Testiranje API-ja u Postman-u

Krešo, Marijana

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Physics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za fiziku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:160:935422

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-26**

Repository / Repozitorij:

Repository of Department of Physics in Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ODJEL ZA FIZIKU

MARIJANA KREŠO

Testiranje API-ja u Postman-u

Diplomski rad

Osijek, 2022.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ODJEL ZA FIZIKU

MARIJANA KREŠO

Testiranje API-ja u Postman-u

Diplomski rad

Priložen Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku

radi stjecanja akademskog naziva magistra edukacije fizike i informatike

Osijek, 2022.

"Ovaj diplomski rad izrađen je u Osijeku pod vodstvom Doc.dr.sc. Ivana Vazlera u sklopu Sveučilišnog diplomskog studija fizike i informatike na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku".

Zahvale

Zahvaljujem se Doc.dr.sc. Ivanu Vazleru na stručnim savjetima i usmjeravanju i za svu pomoć pri realizaciji ovog rada.

Velika hvala mojoj obitelji, dečku i kolegici Andrei Vukoja na podršci.

Sadržaj

1. Uvod 1
2. API 2
2.1. JSON 2
2.2. REST komunikacijski protokol 4
2.3. Kodiranje znakova 7
2.4. Opća načela o API-ju 8
2.4.1. API zaglavlja 8
2.4.2. API parametri 9
3. Postman 10
3.1. Okruženja 14
3.2. Zbirke 16
4. Testiranje Spotify API-ja 18
4.1. Općenito o Spotify-u 18
4.2. Spotify dokumentacija 18
4.2.1. Autentifikacija 20
4.3. Scenarij testiranja 20
4.3.1. Spotify – slanje pristupnog tokena (POST zahtjev) 21
4.3.2. Spotify - GET zahtjevi 23
4.3.2.1. Spotify – dohvaćanje Id-ja umjetnika (GET zahtjev) 23
4.3.2.2. Spotify – traženje albuma prema id-ju umjetnika (GET zahtjev) 25
4.3.3. Spotify – slanje zahtjeva za praćenje umjetnika (PUT zahtjev) 26
4.3.4. Spotify – slanje zahtjeva za prestanak praćenja umjetnika (DELETE zahtjev) 27
5. Automatizirani testovi u Postman-u 29
5.1. Zašto se pišu automatizirani testovi? 29
5.2. Postupak automatiziranja testova u Postman-u 29
5.3. Primjer automatiziranog testiranja Spotify API-ja 31
6. Zaključak 33
Literatura 34
Životopis 37

Sveučilište Josipa Jurja Strossmayera u Osijeku
Odjel za fiziku

TESTIRANJE API-JA U POSTMAN-U

MARIJANA KREŠO

Sažetak

U prvom dijelu diplomskog rada uvedeni su i pojašnjeni pojmovi API-ja, REST komunikacijskog protokola i JSON formata prijenosa podataka. Navedene su i objašnjene glavne HTTP metode za komunikaciju klijent-server kao i sastavni dijelovi zahtjeva i odgovora pri komunikaciji s API-jem. U drugom dijelu rada definiran je i opisan Postman kao glavni alat korišten u radu, njegovi elementi sučelja te načini baratanja sa zahtjevima i pojedinim dijelovima odgovora. Cilj rada bio je opisati postupak testiranja API-ja u Postman alatu što je realizirano u završnom dijelu rada na konkretnom primjeru koristeći Spotify API.

Rad je pohranjen u knjižnici Odjela za fiziku

Ključne riječi: testiranje/Postman/API/JSON/REST/zahtjev/odgovor

Mentor: Doc.dr.sc. Ivan Vazler

Ocjenjivači:

Rad prihvaćen:

J. J. Strossmayer University in Osijek
Department of Physics

Master of Science Thesis

API TESTING IN POSTMAN

MARIJANA KREŠO

Abstract:

In the first part of the thesis, the concepts of API, REST communication protocol and JSON data transmission format were introduced and clarified. The main HTTP methods for client-server communication as well as the components of requests and responses when communicating with the API are listed and explained. In the second part of the paper, Postman is defined and described as the main tool used in the paper, as well as its interface elements and ways of dealing with requests and individual parts of the answers. The aim of the paper was to describe the process of testing the API in the Postman tool, which was realized in the final part of the work on a specific example using the Spotify API.

Thesis deposited in Department of Physics library

Keywords: testing/Postman/API/REST/JSON/request/response

Supervisor: Doc.dr.sc. Ivan Vazler

Reviewers:

Thesis accepted:

1. Uvod

Testiranje softverskih rješenja postalo je jedan od najbitnijih dijelova životnog ciklusa suvremenih programskih aplikacija. Dok je pri testiranju korisničkog sučelja dovoljno vizualno i funkcionalno potvrditi valjanost dijelova aplikacije, testiranje poslovne logike ipak je složenije i zahtijeva drugačiji pristup problematici nego testiranje korisničkog sučelja, uz puno više potrebnog znanja o načinu na koji aplikacija funkcionira te razumijevanja programerske strane aplikacije.

Postupak testiranja poslužiteljskog dijela aplikacije podrazumijeva poznavanje osnovnih dijelova poslužitelja, načina na koji poslužitelj i klijent komuniciraju te očekivanja koja poslužitelj ima pri toj komunikaciji. Nepoznavanjem ili nepravilnim razumijevanjem bilo kojeg od ovih čimbenika može doći do nepravilno ili nepotpuno testirane aplikacije što u najgorim slučajevima dovodi do ogromne materijalne štete i posljedica koje sežu daleko izvan virtualnog svijeta u kojem se aplikacija vrti (npr. greške u sustavima koji barataju novcem ili administracijskim sustavima).

Kako bi se izbjegli takvi scenariji i steklo razumijevanje o pravilnom testiranju poslovne logike neke aplikacije, prvo je potrebno razumjeti što je točno ta poslovna logika (API) i na koji način ona želi komunicirati s klijentom, što podrazumijeva poznavanje REST protokola, JSON formata prijenosa podataka i osnovnih HTTP metoda za komunikaciju. Također je neophodno savladati korištenje nekog od alata za testiranje API-ja, kao što je Postman. Testiranje API-ja složen je proces, no uz pravilno predstavljanje osnova na kojima je API izgrađen moguće je prikazati ovaj proces na jasan i razumljiv način.

2. API

API (eng. *Application Programming Interface*) je sučelje između klijenta i servera, gdje klijent predstavlja korisničko sučelje i šalje zahtjeve (eng. *request*) dok server predstavlja poslovnu logiku te on odgovara na klijentski zahtjev vraćajući odgovor (eng. *response*).

2.1. JSON

JSON (eng. *JavaScript Object Notation*) je format za komunikaciju tj. razmjenu podataka. Služi za vraćanje podataka klijentu sa servera na klijentu razumljiv način. JSON je temeljen na notaciji objekata u JavaScript programskom jeziku, no o njemu je postao potpuno neovisan. Razumljiv je mnogim programskim jezicima, ali i ljudima te je zato idealan za razmjenjivanje podataka (vidi [1]).

JSON je građen na temelju dviju struktura:

- Kolekcija parova ključeva/vrijednosti (također poznato kao i objekt, rječnik, označena lista itd.)
- Poredana lista vrijednosti. U većini jezika struktura je ostvarena kao polje, vektor, lista ili niz.

JSON definira šest tipova podataka, koji se dijele na osnovne i složene tipove. Pod osnovne tipove smatraju se string, number, boolean i null, a pod složene tipove pripadaju niz i objekt.

```
{
   "array": [
      1,
      2,
      3,
      4
   ],
   "boolean": true,
   "color": "#82b92c",
   "null": null,
   "number": 123,
   "object": {
      "a": "b",
      "c": "d",
      "e": "f"
   },
   "string": "Hello World"
}
```


Programski kod 2.1 – JSON tekstualni format [2]

U programskom kodu 2.1 vidljivo je kako su prikazani objekti, nizovi, svojstva u JSON formatu.

Objekt je neuređen skup parova ključeva/vrijednosti, što znači da poredak u objektu nije bitan. Objekt se prepoznaje po vitičastim zagradama {}. Nakon svakog ključa slijedi dvotočka : te se po tome prepoznaje par ključ/vrijednost (dalje u tekstu ćemo parove ključ/vrijednost označavati "*ključ*":"*vrijednost*"), a svaki od tih parova odvojeni su zarezom.

Niz (eng. *Array*) je uređeni skup vrijednosti, što znači da je poredak u polju bitan. Niz se prepoznaje po uglatim zagradama []. Vrijednosti unutar niza odvojene su zarezima.

Vrijednost može biti string pod dvostrukim navodnicima, broj, boolean (true ili false), null, objekt ili niz. Ove se strukture mogu ugnijezditi.

Osnovne karakteristike JSON formata:

- Sadrži samo korisne podatke (podatke koje poslužitelj ili klijent mogu izravno koristiti; nema dodane sintakse za razliku od npr. XML formata)

- Podržava objekte, varijable i polja kao i većina modernih programskih jezika; nema potrebe za prilagodbom formata kako bi ga programski jezik razumio

- Jednostavan oblik zapisa strukture i podataka [3]

2.2. REST komunikacijski protokol

REST (eng. *Representational State Transfer*) arhitektonski je stil za izgradnju web usluga, odnosno skup je pravila koja definiraju oblik komunikacije između klijenta i servera. Definira implementaciju životnog ciklusa zahtjeva, odnosno na koji način će biti realizirano slanje zahtjeva s klijenta, kao i primanje zahtjeva na poslužitelja.

Osnovna svojstva web usluga temeljenih na REST arhitektonskom stilu su:

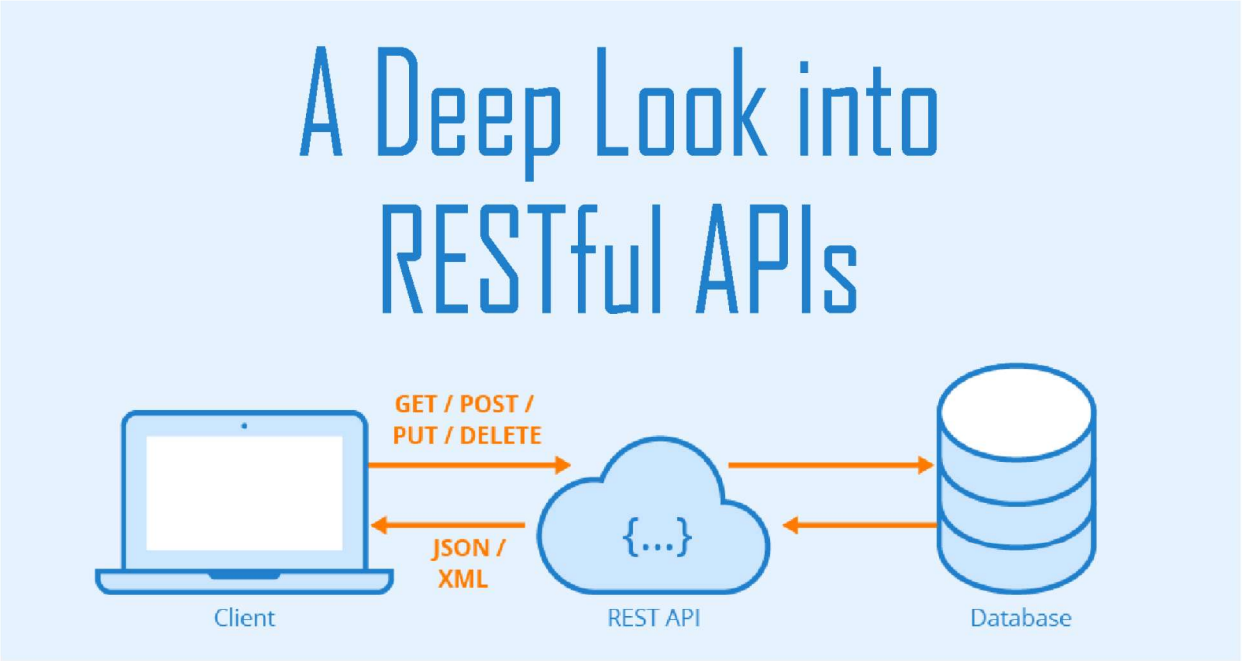
- Klijent-server stil komunikacije
- Predmemorija servera ne koristi se za čuvanje stanja podataka
- Korištenje isključivo HTTP metoda
- Uniformno sučelje [4]

REST API web servis koristi komunikacijske metode HTTP protokola. Najkorištenije HTTP metode su:

- GET – dohvaća podatke sa servera
- POST – stvara nove podatke na serveru
- PUT – uređuje podatke na serveru
- DELETE – briše podatke sa servera

Jedna od glavnih prednosti REST arhitektonskog stila je mogućnost izbora više tekstualnih formata za komunikaciju. Ovaj izbor ovisi o potrebama aplikacije i razvojnom okruženju u kojem se koristi. Dva najpopularnija i najčešće korištena tekstualna formata za komunikaciju su JSON i XML. REST je danas daleko najrašireniji oblik izrade API-ja i koristi se u malim i srednjim tvrtkama, ali i u većini svjetskih IT divova kao što su Amazon, Google, Microsoft itd.

Nakon što je klijent poslao svoj zahtjev serveru, server šalje nazad klijentu odgovor (engl. *Response*) koji se sastoji od poruke (obično u JSON ili XML formatu) i standardni HTTP statusni kod. [5] Prikaz načina komunikacije u REST API-ju prikazuje Slika 2.2.



