodnim poglavljima, za programiranje kompleksnih tijekova rada *(engl. workflows)* korisnik mora imati mogućnost brzog upravljanja i navigacije kroz elemente vizualnoga programskog jezika. Iz tog razloga implementacija mobilne aplikacije mora biti brza i imati bogato korisničko iskustvo. Mobilne aplikacije aplikativno se s toga mogu podijeliti u tri glavne skupine: *native* mobilne aplikacije, web mobilne aplikacije i hibridne mobilne aplikacije.

Native mobilne aplikacije

Kod *native* mobilnih aplikacija većina ili sav kod koji pokreće aplikaciju zadan je u obliku instrukcijskog seta procesora, gdje se i izvršava. U *native* mobilnim aplikacijama direktno su korištene sistemske knjižnice i metode operacijskog sustava.

Web mobilne aplikacije

Druga mogućnost da je aplikacija web aplikacija, to bi značilo da je sav programski kod pisan u *JavaScript* programskom jeziku, te da sav sadržaj je prikazan koristeći HTML i CSS tehnologije.

Za razliku od *native* mobilnih aplikacija koje se pokreću i izvršavaju u procesoru uređaja, web aplikacije su web stranice prilagođene za prikazivanje na mobilnim uređajima. Sav programski kod je pisan u *JavaScript* programskom jeziku, a sav sadržaj prikazuje se koristeći HTML i CSS tehnologije.

Hibridne mobilne aplikacije

Hibridne mobilne aplikacije su web aplikacije pisane HTML i CSS tehnologijama i *JavaScript* programskom kodu, a taj kod se sastavlja *(engl. compile)* u aplikaciju koja sadržaj izvršava i prikazuje u `WebView` komponenti.

Kod *iOS* mobilnog operacijskog sustava, *Apple* ograničava korištenje samo web tehnologija zasnovanih isključivo na *WebKit rendering engineu.* Koristeći *WebKit* umjesto *native* implementacija aplikacije gube se mnoge optimizacije i mnoge značajke sistemskih knjižnica koje omogućuju brzo i ugodno korisničko iskustvo. Jednostavne interakcije poput štipanja za uvećavanje ili povlačenja po ekranu za pomicanje sadržaja su vidljivo različite i iskustveno gore od korisničkog iskustva onih koji su pružene u *native* aplikacijama. Također nije moguće prilagođavati niti mijenjati implementaciju *WebKita* na *iOS* sustavu. Mnoge metode i atributi pronađeni u izvornom kodu *WebKit projekta* nam nisu dostupni na *iOS* platformi. Implementacija se može razlikovati od verzije do verzije mobilnog operacijskog sustava. Gledajući da nije moguće svojevoljno instalirati određene verzije mobilnog operacijskog sustava na *iOS* uređaje, testiranje web i hibridnih aplikacija je ograničeno na simulatore *iOS* uređaja. Takvi simulatori se jedino mogu pokretati na *Mac* računalima i performanse simulatora su ograničene usporedno s pravim uređajima kojega simuliraju. Razlog toga je da simulator koristi instrukcijski set *i368* ili *x86*, ovisno o arhitekturi čipa računala na kojem se pokreće simulator, a ne instrukcijski set uređaja koji je u pravilu *arm64*.

Aplikacija ne zahtijeva eksplicitne interakcije da bi se pokrenula u pozadini na serverima, međutim u određenim slučajevima poželjno je obavijestiti korisnika o završetku tijekova rada. Iz tog razloga aplikacija mora podržavati mogućnost *push* obavijesti. Web aplikacije same po sebi na *iOS* sustavu ne nude mogućnosti *push* obavijesti, dok kod *native* i hibridnih mobilnih aplikacija postoje knjižnice i alati za omogućavanje slanja *push* obavijesti.

Projekti poput *Facebook React Native* omogućuju korištenje *JavaScript* programskog jezika i *native* korisničkih elemenata. Koristeći taj projekt je moguće napisati jedan *codebase* i koristiti ga na *iOS* i *Android* sustavu. Time se ne umanjuje korisničko iskustvo jer se koriste *native* elementi korisničkog sučelja. Ova mobilna aplikacija prikazuje veliki broj elemenata odjednom na zaslonu i omoguće kompliciranе interakcije i geste između njih koje *native* korisnički elementi ne pružaju. Ukoliko bi se koristio projekt poput *Facebook React Nativea* za implementaciju ove aplikacije, bilo bi potrebno implementirati *native* implementacija novih korisničkih elemenata i za *iOS* i *Android* sustav. Takvi projekti također stvaraju dodatni sloj kompleksnosti *codebasea*, smanjuju performanse aplikacije te stvaraju ovisnost o održavateljima tog projekta. S obzirom na to da je potrebno pronaći ili educirati programera koji zna programirati za taj operacijski sustav i koristiti takav projekt uz sve to navedeno imaju i ekonomske nedostatak u vidu da je to dodatni trošak, a i teže je pronaći uže specijalizirane programere.

3.1.1.2. Potporna serverska aplikacija

Potporna serverska aplikacija treba moći izvršiti veliki broj radnji koje korisnici pokreću i omogućiti mijenjanje tijekova rada bez zaustavljanja istih. Potporna serverska aplikacija je podijeljena u dva dijela: dio koji pokreće tokove rada i obrađuje radnje i dio kojemu pristupa mobilna aplikacija. Taj drugi dio omogućuje modificiranje radnih tijekova u obliku pristupne točke.

Za pristupnu točku je odabran programski jezik *Python*. *Python* je dinamičan objektno orijentiran programski jezik što je prednost za pristupnu točku u vidu dinamičnost i fleksibilnost uspoređujući sa statičnim programskim jezicima. Prema *TIOBE* Indexu, siječanj 2019., [25] *Python* programski jezik je treći najzastupljeniji programski jezik. S takvom popularnost lakše je pronaći programere koji bi mogli surađivati na ili održavati *codebase* tog projekta. Dinamički programski jezici omogućuju jednostavnije i brže dodavanje novih metoda i promjene na postojećim. *Python* programski jezik ima veliku zajednicu s više od 100,000 projekata na *PyPI* repozitoriju [1]. Među najpopularnijim

projektima za HTTP servere u *Pythonu* je *Flask* koji nudi veliku podršku i jaku zajednicu [13].

Za obradu i pokretanje tijekova rada je potreban programski jezik koji nudi veliku brzine, fleksibilnost tako da je moguće lagano implementirati nove radnje te da popularan. Popularnost jezika bi omogućila većini programera da mogu jednostavno dodati nove radnje. Prvi odabir koji zadovoljava tim kriterijima je *Java* programski jezik. Prema *TIOBE* Indexu siječnja 2019. [25] *Java* je najpopularniji programski jezik. *Benchmarkovi* programa pisanih u Javi pokazuju da su ti programi gotovo jednako brzi (ili čak i brži) od programa pisanih u *C/C++* programskom jeziku [2]. Takve rezultate Java postiže zahvaljujući *JIT (skr. engl. Just In Time) compileru* koji omogućuje sastavljanje koda u strojni kod tijekom njegovog pokretanja [27]. Jedna od velikih prednosti u odabiru programskog jezika je *OSGi* projekt koji omoguće moduliziranje aplikacija tako je moguće dodati, mijenjati i uklanjati funkcionalnosti aplikacije tijekom izvršavanja [9][10].

3.2. Opis implementacije mobilne aplikacije

3.2.1. *Swift* programski jezik

Swift programski jezik je objektno orijentirani programski jezik kojeg je *Apple* objavio 2014. godine na svojoj konferenciji *Apple World Wide Developer Conference* [14]. Specifikacija programskog jezika, a i sama implementacija kompajlera je otvorenog koda pod *Apache Licencom* [14]. *Swift* je danas dominantni programski jezik za izradu aplikacija za *Apple*-ove operacijske sustave *iOS, macOS, tvOS* i *watchOS* [14]. *Swift* je razvijen kao nasljednik *Objective-C* programskog jezika na čijoj je podlozi nastao [14]. *Objective-C* je objektno orijentirani programski jezik razvijen 1980-tih godina [28]. *Objective-C* uvodi interakciju čovjek-računalo u *C* jezik na principu *Smalltalk*-a te postaje jezik u kojem su programeri izrađivali aplikacije za *OSX* i *iOS* do uvođenja *Swift*-a. Za potrebe razvoja aplikacija, *Apple* u *Objective-C* razvija pristupne točke *(engl. application point interface, skr. API) Cocoa* za OSX i macOS i *Cocoa Touch* za *iOS, watchOS* i *tvOS* operacijske sustave.

