Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42
DOI: https://doi.org/10.24867/28BE42Vrbica

IMPLEMENTACIJA MIKROSERVISNE ARHITEKTURE KROZ VEB APLIKACIJU ZA PRODAJU AUTOMOBILA

IMPLEMENTING A MICROSERVICE ARCHITECTURE THROUGH A CAR SALES WEB APPLICATION

Vladimir Vrbica, Milan Vidaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad se bavi realizacijom aplikacije za prodaju automobila upotrebom mikroservisne arhitekture. Ova aplikacija je prvobitno implementirana upotrebom monolitne arhitekture. Klijentski deo aplikacije je razvijen uz pomoć Angular alata, uz dodatak Bootstrap biblioteke. Serverski deo aplikacije je razvijen u programskom jeziku Java, korišćenjem Spring okruženja. Aplikacija je deploy-ovana putem Docker alata, dok je u aplikaciju integrisan i Stripe sistem za plaćanje.*

Ključne reči: *Java, ActiveMQ, Docker, Rest, Stripe, mikroservisi*

Abstract – *This paper demonstrates a microservice architecture that was implemented through a car sales web application. This application was originally created as a monolithic one. The client part of the application was developed using Angular tools, with the addition of the Bootstrap library. The server part of the application was developed in the Java programming language, using the Spring environment. The application is deployed using the Docker tool, while the Stripe payment system is also integrated into the application.*

Keywords: *Java, ActiveMQ, Docker, Rest, Stripe, microservices*

1. UVOD

U ovom radu biće opisana implementacija mikroservisne arhitekture [1] kroz veb aplikaciju za prodaju automobila. Prvobitna monolitna aplikacija je unapređena u mikroservisnu tako što je implementirana po uzoru na sisteme koji će biti predstavljeni u drugom odeljku. Kada je reč o aplikaciji, klijentski deo je Angular aplikacija, dok je serverski deo Java aplikacija u Spring okruženju.

2. PREGLED SLIČNIH SISTEMA

Inspiracija za odabir ovakvog tipa aplikacije pronađena je u sajtovima Polovni automobili [2] i Mobile.de [3]. Sajtovi su međusobno slični i predstavljaju najpopularnije i najposećenije sajtove za prodaju automobila. Imajući u vidu sličnost, jednako se prikazuju sve neophodne usluge i informacije na jednom sajtu za prodaju automobila. Neke od tih usluga jesu oglašavanje automobila za prodaju putem oglasa, pretraživanje oglasa, uvid u prethodna iskustva drugih kupaca, postupak registracije vozila i slično.

3. KORIŠĆENE TEHNOLOGIJE

Za implementaciju klijentskog dela aplikacije korišćen je Angular [4] alat uz dodatak Bootstrap [5] biblioteke, dok je serverski deo aplikacije napisan u programskom jeziku Java [6], korišćenjem Spring [7] okruženja.

Spring

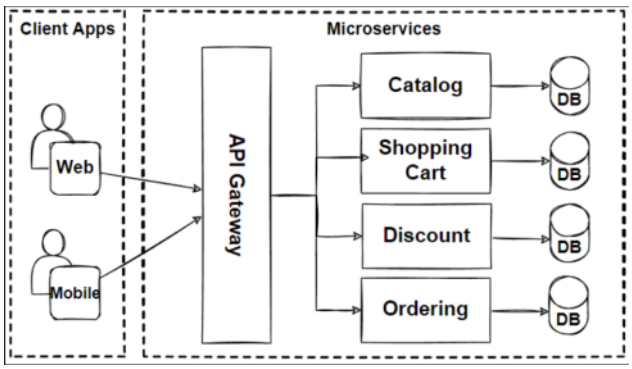
Spring je okruženje koje služi za jednostavniji razvoj poslovnih aplikacija u Javi. Trenutno je najpopularnije i najuticajnije okruženje za razvoj Java veb aplikacija. Nudi podršku za dependency injection mehanizam, veb aplikacije, rad sa podacima iz baze podataka, razmenu poruka itd. Uključuje REST arhitekturu, klijent-server arhitekturu koja koristi HTTP komunikacioni protokol i nudi svoje metode GET, POST, PUT, DELETE.

