Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/22BE10Jevtic

IMPLEMENTACIJA PERFORMANTNOG SISTEMA POMOĆU MIKROSERVISA

IMPLEMENTATION OF PERFORMANT SYSTEM USING MICROSERVICES

Stanko Jevtić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Ovaj rad bavi se opisivanjem **.NET Core** radnog okvira kao i implementacijom performantnog sistema pomoću mikroservisa.*

Ključne reči: ***API,*** *mikroservisi,* ***.NET Core****, logistika*

Abstract – *Describing the **.NET Core** framework and implementation of performant system using microservices.*

Keywords: *API, microservices, .NET Core, logistics*

1. UVOD

Tema ovog rada jeste da analizira ***.NET Core*** frejmvork, kao i opis implementacije zadatka koji na frontendu koristi ***React*** i ***Android,*** dok na bekendu ***.NET 5*** u cilju analize performanse sistema.

Zadatak koji je implementiran i opisan u ovom radu uz pomoć pomenutih radnih okvira jeste sistem za opsluživanje paketa i kontejnera. Kada se kaže opsluživanje, tu se misli na kreiranje porudžbina, generisanje labela za te pakete, praćenje statusa paketa, dostavljanja izveštaja u različitim formatima (***EDI214***, ***CSV***), skeniranje paketa i njihovo sortiranje unutar centra za sortiranje.

Sort centri su skladišta koja se nalaze u Sjedinjenim Američkim Državama, svrha im je da prihvataju pakete i šalju ih dalje ka Američkoj pošti.

Komunikacija između frontend i bekend dela ove aplikacije se vrši preko ***HTTP*** protokola, koristeći ***REST*** arhitekturu.

2. *.NET CORE* radni okvir

.NET Core frejmvork [1] je platforma za izvršavanje aplikacija na ***Windows***, ***macOS*** i ***Linux*** operativnim sistemima.
Glavna namena je implementacija različitih tipova aplikacija kao što su ***mobile***, ***desktop***, ***web***, ***cloud***, ***IoT***, ***machine learning***, ***microservice*** i ***gaming*** aplikacije kao što je prikazano na slici 1.

3. SPECIFIKACIJA SISTEMA

Cilj aplikacije jeste logistika i vođenje evidencije o paketima koji pristižu u sort centar, njihovo slanje na sledeću destinaciju, kao i generisanje labela za pakete i kontejnere.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Željko Vuković.

Slika 1. *Funkcionalnosti i infrastruktura **.NET Core** frejmvorka [2]*

Dodatno, potrebno je implementirati portal koji će obaveštavati krajnje korisnike o statusima paketa.

Ideja samog projekta je da se ubrza dostavljanje paketa u Sjedinjenim Američkim Državama korišćenjem pametnog algoritma za sortiranje paketa po zip kodu.

API koji je implementiran integrisan je sa šiperima koji imaju namenu da dostave pakete na adresu krajnjih korisnika. Sama aplikacija je integrisana sa američkom poštom (***USPS – United States Postal Service***) kao i ***SmartyStreets*** provajderom koji omogućava verifikaciju adresa.

3.1. Entiteti

Na slici 2. prikazan je dijagram klasa sistema za upravljanje paketima.

4. IMPLEMENTACIJA BEKEND DELA SISTEMA

U ovoj sekciji će biti opisana implementacija bekend dela sistema za opsluživanje paketa, ***design pattern***-i korišćeni prilikom formiranja arhitekture, kao i ***clean code*** prakse koje su primenjene u implementaciji.

