Sustav za upravljanje radnim tokom dokumenata

Škoda, Tomislav

Undergraduate thesis / Završni rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, University of Split, Faculty of science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:855775

Rights / Prava: Attribution-NonCommercial 4.0 International/Imenovanje-Nekomercijalno 4.0 međunarodna

Download date / Datum preuzimanja: **2024-04-20**

Repository / Repozitorij:

Repository of Faculty of Science

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

Sustav za upravljanje radnim tokom dokumenata

Tomislav Škoda

Split, rujan 2015.

Sadržaj:

1. Uvod 1

1.1. Sustav za upravljanje dokumentima 2

1.1.1. Povijest 2

1.1.2. Komponente sustava za upravljanje dokumentima 2

1.2. Radni tok 3

1.2.1. Sustav za upravljanje radnim tokom 4

1.2.2. Tipovi sustava za upravljanje radnim tokom 4

1.2.3. Model radnog toka 6

1.2.4. Arhitektura 10

2. Google Apps Script 12

2.1. Sučelje 12

2.1.1. Zapisnik (eng. *Logger*) 13

2.1.2. Otklanjanje grešaka (eng. *Debugger*) 14

2.2. Baza podataka 14

2.2.1. JDBC Service (Java Database Connectivity) 14

2.2.2. Microsoft Azure 15

3. Implemetacija *Helpdesk* aplikacije u Google Apps Script-u 16

3.1. Opis sustava 16

3.2. Radni tok smetnje 17

3.3. Funkcionalnosti aplikacije 18

3.3.1. Sučelje za korisnika 19

3.3.2. Sučelje za tehničara 21

3.3.3. Sandbox način rada 22

3.4. Izgled aplikacije 24

3.4.1. Bootstrap 24

Zaključak **26**

Literatura **27**

1. Uvod

Dokument je pisana, naslikana, predstavljena ili snimljena reprezentacija misli. Izraz je nastao od latinskog *Documentum* što znači lekcija. U prošlosti se najčešće koristio kao termin za pisanu potvrdu korištenu kao dokaz. U informacijsko doba većinom se koristi kao izraz za tekstualnu datoteku, zajedno sa svojom strukturom i dizajnom, uključujući vrste slova (*eng. fonts*), boje i slike. Nastankom elektroničkih dokumenata, termin „dokument“ više ne može biti određen načinom na koji se prenosi, odnosno svojim transimisijskm medijem kao što je to u prošlosti bio papir. Službeni termin dokumenta je definiran u „Library and information science“ („Knjižničarstvo i informacijske znanosti“) kao osnovni teorijski konstrukt. To je sve što može biti sačuvano ili predstavljeno tako da služi kao dokaz za neku svrhu.[1] Francuska knjižničarka Suzzane Briet sa nekoliko zanimljivih primjera opisuje ovu definiciju dokumenta. Primjer glase: „ *Zvijezda na nebu nije dokument ali fotografija te zvjezde jest; kamen u rijeci nije dokument ali kamen koji je izložen u muzeju je; Antilopu koja trči divljinom na Afričkim ravnicama ne bi trebali smatrati dokumentom. Međutim, ako se antilopa zatoči i prebaci u zoološki vrt kao predmet istraživanja, ona pretvorena u dokument. Svaki istraživački rad ili članak napisan o toj antilopi onda je sekundarni dokument, a sama antilipa primarni dokument*.“[2]

U prvom poglavlju su opisani sustavi za upravljanje dokumentima općenito i sustavi za upravljanje radnim tokom. U drugom poglavlju opisano je okruženje u kojem se implementirao sustav. U trećem poglavlju prikazana je implementacija jednostavnog sustava administrativnog tipa koji primjenjuje model radnog toka. Sustav koristi tehnologije poput Google Apps Script okruženja, MSSQL baze podataka te JavaScript, HTML i CSS, da bi omogućio interaktivno korisničko sučelje i automatizaciju procesa prijavljivanja i rješavanja smetnji.

1.1. Sustav za upravljanje dokumentima

Sustav za upravljanje dokumentima je sustav koji se koristi za praćenje, upravljanje i pohranu dokumenata. Većina ovakvih sustava ima i sposobnost praćenja različitih verzija dokumenata. Termin sustav za upravljanje dokumentima ima neka preklapanja sa konceptima sustava za upravljanje sadržajem. Često se na ovakav sustav gleda kao na komponentu sustava za upravljanje poslovnim sadržajem.[3]

1.1.1. Povijest

Početkom 80-ih godina prošlog stoljeća, određeni broj proizvođača je počeo razvijati softverske sustave za upravaljanje papirnatim dokumentima. Takvi sustavi su, osim što su upravljali papirnatim dokumentima, upravljali i štampom, fotografijama. Kasnije su razvojni timovi počeli stvarati drugačije tipove sustava, koji su imali mogućnost upravljanja elektroničkim dokumentima. Ti prvi sustavi za upravljanje elektroničkim dokumentima su upravljali sa vlastitim tipovima dokumenata ili sa ograničenim brojem tipova podataka. Mnogi ovakvi sustavi su kasnije postali poznati kao sutavi za dokumente u slici (*eng. Document imaging systems*) jer su bili fokusirani prema snimanju (skeniranju), pohrani i dohvaćanju slikovnih formata datoteka. Sustavi za upravljanje elektroničkim dokumentima su se razvili do te mjere da su mogli upravljati sa bilo kojim formatom koji može biti pohranjen na lokalnoj mreži. Obuhvaćali su elektroničke dokumente, kolaboracijske alate, sigurnost, radni tok i praćenje. Ovakvi sustavi omogućavali su spremanje formi, fakseva te kopija dokumenata u slikovnom formatu u repozitorij te brzo dohvaćanje istih. Iako mnogi sustavi za upravljanje elektroničkim dokumentima spremaju datoteke u njihvom prirodnom obliku (.doc, .xls, .ppt, .pdf), puno sustava koji se zasnivaju na web tehnologiji (*eng. Web-based*) spremaju dokumente u HTML formatu.[3]

1.1.2. Komponente sustava za upravljanje dokumentima

Sustavi za upravljanje dokumentima pružaju pohranu, verzioniranje, meta podatke, sigurnost, indeksiranje i dohvaćanje te vrlo važnu komponentu radni tok (*eng. Workflow*) dokumenata.

- Meta podatci sadržavaju, na primjer, datum kada je dokument pohranjen i identitet korisnika koji ga je pohranio.
- Sigurnost dokumenata je najvažniji dio u mnogim sustavima za upravljanje dokumentima, posebno kada dokumenti sadržavaju osobne podatke. U nekim sustavima administrator može dozvoliti samo određenim osobama pristup određenim dokumentima. Neki dokumenti se mogu i posebno označiti tako da ih je nemoguće neovlašteno koristiti.
- Indeksiranjem se prate dokumenti da bi se kasnije mogli laše dohvaćati. Može biti jednostavno kao postavljanje jednistvenog identifikatora na svaki dokument, ali i kompliciranije kao uzimanje određenih riječi iz samog dokumenta i korištenje meta podataka kod indeksiranja.
- Dohvaćanje dokumenata iz pohrane također može biti ostvareno na jednostavan način ali može biti i jako komplicirano. Možemo imati jedan indeks preko kojega dolazimo do dokumenta ili više opisnih pojmova kojih definiraju korisnici sustava.[3]

1.2. Radni tok

Koalicija za Upravljanje Radnim tokom (*eng. Workflow Management Coalition*) je konzorcij formiran zbog definicije standarda za interoperabilnost sustava za upravljanje radnim tokom i ona se bavi problemima vezanim za upravljanje radnim tokom. Koalicija je osnovana 1993. i njeni originalni članovi su bili IBM, HP, Fujitsu i još otprilike tristo softverskih tvrtki u poslovnom sektoru. 1994. koalicija je sastavila i izdala *Workflow Reference Model* koji specificira probleme upravljanja radnim tokom. [4]

Radni tok je računalno rješenje za automatizaciju poslovnog procesa. Dotiče se automatizacije poslovnih procesa, odnosno procedura u kojima se proslijeđuju dokumenti, informacije ili zadatci između sudionika, po definiranom nizu pravila za postizanje ili doprinos nekom poslovnom cilju. Iako radni tok može biti organiziran ručno, u praksi je on organiziran u sklopu informatičkog sustava zbog računalne podrške za automatizaciju procedura. Radni tok se često povezuje sa redizajnom poslovnih procesa koji se bavi procjenom, analizom, modeliranjem, definicijom i implementacijom glavnih poslovnih procesa u nekoj organizaciji. Iako sve aktivnosti redizajna poslovnog procesa nisu implementirane radnim tokom, on se često koristi kao rješenje jer razdvaja logiku poslovnog

procesa od njegove informatičke podrške. Tako omogućuje uvođenje naknadnih promjena u proceduralna pravila koja definiraju poslovni proces.

1.2.1. Sustav za upravljanje radnim tokom

Sustav za upravljanje radnim tokom je sustav koji definira radne tokove ,upravlja s njima i izvodi ih kroz izvršavanje softvera po poretku koji je određen sa neokm računalnom reprezentacijom logike radnog toka.