Usporedba jednostavne klase koje ispisuje "Hello World" prilikom poziva metode `helloWorld` u konzolu.

```
class HelloWorld {
```

```
	func helloWorld() {
		print("Hello World")
	}
}
```

Kod 1: *Swift* verzija

```
// HelloWorld.h

@interface HelloWorld: NSObject {

}

- (void)helloWorld;

@end

// HelloWorld.m

#import "HelloWorld.h"

@implementation HelloWorld

-(void)helloWorld {
	NSLog(@"Hello World")
}
```

Kod 2: *Objective-C* verzija

3.2.2. Prilagodba rješenja za mobilne uređaje

Kako bi se korisnicima omogućilo što bolje korisničko iskustvo i olakšalo snalaženje po radnom prostoru aplikacije, potrebno je grafičke elemente radnji i poveznica između njih što optimalnije prikazivati. Poželjno je korisnicima omogućiti korištenje gesti sadržanih u operacijskom sustavu *iOS (npr. štipanje za uvećavanje ili povlačenje po ekranu za pomicanje sadržaja)*. Tome služi `UIScrollView` komponenta. `UIScrollView` nudi velike performanse i geste za uvećavanje i pomicanje sadržaja bez potrebe za ručnom implementacijom istih. Potrebno je samo definirati koji element se može uvećavati i veličinu prozora unutar kojeg se prikazuju elementi. Sama komponenta izračunava koji je dio prozora

vidljiv te renderira samo taj dio prozora. Time se, ako unutar `UIScrollViewa` puno elemenata renderiraju samo oni koji su vidljivi, a ne svi. Tako je moguće imati jako veliki broj elemenata unutar te komponente. Ako postoji više elemenata unutar `UIScrollView` komponente, renderiraju se samo vidljivi elementi, a ne svi koji se nalaze u komponenti. Na ovaj način je moguće sadržati veliki broj elemenata unutar komponente, bez potrebe za renderiranjem svih elemenata.

Svaki od elemenata se nakon pomicanja unutar `UIScrollView` komponente prebacuje na najbližu točku unutar komponente koja je djeljiva s 10. Tako je moguće lagano organizirati elemente. Točke djeljive s 10 su vizualno označene tako korisnik može lagano predočiti i organizirati radni prostor.

Nakon završetka pomicanja elementa mobilni uređaj korisnika lagano zavibrira kako bi dao korisniku do znanja da je radnja gotova i tako dao taktilni osjećaj upravljanja elementima.

3.2.3. Brand identitet, grafički dizajn i korisničko iskustvo

Ime aplikacije i platforme, *bitsmth*, je spoj dviju riječi: engleske riječi *bit (skr.od binary unit – binarna jedinica)*, najmanja jedinica podataka u računalu (0 ili 1) i riječi *smth*, izvedenice iz engleske riječi *smith*, onaj koji izrađuje nešto.

Grafički dizajn aplikacije i sam raspored elemenata inspiriran je *Shortcuts* aplikacijom. Ovakvim pristupom se korisnicima nudi poznato „*Apple look like“* korisničko iskustvo. Korisnici će lakše naučiti koristiti aplikaciju jer su im raspored i ergonomija aplikacije poznati i prirodni.

Prilikom grafičkog dizajna korisničkog sučelja i grafičkih elemenata birane su crne i bijele boje tako da se ističu boje tipova podataka i elemenata radnji te da se sadržaj koji korisnik izradi vizualno ističe od korisničkog sučelja.

Aplikacija se sastoji od 4 glavna zaslona: Prijava, Pregled radnih tijekova, Uređivanje i upravljanje radnim tijekom i Postavke

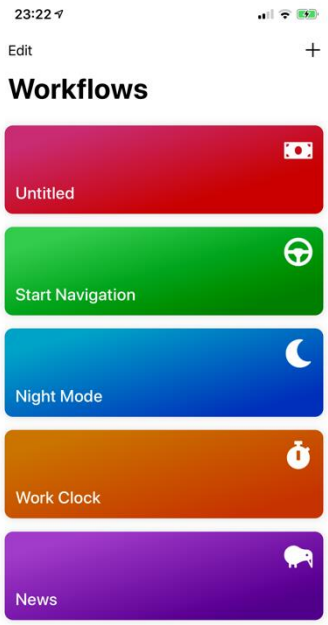
Prijava

Na zaslonu prijave korisnik može izraditi račun, odnosno može se prijaviti ako već ima korisnički račun.

Pregled radnih tijekova

Ovo je glavni zaslon u aplikaciji te je ujedno i zaslon koji će prijavljeni korisnici prvog vidjeti (početni zaslon) Tijekovi rada su predstavljeni u obliku pravokutnika s proizvoljnom bojom, ikonom i naslovom. Dodirom na tijek rada se pokreće isti, a dodirom ikone tri točke u gornjem desnom dijelu pravokutnika se otvara zaslon za uređenje tijeka rada. Korisnici imaju kontrolu nad atributima boje, izgleda ikone i naslova na zaslonu za uređivanje i upravljanje radnim tokom.

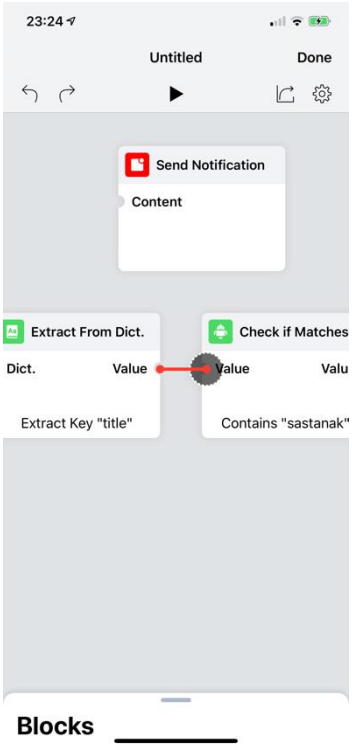
Na navigacijskoj traci u lijevom kutu se nalazi gumb koje omogućava uređivanje svih tijekova rada koje korisnik ima. Pritiskom na gumb prelazi se u način uređivanja radnih tijekova i tamo korisnik može mijenjati njihov redoslijed, brisati ih, kopirati, pristupiti zaslonu za uređivanje i upravljanje tijekova rada. Izlazak iz zaslona uređivanja korisnik ostvaruje dodirom gumba koji se nalazi na istom mjestu na kojem se nalazio i gumb za ulazak u zaslon uređivanja. U desnom kutu navigacijske trake korisnik ima gumb za dodavanje novog tijeka rada. Pritiskom na gumb za dodavanje novog tijeka rada stvara se prazni radni tok te se otvara zaslon za uređivanje i upravljanje novo stvorenog radnog toka. Na dnu navigacijske trake se nalazi traka za pretraživanje. Dodirom na traku za pretraživanje i unošenjem ključnih riječi korisnik može filtrirati radne tokove. Prilikom bilo kojeg unosa teksta u traku za pretraživanje automatski se skrivaju radni tokovi koji u svojem nazivu nemaju ključne riječi koje se pretražuje.



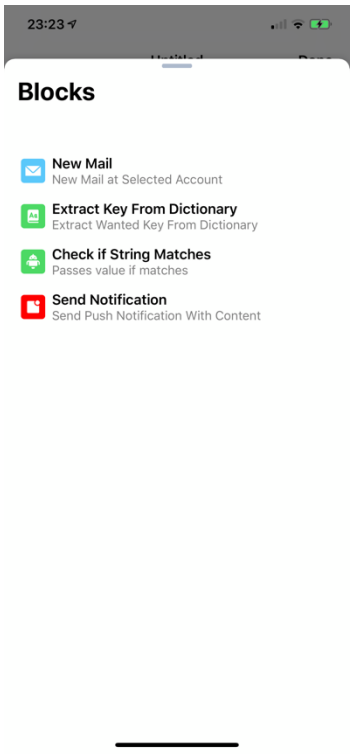
Slika 2: Popis radnih tijekova

Uređivanje i upravljanje radnim tijekom

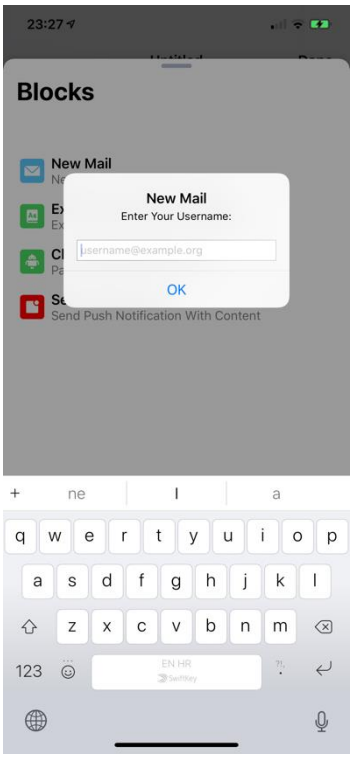
Na ovom zaslonu korisnici mogu manipulirati elementima vizualnoga programskog jezika te mogu pokretati radni tijek. Na ovom zaslonu će se prikazivati skočni prozori putem kojih će korisnik moći imati daljnju interakciju s radnim tijekom. U navigacijskoj traci ovog zaslona u gornjem desnom kutu nalazi se gumb za izlazak iz zaslona, u lijevom donjem kutu se alatna traka s gumbima za poništavanje i ponavljanje neke radnje poput dodavanja, brisanja elementa funkcije ili veze između elemenata. U donjoj sredini alatne trake nalazi se dugme za pokretanje ili zaustavljanje radnog toka ovisno o trenutnom stanju. U desnom donjem dijelu alatne trake nalazi se gumb koje vodi na zaslon za uređivanje atributa radnog toka poput imena, ikone i boje.



