Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE02Novak

BIBLIOTEKA ZA GENERISANJE STANDARDNIH FORMI U ANGULARU

LIBRARY FOR GENERATING STANDARD FORMS IN ANGULAR

Kim Novak, Milan Vidaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisana je specifikacija i implementacija biblioteke za generisanje standardnih formi u Angularu, kao i jednostavne veb aplikacije u koju je biblioteka integrisana.*

Ključne reči: *Angular, Web, REST, JSON*

Abstract – *This paper contains specification and implementation of the library for generating standard forms in Angular, as well as the specification and implementation of a simple web application which integrates the library.*

Keywords: *Angular, Web, REST, JSON*

1. UVOD

Manipulisanje podacima je sastavni deo skoro svakog informacionog sistema. Grafički korisnički interfejs pomoću kojeg se manipuliše podacima se najčešće sastoji od polja za unos grupisanih unutar forme. Ukoliko informacioni sistem koji razvijamo ima veliki broj entiteta, razvoj formi za svakog od njih je dug proces koji može rezultovati neodrživim sistemom. Da bi se ovaj problem rešio, potrebno je napraviti ponovno iskoristive forme koje su generične i implementiraju set funkcionalnosti koje su zajedničke za svaki od entiteta.

Standardne forme predstavljaju jedno rešenje tog problema i na njima se bazira rešenje opisano u ovom radu. Kako bi se razvoj informacionog sistema još više ubrzao potrebno je izdvojiti implementaciju standardnih formi u modul koji može da se integriše u bilo koji drugi informacioni sistem baziran na istim tehnologijama i da se na taj način standardne forme generišu, bez potrebe da se razvijaju ispočetka.

2. OPIS KORIŠĆENIH TEHNOLOGIJA

Biblioteka za generisanje standardnih formi napisana je u TypeScript jeziku, koristeći Angular radni okvir.

Klijentska aplikacija razvijena za potrebe demonstracije funkcionalnosti biblioteke, napisana je u istim tehnologijama, dok je serverska strana napisana koristeći JavaScript jezik, NodeJS i ExpressJS radni okvir.

2.1. TypeScript

JavaScript jezik je jedan od najpopularnijih jezika korišćenih za razvoj veb aplikacija.

JavaScript je dinamičan, slabo tipiziran jezik, koji je ili just-in-time kompajliran ili interpretiran. Ova tri svojstva JavaScript-a ostavljaju dosta prostora da programeri naprave greške prilikom programiranja koje se neće ispoljiti pre vremena izvršavanja. Te greške najčešće su vezane za implicitnu konverziju tipa promenljive.

TypeScript omogućava da se broj ovakvih grešaka svede na minimum, dodavanjem sloja oko JavaScript-a, a taj sloj je TypeScript-ov sistem tipova. TypeScript je i jezik i set alatki za generisanje JavaScript-a.

2.2. Angular

Angular je radni okvir baziran na TypeScript jeziku koji omogućava razvoj klijentskih aplikacija za različite platforme. Moguće je upotrebiti ga za razvoj:

- veb aplikacija,
- mobilnih aplikacija i
- desktop aplikacija.

Fundamentalni koncepti u Angularu su:

- NgModules,
- komponente,
- šabloni,
- direktive i
- umetanje zavisnosti

Arhitektura Angular aplikacije oslanja se na neke od fundamentalnih koncepata. Osnovni gradivni blokovi su NgModules, koji pružaju kontekst prevođenja (eng. compilation context) komponentama [1].

2.3. NodeJS

NodeJS je JavaScript izvršno okruženje. Pošto NodeJS omogućava da se JavaScript izvršava van internet pretraživača, moguće je koristiti ga za razvoj serverske strane veb aplikacije. Da bi se ubrzao razvoj serverske aplikacije u NodeJS-u koristi se neki od radnih okvira namenjenih razvoju veb aplikacija. Najpopularnije rešenje je ExpressJS.

Pomoću ExpressJS-a moguće je jednostavno i brzo razviti serversku aplikaciju spremnu za upotrebu.

