Količina podataka koja danas putuje internetom je velika, ali i kompleksna. Svi ti podaci koji kolaju internetom se koriste za stvaranje izvještaja, donošenje zaključaka, kao i za razne analize kod mnogih tvrtki, institucija i sličnog. Pravi izazov je, zapravo, manipulirati tokom tih podataka te njihovom analizom.

Velike tvrtke razvile su, tijekom proteklih godina, zahtjevne metode koje im dozvoljavaju da prate razvoj tržišta, njihovu poziciju na tržištu, efikasnost usluga koje nude i slično. Kako bi to sve bilo moguće, potreban im je pristup velikim količinama aktualnih podataka.

S takvim količinama podataka, gotovo je nemoguće raditi na vlastitom računalu. „Streaming“ podataka u stvarnom vremenu je postao izazov za mnogo tvrtke.

[1] Big Dana Challanges, <http://www.dbjournal.ro/archive/13/13_4.pdf>

Ljudi skupljaju podatke od početka svog postojanja. Teško je pojasniti sve razloge za ovakav skok u količini podataka. Objašnjenje se može pronaći u nastanku interneta. Ljudi svakodnevno koriste Internet i njihova aktivnost se koristi za komercijalne i znanstvene svrhe. Posebno, pokušavaju se personalizirati prostori na internetu pojedincima korištenjem prikupljenih podataka. U globalu, nada se da, ako bi se svi ti podaci mogli ispravno i efektivno koristiti, znanost bi dosegnula svoj vrhunac, a tehnologija bi postala adaptivna, personalizirana i robustna. To bi, na primjer, mogli uočiti posebno u medicini, gdje bi se onda podaci jednog pacijenta mogli kombnirati i uspoređivati s podacima drugih te bi svatko mogao dobivati personalizirane tretmane, upravo onakve kakvi najviše odgovaraju pojedincu.

U radu s velikim količinama podataka postoje mnogi izazovi koji kreću od prikupljanja pa sve do analize i korištenja podataka. Neki od njih su rad s distribuiranim izvorima podataka, validacija podataka, rad s različitim formatima i strukturama, osiguravanja integriteta i sigurnosti, razvijanje metoda za vizualizaciju velikih skupova podataka i sl.

[2] Frontiers in Massive Data Analysis, [https://nap.nationalacademies.org/catalog/18374/frontiers-in-massive-data-analysis#](https://nap.nationalacademies.org/catalog/18374/frontiers-in-massive-data-analysis)

Što su opće veliki skupovi podataka? Postoje mnoge definicije. Jedna od njih povezuje velike skupove podataka s tri svojstva na slovo „V“ prema engleskim nazivima: volumen (engl. *volume*), brzina (engl. *velocity*) i raznolikost (engl. *variety*). Konkretno se to odnosi na povećanje u količini podataka, povećanje brzine kojom se oni stvaraju te povećanje broja formata i oblika u kojima se pojavljuju. Kasnije se u ovu definiciju dodalo i četvrto „V“, a o to je istinitost (engl. *veracity*). Istinitost uvodi pitanja ispravnosti i nesigurnosti u sadržaj podataka kao i rezultate analize podataka.

[3] <https://arxiv.org/pdf/1309.5821.pdf>

Danas se koriste mnoge tehnologije za rad s velikim skupovima podataka. Kako bih odabrao valjanu tehnologiju, važno je znati njena svojstva. Tvrtke koriste tehnologije za rad s velikim skupovima podataka kako bi dobile bolje uvide u svoj rad i donosili profitabilnije odluke. Tehnologije koje se koriste za rad s velikim skupovima podataka su zapravo softveri koji se primarno koriste za analizu, procese i izvlačenje informacija iz velikih skupova podataka složenih struktura kojima se ne može upravljati koristeći tradicionalne tehnologije za procesiranje podataka (kao npr. RDBMS, datotečni sustavi i sl.).