Slika 3. Uređivanje i upravljanje radnim tijekom



Slika 4: Popis dostupnih blokova



Slika 5: Vodič kroz dodavanje bloka „New Mail“

Postavke

Korisnici mogu se odjaviti iz aplikacije te povezati se s dostupnim servisima na ovom zaslonu. Na ovom zaslonu je prikazan popis servisa s kojim je moguća integracija te dodirom gumba "Poveži" korisniku se otvara zaslon za prijavu na odabrani servis. Ako je korisnik već povezan na neki određeni servis, ponuđena mu je opcija odjave sa servisa i pritiskom na nju korisnik više neće biti povezan s tim servisom. Prijavom na neki određeni servis korisniku se pojavljuju dodatni elementi funkcija vezani uz taj servis u zaslonu za uređivanje i upravljanje radnih tokova.

3.2.4. Arhitektura mobilne aplikacije

3.2.4.1. Upravljanje izvornim kodom

Za sav programski kod mobilne aplikacije se koristi distribuirani sustav za upravljanje izvornim kodom *Git* te pružatelj spremišta *GitHub.com,* koji nudi besplatno spremište za sav otvoreni izvorni kod. Svaki projekt ima svoj *git* repozitorij.

3.2.4.2. *Xcode* razvojno okruženje

Xcode je integrirano razvojno okruženje *(engl. IDE)* koje dostupno samo na *macOS* operacijskom sustavu [29]. Primarna upotreba mu je razvoj aplikacija za *iOS, tvOS i MacOS* platforme [29]. Aplikacije koje se objavljuju na *"App Store" (trgovina aplikacija za iOS, tvOS i MacOS platforme)* moraju biti *kompajlirane* koristeći *Xcode* jer je to jedini način da se potpišu svi potrebni digitalni certifikati aplikacije koje *Apple* zahtijeva za objavu na trgovini *App Store*.

3.2.4.3. *Cocoapods*

Cocoapods je projekt koji omogućuje integraciju knjižnica *Swift* ili *Objective-C* programskog koda iz centralnog repozitorija projekta u projekt aplikacije *Xcode* [30]. Bilo koja osoba može objaviti svoju knjižnicu u centralni repozitorij [30]. *Cocoapods* projekt je otvorenog koda napisan je u *Ruby* programskom jeziku [30]. *Ruby* programski jezik i razvojno okružje se instalira u sklopu *Xcode* aplikacije. Za korištenje *Cocoapods* projekta potrebno je instalirati *Cocoapods* program koristeći program za upravljanjem zavisnostima *(engl. dependency) Ruby* programskog jezika *"gem"*. Komanda za instalaciju je "`sudo gem install cocoapods`". Prilikom instalacije će se korisnika upitati za lozinku računa na kojeg je trenutno prijavljen te će se preuzeti program *Cocoapods* i instalirati na računalo korisnika. *Cocoapods* program se isključivo koristi putem terminala. Ako korisnik želi koristiti *Cocoapods* s *Xcode* projektom i tako instalirati sve željene verzije biblioteka, potrebno je u terminalu podesiti aktivni direktorij koristeći komandu `cd` u direktorij *Xcode* projekta i pokrenuti komandu "`pod init`" koja će stvoriti sve potrebne datoteke za korištenje *Cocoapodsa* u tom *Xcode* projektu. Jedna od novih datoteka je datoteka s nazivom "`Pods`". U njoj su definirani nazivi svih biblioteka koje će se preuzeti i integrirati s *Xcode* projektom pokretanjem komande "`pod install`". *Pod* datoteka je zapravo pisana u *Ruby* programskom jeziku.

Za stvaranje nove knjižnice koja se može integrirati koristeći *Cocoapods* potrebno je pokrenuti komandu "`pod lib create ImeBiblioteke`". Nakon pokretanja komande i odgovaranja na sva pitanja postavljena u terminalu generira se nova mapa s datotekom `ImeBiblioteke.podspec` i nekoliko datoteka koje služe kao početna točka za razvijanje biblioteke. Datoteka koja završava s "`podspec`" je datoteka koja čini mapu

Cocoapod bibliotekom. Nju je moguće ručno generirati, no jednostavnije je pokrenuti komandu i prilagoditi vrijednosti.

3.2.4.4. Organizacija projekta

Implementacija mobilne aplikacije je podijeljena u dva projekta. Prvi projekt je glavni dio koji sadrži svu poslovnu logiku aplikacije, drugi projekt je generirani kod pristupne točke potporne serverske aplikacije. Kod je generiran koristeći *Swagger* programom za generiranje koda iz *Swagger* specifikacije pristupne točke. Drugi projekt je prilagođen i napravljen u obliku *Cocoapod* biblioteke tako da se može jednostavno integrirati u glavni projekt.

Svaki projekt je organiziran u svoju mapu sa svojim pripadajućem *Git* repozitorijem. U prvom projektu koji je i glavni dio korišteni su *Cocoapods* za instalaciju svih potrebnih biblioteka uključujući i za integraciju drugog projekta s metodama za povezivanje s pristupnom točkom.

Podjela aplikacije na ovaj način omogućuje odvajanje osnovne poslovne logike, korisničkih sučelja i metoda za povezivanje s pristupnom točkom. Generiranje koda koristeći *Swagger* program direktno u glavni projekt bi stvorilo dodatne komplikacije i moguće konflikte između klasa pristupne točke i klasa poslovne logike aplikacije.

3.2.4.5. Raspored elemenata korisničkog sučelja

Za izradu rasporeda elemenata korisničkog sučelja je korištena biblioteka *PinLayout* koja omogućuje raspoređivanje direktno u kodu. Jedna alternativa tome je da se koriste *xib* ili *storyboard* datoteke koje su reprezentacije korisničkog sučelja koje se uređuju u vizualnom uredniku *Xcode* aplikacije. U njima je također moguće raspoređivanje elemenata, no uzevši u obzir da je *Xcode* novo razvojno okruženje na tržištu, *Storyboard* vizualni urednik je nepouzdan te je uređivanje kompliciranih struktura elementa kompliciranо. Druga alternativa je korištenje sistemske knjižnice *Auto Layout* koja također omogućuje raspoređivanje, ali je ona sporija od biblioteke *PinLayout* te pravila za raspoređivanje su više ograničena od funkcija biblioteke.

4. Opis implementacije potporne serverske aplikacije

4.1. Korištene metodologije

Za razvoj serverske aplikacije je korišten inkrementalni model u kojem je cilj bio da svaka nova verzija programskog koda uspješno izvršava dio zadane iteracije. U budućim iteracijama taj kod je dalje rafiniran i prilagođen novim iteracijama. Ovakav model rada je odabran jer korišteni standardi i projekti poput *OSGi*-a nisu imali pouzdane i kvalitetno dokumentirane implementacije i alate te je potrebno više puta usred iteracije ponovo napisati značajan dio *codebasea*, naročito jer neki dio možda nije bio kompatibilan s nekim dijelom implementacije *OSGi* projekta ili drugog korištenog projekta/alata.

4.2. *Java* programski jezik

Java programski jezik razvila je tvrtka *Sun Microsystems* 1995. godine [3]. On je objektno orijentiran programski jezik poput *C++* [3]. Klasificira se kao viši programski jezik odnosno programski jezik III. generacije [23][24]. Programski jezici III. generacije su jezici koji ne prikazuju naredbe u obliku ključnih riječi na razumljivome jeziku za razliku od programskih jezika I. generacije koji naredbe specifične za arhitekturu procesora daju kao instrukcijski set binarnog koda ili programskih jezika II. generacije – asemblerskih jezika, koji koriste ključne riječi i atribute po redcima, gdje je svaki redak nova naredba. Naredbe su kratke, od nekoliko slova, npr. `MOV` ili `ADDL` [23].

