Razmjena podataka putem aplikacijsko-programskih sučelja

Omazić, Filip

Sveučilište u Zadru

Odjel za informacijske znanosti

Preddiplomski sveučilišni studij informacijskih znanosti (jednopredmetni - redovni)

Filip Omazić

Razmjena podataka putem aplikacijsko-programskih sučelja

Završni rad

Zadar, 2020.

Sveučilište u Zadru

Odjel za informacijske znanosti

Preddiplomski sveučilišni studij informacijskih znanosti (jednopredmetni - redovni)

Razmjena podataka putem aplikacijsko-programskih sučelja

Završni rad

Student/ica: Filip Omazić

Mentor/ica: Dr. sc. Krešimir Zauder

Zadar, 2020.

Izjava o akademskoj čestitosti

Ja, **Filip Omazić,** ovime izjavljujem da je moj **završni** rad pod naslovom **Razmjena podataka putem aplikacijsko-programskih sučelja** rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Ni jedan dio mojega rada nije napisan na nedopušten način, odnosno nije prepisan iz necitiranih radova i ne krši bilo čija autorska prava.

Izjavljujem da ni jedan dio ovoga rada nije iskorišten u kojem drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Zadar, 10. rujna 2020.

Sadržaj

Sažetak

U ovom radu bit će pojašnjena razmjena informacija putem aplikacijsko-programskih sučelja. Cilj ovog rada je izraditi funkcionalnu web stranicu koja preko aplikacijsko-programskih sučelja dohvaća informacije i daje ih korisniku na uvid kroz stranice web sjedišta. S obzirom na činjenicu da je prednost velikog broja aplikacijsko programskih (API) sučelja to što su besplatni korisnicima, implementacija će biti puno lakša te će pružiti bilo kome tko poznaje programske jezike lakoću izrade sličnog projekta. Ova stranica koristi *Python* programski jezik kao podlogu za poslužitelja implementiranog koristeći Flask biblioteku (modul). Odabran je jednostavan pristup izrade, koji pruža kreiranje kompleksne web stranice na temelju programski jednostavne osnove, a korisniku pristup informacijama u jednostavnom i efikasnom sučelju. Prednost projekta je, uz jednostavnost i brzinu također veličina, jer cijeli projekt zauzima svega nekoliko megabajta dok pruža velik broj primjera i mogućnosti. Svrha je približiti korištenje aplikacijsko-programskih sučelja i konstrukciju web stranice koja koristi aplikacijsko-programska sučelja informacijskim stručnjacima i svim programerima kroz jednostavan kodni temelj. Uz to proučiti formate i načine razmjene podataka te izgled cjelokupne komunikacije u razmijeni podataka putem API sučelja. Efikasnost i brzina aplikacijsko-programskih sučelja omogućuju korisnicima istih da s lakoćom i iznimnom brzinom dohvaćaju formatirane podatke koje kasnije mogu obrađivati ili uz vrlo malo programske obrade proslijediti svojoj web stranici spremne za korisnike.

Ključne riječi

Aplikacijsko programska sučelja (API), razmjena podataka, programiranje, Python, JSON, XML

Uvod

Poznato je da je utjecaj napretka u razmjeni informacija, pogotovo u današnjem tehnološki nastrojenom svijetu izravan utjecaj na razvoj čovječanstva. Glavni izvor informacija i podataka danas više nisu tiskane publikacije, već to mjesto preuzima internet (i to posebno World Wide Web) radi svoje opširnosti i brzine te novih mogućnosti koje daje korisniku. Web je poslužio kao posrednik na kojem se sva znanja, pa bilo iz knjiga, pisama ili pak CD medija i sličnih sredstava za pohranu informacija zapisuju kako bi bila dostupna cijelome svijetu bez potrebe fizičkog prijenosa nositelja informacija. Mrežna razmjena informacija postaje ne samo znatno lakša od svih prijašnjih načina razmjene informacija, već i omogućava svakom čovjeku s pristupom internetu da prouči bilo koje područje ljudskog znanja i poslovanja što donosi revolucionaran progres u poslovnom, informacijskom i edukacijskom svijetu. Ove informacije pohranjene su najčešće u bazama podataka ili u elektroničkim dokumentima raznih formata koji dozvoljavaju spremanje puno veće količine informacija te mogu biti pripremljene za bržu obradu radi strojno čitljivog oblika i strukture. Kako bi se olakšala komunikacija s bazama podataka i uklonila restrikcija na pristup informacijama samo korisnicima koji imaju direktan pristup bazi osmišljena su API sučelja. API, ili Aplikacijsko programsko sučelje omogućuje programerima da s nekoliko linija koda dohvate informacije kako bi ih s lakoćom mogli obraditi. API ubrzo postaje sučelje o kojoj ovisi veliki broj internet servisa kakve danas znamo. Od korporacija i stranica s visokim brojem posjetitelja do malih obiteljskih poslova, bez obzira o tipu stranice, vrlo je vjerojatno da je korišten neki oblik API sučelja u kodu kako bi pružili svojim korisnicima brži i bolji pristup informacijama koje su im potrebne. Ideja API sučelja nije samo pružiti novi način razmjene informacija, već također omogućiti automatizaciju razmjene i učiniti proces dohvaćanja informacija s internet baza podataka lakšim. Današnji stručnjaci su uočili važnost računala i interneta u vidu razmjene podataka. API sučelja su raširena i u znanstvenoj zajednici, kao što je vidljivo pristupom listi besplatnih API sučelja gdje je moguće pronaći sučelja koja nude širok spektar strukturiranih podataka od botaničkih informacija o vrstama biljki do informacija o osnovama astronomije ili pak podaci dohvaćeni od API sučelja koji računalno obrađuju slike koristeći napredne algoritme strojnog učenja za prepoznavanje sadržaja slike. Same informacije moraju biti točne i poticati iz validnih izvora što većina API sučelja i pruža s obzirom na to da se radi o organizacijama koje se bave istim temama čije podatke nudi API poput API sučelja otvorenih informacija vladinih organizacija ili specijaliziranih privatnih tvrtki. Iz ovog razloga korisnici aplikacijsko-programskim

sučeljima daju svoje povjerenje. Ako se uzme u obzir činjenica da se u svijetu znanosti nerijetko radi o obradama velike količine podataka potreban je bolji način od ručnog prolaza kroz veliku količinu podataka te način brže obrade istih. Ukoliko se radi o programskom pristupu podacima radi lakše obrade i formiranja podataka u željeni izlazni format izvrstan način za razmjenu podataka iz baza podataka je upravo korištenje API sučelja.

Razmjena podataka putem aplikacijsko-programskih sučelja

Što je API ?

U vremenu u kojem su računala ključni alat kod velikog broja upravljačkih, organizacijskih i poslovnih procesa, podaci imaju neizmjernu vrijednost te postaju vrijedan resurs. Smješteni u bazama podataka, web sjedištima i zabilježeni u raznim oblicima računalno iskoristive memorije, ti podaci često nisu lako dostupni za razmjenu. Iako razmjena podataka postoji od početka čovječanstva i oslanjala se većinom na rad ljudi i fizičke nosioce informacija, danas najvažniju ulogu u razmjeni i spremanju podataka za čovječanstvo igraju računala. Čovječanstvo danas ima nevjerojatno brz način komuniciranja i razmjene podataka pomoću povezanosti internetom. No s obzirom na količinu podataka ni sam internet nije dosta jer su podaci na njemu neuredno smješteni i velika količina ih je skrivena od korisnika. Strojno čitljivi podaci predstavljaju novi podložak za podatke i daju im veću vrijednost nego ikada kroz strukturiranje za potrebe programske iskoristivosti. Što se računalne razmjene podataka tiče - i to one koje stroj, a i čovjek, mogu pročitati radi svoje elegantnosti i strogo definirane strukture, veliku ulogu u današnjoj razmjeni strukturiranih podataka putem interneta imaju formati poput JSON-a (*javascript object notation)* ili XML-a (*extensible markup language)*, pogotovo kod aplikacijsko-programskih sučelja. Ovakvi formati za razmjenu podataka dovoljno su napredni da se implementiraju kroz poslužitelja u aplikacijsko sučelje kojem će se pristupiti programskim kodom. Ovom idejom osmišljena su i online API sučelja. U ovom smislu, API[1] ili aplikacijsko programsko sučelje je sučelje kojemu korisnik programski pristupa, što mu dozvoljava razmjenu i dohvaćanje podataka kroz sučelje putem datoteka koje poslužitelj nudi preko interneta. API nije samo sučelje i poslužitelj podataka kojem se pristupa programski, već i popis dostupnih podataka kojima omogućuje pristup određenim metodama i načinima. Najčešće ima svojstva poput pristupačnosti, jednostavnosti i kompatibilnosti s raznim programskim pristupima te često omogućuje korisnički odabran prikaz podataka. Ove datoteke najčešće su formatirane i strojno čitljive te predstavljaju datoteku na web sjedištu koju programski kreira ili oblikuje poslužiteljevo računalo. API sučelje sastoji se također od funkcija i procedura koje omogućuju korisnicima pristup podacima nižih razina odabrane teme koje su sami definirali. API sučelja ubrzavaju izradu projekata i realizaciju ideja te dozvoljavaju većem broju korisnika

[1] D.Ashby, C.T.Jensen. APIs for dummies: 3rd IBM Limited Edition, 2018. https://www.ibm.com/downloads/cas/GJ5QVQ7X (15.08.2020)

pristup, a ne samo onima s ovlastima za direktni pristup bazama podataka poslužitelja. Veliki broj web aplikacija i web stranica u svijetu koriste aplikacijsko programska sučelja u bar nekom kontekstu, bili korisnici toga svjesni toga ili ne.

Mogućnost ponude podataka u strojno-čitljivom formatu koji će biti dostupni preko programskog sučelja putem interneta donijela je revoluciju u svijetu softwarea i informacija. Zbog ovakve ideje danas nam je, na primjer, moguće dohvatiti vremensku prognozu klikom gumba na našem pametnom telefonu.

Proces razmjene informacija između poslužitelja i korisnika na slučaju API sučelja

Razmjenom podataka u ovom kontekstu podrazumijeva se slanje i primanje podataka od računala do računala putem interneta odnosno World Wide Weba. Ako se ovakva razmjena odnosi na API sučelja izvršava se s korisnikovim dohvaćanjem podataka koji će služiti za obradu ili implementaciju u programski projekt. Korisnik, na ovom primjeru programer, želi pristupiti API sučelju pomoću svog programskog koda i dohvatiti podatke. Cilj mu je napisati programsku skriptu te se kroz dokumentaciju informirati o načinu rada API sučelja koje želi implementirati u skriptu kako bi ju uspješno izvršio i dobio željene podatke u dogovorenom formatu za razmjenu.

Kodovi odgovora

Kada korisnik aplikacijsko-programskog sučelja pošalje zahtjev za onime što od API sučelja traži dobiva kod odgovora ili *response code*[2] od poslužitelja uz dodatne informacije za razmjenu sadržaja. Kod odgovora je statusni kod koji koriste računala pri razmjeni većine informacija na internetu, ovaj kod je, neovisno o operacijskom sustavu, pretraživaču, internet poslužitelju ili pak programskom jeziku. Kada je zatražena web stranicu putem preglednika, poslužitelj stranice vraća pretraživaču kod odgovora kojeg ne prikazuje, osim kada stranica ne funkcionira - tada je često vidljiv broj 404[3] koji predstavlja "Not found" poruku što najčešće znači da DNS[4] (Domain name system) poslužitelj koji razrješava domene u Internet Protokol (IP) adrese, nije u mogućnosti pronaći zapis s adresom ili podstranica na adresi koja je zatražena ne postoji.
Stoga kod odgovora nije isključivo korišten u API tehnologijama već općenito na većini zahtjeva na internetu danas. Ovaj kod programeru i korisniku API sučelja omogućuje uvid u stanje njegovog zahtjeva, što mu predstavlja koristan podatak jer pomoću ovog koda može znati jeli razmjena informacija uspješno provedena. U potpunosti uspješno provedena razmjena informacija poslat će kod 200 (OK) programeru, koji će pregledati dobivene podatke kako bi ustanovio postoji li greška na razini programskog dijela API sučelja u pružanju informacija za ovaj specifični zahtjev. Postoje i načini autentikacije za API sučelja, kao što postoje i za web

[2] R. Fielding, J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, 2014. https://tools.ietf.org/html/rfc7231 (17.06.2020)
[3] W3C.org. RFC 2616 Fielding chapter 10: Status code definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html (01.07.2020)
[4] J. Klensin. Role of the Domain Name System (DNS), 2003. https://tools.ietf.org/html/rfc3467 (16.07.2020)

stranice, koji daju do znanja korisniku kodom odgovora smije li pristupiti resursu. Ukoliko programer ne priloži validni autentikacijski ključ kada pristupljuje API sučelju koje zahtjeva autentikaciju često će mu poslužitelj odgovoriti kodom odgovora 403 (Forbidden) što predstavlja nedostupnost resursa radi manjka priloženih informacija o autentikaciji korisnika u zahtjevu neovisno radi li se o API sučelju. Ako API poslužitelj u svom radu dođe do greške postoji mogućnost da će kod odgovora biti 500 (Internal Server Error) ili brojevi viši od 500, a manji od 600. Također, ako programer koristi sučelje duže vrijeme, a vlasnik API sučelja odluči nadograditi sučelje i premjestiti ga moguće je da će za svoj zahtjev programer dobiti kod odgovora u rasponu od 300 do 308 koji daju do znanja programeru da je stranica premještena ili da je potrebna dodatna radnja kako bi došao do nove stranice.

