Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/09BE13Lupic

PRIMENA APIGEE PLATFORME ZA UPRAVLJANJE API-JEM

APPLICATION OF APIGEE PLATFORM FOR API MANAGEMENT

Aleksandar Lupić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U ovom radu objašnjeni su osnovni koncepti API menadžmenta i komponente platforme Apigee za upravljanje aplikacionim programskim interfejsima. Prikazana je arhitektura platforme i REST principi na koje se ona oslanja. Predstavljene su najčešće korišćene polise za upravljanje zahtevima, kao i osnovni bezbednosni principi sa akcentom na OAuth framework. Opisani su koncepti deljenih tokova, keširanja i rukovanja greškama i kroz implementaciju rešenja prikazan je rad u okviru platforme za rešavanje problema u praksi.*

Ključne reči: *API menadžment, proksi, REST, polisa, bezbednost, OAuth.*

Abstract – *This paper explains the basic concepts of API management and core components of the Apigee API development platform. Platform architecture as well as REST principles it relies on are displayed. Most used policies for request management are presented, along with the fundamental security principles with OAuth framework covered in detail. Shared flows, caching and error handling concepts are described. In the end is demonstrated a solution of a real-life problem implemented on the platform.*

Keywords: *API management, proxy, REST, policy, security, OAuth.*

1. UVOD

Pojam aplikacionog programskog interfejsa, u terminologiji poznatijeg po skraćenici API (*Application Programming Interface*) označava interfejs kojeg softverski program predstavlja ostalim programima na internetu u svrhu međusobne saradnje i prenosa podataka između aplikacija. API-ji su gradivni elementi koji zajedno omogućavaju interoperabilnost velikih veb-baziranih poslovnih platformi.
Olakšavaju rad poslovnim subjektima i omogućavaju komunikaciju između aplikacije i servera bez posredstva korisnika. Platforme za upravljanje API-jima jesu *API Management* platforme – alati koji predstavljaju proksi za zahteve korisnika, čija je uloga da štite *backend* servisa ili aplikacije od neželjenog ponašanja i zloupotreba, poput prevelikog broja pristiglih upita u kratkom vremenskom periodu ili neautorizovanog pristupa. Predmet ovog rada jeste *Google*-ova platforma *Apigee*, a kroz primere i implementaciju analizirane su njene funkcionalnosti, pravila i polise.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.

1.1. Osnovni koncepti APIGEE platforme

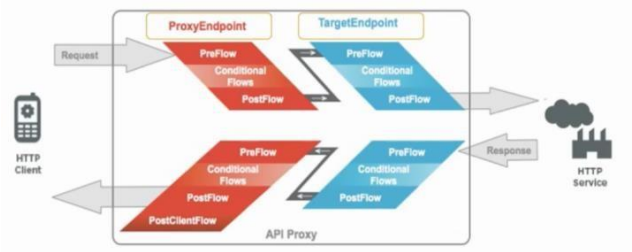
API platformu čine tri osnovne komponente specifičnih funkcionalnosti – *Apigee Edge*, *Apigee Sense* i *Apigee Monetization*. *Apigee Edge* čine servisni, razvojni i sloj za analitiku [1]. Osnovna komponenta u okviru servisnog sloja jeste *Gateway*, čija je uloga rutiranje zahteva od klijenta ka *backend*-u. Ostale komponente arhitekture sistema jesu korisnički interfejsi (UI) za administratore i razvojne portale, infrastrukturni servisi za podatke za analitiku i upravljački server koji skladišti konfiguraciju API-ja i vodi računa o upravljanju.

Komponente servisa međusobno komuniciraju u cilju izvršavanja funkcionalnosti. Ruter nadolazeće zahteve preusmerava *Message Processor*-u u okviru koga se izvršavaju polise.

Management Server je konfiguracioni fajl za sve API-je. Komponenta *cassandra* skladišti konfiguracije aplikacije, API ključeve i OAuth tokene, dok se u *zookeeper*-u čuvaju podaci o konfiguraciji servisa. U okviru servisa za analitiku nalaze se komponente *qpidd* i *Postgres* server i *postgreSQL* baza podataka za upravljanje podacima o analitici.