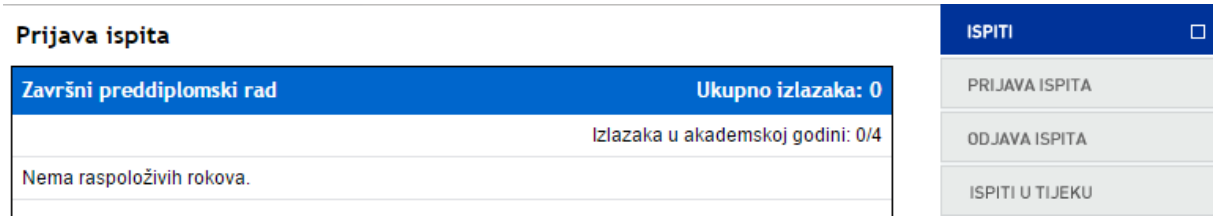
1.2.2. Tipovi sustava za upravljanje radnim tokom

Široko prihvaćena taksonomija razlikuje nekoliko tipova sustava:

- Administrativni
- Ad Hoc
- Kolaborativni
- Produkcijski

Osnovni parametri za ovakvu klasifikaciju su sličnosti između uključenih poslovnih modela i njihove vrijednosti za poduzeća uz koje su vezani.

Administrativni radni tokovi se odnose na birokratske procese u kojima su poznati koraci koje radni tok treba slijediti i u kojima postoje dobro utemeljena pravila koja znaju svi uključeni u sustav. Primjeri takvih radnih tokova su upisivanje na fakultet, prijavljivanje završnog rada, registracija vozila i gotovo bilo koji drugi proces u kojem je potrebno ispunjavati određenu formu i slijediti određene korake. Ovakvi radni tokovi su potaknuli ideju procesiranja formi. Primjer aplikacije koji koristi ovakav radni tok bio bi Studomat(Slika 1.1).

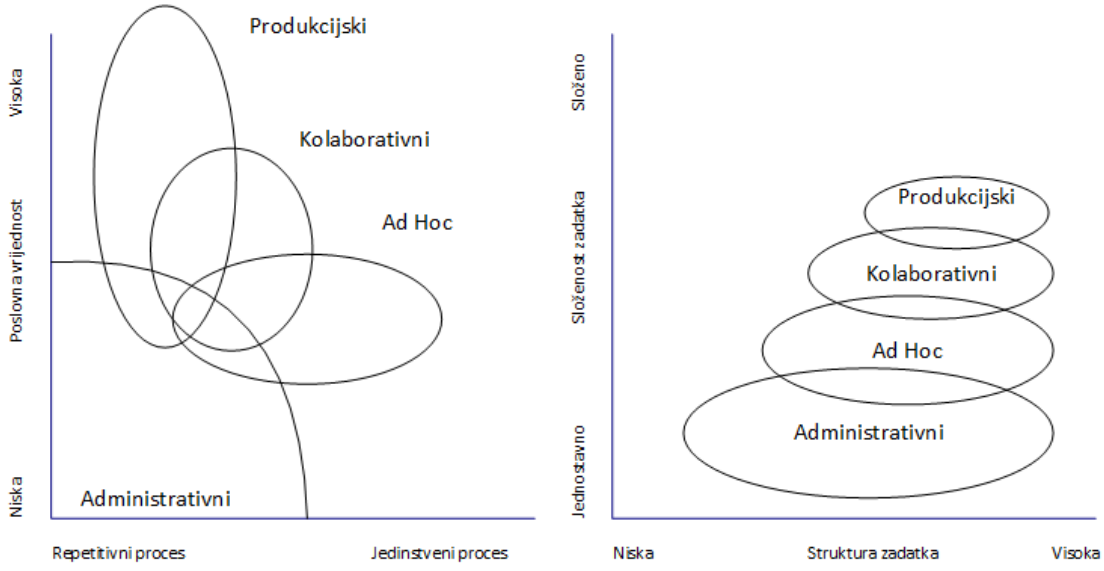


Slika 1.1 Korisničko sučelje aplikacije Studomat

Ad Hoc radni tokovi su slični administrativnim tokovima ali za razliku od njih, kreirani su da se nose sa iznimnim situacijama ili za specifične situacije. Administrativne

radne tokove od ad hoc radnih tokova razlikuje perspektiva onoga koji koristi sustav. Upis na fakultet je sa strane sveučilišta administrativni proces, dok je iz perspektive studenata to nešto što se događa samo jednom. Iz perspektive studenta to je specifična situacija (u većini slučajeva) i tako ad hoc s tog gledišta. Isto tako može postojati slučaj koji nije jedinstven ali je jedinstvena svaka instanca tog slučaja. Kad korisnik mora biti uključen u mnogo različitih procesa dobro dođu ovakvi radni tokovi. Razlog za korištenje ad hoc radnog toka nije problem praćenja svakog od tih procesa posebno već problem praćenja svih procesa odjednom.

Kolaborativni sustavi su uglavnom okarakterizirani brojem uključenih sudionika i interakcija među njima. Za razliku od drugih radnih tokova koji su temeljeni na pretpostavci da uvijek postoji korak naprijed, kolaborativni radni tok može uključivati nekoliko iteracija istog koraka dok se ne postigne dogovor između svih sudionika ili čak vraćanje na prethodni korak ako se dogovor ne uspije postići. Bilo bi gotovo nemoguće moedelirati takav proces korištenjem alata koji nisu za namjenjeni za kolaboraciju jer je praktički nemoguće predodrediti sljedeći korak. Ovakvi radni tokovi su dinamčni jer se zapravo definiraju kako napreduju kroz korake.



Slika 1.2 Gruba klasifikacija sustava za upravljanje radnim tokom

U ekstremnim slučajevima upitno je spadaju li ovakvi sustavi u sustave radnih tokova jer većinu koordnacije odrađuju ljudi, a sustav ima ulogu da im pruži dobro sučelje za interakciju.

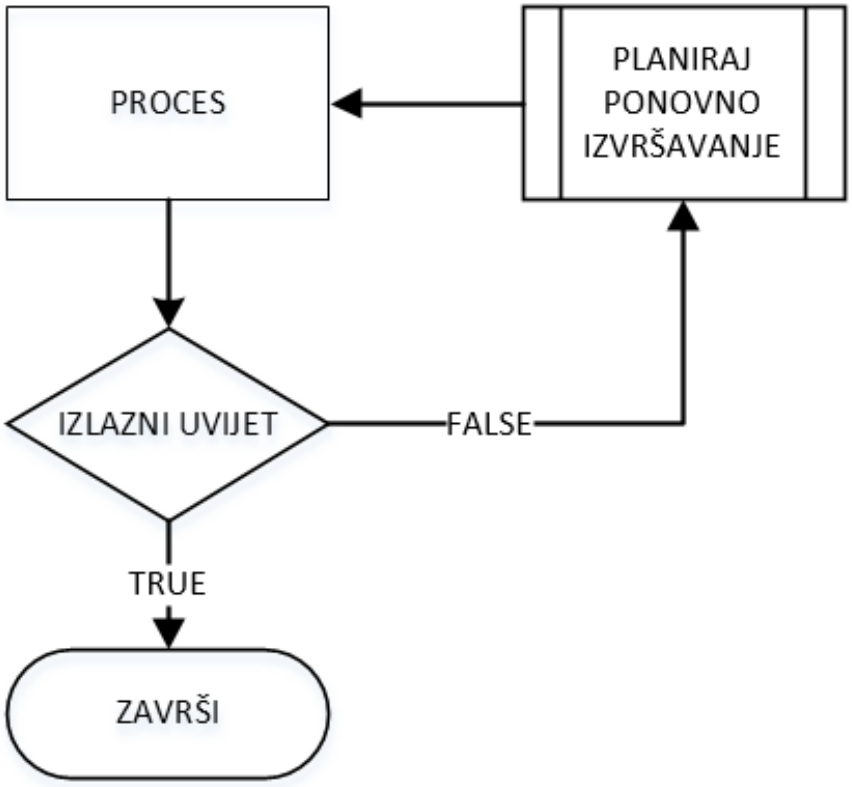
Produkcijski sustavi radnog toka su na višem stupnju od svih ostalih vrsta sustava. Oni su implementacija kritičnih poslovnih procesa. To su poslovni procesi koji su direktno vezani za neku funkciju organizacije. Zahtjevi za kredite i zajmove te potraživanja od osiguranja su tipični primjeri za ovaj tip radnog toka, ali trebamo napomenuti da je razlika između administrativnih radnih tokova i produkcijskih nekad samo stvar perspektive. Najčešće kad govorimo o produkcijskim radnim tokovima naistaknutije točke su veliki razmjeri, kompleksnost i različitost okoline u kojoj se izvršavaju, raznolikost uključenih organizacija i ljudi te priroda zadataka. Konkretno, produkcijski radni tokovi se izvršavaju preko više različitih sustava i jako je važno imati alate za nadgledanje koji dozvoljavaju statističku analizu izvršavanja tih procesa.[5]

1.2.3. Model radnog toka

Ovaj model se temelji modelu IBM-ovog sustava za upravljanje radnim tokom imena *FlowMark*. To je aciklički usmjererni graf, čiji vrhovi predstavljaju korake izvršavanja, a bridovi protok kontrole i podataka između različitih koraka. Ispod su opisane komponente tog modela:

- Proces, opis slijeda koraka koji se trebaju izvršiti da bi se postigao neki cilj. Trebao bi imati ime, verziju, početna i završna stanja i dodatne podatke za sigurnost, reviziju i kontrolu. Proces se sastoji od *aktivnosti* i relevantnih podataka.
- Aktivnost, odnosno korak u procesu ima ime, tip, stanje prije i nakon izvršavanja te ograničenja planirana. Aktivnosti mogu biti *programske aktivnosti* ili *procesne aktivnost*. Programska aktivnost ima svoj program koji se izvršava kad se izvrši aktivnost. Aktivnost se izvršava tako da se dodjeljuje korisnicima koji je mogu izvršiti i svaki korisnik ima *radnu listu* aktivnosti koje treba izvršiti. Procesna aktivnost ima drugi proces vezan na sebe, tako da se cijeli proces izvršava kad se izvrši aktivnost. Svaka aktivnost ima ulazni i izlazni spremnik podatka
- Protok kontrole je redoslijed po kojem se aktivnosti izvršavaju. Specificiran je *kontrolnim konektorima* između aktivnosti.