Slika 2.2 – Komunikacija u REST API-ju [6]

Odgovori su grupirani u pet razreda:

- Informativni odgovori (statusni kodovi 100–199)
- Uspješni odgovori (statusni kodovi 200–299)
- Poruke preusmjeravanja (statusni kodovi 300-399)
- Odgovori na pogreške klijenta (statusni kodovi 400–499)
- Odgovori na pogreške poslužitelja (statusni kodovi 500-599) [5].

U tablici 2.1 navedeni su najčešći HTTP statusni kodovi.

Status kod	Značenje
200	"*OK*" Ovo je standardni statusni kod za uspješan HTTP zahtjev.
201	"*Created*" Ovo je statusni kod koji potvrđuje da je zahtjev bio uspješan i kao rezultat toga stvoren je novi resurs. Obično je to statusni kod koji se šalje nakon POST/PUT zahtjeva.
204	"*No Content*" Ovaj statusni kod potvrđuje da je poslužitelj ispunio zahtjev, ali ne mora vraćati informacije.
301	"*Moved Permanently*" Statusni kod 301 se koristi za trajno preusmjeravanje, što znači da bi se krajnje točke ili zapisi koji vraćaju ovaj odgovor trebali ažurirati.
400	"*Bad Request*" Poslužitelj ne može razumjeti i obraditi zahtjev zbog greške klijenta.
401	"*Unauthorized*" Ovaj statusni kod javlja se kada je potrebna provjera autentičnosti, ali nije uspjela ili nije pružena.
403	"*Forbidden*" Ovo je statusni kod koji se javlja kada je poslan valjani zahtjev, ali ga poslužitelj odbija prihvatiti. Za razliku od statusnog koda 401, autentifikacija se ovdje neće primjenjivati.
404	"*Not Found*" Najčešći statusni kod koji će prosječni korisnik vidjeti. Statusni kod 404 pojavljuje se kad je zahtjev valjan, ali se resurs ne može pronaći na poslužitelju.
500	"*Internal Server Error*" Kodovi razreda 500 slični su kodovima razreda 400 po tome što su pravi kodovi pogreške. Statusni kod 500 se događa kada poslužitelj ne može ispuniti zahtjev zbog neočekivanog problema.

Tablica 2.1 – Uobičajeni HTTP statusni kodovi [18]

2.3. Kodiranje znakova

Kodiranje znakova je postupak kojim se ne-znakovni entiteti (slike, zvuk i sl.) ili znakovni entiteti (tekst) pretvaraju u nizove tekstualnih/binarnih znakova. Ovaj postupak se provodi zbog povećanja prenosivosti podataka bez obzira na podržane formate transportnog medija (npr. zahtjeva prema poslužitelju).

Postoji više vrsta ovakvog kodiranja, no najpoznatije i za potrebe ovog rada najrelevantnije kodiranje znakova naziva se Base64 kodiranje koje se temelji na 64 moguća znaka kojima se mogu kodirati ulazni podaci. Kombiniranjem ovih znakova dobiva se niz znakova koji predstavlja ulazni podatak na način koji je razumljiv poslužitelju.

Prvi korak pri kodiranju je pretvaranje ulaznih podataka u binarni zapis (svaki ulazni znak se pretvara u 8-bitnu riječ). Nakon ove pretvorbe Base64 reprezentacija dobiva se podjelom dobivenog niza binarnih znakova na grupe od 6 bitova, gdje se svaki bit pretvara u Base64 znak prema tablici 2.2.

Index	Char	Binary	Index	Char	Binary	Index	Char	Binary	Index	Char	Binary
0	A	0	16	Q	10000	32	g	100000	48	w	110000
1	B	1	17	R	10001	33	h	100001	49	x	110001
2	C	10	18	S	10010	34	i	100010	50	y	110010
3	D	11	19	T	10011	35	j	100011	51	z	110011
4	E	100	20	U	10100	36	k	100100	52	0	110100
5	F	101	21	V	10101	37	l	100101	53	1	110101
6	G	110	22	W	10110	38	m	100110	54	2	110110
7	H	111	23	X	10111	39	n	100111	55	3	110111
8	I	1000	24	Y	11000	40	o	101000	56	4	111000
9	J	1001	25	Z	11001	41	p	101001	57	5	111001
10	K	1010	26	a	11010	42	q	101010	58	6	111010
11	L	1011	27	b	11011	43	r	101011	59	7	111011
12	M	1100	28	c	11100	44	s	101100	60	8	111100
13	N	1101	29	d	11101	45	t	101101	61	9	111101
14	O	1110	30	e	11110	46	u	101110	62	+	111110
15	P	1111	31	f	11111	47	v	101111	63	/	111111

Tablica 2.2 – Uobičajeni HTTP statusni kodovi [19]

Osim kompatibilnosti, jedan od razloga korištenja kodiranja je i sigurnost podataka, jer se kodirani podaci ne prenose u svom izvornom obliku što otežava potencijalnu krađu podataka i smanjuje njihovu čitljivost prema trećim stranama. Ovakva primjena kodiranja najčešće je vidljiva upravo u API-jima gdje se koristi u autorizacijske svrhe pri slanju zahtjeva, a Base64 kodiranje će upravo na taj način biti korišteno i u ovom radu (više detalja u [19]).

2.4. Opća načela o API-ju

Kada netko govori o API-ju, uglavnom govori o web API-ju. Web API preuzima koncept sučelja između dvije stvari i primjenjuje ga na odnos klijent/server na kojem je izgrađen internet. U web API-ju, klijent je s jedne strane sučelja i šalje zahtjeve, dok je server (ili serveri) s druge strane sučelja i odgovara na zahtjev.

Neki pozivi API-jima mogu promijeniti stvari na serveru, dok drugi vraćaju podatke bez promjene. Siguran zahtjev je onaj koji ne mijenja ništa na serveru. Traženjem od servera informacije o tome što se događa na serveru, ništa se ne mijenja na samom serveru. Ako klijent pošalje zahtjev koji mijenja nešto na serveru, klijent nije napravio siguran zahtjev.

Postoji i API poziv koji je poznat kao idempotentni poziv. Ovakav poziv mijenja stvari samo kad se prvi put izvrši i ne čini nikakve promjene na sljedećim pozivima, odnosno pri svakom idućem pozivu vraća isti rezultat.

Svaki API poziv treba navesti resurs s kojim se radi. To je krajnja točka (eng. endpoint).

Krajnjoj točki pristupa se preko URL-a (eng. *Uniform Resource Locator*). Osim toga, API mora znati što učiniti s navedenim resursom, a to se određuje tako da se definira API akcija.

Postoje četiri glavne API akcije (POST, GET, PUT i DELETE) i često su predstavljene skraćenicom CRUD (*Create*, *Read*, *Update*, *Delete*) (vidi [5]).

2.4.1. API zaglavlja

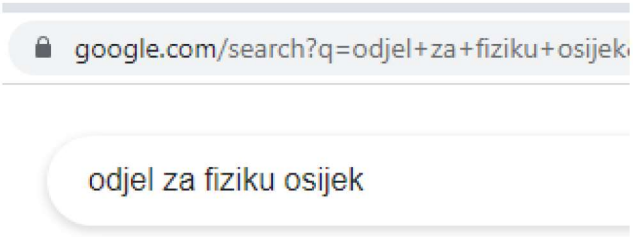
API zaglavlja (eng. *Header*) dijelovi su zahtjeva koji opisuju sam zahtjev neovisno o njegovom sadržaju. Kod testiranja API-ja najčešća su sljedeća zaglavlja:

- Autorizacija (eng. *Authorization*): sadrži podatke o ovlastima, odnosno pravima klijenta za čitanje, pisanje, promjenu i brisanje traženog resursa
- WWW-Authenticate: šalje ga poslužitelj ako mu je potreban oblik provjere prava pristupa prije nego što može odgovoriti sa stvarnim traženim resursom - server traži potvrdu od klijenta da je klijent taj za kojeg se predstavlja (obično se provodi unosom korisničkog imena i šifre); često se šalje zajedno s kodom odgovora 401 (Tablica 2.1)
- Accept-Charset: govori poslužitelju o tome koje kodiranje znakova klijent prihvaća
- Content-Type: označava vrstu medija (tekst/html ili tekst/JSON) odgovora koji poslužitelj šalje klijentu, to će pomoći klijentu u pravilnoj obradi tijela odgovora
- Kontrola predmemorije (eng. *Cache-Control*): politika predmemorije definirana od strane poslužitelja za ovaj odgovor - klijent može pohraniti predmemorirani odgovor i ponovno ga koristiti do vremena definiranog zaglavljem Cache-Control

2.4.2. API parametri

API parametri su promjenjivi dijelovi zahtjeva. Oni određuju vrstu radnje koja se poduzima na resursu. Svaki parametar ima naziv, vrstu vrijednosti i neobavezni opis. Pri izgradnji REST API-ja potrebno je odlučiti koji parametri trebaju biti prisutni u API krajnjoj točki. Jednostavno rečeno, API parametri su opcije koje se mogu proslijediti krajnjoj točki kako bi se utjecalo na odgovor.

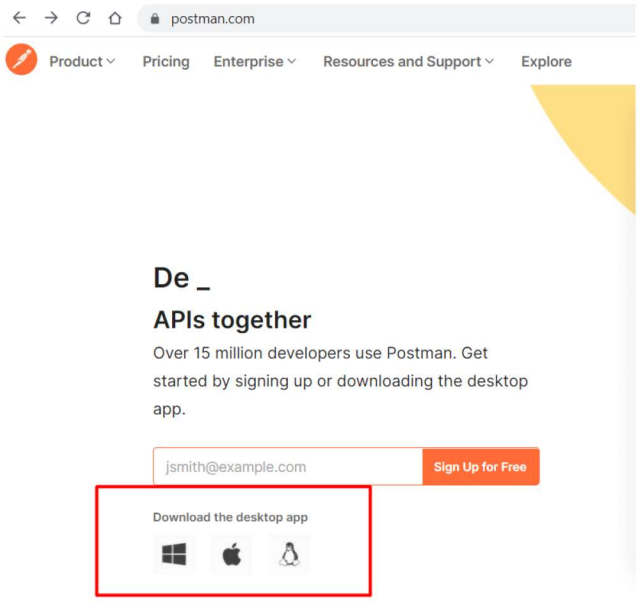
Najčešće korišteni parametri su parametri upita (eng. *Query parameters*) – prepoznaju se prema upitniku u API krajnjoj točki i navedeni su s ključem koji je stavka pretraživanja i vrijednošću koja je upit pretraživanja (vidi [5]). Slika 2.3. prikazuje primjer izgleda URL-a na google.com nakon što su u pretraživač uneseni pojmovi za pretraživanje. Unutar URL-a parametar upita započinje upitnikom, zatim ima svoj par ključa i vrijednosti "*q*":"*odjel+za+fiziku+osijek*".



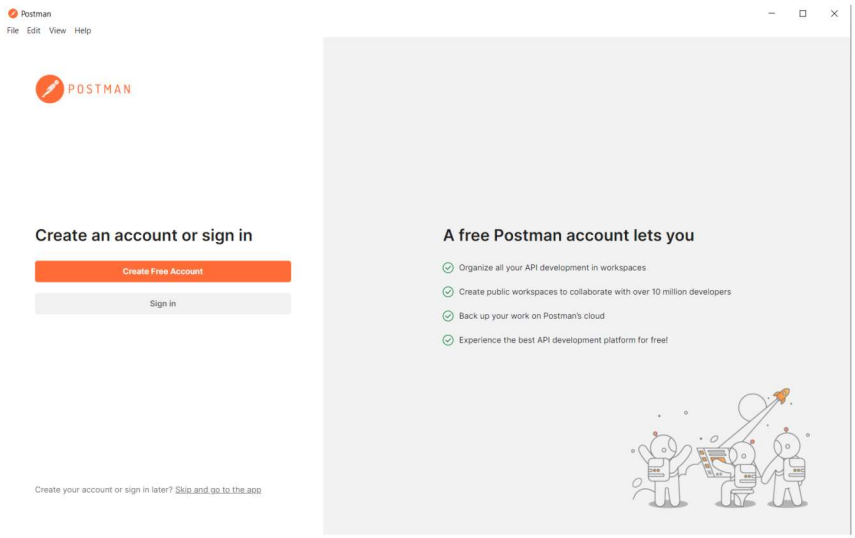
Slika 2.3 – Parametar upita u URL-u

3. Postman

Postman je API platforma za izgradnju i korištenje/testiranje API-ja. Postman se može preuzeti na njegovoj službenoj stranici https://postman.com, osim aplikacije postoji i web sučelje koje se može korisiti i bez preuzimanja (vidi [7]).