Angular

Angular je razvojna platforma bazirana na TypeScript-u. Uključuje okvir zasnovan na komponentama za izgradnju veb aplikacija koristeći HTML. Komponente definišu prikaze koji su zapravo skupovi elemenata koje Angular može da bira i menja u skladu sa programskom logikom.

Mikroservisna arhitektura

Mikroservisna arhitektura podrazumeva postojanje većeg broja manjih servisa koji su međusobno slabo povezani, slika 1.



Slika 1 – *Mikroservisna arhitektura*

Svaki servis ima svoju bazu podataka, svoje servise koji manipulišu tim podacima, kao i svoje kontrolere koji

---

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

predstavljaju pristupne tačke svakog servisa. Zahvaljujući tome, servisi mogu da se implementiraju kao nezavisne, manje aplikacije, koje mogu biti napisane različitim jezicima i u različitim okruženjima.

Docker

Docker [8] je otvorena platforma za razvoj i pokretanje aplikacija. Pruža mogućnost pakovanja aplikacije u labavo izolovanom okruženju koje se zove kontejner. Moguće je istovremeno pokrenuti više kontejnera na datom hostu. Svaki projekat ima svoj Dockerfile, dok se za pokretanje više projekata u zasebnim kontejnerima kreira docker-compose.yml fajl.

```
user:
    container_name: user
    build:
      context: UserApplication
      dockerfile: Dockerfile
    environment:
      - DB_URL=jdbc:postgresql://db1:5432/user-
service
      - DB_USERNAME=postgres
      - DB_PASSWORD=postgres
    ports:
      - "8081:8081"
    depends_on:
      db1:
        condition: service_started
        restart: true
```

Listing 2 – *Primer kreiranja kontejnera user*

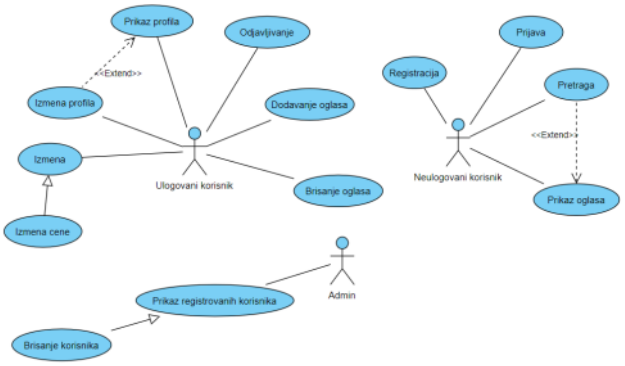
Stripe

Stripe jeste provajder platnih usluga koji omogućava trgovcima da prihvate kreditne i debitne kartice ili druga plaćanja. Njegovo rešenje za obradu plaćanja je najpogodnije za preduzeća koja većinu svoje prodaje ostvaruju putem mreže, odnosno za kompanije koje se bave onlajn prodajom. Podržava plaćanje u različitim valutama i prihvata plaćanje različitim karticama.

4. SPECIFIKACIJA

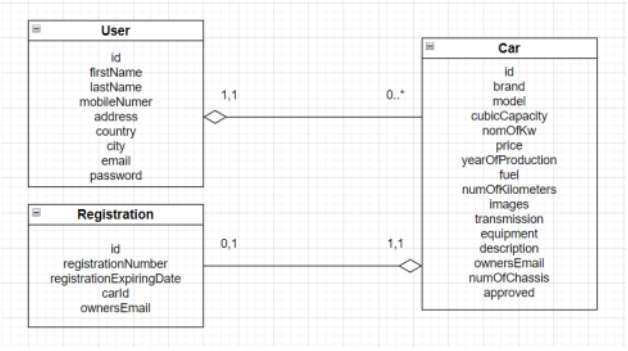
Zadatak samog rada jeste refaktorisanje monolitne veb aplikacije za prodaju automobila, kreiranje više manjih servisa po uzoru na mikroservisnu arhitekturu. Između servisa uspostavljena je sinhrona (REST) odnosno asinhrona (ActiveMQ) komunikacija. Zahtevi koje klijent šalje ka serveru gađaju API Gateway, odakle se zahtevi prosleđuju odgovarajućim mikroservisima. Aplikacija je deploy-ovana korišćenjem Docker platforme. Svi servisi, zajedno sa odgovarajućim bazama podataka, smešteni su u kontejnere i organizovani u Docker okruženju. Uz aplikaciju je integrisana i Stripe platforma koja omogućava bezbedan i siguran vid plaćanja karticom putem interneta.