Ono što razlikuje *Javu* od, primjerice, *C++* je to što koristi JVM *(engl. Java Virtual Machine)* [6]. To je virtualni stroj koji se pokreće u obliku aplikacije putem kojeg se izvršavaju programi pisani u *Java* programskom jeziku [6]. Prednost toga je to što program pisan u *Java* programskom jeziku se može pokrenuti na bilo kojoj arhitekturi procesora bez potrebe da se program ponovo kompajlira za tu arhitekturu procesora [6]. JVM pokreće *Java* strojni kod *(engl. Java Byte Code)*, koji *Java* kompajleri stvaraju iz *Java* programskog koda [6].

4.2.1.1. *Scala*

Scala programski jezik je jezik koji proizlazi iz *Java* programskog jezika, ima drugačiju i moderniju sintaksu te omogućuje dodatne značajke [31]. Program pisan u *Scala* programskom jeziku se *kompajlira* u Java strojni kod kojeg pokreće JVM [31]. Ne postoji distinkcija između kod pisanog u *Java* ili *Scala* programskom jeziku. *Scala* može u većini slučajeva ponuditi značajnu razliku u količini napisanog koda koji radi istu stvar [31].

4.2.2. *Akka* framework

Akka framework je knjižnica napisana u *Java* programskom jeziku [5]. Ona je besplatna i otvorenog koda [5]. Implementira model glumaca za *Java* programski jezik [5]. Model glumaca u programskim jezicima predstavlja apstrakciju gdje je objekt "glumac" *(engl. actor)* [4]. Takav objekt ima mogućnost primanja poruka [4]. U skladu s primljenom porukom objekt ima mogućnost modifikacije varijabli unutar objekta, stvaranja još objekta "glumaca" ili slanja još poruka [4]. Objekti mogu mijenjati svoje stanje, no ne mogu mijenjati stanje drugih objekta. [4] Iz toga razloga su idealan model za izvođenje više paralelnih zadataka bez korištenja blokada poput mutexa [4].

4.2.3. *RabbitMQ* servis

RabbitMQ je serverska implementacija AMQP protokola [18]. AMQP protokol omogućuje razmjenu poruka između više programa koji mogu biti pisani u drugačijim programskim jezicima, i mogu se nalaziti na različitim računalima [19]. *RabbitMQ* omogućuje stvaranje različitih kanala na koje se onda može i preplatiti putem klijentske implementacije AMQP protokola.

AMQP protokol ne uzima pretpostavke oko formata poruka koje obrađuje i šalje [20]. Te poruke mogu biti i u binarnom obliku, a i u JSON formatu.

4.2.4. *Python* programski jezik

Python programski jezik je dinamičan objektno orijentirani jezik [11]. Kao programski jezik lagan je za učenje zbog dinamičnosti i sintakse koja je opuštenija od jezika poput *Jave* i *C++* [11]. Za izvršavanje određene radnje potrebno je napisati manje koda nego, primjerice, u *Javi* ili *C++* programskom jeziku. U istoj datoteci moguće je imati više implementacija klasa, nije potrebno eksplicitno izražavati tipove podataka niti se držati određenog tipa

varijable. Unatoč tim mogućnostima moguće je i opcionalno naglasiti tip varijable što omoguće analizu koda za provjeru da li se varijable prikladno postavljaju i koriste, prema naglašenom tipu varijabli. Za implementaciju pristupne točke mobilne aplikacije ovog završnog rada je odabran *Python* zbog brzine pisanja repetitivnog koda i lakoće rada s ORM knjižnicama.

4.2.5. Arhitektura serverske aplikacije

Serverska aplikacija se sastoji od dva dijela: pristupne točke i dijela koji izvršava radnje koje su korisnici definirali u radnom tokovima.

4.2.5.1. Izvršitelj radnih tokova

Izvršitelj radnih tokova je aplikacija napisana u *Java* programskom jeziku. Koristi *OSGi* standard za upravljanje kodom. Kod aplikacije je podijeljen u *OSGi* „module“. Te module je moguće tijekom rada aplikacije dodavati, upravljati i brisati. Moduli prema *OSGi* projektu su JAR datoteke koje unutar sebe imaju datoteku s nazivom `MANIFEST.MF` [10]. Ta datoteka sadrži informacije o tome modulu [10]. Moduli su često implementirani tako da postoji modul sa specifikacijom implementacije i modul s implementacijom [10]. Na taj način je moguće koristiti neki modul isključivo putem specifikacije modula bez da direktnog pristupanja implementaciji. Takvo odvajanje specifikacije i implementacije omogućuje da se moduli nadograđuju ili mijenjaju bez utjecaja na ostatak aplikacije. Moduli specifikacije sadrže apstraktne klase i *interface* elemente koje moduli implementacije trebaju implementirati. U tom obliku se definira neki standard za taj modul.

Za *OSGi* implementaciju se odabran je *Apache* projekt, projekt otvorenog koda izdan od strane *Apache Software Foundation*. Bilo tko može napisati svoju implementaciju modula prema nekoj specifikaciji te taj modul ugraditi u aplikaciju. Primjerice, netko može napisati implementaciju modula koji pretvara XML oblik podataka u *Java* objekt i obrnuto. Takva implementacija bi omogućila da se šalju poruke putem AMQP sustava u XML obliku.

Tablica 1: Popis modula aplikacije izvršitelja radnih tokova

Naziv modula	Opis funkcionalnosti
`act-api`	Specifikacija za implementaciju apstrakcije „glumaca“ *(engl. actors).* Omogućuje stvaranje i upravljanje „glumcima“ i sustavom glumaca.

`act-akka`	Implementacija specifikacije apstrakcije „glumaca“ *(engl. actors)* koristeći *Akka* projekt.
`commons`	Klase koje se koriste kroz sve module. U ovom slučaju ne postoji podjela na modul specifikacije i implementacije jer su ovdje implementirane jednostavne klase koje se neće često mijenjati kroz verzije.
`kernel`	Implementacija funkcionalnosti aplikacije. Obrađuje sve poruke dobivene preko instanci `transmit-api` klasa. Te poruke sadrže upute za upravljanje radnim tokovima ili pokretanje neke određene radnje. Upute iz tih poruka su izvršene u ovom modulu. Implementacije klasa radnji također se nalaze u ovom modulu.
`kernel-api`	Specifikacije implementacije funkcionalnosti aplikacije.
`marshaller-api`	Specifikacija za klase koje pretvaraju binarni ili tekstualni oblik podataka u objekte određenih klasa.
`marshaller-json`	Implementacije klasa koje pretvaraju binarni ili tekstualni oblik podataka u objekte određenih klasa.
`transmit-amqp`	Implementacija AMQP standarda za slanje i primanje poruka preko AMQP sustava. Koristi knjižicu za spajanje na *RabbitMQ* sustav putem kojeg razmjenjuje podatke s pristupnom točkom.
`transmit-api`	Specifikacija za klase koji omogućuju slanje i primanje poruka preko nekog sustava. Te klase mogu omogućiti slanje poruke u druge aplikacije i primanje poruka iz drugih aplikacije ili na drugim računalima.

4.2.5.2. Pristupna točka

Pristupna točka je pisana u *Python* programskom jeziku. Metode pristupne točke vraćaju i primaju podatke u JSON obliku. Metode omogućuju upravljanje radnih tokova i korisničkom računom. Pristup svim metodama je zaštićen *OAuth* 2.0 protokolom te svi klijenti koji pristupaju metodama trebaju se predstaviti u zaglavlju HTTP upita s

pripadajućim tajnim podacima koji predstavljaju određenu aplikaciju. Za pristup metodama koje omogućuju upravljanje radnim tokovima ili korisničkim računom klijent se treba predstaviti tokenom koji reprezentira specifičnog korisnika. Taj token klijent može dobiti slanjem ispravnog i postojećeg korisničkog imena i lozinke pomoću metode koja kao odgovor šalje token. Ukoliko krajnji korisnik nije registriran, moguće se registrirati putem metode za registraciju. Ta metoda ne zahtijeva token korisnika. Za registraciju metoda je potrebno poslati željeno korisničko ime, email i lozinku. Ukoliko je registracija uspješna klijent dobiva token.