Nakon što je ustanovljeno da je dobiveni odgovor poslužitelja 200 izmjena informacija može nastaviti. No uspješno poslan zahtjev za razmjenu informacija preko API sučelja još uvijek nije formiran. Potrebno je definirati i koju metodu razmjene koristiti. Iako kodovi stranica i pretraživači većinom to danas rade automatski, ako pristupamo API sučelju putem programskog pristupa potrebno je znati razlike između POST i GET metoda[5].

Metode za dohvaćanje podataka od poslužitelja

Dohvaćanje podataka putem HTTP ili HTTPS protokola zahtjeva specificiranje metode kojom ćemo dohvatiti podatke[6]. Ovu metodu određuje sam poslužitelj kao metodu na temelju koje će se razmjena podataka izvršiti, a korisnik u svom programu definira istu. Ako programer pogrešnu metodu odabere zahtjev mu neće biti validan. Kao kod kodova odgovora, u slučaju preglednika ove metode određene su automatski. U slučaju programskog pristupa nužno ih je manualno definirati.

GET metoda[7] se uvijek koristi za dohvaćanje podataka koji su često smješteni u dokumentima, kao na primjer HTML dokumenti.

[5] R. Fielding, J. Gettys,J. Mogul, T. Berners-Lee i sur. W3C.org. RFC 2616 Fielding chapter 9: Method definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html (01.07.2020)

[6] R. Fielding, J. Gettys, J.C. Mogul, L. Masinter, T. Berners-Lee i sur. Hypertext Transfer Protocol: HTTP/1.1, 1999. https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf (27.06.2020)

[7] R. Fielding, J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, 2014. https://tools.ietf.org/html/rfc7231 (17.06.2020) i W3C.org. RFC 2616 Fielding chapter 9: Method definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html (01.07.2020)

HTML[8] ili *HyperText Markup Language* je kodni jezik kojim se definira "arhitektura" web stranice te se na temelju njega dodaju aplikacije - pomoću npr. JavaScript programskog jezika, ili pak stil i estetika stranice pomoću CSS (cascading style sheets) koda.

Danas zahtjev koji koristi GET metodu može dohvaćati i razne druge datoteke osim HTML datoteka. On ne smije i ne bi trebao mijenjati sadržaj poslužiteljevog sustava ili datoteka, već poslužiti postojeće datoteke posjetiteljima na uvid i korištenje. Ovaj oblik u slučaju slanja parametara stranici kako bi se dobio ciljani sadržaj zahtjeva unos parametara kroz URL.

URL[9] (uniform resource locator) - označava lokaciju nekog web sjedišta na internetu kao što, na primjer, hiperveza označava smjer u kojem će preglednik odvesti korisnika kada klikne na nju.

Ovi URL-ovi se prosljeđuju poslužitelju kroz korisnikov zahtjev kako bi poslužitelj poslao traženi resurs. URL, kao što je već rečeno, se može i modificirati dodatnim korisničkim zahtjevima. Ovakva modifikacija izvršava se parametrima nakon domenskog imena u URL-u gdje korisnik može izmijeniti sadržaj kojeg API sučelje ili stranica vraća. Važno je pratiti pravila web stranice formiranjem ovakvog zahtjeva kako bi kod odgovora bio 200-OK, što je potvrda da je zahtjev uspješno izvršen. Parametri koje korisnik može unijeti uz GET zahtjev su ograničeni s obzirom na to da se radi o modifikaciji URL-a koji ne može biti beskonačno dug. Za razliku od GET metode, POST nema ograničenja u odnosu na parametre.

POST metoda[10] nam pomaže, na primjer, pri prijavljivanju na društvene mreže ili provođenju transakcija na internet bankarstvu. Uloga POST zahtjeva nije samo da skriva parametre od malicioznih napadača koji čitaju URL-ove kod osjetljivih informacija poput prijava već i da proslijedi poslužitelju na najefikasniji način upute kako napraviti idući korak u posluživanju web stranice.

Prednosti slanja parametara u "tijelu" (eng. *body*) poruke zahtjeva su mogućnost slanja velikog broja parametara u zahtjevu i mogućnost ispunjavanja velike količine informacija u same parametre. Korisnik modificira tijelo zahtjeva s predefiniranim imenima parametara koje ispunjava svojim parametrima kako bi dobio tražene informacije ili na primjer proveo autentikaciju i došao do željenog resursa.

[8] F. Moraes - HTML.com. HTML Cheat sheet. https://html.com/wp-content/uploads/html-cheat-sheet.pdf (01.09.2020)

[9] T.S. Smith,Uniform Resource Locators (URLs): Powerful Reference Tools for Librarians and Information Professionals, 1996. https://files.eric.ed.gov/fulltext/ED401942.pdf (27.06.2020)

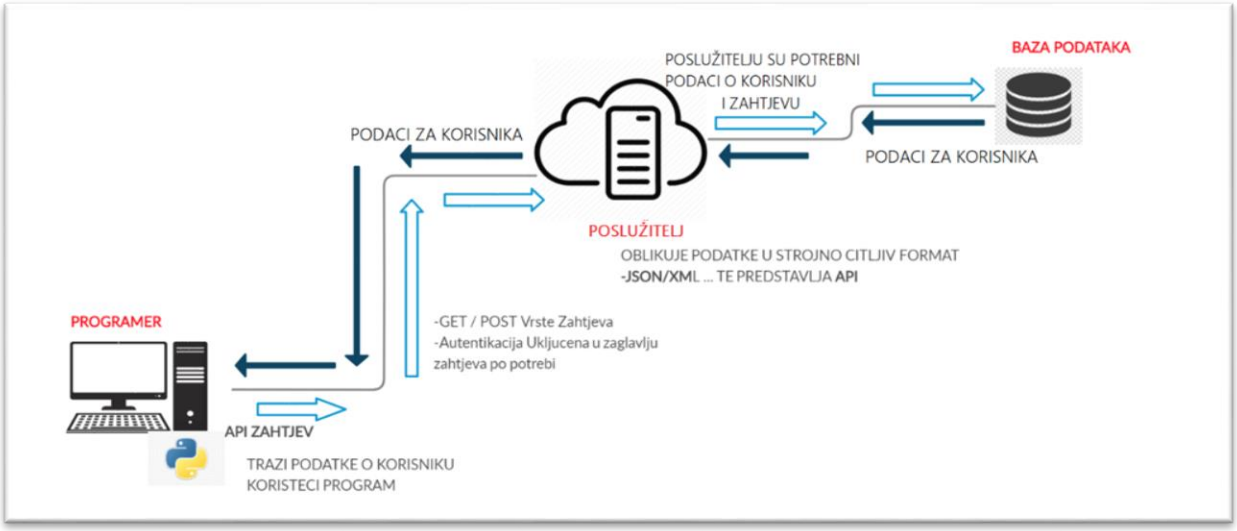
[10] Navedena dijela Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content
i W3C.org. RFC 2616 Fielding chapter 9: Method definitions

Zaključak o ovim metodama je da je POST sigurniji jer se parametri stavljaju u tijelo poruke pa time i omogućuje veću količinu podataka, dok ih GET nosi u URL-u pa je na neki način pristupačniji i omogućuje malo lakši programski pristup.

Nakon što je u potpunosti uspostavljen zahtjev poslužitelju, u ovom slučaju API sučelju, od njega je zatražen prijenos posebno formatiranih podataka putem mreže. API sučelje koristeći poslužiteljske funkcionalnosti šalje dijelove podataka korisniku čiji je zahtjev podnesen te postepeno prenosi cijelu podatkovnu datoteku i potvrđuje prijenos kodom statusa. Uzimajući u obzir da se često radi o jednostavnoj tekstualnoj datoteci, koja ne zauzima puno memorijskog prostora, ova komunikacija odvija se vrlo brzo što omogućuje programeru oslanjanje na API sučelje i kada uspostavi svoj program u produkcijsko okružje.

Od programskog sučelja do API poslužitelja i autentikacija korisnika API sučelja

Podaci za prijenos uređeni su od poslužitelja i strukturirani za razmjenu u strojno-čitljivom formatu. Programerova aplikacija preuzima ove podatke i učitava ih putem odgovarajućeg librarya/modula za potrebe programskog rada s njima odnosno prevodi ih u programske strukture podataka. Ukoliko programer ne može koristiti libraryje/module, može podatke obraditi sam koristeći sve što programski jezik nudi. Ponekad podaci koje API sučelje vraća korisniku nisu formatirani u standardiziranom formatu što znači da ih programer može direktno proslijediti za daljnju obradu ili prikaz. Bez obzira na format, programerov zahtjev poslužitelju najčešće zahtijeva i komunikaciju s bazom podataka poslužitelja, koja na upit šalje tražene informacije poslužiteljevom programu koji formatira podatke za API sučelje onako kako će se proslijediti korisniku.



Slika 1. - Komunikacijski tijek u razmjeni podataka pomoću Aplikacijsko programskih sučelja

Na slici 1. vidljivo je kako radi komunikacija od programera do API sučelja koje dohvaća i obrađuje podatke iz baze podataka u format koji programer očekuje. Primjeri formata koje API sučelja tipično vraćaju korisniku su JSON, XML ili YAML. Radi se o strojno čitljivim formatima namijenjenim za razmjenu podataka. Ovaj poslužitelj s navedenim svojstvima predstavlja API poslužitelja. Ukoliko pogledamo početak zahtjeva od programera poslužitelju možemo uočiti da programer koristi program izrađen u jeziku Python, no ovo može biti bilo koji drugi programski jezik te kroz zaglavlje zahtjeva ili modifikacijom URL-a za API govori

poslužitelju što mu je točno potrebno. Za ovu komunikaciju koriste se GET ili POST[11] metode. Odabir odgovarajuće metode važan je za razmjenu podataka putem API sučelja jer se često radi o osjetljivim informacijama koje su prenesene i dohvaćene u komunikaciji s poslužiteljem. Pogotovo ako poslužiteljski API nema šifriranu konekciju poput HTTPS protokola (HTTP - Secure), to jest ne koristi SSL/TLS (Secure socket layer/Transport Layer Security) kriptografske protokole[12].

Pri korištenju aplikacijsko-programskih sučelja ponekad je potrebno u zahtjevu priložiti i autentikacijski ključ koji je dobiven najčešće putem registriranja na web stranici koja može zahtijevati i financijsku naknadu kod plaćenih aplikacijsko-programskih sučelja. Poslužiteljski program gleda korisnikov autentikacijski ključ (*API Ključ)* i osnovne informacije poput eksterne/vanjske IP (*internet protocol)* adrese kako bi prebrojao koliko je do sada zahtjeva poslano radi ograničenja poslužiteljevog sustava, mreže ili web poslužitelja. Cilj je spriječiti slučajno ili namjerno preopterećenje poslužitelja koje uzrokuje uskraćivanje usluge drugim korisnicima. Stoga je poslužitelju razumno ograničiti pristup kako bi ostao dostupan drugim korisnicima na korištenje i omogućio konzistentan prijenos podataka bez smetnji. Naravno, uz određene nadoplate neki poslužitelji dozvoljavaju beskonačan broj zahtjeva, no i oni moraju paziti na količinu prometa koji im dolazi prema API sučelju kako ne bi došlo do uskraćivanja usluge. Neka API sučelja ne trebaju više od jednostavnog programa za posluživanje podataka jer se radi o pružanju podataka direktno iz baze podataka. Iako se ne vidi često, važno je spomenuti primjer MongoDB baze podataka koja pretraživanjem ispisuje JSON formatiran odgovor koji se može automatski proslijediti korisniku. API sučelje šalje korisniku ono što je tražio u svom zahtjevu i korisnik te podatke, uz kod odgovora i ostale potvrde na razini mreže, prihvaća kako bi provjerio radi li se o ispravnim podacima ukoliko to želi.

[11] C. Shiflett. HTTP: Developer's handbook, 2003. https://books.google.hr/books?id=oxg8_i9dVakC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false (30.08.2020)

[12] International business machines Corporation. Secure Sockets Layer/Transport Layer Security, 2015. https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_73/rzain/rzainpdf.pdf (13.08.2020)

Autentikacija korisnika API sučelja

API sučelja ponekad zahtjevaju plaćanje kako bi korisnik pristupio određenom resursu, ili bio u mogućnosti poslati veliki broj zahtjeva sučelju. Besplatni i plaćeni API oba mogu zahtjevati neku vrstu autentikacije kako bi korisnik dobio onoliko usluge koliko je i predviđeno. Najčešća vrsta autentikacije kod API sučelja je API ključ[13].

Ukoliko programer treba pristupiti API sučelju koje kao autentikaciju traži API ključ, zatražit će ga najčešće od web stranice gdje će se registrirati i ako je potrebno platiti uslugu da dobije ključ. Programer će ključ zatim koristiti u svakom od svojih zahtjeva. API ključ najčešće izgleda kao skup nasumično odabranih brojeva i slova, iako to nije. API ključ prati pravila o univerzalnom i unikatnom identifikatoru IETF (*internet engineering task force*) organizacije[14].

Primjer API ključa : ***zadKLNx.DzvOvjQH923TumGL2UrpjpQh91tumGLQsUbf67vs9***

Također, ukoliko programer podijeli svoj API ključ drugom korisniku, tada drugi korisnik pristupa preko njegovog API ključa i računa što znači da se programeru broji svaki poslan zahtjev kao da ga je on sam poslao. Uzimajući u obzir da API ključevi ne sadrže nikakve informacije o vlasniku, izvrsni su i za dijeljenje s drugim programerima u sklopu projekta koji koristi API sučelje bez dodatnih vrsta autentikacije, što je često slučaj.

