Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/18BE18Brodic

PROŠIRENJE PODRŠKE ZA TIPOVE PODATAKA U CAMUNDA KORISNIČKIM FORMAMA

EXTENDING SUPPORT FOR DATA TYPES IN CAMUNDA USER FORMS

Nikola Brodić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisani su sistemi za upravljanje poslovnim procesima i predstavljeni su najpoznatiji. Detaljno je opisan Camunda Platform BPMS i analizirane su njegove prednosti i nedostaci. Kreirana je i opisana programska implementacija koja rešava uočene nedostatke.*

Ključne reči: *upravljanje poslovnim procesima, sistem za upravljanje poslovnim procesima, Camunda Platform*

Abstract – *This paper describes Business Process Management Systems and presents the most common ones. Camunda Platform BPMS is described in detail and it's pros and cons are analyzed. Software implementation that resolves mentioned disadvantages is created and described.*

Keywords: *Business Process Management, Business Process Management System, Camunda Platform*

1. UVOD

Upravljanje poslovnim procesima (eng. *Business Process Management*, BPM) je organizaciona disciplina koja zahteva od organizacija da sagledaju i analiziraju svoje poslovne procese i njihovo trenutno stanje kako bi uočili moguća poboljšanja.

Upravljanje poslovnim procesima podržano je od strane različitih sistema za upravljanje poslovnim procesima (eng. *Business Process Management System/Suite*, BPMS). Na tržištu je dostupan veliki broj BPM sistema, među kojima *Camunda Platform* predstavlja čest i popularan izbor.

Međutim, i pored dobrih osobina i prednosti, značajan nedostatak *Camunda* platforme predstavlja nedovoljno dobra podrška za kreiranje i generisanje korisničkih formi za unos podataka. Način kreiranja formi najprilagođeniji poslovnim korisnicima je kreiranje generisanih formi koje imaju ograničenu podršku u pogledu polja forme, tipova podataka i validacija. Stoga je ideja ovog rada analiza, ali i implementacija konkretnog rešenja kojim se mogu prevazići postojeći problemi *Camunda* formi.

2. UPRAVLJANJE POSLOVNIM PROCESIMA

Poslovni proces (eng. *business process*) predstavlja skup aktivnosti kojima se postiže određeni organizacioni cilj.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miroslav Zarić, vanr. prof.

Upravljanje poslovnim procesima (eng. *Business Process Management*, BPM) je metodologija za upravljanje procesima i radnim tokovima unutar neke organizacije radi povećanja efikasnosti, performansi i agilnosti u okviru svakodnevnih poslovnih aktivnosti [1]. Cilj organizacija koje primenjuju BPM je kontrola postojećih procesa i njihova kontinuirana optimizacija.

2.1. Sistemi za upravljanje poslovnim procesima

Sistemi za upravljanje poslovnim procesima (eng. *Business Process Management System/Suite*, BPMS) nude platforme i alate koji pomažu organizacijama da dizajniraju, modeluju, izvršavaju, automatizuju i unapređuju skup aktivnosti i zadataka čijim sprovođenjem se postižu organizacioni ciljevi. BPM softverski sistemi su dizajnirani da pomognu organizacijama u optimizaciji svakodnevnih poslovnih procesa radi postizanja maksimalne efikasnosti i produktivnosti [2]. Ovakva rešenja maksimizuju smanjenje troškova, obezbeđuju rast poslovanja, upravljaju zaduženjima i izvršavaju integrisanu poslovnu strategiju.

Opšta arhitektura BPM sistema obuhvata nekoliko komponenti sa jedinstvenim ulogama. *Izvršno okruženje* (eng. *Execution engine, Process engine*) je centralni deo sistema za upravljanje poslovnim procesima koji obezbeđuje koordinirano izvršavanje aktivnosti unutar procesa.

Osnovna namena *alata za modelovanje procesa* (eng. *Process modeling tool*) je da korisnicima omogući kreiranje i izmenu modela poslovnih procesa. *Lista radnih zadataka* (eng. *Worklist handler*) je BPMS komponenta putem koje se učesnicima procesa dodeljuju i/ili nude na izbor radni zadaci koje treba obraditi. *Alati za nadgledanje i upravljanje* (eng. *Administartion and monitoring tools*) služe za upravljanje stvarima neophodnim za nesmetano funkcionisanje sistema za upravljanje poslovnim procesima.