- Ulazni spremnik je slijed varijabli i struktura koje se koriste kao ulaz za pozvanu funkciju.
- Izlazni spremnik je slijed varijabli i struktura u koje se spremaju izlazne vrijednsoti pozvane funkcije
- Protok podataka je specificiran kroz *konektore podataka* između aktivnosti i određuje serije mapiranja između izlaznih i ulaznih spremnika podataka tako omogućavajući aktivnostima da razmijenjuju informacije.
- Stanja specificiraju okolnosti pod kojima će se odvijati određeni događaji. Imamo tri osnovna tipa stanja. *Tranzicijska stanja* su povezana sa kontrolnim konektorima i određuju da li se konektor ocjenjuje kao istinit (*eng. True)* ili lažan (*eng. False*). Kontrolni konektor kojem je vrijednost *false* neće izazvati izvršavanje aktivnosti na njegovom kraju. *Početni uvjeti* određuju kada će aktivnost započeti; na primjer, kad su svi dolazni konektori *true* (uvjet I) ili kad je samo jedan od konektora *true* (uvjet ILI). *Izlazni uvijeti* određuju kada se aktivnost smatra završenom. Nakon izvršavanja aktivnosti provjerava se izlazni uvjet. Ako je uvjet *true* aktivnost se završi, ako je *false*, aktivnost se ponovno planira za izvršavanje.[5]

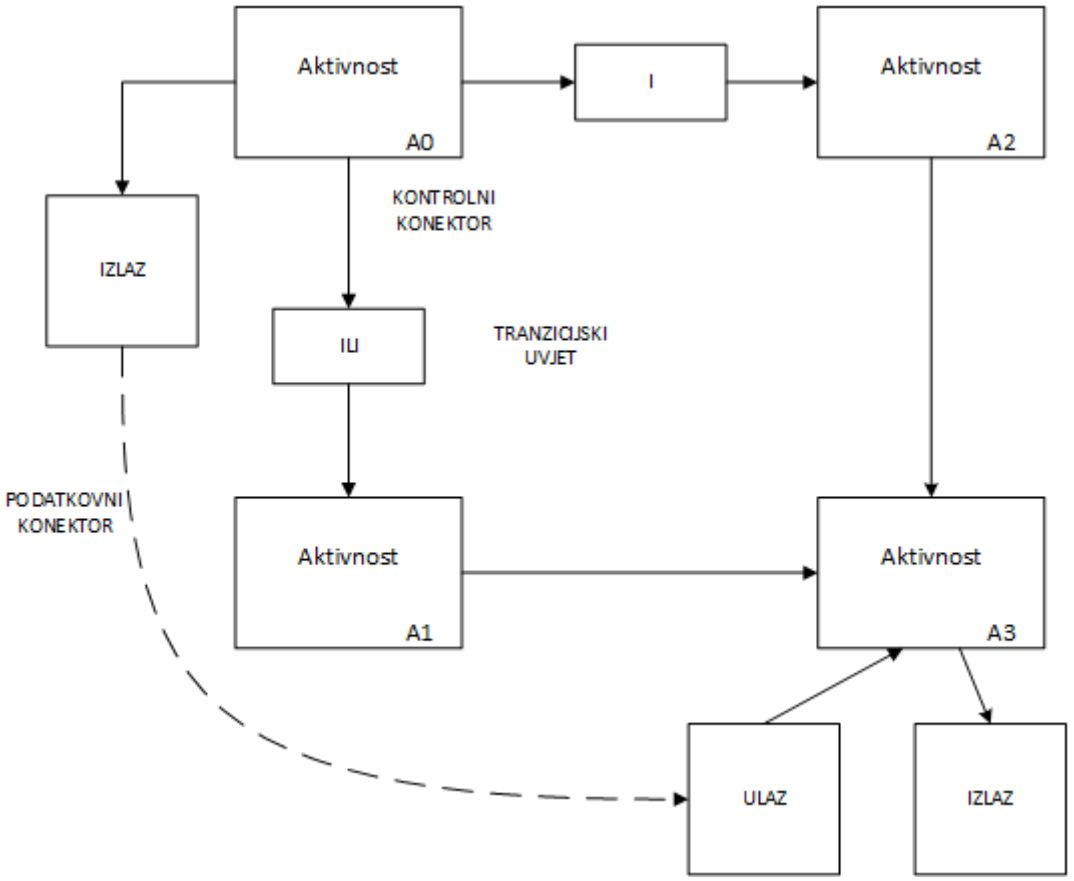


Slika 1.3 Provjera izlaznih uvjeta

Aktivnosti mogu biti u jednom od sljedećih stanja:

- Spremno (*eng. Ready)*, stanje prije nego što započne izvršavanje aktivnosti.
- U radu (*eng. Running*), stanje u kojem se aktivnost izvršava.
- Završeno (*eng. Finished*), stanje nakon što se aktivnost izvrši.
- Terminirano (*eng. Terminated*), stanje nakon što je izlazni uvjet zadovoljen.

Aktivnosti se mogu započeti ručno ili automatski. Unutar procesa, one aktivnosti koje su bez dolaznih kontrolnih konektora nazivamo *početne aktivnosti* procesa i te aktivnosti su postavljene u spremno stanje kad proces započne. Kada se aktivnost završi provjerava se izlazni uvjet i ako nije zadovoljen ona se opet postavlja u spremno stanje (Slika 1.3). Inače, stanje se postavi na terminirano i svi odlazni kontrolni konektori te aktivnosti se provjeravaju.



Slika 1.4 Glavne komponente FlowMark-ovog modela za kontrolu u protok podataka

Kada je početni uvjet aktivnosti zadovoljen, aktivnost se postavlja u spremno stanje. Ako početni uvjet nije zadovoljen i aktivnost se nikada neće izvršavati, postavlja se u terminirano stanje. Ovakva procedura se naziva eliminacija mrtvog puta (*eng. Dead path elimination*). Proces se smatra završenim kada su terminirane sve njegove aktivnosti.

Ključni aspekti sustava za upravljanje radnim tokom su razni uvjeti povezani sa konektorima i aktivnostima jer oni su osnmova za planiranje i raspoređivanje aktivnost. Logika poslovnih procesa je ugrađena u njih. Ove uvjete možemo podjeliti na tri različita tipa informacija:

- Aplikacijski podaci, koji pružaju ulaz vezan za aplikaciju i omogućavaju opis protoka kontrole u smislu posla kojeg ta aplikacija odrađuje. Tipičan primjer bio bi sa podacima koje dohvaćamo korištenjem SQL upita *WHERE Prijave.Status = "riješena" AND Prijave.Korisnik_Id = 1*.
- Podaci izvršavanja, pružaju informacije o tome jesu li aktivnosti uspješno izvršene. Ovo je drugačiji tip podataka od aplikacijski i u većini slučajeva vraća kodove o uspješnosti izvršavanja aktivnosti. To bi na primjeru izvršavanja SQL izjave, korištenjem INSERT naredbe, bio podatak da li je izjava utjecala na ijedan red u tablici ili podatak o broju redaka na koje je ta naredba utjecala. U implementaciji Helpdesk sustava koja je opisana u trećem poglavlju ovog rada koristi se HTML kôd *200 OK* kada se neki podaci uspješno dohvate. Isto tako puno puta se može vidjeti *404 Not found* kada web stranica kojoj se pokušava pristupiti nije dostupna ili ne postoji. Takvu vrstu kôdove vraćaju podaci izvršavanja.
- Vanjski događaji, koji omogućuju sinkronizaciju izvršavanja procesa radnog toka sa pojavama u vanjskom svijetu kao što je to dolazak e-pošte, određeno vrijeme ili datum i slično. U Google Apps Script okruženju koji se koristi u implementaciji, u ovom radu, ovakvo nešto se može izvesti korištenjem okidača (*eng. Triggers*). U implementaciji koja je opisana u ovom radu okidači nisu korišteni jer Google ograničava broj njihovih korištenja po danu. Primjer korištenja okidača u Google Apps Script okruženju možemo vidjeti u aplikaciji *Putni nalog* koju je razvila prof. Divna Krpan. Aplikacija koristi okidače u određenim stadijima odobravanja zahtjeva za financiranje puta. Ti okidači se mogu iskoristiti i za raspodijelu samog sustava u slučaju kad više korisnika istovremeno zahtjevaju iste resurse. [5]

Ova tri tipa ulaznih informacija se obrađuju na različit način jer potječu iz različitih izvora. Iako je moguće kombinirati ih u jednom uvjetu, u praksi svaki od ova tri ulaza je puno korisniji sa zasebnim tipom uvjeta. Aplikacijski podaci se u većini slučajeva koriste za

određivanje kojim putem krenuti i koje aktivnosti se moraju poduzeti, podaci izvršavanja isto određuju kojim putem krenuti ali u slučaju kad se na primjer desi greška, te određuju kada su aktivnosti uspješno završene. Vanjski događaji najčešće okidačima pokreću određeni proces ili aktivnost.

