Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/02BE38Babic

RAZVOJ INFORMACIONOG SISTEMA ZA AUTO-ŠKOLE

DRIVING SCHOOL INFORMATION SYSTEM DEVELOPMENT

Nina Babić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljena je važnost odabira odgovarajućih tehnologija pri razvoju nekog informacionog sistema, kao i analiza odabranih tehnologija. Osim toga, detaljno je analiziran sam razvijani sistem. Sistem je predstavljen i analiziran kroz prikaz modela sistema, implementiranih klasa i delova koda koji su preuzeti iz implementiranih klasa kao i kroz demonstraciju izgleda, odnosno korisničkog interfejsa razvijane aplikacije.*

Ključne reči: *Informacioni sistem, Spring Boot, JHipster, Maven, Hibernate, Angular*

Abstract – *This thesis represents the importance of the choice of technologies used for implementation of certain system, along with analysis of chosen technologies. Furthermore, detailed analysis of developed system is given. System is represented and analyzed through review of system database model, implemented classes and code parts taken from implemented classes, as well as through demonstration of user interface of developed application .*

Keywords: *Information system, Spring Boot, JHipster, Maven, Hibernate, Angular*

1. UVOD

Uporno se stremi ka tome da se svaki segmet poslovanja ili bilo kog delovanja neke institucije digitalizuje, te je sasvim izvesno da se digitalizuju i segmenti obrazovnog sistema.

Kako i pohađanje nastave (kako teorijske, tako i praktične) za obuku za vožnju, predstavlja neki vid obrazovanja, sasvim je logično da će talas digitalizacije pogoditi i ovu vrstu obrazovnih ustanova.

U ovom radu će biti posvećena pažnja tome koje tehnologije su korišćene za razvoj ovog sistema, jer je autor akcenat stavio na to kako odabrati tehnologije koje uslovno rečeno rade za nas. Drugim rečima, kako odabrati tehnologije u zavisnosti od toga kakav sistem razvijamo, jer su tehnologije upravo tu da nam pomognu, a ne da nam odmognu.

Osim toga, osvrnućemo se na to kakav je model baze podataka za razvijani sistem. Dalje će biti dat detaljniji uvid u implementaciju sistema. Konačno, biće napravljen i pregled implementirane aplikacije, odnosno korisničkog interfejsa.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac.

2. KORIŠĆENE TEHNOLOGIJE

Pre početka razvoja same aplikacije, najbitnija stvar je odabrati odgovarajući set tehnologija pomoću kojih ćemo razvijati istu, jer su upravo tehnologije, te koje rade za nas. One treba da nam olakšaju posao i da nam pomognu da postignemo uspeh, dok bi s druge strane, izbor loših tehnologija vrlo lako mogao biti glavni preduslov za propadanje projekta. U nastavku će biti navedene neke od korišćenih tehnologija pri razvoju informacionog sistema za auto-škole.

2.1. JHipster

JHipster je open source alat koji je vrlo zgodan za korišćenje. Poznato je da je nekad prava muka povezati sve željene tehnologije. Tu na scenu stupa JHipster koji to radi za vas. On predstavlja generator aplikacija koji kreira SpringBoot i Angular aplikaciju. Dakle, dobijate aplikaciju sa Java back-end-om i Angular 6 front-end-om. Drugim rečima, korišćenjem JHipster-a dobićemo takozvanu ljušturu projekta, čime nam je ušteđeno vreme na povezivanju tehnologija, koje sada možemo potrošiti na sam razvoj aplikacije.

2.2. Maven

Maven predstavlja alat za automatizovano bildovanje backend-a, kao i za dependency menadžment, između ostalog. Maven je taj koji nam pomaže da definišemo kako će se neka aplikacija build-ovati (build life cycle), odnosno od čega aplikacija zavisi kako bi joj bilo obezbeđeno to od čega zavisi (dependency management).

2.3. Spring Boot

Spring Boot predstavlja alat koji automatski obavlja sve ono što nam je zadavalo muke. Dakle, vrši automatizovanu konfiguraciju Spring-a kao i konfiguraciju Spring Bean-ova, ali konfiguraciju koju je moguće u svakom trenutku pregaziti sa malim izmenama koje su nama potrebne.