2.2. Izbor BPM sistema

Kao što i samo ime nagoveštava, upravljanje poslovnim procesima u fokus stavlja poslovanje, umesto tehnologije. Ipak, to poslovanje treba da bude podržano adekvatnom tehnologijom. Zato je bitno odabrati sistem koji je pogodan za poslovanje, zahteva minimalno kodiranja i omogućava poslovnim korisnicima da samostalno kreiraju i upravljaju procesima uz visok stepen IT podrške, dok IT podrška treba da upravlja samim sistemom i pravima pristupa.

Trenutno je na tržištu dostupan veliki broj sistema za upravljanje poslovnim procesima, pa izbor najprikladnijeg od njih predstavlja izazov za svaku organizaciju. Ne

postoji univerzalno pravilo prema kom se BPMS može odabrati, jer nemaju sve organizacije iste zahteve i potrebe. Ništa manje bitna stavka je i budžet, jer ovi sistemi predstavljaju veliki izdatak za svaku kompaniju, s obzirom da na cenu utiče i broj zaposlenih koji će ga koristiti. Stoga je bitno da organizacije ozbiljno razmotre sve raspoložive opcije pre nego što se odluče za BMPS koji će primeniti u svom poslovanju, jer je kasnija promena sistema zahtevan i mukotrpan posao. Neki od sistema za upravljanje poslovnim procesima vredni razmatranja su: *Kissflow*, *Bizagi*, *ProcessMaker*, *Bonita* i *Camunda*.

3. CAMUNDA PLATFORM

Camunda Platform je platforma otvorenog izvornog koda razvijena u Java programskom jeziku. Dolazi sa alatima za kreiranje modela poslovnih procesa i modela odlučivanja, izvršavanje *deploy*-ovanih modela u produkciji i izvršavanje radnih zadakata od strane učesnika procesa.

Camunda je resursno nezahtevan radni okvir (eng. *lightweight framework*) koji se može koristiti kao samostalno izvršno okruženje ili se može ugraditi u namenske Java aplikacije [3]. Celokupna *Camunda* platforma sastoji se od nekoliko komponenti namenjenih različitim tipovima korisnika.

Modeler je desktop aplikacija koja omogućava programerima i poslovnim analitičarima da kreiraju i menjaju BPMN dijagrame i DMN tabele odlučivanja [3].

Fajlovi kreirani *Modeler*-om se *deploy*-uju u izvršno okruženje, tj. *Process Engine*. *Process Engine* koristi BPMN parser da pretvori BPMN 2.0 i DMN XML fajlove u Java objekte. Hardverski je nezahtevan i zauzima svega 3MB prostora na disku.

Camunda veb aplikacije uključuju sledeće alate za programere i poslovne korisnike [4]:

- REST API (skr. od ***RE****presentational* ***S****tate* ***T****ransfer Application Programming Interface*) omogućava upotrebu *Process Engine*-a pozivanjem iz udaljene ili JavaScript aplikacije.
- *Tasklist* učesnicima procesa nudi uvid u radne zadatke koji su im dodeljeni i omogućava im da otvaraju forme u kojima treba da unesu tražene podatke i na taj način ispune svoje radne zadatke.
- *Cockpit* operatorima procesa omogućava pretragu i praćenje izvršavanja procesnih instanci, kao i vršenje intervencija u slučaju incidenata.
- *Admin* administratorima procesa omogućava upravljanje pojedinačnim korisnicima, grupama korisnika i njihovim pravima pristupa.

3.1. Arhitektura Camunda platforme

Kako je *Camunda* razvojna platforma bazirana na Javi, osnovne komponente napisane su u Java programskom jeziku. Ideja je da se Java programerima pruže alati neophodni za dizajniranje, implemetaciju i izvršavanje poslovnih i radnih procesa na Java virtuelnoj mašini (eng. *Java Virtual Machine*, JVM). Za programere koji rade u drugim programskim jezicima, *Camunda* nudi i REST API što omogućava kreiranje aplikacija koje se povezuju na udaljeno izvršno okruženje.