Uz API ključ postoje i drugi načini autentikacije kao ona putem samog HTTP zahtjeva[15] gdje se u parametrima zaglavlja pruža informacija o autentikaciji. Radi niske sigurnosti ovog načina autentikacije kreirani su i kompleksniji pristupu poput OAuth autorizacije ili na primjer OpenID Connect autentikacije.

OAuth[16] ili *Open Authorization* omogućuje korisniku pristup njegovom računu na web sjedištu te resursima koje sjedište nudi korisniku. Jedna od prednosti OAuth autorizacije je činjenica da korisnik ne mora unositi svoje korisničko ime ili e-mail i lozinku nego često samo u zahtjevu priložiti OAuth ključ. Primjer OAuth autorizacije bila bi stranica koja nudi zapisivanje tekstualnih zapisa i spremanje na računu korisnika. Ovim datotekama se može s lakoćom pristupiti koristeći OAuth autorizaciju. Za razliku od API ključa OAuth se koncentrira na

[13] RapidApi.com. API glossary: API Key – What is an API Key. https://rapidapi.com/blog/api-glossary/api-key/ (04.07.2020)
[14] Leach, P.; Mealling, M.; Salz, R. A Universally Unique IDentifier (UUID) URN Namespace, 2005. https://tools.ietf.org/html/rfc4122
[15] CISCO.COM. REST API Chapter 3: Client authentication. https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/restapi/restapi/RESTAPIclient.pdf (31.08.2020)
[16] H. Tschofenig, B. Leiba, B. Cook, R. van Eijk. BROWSER SUPPORT FOR THE OPEN AUTHORIZATION (OAUTH) PROTOCOL, 2011. https://www.w3.org/2011/identity-ws/papers/idbrowser2011_submission_32.pdf (02.07.2020)

dijeljenje pristupa korisničkog računa drugim osobama, gdje će te osobe često imati ograničen pristup.

Važno je napomenuti i kompleksnije strukture s ciljem održavanja sigurnosti u razmjeni podataka putem API sučelja, što je na primjer OpenID Connect. OpenID Connect[17] je identitetski sloj na OAuth 2.0 protokolu koji dozvoljava klijentu da potvrdi korisnika autentikacijskog ključa na temelju zapisa poslužitelja o autentikaciji i autorizaciji. OpenID uz OAuth autorizaciju nudi i autentikaciju korisnika što igra važnu ulogu u projektima koji koriste API sučelja kako bi se pratili svi zahtjevi.

[17] Openid.net. Welcome to OpenID Connect. https://openid.net/connect/ (05.07.2020)

Formati za razmjenu informacija putem API sučelja

Iako postoji veliki broj formata za razmjenu podataka putem aplikacijsko-programskih sučelja, a i šire od njih, podjela na najčešće korištene formate poslužit će kao vodič. Najčešće korišteni formata za razmjenu podataka putem API sučelja svodi se na XML i JSON strojno čitljive formate, uz YAML kao čitljiviji konkurent JSON-u,te CSV kao format koji se primarno koristi za razmjenu strogo tabličnih podataka.

JSON

JSON[18] (*JavaScript Object Notation*) – je strojno čitljiv hijerarhijski strukturiran[19] format lak za čitanje i pisanje čovjeku, a i računalima. Ovaj format dolazi u tekstualnom obliku čiji su podaci formatirani u obliku ključa i vrijednosti. Ključ i vrijednost u JSON podacima se često rasporedom vitičastih zagrada i ponekad indentacijom (TAB-ovima) smjesti unutar hijerarhije. Iako indentacija nije apsolutno potrebna olakšava čitljivost ljudskom korisniku, a računalu ne predstavlja grešku. JSON-ova jednostavnost i čitljivost, prilagođenost programskim strukturama podataka (npr. svaki JSON zapis je validan, između ostalog, JavaScript i Python kod) te brzina ručnog pisanja JSON zapisa daje mu prednost nad drugim formatima za razmjenu podataka.

JSON Standard:

JSON je standardiziran ECMA-404[20] standardom te potiče iz ECMA-262[21] standarda objavljenog 1999. godine. Ne ovisi o niti jednom programskom jeziku iako je izgrađen na temelju ECMAScript[22] programskog jezika. ECMA-404 definira sintaksu JSON-a, govori o strukturi i korištenju formata, njegovoj namjeni te prikazuje oblik JSON objekta i vrijednosti koje prima.

---

[18] https://www.json.org/json-en.html (21.08.2020)
[19] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON), 2006. https://tools.ietf.org/html/rfc4627 (16.07.2020)
[20]Ecma-international.com. Standard ECMA-404: The JSON Data Interchange Syntax (Drugo izdanje), 2017. https://www.ecma-international.org/publications/standards/Ecma-404.htm (05.07.2020)
[21] Ecma-international.com. ECMAScript Language Specification (treće izdanje), 1999. http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262, 3rd edition, December 1999.pdf (05.07.2020)
[22] Ecma-international.com. Standard ECMA-262: ECMAScript Language Specification (izdanje 5.1), 2011. https://www.ecma-international.org/ecma-262/5.1/#sec-15.12 (07.07.2020)

XML

XML[23] (*Extensible Markup Language*) je, poput JSON-a, strojno čitljiva vrsta dokumenta. Izgled XML-a sliči HTML kodu (na temelju kojeg je i nastao), a poput JSON formata koristi hijerarhijski strukturirane podatke, a definira ih između dva imena ili naslova vrijednosti (slično JSON ključevima). Ova imena ili naslovi vrijednosti koriste znak manje za oznaku početka imena vrijednosti, te znak više za kraj. Razlikuju se samo po kosoj crti prije drugog imena vrijednosti čije cijelo ime vrijednosti uz ovu crtu predstavlja oznaku kraja definiranja vrijednosti. Početak XML-a objavljen je uz SGML (*Standard Generalized Markup Language*) jezik te standardiziran kroz ISO 8879[24]. *World Wide Web* konzorcij je objavio prvu verziju specifikacije XML-a[25] u 1998. godini što ga čini godinu starijim od JSON formata. XML je namijenjen za spremanje i razmjenu podataka te svojom poznatom sintaksom osvaja svijet razmjene informacija. Što se razmjene informacija putem aplikacijsko-programskih sučelja tiče i danas drži status jednog od najkorištenijih formata.

XML standard:

Uz činjenicu da je format za razmjenu podataka, XML je prije svega *MarkUp* jezik. Jedan od ciljeva XML-a bio je procesiranje SGML jezika na webu. Dizajniran je da bude interoperabilan sa SGML-om i HTML-om[26]. XML je "Aplikacijski profil"[27] SGML-a te je njegovo korištenje omogućeno WebSGML adaptacijom[28] u ISO 8879 standardu. Danas koristi za što je originalno namijenjen, razmjenu podataka.

[23] XML.COM. https://www.xml.com/pub/a/98/10/guide0.html (21.08.2020)
[24] Internacionalna organizacija za standardizaciju. ISO 8879:1986 Information processing : Text and office systems — Standard Generalized Markup Language (SGML), 1986. https://www.iso.org/standard/16387.html (14.07.2020)
[25] xml.silmaril.ie. The XML FAQ (Frequently Asked Questions) http://xml.silmaril.ie/responsible.html (14.07.2020)
[26] W3C.org. Extensible Markup Language (XML) 1.0 (Peto izdanje), 2008. https://www.w3.org/TR/REC-xml/ (02.07.2020)
[27] xml.silmaril.ie. The XML FAQ (Frequently Asked Questions) http://xml.silmaril.ie/standards.html (14.07.2020)
[28] C. F. Goldfarb. Document Processing and Relating Communication: Technical Corrigendum for WebSGML Adaptations, 1997. http://xml.coverpages.org/wg8-n1929-g.html (30.07.2020)

Usporedba JSON i XML formata na primjeru uporabe i svojstava

Prikaz JSON i XML formata vizualno je sličan po hijerarhiji ukoliko uzmemo indentaciju u obzir, a ako se koncentriramo na samu sintaksu vrlo su različiti. Oba dozvoljavaju prijenos podataka i računalno su čitljivi te najbolje funkcioniraju koristeći odgovarajuću kodnu stranicu za tekst kako bi bili čitljivi cijelom svijetu. Tako se ove dvije vrste formata za razmjenu najčešće poslužuju u UTF-8 kodnoj stranici kako bi se omogućila čitljivost rezultata internacionalne razine.
UTF-8[29] (8-bit Uniform Translation format) - Definiran po UNICODE standardu za računalni prikaz teksta, opisan u ANNEX D - ISO/IEC 10646-1, UTF-8 je način zapisa kodnih točaka. UTF-8 koristi bajtove od 8 bita uz UNICODE standard kako bi korisnik mogao vidjeti, na primjer, slovo "Č" na svom ekranu. Ova slova šifrirana su u sljedovima od 1 do 4 bajta, što je zapravo oznaka 8 mjesta za bitove (jedinica ili nula) što čini jedan byte (bajt).

Sintaksa samih zapisa drukčija je u pogledu i činjenici da se JSON bazira na odnosu ključa i vrijednosti, dok XML prati klasični HTML način zapisa informacija u kojem otvara i zatvara svaki unos.

```
<?xml version="1.0" encoding="utf-8">
<country name="Croatia">
      <localname> Hrvatska </localname>
      <tag> HR </ tag>
      <continent> Europe </ continent>
</country>
```

Ispis 1 - Pojašnjenje uporabe XML formata na primjeru

S obzirom na to da je zahtjev na zapisima XML-a definirati verziju i shemu kodiranja radi strojne čitljivosti s raznim sustavima prateći sintaksu potrebno je zapisati ovu liniju kao prvu. Na ovom kodu lako je primijetiti razlike u hijerarhiji između početka retka unosa <*country*> te na primjer unosa <*localname*>. Indentacija poput ove u XML kodu predstavlja da u oznaci *country* pripadaju oznake *localname, tag* te *continent* kao podaci hijerarhijski niže razine. Kao i kod JSON-a, uvlačenje nije zahtjev, ali poboljšava čitljivost koda. *Localname* redak imenuje lokalno ime države u unosu *country* iznad. U retku *country* definirana je i država putem atributa

[29] F. Yergeau. UTF-8, a transformation format of ISO 10646, 2003. https://tools.ietf.org/html/rfc3629 (01.07.2020)

name. Nakon *localname* linije definiran je i *tag* unos čija vrijednost predstavlja internacionalnu oznaku države u dva slova prateći odredbe ISO 3166-1 alpha-2 standarda[30]. Na kraju je definiran i kontinent, u ovom slučaju Europa te je svaki unos, poput HTML koda i njegovih unosa zatvoren prateći pravila XML sintakse.

```
{
	"naslov" : "Hamlet",
	"autor" : "Shakespeare",
	"podaci" : {
		"mjesto" : "Danska",
		"stoljeće": 15,
		"zanr" : "Tragedija"
		}
}
```

Ispis 2 - Pojašnjenje uporabe JSON formata na primjeru

Ovaj zapis prikazuje JSON format gdje se znakom „:“ definira odnos ključa i vrijednosti. Ključevi u JSON-u moraju biti tekst pa su stoga svi pod navodnicima. Primjer ključa i vrijednosti u ovom slučaju je "*naslov"* kao ključ te *"Hamlet"* kao vrijednost. Također, u ključu „podaci“ spadaju „mjesto“, „stoljeće“ te „zanr“ kao niža hijerarhijska razina. Vitičaste zagrade u JSON formatu definiraju početak i kraj objekta. Kao što je vidljivo na primjeru u glavnom zapisu imamo i zapis "*Podaci*" na razini ispod što je još jedan objekt.

Iako su vrlo različiti po sintaksi, slični su po svom svojstvu strojne čitljivosti i izgledu hijerarhije oslonjenom na indentacijama. No postoji veliki broj svojstava koji ulaze u razlikovanje ovih formata

JSON	**XML**
Podupire tekst (tj. *string* vrstu podataka), brojeve (cijele i decimalne), boolove vrijednosti (True i False), liste te *null* vrijednosti (Nema vrijednosti), vrijeme i datum	Podupire *string* tekst, brojeve, bool vrijednosti, vrijeme i datum

[30] Internacionalna organizacija za standardizaciju. ISO 3166: COUNTRY CODES https://www.iso.org/iso-3166-country-codes.html (30.07.2020)

Fokusiran strogo na podatke	Može ugrađivati strukturirane podatke i u prirodan tekst
Sintaktički „kraći“	Sintaktički „duži“ – većinom radi tag-ova za zatvaranje
Koristi sintaksu vitičastih zagrada i odnos ključa i vrijednosti, odnosno sintaksu kojom se u JavaScriptu i drugim jezicima zapisuje struktura podataka koja se često naziva *dictionary* odnosno rječnik	Koristi sintaksu sličnu HTML kodu, otvara i zatvara svaku ime svake vrijednosti te pri kreiranju imena vrijednosti dozvoljava unos dodatnih atributa odvojenih razmakom

Tablica 1 - Usporedba XMLa i JSONa

YAML[31] (*YAML Ain't Markup Language*) je standard za serijalizaciju podataka poznat po čitljivosti za ljude uz izvrsnu strojnu čitljivost. Iako po izgledu koristi indentaciju putem TAB tipke u sintaksi ona nije validna za indentaciju, indentacija mora biti kreirana SPACE tipkom. YAML koristi vrlo čitljivu sintaksu i odnose slične odnosima ključa i vrijednosti kod JSON datoteke, no YAML ne zahtjeva obvezno korištenje zagrada i navodnika te omogućuje definiranje korelacije ključa i vrijednosti dvotočkom.