2.4. Hibernate

S obzirom na to da vrlo lako može da dođe do neslaganja između objektnog i relacionog modela sistema u smislu predstave podataka u jednim i drugim, potreban nam je alat koji će nam skratiti vreme razvoja samog sistema, time što će olakšati objektno-relaciono mapiranje. Upravo iz tog razloga nam je potreban Hibernate alat koji, između ostalog služi za mapiranje objektnog u relacioni model i primenu objektno-orijentisanih koncepata u relacionim bazama podataka.

2.5. Angular

Angular predstavlja TypeScript baziranu platformu koja je tu da nam olakša razvoj front-end dela naše web aplikacije. Karakteristično za Angular jeste to što predstavlja MVC framework, što samim developerima omogućava da primene MVC strukturu na klijentsku stranu aplikacije koju razvijaju.

3. MODEL BAZE PODATAKA

Baza podataka je neizostavna komponenta svakog informacionog sistema, bez obzira na to da li je u pitanju sistem sa kompleksnom biznis logikom, ili se radi relativno jednostavnom sistemu.

S obzirom na to da je reč o za sada nekomplikovanom sistemu, model baze za auto-školu se sastoji od četiri, uslovno rečeno, obične tabele, kao i od tri vezne tabele. Sama baza podataka je kreirana u MySql-u.

Svaka od tabela ima svoj primarni ključ, dok neke tabele sadrže i strani ključ drugih tabela, jer su na neki način povezane.

Osim toga, kao što je već napomenuto postoje i tri vezne tabele, koje predstavljaju vezu više-prema-više, odnosno Many-to-many relaciju.

Model baze podataka za sistema auto-škola se sastoji iz sledećih tabela:

1. instruktor
2. ucenik
3. vozilo
4. raspored
5. raspored_ucenik
6. raspored_vozilo
7. raspored_instruktor

4. IMPLEMENTACIJA

U nastavku će biti navedene neke od klasa implementiranog sistema, sa prikazom delova koda nekih od njih.

4.1. Bean

U aplikaciji imamo nekoliko entiteta, to su učenik, instruktor, vozilo i raspored. Svaki od tih entiteta ima klasu, odnosno bean koji ga predstavlja. Bitno je napomenuti da je svaki od bean-ova anotiran anotacijom @Entity, koja ističe da je dati objekat, odnosno datu klasu moguće mapirati na tabelu u bazi podataka.

4.2. Controller

Kontroleri, spadaju u prezentacioni sloj aplikacije. Kada korisnik pošalje neki zahtev, odnosno odluči da na primer ažurira neki entitet, taj zahtev prihvata upravo kontroler. Drugim rečima, kontroleri rukuju Http request-ovima.

U kontroleru se nalaze osnovne CRUD (Create, Read, Update, Delete) operacije, međutim sama logika nije u njima. Kontroleri nam između ostalog služe da obave proveru da li su zahtevi klijenta validni.

Ukoliko određeni zahtev nije validan, kontroler će poslati odgovor u vidu greške. Ako zahtev klijenta zaista jeste validan, kontroler gađa servisnu klasu u kojoj se u stvari nalazi sva logika za operaciju za koju je klijent poslao zahtev. Svaki od kontrolera je anotiran anotacijom @RestController, kako bi taj kontroler postao vidljiv Spring kontejneru.

4.3. Service

Sva biznis logika aplikacije smeštena je upravo u servisne klase. Servisne klase su korišćene kako bi se jasno odvojilo ono što ima veze sa web delom aplikacije, od onoga što predstavlja neku uopštenu biznis logiku.
Servisne klase označene su @Service anotacijom, iz istog razloga iz kog su kontroleri bili označeni @Controller anotacijom.

4.4. Module

Dok su prethodno navedene klase pripadale back-end delu aplikacije, moduli, kao i klase koje će biti navedene u nastavku pripadaju front-end delu aplikacije.

Module (modul) predstavlja kontejner koji sadrži različite komponente razvijanog sistema (servise, direktive i sl.). Svaki od entiteta (u kontekstu Angular-a svaki entitet predstavlja komponentu), ima svoj modul u kom su navedene komponente od kojih ta komponenta (npr. komponenta učenik) zavisi. Osim toga svaki modul sadrži deklarisanje komponenti koje mu pripadaju.