Camunda Platform, kao fleksibilan radni okvir, može biti *deploy*-ovan na različite načine. Tri najčešća načina su: ugrađeno izvršno okruženje, deljeno izvršno okruženje i samostalno izvršno okruženje [5]. Ugrađeno izvršno okruženje (eng. *Embedded Process Engine*) podrazumeva da se *Process Engine* dodaje kao biblioteka u proizvoljnu aplikaciju. Deljeno izvršno okruženje (eng. *Shared, Container-Managed Process Engine*) podrazumeva da se *Process Engine* pokreće unutar izvršivog kontejnera, kao što su *Servlet Container*, *Application Server* i slično. Samostalno izvršno okruženje (eng. *Standalone (Remote) Process Engine Server*) podrazumeva da se *Process Engine* nudi kao mrežni servis.

Da bi se pružilo skaliranje (eng. *scale-up*) ili smanjila mogućnost nedostupnosti sistema (eng. *fail-over*), izvršno okruženje se može distribuirati na više čvorova u klasteru. Pritom, svaka instanca izvršnog okruženja mora biti povezana na deljenu bazu podataka.

Radi opsluživanja više nezavisnih strana jednom *Camunda* instalacijom, izvršno okruženje podržava višeklijentski režim (eng. *multi-tenancy*). Konkretno su podržani sledeći modeli:

- Razdvajanje podataka na nivou tabela upotrebom različitih šema baze podataka ili različitih baza podataka
- Razdvajanje podataka na nivou redova upotrebom označavanja klijenata (eng. *tenant marker*)

Korisnici treba da odaberu model koji najbolje odgovara njihovim potrebama za razdvajanjem podataka.

3.2. Prednosti Camunda platforme

Camunda platforma nudi mnoštvo alata koji su otvorenog izvornog koda i besplatni za korišćenje, kao na primer *Camunda Modeler*.

Bez obzira što ne važi za *low-code* platformu, *Camunda* nudi prijatno korisničko iskustvo za iskusne Java programere. Iako zahteva poznavanje programiranja, platforma je jasno koncipirana i laka za korišćenje.

Još jedna prednost je što omogućava jednostavan *deployment* monolitne platforme. Organizacije su u mogućnosti da *deploy*-uju niz manjih procesa koji se bave delovima nekog velikog procesa bez potrebe za značajnim izmenama [6].

Cilj *Camunda* platforme je da istovremeno bude prihvatljiva i programerima i poslovnim korisnicima, što nije jednostavan zadatak. Da bi udovoljila programerima, *Camunda* ima veoma hardverski nezahtevno jezgro i ne uslovljava da razvoj procesa bude zasnovan isključivo na kreiranju modela. Sa druge strane, za poslovne korisnike očuvava jednostavnost upotrebe *Modeler*-a, omogućava im da jednostavno menjaju i prilagođavaju elemente aplikacije i pruža intuitivno nadgledanje u realnom vremenu putem anotacija modela [7].

3.3. Nedostaci Camunda platforme

Iako je besplatna, *Community* verzija *Camunda* platforme primamljiv izbor, mnogi alati koji sistem za upravljanje poslovnim procesima čine korisnim dodatkom poslovanju dostupni su samo kroz plaćenu, *Enterprise* verziju. Još jedna mana *Community* verzije su i visoki troškovi u slučaju potrebe za tehničkom podrškom.

Iskusni Java programeri hvale *Camunda* platformu, ali i pored toga priznaju da njena logika nije najbolje prilagođena običnim poslovnim korisnicima.

Pored opisanih nedostataka, problem *Camunda* platforme predstavlja i nedovoljno dobra podrška za kreiranje i generisanje korisničkih formi za unos podataka. Iako postoji nekoliko vrsta formi, svaka ima svoje nedostatke. Zajednička mana za neke od njih je to što ih ne mogu kreirati obični poslovni korisnici, jer zahtevaju znanje programiranja i programskih jezika. Sa tog stanovišta najbolje su generisane forme (eng. *Generated Task Forms*), međutim i one imaju neke značajne nedostatke opisane u nastavku.