Stanja mogu biti neprocijenjena, djelomično procijenjena i procijenjena. Ovisno o tipu informacija koje koriste, ova stanja mogu biti privremena ili stalna. Stanja temeljena na vanjskim događajima su promijenjiva odnosno dinamička su. Stanja temeljena na aplikacijskim podacima i podacima izvršavanja mogu se jedino mijenjati iz neprocijenjenih u djelomično procijenjena i procijenjena stanja, odnoso takva stanja su statična. Kad jednom dođu do procjenjenog stanja, procjena se više ne mijenja ili su *true* ili *false*. Kao rezultat toga, lakše je nositi se sa aplikacijskim podacima i podacima izvršavanja, ne samo iz aspekta dizajna procesa radnog toka, nego i sa strane implementacije radnog toka u stvarni sustav. Vanjski događaji, mogu zahtjevati uvođenje određenog vremenskog rasuđivanja u sustav kao i sposobnosti da se sustav nosi sa promijenjivim stanjima. U ovakvim slučajevima, semantiku stanja je teško definirati, jer se aktivnost može izvršavati i kad su stanja koja su potaknula njeno izvršavanje *false*. Međutim, vanjski događaji mogu biti ključni kod sinkornizacije radnog toka, jer stanja uključena u vanjski događaj mogu biti korištena kao točke sinkornizacije.

1.2.4. Arhitektura

Sustavi za upravljanje radnim tokom pružaju podršku u tri funkcijska područja: Dužina izgradnje (*eng. Buildtime*), dužina trajanja (*eng. Runtime*) i interakcije dužine trajanja (*eng. Runtime interactions*). *Buildtime* funkcije podržavaju definiciju i modeliranje procesa radnog toka. Funkcije *runtime* kontrole rukuju izvršavanjem procesa. *Runtime* interakcije pružaju sučelja sa korisnicima i aplikacijama. *Buildtime* i *runtime* kontrole su u većini sustava centralizirane. *Buildtime* je centralizirana zato jer je dostupna malom skupu dizajnera radnog toka, a *runtime* jer je zajednička svim korisnicima i često ima velike zahtjeve u smislu kapaciteta pohrane.

Runtime kontrola ima dva aspekta:

- Trajna pohrana (*eng. Persistent storage*)
- Navigacija procesa

Trajna pohrana omogućuje oporavak sustava u slučaju grešaka bez gubljenja podataka. Isto tako pruža sredstva za održavanje i reviziju izvršavanja procesa. Navigacijska logika kontrolira izvršavanje procesa. Tako, unutar *runtime* kontrole imamo dvije komponente:

- Navigacijski poslužitelj
- Poslužitelj za pohranu

Slično tome postoje dva tipa *runtime* interkcija. Interakcije sa korisnicima, koje su zapravo radne liste dodijeljene korisnicima i interakcije sa pozvanim aplikacijama, koje su sučelje za aplikacije koje se izvršavaju kao dio radnog toka. Smatramo ih različitim komponentama:

- Korisničko sučelje (*eng. User interface*)
- Aplikacijsko sučelje (*eng. Application interface*)[5]

2. Google Apps Script

Google Apps Script je programski jezik temeljen na programskom jeziku JavaScript verziji *1.6* sa dijelovima iz verzija *1.7* i *1.8* nastao 2009. godine. Najbitnija razlika između JavaScript i jezika Apps Script je ta što se JavaScript izvršava na strani klijenta dok se Apps Script izvršava na Google Cloud usluzi, odnosno na strani poslužitelja. Google Apps Script se koristi, kako i sam naziv nalaže, za izradu aplikacija vezanih za Google Usluge. Omogućuje komunikaciju sa uslugama poput AdSense, Analytics, Kalendar, Disk, Gmail i Mape. Može se koristiti za objavljivanje samostalne web stranice ili aplikacije ili njenog ugrađivanja u već postojeću Google Site stranicu. Također mogu se stvarati i dodaci za Google Dokumente, Google Tablice i Google Forme. U ovom radu Apps Script se koristi kod izrade mrežne aplikacije *Helpdesk*. Prednosti ovog jezika su što nije težak za učenje, pogotovo ako oni koji ga koriste već imaju iskustva sa jezikom JavaScript. Za razvoj aplikacija u ovom jeziku nije potrebno instaliranje nikakve dodatne programske podrške kao što je Visual Studio kod razvoja ASP.NET aplikacija. Sva potrebna programska podrška je na Google Disku i jedino što je potrebno za razvoj je web preglednik. Aplikacije koje se objavljuju u Google Apps Script okruženju su odmah "žive" bez potrebe za odradom dodatnih koraka za njihovo postavljanje na web.[6]

2.1. Sučelje

Sučelje se pokreće preko Google Disk usluge. Kada se želi stvoriti novi projekt, pritiskom na desnu tipku miša bilo gdje na radnoj površini Google Diska otvara se izbornik. Odabirom opcije *Nova datoteka* otvara se još jedan izbornik. Na tom izborniku je lista svih vrsta datoteka koje se mogu stvoriti na Google Disku. Osim izlistanih aplikacija, postoji i mogućnost za povezivanje dodatnih aplikacija što omogućuje stvaranje dodatnih vrsta

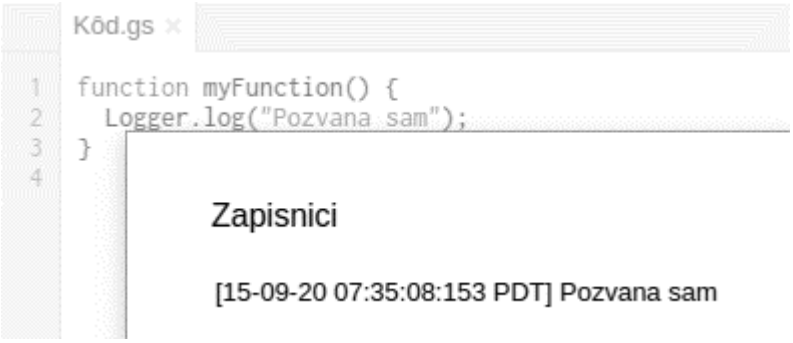
datoteka. Klikom na Skripta za Google Apps (Google Apps Script) otvara se novi projekt. Prije samog početka izrade aplikacije potrebno je odabrati jedan od predložaka za stvaranje projekta kao što je "Web-aplikacija" koji u projekt automatski umetne sve potrebne datoteke za izradu web aplikacije. Može se odabrati i prazan projekt pa posebno dodavati potrebne datoteke u projekt. Nakon što se projektu da ime može se započeti s radom.

Uređivač se sastoji od dijela za datoteke, uređivanje, prikaz, pokretanje, objavu ,resurse i pomoć.

U Apps Script projektu moguće je stvoriti samo dvije vrste datoteka, Script datoteku sa nastavkom .gs te običnu HTML datoteku. U Script datotekama je kôd koji se izvršava na poslužitelju i piše se , naravno, u Apps Script jeziku. Sve što se izvršava preko HTML servisa, uključujući i JavaScript i CSS, se piše u HTML datotekama. Ako se zbog preglednost poželi pisati neki JavaScript kôd odvojeno, u nekoj drugoj datoteci, može se stvoriti nova HTML datoteku u kojoj će se JavaScript kôd pisati između <*script*> HTML oznaka. Ta datoteka se onda uključuje u glavnu HTML datoteku u kojoj je referenicrana. Isto je i sa CSS dijelom samo se CSS zapisuje između <*style*> HTML oznaka.

2.1.1. Zapisnik (eng. *Logger*)

Jedno vrlo korisno svojstvo Apps Script uređivača je *Zapisnik*. Omogućuje da u bilo kojem dijelu kôda zapiše neku vrijednost.



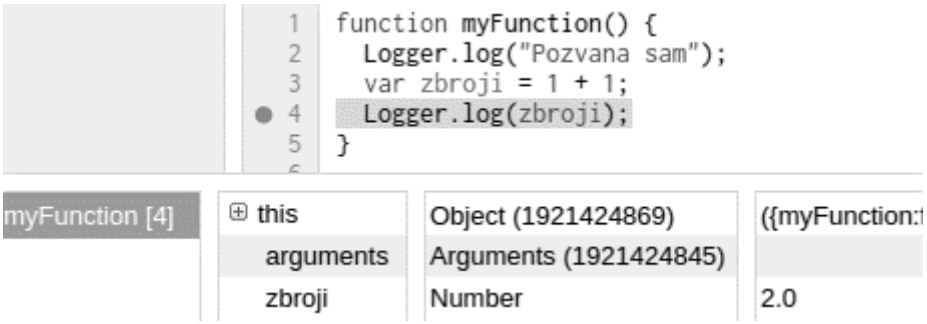
Slika 2.1 Logger

U izradu *Helpdesk* aplikacije najviše se koristio kod provjere da li funkcija prima prave parametre kad se poziva i da li je uopće pozvana. Vrijednosti koje su u zapisniku se brišu kad

se ponovno pokrene funkcija u kojoj je pozvano upisivanje. Jednostavnim ubacivanjem kôda *Logger.log("Pozvana sam!");* u neku funkciju radi se pozivanje zapisnika kod pokretanja te funkcije.

2.1.2. Otklanjanje grešaka (eng. *Debugger*)

Otklanjanje grešaka je omogućeno samo za Apps Script datoteke. Za otklanjanje grešaka potrebno je odabrati funkciju koja se želi analizirati, odabrati mjesto u kodu na kojem se traže vrijednosti i pritiskom na ikonu za otklanjanje grešaka. Tu se može analizirati svaka linija kôda i otkriti gdje se nalazi greška.