Developer Portal i *mySQL* baza podataka su unutar razvojnog sloja i služe za izlaganje API dokumentacije, registraciju eksternih programera i njihovih aplikacija. *Apigee* organizaciju čine programeri, aplikacije, API proizvodi i proksiji koje kreira API tim.

U okviru organizacije smeštaju se polise – koraci u proksi toku. Svaki proksi ima dva *endpoint*-a – *ProxyEndpoint* i *TargetEndpoint* i prikazani su na slici 1. Oni se sastoje od tokova (*flows*) i putanja. Postoje tri tipa tokova – *PreFlow*, jedan ili više *ConditionalFlow*-ova i *PostFlow*.



Slika 1. *Proksi tokovi*

Upotrebom REST (*Representational State Transfer*) principa pri pristupu i radu sa resursima, API-ji postaju razumljivi i jednostavni za korišćenje. Za upravljanje CRUD operacijama koriste se standardni HTTP glagoli – GET, POST, PUT i DELETE. Uz odgovor i *payload* vraća se odgovarajući status kod.

2. RAZVOJ PROKSI SERVERA

Kreiranje *OpenAPI* specifikacije najčešće je prvi korak razvoja proksija. U okviru specifikacije navode se resursi kojima je moguće pristupiti, definišu putanje, metode, format odgovora i status kodovi. Specifikaciju je moguće pisati u JSON ili YAML formatu. Pre implementacije polisa, potrebno je poznavati koncept uslova, koji omogućavaju restrikciju izvršavanja polisa. Njihov rezultat je *boolean* vrednost, pa se telo uslova izvršava ukoliko je njegova vrednost `true`. Pri pisanju uslova, veoma je česta upotreba šablona (*patterns*). Najčešće korišćeni šabloni su *Matches*, *JavaRegex* i *MatchesPath* [2]. Pravila rutiranja omogućavaju rutiranje zahteva ka različitim *Target Endpoint*-ima – jednom ili više njih. Jedna aplikacija na taj način može doći do informacija sa više različitih servera. Prvo pravilo rutiranja zove se *no- route*, gde sistem vraća podatke bez usmeravanja na *backend* server, a poslednje pravilo jeste podrazumevana (*default*) ruta, kada su sva prethodna pravila ruta bila neuspela. Podrazumevana ruta mora biti navedena poslednja.

2.1. Polise

U API menadžmentu, polise predstavljaju module koji izvršavaju određenu funkciju. Podesive su i zajedno čine tok. *Apigee Edge* uključuje preko 30 ugrađenih polisa za različite namene. Upotrebom ovih polisa konfiguriše se ponašanje API-ja. Polise se mogu implementirati na više mesta unutar proksija, u zavisnosti od njihove namene. Prema nameni, polise se grupišu u četiri kategorije – polise za kontrolu zahteva, polise za transformacije, bezbednosne polise i polise koje omogućavaju različita programska proširenja.

Spike Arrest polisa štiti *backend* API proksija od *denial of service* napada i sprečava slanje velikog broja zahteva u kratkom vremenskom periodu (*traffic spikes*). Unutar `<Rate>` taga, unosi se vrednost koja predstavlja broj zahteva u određenom vremenskom periodu (minute, sekunde). Opcioni tagovi su `<Identifier>` koji omogućava grupisanje zahteva da bi se polisa primenila u zavisnosti od klijenta, `<MessageWeight>` koji podešava težine za različite klijente i `<UseEffectiveCount>` *boolean* tipa koji, kada je vrednost *true*, distribuira brojač nadolazećih zahteva na preostale *Message Processor*-e. *Quota* polisa ograničava broj API poziva na određeni vremenski period. Može biti primenjena na sve učesnike u komunikaciji, ili samo na specifični API proizvod ili određenog korisnika. Poput *Spike Arrest*-a koristi se u okviru *PreFlow*-a. Upravlja jednim brojačem bez obzira na broj tokova na koje je primenjena. To znači da ukoliko je *Quota* polisa pod istim imenom ugrađena u više tokova, a dozvoljeni broj zahteva 5, brojač će uvećavati sve instance. Davanje različitih imena različitim instancama polise dovešće do odvojene kontrole protoka zahteva. Ukoliko korisnik u kraćem vremenskom periodu iskoristi odnosno dostigne propisani dozvoljeni broj poziva API-ja, moguće je „oduzeti“ mu određeni broj poziva za propisani period. To se postiže upotrebom *Reset Quota* polise. Po isteku perioda, korisnik ponovo na raspolaganju ima prvobitni, zadati broj zahteva u okviru *Quota* polise. Za unapređivanje performansi API-ja koristi se *Response Cache* polisa. Ova polisa kešira čitav HTTP odgovor sa *backend*-a. Koristi se kada se ne očekuju veće promene podataka u određenom periodu. Uglavnom se koristi pri GET pozivima, pri preuzimanju podataka, i ugrađuje se u tokove zahteva i odgovora u API proksiju.