3. SPECIFIKACIJA APLIKACIJE

Informacioni sistemi često zahtevaju implementiranje CRUD (Create, Read, Update, Delete) operacija za entitete koje sadrže. Entiteta može biti veliki broj i ukoliko bi se za svaki od entiteta svaki put ispočetka implementirale CRUD operacije, razvoj bi trajao dugo i sistem bi postao teško održiv.

Neophodno je prepoznati delove koji se mogu generalizovati i njih izdvojiti u ponovno iskoristivi modul.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Vidaković, red. prof.

Aplikacija rešava navedeni problem upotrebom standardnih formi. Standardne forme predstavljaju grafički korisnički interfejs koji omogućava jednostavan rad sa podacima. Operacije koje standardne forme pružaju su:

- pregled podataka,
- dodavanje novih podataka,
- izmena postojećih podataka,
- brisanje postojećih podataka i
- pretraga nad podacima.

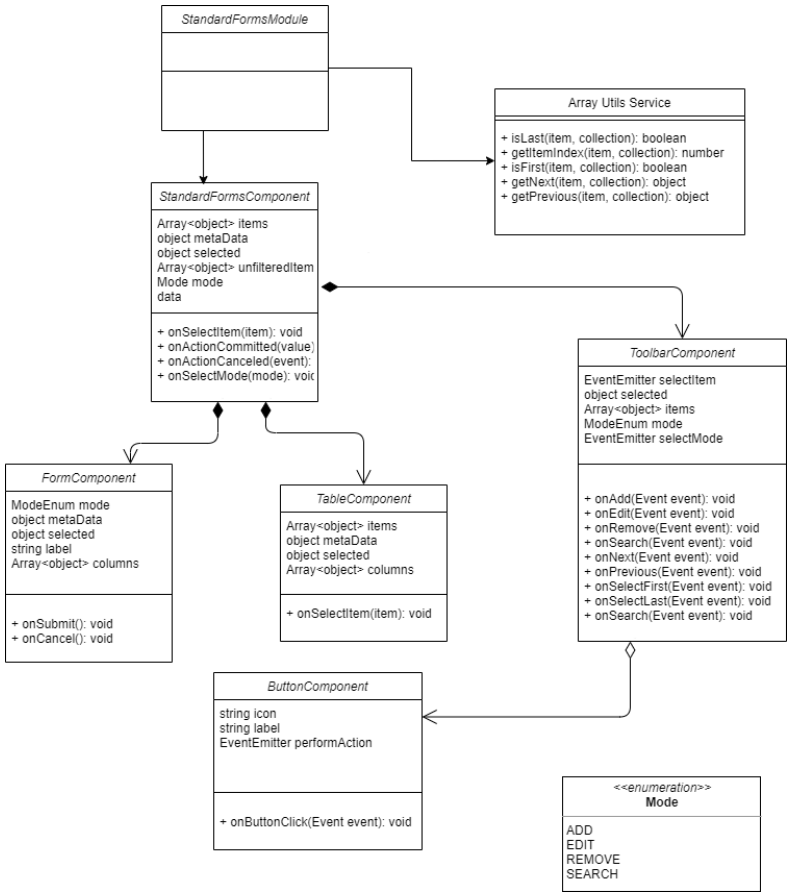
Sa aspekta funkcionalne analize zahteva, sistem će omogućiti korisniku sledeće funkcionalnosti:

- odabir režima rada, sistem se može nalaziti u režimu izmene, unosa, brisanja ili pretrage,
- unos podataka,
- čuvanje izmena i
- odbacivanje izmena

Na slici 1 prikazan je dijagram klasa. Biblioteka za generisanje standardnih formi je modul koji se sastoji od komponenti i servisa. Korenska komponenta je StandardFormsComponent koja u sebi sadrži ToolbarComponent, TableComponent i FormComponent.

ToolbarComponent sadrži više ButtonComponent-i.

Biblioteka za generisanje standardnih formi ima definisan konačan broj režima rada u kom forma može da bude, navedenih kroz nabrajanje. Ovo nabrajanje prikazano je u dijagramu pod nazivom Mode.