Slika 2.2 Debugger

Može se vidjeti vrijednosti pojedinih varijabla u određenom trenutku i osim lakšeg pronalaženja grešaka može se i bolje shvatiti kako funkcionira pojedina funkcija.

2.2. Baza podataka

Što se tiče baze podatka kod Apps Script-a postoji nekoliko mogućnosti. Puno korisnika se odlučuje za korištenje Google Tablica kao bazu podatka. Apps Script preko SpreadsheetApp servisa pristupa korisnikovim Google Tablicama i u njih može upisivati podatke. Osim Google Tablica može se koristiti i JDBC (Java Database Connectivity).

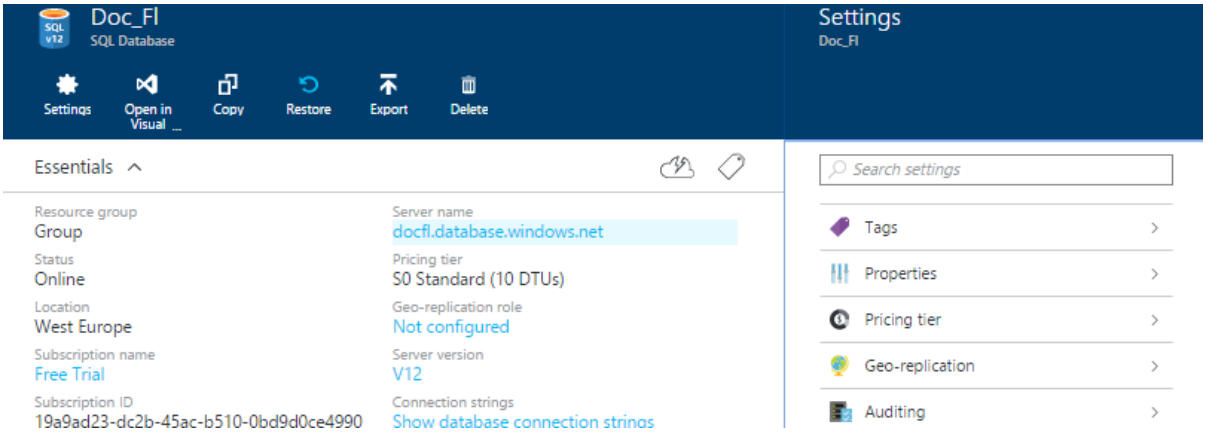
2.2.1. JDBC Service (Java Database Connectivity)

JDBC je tehnologija baze podataka u programskom jeziku Java. Omogućuje pristup bazi podataka u Javi. Google je ovu tehnologiju implementirao u Apps Scirpt kao JDBC

service i uz pomoć te tehnologije može se pristupiti Google Cloud Sql, MySql, Microsoft SQL te Oracle bazama podataka. U ovom projektu je odabrana JDBC tehnologija. Preko connection string-a se spaja na Microsoft SQL bazu podataka koja je postavljena na Microsoft Azure serveru. [7]

2.2.2. Microsoft Azure

Microsoft Azure je platforma za računalstvo u oblaku. Koristi se za stvaranje, objavu i održavanje aplikacija i servisa u oblaku na globalnoj Microsoftovoj mreži. Nastala je 2008. godine pod nazivom Windows Azure, a 2014. je preimenovana u Microsoft Azure. Pruža Paas (Platforma kao servis) i Iaas (infrastruktura kao servis) usluge u oblaku, podržava mnogo različitih programskih jezika, alata i okruženja. Ono što je bitno za ovaj projekt ,u vezi Azure platforme je da, između ostalih baza podataka, podržava i Microsoft SQL bazu podataka te omogućuje spajanje na takvu bazu preko Apps Scripta. Osim baze na Azure platformi, korištena je i dodatna baza na free hosting servisu Smarter.asp u slučaju da ova na Azure bude nedostupna. Još jedna prednost Azure platforme, a koja je bila bila korisna u ovo je implementaciji, je što podržava upravljanje bazom iz okruženja Visual Studio korištenjem SSDT (Sql Server Data Tools) . [8]



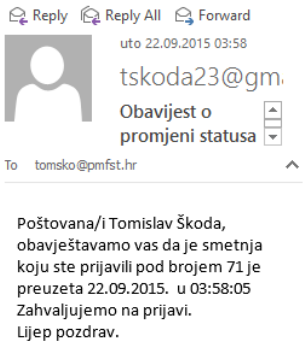
Slika 2.3 Microsoft Azure Portal

3. Implemetacija *Helpdesk* aplikacije u Google Apps Script-u

Google Apps Script okruženje je odabrano u ovom radu iz nekoliko razloga. Prvenstveno jer je aplikacija namjenjena za korištenje u sklopu već postojećeg sustava koji koristi isto okruženje. Isto tako Google Apps Script je zanimljivo okruženje zbog raznih tehnologija koje koristi.

3.1. Opis sustava

Helpdesk aplikacija riješava problem prikupljanja podataka o neispravnoj opremi koja se koristi u učionicama na Prirodoslovno-Matematičkom Fakultetu u Splitu. Sustav je namjenjen profesorima i asistentima koji održavaju nastavu u tim učionicama (njih u ostatku teksta nazivamo korisnicima) te onima koji taj sustav održavaju (njih nazivamo tehničarima). Korisnicima omogućuje lakše i brže prijavljivanje kvarova, a tehničarima olakšava praćenje svih smetnji koje su prisutne na području kojeg održavaju. Naravno, aplikacija nije ograničena za primjenu samo na ovoj ustanovi. Moguće ju je prilagoditi bilo kojem sustavu koji ima iste potrebe. Osim mogućnosti prijavljivanja smetnje, sustav korisnicima šalje obavijesti o statusu smetnje korištenjem MailApp servisa. Korisnik može saznati u kakvom je stanju smetnja preko obavijesti koje dolaze na e-mail adresu (Slika 3.1) ili na samom sučelju aplikacije. Dokument kojim upravlja ovaj sustav je sama smetnja koju korisnik prijavljuje.

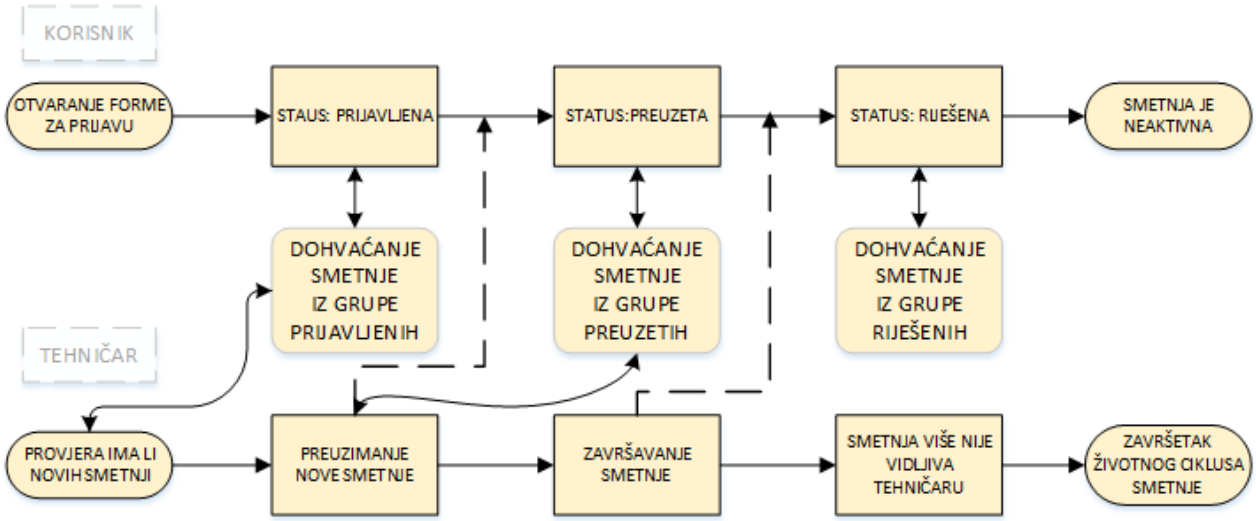


Slika 3.1 Obavijest o statusu smetnje

3.2. Radni tok smetnje

Radni tok smetnje ukratko bi mogli opisati kao put smetnje od prijave do završetka.

- Sa korisničke strane
    1. Otvaranje forme za prijavu
    2. Ispunjavanje podataka o smetnji
    3. Slanje smetnje
- Sa strane tehničara
    1. Provjera ima li novih smetnji
    2. Preuzimanje nove smetnje
    3. Završavanje smetnje



Slika 3.2 Radni tok smetnje

Postoje tri grupe smetnji koje se mijenjaju ovisno promjeni stanja smetnje:

- Grupa prijavljenih smetnji
- Grupa preuzetih smetnji
- Grupa riješenih smetnji

Nakon što korisnik prijavom stvori formu smetnje ona je u grupi prijavljenih smetnji. Ako je tehničar preuzme on će je dohvatiti iz grupe prijavljenih i ona će preći u grupu preuzetih. Korisnik može vidjeti sve svoje prijavljene i preuzete smetnje u tablici aktivnih smetnji. Nakon što tehničar završi smetnju ona prelazi u grupu riješenih smetnji. Tada je dohvatiti može samo korisnik u tablici neaktivnih smetnji. Radni tok koji koristi ova aplikacija je

administrativog tipa. Koristi formu za prijavu smetnje i ima točno određene korake koji se izvršavaju se dok se smetnja ne riješi. Radni tok aplikacije je određen pravilima (*eng. Rules-based*) koji upravljaju usmjeravanjem smetnje od korisnika prema tehničaru te načinima kojima korisnik dohvaća smetnje. Tok smetnje je linearan i svaki put kad se završi određeni proces krene se odvijati novi korak. Također radni tok koristi i grananje. Ovisno tko pristupa dokumentu prikazuju se različite mogućnosti. Tako na primjer ako korisnik pristupi detaljima smetnje on neće moći utjecati na taj dokument nego ga samo dohvatiti, dok će tehničar s druge strane moći završiti tu smetnju i promijenom njenog statusa u *rijesena* prebaciti je u grupu riješenih smetnji.