Polise za transformacije služe za transformaciju zahteva i odgovora pri izvršavanju proksija. *JSON to XML* i *XML to JSON* su polise koje pretvaraju jedan format zapisa odgovora u drugi, koji zahteva korisnik. Nakon ove polise, koristi se *XSL Transform* polisa, koja omogućava primenu XSLT-a na povratnu vrednost u XML-u. *SOAP Message Validation* upotrebljava se za validaciju poruka u SOAP ili nekom drugom formatu. Validira XML na osnovu XSD šeme, SOAP na osnovu WSDL-a i proverava da li su XML i JSON dobro formirani. Najčešće korišćena polisa iz ove grupe jeste *Assign Message* (slika 2). Služi za kreiranje ili izmenu zahteva, odgovora i podešavanje promenljivih toka (*flow variables*), *header*-a, parametara upita, parametara formi i povratnih vrednosti.

```
<AssignMessage async="false" continueOnError="false"
enabled="true" name="Assign-Message-ClientID">
        <DisplayName>
           Assign-Message-ClientID
                 </DisplayName>
        <Properties/>
        <Set>
            <Headers>
                <Header name="clientId">
                      {client-id}
                 </Header>
            </Headers>
        </Set>
        <AssignTo createNew="false"
           transport="http" type="request"/>
</AssignMessage>
```

Slika 2. *Assign Message polisa*

Extract Variables polisa podešava promenljive toka koje se koriste u kasnijem izvršavanju proksija. Polisa izdvaja podatke iz putanje, HTTP *header*-a, parametara forme i upita i JSON/XML povratnih vrednosti.

U situacijama kada ne postoji predefinisana polisa kojom je moguće postići željenu funkcionalnost, koriste se dodatne polise, odnosno *Apigee* proširenja.

Primer takve polise jeste *JavaScript Callout*. U okviru ove polise piše se JavaScript kod koji se izvršava u okviru konteksta API proksi toka. Pristupa promenljivama toka u zahtevu i odgovoru, a dozvoljava i kreiranje novih promenljivih. Kod mora biti unutar *try-catch* bloka, kako bi se sprečila pojava nepoznatih izuzetaka. Najčešće se koristi za rukovanje greškama, kriptografiju ili promenu *target* servera dinamički. Polisa *Service Callout* omogućava slanje zahteva na treće servise, poput Google API-ja ili drugih proksija u okviru *Apigee* platforme.

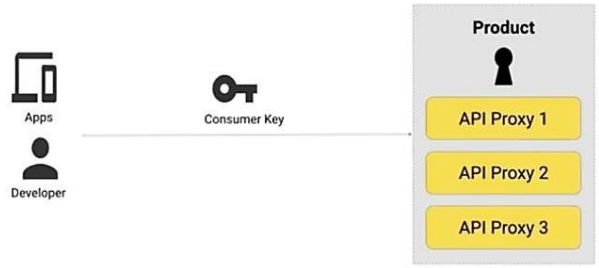
Preporučljivo je kreiranje ovih zahteva uz *Assign Message* polisu, a odgovor nad kojim će se vršiti određene aktivnosti treba preuzeti putem *Extract Variables* polise. Veoma slična prethodnoj polisi je *Flow Callout* polisa, sa razlikom da *Flow Callout* može slati zahteve samo deljenom toku. Deljeni tok mora postojati i mora biti postavljen u okviru iste organizacije i okruženja.

Statistics Collector polisa sakuplja statističke podatke u poruci, poput ID-ja proizvoda, REST akcije, *target* URL-a, dužine poruke i slično. Svi ovi podaci mogu biti smešteni u predefinisanim promenljivama toka ili korisnički definisanim promenljivama.