CSV[32] (*Comma separated values)* ili tzv. *tab delimited text* je format gdje su vrijednosti razdvojene zarezom ili nekim drugim dogovorenim graničnim znakom poput tabulatora. CSV nije niti sličan XML-u, JSON-u ili YAML-u jer po sintaksi izgleda u potpunosti drukčije te jer se ne radi o hijerarhijski formiranim podacima već tablično. CSV definira "naslovne" podatke u prvom redu te mnoštvo unosa koji su "vrijednosti" podijeljeni zarezom. Ako su u prvom redu naslovi "*korisnik*" te "*brojtelefona"* svaki idući red bit će podaci o raznim korisnicima i njihovim telefonskim brojevima. S obzirom na djelomično zastarjelu strukturu i mogućnosti CSV se sve manje koristi u današnjim aplikacijsko-programskim sučeljima, ali je još uvijek čest prilikom razmjene statističkih podataka ili izvoza iz relacijskih baza podataka.

[31] The Official YAML Website. https://yaml.org (22.08.2020)
[32] Y. Shafranovich. Common Format and MIME Type for Comma-Separated Values (CSV) Files, 2005. https://www.ietf.org/rfc/rfc4180.txt (08.09.2020)

Podjele aplikacijsko-programskih sučelja

API sučelja se mogu podijeliti na više načina, no najčešće se API sučelja dijele klasifikacijama poput onih po pristupačnosti, arhitekturi i slično. Poznavati podjele API sučelja kako bi se odabrao odgovarajući API programeru je vrlo važno kako bi ostvario svoje ideje i na najefikasniji način preuzimao potrebne podatke.

Podjela API sučelja prema pristupačnosti sučelju

U podjeli po pristupačnosti API sučelju možemo razvrstati[33]:

- **Besplatni i otvoreni API** – javno dostupni API za kojeg ne postoji restrikcija za pristup podacima osim one za broj zahtjeva radi održavanja stabilnosti poslužitelja API sučelja. Otvoreni API svojom pristupačnošću i činjenicom da je besplatan za korištenje potiče dijeljenje podataka i znanja te otvara mogućnosti organizacijama ili fizičkim osobama da pokrenu svoje projekte na kojima će zaraditi, podijeliti znanje, ili si olakšati život na neki način i naučiti nešto novo koristeći ovaj besplatni resurs. Otvoreni API najčešće podrazumijeva praćenje pravila OAI konzorcija (Open API initiative)[34] koji se kroz Open API specification zalaže za slobodu dijeljenja podataka[35] putem API sučelja.
- **Partnerski API** – Zahtjeva posebne ovlasti kao restrikciju za korištenje. Najčešće podrazumijeva registraciju na web sjedištu te ponekad plaćanje za pristup ili povećanje mogućnosti.e
- **Internalni ili privatni API** – namijenjen za korištenje unutar neke organizacije s ciljem bržeg i efikasnijeg poslovanja, unaprjeđenja proizvoda ili usluge.
- **Miješani ili *Mixed/Composite* API** – ovaj tip kombinira različite vrste pristupačnosti API sučelju. Glavni ciljevi ovakvih API-ja su poboljšanje performansi te automatizacija ili olakšanje nekog procesa rada s podacima.

Podjela API sučelja prema arhitekturi i protokolima

SOAP (Simple Object Access Protocol)[36] je protokol koji koristi XML kao format za prijenos i dohvaćanje podataka. Njegova glavna funkcija je definirati strukturu poruka i način komunikacije. SOAP također koristi WSDL ili Jezik definicije web usluga kako bi definirao svoju uslugu, metode i način korištenja. SOAP zahtjeva pisanje XML zapisa za dohvaćanje informacija pa zahtjeva više vremena nego ostali protokoli i arhitekture, ali pruža mogućnosti ekspanzije poput *Web Service Security* ekspanzije koja, iako nije vrlo potrebna u slučaju izrade

[33] RapidApi.com. Types of APIs, 2020.
https://rapidapi.com/blog/types-of-apis/ (08.09.2020)
[34] OpenAPI initiative. https://www.openapis.org/about (30.08.2020)
[35] Swagger: API Documentation and Design Tools for Teams. https://swagger.io/specification/ (11.08.2020)
[36] W3Schools.com. XML SOAP. https://www.w3schools.com/xml/xml_soap.asp (04.07.2020)

otvorenog API sučelja, kod API sučelja zatvorenog ili ograničenog pristupa može igrati značajnu ulogu u sigurnosti[37].

RPC[38] ili *Remote procedure call* je protokol koji dozvoljava računalnom programu da izvrši neku proceduru na drugom računalu u mreži na temelju klijent-poslužitelj interakcije bez da programer mora posebno definirati i programirati ovakvu konekciju. Osim namjene za programska rješenja, postoje i rješenja gdje RPC koristi posebno formatirane pozive na funkcije ili procedure.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>
</soap:Envelope>
```

Ispis 3 - Primjer SOAP odgovora (Cijena jabuka u trgovini)

XML-RPC (Extensible Markup Language - Remote procedure call)[39]: je niz implementacija koje omogućuju programski pristup putem RPC protokola. XML RPC koristi poseban XML format za prijenos podataka u odnosu na SOAP koji koristi obični. Pošto koristi HTTP za razmjenu podataka i zadavanje zadataka poslužitelju, a XML za format, omogućuje i jednostavno korištenje čak i u slučaju kompleksnih struktura podataka. XML-RPC koristi minimalnu propusnost i puno je jednostavniji od SOAP-a. XML-RPC ubrzo postaje temelj za SOAP protokol.

```
<?xml version="1.0"?>
<methodCall>
  <methodName>studenti.dohvatiImeStudenta</methodName>
  <params>
    <param>
```

[37] P.K. Potti. On the Design of Web Services: SOAP vs. REST, 2011. https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1139&context=etd (08.09.2020)
[38] A.D.Birrell, B.J.Nelson. Implementing Remote Procedure Calls // ACM Transactions on Computer Systems, Vol. 2, No. 1, 1984. http://web.eecs.umich.edu/~mosharaf/Readings/RPC.pdf (31.08.2020)
[39] XMLRPC.COM. http://xmlrpc.com (22.08.2020)

```
        <value><id>15</id></value>
    </param>
  </params>
</methodCall>
```

Ispis 4 - Primjer XML zahtjeva u razmjeni putem RPC protokola

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>Rebeka Horvat</string></value>
    </param>
  </params>
</methodResponse>
```

Ispis 4.2 - Primjer XML odgovora u razmjeni putem RPC protokola

Na ovom pojednostavljenom primjeru zahtjeva i odgovora uočljiv je poziv funkcije *dohvatiImeStudenta* s identifikacijskim brojem 15 koji vraća korisniku vrijednost *Rebeka Horvat*.

JSON-RPC:[40] koristi protokol poput XML-RPC implementacije, ali umjesto korištenja XML formata za prijenos podataka, koristi JSON. Jednostavan je za korištenje te iako se u komunikaciji između poslužitelja i klijenta prenosi samo jedan JSON set podataka dozvoljen je unos više parametara. Moguće je i vraćanje više vrijednosti od poslužitelja klijentu radi efikasnosti strukture JSON formata i lakoće prijenosa podataka putem RPC protokola.

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": {"firstnumber": 7, "secondnumber": 5}, "id": 1}
<-- {"jsonrpc": "2.0", "result": 2, "id": 1}
```

Ispis 5 - Primjer JSON zahtjeva i odgovora za razmjenu putem RPC protokola

Strjelicom prema desno prikazan je zahtjev poslužitelju s parametrima *firstnumber* te *secondnumber*. Ovi parametri će se metodom *substract* odnosno oduzmi u prolazu kroz sučelje oduzeti i vratiti korisniku *result* ili rezultat "2".

JSON-RPC i XML-RPC oba rade tako da programer formira sintaksno validni upit kojeg šalje poslužitelju da obavi neku radnju ili izvrši proceduru, koji kroz sintaksu istog formata vraća odgovor.

[40] R. Koebler. RPC: JSON-RPC, 2008. http://www.simple-is-better.org/rpc/ (08.09.2020)

REST (Representational state transfer)[41]*:* REST nije protokol poput ostalih, već je skup arhitektonskih načela. Važno je REST svrstati među protokole iako nije jedan od njih radi svoje sličnosti protokolima (na primjer sličnost sa SOAP-om što se tiče HTTP prijenosa podataka) te radi činjenice da se vrlo često u projektima bira između SOAP ili REST opcije pri odabiru strukture API sučelja sa strane poslužitelja pri kreaciji. Svoju razmjenu podataka vrši putem HTTP protokola te mu je prednost što istovremeno definira koliko točno poslužitelj može prihvatiti zahtjeva i poslati odgovora. Realizacija ove arhitekture može prenositi podatke bez velikog napora za poslužitelja i pružiti jednostavnost inicijalizacije razmjene podataka radi korištenja HTTP protokola. On je *lightweight* odnosno lakša, a istovremeno i brža alternativa SOAP protokolu. REST se, za razliku od SOAP-a , ne oslanja nužno na formiranje kompleksnog zahtjeva i teksta zahtjeva za razmjenu podataka već to može obaviti na primjer jednostavnim GET ili POST zahtjevom od korisnika, a podupire i druge HTTP metode.

Najpoznatija podjela danas, što se tiče odabira vrste API-ja po obliku komunikacije i arhitekture su SOAP i REST[42]. Iako smo ih već pojasnili, s obzirom na njihovu važnost potrebno je zakoračiti dublje u razlike.

SOAP	**REST**
Protokol razmjene poruka baziran na XML-u	Arhitekturalni stil
Koristi WSDL kao jezik za definiciju web usluga i XML kao format	Najčešće koristi POST i GET metode za definiciju web usluga, a JSON i XML kao format
Ima stroga pravila i naprednu sigurnost koju treba slijediti.	Postoje labave smjernice koje omogućuju programerima laku prilagodbu
Pokreće ga funkcija , Potreban je veći *bandwith (propusnost)*	Pokreću ga Podaci, Potreban je minimalni *bandwith*

Tablica 2 - Razlike između SOAP protokola i korištenja REST arhitekture

[41] S. Kuserbajn. REST Arhitektura, 2016. https://zir.nsk.hr/islandora/object/vus:377/preview (01.09.2020)
[42] V. Kumari. Web Services Protocol: SOAP vs REST // International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. 4, No. 5, 2015. http://ijarcet.org/wp-content/uploads/IJARCET-VOL-4-ISSUE-5-2467-2469.pdf (17.07.2020)

Na prikazanoj tablici uočljivo je da SOAP nije bolji od REST-a, niti obrnuto, već su svaki od njih namijenjeni za individualne slučajeve korištenja.

Postoje i ostale, manje spomenute vrste API sučelja baziranih na raznim tehnologijama poput ASP.NET[43] služi kreiranju web aplikacija i sličnih servisa. Koristeći ASP.NET moguće je izraditi API sučelje koje koristi REST arhitekturu, dizajnirano oko .NET tehnologije i C# programskog jezika.

[43] Microsoft.com. ASP.NET Web APIs. https://dotnet.microsoft.com/apps/aspnet/apis (09.08.2020)

Vrste API sučelja po kategoriji podataka koje pružaju

Prvo pitanje koje se postavlja pri odabiru API sučelja je sama tema podataka koje sučelje nudi. Prema informacijama koje poslužuje i temama podjela API sučelja nudi širok spektar kategorija.

API Kategorije (Tematika)					
Glazba i video	Vrijeme u svijetu te datumi, UNIX Timestamps...	Kuponi	Wiki - informacije o pojmovima	razmjena ili izmjena podataka (npr. URL Shortening)	Komunikacije sa serverskim rješenjima
Analitike	Statusi i informacije	Rječnici	Mail	Rezervacije	Plaćanja i financije
Cloud	Novosti/Vijesti	Vrijeme i meteorološka prognoza	Strojno učenje	Geolokacija i informacije IP adresa	Događaji (Eventi)
Pretraživanja	Kriptovalute	Podaci o knjigama	"Datasets" - setovi podataka	Blogovi i forumi	Anti-Malware

Tablica 3 - Primjena API-ja

No ovo nisu sve kategorije API sučelja po njihovoj temi, već samo primjeri. S obzirom na to da ne postoji formalna klasifikacija te sve teme obuhvaćaju sve teme svijeta pa točna podjela nije moguća. Stoga su postavljeni razni primjeri s ciljem približavanja koncepta podijele korisniku.

Zaključak podjela i dodatni pristupi API sučeljima

Što se podjela tiče vidljivo je da se radi o širokom spektru. Ovo omogućava korisnicima da pronađu točno ono što žele u formatu u kojem to žele.

Važno je spomenuti iz perspektive *hardware*a poslužitelja da upravljanje API sučeljima sa strane pružatelja usluge može biti i koristeći Cloud infrastrukturu i Cloud *hardware*. Ovo se naziva Cloud based API.

Cloud based API[44]*:* vrsta aplikacijskog programskog sučelja koje olakšava unaprjeđenje neke usluge ili aplikacije za pružanje Cloud platformi, hardvera i softvera te dozvoljava naprednu razmjenu podataka.

Još jedna važna opcija za spomenuti je GraphQL kao bolji način za slanje zahtjeva API sučeljima.

U razgovoru o formiranju parametara zahtjeva korištenje jezika poput GraphQL[45] i njegovih metoda dohvaćanja podataka API sučelja čine proces razmjene podataka preciznijim i bržim. GraphQL je, poput SQL-a, "query" jezik[46], što znači da mu je glavna namjena dohvaćanje strukturiranih i korisnički odabranih podataka bilo direktno iz baza podataka, ili u ovom slučaju s namjenom za razmjenu putem API sučelja s odabranim dijelovima povratnih informacija. Namjena GraphQL-a je strukturiranje parametara zahtjeva, prateći korisnički unos, koji će biti poslan poslužitelju. Ovakvo formiranje zahtjeva omogućuje sužavanje samog zahtjeva i samim time optimizaciju s obje strane. GraphQL dozvoljava specificiranje točno onoga što klijent želi od poslužitelja, ukoliko mu API sučelje već to ne može pružiti, bez podataka koji predstavljaju višak.

Pošto su definirane podjele API sučelja, kako bi u potpunosti bilo jasno koje API sučelje odabrati, potrebno je odrediti u kojem formatu će podaci biti dobavljeni od poslužitelja.

[44] P. Mell, T. Grance. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology, 2011. http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf (09.08.2020)
[45] GraphQL: A query language for your API. https://graphql.org (11.08.2020)
[46] N. Schweikardt, T. Schwentick, L. Segoufin. Database theory: Query languages, 2011. http://www.lsv.fr/~segoufin/Papers/Mypapers/DB-chapter.pdf

Praktični primjeri korištenja API sučelja

Velik broj ljudi pogleda svakodnevno vremensku prognozu na svom pametnom telefonu ne uzimajući u obzir kako se ova razmjena informacija događa. Mnoštvo aplikacija na pametnim telefonima koristi aplikacijsko-programska sučelja za dohvaćanje informacija i jedna od njih je, naravno, aplikacija za vremensku prognozu. Svjetski poznate društvene mreže, stranice koje pružaju vijesti, kompleksni web shopovi i slične web stranice najčešće se oslanjaju na bar jedno API sučelje. Razlog ovom je činjenica da je efikasno i lako koristiti API sučelja, a stranicama su potrebne razne nadogradnje i funkcionalnosti. API sučelja su svojom elegantnošću pozitivno iznenadila svijet razmjene podataka na internetu. Veliki projekti oslonjeni na internetske razmjene informacija i programski pristup uz API sučelja lakše postižu svoj cilj. U poslovnom svijetu gdje god se radi s velikom količinom podataka koji se obrađuju ili nadgledaju najvjerojatnije postoji i API sučelje. S obzirom na rasprostranjenost API sučelja i popularnost nužno je programerima, a i informacijskim stručnjacima znati ih koristiti.

Jednostavan primjer dohvaćanja podataka putem API sučelja koristeći Python programski jezik

Opće je poznato u okružju programera i programerskom svijetu da Python programski jezik ima vrlo jednostavnu sintaksu, a dovoljno je spretan i moćan da ostvari većinu programskih projekta s lakoćom. Stoga je Python idealan programski jezik za namjene ovog rada i često se koristi upravo za ovakav rad s podacima.