Aplikacija reprezentuje sajt za prodaju automobila. Korisnicima nudi mogućnosti oglašavanja automobila na prodaju, pretraživanja oglasa kao i ažuriranja postojećih oglasa. Na početnoj stranici je omogućeno višestruko pretraživanje postojećih oglasa, kao i prijava odnosno registrovanje na sistem. Prijavljeni korisnik ima mogućnost objavljivanja novog oglasa. Kada popuni obavezna (marka, model, godina proizvodnje..) i opciona polja, oglas se šalje administratoru sistema koji može da odobri oglas, nakon čega će on biti vidljiv u sistemu, ili da ga odbije. Pored prikaza pretraživanih automobila, korisnik ima mogućnost pregleda profila svakog automobila, gde su prikazane osnovne informacije o automobilu, slike, opisi kao i podaci o kontaktu prodavca. Opisane usluge prikazane su na dijagramu slučajeva korišćenja prikazanom na slici 3.



Slika 3 – *Dijagram slučajeva korišćenja*

Sam domen aplikacije je veoma jednostavan i sastoji se iz tri klase koje su karakteristične za ovaj sistem, a to su Car, User i Registration. Kao što im samo ime kaže, Car klasa se odnosi na automobile i sadrži polja koja predstavljaju sve informacije koje se nalaze na jednom oglasu. Za nju je vezana klasa Registration koja reprezentuje registracionu oznaku automobila. Oznaka je vezana za samo jedan automobil, ali ona ne mora uvek da bude navedena. Poslednja, User klasa sadrži informacije o korisniku. Slika 4 prikazuje opisani dijagram klasa.



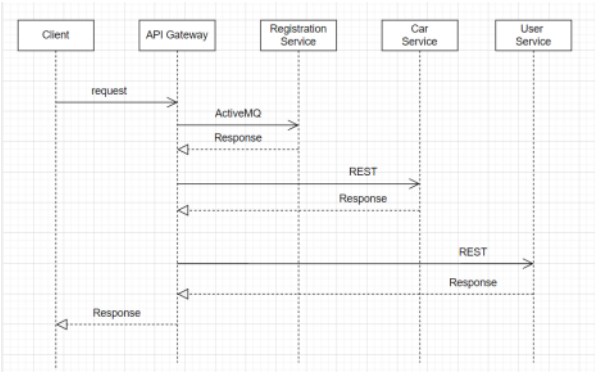
Slika 4 – *Dijagram klasa*

Na slici 5 prikazan je dijagram sekvenci koji predstavlja arhitekturu aplikacije. Zahtev koji stiže sa klijentske strane, pogađa prvo API Gateway odakle se, analizom pristiglog zahteva, on prosleđuje ka odgovarajućem mikroservisu odakle se nakon obrade vraća odgovor.

5. IMPLEMENTACIJA

Aplikaciju čine dve celine, klijentski i serverski deo. Serverski deo implementiran je u programskom jeziku Java, uz pomoć Spring okruženja, dok je klijentski deo aplikacije implementiran pomoću Angular alata uz

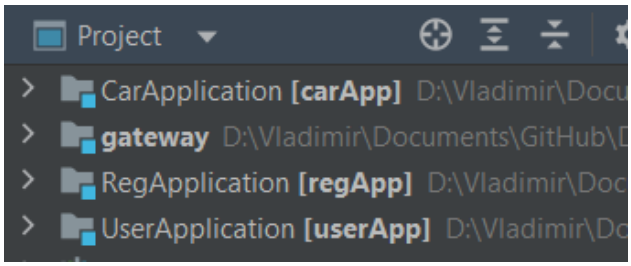
dodatak Bootstrap biblioteke. Serverski deo je mikroservisno orijentisan i deploy-ovan pomoću Docker alata. Takođe, uz aplikaciju je integrisan i sigurnosni sistem za plaćanje pod imenom Stripe.