Za asinhrono logovanje, potrebno je uključiti ovu polisu u poslednji, *PostClientFlow* tok.

3. BEZBEDNOST

Bezbednosni principi spadaju u najvažnije koncepte razvoja i upravljanja aplikacionim programskim interfejsima. I najjednostavniji API-ji zaštićeni su nekom od bezbednosnih polisa, a u radu su detaljno obrađene one najčešće korišćene. *Basic Authentication* polisa enkodira korisničko ime i lozniku na osnovu Base64 šeme, kako bi se poslali sistemu u pozadini. Enkodirane informacije šalju se unutar HTTP *header*-a. Pri preuzimanju kredencijala, putem iste šeme vrši dekodiranje vrednosti iz *header*-a. Verifikaciju API ključeva vrši polisa *Verify API Key*. Da bi ova polisa vršila funkciju, potrebno je prethodno kreirati nekoliko komponenti unutar *Apigee* platforme – API proksi, *Apigee* proizvod, registrovanog *developer*-a i razvojnu aplikaciju. Platforma potom generiše *client ID* (*consumer key*) i *secret* (slika 3).



Slika 3. *Bezbednosna komunikacija*

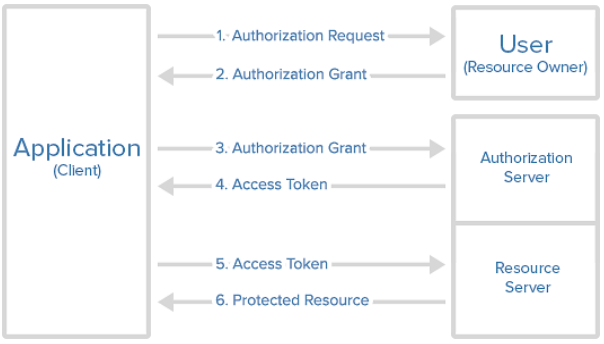
Maliciozni korisnik API-ja i podataka koji on pruža može poslati određeni sadržaj putem HTTP zahteva koji može imati negativan i destruktivan uticaj na servere u pozadini. Napadi poput SQL *Injection* i *JavaScript Injection* mogu dovesti do kompromitovanja ili uništenja velike količine podataka u bazi. Polisa *JSON Threat Protection* specificira ograničenja na različite JSON strukture, poput maksimalnog broja elemenata u nizu ili maksimalne dužine stringa. Ograničenja na XML strukture omogućava *XML Threat Protection* polisa.

Elementi ove polise upotrebljavaju se za različita limitiranja poput dubine i broj čvorova i atributa, ograničenja na dužinu imena, atributa, komentara, URI-ja itd. Od napada na HTTP *header*-e, parametre upita ili telo zahteva štiti polisa *Regular Expression Protection*. Ovom polisom specificiraju se regularni izrazi na osnovu kojih će se evaluirati poruka, što ima za cilj sprečavanje različitih tipova *injection* napada.

OAuth 2.0 autorizacioni *framework* omogućava trećoj aplikaciji limitirani pristup HTTP servisu na račun vlasnika zaštićenih resursa dozvoljavajući interakciju između vlasnika i servisa, ili dozvoljavajući aplikaciji pristup dobije sama [3]. Time se izbegava praksa da korisnik svoje kredencijale otkriva aplikaciji. Tok OAuth- a prikazan je na slici 4. U OAuth sistemu razlikujemo nekoliko uloga.

Vlasnik resursa (*Resource Owner*) ima pristup zaštićenom resursu. Server resursa (*Resource Server*) je zaštićeni server koji prihvata pristupne tokene (*Access Token*). Klijent (*Client*) je aplikacija koja pokušava da pristupi zaštićenom serveru resursa. Može biti mobilna ili klasična veb aplikacija. Aplikacija šalje zahtev zaštićenim resursima i od vlasnika tih resursa mora dobiti dozvolu za pristup.

Autorizacioni server (*Authorization Server*) izdaje pristupne tokene klijentima, nakon uspešne autentifikacije vlasnika resursa. Klijent se autentifikuje pomoću *client ID*-ja i *secret*-a.