Za pozivanje većine metoda pristupna točka u pozadini komunicira s bazom podataka u koju sprema podatke o radnim tokovima, korisnicima i korisničkim tokenima za pristup metodama. Baza podataka koja se koristi je *PostgreSQL* verzije 11.0, a za komunikaciju se koristi ORM *(skr. engl. Object-relational mapping)* sloj koji omogućuje pristupanje objektima u bazi podataka bez pisanja SQL upita.

4.2.6. Tehnički zahtjevi serverske aplikacije

Serverska aplikacija koristi *PostgreSQL* bazu podataka u pozadini te ju aplikacija zahtijeva za korištenje. Minimalna verzija *PostgreSQL* baze podataka je 11.0. Za slanje poruka između aplikacija koristi se implementacija AMQP protokola *RabbitMQ*. Minimalna podržana verzija *RabbitMQ-a* je 3.7.9. *RabbitMQ* u pozadini koristi *Redis* aplikaciju, koju je također potrebno instalirati. Obzirom da se koristi *Java* programski jezik za određene aplikacije te da JVM koristi više memorije od programa pisanih u, primjerice, *C* i *C++*, potrebno je osigurati dovoljan memorijski kapacitet na serveru te valjanu instalaciju *Java* okruženja. Implementacijama *OSGi* standarda je potrebno više vremena za inicijalno pokretanje aplikacije što je također potrebno uzeti u obzir. Pristupna točka je pisana u *Python* programskom jeziku te je potrebno osigurati *Python* okruženje minimalno verzije 3.6. Obzirom da aplikacija koja izvršava radne tokove ima varijabilnu količinu opterećenja (opterećenje ovisi o ponašanjem korisnika) te da ta opterećenja mogu blokirati sve resurse servera preporučuje se odvajanje aplikacije pristupne točke i aplikacije za izvršenje radnog toka. Tako odvajanje moguće je i obliku nekog rješenja virtualizacije.

4.2.7. Prilagodbe implementacije s obzirom na opterećenje servera

Postoje dva dijela serverske aplikacije i zbog toga je svaki dio moguće zasebno izvršavati na odvojenim serverima. Takav pristup omogućuje kontrolu i prikladnu reakciju na opterećenje servera. Ukoliko se koriste *cloud* ili druga rješenja virtualizacije u slučaju da postoji veliko opterećenje na pristupnoj točki, moguće je prilagoditi specifikaciju servera pristupne točke. Pošto pristupna točka dalje komunicira samo s bazom podataka i AMQP sustavom, moguće je pokrenuti više instanci pristupnih točaka na više servera te koristiti *load-balancere* za dinamički odabir na koja će instanca zaprimiti i obraditi HTTP zahtjev. Takav pristup omogućuje jednostavnu, efikasnu i brzu prilagodbu na opterećenje servera. Obzirom da aplikacija za izvršavanje radnih tokova obrađuje poruke AMQP sustava, moguće je pokrenuti više instanci aplikacije na više servera kako bi se smanjilo opterećenje.

5. Rezultati i rasprava

Kao prototip u ovom radu izrađen je radni tijek koji provjerava pretinac e-pošte, pretražuje naslov za ključne riječi i kreira podsjetnik za rješavanje upita iz maila do kraja radnog vremena.

Slika 6: Pojednostavljeni dijagram radnji radnog tijeka

Radni tijek je testiralo sedam korisnika, različitog spola, u dobi između 18 i 50 godina i različite informatičke pismenosti. Korisnici su tijekom testiranja ispunjavali formu za ispitivanje u koju su bilježili svoja opažanja (Prilog 1.)

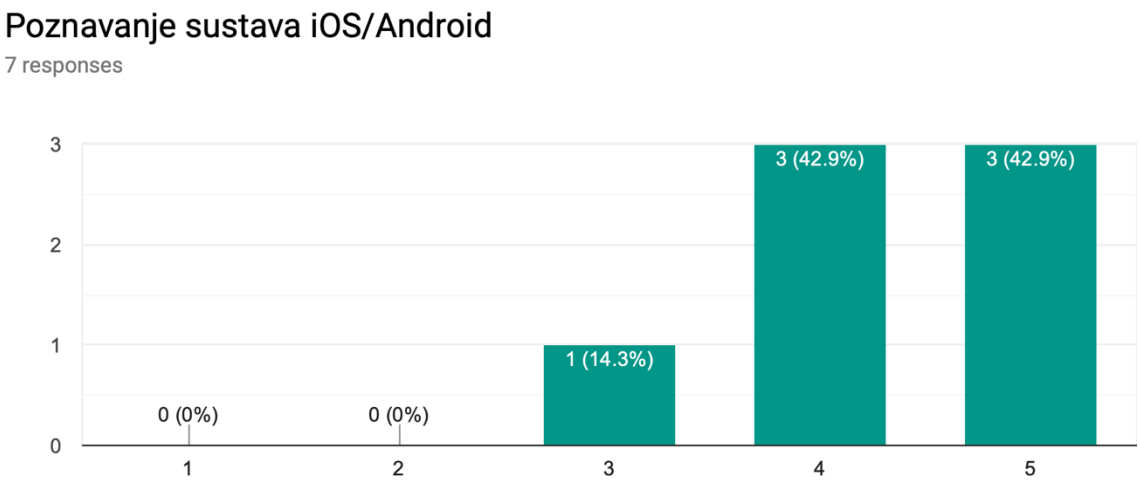
Osnovni podaci korisnika koji su sudjelovali u testiranju nalaze se u tablici ispod:

Tablica 2: Osnovni podaci o korisnicima koji su sudjelovali u testiranju

Tester	Spol	Dob	Stručna sprema	Mobilni uređaj	Koristio *IFTTT*	Koristio *Apple Shortcuts*
1	MUŠKI	23	SSS/SŠS	*iPhone X/XS/XS Max/XR*	NE	DA
2	MUŠKI	18	NKV	*iPhone 8/8+*	NE	DA
3	ŽENSKI	43	VSS	*iPhone 7/7+*	NE	NE
4	ŽENSKI	21	SSS/SŠS	*iPhone 6/6+*	NE	NE
5	MUŠKI	56	VŠS	*iPhone 6s/6s+*	NE	NE
6	ŽENSKI	52	VSS	*iPhone 8/8+*	NE	NE
7	MUŠKI	23	VSS	*iPhone 7/7+*	DA	DA

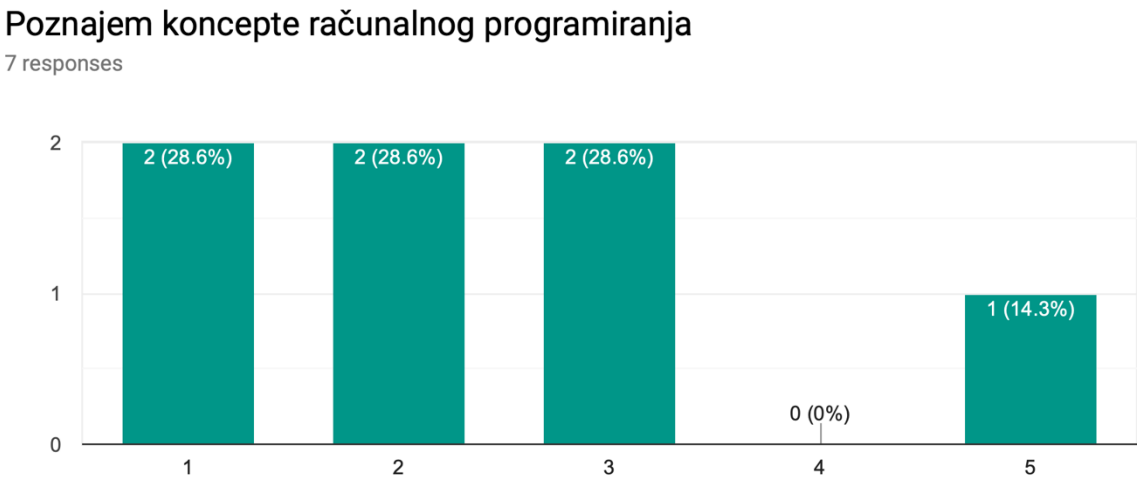
Četiri od sedam korisnika ocijenilo je svoje poznavanje operacijskih sustava *Windows*/*Linux*/*macOS* vrlo dobrim, dvoje je to poznavanje ocijenilo dobrim i dvoje je to poznavanje ocijenilo odličnim.

Svi testeri ocjenjuju svoje znanje i poznavanje sustava *iOS* i *Android* minimalno dobrim (Slika 6).