4.5. Model

Kao i na back-end-u, tako je i na front-end-u potrebno da imamo klasu koja će predstavljati određeni entitet. Na front-end-u je to tzv. Model. Model predstavlja reprezentaciju DTO-a (Data Transfer Object) kog dobavljamo sa back-end-a.

4.6. Service

Kako bi se izbeglo da komponente u Angular-u budu te koje manipulišu podacima, jer to definitvno nije dobra praksa, uvedeni su servisi. Uvođenjem servisa, omogućeno je da komponente budu čiste i efikasne.

Ono što je bitno napomenuti, jeste upotreba HttpClient-a u svakom od implementiranih servisa. HttpClient predstavlja klijentski HTTP API za Angular aplikacije. Upravo njega koristimo kako bismo poslali određeni http zahtev, bio to GET, POST, PUT, DELETE ili sl. Na slici 1 prikazana je metoda update, koja šalje http zahtev PUT, kako bi se ažurirali podaci određenog učenika.

```
update(ucenik: IUcenik): Observable<EntityResponseType> {
    const copy = this.convertDateFromClient(ucenik);
    return this.http
        .put<IUcenik>(this.resourceUrl, copy, { observe: 'response' })
        .pipe(map((res: EntityResponseType) => this.convertDateFromServer(res)));
}
```

Slika 1. *Prikaz metode za ažuriranje podataka učenika*

4.7. Component

Kada pričamo o komponentama, možemo slobodno reći da je sve u Angularu neki vid komponente. Na komponente možemo gledati kao osnovne jedinice građe nekog modula.

Svaka od komponenti, označena je @Component dekoratorom. Funkcija ovog dekoratora jeste da obavesti Angular da se radi o komponenti, osim toga za ovaj dekorator su vezani i određeni metapodaci, koji Angularu mogu da kažu mnogo toga, kao što je npr. putanja do template-a ove komponente.

4.8. Template

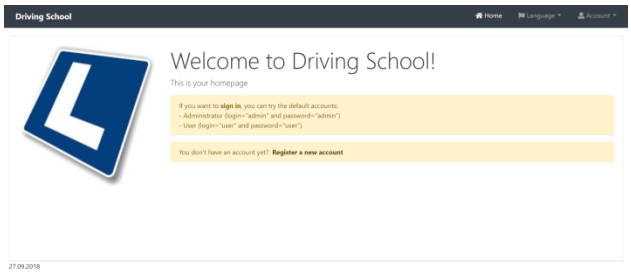
Dok smo implementacijom komponente definisali ono šta korisnik može da radi, implementacijom template-a defi-

nišemo ono šta je korisniku vidljivo. Jezik koji koristimo za implementaciju template-a u Angular-u jeste HTML.

U okvirima template-a, skoro sva HTML sintaksa je validna. Osim toga, rečnik HTML-a template-a možemo obogatiti našim direktivama i komponentama, koje se javljaju kao novi elementi ili atributi.

5. PRIKAZ APLIKACIJE

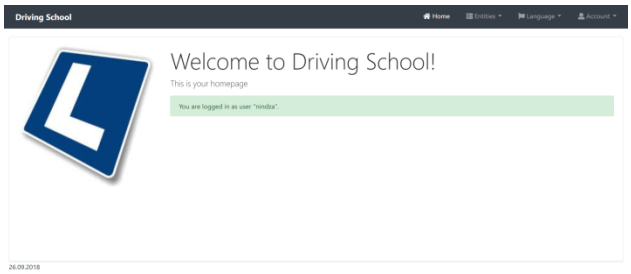
Nakon što je korisnik pristupio aplikaciji, ono što mu se inicijalno prikazuje jeste home stranica. Na home stranici korisnik može da odabere željeni jezik, pre logovanja u aplikaciju. Na slici 2 je moguće videti kako izgleda home stranica razvijane aplikacije.



Slika 2. *Početna stranica pre prijave korisnika*

Kako bi se korisnik ulogovao u aplikaciju, potrebno je da unese svoje kredencijale, odnosno korisničko ime i lozinku.