Generisane forme se u *Camunda*-i generišu na osnovu metapodataka o formi (eng. *Form Data Metadata*) dostupnih u okviru BPMN 2.0 XML notacije. Metapodaci o formi predstavljaju skup *Camunda*-inih proširenja BPMN 2.0 notacije kojima je omogućeno kreiranje formi i njihovih polja u okviru BPMN 2.0 XML fajla [8]. Ti metapodaci, osim teksualno, mogu biti i grafički kreirani i menjani upotrebom *Camunda Modeler* alata, što je preferirani i jednostavniji način iz ugla korisnika. Njegov *Properties* panel sadrži karticu *Forms* putem koje je moguće definisati željena polja forme.

Prilikom kreiranja forme, za svako polje je neophodno definisati tip. Od tipova, *Camunda* podrazumevano nudi *string*, *long*, *date*, *boolean* i *enum*. Pored tipova, za svako polje forme se mogu definisati i uslovi validnosti, odnosno validacije (eng. *validations*), i proizvoljna svojstva (eng. *properties*) u tekstualnom obliku. Dve ugrađene validacije zajedničke za sve tipove polja su *required*, kojom se zahteva da polje bude popunjeno, i *readonly*, kojom se zahteva da polje ostane ne popunjeno.

Međutim, podržani tipovi podataka i validacije nisu dovoljni za modelovanje realnih poslovnih procesa. Nedostaju tipovi podataka za razlomljene brojeve, fajlove i zadavanje vremena, kao i validacije za granularnost unete vrednosti, minimalne i maksimalne vrednosti za datum i vreme i tip fajla. Uz to, ne postoji način da se dinamički definišu stavke za izbor u okviru polja *enum* tipa, niti mogućnosti izbora više od jedne ponuđene stavke.

Iako nove verzije *Camunda* platforme redovno izlaze dva puta godišnje, unapređivanje generisanih formi se ostavlja po strani. Korisnici se putem foruma podstiču na upotrebu ugrađenih formi, iako one ne predstavljaju adekvatnu zamenu, jer nisu namenjene istim tipovima korisnika. Stoga je u studiji slučaja predstavljena implementacija uočenih nedostataka, kao praktični doprinos ovog rada.

4. STUDIJA SLUČAJA

Ideja ovog rada je analiza, ali i implementacija konkretnog rešenja kojim se mogu prevazići postojeći problemi *Camunda* formi. Kao osnova za nadogradnju i unapređenje uzete su generisane forme, jer predstavljaju jedinu vrstu formi koje se mogu kreirati direktno iz *Modeler*-a i od strane različitih tipova korisnika, koji nisu isključivo programeri. Implementacija rešenja se sastoji od *Camunda Forms Extension* Java arhive (eng. *Java ARchive*, JAR) i *Form Generator* JavaScript fajla i nalazi se na GitHub repozitorijumu sa nazivom „Camunda Forms Extension“ [9], zajedno sa primerom upotrebe kreiranog rešenja.

Camunda Forms Extension je osmišljen kao dodatak (eng. *plugin*) za Java aplikacije napisane upotrebom *Spring Boot* radnog okvira i prvenstveno se oslanja na *Camunda Spring Boot Starter for Webapps 7.15.0* Maven zavisnost (eng. *dependency*). Sadrži implementaciju dodatnih tipova podataka i validatora koji nedostaju u osnovnoj *Camunda* biblioteci i funkcije za obradu podataka unetih putem generisanih formi. Cilj ovog dodatka je podrška šireg skupa poslovnih procesa kod kojih korisničke forme zahtevaju tipove polja koje osnovna *Camunda* biblioteka ne nudi.

Form Generator je osmišljen kao dodatak za prezentacioni sloj razvijen upotrebom HTML, CSS i JavaScript jezika. Namenjen je generisanju HTML formi na osnovu podataka definisanih u okviru *Forms* kartice u *Camunda Modeler*-u, slično kao što to radi *Camunda Tasklist*, samo za veći broj tipova polja. Sastoji se od nekoliko enkapsuliranih funkcija za generisanje HTML elemenata za unos podataka i ulazne funkcije koja za svako polje definisano u *Forms* kartici poziva neku od tih enkapsuliranih funkcija. S obzirom da je stilizacija formi neizbežna u današnje vreme, omogućeno je i definisanje CSS klasa za svaki generisani HTML element, kao i izbor načina za označavanje polja forme, pa se polja mogu označiti labelama ili *placehoder*-ima.