3.3. Funkcionalnosti aplikacije

Aplikacija je dizajnirana tako da joj se može pristupiti na dva načina:

- Kao korisnik
- Kao tehničar

Ako aplikaciju pokreće korisnik prikazat će se sučelje za korisnika, u suprotnom prikazat će se sučelje za tehničara. Aplikacija prima podatke o korisniku preko query stringa koji je uključen u samom URL-u aplikacije kad korisnik klikne na njega.

```
?idk=2&ime=Tomislav&pre=Škoda&email=tomsko@pmfst.hr&uloga=korisnik
```

Kôd 3.1 Query string kod pristupa stranici

Kad aplikacija primi ulogu *korisnik* kao u kodu (Kôd 3.1) odlučit će koja će HTML stranica biti prikazana. Osim uloge *korisnik* u query stringu može biti i uloga *tehnicar*. U kôdu ispod vidimo kako aplikacija, ovisno o ulozi onog koji pristupa aplikaciji, odlučuje što će se prikazati.

```
  if(uloga != 'tehnicar'){
    loginKorisnika();
    return HtmlService.createTemplateFromFile("Index").evaluate()
    .setTitle("Prijava smetnje")
    .setSandboxMode(HtmlService.SandboxMode.IFRAME);
  }
  loginKorisnika();
  return HtmlService.createTemplateFromFile("Popravci").evaluate()
  .setTitle("Rješavanje smetnji")
  .setSandboxMode(HtmlService.SandboxMode.IFRAME);
}
```

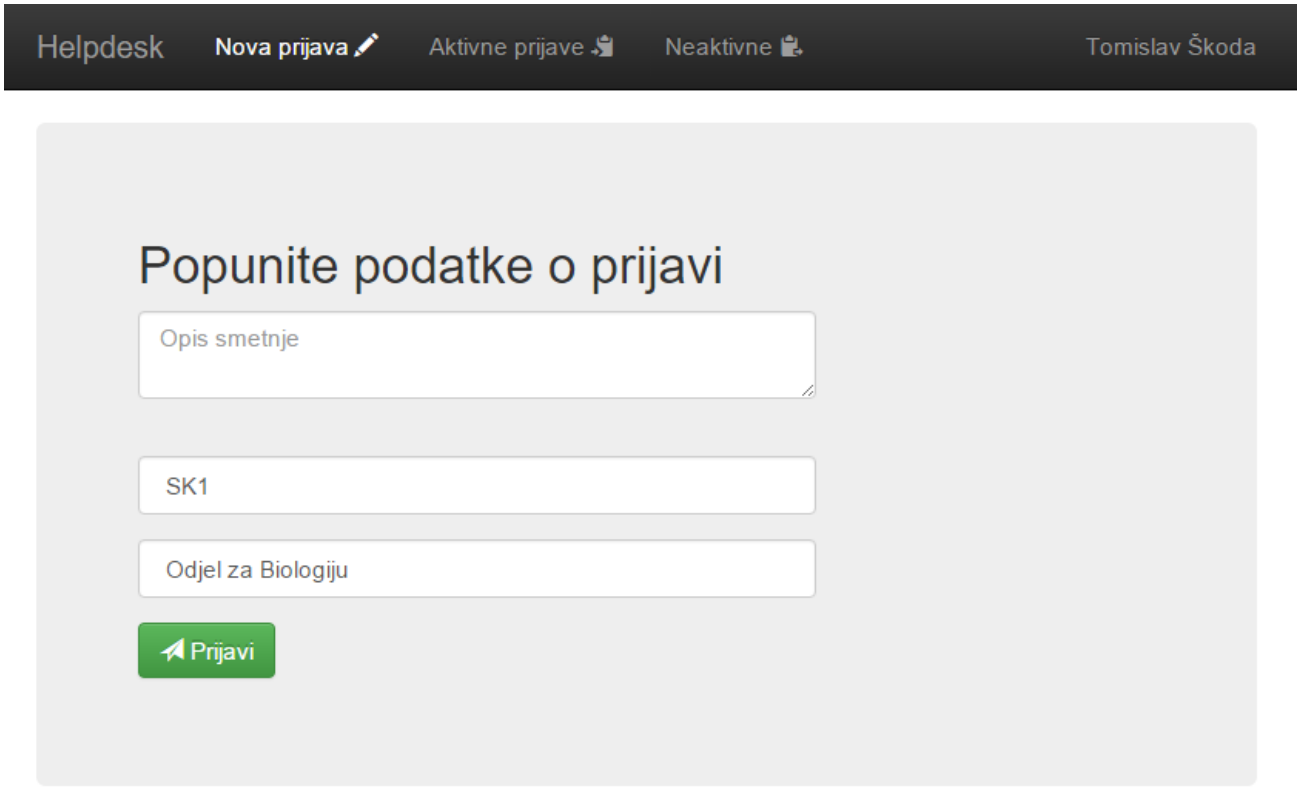
Kôd 3.2 Odluka o prikazu stranice ovisno o ulozi

3.3.1. Sučelje za korisnika

Sučelje za korisnika se sastoji od dva glavna dijela:

- Glavnog izbornika sa stavkama:
  - *Nova prijava*
  - *Aktivne prijave*
  - *Neaktivne prijave*
- Dijela za sadržaj u kojem se prikazuje sadržaj ovisno o odabranoj opciji na izborniku

Kao što vidimo na slici (Slika 3.3) kod stvaranja nove prijave korisnik ima mogućnost da odabere odgovorni odjel i učionicu u kojoj je kvar prisutan. Podaci o učionici i odjelu se dohvaćaju iz MSSQL baze podataka kad se aplikacija učitava. Također smetnju može detaljno opisati u dijelu za opis.