Tipovi tehnologija za rad s velikim skupovima podataka:

Operativne tehnologije

* To se odnosi na podatke koji se stvaraju iz svakodnevnih aktivnosti, poput Internet transakcija, društvenih platformi i sličnog. Takvi podaci su neobrađeni. Takve tehnologije koriste na primjer za online kupovine, za društvene mreže ili podatke po zaposlenicima.

Analitičke tehnologije

* Mogu se vidjeti kao modificirane varijante koje su kompleksnije od operativnih. Analitičke se tipično koriste kada se trebaju mjeriti performanse i kada se trebaju donijeti kritične poslovne odluke temeljene na izvještaju dobivenom analize. Koriste se za prognozu vremena, medicinska izvješća i slično.

Tehnologije se mogu podijeliti i na:

* Spremanje podataka
* Rudarenje podataka
* Analiza podataka
* Vizualizacija podataka

[4] <https://www.interviewbit.com/blog/big-data-technologies/>

Spremanje podataka

Zbog svojih veličina, velikim skupovima podataka se rukuje koristeći drugačija spremišta od tradicionalnih. Tvrtke poput Googlea i Amazona imaju velike podatkovne centre koji mogu spremati i obrađivati podatke s minimalnim kašnjenjem. To nam pokazuje da tradicionalni USB-ovi i vanjski diskovi nisu korisni za rad s velikim skupovima podataka.

The data warehouse

Skladište podataka (engl. *Data warehouse*) je postupak skupljanja i upravljanja podacima iz različitih izvora za sakupljanje informacija. Tipično se koriste za spajanje i analizu podataka iz različitih izvora.

The data lake

Jezero podataka (engl. *Data Lake*) je središnji spremišni repozitorij koji sprema podatke iz raznih izvora. Može spremati podatke u strukturiranom, polu-strukturiranom ili nestrukturiranom obliku.

Za razliku od jezera podataka, skladište podataka je repozitorij strukturiranih i filtriranih podataka koji se već oblikovao za posebnu svrhu.

NAS

NAS (engl. *Network Attached Storage*) je uređaj za spremanje podataka kojem se može pristupiti spajanjem na Internet. Sadrži procesore i operacijske sustave koji mu dozvoljavaju da vrti aplikacije i jednostavno dijeli datoteke između autoriziranih korisnika.

The Cloud

Pomoću oblaka, podaci se spremaju online i mogu se dohvatiti od bilo kuda, bez potrebe za direktnim pristupom računalu.

Spremište objekata

Svi podaci se spremaju u veliki repozitorij koji se može distribuirati preko više fizičkih spremišnih uređaja.

[5] <https://www.bocasay.com/how-to-store-big-data/>

Velike količine podataka neprestano okružuju ljude, počevši od korištenja društvenih mreža, poput Facebooka ili Instragrama, pa sve do zdravstvenih usluga, vremenskih prognoza, GPS-a. Kod svih njih neprestano teku podaci bez kojih bi njihov rad bio nemoguć. Aplikacije danas, prikupljanjem podataka, pokušavaju što više personalizirati svoje sadržaje korisnicima. Kako bi to bilo moguće, prikuplja se gotovo sve što korisnik radi dok koristi aplikaciju te se analizom i obradom odredi kakav će se sadržaj prikazati.

U nastavku ovog rada otkrivaju se problematike prikupljanja, analize i vizualizacije podataka te se prikazuje primjer toka podatak. Detaljnije prikazuje sljedeća slika:

SLIKA!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Kao izvor podataka koristit će se Spotify API (<https://developer.spotify.com/documentation/web-api>). Spotify je aplikacija za slušanje glazbe koja prikupljanjem podataka o korisniku može preporučiti pjesme ili popise pjesama za reprodukciju. Podatke prikuplja Apache Kafka, distribuirana platforma otvorenog koda za strujanje događaja koju koriste tisuće tvrtki za stvaranje visoko učinkovitih podatkovnih cjevovoda [6]. Pomoću Confluent Platform platforme, stvara se Kafka poveznica (engl. *Kafka Connection*) s Microsoft Azure Cloudom. Confluent Platform je platforma koja omogućuje organiziranje i upravljanje podacima iz mnogo različitih izvora s jednim pouzdanim sustavom visokih performansi [7]. Podaci putuju i spremaju se u *json* obliku unutar Azure Blob Storagea, koji se koristi za masivnu i sigurnu pohranu objekata [8].