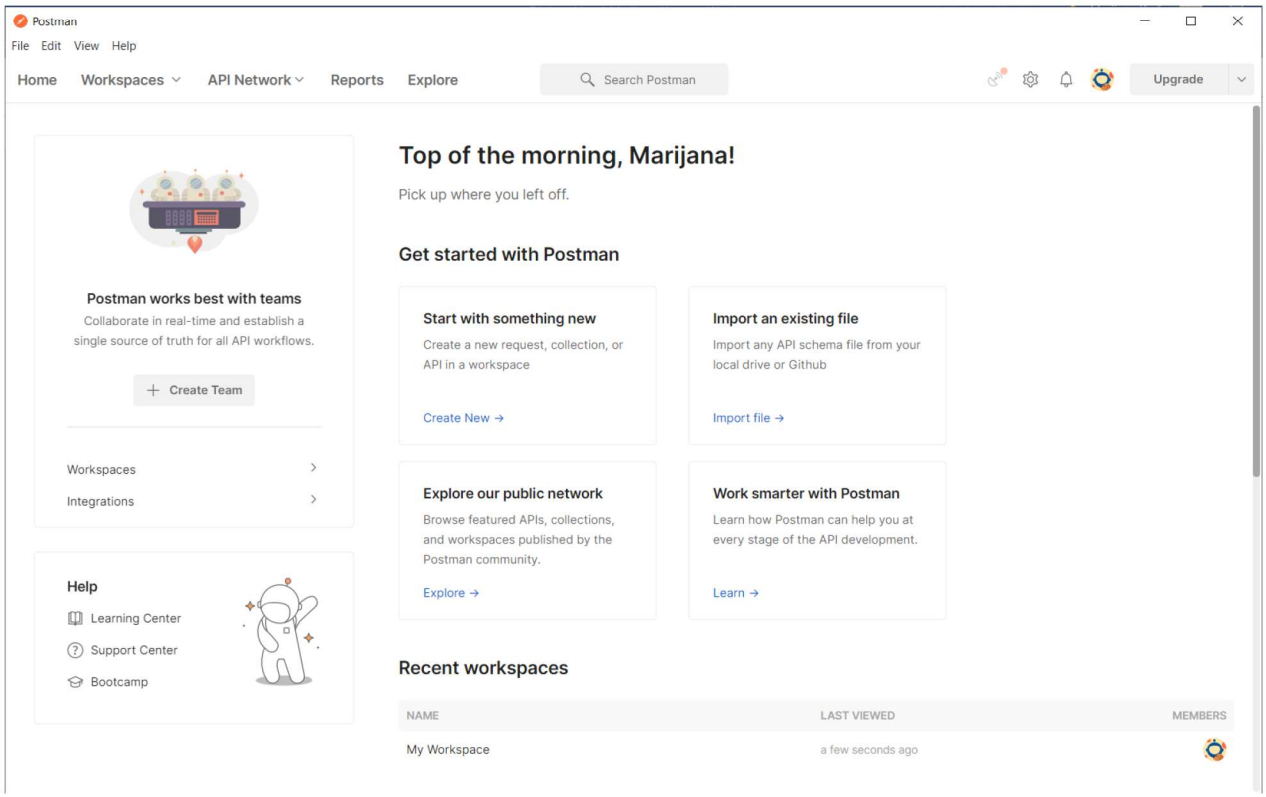


Slika 3.1 – službena Postman stranica - https://postman.com



Slika 3.2 – Postman-ov prozor za prijavu

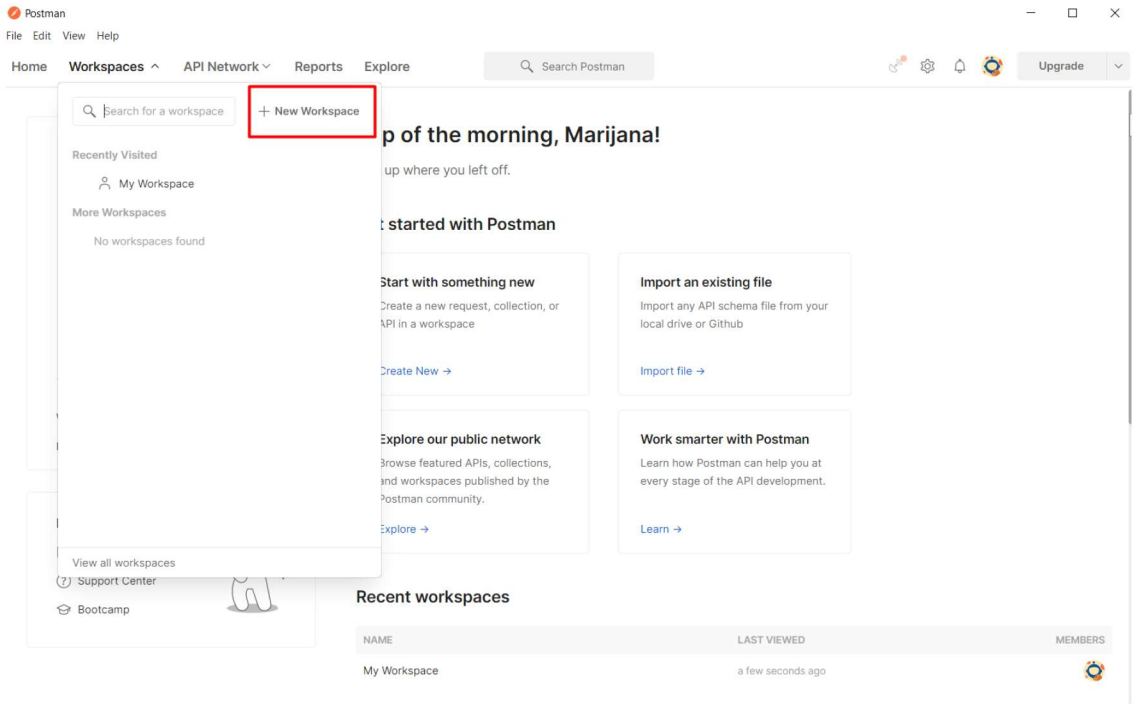
Nakon uspješne instalacije, u Postman-u se prikazuje prozor za prijavu u aplikaciju odnosno opcija za stvaranje novog računa (Slika 3.2). Nakon prijave, može se vidjeti glavni zaslon s mnoštvo različitih opcija za početak što se može učiniti s Postman-om (Slika 3.3).



Slika 3.3 – Postman-ov glavni zaslon

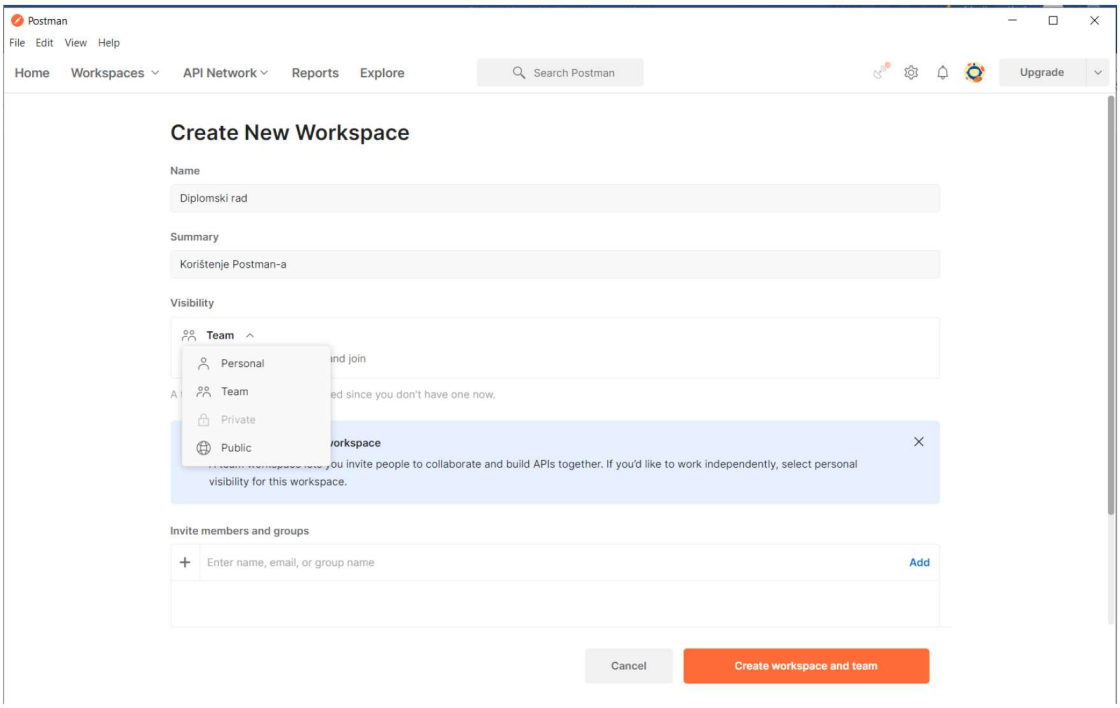
Ono što je zanimljivo za napraviti u Postman-u je "Radni prostor" (eng. *Workspace*). Radni prostor pruža mogućnost upravljanja i organiziranja različitih skupova zbirki (eng. *collections*). Opcijom stvaranja radnog prostora, Postman pruža priliku da se koristi i za privatne i poslovne svrhe, u tom slučaju može se stvoriti i *radni, osobni* i *timski* radni prostor (vidi [8]).

Klikom na "Workspace" u gornjem izborniku, otvori se "Workspace" modal u kojem postoji opcija za napraviti novi radni prostor (Slika 3.4).



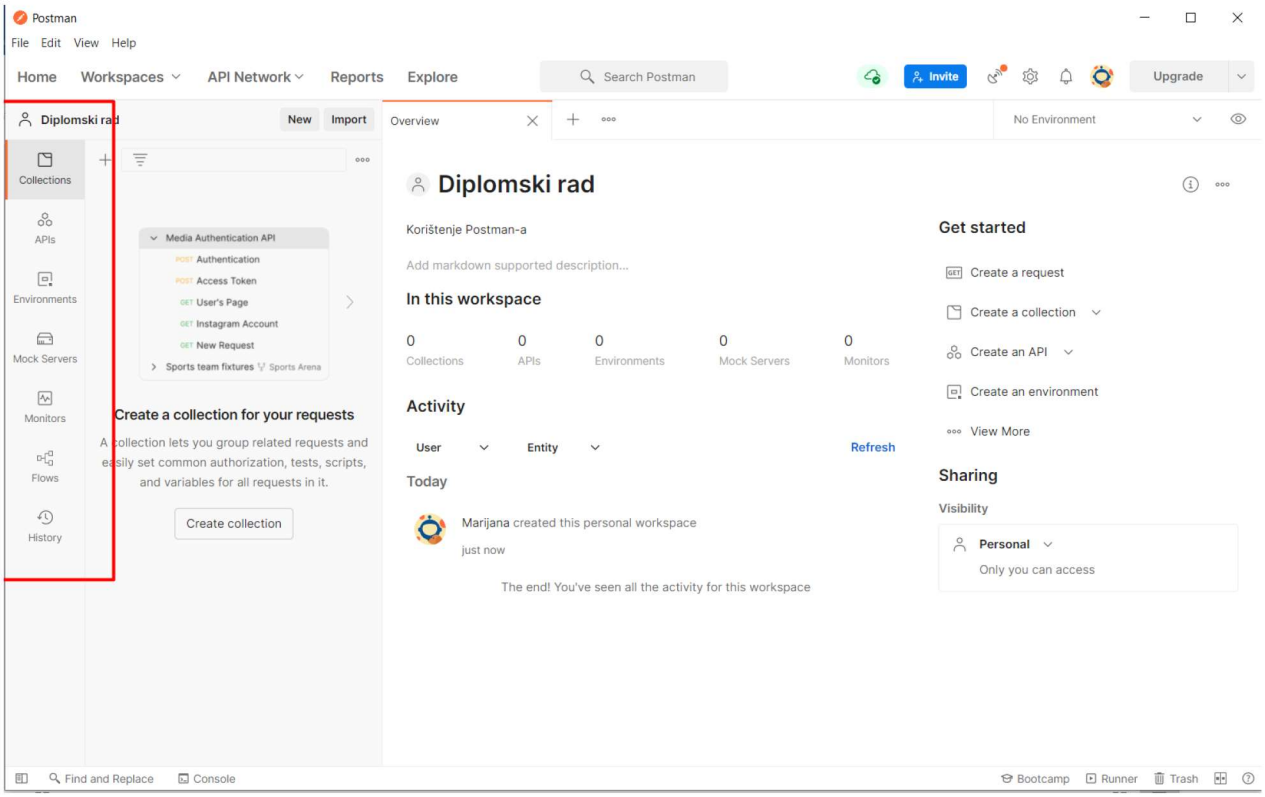
Slika 3.4 – Stvaranje novog radnog prostora

Vidljivo na slici 3.5, klikom na "New Workspace" otvori se novi prozor u kojem se ispunjavaju osnovni podaci o radnom prostoru (ime, opis, opcija vidljivosti, itd.).



Slika 3.5 – Uređivanje radnog prostora

Nakon što se napravi radni prostor, unutar njega Postman u lijevom izborniku sadrži još opcija za rad (Slika 3.6), od kojih su najviše korištene zbirke (eng. *Collections*) u kojima se mogu organizirati zahtjevi (eng. *Requests*) te okruženje (eng. *Environments*), za automatizaciju testova postoji opcija Protok (eng. *Flows*) koja će biti opisana u poglavlju 5.