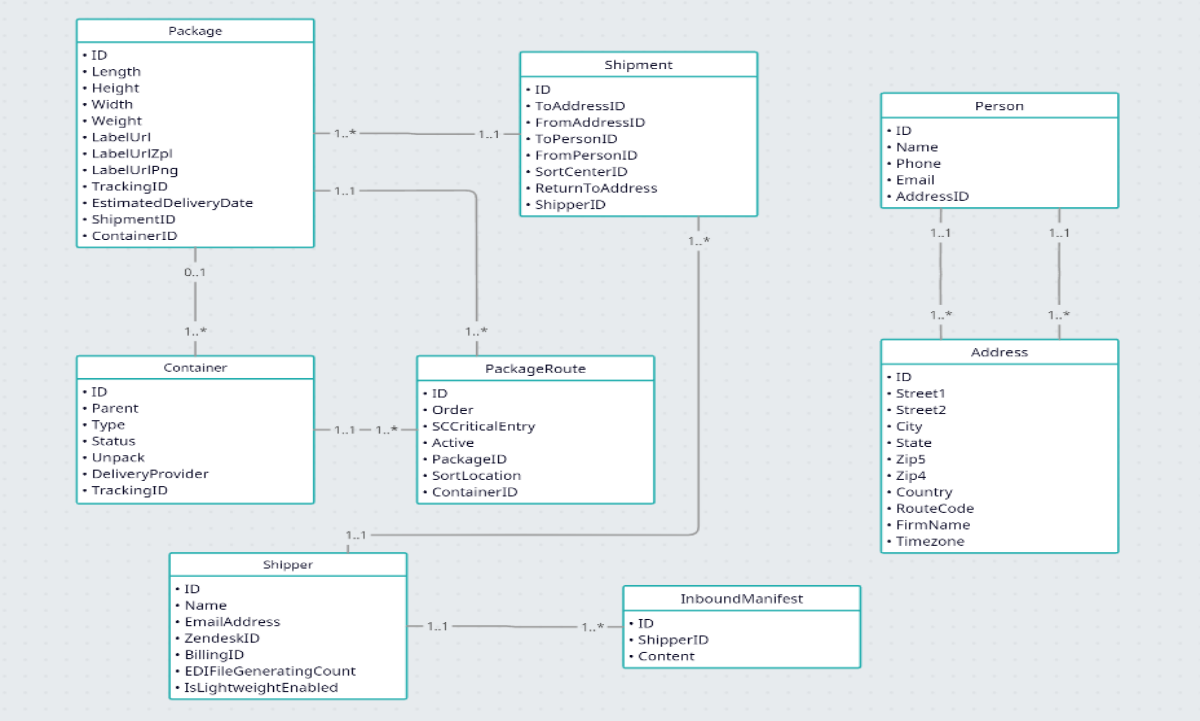
4.1. Arhitektura bekend aplikacije

Arhitektura aplikacije predstavlja samu strukturu projekta i biblioteka koje su implementirane i korišćene. Prilikom implementacije sistema za opsluživanje paketa, korišćena je N-slojna arhitektura.

Ona predstavlja razdvajanje logičkih celina kao što su prezentacioni sloj, logički sloj i ***data-access*** sloj. Ovim pristupom se zadovoljava ***design pattern*** - ***separation of concerns.***

Slojevi sistema:

- ***API*** sloj - u ovom sloju se nalaze kontroleri koji su otvoreni spolja i sadrže ***endpoint***-e za logovanje, kreiranje i sortiranje paketa, dobavljanje statusa paketa itd.



Slika 2. *Klas dijagram entiteta*

- ***Core*** sloj - u ovom sloju se nalazi biznis logika aplikacije, npr. Servisi koji interaktuju sa repozitorijumima koji imaju pristup bazi. Ovaj sloj nema direktan pristup bazi, sa ovim pristupom smo omogućili da sa zahtevom za izmenu ***ORM***-a kao što je ***Entity Framework*** menjamo samo jedan sloj, a sloj koji ima pristup bazi ostane netaknut.
- ***Data*** sloj - u ovom sloju se nalaze migracije koje se izvršavaju prilikom prve inicijalizacije projekta i koje kreiraju praznu bazu podataka.
- ***Domain*** sloj - u ovom sloju se nalaze svi modeli tj. entiteti aplikacije. Pristup koji se koristio u ovoj aplikaciji je ***code-first***.

Repository sloj - u ovom sloju se nalaze repozirojumi koji imaju pristup samoj bazi podataka. Uz pomoć repozitorijuma je moguće dodati određeni rekord u bazu, dobaviti ga, obrisati ili izmeniti.