Azure Databricks se kasnije koristi za transformaciju podataka pomoću Apache Sparka i Apache Iceberga. Unutar Databricksa kreira se radni prostor u kojem se pišu naredbe, u ovom slučaju u PySparku. PySpark je Pythonov API namijenjen za Apache Spark. Za samu transformaciju podataka koristi se Spark. Podaci se prepisuju u Iceberg tablice te se u *parquet* obliku zapisuju natrag u Azure Blob Storage. Nakon prikupljanja, transformacije i pohrane podataka, podaci se vizualiziraju unutar Power BI-ja.

[6] <https://kafka.apache.org/>

[7] <https://docs.confluent.io/platform/current/overview.html>

[8] <https://azure.microsoft.com/en-us/products/storage/blobs>

Izvor podataka predstavlja početnu lokaciju na kojoj se podaci stvaraju. Konkretno, izvor podataka može biti baza podataka, neprekinuta datoteka (engl. *flat file*), mjerenja dobivena s uređaja, podaci prikupljeni s web izvora ili bilo koja od bezbroj statičkih ili tokovnih podatkovnih usluga kojih ima u izobilju na internetu [9].

Izvori podataka trebaju pomoći korisnicima i aplikacijama da se povežu i premjeste podatke tamo gdje trebaju biti. Oni prikupljaju relevantne tehničke informacije na jednom mjestu i skrivaju ih kako bi se potrošači podataka mogli usredotočiti na obradu i identificirati kako najbolje iskoristiti svoje podatke [9].

[9] <https://www.talend.com/resources/data-source/#:~:text=A%20data%20source%20is%20the,process%20accesses%20and%20utilizes%20it>.

Većina izvora podataka može se podijeliti u dvije glavne kategorije: strojni izvor podataka i datotečni izvor podataka[10].

Kod strojnog izvora podataka, podaci se stvaraju na korisnikovom uređaju. Taj uređaj može biti njegovo računalo, mobilni telefon, IoT ili neki drugi uređaj. Podaci su dostupni korisniku koji je trenutno prijavljen u sustav i ne mogu se dijeliti s drugim uređajima. Dalje se mogu dijeliti na korisnikove izvore podataka, kod kojeg su podaci dostupni samo određenom korisniku, i sistemske izvore podataka, kod kojeg su podaci dostupni svim sistemskim korisnicima. Neki primjeri su zapisi u mrežnom prometu, zapisi sustava i aplikacija, izlazi iz senzora, podaci o događajima na IoT uređajima, rezultati upisa na bazama podataka itd.[10].

Datotečni izvori podataka nisu dodijeljeni određenim strojevima, aplikacijama ili korisnicima. Mogu se dijeliti između uređaja, a obično su pohranjeni u zasebnim tekstualnim datotekama. Takvi izvori podataka uključuju tablice, tekstualne datoteke, PDF datoteke, slike, audio i video datoteke [10].

Podaci se prenose uz pomoć postojećih mrežnih protokola. Protokol za prijenos datoteka (engl. *File Transfer Protocol – FTP*) i protokol za prijenos hiperteksta (engl. *HyperText Transfer Protocol – HTTP*) najčešće su korišteni. Ostali protokoli za dohvaćanje podataka između sustava, posebice na webu, uključuju NFS (engl. *Network File Sharing*), SMB (engl. *Server Message Block*), SOAP (engl. *Simple Object Access Protocol*), REST (engl. *REpresentational State Transfer*) i WebDAV (engl. *Web-based Distributed Authoring and Versioning*) [10].