Slika 4. *OAuth dijagram toka*

OAuth definiše 4 *grant* tipa. *Grant* tipovi su različiti načini interakcije na koje aplikacija može dobaviti pristupni token. Izbor *grant* tipa vrši se u zavisnosti od potreba, zadataka i ciljeva koje je potrebno ispuniti. Svaki *grant* tip ima svoje prednosti i nedostatke, pa izbor treba vršiti pažljivo i u skladu sa zahtevima. Četiri *grant* tipa su *Client Credentials*, *Authorization Code*, *Resource Owner Password Credentials* i *Implicit grant* tip, i svaki od njih je detaljno objašnjen u radu.

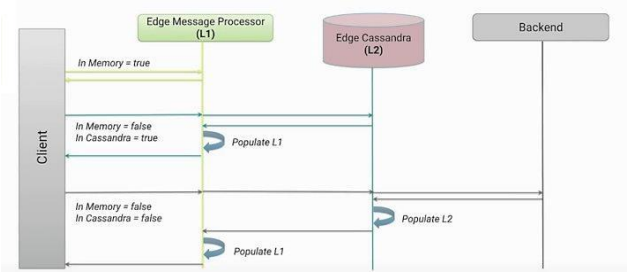
4. DELJENI TOKOVI

Koncept deljenog toka u *Apigee* API menadžmentu predstavlja kolekciju polisa koju je moguće primeniti na sve API proksije u okviru jedne organizacije. Upotrebom deljenih tokova izbegava se redundantnost pri razvoju i implementaciji proksija. Najčešće se primenjuje nad bezbednosnim polisama (*OAuth*), polisama za ograničenje pristupa (*Rate Limiting*) i polisama za transformacije. Deljeni tokovi se uključuju u proksi na dva načina. Jednostavniji i intuitivniji način jeste jednostavnim odabirom *FlowCallout* polise, pri čijoj se konfiguraciji iz padajućeg menija bira prethodno kreirani deljeni tok na osnovu imena. Drugi način podrazumeva upotrebu *flow hook*-ova, specifičnih tačaka u okviru toka. Implementacija ovog koncepta zahteva odgovarajuće permisije i omogućena je samo administratorskim korisnicima, jer kod pomoću kojeg se realizuje prevazilazi standardnu API proksi logiku i koncepte.

5. KEŠIRANJE

Keševi unapređuju performanse eliminišući redundantno procesiranje zahteva i odgovora. Takođe smanjuju količinu nadolazećih zahteva na *backend* servere, što ih čini stabilnijim. Utiču i na skalabilnost, jer podržavaju više transakcije u sekundi bez dodatnog hardvera. Sastoje se iz dva nivoa – *in-memory* (L1) i *persistent* nivoa (L2) [4]. Podaci se u L1 nivou uklanjaju u redosledu nastanka – od najstarijih ka najnovijim, dok u drugom nivou ne postoji ograničenje na broj odnosno količinu keševa. Stavke imaju podešeno vreme isteka i samo na osnovu njega mogu biti uklonjene iz keša. *ResponseCache* polisa kešira čitav HTTP odgovor, uključujući *header*-e i *payload* sa servera u pozadini. Tako smanjuje broj zahteva na *backend*. Takođe omogućava konfigurisanje vreme

života svake stavke u okviru keša. *Populate Cache* i *Lookup Cache* polise omogućavaju pristup deljenom kešu i konfigurisanje keša za detaljniju kontrolu performansi. Svi keširani podaci postaju dostupni u višestrukim tokovima zahteva i odgovora. Na slici 5 prikazan je tok preuzimanja podataka iz keša u zavisnosti od lokacije na kojoj je keš smešten.