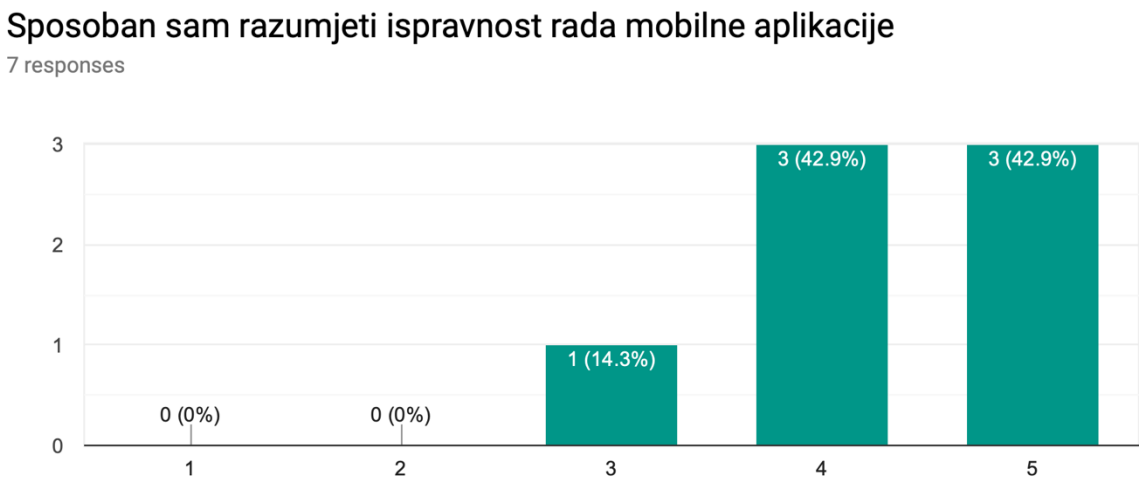
Slika 7: Poznavanje sustava *iOS/Android*, izvor: *Google Forms* obrazac za testiranje, veljača 2019.

Od sedam korisnika, samo jedan korisnik (14,3%) ocjenjuje svoje poznavanje koncepata programiranja ocjenom izvrstan, dok ostali ocjenjuju s maksimalno dobar (28,6%) (Slika 7).

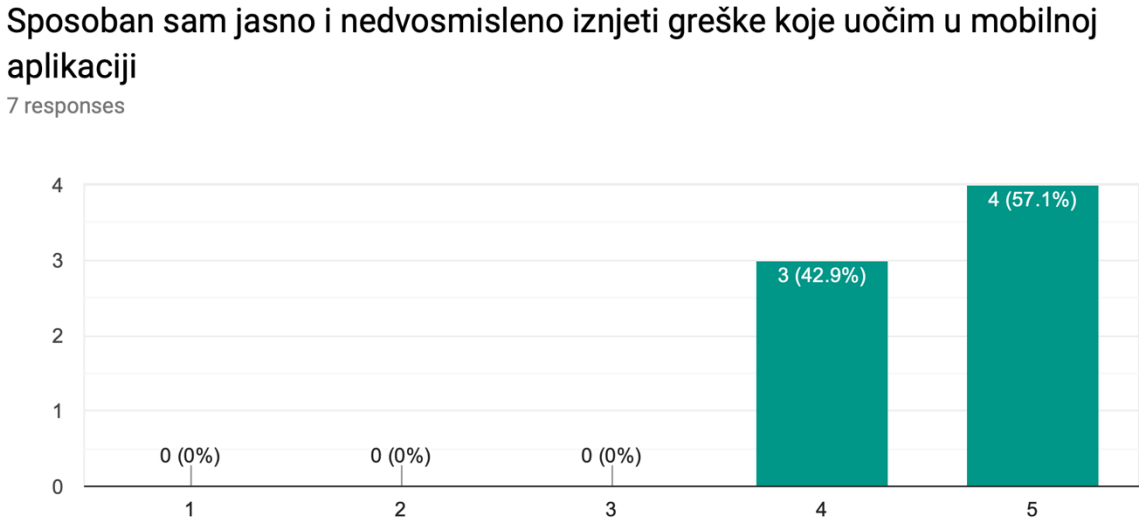


Slika 8: Samoprocjena poznavanja koncepata računalnog programiranja korisnika, izvor: *Google Forms* obrazac za testiranje, veljača 2019.

Svi korisnici su izjavili da minimalno dobro mogu odrediti ispravnost rada mobilne aplikacije (Slika 8) te da vrlo dobro mogu iznijeti uočene greške (Slika 9).



Slika 9: Samoprocjena poznavanja koncepata računalnog programiranja korisnika, izvor: *Google Forms* obrazac za testiranje, veljača 2019.

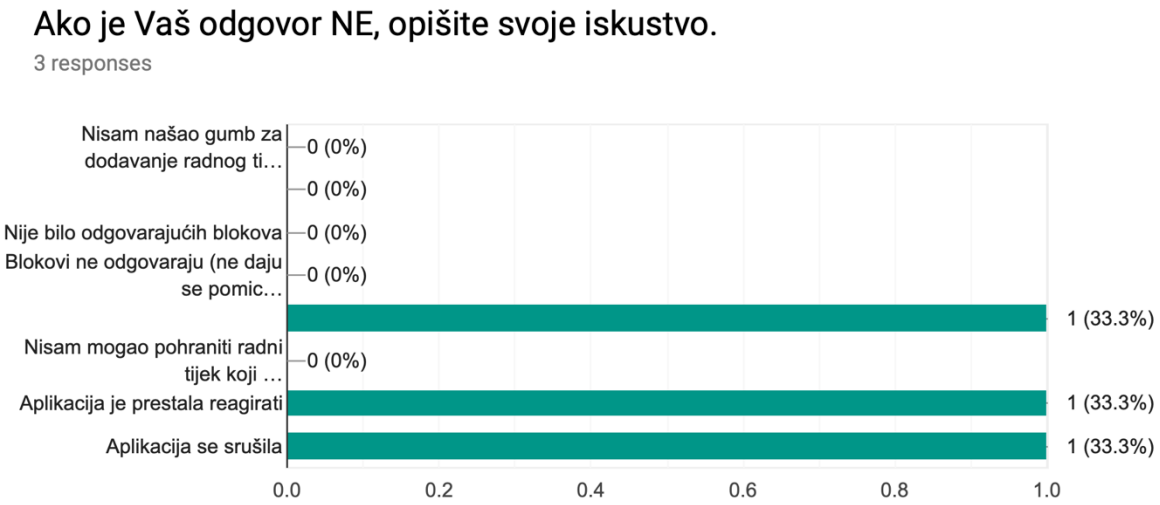


Slika 10: Samoprocjena mogućnosti iznošenja kritike mobilne aplikacije, izvor: *Google Forms* obrazac za testiranje, veljača 2019.

5.1. Rezultati testiranja korisnika

Korisnici su prema obrascu za testiranje mobilne aplikacije *bitsmth* u svrhu testiranja odgovarali na 10 pitanja o funkcionalnostima mobilne aplikacije. 100% korisnika uspješno je pokrenulo mobilnu aplikaciju na svojim pametnim telefonima, registriralo se na servis i

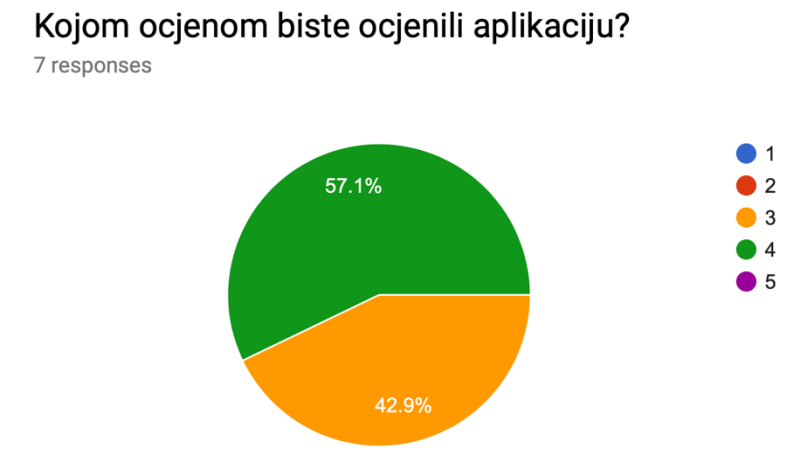
prijavilo vjerodajnicama. Četiri od sedam korisnika uspješno je izradilo i pokrenulo radni tijek, koji se izvršio. Dvoje korisnika aplikacija se „zaledila“ ili „srušila“, dok jedan korisnik nije mogao „povezati blokove“.



Slika 11: Razlozi rušenja aplikacije, izvor: *Google Forms* obrazac za testiranje, veljača 2019.