Druga metoda prijenosa podataka od izvora do odredišta je korištenje aplikacijskih programskih sučelja (engl. *application programming interface - API*) koje pružaju web stranice, umrežene aplikacije i druge usluge [10].

[10] <https://www.alphaservesp.com/blog/what-is-a-data-source-definitions-types-examples>

API je kod koji omogućuje komunikaciju dvaju softverskih programa. API definira kako razvojni programer treba zatražiti usluge od operacijskog sustava (OS) ili druge aplikacije i izložiti podatke unutar različitih konteksta i na više kanala.

API se sastoji od:

* Specifikacije koja opisuje kako se informacije razmjenjuju između programa u obliku zahtjeva i povratu potrebnih podataka
* Softverskog sučelja koji je napisan prema tim specifikacija i objavljen za korištenje

Za softver koji želi pristupiti značajkama i mogućnostima API-ja kaže se da ga poziva, a za softver koji stvara API kaže se da ga objavljuje[11].

[11] <https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API>

Spotify API je REST API koji se može koristiti za povlačenje Spotify podataka. Što točno znači da je API RESTful? Ukratko, ako se API shvati kao skup pravila koja dopuštaju jednom dijelu softvera da komunicira s drugim, tada se REST može shvatiti kao standardni skup pravila kojeg programeri slijede za stvaranje API-ja. Kako bi se API smatrao RESTful, mora ispoštovati određena pravila:

* Treba omogućavati **klijent-poslužitelj** arhitekturu sa zahtjevima kojima se upravlja putem HTTP-a.
* Svi zahtjevi trebaju biti **neovisni** jedni o drugima.
* API bi trebao podatke spremiti u **predmemoriju** (engl. *cache*).
* Treba postojati **uniformno sučelje** između komponenti tako da se informacije prenose u standardnom obliku.
* Klijent koji podnosi zahtjev ne mora znati komunicira li sa stvarnim poslužiteljem ili s posrednikom, tj. arhitektura je **slojevita**.
* Treba omogućiti slanje izvršnog koda s poslužitelja klijentu na njegov zahtjev.

Pravilo klijent-poslužitelj RESTful API-ja znači da bi trebalo biti moguće primiti dio podataka (resurs – engl. *resource*) kada se pristupi unaprijed definiranom URL-u [12].

Zahtjevom na Spotify API, mogu se dobiti podaci o glazbenicima, njihovim albumima, pjesmama, popisima za reprodukciju itd..

Jedan primjer unaprijed definiranog URL-a kojem se može pristupiti je sljedeći zahtjev za popisom za reprodukciju:

<https://api.spotify.com/v1/playlists/3cEYpjA9oz9GiPac4AsH4n>. Pristupanjem tom URL-u u nekom od preglednika dobije se *json* odgovor koji vjerojatno glasi:

{

"error": {

"status": 401,

"message": "No token provided"

}

}

Ono što se zapravo dogodilo je da je zahtjev odbijen zbog nevaljane autentifikacije. Kako bi se uspješno autentificirao, korisnik treba predati tzv. „*access token*“ ili pristupni token [13].

Autorizacija se odnosi na postupak davanja dopuštenja pristupa korisniku ili aplikaciji Spotify podacima i značajkama (npr. aplikacija treba dopuštenje korisnika za pristup njihovim popisima za reprodukciju) [15].

Prije početka rada sa Spotify API-jem potrebno je imati Spotify račun za razvojne programere. Taj je račun isti kao račun za Spotify aplikaciju i ne treba Spotify Premium verziju računa. Sljedeći korak je stvoriti aplikaciju unutar Spotify API-ja ( LINKKKK NA APIIII) klikom na *create an app* gumb. Kada se to učini moguće je dohvatiti javni i privatni ključ koji su potrebni kako bi se API mogao koristiti [14].