4.2. Mikroservisna arhitektura

Mikroservisna arhitektura [3] je arhitektonski stil koji se koristi kod modernih aplikacija koje je potrebno skalirati. Neke od glavnih osobina mikroservisne arhitekture su:

- ***Highly maintainable and testable*** - Olakšano održavanje manjih delova sistema, kao i mogućnost testiranja određenih ***unit***-a
- ***Independently deployable*** - Izmena malog dela aplikacije ne zahteva ponovan ***deployment*** cele aplikacije
- ***Scalability*** - Obezbeđuje karakteristike koje doprinose horizontalnoj skalabilnosti odvojenih delova sistema, ukoliko sistem poraste ovo doprinosi uštedi novca za servere potrebne da bi aplikacija radila
- ***Separation of concerns*** - Odvojeni timovi mogu biti zaduženi za različite delove sistema što doprinosi podeli odgovornosti između timova

U mikroservisnoj arhitekturi, aplikacija je izdeljena na više servisa za razliku od monolitne arhitekture. Svaki mikroservis sadrži svoju poslovnu logiku i u većini slučajeva sadrži i zasebnu bazu podataka. U monolitnoj arhitekturi, nema podele, čime se eliminiše ***separation of concerns*** svojstvo.

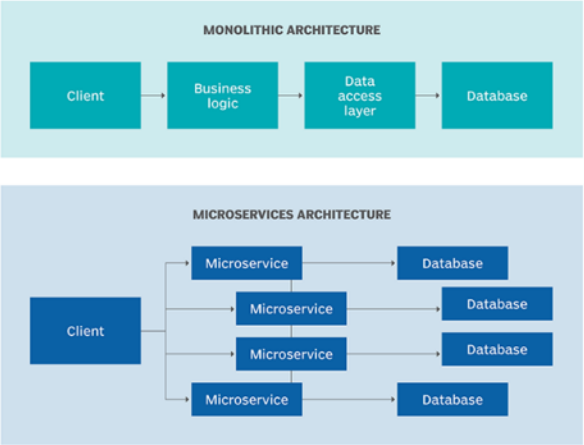
Aplikacije implementirane po ugledu na monolitnu arhitekturu otežavaju testiranje i izmenu softvera, jer problem koji je nastao može da se nalazi bilo gde u sistemu.

Ukoliko je sistem porastao, to može uzeti dosta više vremena u pronalaženju problema. Bilo koja manja promena u sistemu zahteva ***deployment*** celog sistema. Skaliranje monolitnih aplikacija može da bude teško i skupa za razliku od mikroservisne arhitekture, iz razloga jer je nemoguće skalirati samo odvojene manje delove sistema gde postoji veći ***load***.

Primer monolitne i mikroservisne arhitekture prikazan je na slici 3.

U sistemu za opsluživanje paketa koristi se mikroservisna arhitektura. Aplikacija se sastoji iz četiri različita mikroservisa:

- Autorizacioni mikroservis – Ovaj servis, kao što sam naziv kaže, ima namenu da autorizuje sve zahteve koji dolaze ka sistemu. Ovaj servis je ujedno i ***API Gateway***, sadrži bazu podataka koja ima informacije o validnim korisnicima koji mogu nesmetano da koriste aplikaciju kao i njihovim dozvolama i rolama.



Slika 3. *Primer monolitne i mikroservisne arhitekture*

- Servis za generisanje labela – Servis čija je namena generisanje labela. Labele je potrebno generisati u ***ZPL***, ***PNG*** i ***PDF*** formatima.
- Servis za komunikaciju sa ***USPS***-om – Servis koji sadrži biznis logiku za kreiranje ***outbound manifest***-a, povlačenje statusa paketa od momenta kada ***USPS*** dostavljač preuzme pakete, kao i svu logiku za kategorizaciju paketa za ***USPS***.
- Servis koji opslužuje sve akcije unutar sort centra – Biznis logika same aplikacije, namena je kreiranje paketa, sortiranje, pravljenje kontejnera i slanje kontejnera i paketa do sledeće destinacije.

4.3. Verifikacija adresa

Prilikom kreiranja porudžbine, potrebno je bilo verifikovati da li je adresa krajnjeg korisnika validna kako bi se izbeglo dostavljanje na nepostojeće adrese i kako bi se eliminisao dodatni napor koji bi bio potreban da se započne dostavljanje paketa pa ponovno vraćanje na povratnu adresu. Za ove potrebe postoji više rešenja na tržištu, neki od ***provider***-a koji nude verifikaciju adresa u Sjedinjenim Američkim Državama su:

- ***USPS Address Verification API*** [4]
- ***SmartyStreets*** [5]

Prvobitno, sistem za opsluživanje paketa se integrisao sa ***USPS Address Verification provider***-om. Sama integracija nije bila najjednostavnija jer se podaci šalju preko ***XML***-a i sam ***API*** je zastareo i nije u skladu sa današnjim arhitekturalnim konvencijama.