Slika 5. *Keširanje*

6. RUKOVANJE GREŠKAMA

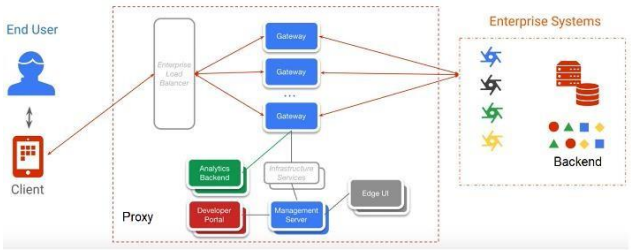
Na *Apigee* platformi postoje dva načina na koje se može pojaviti greška. Prvi način je uključivanjem *RaiseFault* polise u tok, a drugi podrazumeva da *Apigee* objavi poruku o grešci tokom izvršavanja polise. Po detekciji greške, šalje se zahtev komponenti za obradu grešaka (*Fault handler*) takođe zvanog tokom grešaka. Komponenta za obradu grešaka je definisana tagom `<FaultRule>` i ima uslov. Mogu biti implementirane na dve lokacije – *Proxy Endpoint* i *Target Endpoint*. Ukoliko ne postoji *Fault handler*, klijentu se vraća podrazumevana *Apigee* poruka o grešci. Za razliku od pravila ruta, rukovanje greškama funkcioniše na način da *Proxy Endpoint* proverava greške prolazeći kroz elemente implementacije sa dna ka vrhu [5]. Ukoliko unutar `<FaultRules>` elementa ima više stavki, poslednje navedena će biti evaluirana prva. Na *Target Endpoint*-u, greške se evaluiraju intuitivnije, sa vrha ka dnu.

7. IMPLEMENTACIJA REŠENJA

U ovom poglavlju u radu, prikazana je implementacija API proksija u skladu sa specifikacijom i zahtevima, sa detaljno opisanim razvojem i dobijenim rezultatima. Zadatak obuhvata kreiranje API proksija uz predefinisanu *OpenAPI* specifikaciju (*Swagger*).

Proksi treba da prihvata zahteve sa interneta i prosleđuje ih na postojeći servis. Serveri u pozadini moraju biti zaštićeni od učestalog slanja zahteva u kratkom vremenskom periodu i *denial of service* napada. Potrebno je kreirati novu promenljivu koja će se preuzimati iz parametra forme i smeštati u header odgovora.

Nad API proksijem mora biti implementiran bezbednosni princip i CORS funkcionalnost. Rešenje je realizovano upotrebom polisa *Spike Arrest*, *Assign Message*, *Verify API Key* a njihova konfiguracija je opisana u radu. Za potrebe bezbednosnih polisa u posebnim okruženjima kreiran je i novi API proizvod, *developer* i aplikacija. CORS funkcionalnost je implementirana pomoću *Assign Message* polise. Topologija mreže prikazana je na slici 6.



Slika 6. *Topologija mreže*

8. ZAKLJUČAK

U ovom radu detaljno su analizirani i opisani osnovni koncepti API menadžmenta uz oslonac na *Google*-ovu platformu *Apigee Edge*. Prikazana je arhitektura API proksija i tok zahteva i odgovora kao segment komunikacije klijenta i servera u sistemu. Primena API proksija je veoma široka i postali su neizostavan deo digitalnih poslovnih sistema. Omogućavaju kompanijama efikasan način održavanja i čuvanja njihovih podataka i sistema, kao i lakši i brži pristup i upotrebljivost tih podataka od strane korisnika. API-ji čine aplikacije skalabilnim i stabilnim, a pomoću platformi poput *Apigee*-ja, omogućavaju niz funkcionalnosti poput praćenja životnog ciklusa API-ja putem analitike sistema. Kao možda najznačajnija funkcionalnost platformi za upravljanje aplikacionim programskim interfejsima izdvajaju se bezbednosni principi i polise. API-ji su vremenom postali standard za razvoj i povezivanje modernih aplikacija, a ekspanzijom API menadžmenta kao posebne oblasti u okviru informacionih tehnologija očekuje se sve veći broj potencijalnih klijenata i veća potražnja za inženjerima ovog profila u IT sektoru.

9. LITERATURA

[1] What is Apigee Edge? https://docs.apigee.com/api-platform/get-started/what-apigee-edge [Datum pristupa: 28.09.2019.]

[2] Conditions Reference https://docs.apigee.com/api-platform/reference/conditions-reference [Datum pristupa: 04.10.2019.]

[3] Aaron Parecki, *An Introduction to OAuth 2*. O'Reily Webcast, 2012

[4] Apigee: Edge Caching in Detail https://community.apigee.com/articles/1620/apigee-edge- caching-in-detail.html [Datum pristupa: 16.10.2019.]

[5] Handling Faults https://docs.apigee.com/api-platform/fundamentals/fault-handling [Datum pristupa: 18.10.2019.]

Kratka biografija:

Aleksandar Lupić rođen je u Novom Sadu 1995. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2019. godine.
kontakt: aleksandarlupic@gmail.com