5.2. Povratne informacije

Unatoč greškama na koje su neki korisnici naišli, prosječna ocjena aplikacije iznosi visokih 4,85. Aplikacija je korisnike privukla svojom jednostavnošću, intuitivnošću i brzinom rada. Korisnici su pri testiranju uzimali u obzir da se radi o prototipu.



Slika 12: Prosječna ocjena aplikacije, izvor: *Google Forms* obrazac za testiranje, veljača 2019.

6. Zaključak

Ovaj rad je prototip i prikazuje da je moguće napraviti vizualni programski jezik koji je jednostavan i lagan za korištenje na mobilnim uređajima. Mobilni uređaji polagano zamjenjuju sve više aspekta naših života. Jedan od aspekata koje nikada nisu zamijenili niti je jednostavno zamijeniti je programiranje i upravljanje podacima. Temeljem ovog rada moguće je razviti mnoge funkcionalnosti koje olakšavaju izvedbu svakodnevnih repetitivnih zadataka. Primjerice, koristeći ovakav način programiranja bilo bi moguće isprogramirati aplikacije koje bi se pokretale na računalima i serverima. Zahvaljujući jakim servisima u oblaku, moguće je procesorsku moć računala programera prebaciti u oblak, te mobilni uređaji mogu iskoristi tu moć i omogućiti programiranje i razvijanje u hodu.

Rad je zamišljen kao lagano nadogradiv. Svaki dio se da zamijeniti i prilagoditi bez prevelikih modifikacija ostalih dijelova. To su omogućili programski jezici *Java*, *Python* i *Scala* te projekti poput *OSGi*-a. Vizija je da se ovakva platforma razvije i da se stvori zajednica programera i entuzijasta koji bi nadograđivali rad i proširivali ga da radi s više sustava i više podataka.

Popis kratica

AMQP	*Advanced Message Queuing Protocol*
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated development environment*
IEC	*International Electrotechnical Commission*
IFTTT	*If This Then That*
IT	*Information technology*
JIT	*Just-in-time (JIT) compilation*
JSON	*JavaScript Object Notation*
JVM	*Java virtual machine*
ORM	*Object-relational mapping*
OSGi	*Open Services Gateway initiative*
SQL	*Structured Query Language*
WWDC	*Apple Worldwide Developers Conference*

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] PyPi: https://pypi.org/, siječanj 2019.

[2] Java Benchmark Game: https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/java.html, siječanj 2019.

[3] Java Programski jezik: https://hr.wikipedia.org/wiki/Java_(programski_jezik)), siječanj 2019.

[4] Actor Model: https://en.wikipedia.org/wiki/Actor_model, siječanj 2019.

[5] Akka Actor Framework Docs: https://akka.io/docs/, siječanj 2019.

[6] Java Virtual Machine: https://en.wikipedia.org/wiki/Java_virtual_machine, siječanj 2019.

[7] Apple Shortcuts app Guide: https://support.apple.com/guide/shortcuts/welcome/ios, siječanj 2019.

[8] OSGi™ Alliance: https://www.osgi.org, siječanj 2019.

[9] What is OSGi: https://www.osgi.org/developer/what-is-osgi/, siječanj 2019.

[10] OSGi: https://en.wikipedia.org/wiki/OSGi, siječanj 2019.

[11] Python Porgramming Language: https://en.wikipedia.org/wiki/Python_(programming_language), siječanj 2019.

[12] Python 3 Docs: https://docs.python.org/3/, siječanj 2019.

[13] Flask: http://flask.pocoo.org, siječanj 2019.

[14] Swift Programming Language: https://en.wikipedia.org/wiki/Swift_(programming_language), siječanj 2019.

[15] Swift Book: https://docs.swift.org/swift-book/, siječanj 2019.

[16] IFTTT: https://ifttt.com, siječanj 2019.

[17] IFTTT: https://en.wikipedia.org/wiki/IFTTT, siječanj 2019.

[18] RabbitMQ project: https://www.rabbitmq.com, siječanj 2019.

[19] Advanced Message Queuing Protocol Workgroup: https://www.amqp.org, siječanj 2019.

[20] Advanced Message Queuing Protocol: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol, siječanj 2019.

[21] IEC 61499: https://en.wikipedia.org/wiki/IEC_61499, siječanj 2019.

[22] SQL: https://en.wikipedia.org/wiki/SQL, siječanj 2019.

[23] Programming languagegenerations: https://en.wikipedia.org/wiki/Programming_language_generations, siječanj 2019.

[24] Istorijski Razvoj Programskih Jezika: https://raf.edu.rs/citaliste/programiranje/3673-istorijski-razvoj-programskih-jezika, siječanj 2019.

[25] Tiobe Indeks: https://www.tiobe.com/tiobe-index/ , siječanj 2019.

[26] IFTTT Partners FAQ: https://platform.ifttt.com/docs/faq, siječanj 2019.

[27] Java Programming Language: https://en.wikipedia.org/wiki/Java_(programming_language) , siječanj 2019.

[28] Objective-C Programming Language: https://en.wikipedia.org/wiki/Objective-C , siječanj 2019.

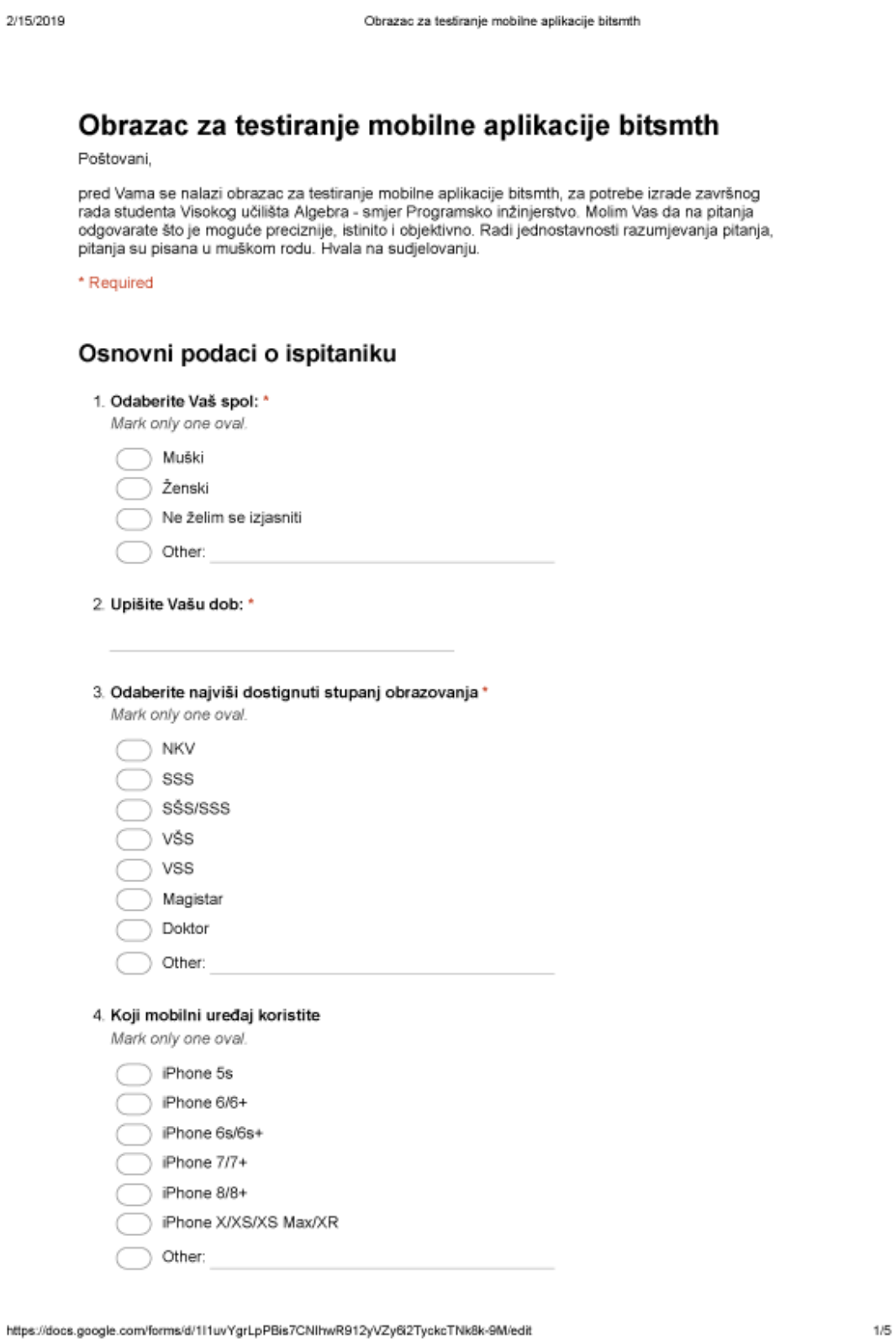
[29] Xcode: https://en.wikipedia.org/wiki/Xcode, siječanj 2019.