Nakon uspešne integracije, počeli su da se dešavaju određeni izazovi kao što je nedostupnost ***USPS Address Verification API***-ja, a u određenim momentima ***API*** vraća response za 250-300 milisekundi, a ta vrednost neretko ode i na 500 milisekundi.

Nakon susretanja sa ovim problemima, odlučeno je da se pređe na ***SmartyStreets provider***-a i da se on koristi za verifikaciju adrese. Međutim, usled situacija gde je i ***SmartyStreets*** bio nedostupan, krajnja odluka je bila da se koriste oba provajdera za verifikaciju adrese u paraleli.

4.4. Generisanje labela

Za generisanje labela koristio se ***ZPL*** ili ***Zebra programming language*** [6]. Zebra uređaji su kompatibilni i često se integrišu sa ***android*** uređajima, imaju ugrađenu podršku za automatsko štampanje formata bez ikakve potrebe za dodatnim instalacijama i podešavanjima.

Komande ***ZPL*** jezika uvek počinju sa jednim od dva znaka, `^` ili `~`. Postoji ukupno 170 komandi i svaki format komande mora da počne sa `^XA` i da se završi `^XZ`. Jedan primer komande je `^AND,n,m` gde je `n` font veličina, dok je `m` razmak karaktera.

Na slici 4 vidi se jedan primer ***USPS*** labele izgenerisane ***ZPL*** komandama.

Slika 4. *Primer izgleda labele za pakete*

Alat koji se koristi za konverziju ***ZPL*** formata u ***PDF*** i ***PNG*** naziva se ***Labelary***. ***Labelary*** je ***online viewer*** koji omogućava okruženje gde se može testirati i pogledati izgled labele izgenerisan pomoću određenih ***ZPL*** komandi.

Labelary je besplatan, nudi ***public API*** koji može da koristi svako ko ima pristup internetu, ali ***public API*** nije namenjen za produkciju, pa samim tim ***Labelary*** nudi ***offline*** verziju koja se koristi lokalno, čime se eleminiše i kašnjenje.

5. ZAKLJUČAK

U okviru ovog rada predstavljen je ***.NET Core***, jedan od najperspektivnijih frejmvorka sadašnjice. ***Microsoft*** sa svakom verzijom poboljšava skup funkcionalnosti i ubrzava performanse samog frejmvorka. Od kako je ***.NET Core*** zamenio ***.NET Framework***, ovaj frejmvork ima veliku primenu u praksi, s obzirom da je ***cross-platform***. Iz tog razloga je i odabran taj frejmvork za implementaciju sistema za opsluživanje paketa.

Sam sistem je performantan i veoma lako ga je migrirati na novije verzije ***.NET Core***-a koje će izaći u budućnosti, to je još jedna odlična odlika ovog radnog okvira. Neke od stvari kojima bi sistem za opsluživanje paketa mogao biti unapređen jeste ubrzavanje poziva ka bazi podataka korišćenjem čistih ***SQL query***-a umesto ***ORM***-a.

Ovo je izazov sa kojim se susreću svi veći sistemi koji dođu u ovu fazu. Na početku je ***Entity Framework*** bio najbrže rešenje za efikasnost prilikom implementacije, međutim to može da bude dug koji kasnije dolazi na

naplatu. Primer ***ORM***-a koji je performantniji od ***Entity Framework***-a je ***Dapper***.

6. LITERATURA

[1] https://dotnet.microsoft.com

[2] https://agileviet.vn/whats-new-in-net-5

[3] https://www.techtarget.com/searchapparchitecture

[4] https://www.usps.com/business/web-tools-apis

[5] https://www.smarty.com/products/single-address

[6] https://www.zebra.com

Kratka biografija:

Stanko Jevtić rođen je u Novom Sadu 1994. god. Završio je tehničku školu „9. Maj“ u Bačkoj Palanci 2013. godine. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu je završio 2017. Master akademske studije na Fakultetu tehničkih nauka u Novom Sadu upisao je 2017. godine.

kontakt: stanjevtic@gmail.com