```
import urllib.request

import json

link                                                                                    =
"http://openlibrary.org/api/books?bibkeys=ISBN:0451526538,ISBN:0385472579&format=js
on"

zahtjev = urllib.request.urlopen(link)

zahtjev = zahtjev.read()

zahtjev = zahtjev.decode("utf-8")

obradeni_podaci = json.loads(podaci)

for isbn in obradeni_podaci:

       print("[+]",isbn)

       print("[+]",obradeni_podaci[isbn]["preview_url"])
```

Ispis 6 - jednostavan primjer dohvaćanja podataka putem API sučelja OpenLibrary

Kao primjer korištenja API sučelja korišten je besplatni API "*OpenLibrary*" koji besplatno nudi podatke o knjigama. U Python programski kod potrebno je priložiti i dva vanjska modula (koja dolaze sa standardnom instalacijom Pythona) "*urllib*"[47] koji pomaže pri slanju i formiranju zahtjeva prema poslužiteljima koji dijele podatke putem HTTP i HTTPS protokola te "*json*"[48] modul koji pomaže pri učitavanju (dekodiranju i kodiranju) JSON datoteka te daje programski pristup podacima omogućen od svojstva strojne čitljivosti JSON datoteke. Nakon što su moduli uneseni u kod potrebno je definirati URL s kojeg će informacije biti dohvaćene. S obzirom na činjenicu da se radi o GET metodi zahtjeva, parametri o onome što će API sučelje vratiti u odgovoru na zahtjev ovise o unosu parametara koji su uneseni u URL. Na ovom primjeru unesena su dva ISBN koda (dvije različite jedinice građe) te je odabran format JSON kroz parametar kako bi se podaci dohvatili u JSON formatu (*&format=json*).
Kako bi zahtjev bio poslan koristeći *Urllib* modul potrebno je pozvati funkciju *urlopen()* te istoj dati argument URL-a, koji je u ovom slučaju varijabla "*link*". Nakon što je zahtjev definiran potrebno je specificirati koji dio odgovora poslužitelja želimo pročitati kako bi dobili JSON sadržaj iz odgovora što nam omogućuje funkcija *.read()*. Redefiniranjem varijable *zahtjev* u nju je smješten tekst koji je odgovor API sučelja, u JSON formatu.
Pošto su dohvaćeni podaci kodirani internacionalnim standardima, potrebno je koristiti Pythonovu funkciju *.decode("utf-8")* pomoću koje je dekodiran dohvaćeni sadržaj u "UTF-8" kodni format, što čini dohvaćene podatke lako čitljivima i bez grešaka kodnih formata te će sva slova biti prikazana točno kako je namijenjeno.

Kako bi Python shvatio JSON datoteku kao JSON format potrebno je učitati varijablu s JSON podacima u njima odgovarajući python modul *JSON*. Podaci dohvaćeni u tekstualnom obliku s API poslužitelja, formatirani kao JSON bit će shvaćeni kako je namijenjeno pomoću funkcije *JSON* modula *.loads()* koja će prihvatiti kao argument varijablu *podaci* te će ova konverzija biti smještena u varijablu *obradeni_podaci*. Razvrstavanje podataka iz varijable *obradeni_podaci* olakšat će *for petlja* kroz koju se izdvaja svaku ISBN oznaka te URL za posjećivanje stranice iste knjige na *OpenLibrary* web stranici. Ovaj podatak o URL-u knjige pružio je OpenLibrary API u odgovoru na naš zahtjev te je pristupljen u kodu referiranjem na varijablu i njenom unosu više hijerarhijske razine koja predstavlja ISBN broj te niže "*preview_url*" koja predstavlja

[47] Python.org. Urllib.request (Documentation): extensible library for opening URLs. https://docs.python.org/3.0/library/urllib.request.html (09.09.2020)
[48] Python.org. Json (Documentation): JSON encoder and decoder. https://docs.python.org/3.0/library/json.html (01.09.2020)

traženi URL. Ovo omogućuje ispis vrijednosti "*preview_url*" bez obzira na ugniježđenost podataka jer je korišten *Pythonov* način navigacije kroz strukturirane podatke koristeći uglaste zagrade.

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\filip\Desktop\GetOpenLibrary.py ==========
[+] ISBN:0451526538
[+]URL: http://openlibrary.org/books/OL1017798M/The_adventures_of_Tom_Sawyer
[+] ISBN:0385472579
[+]URL: http://openlibrary.org/books/OL1397864M/Zen_speaks
>>>
```

Slika 2 - ispis programa, primjer dohvaćanja informacija putem API sučelja


```
{"ISBN:0451526538": {"info_url": "http://openlibrary.org/books/OL1017798M/The_adventures_of_Tom_Sawyer", "bib_key": "ISBN:0451526538", "preview_url": "http://openlibrary.org/books/OL1017798M
/The_adventures_of_Tom_Sawyer", "thumbnail_url": "https://covers.openlibrary.org/b/id/295577-S.jpg", "details": {"identifiers": {"librarything": ["2236"], "project_gutenberg": ["74"], "goodreads":
["1929684"]}, "subject_place": ["Mississippi River Valley", "Missouri"], "covers": [295577], "lc_classifications": ["PS1306 .A1 1997"], "latest_revision": 9, "genres": ["Fiction."],
"source_records": ["marc:marc_records_scriblio_net/part25.dat:213801824:997", "marc:marc_loc_updates/v40.i23.records.utf8:2463307:1113", "amazon:0451526538"], "title": "The adventures of Tom
Sawyer", "languages": [{"key": "/languages/eng"}], "subjects": ["Sawyer, Tom (Fictitious character) -- Fiction", "Runaway children -- Fiction", "Child witnesses -- Fiction", "Boys -- Fiction",
"Mississippi River Valley -- Fiction", "Missouri -- Fiction"], "publish_country": "nyu", "by_statement": "Mark Twain ; with an introduction by Robert S. Tilton.", "oclc_numbers": ["36792831"],
"type": {"key": "/type/edition"}, "revision": 9, "publishers": ["Signet Classic"], "last_modified": {"type": "/type/datetime", "value": "2019-04-03T20:54:26.362929"}, "key": "/books/OL1017798M",
"authors": [{"name": "Mark Twain", "key": "/authors/OL18319A"}], "publish_places": ["New York"], "pagination": "xxi, 216 p. ;", "classifications": {}, "created": {"type": "/type/datetime", "value":
"2008-04-01T03:28:50.625462"}, "lccn": ["96072233"], "notes": "Includes bibliographical references (p. 213-216).", "number_of_pages": 216, "dewey_decimal_class": ["813/.4"], "isbn_10":
["0451526538"], "publish_date": "1997", "works": [{"key": "/works/OL53919W"}]}, "preview": "noview"}, "ISBN:0385472579": {"info_url": "http://openlibrary.org/books/OL1397864M/Zen_speaks", "bib_key":
"ISBN:0385472579", "preview_url": "http://openlibrary.org/books/OL1397864M/Zen_speaks", "thumbnail_url": "https://covers.openlibrary.org/b/id/240726-S.jpg", "details": {"identifiers": {"goodreads":
["979250"], "librarything": ["192819"]}, "subtitle": "shouts of nothingness", "contributors": [{"role": "Translator", "name": "Chan shuo"}, {"role": "Illustrator", "name": "Tsai Chih Chung"},
{"role": "Adaptation", "name": "Tsai Chih Chung"}, {"role": "Translator", "name": "Brian Bruya"}], "covers": [240726], "local_id": ["urn:sfpl:31223062535233"], "lc_classifications": ["BQ9265.6
.T7313 1994"], "latest_revision": 9, "contributions": ["Bruya, Brian, 1966-"], "uri_descriptions": ["Publisher description"], "genres": ["Caricatures and cartoons."], "source_records":
["marc:marc_openlibraries_sanfranciscopubliclibrary/sfpl_chq_2018_12_24_run02.mrc:125374412:1424", "amazon:0385472579"], "title": "Zen speaks", "languages": [{"key": "/languages/eng"}], "subjects":
["Zen Buddhism -- Caricatures and cartoons."], "publish_country": "nyu", "type": {"key": "/type/edition"}, "uris": ["http://www.loc.gov/catdir/description/random046/93005405.html"], "revision": 9,
"publishers": ["Anchor Books"], "last_modified": {"type": "/type/datetime", "value": "2019-11-28T22:31:10.857685"}, "key": "/books/OL1397864M", "authors": [{"name": "Zhizhong Cai", "key": "/authors
/OL223368A"}], "publish_places": ["New York"], "lccn": ["93005405"], "pagination": "159 p. :", "classifications": {}, "created": {"type": "/type/datetime", "value": "2008-04-01T03:28:50.625462"},
"url": ["http://www.loc.gov/catdir/description/random046/93005405.html"], "number_of_pages": 159, "dewey_decimal_class": ["294.3/927"], "isbn_10": ["0385472579"], "publish_date": "1994",
"work_title": ["Chan shuo."], "works": [{"key": "/works/OL1866073W"}]}, "preview": "noview"}
```


Slika 3 - Prikaz preuzete JSON datoteke s OpenLibrary API-ja u zadanom formatu

Ovim primjerom lako je primijetiti da se radi o vrlo jednostavnoj programskoj skripti koja dohvaća URL knjige putem OpenLibrary API sučelja na temelju ISBN broja bez obzira na kompleksan odgovor API sučelja. S obzirom na pristupačnost OpenLibrary API sučelja te jednostavnost sintakse programskog jezika Python ova razmjena podataka izvršena je i podaci su u potpunosti formatirani po programerovoj želji u samo 10 linija koda.

Kompleksni primjer razmjene podataka putem API sučelja koristeći Python

Na idućem prikazu programskog koda prikazan je primjer korištenja API sučelja kompleksne baze podataka poput Wolfram Alphe. Ovaj API ima veliki broj mogućnosti te odgovara na razna pitanja koja su mu postavljena. Potrebno je instalirati biblioteku, to jest modul WolframAlpha[49] pomoću *pip* alata za Python te pri programiranju dodati WolframAlpha modul u kod koristeći import komandu. Ovaj modul nam dozvoljava spajanje na WolframAlpha usluge putem API sučelja te razmjenu podataka.

```
import worlframalpha
def robot(pitanje):
	klijent = wolframalpha.Client("E5Lk5U-RPYT46XTQv")
	zahtjev_s_pitanjem = klijent.query(pitanje)
	for odgovor in zahtjev_s_pitanjem.pods:
		print("{o.title}: {o.text}".format(o=odgovor))
korisnicki_unos = str(input("Unesite Vaše pitanje:"))
robot(korisnicki_unos)
```

Ispis 7 - Korištenje WolframAlpha API sučelja programskim pristupom

Zatim je potrebno definirati funkciju, u našem slučaju "*robot*" koja će prihvatiti argument "*pitanje*" koje korisnik postavlja. Taj će taj argument proslijediti WolframAlpha poslužitelju putem API sučelja pomoću "*.query()*" funkcije. Kako bi poslani zahtjev vratio očekivani odgovor, WolframAlpha API sučelju potrebno je priložiti API ključ za autentikaciju. U ovom primjeru potrebno je svrstati API ključ u varijablu klijent, gdje će se koristiti "*.Client()*" funkciju WolframAlpha modula da se uključi API ključ u zahtjevu. Bez API ključa u zahtjevu poslužitelj bi nam odgovori kodom odgovora "403 - forbidden" ili sličnim odgovorima koji nam odbijaju pristup radi manjka odgovarajućih autentikacijskih podataka. U ovom slučaju u funkciju "*.query()*" smještena je varijabla "*pitanje*" koja je preimenovana prihvaćanjem argumenata u *robot* funkciji na dnu koda koristeći "*korisnicki_unos*" varijablu za arbitrarni unos od korisnika. Radi formata odgovora poslužitelja na upit potrebno je, prateći dokumentaciju WolframAlpha API sučelja, pokrenuti "*for petlju*" koja će proći kroz svaki odgovor u varijabli "*zahtjev_s_pitanjem*" te njenoj potkategoriji, definiranoj također od WolframAlpha dokumentacije za API, "pods". Zatim, kako bi naslov i tekst odgovora bili formatirani čitljivo potrebno je i formatirati ispis. Ispis rezultata na ovaj način daje nam širok spektar odgovora,

[49] D. Birraux, A. Buzing. Announcing the Wolfram Client Library for Python, 2019. https://blog.wolfram.com/2019/05/16/announcing-the-wolfram-client-library-for-python/ (01.09.2020)

uključujući i "pretpostavke" WolframAlpha API sučelja na naše pitanje. "Pretpostavke" pomažu Wolfram Alpha poslužitelju pri odabiru relevantnih odgovora, stoga u ovom slučaju "*.pods*" i "*for petlja*" zajedno dozvoljavaju više vrsta odgovora i njihove naslove.