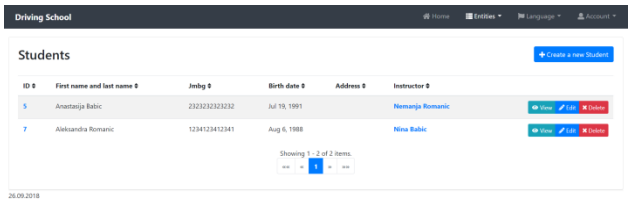
Po uspešnom prijavljivanju na sistem, korisniku se prikazuje početna stranica, koju možete videti na slici 3.



Slika 3. *Početna stranica nakon uspešne prijave korisnika*

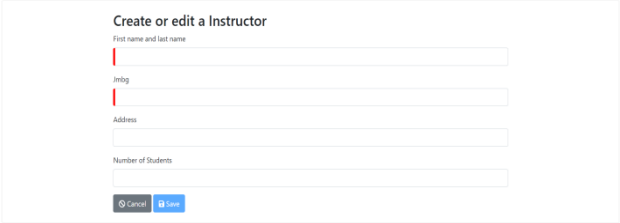
U meniju je korisniku ponuđen Home, koji vodi na početnu stranicu. Ono što korisnik još može da vidi u meniju, između ostalog jesu entiteti u aplikaciji. Osim toga korisnik u svakom trenutku može da promeni jezik sistema. Mogući jezici su engleski i srpski jezik. Takođe, ulogovani korisnik može da upravlja svojim nalogom, moguće funkcionalnosti se nalaze u stavci menija pod nazivom Account.

Kao što je ranije navedeno, entiteti u sistemu su instruktori, učenici, vozila i rasporedi. Na slici 4 moguće je videti kako izgleda stranica koja predstavlja entitet učenik.



Slika 4. *Entitet učenik*

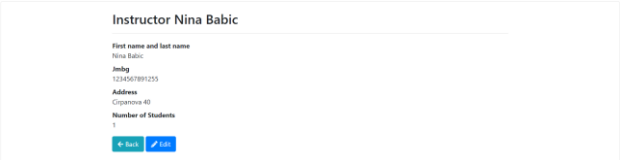
Svaki od entiteta je moguće izmeniti (Edit), videti detalje istog (View), kao i obrisati ga (Delete). Osim toga, korisnik u svakom trenutku može da doda novog instruktora, učenika, vozilo i raspored (Create new ...). Na slici 5 je moguće videti kako izgleda forma za unos novog instruktora.



Slika 5. *Forma za kreiranje novog instruktora*

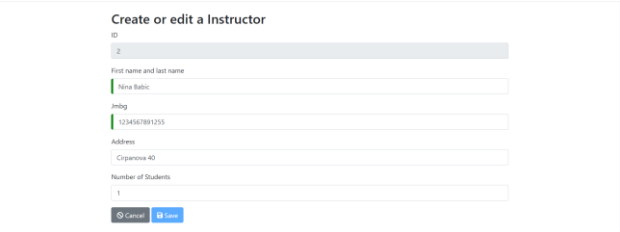
Kao što se vidi na slici 5, neka polja imaju indikator crvene boje, dok druga nemaju. Ti indikatori predstavljaju vid validacije na front-end-u. U ovom slučaju je to validacija za obaveznost polja. To nam govori da sve dok korisnik ne unese podatke za sva obavezna polja, on neće biti u mogućnosti da submit-uje formu, odnosno neće moći da kreira novog Instruktora.

Kao što je prethodno napomenuto, moguće je videti detalje svakog entiteta. Forma za detalje služi isključivo za čitanje, odnosno u njoj nije moguće menjati vrednosti koje su unete za datog Instruktora. Međutim, klikom na dugme Edit u formi za detalje, moguće je preusmeriti se na formu za ažuriranje podataka o Instruktoru. Kako izgleda forma za detalje, moguće je videti na slici 6.



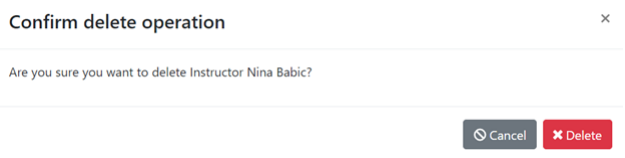
Slika 6. *Forma za pregled detalja instruktora*

Formu za izmenu podataka o instruktoru možete videti na slici 7. Ova forma izgleda veoma slično formi za unos novog instruktora. U implementaciji one predstavljaju istu formu, sa određenim uslovima vezanim za to šta u kojoj formi treba da bude vidljivo korisniku, te ih upravo iz tog razloga korisnik ne vidi u istom formatu.