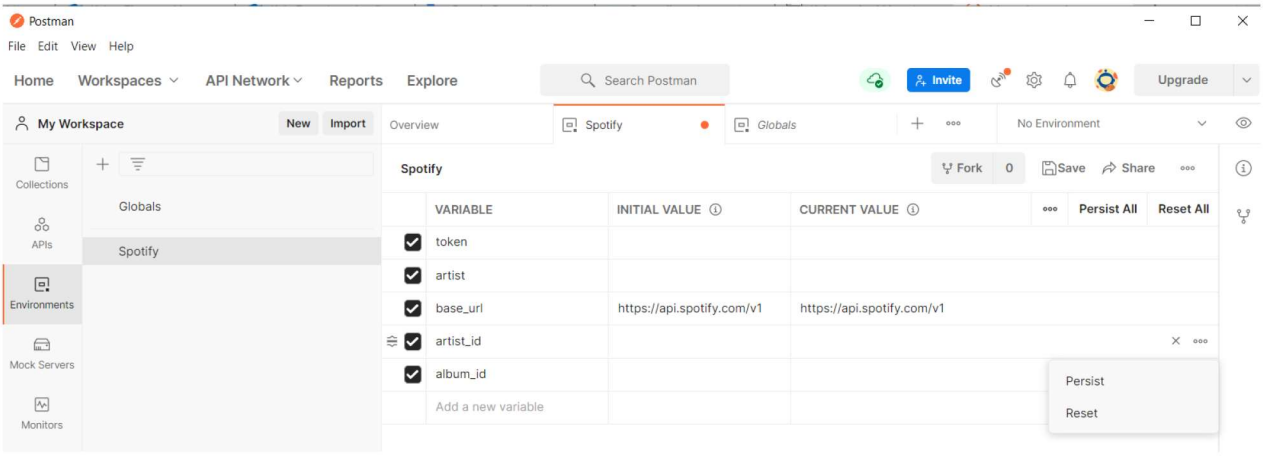


Slika 3.6 – Lijevi izbornik u radnom prostoru

3.1. Okruženja

Okruženja omogućuju da se krajnje točke konfiguriraju tako da koriste određene varijable koje olakšavaju upotrebu istih krajnjih točaka između različitih okruženja. Primjerice, moguće je imati istu /playlists krajnju točku i u razvojnom i u proizvodnom okruženju, ali te krajnje točke sadržavaju različite domene. Okruženja omogućuju upravljanje jednim zahtjevom s promjenjivom domenom (vidi [9]).

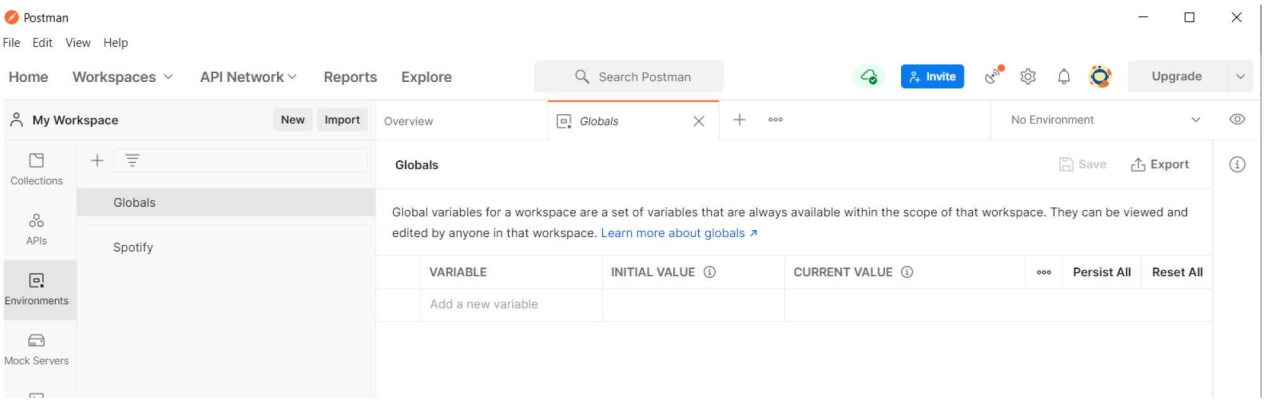
Klikom na Okruženja u lijevom izborniku, može se pristupiti svim varijablama koje su napravljene u okruženju. Na slici 3.7 vidi se primjer spremljenih varijabli u Spotify okruženju.



Slika 3.7 – Spremljene varijable u Spotify okruženju

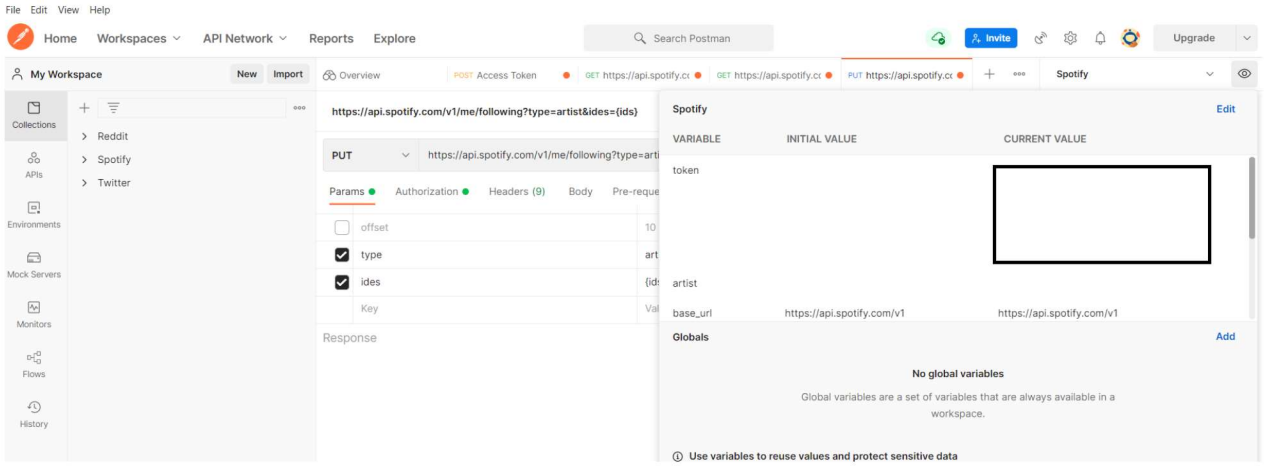
Nakon što se otvori kartica (eng. *Tab*) u odabranom okruženju, nove varijable se mogu dodavati, preimenovati te im postaviti vrijednost, stare nekorištene varijable mogu se brisati ili staviti u neaktivno stanje klikom na kvačice u potvrdnom okviru.

Na slici 3.8 vidi se mogućnost stvaranja globalnih varijabli u okruženju. Globalne varijable su varijable koje imaju opću namjenu i dostupne su svim zahtjevima na Postman-u, nebitno tome kojoj zbirci pripadaju. Ove varijable treba uglavnom izbjegavati i koristiti ih samo u potrebi brze izrade prototipa, jer su poprilično nepouzdane budući da svaki odlomak koda može pristupiti odnosno izmijeniti njihove vrijednosti (vidi [9]).



Slika 3.8 – Globalne varijable u okruženju

Alternativno, može se otvoriti Brzi pogled okruženja (eng. Environment Quick Look), odnosno klikom na ikonu "Oko" u gornjem desnom kutu otvoriti će se skočni prozor s detaljima oko trenutnih globalnih varijabli uključujući varijable napravljene u okruženju, što se može vidjeti na slici 3.9.



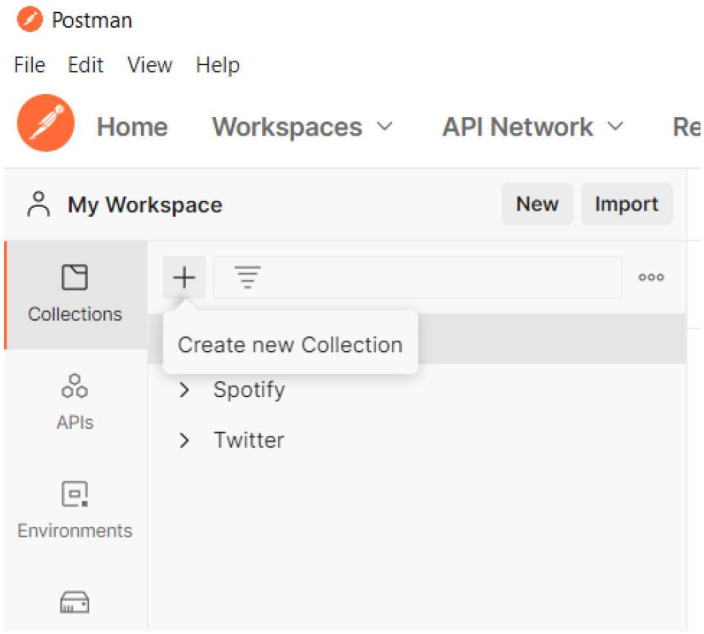
Slika 3.9 – Brzi pogled okruženja (eng. *Environment Quick Look*)

3.2. Zbirke

Zbirka je skupina Postman zahtjeva. Služi za organiziranje zahtjeva u različite grupe, ovisno o pripadnosti nekom API-ju i funkcionalnosti zahtjeva. Uzme li se za primjer Spotify API i Twitter API, moguće je napraviti zbirku koja će obuhvatiti sve Spotify zahtjeve te drugu zbirku koja će obuhvatiti sve Twitter zahtjeve.

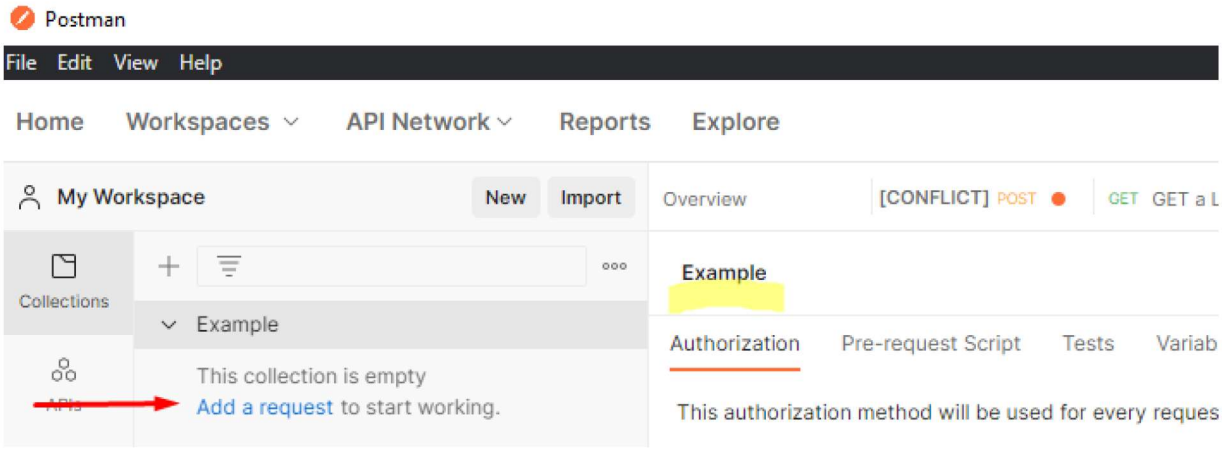
No, unutar takvih zbirki (ili neovisno o njima) mogu se napraviti dodatne zbirke koje će grupirati zahtjeve prema funkcionalnosti ili području djelovanja (npr. Spotify Artists API za sve zahtjeve koji se dotiču podataka o izvođačima ili Spotify Accounts API koji objedinjuje sve zahtjeve vezane za korisničke račune) (više detalja u [9]).

Klikom na Zbirke u lijevom izborniku, može se pristupiti svim već napravljenim zbirkama te se postoji opcija pravljenja nove zbirke. Na slici 3.10 vidi se primjer dviju napravljenih zbirki Twitter i Spotify, također je vidljiva opcija za kreiranje zbirke.



Slika 3.10 – Zbirke Spotify i Twitter

Zbirka se može nazvati proizvoljno bilo kojim imenom te se u zbirci mogu dodavati zahtjevi. Slika 3.11 prikazuje primjer nove zbirke Example.



Slika 3.11 – Zbirke Spotify i Twitter

4. Testiranje Spotify API-ja

4.1. Općenito o Spotify-u

Spotify je aplikacija za udaljeno slušanje glazbe putem interneta u realnom vremenu, koja nudi mogućnosti odabira izvođača i pjesme kao i mogućnost skidanja pjesama za kasnije korištenje bez internetske veze.

Daleko je najrašireniji i najpopularniji servis ove vrste na svijetu te ima najveću glazbenu kolekciju. Upravo zbog njegove raširenosti, jednostavnosti korištenja i pristupačnosti prosječnom korisniku Spotify je idealan primjer za prikazivanje postupka API testiranja.