Nakon funkcije "*robot*" definirana je varijabla za korisnički unos koja omogućuje pokretaču programa da upiše proizvoljno pitanje, koje je pretvoreno u *string* i proslijeđeno funkciji.

Pri pokretanju programa postavljeno je pitanje "Who is Elon Musk" i dobiven je opširni odgovor.

```
Input interpretation: Elon Musk (physicist, etc.)
Basic information: full name | Elon Musk
date of birth | Monday, June 28, 1971 (age: 48 years)
place of birth | South Africa
Image: None
Timeline: None
Physical characteristics: height | 1.81 meters
Familial relationships: Maye Haldeman | Errol Musk
Notable films: Iron Man 2 (2010) | Transcendence (2014) | Machete Kills (2013) | Revenge of the Electric Car (2012) | Why Him? (2016
) | ... (total: 9)
Estimated net worth: $19 billion (US dollars)
(as of 2018)
Wikipedia summary: Elon Reeve Musk (born June 28, 1971) is an engineer, industrial designer, and technology entrepreneur. He is a citize
n of South Africa, the United States (where he has lived most of his life and currently resides), and Canada. He is the founder, CEO and chi
ef engineer/designer of SpaceX; co-founder, CEO and product architect of Tesla, Inc.; founder of The Boring ...
Wikipedia page hits history: None
>>>
```

Slika 4 - odgovor WolframAlpha API sučelja na postavljeno pitanje 1

Pri postavljanju jednostavnog pitanja "Who is Elon Musk" API sučelju dobiven je opširan odgovor s naslovima i sadržajem. Radi se o vrlo preglednom odgovoru i svim dobivenim podacima koji su zatraženi.

Implementacija web stranice s mogućnošću razmjene podataka i datoteka pomoću API sučelja

Ovaj projekt sadrži više vrsta API sučelja. Bit će prikazan API za preuzimanje i prenošenje datoteka (*dropbox*), API za trenutno meteorološko stanje (*weather API*), API koji nasumično odabire iz liste top 100 knjiga ove godine - te prikazuje korisniku "Daily" knjigu kao preporuku , API za vijesti iz svijeta (*news API*), API za obradu datoteka i vraćanje informacija - u ovom slučaju VirusTotal API za provjeravanje malicioznosti datoteke i graf API (*graph API*) u koji se dobiveni podaci od *Dropbox* API-ja o statusu memorije dropbox računa pohranjuju u URL koji pristupom formira graf na temelju ovih podataka koji se implementira u stranicu. Ovaj projekt bit će izrađen uz pomoć HTML i CSS koda za izradu stranice te Python jezika uz Flask modul koji predstavlja poslužiteljski element. Ideja je prikazati razne načine funkcioniranja API sučelja, kako izgledaju kada se implementiraju u korisniku dostupno sučelje te kako ih poslužitelj poslužuje korisnicima i kako izgleda sama razmjena podataka putem API sučelja.

Znanjem o razmjeni informacija putem API sučelja i uspostavi poslužitelja putem Flask modula kroz Python kod vrlo je lako uspostaviti stabilno web sjedište čije stranice, kodirane u HTML i CSS kodu pružaju korisniku razne podatke. Širok je spektar ovih podataka radi opširnosti API kategorija po temi, a i mogućnosti koje nudi Python uz Flask modul.

```
from flask import Flask, render_template, send_from_directory
from flask import request
from werkzeug.utils import secure_filename
app = Flask(__name__)


@app.route("/" ,methods=["POST","GET"])
def main():
    return render_template("index.html",headtwo="MarkoHorvat")


@app.route("/menu", methods=["POST","GET"])
def menu():
    try:
        import fileLister
        global allthefiles
        allthefiles = fileLister.filelister()
        try:
```

```
            import Memstat
            freespace = Memstat.status()
            global memory
            memory = freespace.replace(" ","")
            return render_template("menu.html",allthefiles=allthefiles,
       memory=memory)
        except Exception as e:
            return render_template("menu.html", allthefiles=allthefiles,
       memory=e)
    except:
        return render_template("menu.html", allthefiles="Could not get  files.")


if __name__ == "__main__":
    app.run(debug=True, host="192.168.1.105", port=80)
```

Ispis 8 - Flask modul kao poslužitelj za web stranicu, direktoriji i funkcionalnost kroz Python kod

Radi jednostavnosti uspostave web sjedišta odabran je modul Flask[50] koji u ovom slučaju igra ulogu poslužitelja. Uz Flask modul u program unesen je i workzeug.utils dio Flask modula te njegovi podmoduli poput *request, render_template* i *send_from_directory*. Nakon unošenja modula, radi Flask sintakse potrebno je definirati ime aplikacije. U ovom slučaju aplikacija će prenijeti ime *__name__* prateći dokumentaciju Flask *frameworka*. Ova vrijednost bit će pružena Flask() funkciji kao argument i predstavljati ime aplikacije. Kako bi direktorij na stranici bio usmjeren prema HTML datoteci i ispravno učitao stranicu potrebno je definirati rutu koja će kada joj korisnik pristupi učitati definiranu datoteku. Definirali direktorija i povezivanje s datotekom u Flask modulu ne predstavlja problem. Potrebno je prvo koristiti *@app* što u Pythonu predstavlja dekoratora - koji preuzima funkciju, daje joj neka svojstva ili funkcionalnosti te ih vraća programu. Nakon definiranja dekoratora nužno je pozvati funkciju *.route()* koja prihvaća argumente putanje/direktorija na web sjedištu te listu dozvoljenih metoda za pristup. Za sada su definirane dva direktorija ili rute. Direktorij "/" podrazumijeva glavnu stranicu koju korisnik vidi pri posjetu same adrese te "/menu" kao stranicu koju korisnik može posjetiti klikom na gumb "*menu*" koji je kodiran u html datoteci. Definirati dozvoljene metode za slanje zahtjeva prema tom direktoriju od korisnika također je potrebno. U ovom slučaju to podrazumijeva GET i POST naredbe definirane u tekstualnom formatu smještene u listu i pružene kao argument funkciji *.route()*. Ako metoda nije dozvoljena, korisniku će od Flaska biti poslan kod greške. Zatim dolazi definiranje samih funkcija koje vraćaju stranicu koristeći

[50] PalletsProjects.com. Flask. https://palletsprojects.com/p/flask/ (01.09.2020)

"return render_template()" liniju čijoj se funkciji *render_template()* pružaju argumenti same web stranice kao prvi argument te ukoliko je potrebno ostale varijable kako bi se proslijedile za prikaz i/ili koristile na samoj stranici. Funkcija *main* jednostavan je primjer. U *main* funkciji korisniku je poslana datoteka "*index.html*" kako bi predstavila naslovnu stranicu te argument *headtwo* koji se implementira kroz HTML kod kako bi bio vidljiv korisniku. *Headtwo* je varijabla koja u ovom slučaju sadrži tekst korisničkog imena.

```
<ol start="0">
      {%for i in allthefiles%}
            <li>{{ i }}</li>
      {%endfor%}
</ol>
```

Ispis 9 - Čitljiv kod za Flask zapisan u HTML datoteci kao posrednik za razmjenu varijabla

Funkcija *menu,* s druge strane, komplicirana je i dohvaća informacije iz vanjskih modula. Koristeći *try* i *except* za dohvaćanje i obradu grešaka kako bi se korisniku poslužila stranica, bez obzira hoće li vanjski moduli biti funkcionalni, unesen je modul *fileLister*. Ovaj modul izlistava sve datoteke korisnika spajajući se na Dropbox API sučelje, no kako bi se ispis datoteka prikazao na stranici potrebno je smjestiti odgovor Dropbox API sučelja u varijablu kako bi se kroz *return render_template()* liniju proslijedila stranici. Zatim uz još jedan *try/Except* blok koda s istom namjenom kao i prošli unesen je modul *Memstat* čiju je vrijednost funkcije *status()* , kao i vrijednost dobivenu od *fileLister* modula važno smjestiti u globalne varijable kako bi im se omogućio lakši pristup u budućim funkcijama poput (još uvijek nedefiniranih) direktorija */upload* i */download* koji korisniku mogu omogućiti da preuzima i prenosi datoteke na svoj Dropbox račun. Ova vrijednost predstavlja status memorije Dropbox računa koju Flask svrstava u HTML kod stranice, gdje su očekivani podaci smješteni u URL drugog (*graph*) API sučelja kako bi se na stranici formirala slika grafa zauzeća memorije. Na kraju je vraćena HTML datoteka te tekstualni argumenti preuzeti od modula *fileLister* i *Memstat* što znači da korisnik pri ulasku u /*menu* stranicu vidi zauzeće svog Dropbox računa kroz graf i sve datoteke izlistane.

Ostale funkcionalnosti ovih web stranica definirane su na isti način kao *main* i *menu*. Te poput *menu* funkcije, uz *try/except* blok koda dohvaćaju strane module koji se spajaju na API sučelja i razmjenjuju podatke. Ako programer, na primjer, želi ukrasiti svoju web stranicu dnevnim novostima u svijetu može to obaviti jednostavnim pozivima besplatnom API sučelju koje ih

nudi. Obrađene novosti tada prosljeđuje Flasku koristeći isti način prijenosa podataka između modula kao i kod */menu* stranice. Korištenjem modula s kojima dolazi Python i API sučelja poput NewsAPI[51] sučelja izrada ovakvog programa vrlo je lako izvediva.

```
import requests
import json
import random


def getwsj():
    apikljuc = "a6566edbe46"
    URL = f"http://newsapi.org/v2/everything?domains=wsj.com&apiKey={apikljuc}"
    lista_novosti = []
    try:
        get = requests.get(URL)
        get = json.loads(get.text)
        clanci = get['articles']
        for clanak in clanci:
            podaci = clanak['title']
            podaci = podaci.replace("- The Wall Street Journal","")
            lista_novosti.append(podaci)
    except:
        pass
    return lista_novosti
```

Ispis 10 - Dohvaćanje vijesti iz svijeta s WallStreet Journal izvorom od NewsAPI besplatnog sučelja

S obzirom na to da se u datotekama nalaze podaci, nerijetko i oni strukturirani, svrstati ih među interesantne potencijalne dijelove za ovaj projekt nije loša ideja. Razmjena datoteka putem API sučelja poput Dropbox API-ja omogućuje razmjenjivanje datoteka uz samo nekoliko linija koda. Za autentikaciju korisnika na čiji će račun datoteke biti prenesene ili s kojeg će biti preuzete potreban je API ključ te je radi sigurnosti svaki zahtjev zabilježen.

```
def download(filename):
    import dropbox
    api_key = "YdLw7bxQVCAAAAAAAAAAG"
```

[51] NewsApi.com. Getting started with NewsAPI (dokumentacija). https://newsapi.org/docs/get-started (17.06.2020)

```
    dbx =    dropbox.Dropbox(api_key)
    dbx.files_download_to_file(\
"./downloads/{}".format(filename),"/{}".format(filename))
    return "/downloads/{}".format(filename)
```

Ispis 11 - Kod za dohvaćanje datoteka s Dropbox API sučelja

Preuzimanje datoteka s Dropbox API sučelja koristeći Python programski jezik zahtjeva samo unos Dropbox modula, koji u funkciji *.Dropbox()* prihvaća argument API ključa za autentikaciju korisnika te ime i putanju datoteke koja će biti preuzeta iz datoteka korisničkog računa na korisnikovo računalo.

```
def upload(fullpath):
    import dropbox
    api_key = "YdLw7bxQVCAAAAAAAAAAG"
    dbx = dropbox.Dropbox(api_key)
    with open(fullpath, 'r',encoding="utf-8") as thetexts:
        mydata = thetexts.read()
    justthefilename = fullpath.split("/")
    mode = dropbox.files.WriteMode.add
    dbx.files_upload(bytes("{}".format(mydata),"utf-
8"),"/{}".format(justthefilename),mode)
```

Ispis 12 - Kod za prenošenje datoteka na račun putem Dropbox API sučelja

Prenošenje datoteka na račun pomoću Dropbox API-ja koristeći Python programski jezik kompleksniji je slučaj. Osim što je, kao kod preuzimanja datoteka, potrebno unijeti modul Dropbox i unijeti API ključ funkciji *.Dropbox()*, potrebno je i specificirati putanju do datoteke koju korisnik želi prenijeti, ime datoteke te ju učitati i poslati Dropbox sučelju u *bytes* formatu (bajtovi).