Kada se posjeti aplikacija koja se prethodno stvorila, tamo se mogu dohvatiti dva parametra: *client ID* i *client secret*. Oba su potrebna za autentifikaciju i dohvat pristupnog tokena. Nakon dohvaćanja tih dvaju parametara, može ih se spremiti lokalno u zasebnu datoteku, recimo *.env*.

Pristup zaštićenim resursima određen je jednim ili više opsega (engl. *scope*). Opsezi omogućuju vašoj aplikaciji pristup određenim funkcijama (npr. čitanje popisa za reprodukciju ili samo strujanje podataka) u ime korisnika. Skup opsega koje ste postavili tijekom autorizacije određuje dopuštenja pristupa koja se od korisnika traži da odobri. Detaljne informacije o opsegima možete pronaći u dokumentaciji opsega (<https://developer.spotify.com/documentation/web-api/concepts/scopes>) [15].

Jedna je vrsta opsega, koji će se koristiti u primjeru, da se autentificira bez posebnog korisnika. Na taj način moguće je koristiti općenite značajke Spotifyja, a ne mogu se vidjeti statistike vezane uz posebnog korisnika, kao što su njegovi popisi za reprodukciju. Za takvu autentifikaciju potrebni su *client ID*  i *client secret*[14].

Nakon prijave u Spotify API sučelje, izrade aplikacije i dohvaćanja potrebnih vjerodajnica, slijedi pisanje programskog koda za stvaranje pristupnog tokena. Kod je pisan u *pythonu* te je ideja da taj kod bude dijeljen između nekoliko budućih programskih kodova kojim će se dohvaćati određeni podaci. Zbog toga je odabrano ime datoteke *common\_functions.py* što zapravo predstavlja dijeljene funkcionalnosti.

Slika na kojoj se prikazuje tekst, Font, snimka zaslona

Opis je automatski generiran

Prethodni isječak koda prikacuje učitavanje .env datoteke i spremanje *client ID* i *client secret* vjerojadajnica u dvije varijable. Sljedeće što želimo je dobiti pristupni token pomoću te dvije varijable.

Definiramo metodu *get\_token*:

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, softver

Opis je automatski generiran

Prethodni isječak koda prikazuje dohvaćanje pristupnog tokena slanjem POST zahtjeva na Spotify API. Slični primjeri dohvaćanja tokena vidljivi su pretraživanjem dokumentacije Spotify API-ja.

Pristupni token niz je koji sadrži vjerodajnice i dopuštenja koja se mogu koristiti za pristup određenom resursu (npr. izvođačima, albumima ili pjesmama) ili korisničkim podacima (npr. vašem profilu ili vašim popisima za reprodukciju) [15]. Korištenjem tokena jedan cURL zahtjev za dohvaćanjem zapisa GET metodom bi izgledao nešto poput:

curl --request GET \

'https://api.spotify.com/v1/tracks/2TpxZ7JUBn3uw46aR7qd6V' \

--header "Authorization: Bearer NgCXRK...MzYjw"

Ako se pozove *get\_token* metoda s ispravnim vjerdajnicama dobije se pristupni token koji vrijedi 60 minuta. Pri isteku 60 minuta dobije se poruka o nevaljaloj autorizaciji. Tada je potrebno ponovno zatražiti token.

***PRIMJER JEDNOG ZAHTJEVA U POSTMANU? OPIS JEDNOG ODGOVORA?***

[12] možda prvo pričat što je REST api? <https://www.newline.co/courses/build-a-spotify-connected-app/okay-but-what-is-a-rest-api>

[13] <https://engineering.atspotify.com/2015/03/understanding-spotify-web-api/>

[14] <https://towardsdatascience.com/extracting-song-data-from-the-spotify-api-using-python-b1e79388d50>

[15] <https://developer.spotify.com/documentation/web-api/concepts/authorization>