[30] Cocoapods: https://en.wikipedia.org/wiki/CocoaPods, siječanj 2019.

[31] Scala Programming Language: https://en.wikipedia.org/wiki/Scala_(programming_language) , siječanj 2019.

Prilozi

Prilog 1: Obrazac za testiranje mobilne aplikacije *bitsmith*



2/15/2019 Obrazac za testiranje mobilne aplikacije bitsmth

Obrazac za testiranje mobilne aplikacije bitsmth

Poštovani,

pred Vama se nalazi obrazac za testiranje mobilne aplikacije bitsmth, za potrebe izrade završnog rada studenta Visokog učilišta Algebra - smjer Programsko inženjerstvo. Molim Vas da na pitanja odgovarate što je moguće preciznije, istinito i objektivno. Radi jednostavnosti razumjevanja pitanja, pitanja su pisana u muškom rodu. Hvala na sudjelovanju.

* Required

Osnovni podaci o ispitaniku

1. **Odaberite Vaš spol:** *
Mark only one oval.
- ◯ Muški
- ◯ Ženski
- ◯ Ne želim se izjasniti
- ◯ Other: _____

2. **Upišite Vašu dob:** *
_____

3. **Odaberite najviši dostignuti stupanj obrazovanja** *
Mark only one oval.
- ◯ NKV
- ◯ SSS
- ◯ SŠS/SSS
- ◯ VŠS
- ◯ VSS
- ◯ Magistar
- ◯ Doktor
- ◯ Other: _____

4. **Koji mobilni uređaj koristite**
Mark only one oval.
- ◯ iPhone 5s
- ◯ iPhone 6/6+
- ◯ iPhone 6s/6s+
- ◯ iPhone 7/7+
- ◯ iPhone 8/8+
- ◯ iPhone X/XS/XS Max/XR
- ◯ Other: _____

https://docs.google.com/forms/d/1l1uvYgrLpPBis7CNIhwR912yVZy6i2TyckcTNk8k-9M/edit 1/5

5. **Jeste li koristili aplikaciju IFTTT** *
Mark only one oval.

- ◯ Da
- ◯ Ne

6. **Jeste li koristili aplikaciju Apple Shortcuts** *
Mark only one oval.

- ◯ Da
- ◯ Ne

Informatička pismenost

Procjenite Vašu informatičku pismenost ocjenjivanjem sljedećih tvrdnji od 1 do 5. U svako pitanje nalazi se objašnjenje skale.

7. **Poznajem i koristim operativni sustav Windows/macOS/Linux** *
Mark only one oval.

	1	2	3	4	5	
uopće ili nedovoljno sam upoznat s funkcionalnostima u sklopu tvrdnje	◯	◯	◯	◯	◯	sposoban sam napredno koristiti navedene funkcionalnosti u sklopu tvrdnje.

8. **Poznavanje sustava iOS/Android** *
Mark only one oval.

	1	2	3	4	5	
uopće ili nedovoljno sam upoznat s funkcionalnostima u sklopu tvrdnje	◯	◯	◯	◯	◯	sposoban sam napredno koristiti navedene funkcionalnosti u sklopu tvrdnje.

9. **Poznajem koncepte računalnog programiranja** *
Mark only one oval.

	1	2	3	4	5	
uopće ili nedovoljno sam upoznat s funkcionalnostima u sklopu tvrdnje	◯	◯	◯	◯	◯	sposoban sam napredno koristiti navedene funkcionalnosti u sklopu tvrdnje.

10. **Sposoban sam razumjeti ispravnost rada mobilne aplikacije** *
Mark only one oval.

	1	2	3	4	5	
uopće se ne slažem	◯	◯	◯	◯	◯	u potpunosti se slažem

11. **Sposoban sam jasno i nedvosmisleno iznjeti greške koje uočim u mobilnoj aplikaciji** *
Mark only one oval.

	1	2	3	4	5	
uopće se ne slažem	◯	◯	◯	◯	◯	u potpunosti se slažem

Upitnik o zadovoljstvu radom i korisničkim iskustvom u aplikaciji

U nastavku se nalazi set pitanja potrebnih za uspješno testiranje aplikacije. Nakon što ste koristili aplikaciju prema danim uputama, molim da iskreno i objektivno odgovorite na postavljena pitanja. Kako biste započeli s unosom, pritisnite na nastavi.

12. **Aplikacija se pokrenula na mom mobilnom uređaju (zaslon za prijavu se prikazao)** *
Mark only one oval.

- ◯ Da
- ◯ Ne

13. **Ukoliko je Vaš odgovor bio NE, opišite Vaše iskustvo**
Check all that apply.

- ☐ Aplikacija se srušila prilikom pokretanja
- ☐ Aplikacija se pokrenula, no zaslon za prijavu se nije pojavio
- ☐ Aplikacija se pokrenula, zaslon za prijavu se pojavio, aplikacija se srušila
- ☐ Other: _____

14. **Uspješno sam se registrirao na servis** *
Mark only one oval.

- ◯ Da
- ◯ Ne

15. **Ako je Vaš odgovor NE, opišite Vaše iskustvo.**
Gumb ne odgovara - pritisak na gumb ne pokrene ništa; Ništa se ne prikazuje - nema elemenata na zaslonu (zaslon je prazan)
Check all that apply.

- ☐ Gumb za registraciju ne odgovara
- ☐ Dodir na gumb za registraciju otvori sljedeći zaslon, no ništa se ne prikazuje
- ☐ Upisao sam podatke za registraciju, no gumb za dovršetak reakcije ne odgovara
- ☐ Other: _____

Vjerodajnice su vaše izabrano korisničko ime i lozinka.

16. **Uspješno sam se prijavio postojećim vjerodajnicama (svojim korisničkim imenom i lozinkom) na servis** *
Mark only one oval.

- ◯ Da
- ◯ Ne

17. **Ako je Vaš odgovor NE, opišite Vaše iskustvo:**
Check all that apply.

- [ ] Aplikacija odbija moje vjerodajnice
- [ ] Aplikacija prihvaća moje vjerodajnice, ali ništa se ne prikazuje
- [ ] Aplikacija mi ne dozvoljava upis vjerodajnica
- [ ] Other: _____

18. **Uspješno sam kreirao radni tijek koji mi je zadan za testiranje.** *
Mark only one oval.

- ◯ Da
- ◯ Ne

19. **Ako je Vaš odgovor NE, opišite svoje iskustvo.**
Check all that apply.

- [ ] Nisam našao gumb za dodavanje radnog tijeka
- [ ] Uspješno sam kreirao radni tijek, no nisam mogao dodati elemente sučelja (blokove)
- [ ] Nije bilo odgovarajućih blokova
- [ ] Blokovi ne odgovaraju (ne daju se pomicati, uređivati tekst na njima)
- [ ] Nije bilo "žica" (Nisam mogao povezati blokove)
- [ ] Nisam mogao pohraniti radni tijek koji sam kreirao
- [ ] Other: _____

20. **Da li je okidač pokrenuo radni tijek?** *
Mark only one oval.

- ◯ Da
- ◯ Ne

21. **Ako je Vaš odgovor ne, opišite Vaše iskustvo.**
Check all that apply.

- [ ] Nije se kreirao podsjetnik
- [ ] Aplikacija nije pronašla pošiljatelja
- [ ] Aplikacija nije pronašla definiranu ključnu riječ
- [ ] Other: _____

Sveukupno zadovoljstvo

Molim Vas da ocjenite sveukupno zadovoljstvo aplikacijom i korisničkim iskustvom ocjenom od jedan do pet, gdje je 1- uopće nisam zadovoljan, a 5 - u potpunosti sam zadovoljan.

22. **Aplikacija je jednostavna i razumljiva za upotrebu.** *
Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

23. **Boje i oblici korišteni u aplikaciji prepoznatljivi su i intuitivni** *
Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

24. **Način na koji kreiram radne tijekove je jednostavan i intuitivan** *
Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

25. **Način na koji definiram elemente radnog tijeka je jednostavan i intuitivan** *
Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

26. **Radni tijek i njegovi elementi su pregledni i lako se navigira po njima.** *
Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

27. **Kako biste opisali Vaše iskustvo korištenja aplikacije?**

28. **Kojom ocjenom biste ocjenili aplikaciju?**
Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

Hvala na sudjelovanju.