Od dodatno podržanih tipova, *Camunda Forms Extension* sadrži *double*, *integer*, *time*, *textarea*, *radio* i *file* tipove polja. Zajedničko za ove tipove je da svaki od njih nasleđuje *SimpleFormFieldType* klasu iz osnovne *Camunda* biblioteke. Tom klasom se obezbeđuje struktura koju svaki tip mora da ispoštuje da bi *Process Engine* mogao da ga prihvati i obradi.

Tip *double* uveden je kako bi se podržao rad sa razlomljenim brojevima, odnosno brojevima sa decimalnim ciframa. Tip *integer* uveden je kako bi se podržao rad sa 32-bitnim celim brojevima, jer *Camunda* nudi podršku samo za 64-bitne cele brojeve putem *long* tipa. Za numeričke tipove *double*, *integer* i *long* kreiran je *step* validator. On omogućava definisanje i proveru granularnosti za unete numeričke vrednosti. Pored *step* validatora, obezbeđeni su i novi *min* i *max* validatori čija je namena proširena tako da, osim za numeričke, mogu da se koriste i za vrednosti koje predstavljaju datum i vreme. Tip *time* je kreiran kako bi se podržale vrednosti koje iskazuju vreme. Tipovi *textarea*, *radio* i *file* uvedeni su kako bi se podržao širi skup HTML elemenata za kreiranje formi. Kao osnova za njihovu implementaciju upotrebljen je ugrađeni tip *string*, a potom su uvedene neophodne izmene. Uvođenjem podrške za fajlove, bilo je neophodno kreirati i validacije *multiple* i *accept*. Pored toga, za sva polja *string* tipa je obezbeđen i *pattern* validator koji omogućava da se uneta tekstualna vrednost proveri zadatim regularnim izrazom.

Pored tipova i validacija, *Camunda Forms Extension* nudi i servisnu klasu nazvanu *CamundaService* koja, između ostalog, poseduje metode neophodne za pokretanje procesne instance, dobavljanje definicije generisane forme, *submit* podataka unetih kroz formu, završavanje radnog zadatka itd. U okviru ove klase kreirana je i

metoda koja omogućava dinamičko postavljanje opcija za izbor kod polja *enum* tipa.

Metoda *getFormData()* klase *CamundaService* za prosleđene identifikatore procesne instance i radnog zadatka vraća objekat koji u sebi sadrži listu polja forme (Listing 4.1). Svako polje forme sadrži konfiguraciju tog polja definisanu putem *Forms* kartice *Modeler*-a. Ta konfiguracija je u *Form Generator*-u iskorišćena za generisanje odgovarajućih HTML elemenata forme.

```
public FormDataDTO getFormData(String procInstanceId,
                               String taskId) {
  TaskFormData tfd = formService.getTaskFormData(taskId);
  List<FormField> formFields = tfd.getFormFields();
  this.setEnumValues(procInstanceId, formFields);

  return new FormDataDTO(procInstanceId, taskId, formFields);
}
```

Listing 4.1 Implementacija metode *getFormData()*

Kao što je na početku spomenuto, *Form Generator* je JavaScript fajl sa funkcijama koje generišu različite tipove HTML elemenata za kreiranje formi, a na osnovu podataka dobijenih pozivom metode *getFormData()* iz *Camunda Forms Extension* dodatka. Funkcija koju treba pozvati da bi izgenerisala polja forme nazvana je *generateFormFields()*. Kao što se može videti iz listinga 4.2, ova funkcija prima nekoliko parametara. Parametar *formId* predstavlja identifikator HTML forme u koju će se dodati izgenerisana polja. Sledeći parametar, nazvan *jsonResponse*, predstavlja listu konfiguracija svih polja forme u JSON formatu dobijenu udaljenim pozivom metode *getFormData()*. Parametar *css* omogućava zadavanje CSS klasa za stilizovanje izgenerisanih HTML elemenata. Poslednji parametar, sa nazivom *withLabels*, omogućava pozivaocu funkcije da odabere da li će se polja forme imenovati labelama ili *placeholder*-ima. Uloga ove funkcije je da prođe kroz sva dobijena polja forme, za svako od njih izgeneriše HTML element, i taj element doda u zahtevanu formu.

```
function generateFormFields(formId, jsonResponse, css = null,
                            withLabels = false) {
  let fragment = new DocumentFragment();
  jsonResponse.formFields.forEach(field => {
    let div = createFormField(field, css, withLabels);
    fragment.append(div);
  });
  document.getElementById(formId).prepend(fragment);
}
```