Znanje o ovim tehnologijama programeru daje mogućnost izrade funkcionalne web stranice nad kojom ima potpunu kontrolu nad svakim dijelom ovog projekta, osim statusa API sučelja o kojem se brine poslužitelj samog sučelja. Srećom zamjena API sučelja u slučaju, na primjer uskraćene usluge, vrlo je lako izvediva. Prijenos datoteka također spada u razmjenu podataka putem API sučelja jer se u uobičajenom prijenosu kod API sučelja zapravo dohvaća dokument u čistom tekstualnom formatu, formatiran sintaktički od poslužiteljevog programskog sloja iza API sučelja. Više o načinu funkcioniranja API sučelja moguće je saznati kroz jednostavan primjer kreacije samog.

Izrada API sučelja

Sada, kada je dohvaćanje podataka putem API sučelja pojašnjeno važno je objasniti kako samo sučelje daje podatke na preuzimanje korisnicima. Izvrstan primjer bio bi prikaz izrade samog API sučelja i vrijednosti koje vraća.

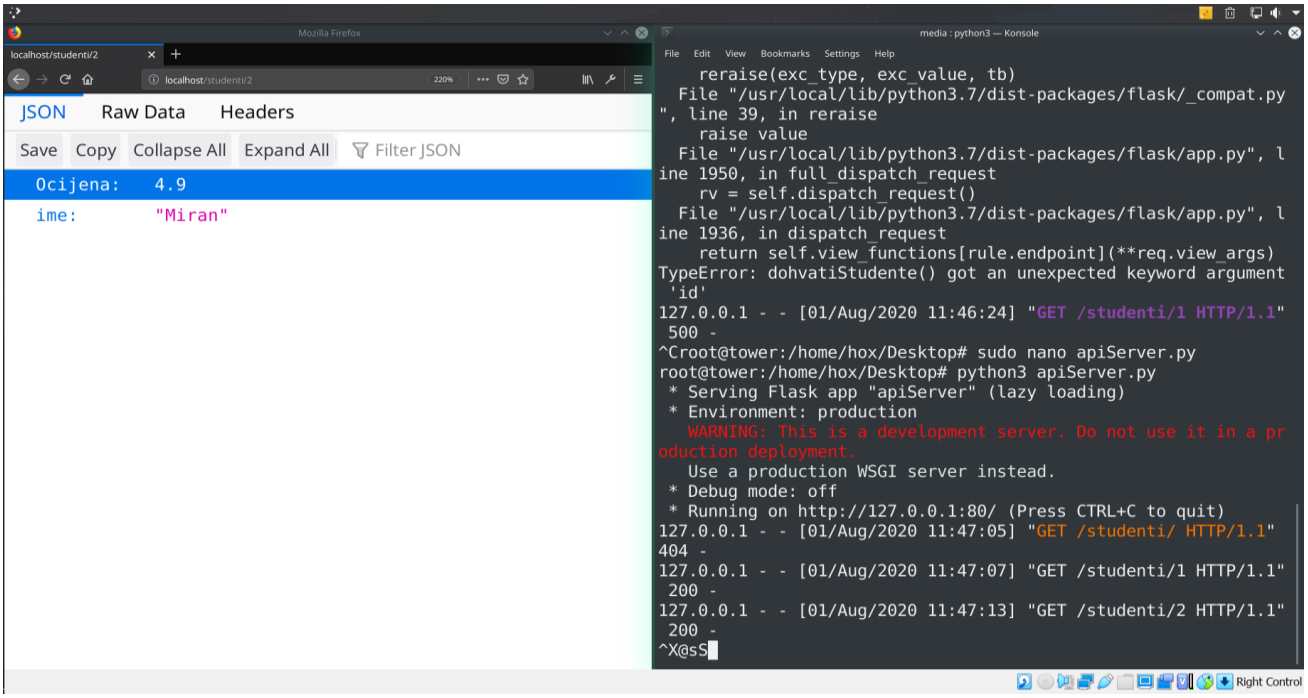
```
from flask import Flask, jsonify
app = Flask(__name__)

@app.route("/")
def helloWorld():
	return "{'porukaDana'}: {'Pozdrav Svijete'}"
studenti = [
	{"ime":"Mate", "Ocijena":5.0},
	{"ime":"Miran", "Ocijena":4.9},
	{"ime":"Filip", "Ocijena":4.5}
]
@app.route("/studenti/<id>", methods=["GET"])
def dohvatiStudente(id):
	id = int(id) - 1
	return jsonify(studenti[id])
if __name__ == "__main__":
	app.run("127.0.0.1",port=80)
```

Ispis 13 : Primjer API poslužitelja koristeći Flask modul u Python programskom jeziku

Vrlo jednostavnom *Python* skriptom, koristeći Flask uspješno je izrađen poslužitelj koji nudi podatke korisnicima u JSON formatu. Modifikacijom funkcije *dohvatiStudente* omogućili smo i modificiranje rezultata koji će se pružiti korisniku-programeru. Varijabla *id* je ona koja omogućuje korisniku dohvaćanje specifičnih informacija. Ovo znači da je uspješno izrađeno API sučelje koje vraća rezultate ovisno o zahtjevu korisnika.

Slika 5 - primjer jednostavnog odgovora API sučelja izrađenog koristeći Pythonov modul Flask

Pri posjetu na glavnu stranicu prikazana je jednostavna poruka u JSON formatu. Ova stranica korisniku ne predstavlja puno koristi jer dobiva sve informacije na jednoj stranici što ponekad nije dosta, ili je čak previše podataka na jednom mjestu. Iako se *porukaDana* mijenja svaki dan što bi korisniku išlo u korist, on i dalje ne može modificirati zahtjev i zatražiti specifični dio odgovora. Nekima od korisnika će ovo biti dovoljno, no važno je pobrinuti se da je vrlo jednostavno preuzeti točno one informacije koje korisnik želi pri izradi API sučelja. Ovo svojstvo čini API sučelje puno boljim i pristupačnijim. Srećom - ova mogućnost već je napisana u kodu. Korisnik u URL unosi "*/student/2"* na kraj što mu vraća podatke o drugom studentu po redu. Ovim zahtjevom korisnik je zatražio podatke od samo drugog studenta po redoslijedu, što mu je ovo API sučelje i pružilo.



Slika 6 - Log zapisi izrađenog API poslužitelja te prikaz odgovora na specifični zahtjev korisnika

Poslužitelj (Flask) bilježi korisnički posjet i ispisuje ga vlasniku, dok korisniku pruža podatke o specifičnom zahtjevu za studenta 2. Drugi student je "*Miran*", a njegova ocjena iz imaginarnog predmeta je 4.9. Ovaj zahtjev i odgovor dokaz su funkcionalnosti ovog API sučelja i jednostavnosti korištenja istog.

Izrada vlastitog API sučelja uz Flask i Python je izvršena, s obzirom da se radi o vrlo jednostavnom sučelju potrebno je dodati još podataka što će biti lako izvedivo jer je funkcionalnost modificiranja URL-a prema korisniku preferiranom broju unosa već ugrađena i vraća vrijednosti.

Zaključak

Zamisao aplikacijsko-programskih sučelja omogućila je jednostavnost razmjene podataka bez direktnog zadiranja programera u baze podataka te s vrlo jednostavnim programski realiziranim zahtjevima prema poslužitelju daje programeru šansu da spretno i brzo dohvati veliku količinu podataka, programski ih obradi i implementira u svoj projekt. Odabrani formati koje API sučelja koriste također pridonose novim mogućnostima radi mogućnosti obrade podataka s bilo kojim od programskih jezika. Izrada potpuno funkcionalnog web sjedišta, čiji se podaci temelje na podacima dobivenih od API sučelja, danas je lakša nego ikad uz pomoć sintaksno jednostavnih programskih jezika poput Pythona, *frameworcima* poput Flaska i API sučeljima te modulima poput "Requests" ili "Urllib" za dohvaćanje podataka sa sučelja. Ovaj rad prikazuje kako s nekoliko linija koda, koristeći jednostavne module i znanje o razmjeni podataka putem API sučelja, programski ostvariti u potpunosti funkcionalnu stranicu s mnoštvom mogućnosti za korisnika i s implementiranim raznim svojstvima koje korisnik treba za efikasan rad na stranici. Rad naglašava važnost API sučelja kao izvrsnog izvora podataka za izradu programa za svakodnevno korištenje, programa u organiziranim projektima i programa za poslovni svijet što je i prikazano na praktičnom primjeru. Ovaj praktični primjer je projekt čiji se proizvod može koristiti u svakodnevnom korištenju kao svojevrsna zamjena za Dropbox aplikaciju s mogućnostima preuzimanja datoteka s računa te prenošenja datoteka na račun, dohvaćanje informacija o zauzeću prostora računa kao i formiranje grafa zauzeća sve na jednoj stranici koristeći samo dva besplatna API sučelja. Funkcionalnosti ovog projekta su mnoge, uključujući stranicu "infoboard" koja dohvaća informacije o vremenskoj prognozi, vijesti u svijetu i kao zanimljiv dodatak preporučuje korisniku novu knjigu za čitanje. No glavna namjena ovog projekta nije napraviti jednostavniju zamjenu za Dropboxovu aplikaciju koja zauzima manje mjesta na disku i radi otprilike jednakom brzinom, već predstaviti važnost aplikacijsko-programskih sučelja informacijskim stručnjacima, programerima i svim ljubiteljima informacijskog svijeta kroz tehnologiju. Razmjena podataka aplikacijsko-programskim sučeljima čini svijet informacija i podataka puno naprednijim i funkcionalnijim. Samom činjenicom da je razmjena informacija ubrzana bilo koji način, u ovom slučaju radi dolaska API sučelja, kao djeca doba tehnologije i interneta spoznajemo koliku ulogu u napretku svijeta API sučelja donose. Sve dok razmjena podataka praktički drži cijeli današnji svijet informacija i znanja na sebi, a pogotovo razmjena putem interneta, novi i bolji načini za razmjenu su dobrodošli. Ako se radi o strojno čitljivim podacima, oni automatski nose veliku prednost programskom svijetu i bilo kojem

informacijskom području jer im je moguće programski pristupiti i dohvatiti ih za obradu. NASA, NSA, Apple, Facebook, Google, Youtube, Instagram i mnogi drugi koriste API sučelja ne samo kako bi korisnicima omogućili nove funkcionalnosti već kako bi izgradili unutrašnjost svojih programskih rješenja i unaprijedili im efektivnost. Povjerenje korisnika API sučelja dobila su činjenicom da ih većina nudi točne i provjerene te strojno čitljive podatke. Ova činjenica čini API sučelja svojevrsnim oazama podataka gdje korisnik sam dohvaća podatke i od njih formira informacije te znanje. Ako čitatelj shvati što su API sučelja i kako se koriste za razmjenu podataka, onda će cilj ovog rada biti ostvaren.

Popis literature

Članci na mrežnim stranicama

Birraux, D; Buzing, A. Announcing the Wolfram Client Library for Python, 2019. https://blog.wolfram.com/2019/05/16/announcing-the-wolfram-client-library-for-python/ (01.09.2020)

Crockford, D. The application/json Media Type for JavaScript Object Notation (JSON), 2006. https://tools.ietf.org/html/rfc4627 (16.07.2020)

Ecma-international.com. Standard ECMA-404: The JSON Data Interchange Syntax (Drugo izdanje), 2017. https://www.ecma-international.org/publications/standards/Ecma-404.htm (05.07.2020)

Fielding, R.; Gettys, J.; Mogul, J.; Berners-Lee, T. i sur. Hypertext Transfer Protocol: HTTP/1.1, 1999. https://tools.ietf.org/html/rfc2616 (16.07.2020)

Fielding, R.; Reschke, J; Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, 2014. https://tools.ietf.org/html/rfc7231 (17.06.2020)

Goldfarb, C. F. Document Processing and Relating Communication: Technical Corrigendum for WebSGML Adaptations, 1997. http://xml.coverpages.org/wg8-n1929-g.html (30.07.2020)

Internacionalna organizacija za standardizaciju. ISO 3166: COUNTRY CODES https://www.iso.org/iso-3166-country-codes.html (30.07.2020)

Internacionalna organizacija za standardizaciju. ISO 8879:1986 Information processing : Text and office systems — Standard Generalized Markup Language (SGML), 1986. https://www.iso.org/standard/16387.html (14.07.2020)

Klensin, J.; Role of the Domain Name System (DNS), 2003. https://tools.ietf.org/html/rfc3467 (16.07.2020)

Leach, P.; Mealling, M.; Salz, R. A Universally Unique IDentifier (UUID) URN Namespace, 2005. https://tools.ietf.org/html/rfc4122

Koebler, R. RPC: JSON-RPC, 2008. http://www.simple-is-better.org/rpc/ (08.09.2020)

Mell, P.; Grance, T. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology, 2011. http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf (09.08.2020)

Microsoft.com. ASP.NET Web APIs. https://dotnet.microsoft.com/apps/aspnet/apis (09.08.2020)

NewsApi.com. Getting started with NewsAPI (dokumentacija). https://newsapi.org/docs/get-started (17.06.2020)

Openid.net. Welcome to OpenID Connect. https://openid.net/connect/ (05.07.2020)

PalletsProjects.com. Flask. https://palletsprojects.com/p/flask/ (01.09.2020)

Python.org. Json (Documentation): JSON encoder and decoder. https://docs.python.org/3.0/library/json.html (01.09.2020)