Slika 5 – Dijagram sekvenci

Serverski deo

Serverski deo aplikacije, kao što je već napomenuto, je mikroservisno orijentisan i sastoji se iz četiri projekta, a to su Gateway, CarApplication, UserApplication i RegistrationApplication (slika 6). Aplikacije mežusobno komuniciraju sinhronim (REST) odnosno asinhronim putem ActiveMQ.



Slika 6 – *Struktura aplikacije*

Zahtevi koji pristužu iz klijentskog dela aplikacije pogađaju Gateway kontroler, odakle se nakon obrade prosleđuju odgovarajućim servisima na dalju obradu. Kako je aplikacija deploy-ovana putem Docker-a, svaki projekat sadrži svoj Dockerfile u kom se navodi jasna konfiguracija projekta koja će se preneti na Docker image. Da bi se aplikacija uspešno pokrenula, sva četiri projekta su objedinjena unutar docker-compose.yml fajla uz pomoć kog se kreiraju kontejneri koji će pokrenuti svaki servis zasebno.

Svaki servis ima svoj kontroler, servis i repozitorijum, kao i svoju bazu podataka. Kontroleri predstavljaju pristupnu tačku svakog mikroservisa, oni primaju zahteve, obrađuju ih i prosleđuju servisima. Dalje, servisi komuniciraju sa repozitorijumima koji dobavljaju podatke ili ažuriraju bazu podataka.

Nakon toga, odgovor se vraća na istu lokaciju odakle je zahtev došao. Metode unutar kontrolera anotirane su, u zavisnosti od tipa zahteva koji obrađuju, sa @GetMapping, @PostMapping ili @PutMapping, gde se u zagradi navodi URL putanja na koju treba zahtev proslediti. Metodama, anotiranim pomenutim anotacijama, se parametri prosleđuju kroz putanju (@PathVariable), zahtev (@RequestBody) ili URL (@RequestParam).

Stripe server

Kao što je već pomenuto, uz aplikaciju je integrisan i sigurnosni sistem za plaćanje pod imenom Stripe. Podaci o kartici putem koje se vrši plaćanje, šalju se u okviru tokena sa klijentske strane ka serveru, kako bi se plaćanje evidentiralo. Tajni ključ obezbeđuje da se plaćanje izvrši što bezbednijim putem, i on je deklarisan na serverskoj strani. Na listingu 7 prikazan je način evidentiranja uspešnog plaćanja.

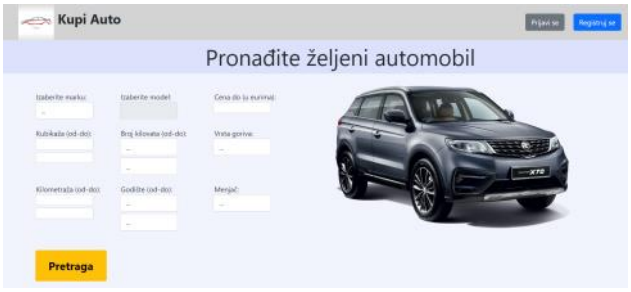
```
@Value("${api.stripe.sk}")
private String secretKey;

@PostConstruct
public void init() {
    Stripe.apiKey = secretKey;
}

public StripeChargeDTO
chargePayment(StripeChargeDTO
chargeRequest) throws StripeException {
    chargeRequest.setSuccess(false);
    Map<String, Object> chargeParams =
new HashMap<>();
    chargeParams.put("amount", (int)
(chargeRequest.getAmount() * 100));
    chargeParams.put("currency",
"usd");
    chargeParams.put("source",
chargeRequest.getStripeToken());

    Charge charge =
Charge.create(chargeParams);

    if(charge.getPaid()){

chargeRequest.setChargeId(charge.getId(
));
        chargeRequest.setSuccess(true);
    }

    return chargeRequest;
}
```

Listing 7 – *Evidentiranje plaćanja*

Klijentski deo

Klijentski deo aplikacije čini view komponenta u arhitekturi aplikacije. Zadatak ove komponente jeste prikaz podataka korisniku. Različite stranice definisane su u okviru komponenti, uz pomoć .html, .css i .ts fajlova, gde se kreira sam sadržaj stranice, kao i pozadinska, programska logika.