Listing 4.2 Implementacija funkcije *generateFormFields()*

5. ZAKLJUČAK

Upravljanje poslovnim procesima je neizostavna metodologija kod organizacija u kojima se odvija veći broj poslovnih procesa. Organizacije koje ne koriste sisteme za upravljanje poslovnim procesima suočavaju se sa problemom neefikasnog i neorganizovanog sprovođenja poslovnih aktivnosti, kao i sa nedovoljnom iskorišćenošću raspoloživih ljudi i resursa. Stoga je broj organizacija koje uvode ovu metodologiju u svoje poslovanje sve veći. U takvoj situaciji organizacije moraju da sagledaju prednosti i nedostatke dostupnih BPM sistema i odluče se za onaj koji najviše odgovara njihovim potrebama.

U radu je izdvojena *Camunda* platforma, kao popularan izbor koji pored plaćene nudi i besplatnu varijantu zavidnih mogućnosti. Ipak, ozbiljan nedostatak predstavlja nemogućnost kreiranja složenih generisanih formi sa raznolikim poljima, obrađen i rešen u okviru ovog rada.

Predstavljena je implementacija rešenja koje omogućava da se uočeni nedostaci prevaziđu i na taj način prošire mogućnosti platforme u pogledu kreiranja korisničkih formi. Implementacija se sastoji od *Camunda Forms Extension* Java arhive i *Form Generator* JavaScript fajla i dostupna je na javnom GitHub repozitorijumu [9].

Raznovrsnost polja forme, njihovih tipova i validacija je velika, pa ih iz tog razloga nije bilo moguće sve podržati. Implementacija je usmerena na podršku onih koji su najčešće korišćeni, kako bi se maksimizovala njegova primenljivost. To ostavlja prostor za unapređenje implementacijom preostalih polja forme, tipova podataka i validacija. Stilizacija je omogućena kroz zadavanje CSS klasa, ali je mogućnost preraspoređivanja (eng. *layouting*) polja forme ograničena redosledom kojim su definisani u *Modeler*-u. Ovo takođe predstavlja oblast na kojoj bi se moglo poraditi u budućnosti, s obzirom da je preraspoređivanje gotovo neizbežno zbog sve veće primene mobilnih uređaja kako u životu, tako i u poslovanju.

6. LITERATURA

[1] https://www.pega.com/bpm, pristupljeno u novembru 2021.)

[2] https://kissflow.com/workflow/bpm/business-process-management-systems-top-features (pristupljeno u novembru 2021.)

[3] https://en.wikipedia.org/wiki/Camunda (pristupljeno u decembru 2021.)

[4] https://docs.camunda.org/manual/7.15/introduction (pristupljeno u decembru 2021.)

[5] https://docs.camunda.org/manual/7.15/introduction/architecture (pristupljeno u decembru 2021.)

[6] https://www.processmaker.com/wp-content/uploads/2020/05/Camunda_vs_ProcessMaker_Comparison.pdf (pristupljeno u decembru 2021.)

[7] https://www.srijan.net/resources/blog/business-process-management-camunda-workflow (pristupljeno u decembru 2021.)

[8] https://docs.camunda.org/manual/7.14/user-guide/task-forms (pristupljeno u decembru 2021.)

[9] https://github.com/NikolaBrodic/camunda-forms-extension (pristupljeno u decembru 2021.)

Kratka biografija:

Nikola Brodić rođen je 28.05.1997. godine u Sremskoj Mitrovici. Osnovne akademske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu upisao je 2016. godine. Diplomirao je 18.09.2021. godine i iste godine upisao master akademske studije.