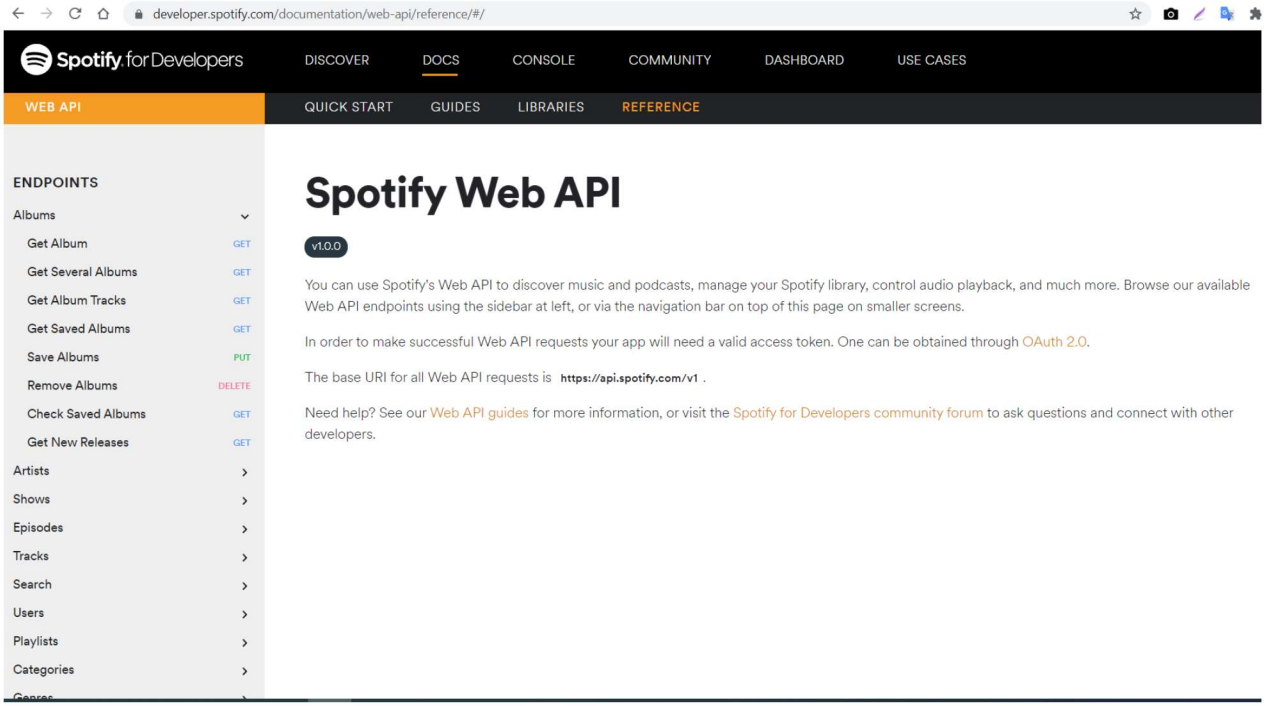
Spotify zahtijeva korisnički račun za korištenje, a uz samu reprodukciju glazbe neke od osnovnih funkcionalnosti su mu:

- pretraga po raznim kriterijima (po imenu umjetnika, nazivu albuma, nazivu pjesme…)
- praćenje umjetnika ili drugih korisnika
- prestanak praćenja umjetnika ili drugih korisnika

Ove osnovne funkcionalnosti imaju svoje odgovarajuće krajnje točke na Spotify API-ju, a upravo te krajnje točke će biti predmet demonstracije testiranja API-ja na ovakav način (vidi [17]).

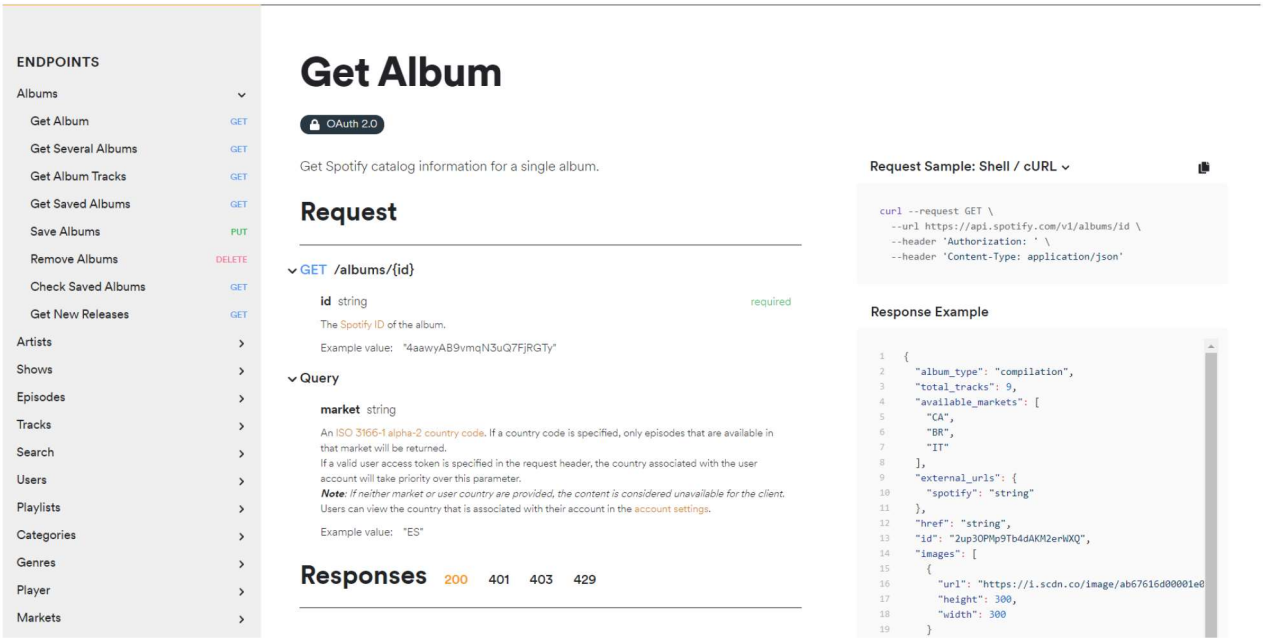
4.2. Spotify dokumentacija

Prije početka testiranja Spotify API-ja poželjno je na internetu pronaći dokumentaciju vezanu za Web API koja uveliko pomaže pri testiranju. Službena otvorena dokumentacija za Spotify Web API nalazi se na stranici https://developer.spotify.com/documentation/web-api/ [10], dok je službena stranica o krajnjim točkama koje se mogu testirati na Spotify-ju stranica https://developer.spotify.com/documentation/web-api/reference/#/ (Slika 4.1) (vidi [11]).



Slika 4.1 – Spotify Web API

Na slici 4.2 nalazi se primjer GET zahtjeva za GET Album, krajnja točka je albums/id, što znači da se šalje zahtjev za ID jednog albuma.



Slika 4.2 – Get Album – primjer unutar dokumentacije

U dokumentaciji, među ostalom, može se pročitati da je https://api.spotify.com osnovna adresa Spotify web API-ja je [12], koji statusni kodovi HTTP-a se mogu očekivati u odgovoru te da za pristup privatnim podacima putem Web API-ja, kao što su korisnički profili i popisi za reprodukciju, aplikacija mora dobiti dopuštenje korisnika za pristup podacima. Autorizacija se vrši putem usluge Spotify Accounts.

4.2.1. Autentifikacija

Pristup resursima mora biti ograničen, tj. ukoliko se radi o resursima osjetljive prirode (osobni podaci, podaci koji nisu javno dostupni itd.) obično se na takve resurse stavlja sustav filtriranja dolaznih zahtjeva prema razini prava pristupa zatraženom resursu.

Takav se sustav naziva autentifikacija, a predstavlja povezivanje autora zahtjeva s korisničkim podacima za provjeru prava pristupa (korisničko ime i lozinka). Navedena provjera prava pristupa odvija se na temelju podataka u bazi registriranih korisnika, usporedbom podataka iz baze s podacima koje je korisnik priložio uz zahtjev (vidi [13]).

Ukoliko je autentifikacija neuspješna zahtjev se odbija, uz slanje koda greške 401 ili 403 kao odgovor (Tablica 2.1).

Svi zahtjevi za Spotify Web API zahtijevaju provjeru autentičnosti. To se postiže slanjem važećeg OAuth pristupnog tokena u zaglavlju zahtjeva.

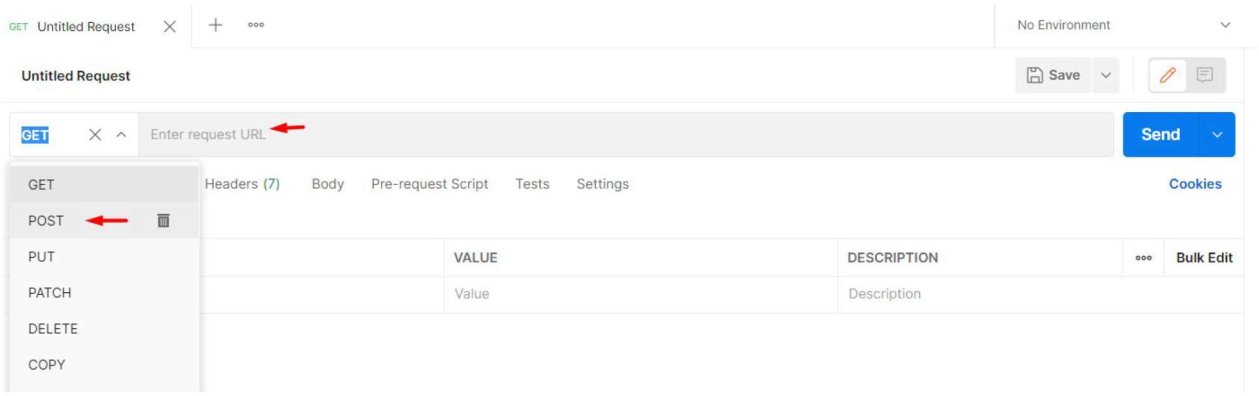
Postoje i drugi načini autentifikacije koji ovise od projekta do projekta i od načina na koji je poslužitelj posložen. Zbog toga je uvijek najbitnije pomno pročitati dokumentaciju jer je upravo tamo opisan način autentifikacije za taj projekt.

4.3. Scenarij testiranja

U ovom testnom scenariju opisano je dohvaćanje albuma određenog umjetnika, zapraćivanje tog umjetnika i prestanak praćenja istog. Ovaj skup testova predstavlja određene vrlo česte operacije koje se izvode u aplikaciji Spotify i primjer je tipičnog korisničkog iskustva pri korištenju aplikacije. Kao nekih od najčešćih operacija koji korisnici izvršavaju ove krajnje točke idealne su za stvaranje testnog scenarija u svrhu ovog rada.

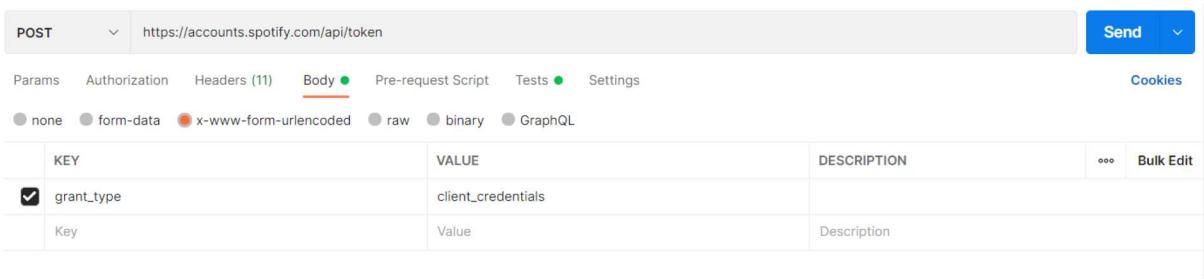
4.3.1. Spotify – slanje pristupnog tokena (POST zahtjev)

Budući da svi zahtjevi za Spotify Web API zahtijevaju provjeru autentičnosti, prvo klijent mora poslati svoj važeći OAuth pristupni token u zaglavlju zahtjeva. Klijent mora napraviti POST zahtjev "https://accounts.spotify.com/api/token" URL-u. S padajućeg izbornika klijent odabire POST te u polje unutar kojeg piše "Enter request URL" upisuje URL zahtjeva (Slika 4.3).



Slika 4.3 – Vrsta zahtjeva : POST

Tijelo ovog POST zahtjeva mora sadržavati sljedeće parametre kodirane u aplikaciji/x-www-form-urlencoded: te par ključ/vrijednost "*grant_type*":"*client_credentials*“ [14] (Slika 4.4).

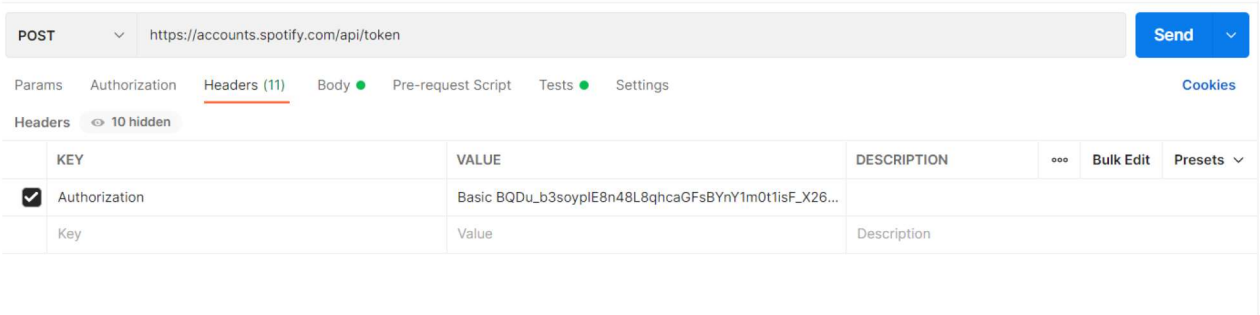


Slika 4.4 – Tijelo POST zahtjeva s parom ključ/vrijednost

Zahtjev mora sadržavati sljedeće HTTP zaglavlje: par ključ/vrijednost "*Authorization*":"*Basic {Token}"* (Slika 4.5). Token je Base64 kodirani niz koji sadrži ID klijenta i tajni ključ klijenta (vidi [14]).

ID klijenta i tajni ključ klijenta mogu se naći na stranici https://developer.spotify.com/dashboard/, ali samo ako klijent ima svoj Spotify račun (vidi [15]).

Nakon što iskopira svoj ID i tajni ključ, može ih enkodirati u Base64 format na stranici https://www.base64encode.org/ (vidi [16]).