Slika 1. *Dijagram klasa*

4. IMPLEMENTACIJA

Biblioteka za generisanje standardnih formi implementirana je kao Angular biblioteka. Da bi se ispitala funkcionalnost ove biblioteke, integrisana je u jednostavnu Angular klijentsku aplikaciju koja komunicira sa serverom kako bi čitala i skladištila podatke.

4.1. Arhitektura

Arhitektura rešenja prati principe slojevite arhitekture (eng. Layered Architecture). Slojevita arhitektura podrazumeva organizaciju komponenti u slojeve, gde svaki sloj obavlja određenu ulogu u sistemu. Iako slojevita arhitektura ne određuje koliko slojeva mora postojati, većina slojevitih arhitektura sastoji se iz četiri sloja:

- prezentacioni sloj,
- sloj biznis logike,
- perzistencioni sloj i
- sloj baza podataka [2].

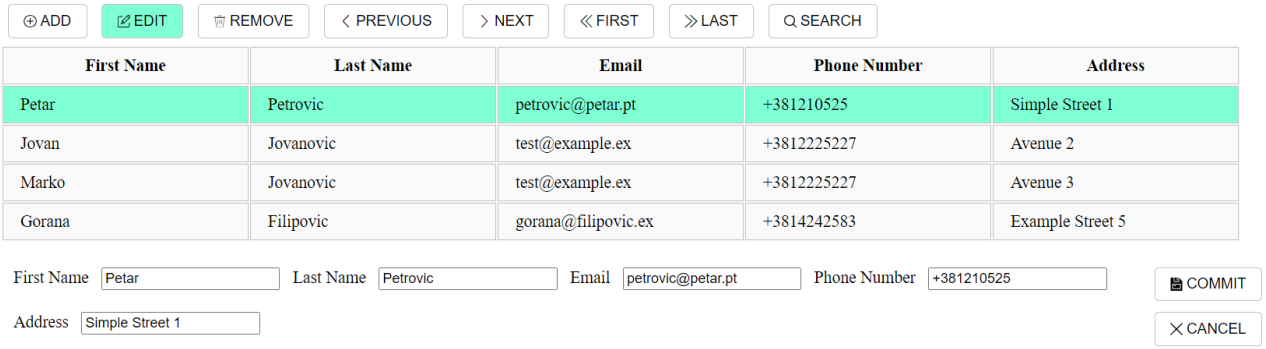
Aplikacija koja je razvijena za potrebe integracije biblioteke za generisanje standardnih formi spaja sloj biznis logike i perzistencioni sloj u jedan iz razloga što ne predstavlja rešenje namenjeno za komercijalnu ili širu upotrebu, već samo za demonstraciju funkcionalnosti biblioteke. Komunikacija između klijenta i servera obavlja se razmenom HTTP zahteva i odgovora. Serverska strana aplikacije implementirana je kao skup RESTful Web Service-a. Klijent i server razmenjuju podatke međusobno u JSON formatu.

4.2. Biblioteka za generisanje standardnih formi

Standardna forma omogućava korisniku da čita, unosi, menja, briše i pretražuje podatke. Sastoji se iz palete alatki, u kojoj korisnik može da izabere režim rada forme, kao i

alatke za jednostavnije i brže kretanje kroz podatke. Mogući režimi rada forme su režim izmene, režim unosa, režim brisanja i režim pretrage. Alatke za kretanje kroz podatke koje standardna forma omogućava su odabir prethodnog elementa, odabir sledećeg elementa, odabir prvog elementa i odabir poslednjeg elementa. Drugi sastavni deo standardne forme jeste tabela koja služi za prikaz podataka. Treći deo čini forma, koja se sastoji od polja za unos i dva dugmeta, dugmeta za potvrdu izmena i dugmeta za odbacivanje izmena. U zavisnosti od režima rada u kom se standardna forma nalazi, forma ima drugačije ponašanje. Ako se forma nalazi u režimu unosa, prilikom odabira režima, sva polja u formi će biti ispražnjena kako bi korisnik mogao lakše da unosi nove podatke. Ako se forma nalazi u režimu rada za izmenu, brisanje ili pretragu, polja za unos biće popunjena podacima trenutno odabranog elementa. Početno stanje forme je režim za izmene. Prikaz početnog stanja standardne forme dat je na slici 2. Na slici se vidi da je forma u režimu izmene i da je odabran prvi element u tabeli. Polja forme popunjena su podacima odabranog elementa.