Python.org. Urllib.request (Documentation): extensible library for opening URLs. https://docs.python.org/3.0/library/urllib.request.html (09.09.2020)

RapidApi.com. API glossary: API Key – What is an API Key. https://rapidapi.com/blog/api-glossary/api-key/ (04.07.2020)

RapidApi.com. Types of APIs, 2020. https://rapidapi.com/blog/types-of-apis/ (08.09.2020)

Shafranovich, Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files, 2005. https://www.ietf.org/rfc/rfc4180.txt (08.09.2020)

Tschofenig, H.; Leiba, B.; Cook, B.; Van Eijk, R. Browser Support for the open authorization (OAUTH) protocol, 2011. https://www.w3.org/2011/identity-ws/papers/idbrowser2011_submission_32.pdf (02.07.2020)

W3C.org. Extensible Markup Language (XML) 1.0 (Peto izdanje), 2008. https://www.w3.org/TR/REC-xml/ (02.07.2020)

W3C.org. RFC 2616 Fielding chapter 9: Method definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html (01.07.2020)

W3C.org. RFC 2616 Fielding chapter 10: Status code definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html (01.07.2020)

W3Schools.com. XML SOAP. https://www.w3schools.com/xml/xml_soap.asp (04.07.2020)

xml.silmaril.ie. The XML FAQ (Frequently Asked Questions) http://xml.silmaril.ie/responsible.html (14.07.2020)
xml.silmaril.ie. The XML FAQ (Frequently Asked Questions) http://xml.silmaril.ie/standards.html (14.07.2020)
Yergeau, F. UTF-8, a transformation format of ISO 10646, 2003. https://tools.ietf.org/html/rfc3629 (01.07.2020)

Mrežne stranice

GraphQL: A query language for your API. https://graphql.org (11.08.2020)
JSON.org. https://www.json.org/json-en.html (21.08.2020)
OpenAPI initiative. https://www.openapis.org/about (30.08.2020)
Swagger: API Documentation and Design Tools for Teams. https://swagger.io/specification/ (11.08.2020)
The Official YAML Website. https://yaml.org (22.08.2020)
XML.COM. https://www.xml.com/pub/a/98/10/guide0.html (21.08.2020)
XMLRPC.COM. http://xmlrpc.com (22.08.2020)

Časopisi

Birrell, A.D.; Nelson, B.J. Implementing Remote Procedure Calls // ACM Transactions on Computer Systems, Vol. 2, No. 1, 1984. http://web.eecs.umich.edu/~mosharaf/Readings/RPC.pdf (31.08.2020)

Članci u elektroničkim časopisima

Kumari, V. Web Services Protocol: SOAP vs REST // International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. 4, No. 5, 2015. http://ijarcet.org/wp-content/uploads/IJARCET-VOL-4-ISSUE-5-2467-2469.pdf (17.07.2020)

Elektronički izvori

Ashby, D; Jensen, C.T. APIs for dummies: 3rd IBM Limited Edition, 2018. https://www.ibm.com/downloads/cas/GJ5QVQ7X (15.08.2020)
CISCO.COM. REST API Chapter 3: Client authentication. https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/restapi/restapi/RESTAPIclient.pdf (31.08.2020)
Ecma-international.com. ECMAScript Language Specification (treće izdanje), 1999. http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262, 3rd edition, December 1999.pdf (05.07.2020)
Ecma-international.com. Standard ECMA-262: ECMAScript Language Specification (izdanje 5.1), 2011. https://www.ecma-international.org/ecma-262/5.1/#sec-15.12 (07.07.2020)
International business machines (IBM) Corporation. Secure Sockets Layer/Transport Layer Security, 2015. https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_73/rzain/rzainpdf.pdf (13.08.2020)
Kuserbajn, S. REST Arhitektura, 2016. https://zir.nsk.hr/islandora/object/vus:377/preview (01.09.2020)
Moraes, F. (HTML.com). HTML Cheat sheet. https://html.com/wp-content/uploads/html-cheat-sheet.pdf (01.09.2020)
Potti, P.K. On the Design of Web Services: SOAP vs. REST, 2011. https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1139&context=etd (08.09.2020)
Schweikardt, N.; Schwentick, T.; Segoufin, L. Database theory: Query languages, 2011. http://www.lsv.fr/~segoufin/Papers/Mypapers/DB-chapter.pdf
Shiflett, C. HTTP: Developer's handbook, 2003. https://books.google.hr/books?id=oxg8_i9dVakC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false (30.08.2020)
Smith, T.S. Uniform Resource Locators (URLs): Powerful Reference Tools for Librarians and Information Professionals, 1996. https://files.eric.ed.gov/fulltext/ED401942.pdf (27.06.2020)

Ostali izvori

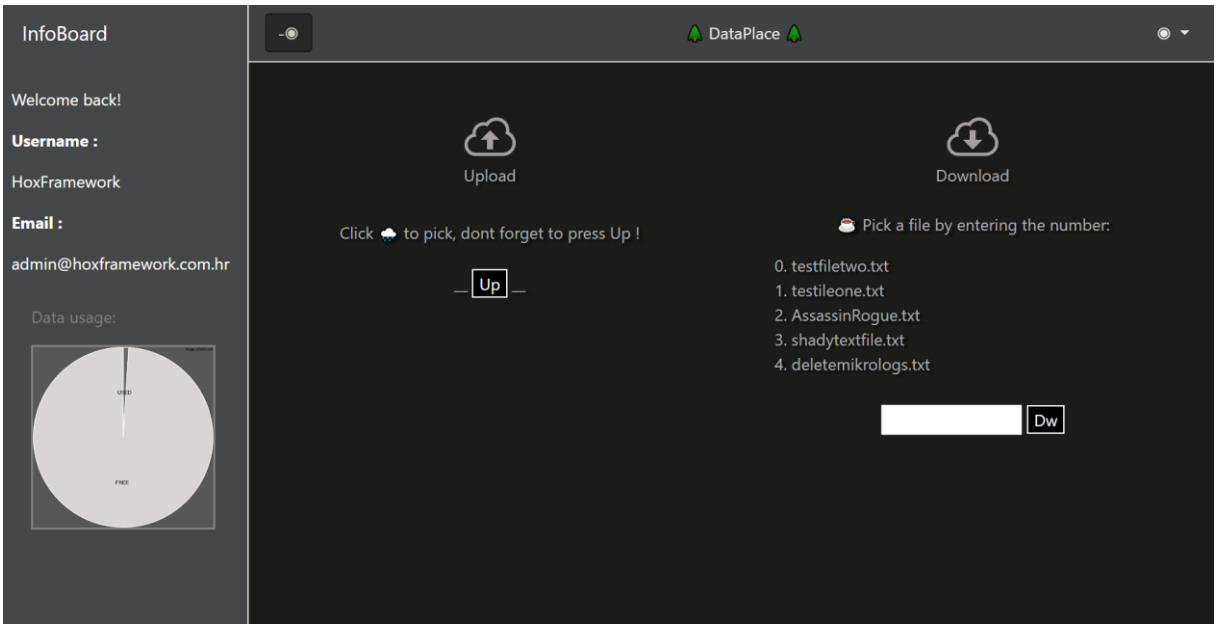
Bortenschlager, M.; Willmott, S. THE API OWNER'S MANUAL. https://www.redhat.com/cms/managed-files/mi-3scale-api-owners-manual-ebook-f7865-201706-en.pdf (02.09.2020)

Lane, K. API 101, 2013. https://s3.amazonaws.com/kinlane-productions/whitepapers/API+Evangelist+-+API+101.pdf (10.09.2020)

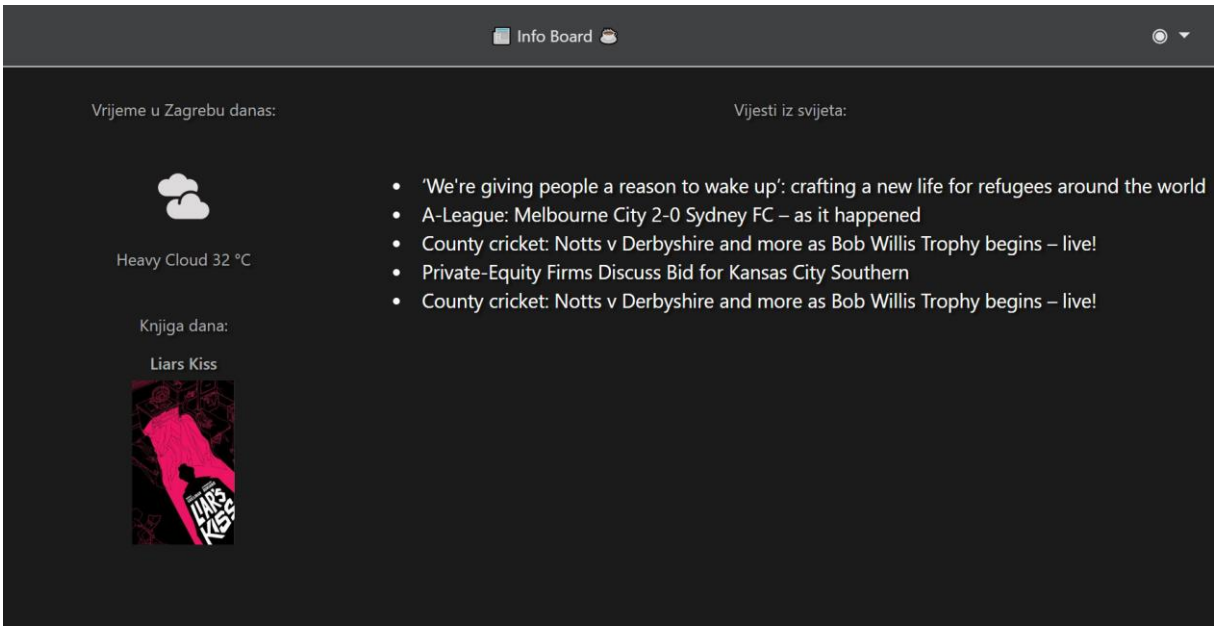
SWIFT.com. Application Programming Interfaces Delivering a global platform for the financial services API economy, 2018. https://www.swift.com/sites/default/files/documents/swift_standards_whitepaper_api.pdf

Prilozi

Izrađena web stranica:



Slika 7 - Menu stranica



Slika 8 - Infoboard stranica

Dodatni moduli i lakoća integracije novih funkcija u stranicu:

```
def scanner(file):
    from virus_total_apis import PublicApi as vtpapi
    import hashlib
```

```
    import json
    api_key = "8fa81965aff2"
    vt = vtpapi(api_key)
    md5 = hashlib.md5()
    with open(file,'rb') as newfile:
        newfile = newfile.read()
        md5.update(newfile)
        EICAR_MD5 = md5.hexdigest()
    response = vt.get_file_report(EICAR_MD5)
    respdata = response['results']
    try:
        outdata = "Virustotal says: {0} are infected out of {1} different scans".format(respdata['positives'], respdata['total'])
        return outdata
    except KeyError:
        try:
            scannerthing = vt.scan_file(file)
            outdata = scannerthing['results']['verbose_msg']
            return outdata
        except Exception as e:
            outdata = "Something went wrong. {}".format(e)
            return outdata
```

Ispis 14 - Kod za provjeru virusa datoteke, primjer lake integracije dodatnih modula u sučelje

```
@app.route("/vcheck", methods=["POST","GET"])
def vcheck():
    stats = "[Status] Waiting..."
    return render_template("vcheck.html", totalvirstat=stats)

@app.route("/vcheckon", methods=["POST","GET"])
def vcheckon():
    if request.files['vchecker']:
        thevirus = request.files['vchecker']
        thevirus.save(os.path.join("C:/putanja/vcheck/", thevirus.filename))
        import vcheck
        scanresult = vcheck.scanner(os.path.join("C:/putanja/vcheck/", thevirus.filename))
        stats = "[Status] Upload successful."
        return render_template("vcheck.html", totalvirstat=stats,scanresult=scanresult)
```

```
    elif not request.files['vchecker']:
        stats = "[Status] No file given."
        return render_template("vcheck.html", totalvirstat=stats)
    else:
        return render_template("vcheck.html", totalvirstat="Something went wrong.")
```

Ispis 15 - kod integriran u Flask poslužitelja

```
<p>{{ totalvirstat }}</p> <br / >
<p><b>{{ scanresult }}</b></p>
```

Ispis 16 - HTML kod pomoću kojeg Flask šalje varijablu stranici te ju ona učitava na primjeru modula za provjeru virusa

University of Zadar

Department of information sciences

Information sciences

Data exchange using application-programming interfaces

Final paper

Student: Filip Omazić

Mentor: Dr. sc. Krešimir Zauder

Zadar, 2020.

Summary

In this final paper application programming interfaces will be explained in detail. The goal of this paper is to create a fully functional web application that uses application programming interfaces in different contexts in order to explain their usage by gathering information from API interfaces and displaying it to the user. Considering that a big number of API (application programming interfaces) are free to access and use, the creation of this project will be significantly easier. The web application will use Python programming language as a server implemented using the flask framework. A simple way of creating the whole application is picked and yet it will allow for more complex websites to be implemented, and more complex modules to be added easily. The advantage of the project is, along with the speed and simplicity the fact that it allows for a great number of possibilities within a few megabytes that it takes up. The purpose is to introduce API interfaces and the construction of the website that uses API interfaces to information experts and all programmers trough a simple programming and coding foundation. Along with that the idea is to learn about the formats used in data exchange and the whole communication in data exchange using API interfaces. Efficiency and speed of API interfaces allow the users to get the data formatted the way they expect it to be fast and easy. This data can later be edited and processed with knowledge about programming and displayed to the user trough a website.

Keywords

Application programming interface (API), data exchange, programming, Python, JSON, XML