Slika 4.5 – Zaglavlje POST zahtjeva s parom ključ/vrijednost

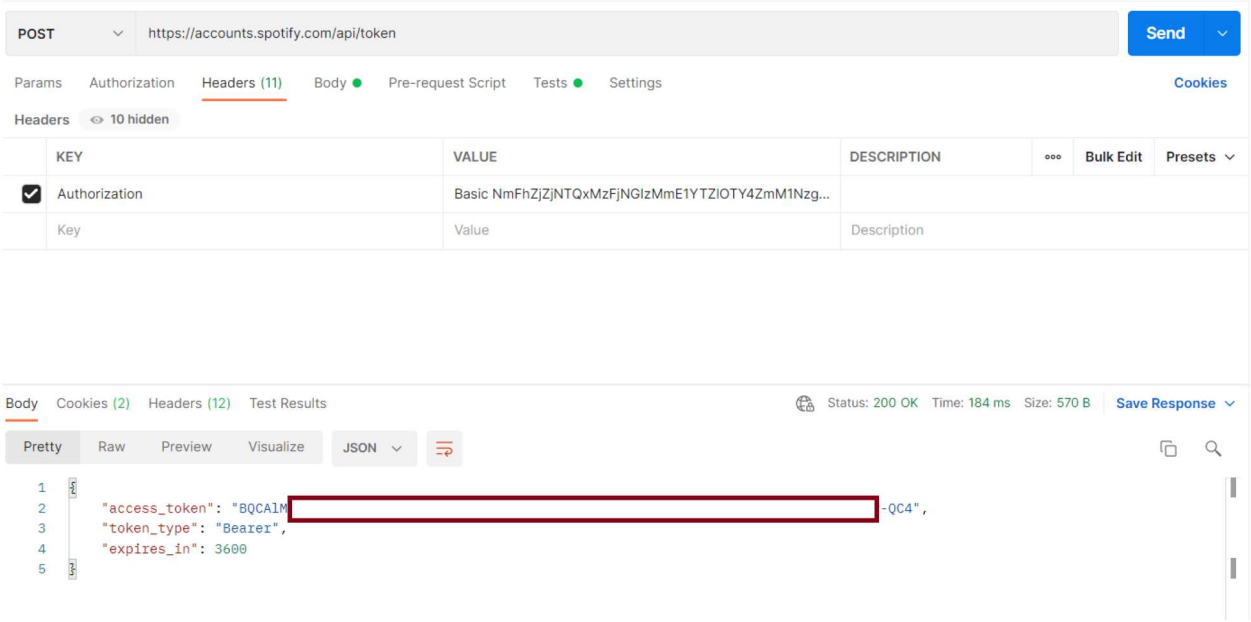
Nakon što su upisani svi potrebni parovi ključ/vrijednost u tijelo i zaglavlje, klikom na dugme Pošalji klijent šalje zahtjev Spotify API-ju. Slika 4.6 pokazuje kako izgleda uspješan odgovor.

Na dnu Postman-a klijent bi trebao vidjeti tijelo odgovora na API zahtjev u JSON formatu. U tijelu se nalazi pristupni token (eng. *Access token*) kojeg je klijent i tražio preko POST zahtjeva.

Osim ovog pristupnog tokena bitno je naglasiti i vrstu tokena koja se šalje pri autorizaciji. Ključne riječi koje označavaju dvije glavne vrste tokena koji se na ovaj način šalju su:

- Basic: Osnovni token dobiven slanjem pristupnih podataka koji se šalje pri prvotnom zahtjevu za autorizaciju, kao odgovor na ovaj zahtjev dobiva se pristupni token koji se uvrštava u sve buduće zahtjeve koji traže autorizaciju
- Bearer: Glavni način autorizacije nakon prvotnog zahtjeva, koristi se u svim daljnjim zahtjevima

Na Slici 4.6 u tijelu odgovora vidljiv je tip tokena koji je Bearer i vremenski period (u sekundama) za koji vrijedi pristupni token. Klijent također vidi s desne strane HTTP statusni kod, koji je u ovom primjeru jednak "200 OK" što znači da je zahtjev bio uspješan (Tablica 2.1), vrijeme koliko dugo je trajalo da se zahtjev završi i veličinu podataka u odgovoru u kB.



Slika 4.6 – Odgovor na uspješan zahtjev

4.3.2. Spotify - GET zahtjevi

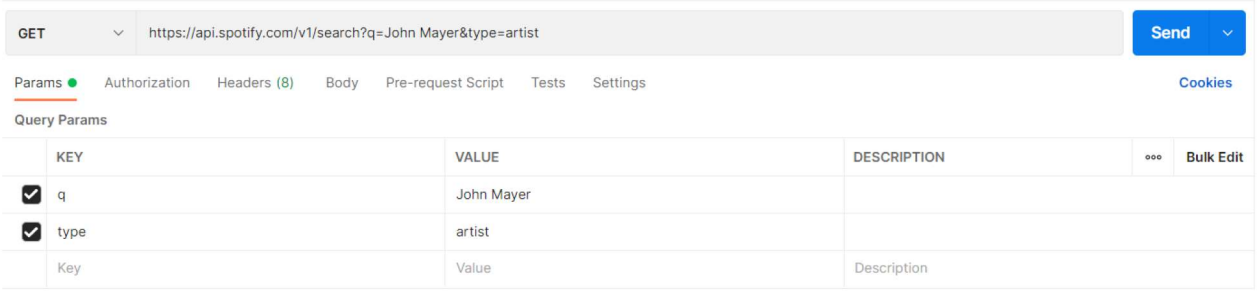
Slanjem GET zahtjeva s endpointom /artists/{id}/albums žele se dohvatiti podaci o svim albumima izabranog umjetnika. {Id} u ovom endpointu odnosi se na Id umjetnika koji je jedinstven. Nakon što se napravi POST zahtjev, klijent koristi Bearer token što je dobio u tijelu odgovora na POST zahtjevu te ga koristi za daljnju autentifikaciju u zaglavljima.

4.3.2.1. Spotify – dohvaćanje Id-ja umjetnika (GET zahtjev)

Prije nego klijent pošalje GET zahtjev za albume određenog umjetnika, mora prvo poslati GET zahtjev za jedinstveni ID željenog umjetnika. S padajućeg izbornika klijent odabire GET te u polje unutar kojeg piše "Enter request URL" upisuje osnovni URL zahtjeva: "https://api.spotify.com/v1/search" (Slika 4.7) (vidi [11]).

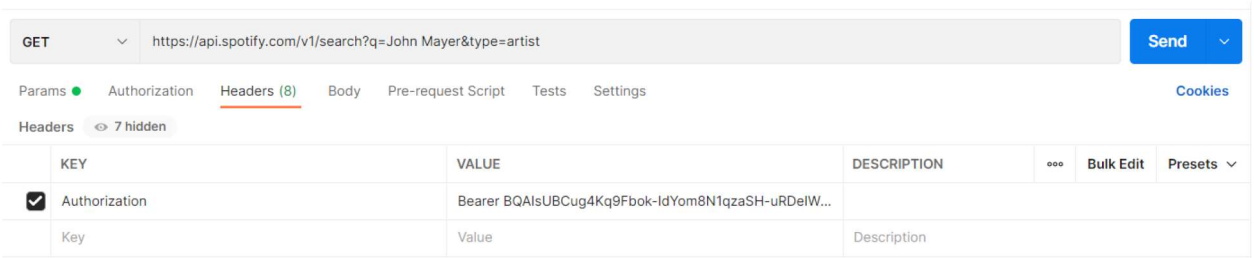
Na slici se vide parametri u koje je klijent dodao parove ključ/vrijednost, "*q*":"*{ArtistName}*", "*type*":"*{Type}*". Ključ "*q*" označava upit za pretraživanje, u vrijednost se upisuje točan pojam pretrage kako bi se suzio izbor pretraživanja. Kod ključa "*type*", u vrijednost se mogu upisati samo određene vrijednosti. Dozvoljene vrijednosti su "*album*", "*artist*", "*playlist*", "*track*", "*show*" i "*episode*". Budući da je klijent upisao par "*type*":"*artist*", API u odgovoru vraća samo umjetnike koji sadržavaju "*John Mayer*" u svom imenu, odnosno da klijent nije upisao ovaj

par ključ/vrijednost API bi u svom odgovoru imao i albume, pjesme, epizode koji u svom imenu sadržavaju "*John Mayer*".



Slika 4.7 – Parametri GET zahtjeva s parovima ključ/vrijednost

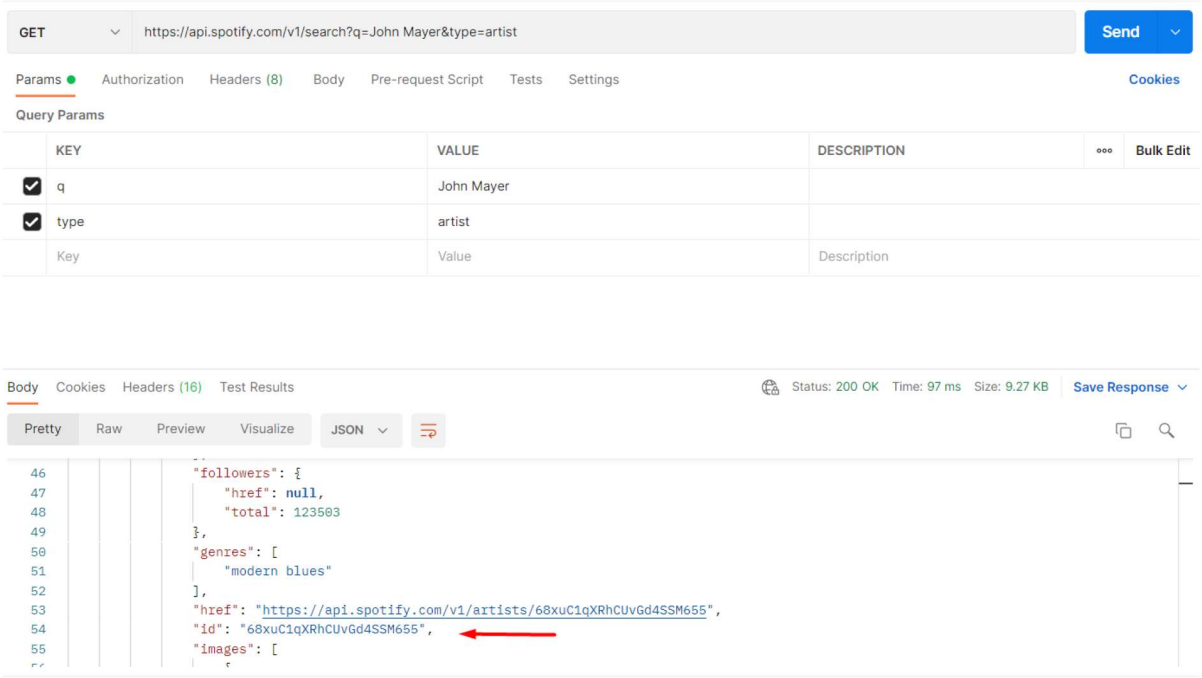
Zahtjev mora sadržavati sljedeće HTTP zaglavlje: par ključ/vrijednost, "*Authorization*":"*Bearer {Token}*" (Slika 4.8).



Slika 4.8 – Zaglavlje GET zahtjeva s parom ključ/vrijednost

Nakon što su upisani svi potrebni parovi ključ/vrijednost u parametrima i zaglavlju, klikom na dugme Pošalji klijent šalje zahtjev Spotify API-ju. Slika 4.9 pokazuje kako izgleda uspješan odgovor.

Na dnu Postman-a klijent bi trebao vidjeti tijelo odgovora na API zahtjev u JSON formatu. U tijelu se nalazi umjetnik pod imenom John Mayer te njegov jedinstveni Id po kojem će klijent tražiti njegove albume. Klijent također vidi s desne strane HTTP statusni kod, koji je u ovom primjeru jednak "200 OK" što znači da je zahtjev bio uspješan (Tablica 2.1).



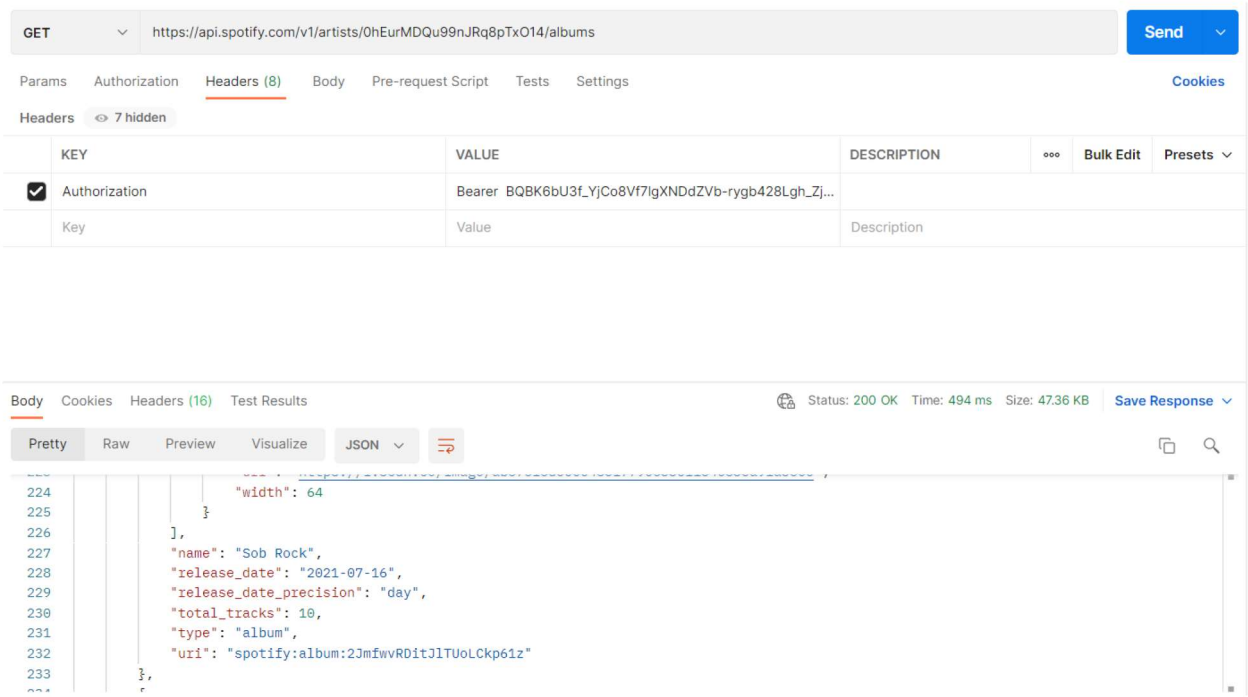
Slika 4.9 – Odgovor na uspješan zahtjev

4.3.2.2. Spotify – traženje albuma prema id-ju umjetnika (GET zahtjev)

S padajućeg izbornika klijent odabire GET te u polje unutar kojeg piše "Enter request URL" upisuje URL zahtjeva: "https://api.spotify.com/v1/artists/{id}/albums" [11], gdje je Id unutar URL-a jednak ID-ju koji je dobiven u prethodnom GET zahtjevu.