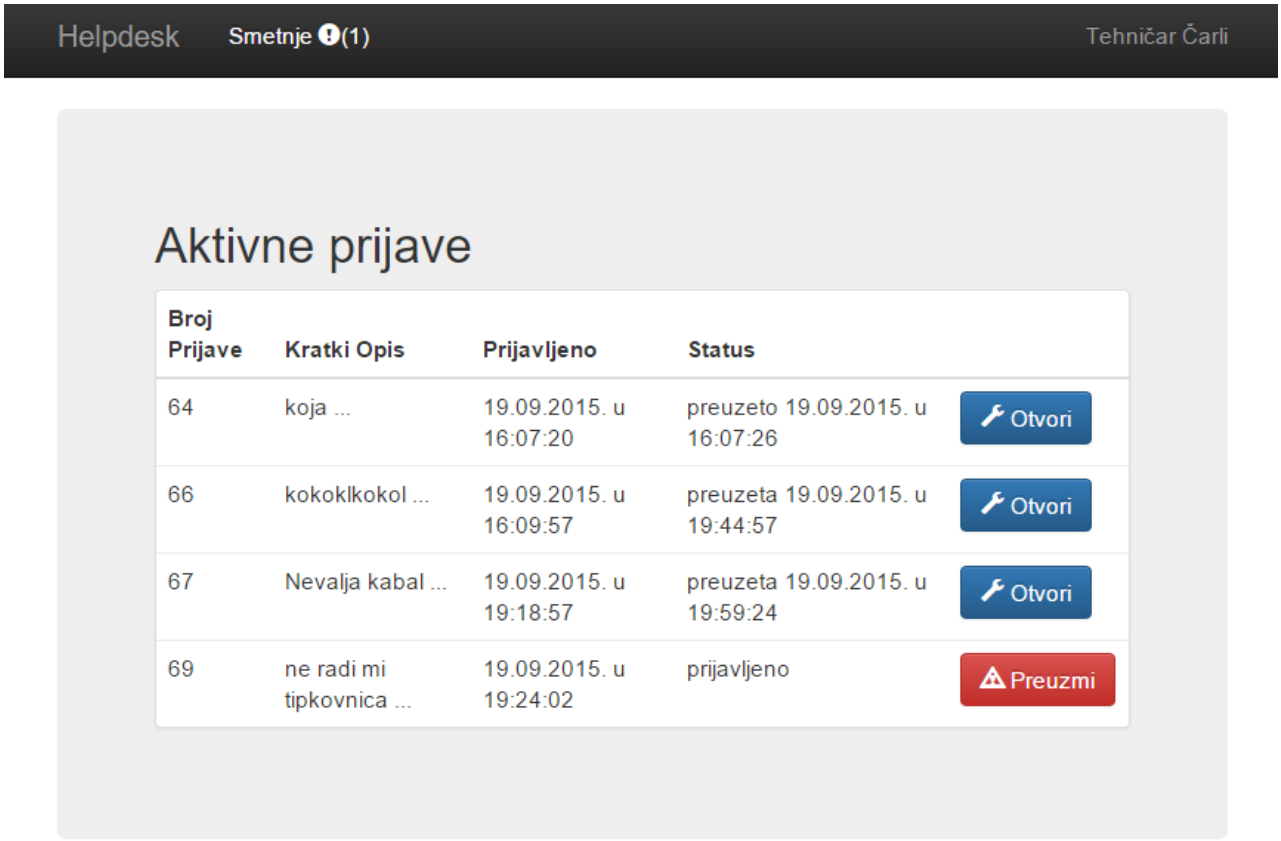


Slika 3.3 Sučelje za ulogu *korisnik*

Klikom na gumb *Prijavi* smetnja se sprema u bazu i biti će vidiljiva na u tablici aktivnih smetnji te na sučelju tehničara. Zbog ograničenja Google Apps Script-a cijela aplikacija se odvija na jednom URL-u. I „klijentski“ dio aplikacije se ne izvršava na strani klijenta. Zapravo se izvršava u sigurnosnom *Sandbox* načinu rada na poslužitelju. Google koristi takav način rada da bi zaštitio korisnike od zlonamjernog HTML i JavaScript kôda. Takav način izvršavanja ograničava naš kôd i naša aplikacija ne funkcionira na isti način na koji bi funkcionirala da je obična HTML stranica. Tako se prikazivanje pojednih „stranica“ zapravo ne događa, nego se samo skrivaju i prikazuju određeni elementi HTML stranice. Takvu funkcionalnost sam postigao korištenjem jezika JavaScript i bibilioteke JavaScript jezika jQuery.

3.3.2. Sučelje za tehničara

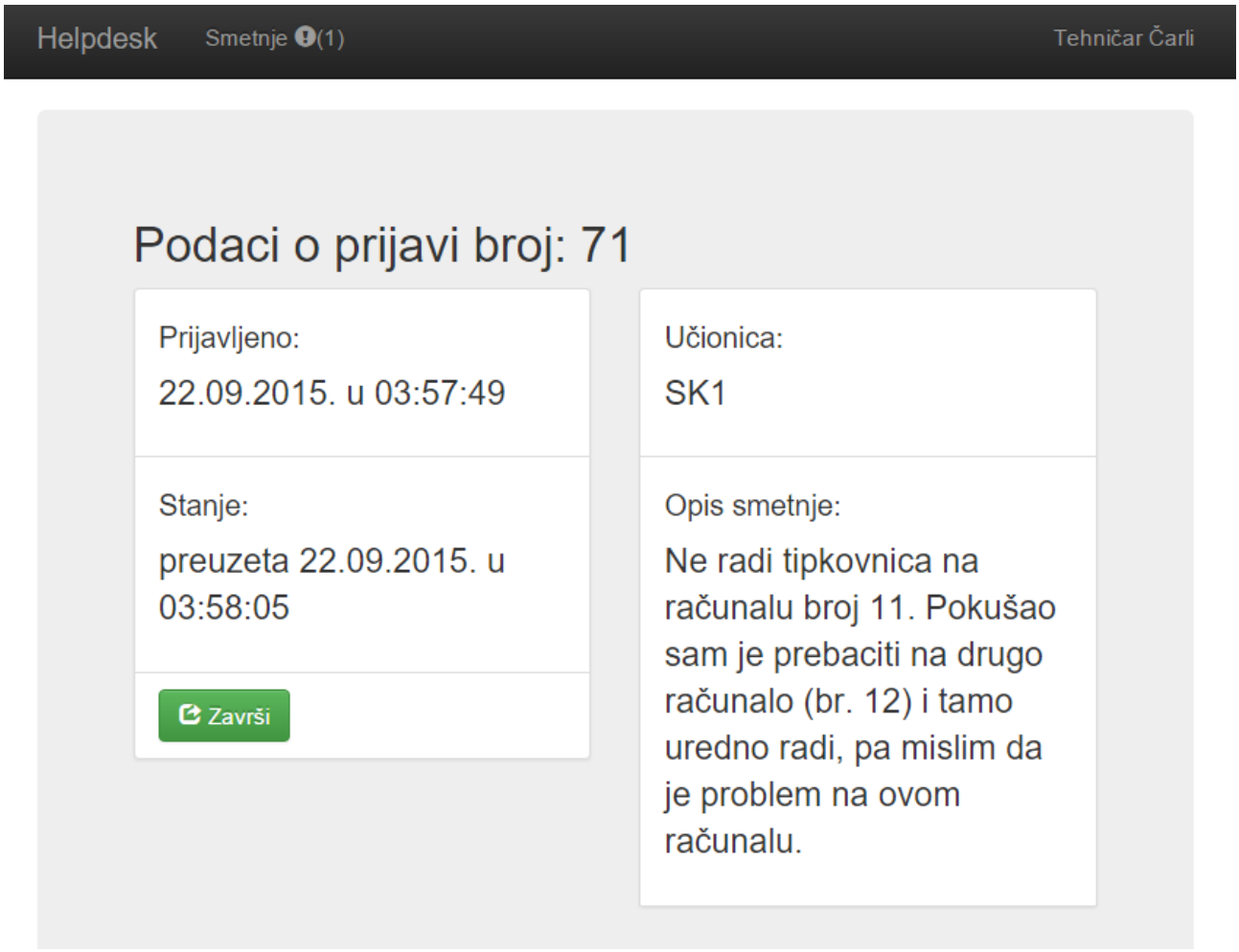
Sučelje za tehničara se ne razlikuje puno od sučelja za korisnika. Tehničar samo ima različite mogućnosti. On tako može preuzimati prijavljene smetnje kad započne sa radom na otklonu kvara te ih završavati nakon što se smetnja otkloni.



Slika 3.4 Sučelje za ulogu „tehničar“

Kada netko od korisnika prijavi novu smetnju, na sučelju tehničara je vidljiv broj novih smetnji. Smetnje koje se već odrađuju moguće je otvoriti da se vide detaljni podaci o tim smetnjama, a nove smetnje će se klikom na *Preuzmi* označiti kao preuzete i također će se otvoriti detaljan opis te smetnje. Na slici (Slika 3.5) vidimo i gumb *Završi*. Klikom na *Završi* smetnja se označava kao riješena i briše se sa liste prijavljenih smetnji. Korisnik koji je

prijavio tu smetnju može je i dalje vidjeti pod *Neaktivne prijave*. Detaljni prikaz na sučelju korisnika je gotovo jednak ovom. Jedina razlika je opcija za završavanje smetnje.



Slika 3.5 Detaljni prikaz smetnje

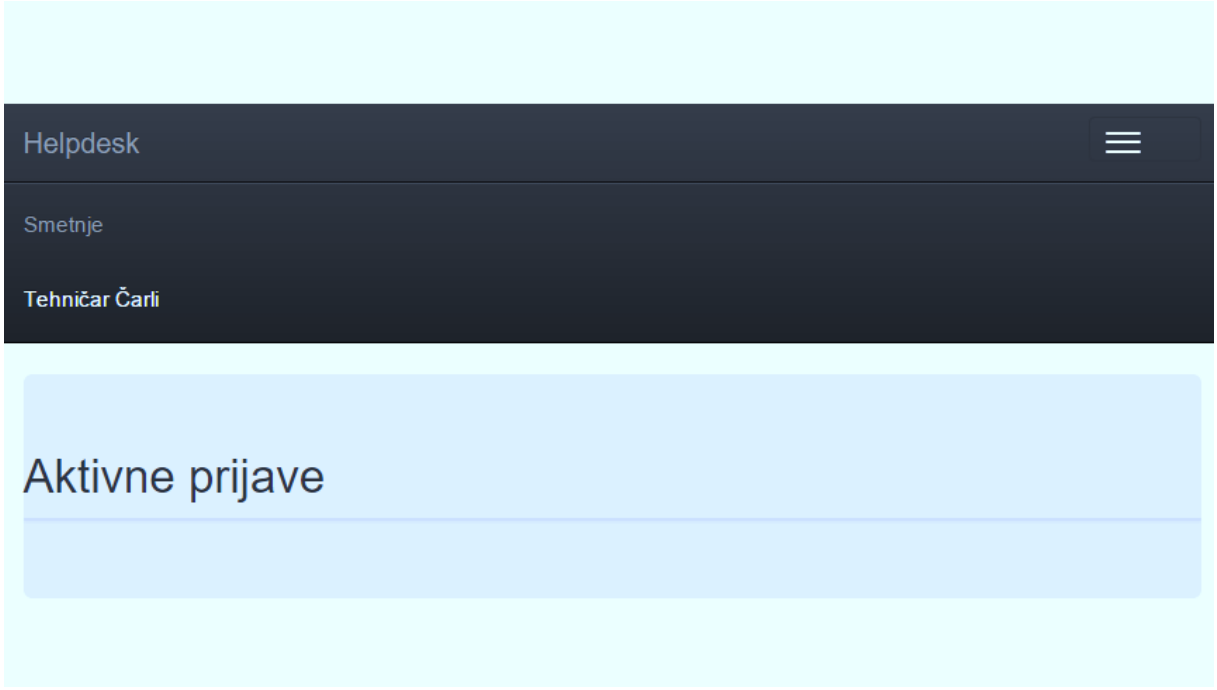
3.3.3. Sandbox način rada

Kao što je prethodno napisano HTML datoteka koje se izvršavaju korištenjem HTML servisa , izvršavaju se u Google *Sandbox* načinu rada. Kao što vidimo u kôdu (Kôd 3.2) potrebno je postaviti *SandboxMode*, za postavljanje imamo tri mogućnosti:

- IFRAME
- NATIVE

- EMULATED

IFRAME, način koji se koristi i u ovoj aplikaciji je najbolji. Najbrže se izvršava i u njemu se može dobiti nešto najsličnije običnim HTML stranicama. Omogućava gotovo sve primjene jezika JavaScript i CSS. Naime, postoje i ograničenja. Neki web preglednici kao Internet Explorer 9 i niže ne podržavaju IFRAME način. NATIVE način podržavaju svi izbornici ali ima velika ograničenja. Izvršava se puno sporije, što se može i primjetiti kod učitavanja stranice. Također, nije moguće koristit neke od dodatka vezanih za sam izgled. Na slici (Slika 3.6) je prikazano kako bi sučelje aplikacije izgledalo u NATIVE načinu rada.