Početna stranica aplikacije nudi mogućnost pretraživanja oglasa, kao i prijavu odnosno registrovanje novog korisnika na sistem (slika 8).



Slika 8 – Početna stranica

Pretraga se može vršiti po jednom ili više parametara. U slučaju da korisnik ne navede parametre, prikazuju se svi dostupni oglasi na sajtu.

Korisniku se prikazuju rezultati pretrage na drugoj stranici, gde je moguće otvoriti svaki oglas ponaosob. Profil oglasa nudi informacije o automobilu, kao i o kontaktu prodavca. Takođe, ukoliko ih poseduje još, korisnik može da izlista sve registrovane automobile postavljene od strane tog kupca.

Dodavanje novog oglasa je moguće od strane registrovanog korisnika, i ono podrazumeva da se osnovne informacije o automobilu (marka, model, kubikaža..) obavezno navedu, dok su ostale informacije opcione.

Stripe biblioteka

U okviru index.html stranice, u klijentski deo aplikacije ubačena je i Stripe.js biblioteka koja nudi formu za onlajn plaćanje. Kako se svaki oglas naplaćuje, završni korak ka uspešnom registrovanju novog oglasa na sistem jeste popunjavanje forme podacima o kartici. Nakon popunjavanja svih polja, podaci se pakuju u token, i uz javni ključ šalju ka serverskom delu aplikacije gde se plaćanje i evidentira.

6. ZAKLJUČAK

Prvobitno kreirana aplikacija predstavljala je monolitnu aplikaciju za prodaju automobila implementiranu po uzoru na sisteme opisane u drugoj sekciji. Za potrebe rada, aplikacija je refaktorisana i organizovana kao skup mikroservisa koji međusobno komuniciraju. Mikroservisna arhitektura implementirana je sa ciljem da ukloni svaku zavisnost između modula aplikacije. Kao takva, omogućava nezavisno implementiranje, razvoj kao i testiranje, u različitim jezicima i okruženjima. Pored mikroservisne arhitekture, aplikacija je deploy-ovana putem Docker-a, a uz aplikaciju je integrisan i sigurnosni sistem za plaćanje pod nazivom Stripe.

Iz svega pomenutog, može se zaključiti da je projekat razvijan u tehnologijama koje su u današnje doba veoma popularne i sklone unapređivanju. Samim tim, i aplikacija je sklona daljem unapređenju i usavršavanju.

Kombinacija mikroservisne arhitekture i Docker-a predstavlja jedan čvrst i stabilan temelj za budući rast i razvoj i omogućava aplikaciji da se lako prilagođava promenama i zahtevima tržišta.

7. LITERATURA

[1] Mikroservisna arhitektura – *https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices*

[2] Polovni automobili - *https://www.polovniautomobili.com/*

[3] Mobile.de – *https://www.mobile.de/*

[4] Angular – *https://angular.io/*

[5] Bootstrap – *https://getbootstrap.com/*

[6] Java – *https://docs.oracle.com/javase/8/docs/api/*

[7] Spring – *https://spring.io/*

[8] Docker - *https://docs.docker.com/get-started/overview/*

Kratka biografija:

Vladimir Vrbica rođen je 24.01.2000. u Subotici. Završio je Osnovnu školu "Sonja Marinković" u Subotici, kao i Gimnaziju "Svetozar Marković". Završio je osnovne akademske studije na Fakultetu tehničkih nauka 2022. godine, smer Računarstvo i automatika. Nakon završenih osnovnih studija upisao je master studije na istom fakultetu, smer Računarstvo i automatika, modul Elektronsko poslovanje. Položio je sve ispite predviđene planom i programom.

Milan Vidaković rođen je 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*.