Implementacija biblioteke za generisanje standardnih formi razlikuje se od već postojećih rešenja kao što je Kendo UI, po tome što uvodi još jedan nivo apstrakcije i time pojednostavljuje upotrebu biblioteke. Odnosno, meta podaci se prosleđuju biblioteci zajedno sa podacima, a dobavljaju se sa servera. Pomoću meta podataka generišu se tabela i forma, čime upotreba biblioteke postaje jednostavnija jer ne moraju eksplicitno da se definišu kolone i oznake.



Slika 2. *Početno stanje standardne forme*

Standardna forma implementirana je tako da se promene čuvaju tek nakon što korisnik potvrdi izmene klikom na dugme za potvrdu izmena. Nakon što korisnik klikne na dugme za potvrdu izmena, podaci iz forme se prosleđuju roditeljskoj komponenti, u ovom slučaju to je StandardFormsComponent. Slanje podataka roditeljskoj komponenti u Angularu je implementirano emitovanjem prilagođenog događaja u koji se smeštaju podaci koje je potrebno poslati.

Roditeljska komponenta osluškuje na prilagođeni događaj i kada ga dete komponenta emituje, roditeljska komponenta reaguje na događaj pozivom metode za obrađivanje tog događaja.

Komponenta StandardFormsComponent potom, na osnovu režima rada u kom se nalazi, emituje odgovarajući događaj svojoj roditeljskoj komponenti, odnosno aplikaciji koja integriše ovu biblioteku. Na ovaj način, implementacija dodavanja, izmene i brisanja prepuštena je sistemu koji integriše biblioteku.

Biblioteka reaguje na svaku promenu podataka od strane aplikacije koja je integriše i u skladu sa novim podacima osvežava komponente.

Korisnik može da odbaci izmene koje je uneo klikom na dugme za odbacivanje izmena. U zavisnosti od režima rada u kom se nalazi standardna forma, događaj odbacivanja izmena izazvaće različito ponašanje. Ako se forma nalazi u režimu unosa podataka, odbacivanje izmena će isprazniti sva polja za unos. Ako se forma nalazi u režimu izmene ili brisanja podataka, odbacivanje izmena će vrednosti u poljima za unos vratiti u stanje u kom su bili pre izmena. Ako se forma nalazi u režimu pretrage podataka, odbacivanje izmena će isprazniti sva polja za unos i poništiti filtere koji su primenjeni nad podacima. Osim izmene stanja polja za unos, nakon što se desi događaj odbacivanja izmena, komponenta Form će emitovati prilagođeni događaj roditeljskoj komponenti StandardFormsComponent. Roditeljska komponenta osluškuje na ovaj događaj i kada se on desi poziva metodu za obradu ovog događaja koja će poništiti primenjene filtere ukoliko je forma bila u režimu rada pretrage podataka.

4.3. Klijentska aplikacija

Sistem u koji je integrisana biblioteka je jednostavan i služi za manipulisanje podacima o korisnicima. Klijentska aplikacija napisana je u Angular-u i omogućava prikaz, dodavanje, izmenu, brisanje i pretragu korisnika.

Klijentska aplikacija komunikacijom sa serverom dobija podatke i meta podatke koje treba da prosledi biblioteci za generisanje standardnih formi. Da bi se ostvarila komunikacija sa serverom, Angular pruža HttpClient servis klasu u sklopu @angular/common/http paketa.

Kako bi klijentska aplikacija integrisala biblioteku za generisanje standardnih formi, neophodno je ubaciti taj modul kao zavisnost u klijentsku aplikaciju. Ubacivanjem StandardFormsModule-a u glavni modul klijentske aplikacije omogućava korišćenje komponenti koje StandardFormsModule izvozi.

Klijentska aplikacija potom vrši vezivanje podataka dobijenih sa servera na podatke koji su potrebni standardnoj formi. Takođe, vezuju se i obrađivači događaja za događaje koje emituje standardna forma.