Slika 3.6 Izgled stranice u NATIVE načinu rada

Vidljivo je da izbornik nije fiksiran na vrhu, tablica se uopće ne prikazuje i ne rade obavijesti o smetnjama.

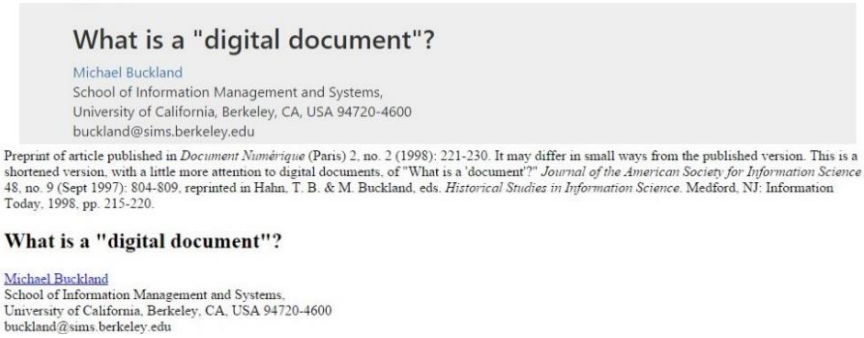
EMULATED način je još ograničeniji od NATIVE. U slučaju da na web pregledniku nije uključena mogućnost za NATIVE način, sučelje se prikazuje u EMULATED načinu. U ovom načinu, aplikacija je još sporija, ali ovaj način podržavaju gotovo svi preglednici, uključujući i prethodno spomenuti Internet Explorer 9 (jako stari web preglednici uključujući Internet Explorer 8 i niže obično uopće ne podržavaju Google-ov HTML servis.

3.4. Izgled aplikacije

Izgled aplikacije je gotovo u potpunosti napravljen korištenjem Bootstrap okruženja te jQuery biblioteke.

3.4.1. Bootstrap

Bootstrap je okruženje (*eng. Framework*) za razvoj responsvinih web aplikacija, sa naglaskom na mobile first pristup koji se fokusira da stranica dobro izgleda i na sučeljima mobilnih uređaja. Nažalost, sve mogućnosti Bootstrapa nisu podržane u Google Apps Scriptu tako da se aplikacije neće baš dobro prilagoditi sučelju mobilnog uređaja. Meni u ovom projektu nije bio cilj mobile first dizajn već da što manje vremena potrošim na stvaranje izgleda stranice. Jako puno programera baš iz tog razloga koristi Bootstrap, da ne pišu cijeli kod za izgled stranice. Umjesto pisanja CSS i JavaScript kôda koji utječe na izgled, potrebno je , ovisno o onom što želimo postići, odabrati pravu CSS klasu koju Bootstrap već ima definiranu. Na taj način lakše dolazimo do željenog izgleda web stranice. [9]



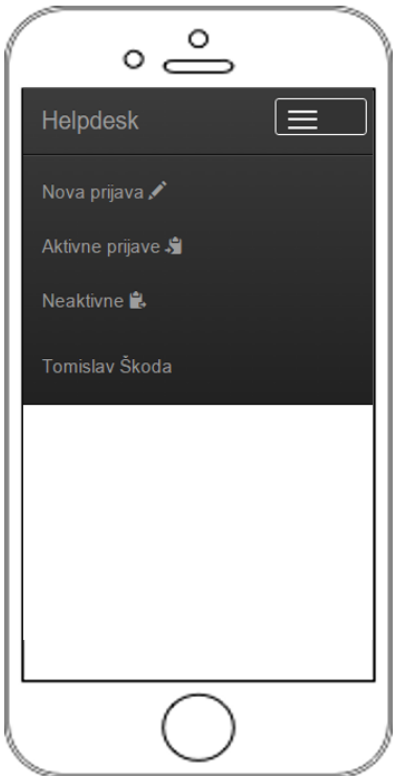
Slika 3.7 Izgled sa i bez korištenja Bootstrap biblioteke

Na slici (Slika 3.7) je prikazana ista web stranica, cijeli izgled je promijenjen tek sa nekoliko linija koda u kojima je ubačena hiperveza na bootsrap.js i bootstrap.css datoteke te pozvana jedna klasa.

```
<nav class="navbar navbar-inverse navbar-fixed-top">
```

Kôd 3.3 Postavljanje klase iz bootstrap.css bibilioteke

Kôdom (Kôd 3.3) postavlja se na <*nav*> HTML element css klasa definirana u okruženju Bootstrap za izbornik koji je fiksiran na vrhu stranice. Taj izbornik onda izgleda kao na slici (Slika 3.3). Kad bi se aplikacija pokrenula na mobilnom uređaju izgledala bi ovako (Slika 3.8), ali zbog ograničenja izvršavanja u *Sandbox* načinu rada aplikacija se ne prebaci automatski na ovakav izgled kad joj se pristupa sa mobilnog uređaja. Bootstrap je vrlo učinkovit kada je potrebno brzo rješenje za izgled web stranice ili aplikacije. Primjena okruženja Bootstrap je puno šira nego što je to prikazano u *Helpdesk* aplikaciji.



Slika 3.8 Izgled aplikacije na mobilnom uređaju

Zaključak

Iako je prošlo više od dvadeset godina od prvog *Workflow Reference Model* izdanja, još uvijek postoje sustavi koji nemaju nikakvu implementaciju upravljanja radnog toka dokumenata te upravljanja radnog toka općenito. Informacije o ovakvim sustavim nisu lako dostupne niti su raširene ili ako jesu ti sustavi su prikazani kao sustavi velikih razmjera koji potražuju ogromne resurse. Međutim, postoje i skalabilni sustavi koji ne potražuju puno resursa kao decentralizirani uređeni Peer-to-peer sustavi. Korištenje ovih sustava, njihovih principa i načina pristupa poslovnim procesima, kojih su dokumenti gotovo pa ključna stavka, trebalo bi biti vitalno za svako moderno poduzeće, instituciju ili organizaciju. Iako je implementacija spora, vidimo kako se danas sve više uvode elektronički dokumenti i iskorištava informacijska tehnologija koja polako ali sigurno izbacuje papir iz upotrebe.

U ovom radu je opisan dokument, sustav za upravljanje dokumentima, te sustav za upravljanje radnim tokom dokumenata. Također opisano je okruženje Google Apps Script, tehnologije koje su u tom okruženju korištene i njegova primjena kod izrade *Helpdesk* mrežne aplikacije koja je implementacija sustava za upravljanje radnim tokom dokumenata (koji je u ovom slučaju smetnja) u okruženju Google Apps Script.

Literatura

[1] Wikipedija, *Document* , **URL** https://en.wikipedia.org/wiki/Document

[2] Buckland M., *What is a „digital document?“*, **URL**
http://people.ischool.berkeley.edu/~buckland/digdoc.html

[3] Wikipedija , *Document management system*,
https://en.wikipedia.org/wiki/Document_management_system

[4] David Hollingsworth , Workflow Management Coalition, Studeni 1994, *The Workflow Reference Model*

[5] G. Alonso, D. Agrawal, A. El Abbadi, C. Mohan, *Functionality and Limitations of Current Workflow Management Systems*

[6] Wikipedija, *Google Apps Script,* **URL**
https://en.wikipedia.org/wiki/Google_Apps_Script

[7] Wikipedija, *Java Database Connectivity,* **URL**
https://en.wikipedia.org/wiki/Java_Database_Connectivity

[8] Wikipedija, *Microsoft Azure,* **URL** https://en.wikipedia.org/wiki/Microsoft_Azure

[9] Wikipedija, *Bootstrap front-end framework*, **URL**

https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

Sustav za upravljanje tokom dokumenata

Sažetak

Upravljanje radnim tokom dokumenta je jedna od najvažnijih komponenti sustava za upravljanje dokumentima. Primjena ovakvih sustava u svakodnevnom poslovanju je u današnje vrijeme jako česta pojava. Ti sustavi automatiziraju procese u koje je uključeno rukovanje bilo kojom vrstom dokumenata. Google Apps Script jezik je lak za učenje, baziran na JavaScriptu. Izvršava se na Google Cloud usluzi i dostupan svima sa računom Google. U implementaciji je korišten spoj više različitih tehnologija(JDBC, Azure, MSSQL, Bootstrap). da bi se omogućuilo dobro korisničko iskustvo.

Document workflow management system

Abstract

Workflow management is one of the main components of document management software.These systems are very often used in everyday buissenes processes. These system automatize processes involved with any kind of document. Google Apps Script is script programming language. It's easy to learn, based on JavaScript. It executes on Google Cloud Service and it's available to anyone with Google Account. For implementation of document worfklow many different technologies where used (JDBC, Azure, MSSQL, Bootstrap) to provide good user expirience.