Slika 7. *Forma za ažuriranje podataka instruktora*

Kao što je već navedeno, korisnik može da obriše određeni entitet. Kada korisnik pozove akciju brisanja klikom na dugme Delete koje je moguće videti na slici 4, otvara se pop-up dijalog (slika 8).



Slika 8. *Potvrda za brisanje Instruktora*

Dijalog koji je prikazan na slici 8, namenjen je za to da korisnik potvrdi da zaista želi da obriše određeni entitet. Ukoliko korisnik zaista želi da obriše odabrani entiet, potrebno je da klikne na dugme Delete. Međutim, ukoliko želi da odustane, potrebno je da klikne na dugme Cancel.

6. ZAKLJUČAK

Ovim radom je obuhvaćen proces od odluke koje tehnologije koristiti do toga kako je sam sistem implementiran. Sama razvijana aplikacija predstavlja samo začetak ideje o digitalizaciji sistema neke auto-škole. Za sada je namenjena samo instruktorima i administrativnim radnicima zaposlenim u datoj auto-školi. Međutim, autor ima razvijenu ideju kako taj sistem unaprediti, kao i šta je potrebno uraditi kako bi sama aplikacija mogla da donese koristi svim potencijalnim članovima jedne auto-škole, od učenika, preko instruktora, administrativnih radnika, do menadžmenta iste.

Kada se govori o učenicima, ideja je da se aplikacija proširi posebno dizajniranim rasporedom samo za njih, koji će obuhvatati detaljni nedeljni raspored prvo teorijske, a zatim i praktične nastave. Osim toga, postoji i ideja da se sistem proširi i testovima koji će olakšati učenicima da se spreme za polaganje teorijskog ispita. Sam entitet učenik bi mogao da se proširi još jednim poljem koje bi označavalo broj izlazaka na teorijske ispite, kao i poljem koje bi označavalo i broj izlazaka na teren kako bi se polagao praktičan ispit vožnje. Dakle, konačna ideja je da učenici imaju poseban korisnički nalog, putem kog bi videli elemente aplikacije koje su njima od interesa.

Kada je reč o instruktorima, takođe postoji ideja da se napravi optimizovaniji raspored nastave, u odnosu na impelementirani, kako bi im bilo omogućeno još jednostavnije praćenje obaveza koje imaju.

Osim toga, moguće je sistem proširiti i izveštajima, koji bi u mnogome olakšali posao administrativnim radnicima zaposlenim u auto-školi, te bi ih tako jedan klik delio od toga da dobiju uvid u npr. prošlomesečno zauzeće nekog instruktora i sl.

Korišćenjem ovog sistema bi se drastično smanjila mogućnost oko pogrešnog ugovaranja termina nastave, ili prezauzetosti određenog instruktora, dok drugi instruktor nema u nadležnosti nijednog učenika. Osim toga mogao bi da se prati status svakog učenika uvidom u to koliko puta je izašao na teorijski odnosno praktičan ispit. Zatim bi se olakšano pratila potreba za npr. proširivanjem voznog parka, ili nastavničkog osoblja. Te bi to predstavljalo odličan primer korišćenja tehnologija tako da nam olakšaju svakodnevicu.

7. LITERATURA

[1] *About JHipster,* https://www.jhipster.tech/ (pristupljeno u septembru 2018.)

[2] *What is Maven,* https://maven.apache.org/what-is-maven.html (pristupljeno u septembru 2018.)

[3] *Spring Boot,* https://spring.io/projects/spring-boot (pristupljeno u septembru 2018.)

[4] *What is Hibernate* https://www.linkedin.com/learning/java-database-access-with-hibernate/what-is-hibernate (pristupljeno u septembru 2018.)

[5] *What is Angular* https://angular.io/docs (pristupljeno u sept. 2018.)

Kratka biografija:

Nina Babić rođena je u Bačkoj Topoli 23.10.1991. Školske 2010/2011 je upisala Fakultet tehničkih nauka u Novom Sadu. Zvanje diplomirani inženjer elektrotehnike i računarstva stekla je 2015. godine. Položila je sve predmete predviđene planom i programom na master akademskim studijama na odseku: Primenjeno softversko inženjerstvo.