Zahtjev mora sadržavati sljedeće HTTP zaglavlje: par ključ/vrijednost "*Authorization*":"*Bearer {Token}*" koji je dobiven u POST zahtjevu (Slika 4.10).

Nakon što su upisani svi potrebni parovi ključ/vrijednost u zaglavlju, klikom na dugme Pošalji, klijent šalje zahtjev Spotify API-ju. Slika 4.10 pokazuje kako izgleda uspješan odgovor.



Slika 4.10 – Odgovor na uspješan GET artists/{id}/albums zahtjev

4.3.3. Spotify – slanje zahtjeva za praćenje umjetnika (PUT zahtjev)

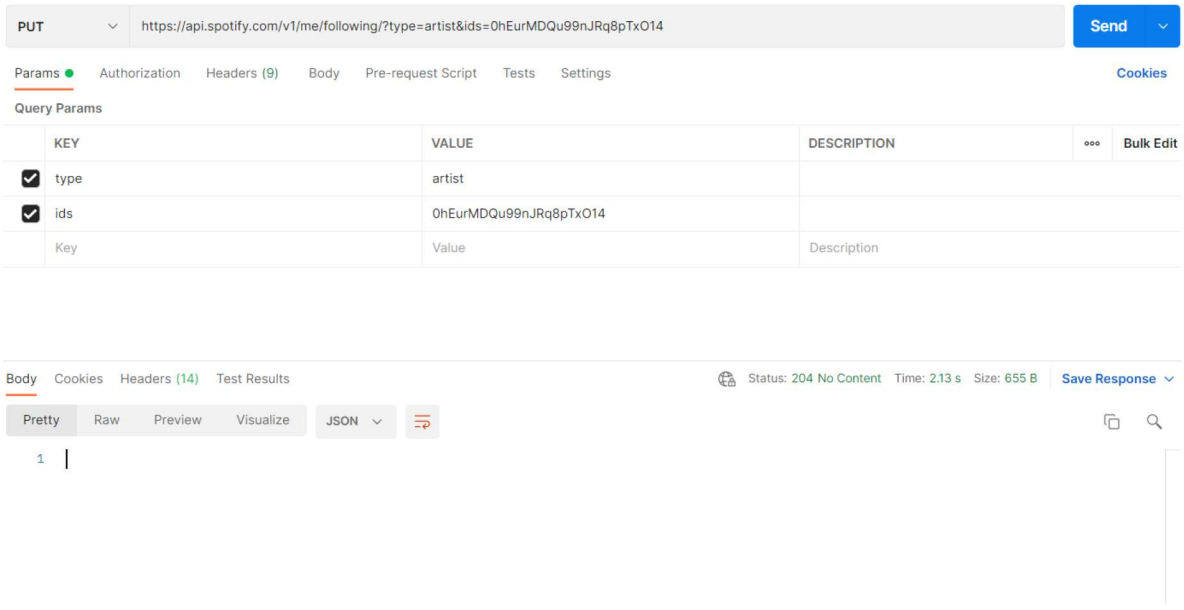
Slanjem PUT zahtjeva s endpointom /me/following mogu se urediti podaci na serveru, odnosno klijent kao korisnik Spotify-a može slanjem PUT zahtjeva na URL "https://api.spotify.com/v1/me/following/?type={Type}&ids={Id}" zapratiti umjetnika ("*type*":"*artist*") ili drugog korisnika ("*type*":"*user*") ako poznaje njihov Id.

S padajućeg izbornika klijent odabire PUT te u polje unutar kojeg piše "Enter request URL" upisuje URL zahtjeva: https://api.spotify.com/v1/me/following (vidi [11]).

Zahtjev mora sadržavati sljedeće HTTP zaglavlje: par ključ/vrijednost "*Authorization*":"*Bearer {Token}*" koji je dobiven u POST zahtjevu.

Na slici se vide parametri u koje je klijent dodao parove ključ/vrijednost "*type*":"*artist*", "*ids*":"*{Id}*", gdje je Id jednak Id-ju koji je dobiven u prijašnjem GET zahtjevu.

Nakon što su upisani svi potrebni parovi ključ/vrijednost u parametrima i zaglavlju, klikom na dugme Pošalji, klijent šalje zahtjev Spotify API-ju. Slika 4.11 pokazuje kako izgleda uspješan odgovor. S desne strane nalazi se HTTP statusni kod, koji je u ovom primjeru jednak "204 No Content" što znači da je server uspješno obradio zahtjev, ali ne vraća nikakav sadržaj kao što se može vidjeti u tablici 2.1.



Slika 4.11 – Odgovor na uspješan PUT zahtjev

4.3.4. Spotify – slanje zahtjeva za prestanak praćenja umjetnika (DELETE zahtjev)

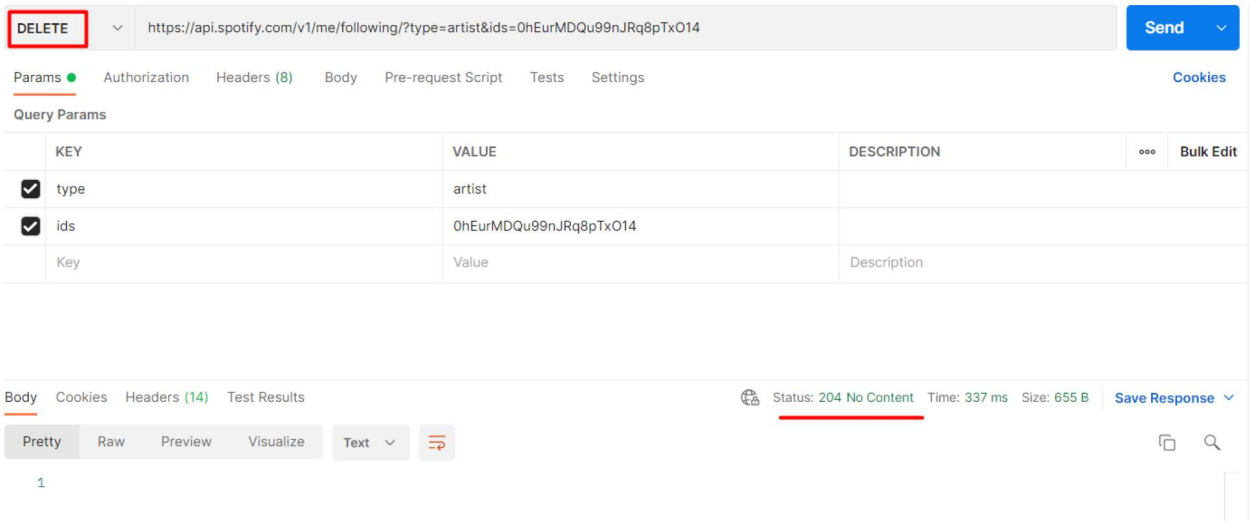
Slanjem DELETE zahtjeva s endpointom /me/following mogu se obrisati podaci na serveru, odnosno klijent kao korisnik Spotify-a može slanjem DELETE zahtjeva na URL https://api.spotify.com/v1/me/following/?type={Type}&ids={Id} prestati pratiti umjetnika ("*type*":"*artist*") ili drugog korisnika ("*type*":"*user*") ako poznaje njihov Id.

S padajućeg izbornika klijent odabire DELETE te u polje unutar kojeg piše "Enter request URL" upisuje URL zahtjeva: https://api.spotify.com/v1/me/following (vidi [11]).

Zahtjev mora sadržavati sljedeće HTTP zaglavlje: par ključ/vrijednost "*Authorization*":"*Bearer {Token}*" koji je dobiven u POST zahtjevu.

Na slici se vide parametri u koje je klijent dodao parove ključ/vrijednost "*type*":"*artist*", "*ids*":"*{Id}*", gdje je Id jednak Id-ju koji je dobiven u prijašnjem GET zahtjevu.

Nakon što su upisani svi potrebni parovi ključ/vrijednost u parametrima i zaglavlju, klikom na dugme Pošalji, klijent šalje zahtjev Spotify API-ju. Slika 4.12 pokazuje kako izgleda uspješan odgovor. S desne strane nalazi se HTTP statusni kod, koji je u ovom primjeru jednak "204 No Content" što znači da je server uspješno obradio zahtjev, ali ne vraća nikakav sadržaj kao što se može vidjeti u tablici 2.1.



Slika 4.12 – Odgovor na uspješan DELETE zahtjev

5. Automatizirani testovi u Postman-u

5.1. Zašto se pišu automatizirani testovi?

Jedan od načina testiranja softvera jest automatizirano testiranje. Glavni razlog uvođenja automatizacije u postupak testiranja je olakšavanje izvršavanja testova koji se ponavljaju više puta u razvojnom procesu. Takvim tzv. regresijskim testovima utvrđuje se funkcionira li prethodno razvijeni dio aplikacije nakon nadogradnje ili implementacije novog dijela koda jednako kao i prije.

Regresijski se testovi izvršavaju nakon svakog razvojnog ciklusa kako ne bi došlo do nazadovanja u ponašanju aplikacije (eng. *regression*). Obzirom na učestalo izvršavanje, ovakvo testiranje može biti iscrpljujuće i zahtijevati puno vremena i ljudskih resursa. Zbog toga se regresijski testovi često automatiziraju putem posebnih alata kako bi se kontrolirano izvršavali nakon svakog razvojnog ciklusa.

Prednosti automatiziranog testiranja:

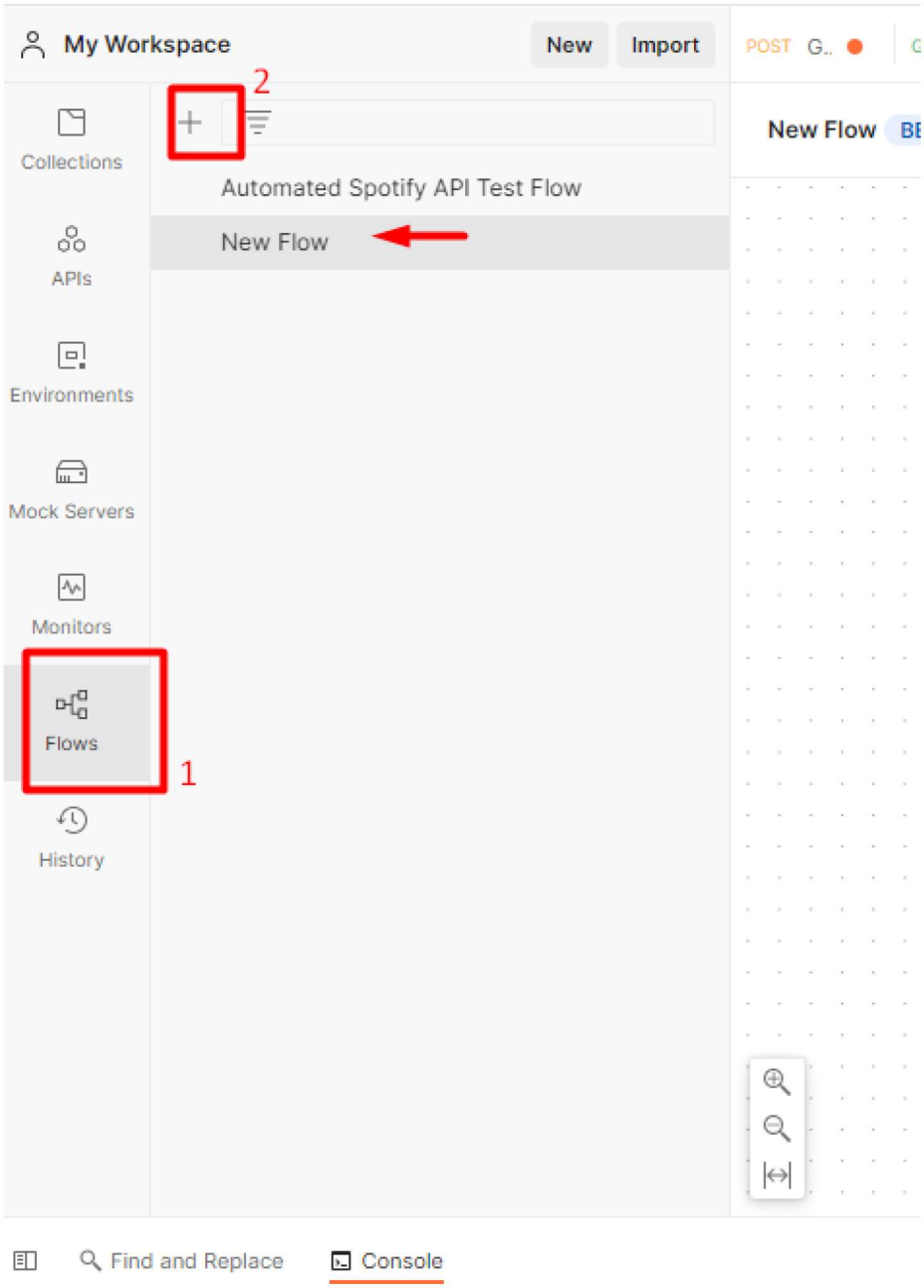
- Brže izvršavanje testova
- Ušteda na ljudskim resursima i vremenu
- Kontrolirana priroda izvršavanja testova (u točno vrijeme pod istim uvjetima)
- Lakši postupak dokumentiranja rezultata testova nakon njihovog izvršavanja (jedan dokument za cijeli postupak regresijskog testiranja) [20]