4.4. Serverska aplikacija

Serverska aplikacija je jednostavna NodeJS aplikacija napisana koristeći ExpressJS. Aplikacija omogućava čitanje, čuvanje, izmenu i brisanje podataka o korisnicima. Sadrži jedan endpoint i to /users kojem je moguće slati zahteve GET, OPTIONS, POST, PUT i DELETE HTTP metodama. Aplikacija očekuje podatke u JSON formatu i za prebacivanje JSON string-a u JavaScript objekat koristi bodyParser. Podaci se mogu čuvati i lako dobavljati iz baza podataka, ali se za potrebe demonstriranja funkcionalnosti čuvaju u memoriji procesa, odnosno ne čuvaju se trajno. Podaci su podeljeni u dva segmenta, jedan koji predstavlja meta podatke i drugi koji predstavlja podatke koji su rezultat upita.

Uloga serverske aplikacije jeste da meta podatke, zajedno sa podacima prebaci u JSON format. Meta podaci se koriste na klijentskoj strani za generisanje standardne forme. Serverska aplikacija registruje 4 metode za obradu HTTP zahteva koji stignu na /users endpoint. Prilikom primanja zahteva na osnovu HTTP metode određuje se koja od tih metoda će obraditi pristigli zahtev. Nakon obrade zahteva, serverska aplikacija šalje klijentskoj aplikaciji HTTP odgovor.

5. ZAKLJUČAK

Zadatak ovog rada bio je da se razvije biblioteka za generisanje standardnih formi. Standardne forme su generične, odnosno ponovno iskoristve za različite entitete u informacionom sistemu, čime je razvoj informacionog sistema znatno ubrzan jer se ne razvija forma za svaki entitet posebno.

Zadatak je implementiran upotrebom Angular radnog okvira za razvoj klijentskih aplikacija, baziran na TypeScript jeziku. Sistem u koji je biblioteka integrisana za potrebe demonstriranja funkcionalnosti razvijen je upotrebom Angular-a i NodeJS-a. Angular je stabilan i moderan radni okvir, koji pruža mnogo funkcionalnosti out-of-the-box, što smanjuje broj 3rd party biblioteka o kojima zavisi aplikacija.

Biblioteka za generisanje standardnih formi dodatno je ubrzala razvoj informacionog sistema iz razloga što se čitav grafički korisnički interfejs i funkcionalnosti mogu dobiti jednostavnom integracijom u sistem, bez potrebe da se forma razvija ispočetka.

Razvojem biblioteke za generisanje standardnih formi omogućeno je da bilo ko može da je integriše u svoj informacioni sistem ukoliko to želi. Prednost ove biblioteke u odnosu na druge biblioteke sličnog tipa, jeste ta da se struktura podataka ne mora definisati eksplicitno, već se može pročitati iz meta podataka dobijenih sa servera, što čini integraciju u sistem jednostavnijom.

Biblioteku za generisanje standardnih formi moguće je dodatno unaprediti dodavanjem prilagođene validacije, čija pravila bi bila definisana u meta podacima. Još jedan vid poboljšanja bi mogao biti da se u meta podacima definišu i relacije između entiteta, kako bi mogla da se implementira Zoom funkcionalnost, odnosno da kada se zumira entitet, da se prikaže detaljno entitet kojeg on referencira.

6. LITERATURA

[1] https://angular.io/guide/architecture (pristupljeno u oktobru 2020.)

[2] https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html (pristupljeno u oktobru 2020.)

Kratka biografija:

Kim Novak rođena je 26.03.1993. godine u Novom Sadu. 2017. godine završila završila je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, smer Računarstvo i automatika i stekla zvanje diplomiranog inženjera elektrotehnike i računarstva. Iste godine na istom fakultetu upisuje master akademske studije smer Primenjene računarske nauke i informatika, modul Softversko inženjerstvo.

Milan Vidaković je rođen u Novom Sadu 1971. godine. Na Fakultetu tehničkih nauka u Novom Sadu završio je doktorske studije 2003. godine. Na istom fakultetu je 2014. godine izabran za redovnog profesora iz oblasti Primenjene računarske nauke i informatika.