5.2. Postupak automatiziranja testova u Postman-u

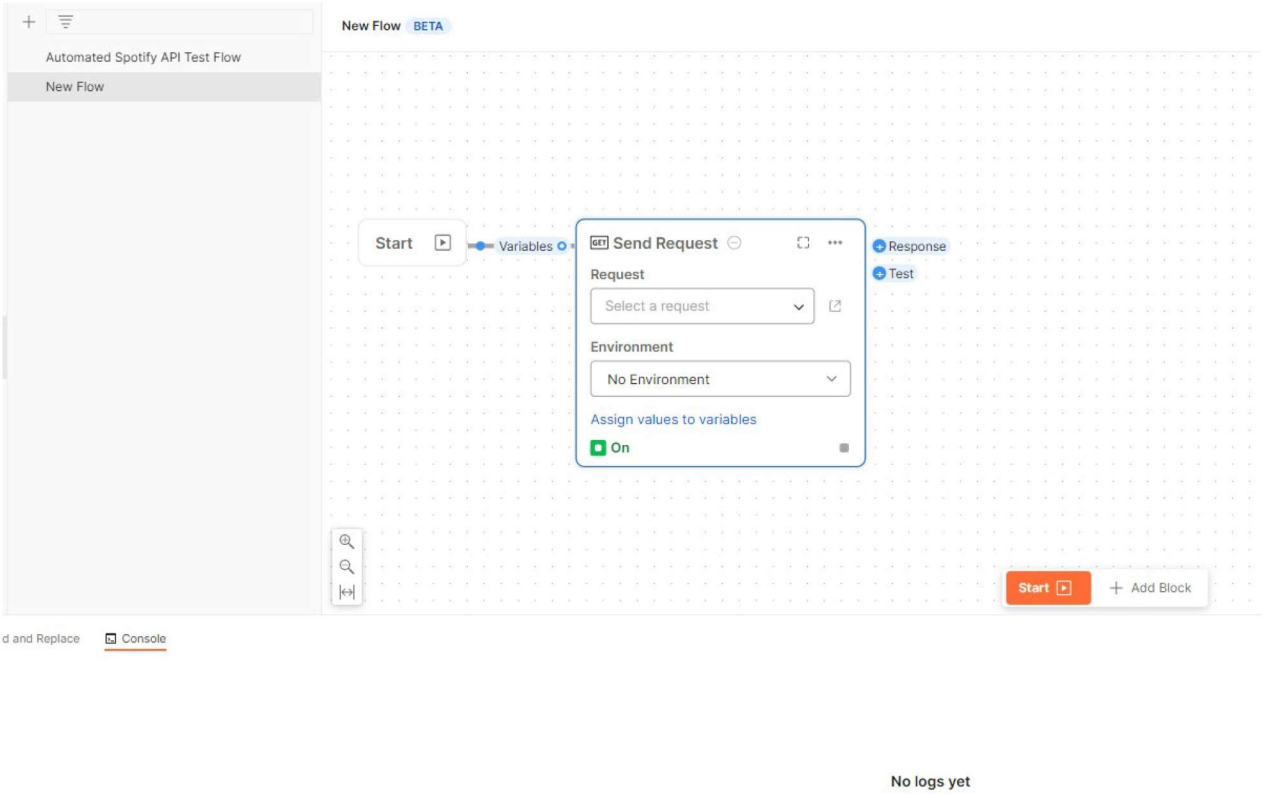
Automatiziranje izvršavanja zahtjeva u Postman-u ostvaruje se korištenjem tzv. protoka (eng. *Flow*), koji su zapravo skupovi ulančanih zahtjeva koji se izvršavaju jedan iza drugog. Pri ovakvom izvršavanju vidljivi su statusni kodovi svih odgovora na ove zahtjeve te je lako uočiti što je pogreška ukoliko do nje dođe. Ovakvo grupiranje zahtjeva u cjeline olakšava izvršavanje testova te daje jasan pregled njihovih rezultata.

U lijevom izborniku klikom na opciju "Protok" (Slika 5.1) otvara se pregled svih stvorenih protoka. Opcija "+" u gornjem dijelu ovog pregleda stvara novi protok. Nakon što se novi protok

otvori može se krenuti s dodavanjem pojedinih zahtjeva i njihovim povezivanjem. Na slici 5.2 vidljiv je prozorčić za dodavanje novog zahtjeva u protok. Njegovi ključni elementi su padajući izbornik za odabir željenog zahtjeva u protok te drugi padajući izbornik za odabir okruženja. Kad se ostvari željeni broj zahtjeva, pritiskom na gumb "Start" počinje izvršavanje zahtjeva na potpuno automatski način i njihovo stanje može se vidjeti u konzoli na dnu prozora (vidi sliku 5.2).



Slika 5.1 – Stvaranje novog protoka

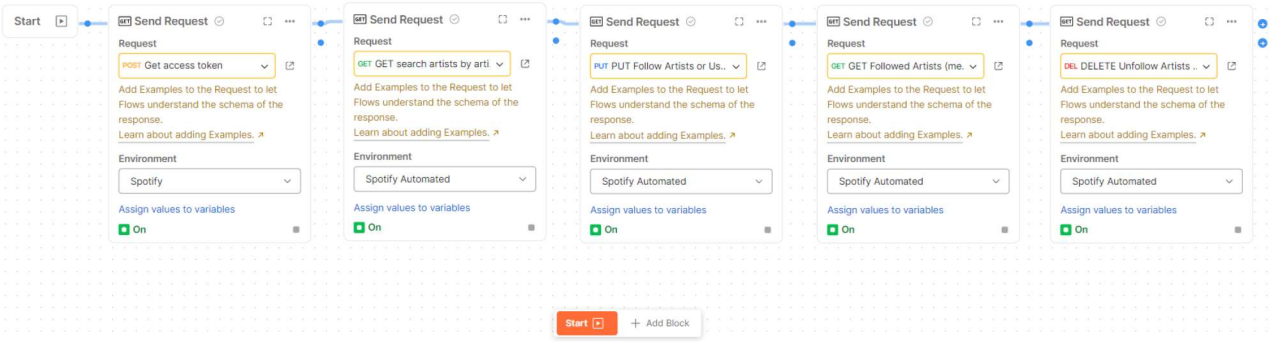


Slika 5.2 – Stvaranje prvog zahtjeva u protoku

5.3. Primjer automatiziranog testiranja Spotify API-ja

U protoku na slici 5.3 spojeno je pet zahtjeva u zajednički testni scenarij, koji započinje POST zahtjevom za dobivanje pristupnog tokena. Nakon dobivanja pristupnog tokena mogu se raditi ostali zahtjevi. Na POST zahtjev nadovezuje se GET zahtjev za dohvaćanje umjetnika po imenu. Potom se izvršava PUT zahtjev za praćenje dohvaćenog umjetnika s prethodnog zahtjeva, šalje se GET zahtjev koji vraća sve popraćene umjetnike i zaključno se izvršava DELETE zahtjev za prestanak praćenja umjetnika koji je maloprije bio popraćen. Stanje zahtjeva u svakom trenutku je vidljivo u konzoli na slici 5.4.

Na istoj slici također je vidljivo vrijeme potrebno za svaki pojedini zahtjev, osoba koja testira noviju verziju API-ja može iskoristiti i ovaj podatak kako bi uočila moguće probleme koji su se pojavili.



Slika 5.3 – Protok sa svim zahtjevima

Slika 5.4 – Konzola prikazuje provjeru valjanosti prema statusnom kodu

6. Zaključak

U ovom diplomskom radu obrađena je tematika pravilnog testiranja API-ja pomoću aplikacije Postman, no prošireni sadržaj rada koristan je za stjecanje shvaćanja o načinu rada API-ja kao takvog.

Opisani su i definirani svi potrebni stručni pojmovi kako bi čitatelj mogao razumjeti tematiku rada, te su pojašnjeni najbitniji čimbenici svakog od tih pojmova. Počevši od samog pojma API-ja za čije je razumijevanje potrebno razložiti i opisati REST protokol, HTTP komunikacijske metode i JSON format prijenosa podataka, nastavno s predstavljanjem Postman alata za testiranje API-ja i njegovih najbitnijih elemenata, dana je podloga za razumijevanje ovog složenog postupka. Kao završna cjelina predstavljena je glavna točka samog rada: konkretan primjer testiranja na temelju postojećeg i globalno raširenog Spotify API-ja.

Pojmovi i postupci opisani u radu dovoljni su za ulazak u svijet API testiranja, no svijet testiranja suvremenih aplikacija mijenja se iz dana u dan tako da je nemoguće napisati jedan rad koji će pokriti apsolutno sve najsuvremenije načine testiranja i pružiti jedinstven, samodostatan vodič. Uvijek će postojati potreba za trajnim doživotnim učenjem i napredovanjem u ovom području, no ovaj rad predstavio je potrebno znanje za razumijevanje osnova trenutno aktualnih i modernih postupaka testiranja API-ja putem aplikacije Postman.

Literatura

[1] Introducing JSON
URL: https://www.json.org/json-en.html
(Pristupljeno: 2. listopada 2021.)

[2] Format JSON Control
URL: https://pcf.gallery/format-json-control/
(Pristupljeno: 2. listopada 2021.)

[3] JSON
URL: www.fer.unizg.hr/_download/repository/OR_2019_20_Predavanje_08_JSON.pdf
(Pristupljeno: 2. listopada 2021.)

[4] REST API Tutorial
URL: https://restfulapi.net/
(Pristupljeno: 9. listopada 2021.)

[5] Wersterveld, Dave. API Testing and Development with Postman. Published by Packt Publishing Ltd. First published: April 2021

[6] REST API
URL: https://www.astera.com/wp-content/uploads/2020/01/rest.png
(Pristupljeno: 9. listopada 2021.)

[7] Službena stranica Postman-a
URL: https://www.postman.com/
(Pristupljeno: 23. listopada 2021.)

[8] Vodič za Postman

URL: https://hr.myservername.com/postman-tutorial-api-testing-using-postman

(Pristupljeno: 23. listopada 2021.)

[9] Kako testirati i igrati se s web API-jevima

URL: https://hr.ilusionity.com/1456-how-to-test-and-play-with-web-apis-the-easy-way-with-postman

(Pristupljeno: 6. studeni 2021.)

[10] Spotify Web API dokumentacija

URL: https://developer.spotify.com/documentation/web-api/

(Pristupljeno: 13. studeni 2021.)

[11] Spotify Web API dokumentacija – krajnje točke

URL: https://developer.spotify.com/documentation/web-api/reference/#/

(Pristupljeno: 13. studeni 2021.)

[12] Osnovna adresa Spotify web API-ja

URL: https://api.spotify.com

(Pristupljeno: 13. studeni 2021.)

[13] Django REST Framework framework

URL: https://www.django-rest-framework.org/

(Pristupljeno: 13. studeni 2021.)

[14] Spotify Web API dokumentacija – autentifikacija

URL: https://developer.spotify.com/documentation/general/guides/authorization/code-flow/

(Pristupljeno: 20. studeni 2021.)

[15] Spotify Web API dokumentacija – nadzorna ploča
URL: https://developer.spotify.com/dashboard/
(Pristupljeno: 20. studeni 2021.)

[16] Base64 encode
URL: https://www.base64encode.org/.
(Pristupljeno: 20. studeni 2021.)

[17] Spotify stranica
URL: https://www.spotify.com/us/
(Pristupljeno: 4. prosinac 2021.)

[18] Complete list of HTTP Status Codes
URL: https://umbraco.com/knowledge-base/http-status-codes/
(Pristupljeno: 7. prosinac 2021.)

[19] Base64 Encoding & Decoding Examples
URL: https://www.woolha.com/tutorials/deno-base-64-encoding-decoding-examples
(Pristupljeno: 8. prosinac 2021.)

[20] Automatizacija testiranja
URL: http://mrkve.etfos.hr/pred/ozm/si/sem12.pdf
(Pristupljeno: 8. prosinac 2021.)

Životopis

Marijana Krešo rođena je 9.6.1992. u Frankfurtu am Main, a živi u Osijeku gdje je završila osnovnu školu i jezičnu gimnaziju. 2018. godine upisala se na diplomski studij fizike i informatike na Odjelu za fiziku, Sveučilišta Josipa Jurja Strossmayera u Osijeku.