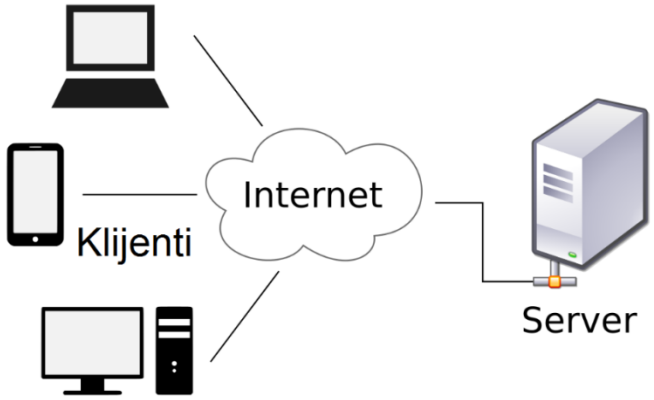
Serveri

Tokom vašeg rada kao front-end programeri ćete se susreti sa različitim serverima koji pružaju različite funkcije. U ovom segmentu ćemo da se pozabavimo analizom servera i uslugama koje oni pružaju, kako bismo vas spremili za rad van jednostavnih Node servera koje koristimo na kursevima.

Ekosistem veb-baziranih sistema

Za početak ćemo se osvrnuti na arhitekturu modernih klijen-server sistema, koja je ilustrovana na slici 1.



Slika 1 Arhitektura klijent-server veb-baziranih sistema

Korisnik sistema želi da pristupi funkcijama i podacima koje nudi određen sistem. Da bismo ovo ilustrovali pogledaćemo nekoliko veb-baziranih sistema koji su u širokoj upotrebi:

- *GMail* – Korisnik ovog sistema želi da pristupi svojim imejlovima i koristi ovu aplikaciju da šalje imejlove;
- *Facebook* – Korisnik ovog sistema pristupa svom profilu, profilu svojih prijatelja, stranicama i grupama koje prati, koristi čet, igra igrice, i još mnogo toga;
- *IMDB* – Korisnik ovog sistema pregleda podatke o filmovima, serijama, ocenjuje, ostavlja recenzije, i još mnogo toga.

Šta se zapravo dešava u svim ovim interakcijama? Grubo gledano, proces koji se odvija se sastoji od sledećih koraka:

1. Korisnik, upotrebom veb-čitača, kontaktira određeni veb-server i traži od njega front-end aplikaciju koju ima da ponudi (klikom na link, direktnim kucanjem URL-a u adresnu liniju);
2. Front-end aplikacija se učitava kod korisnika u veb-čitaču;
3. Svi podaci i sve funkcije koje korisnik želi da koristi podrazumevaju slanje HTTP zahteva od front-end aplikacije do veb-servera, gde se nalazi back-end aplikacija;
4. Back-end aplikacija prihvata zahteve, vrši određenu poslovnu logiku i komunikaciju sa nekim skladištima podataka (bilo da su baze podataka, obične datoteke ili nešto treće) i šalje odgovor front-end aplikaciji;

5. Front-end aplikacija prihvata odgovor i menja interfejs na prikladan način da korisnika obavesti o rezultatu operacije, bilo da je to neka notifikacija, prikaz podataka, ili nešto treće (u zavisnosti od zahteva korisnika).

Back-end aplikacije mogu biti napravljene u raznim tehnologijama, koristeći Java, Python i mnoge druge programske jezike (uključujući i JavaScript, kao što smo videli na kursu). Isto kao što mi koristimo radni okvir da konstruišemo moćne front-end aplikacije i biblioteke poput Ng Bootstrap, tako se i za back-end tehnologije koriste radni okviri (npr. Spring, Django,...) i razne biblioteke da se prave moćne back-end aplikacije.

Naravno, programiranje front-end i back-end aplikacija nije isti posao, iako ima sličnosti. Osnovna logika jeste ista i koncepti koji se koriste u jednom domenu se primenjuju u drugom, ali svaka oblast ima svoj skup izazova koji je specifičan za tu celinu. Back-end programeri treba da brinu o interakciji sa bazom podataka i drugim sistemima, o skalabilnosti (https://en.wikipedia.org/wiki/Scalability) i performansama. Sa druge strane, front-end programeri treba da naprave korisnički interfejs koji je pristupačan, estetski privlačan i ima dobar izgled kako na malim ekranima mobilnih telefona, tako i na širokim ekranima monitora.

Sada kada smo se osvrnuli na okruženje u kom radimo, možemo strogo definisati granice našeg posla, kao front-end programeri. Naime, sve što se prikazuje i izvršava u veb-čitaču je ono sa čime se mi bavimo, zaključno sa komunikacijom koju uspostavljamo sa serverima koji nam dostavljaju podatke i pružaju funkcije.

Rad sa serverima

Ako bi krenuli da realizujete projekat *Skijanija* prateći sve šablone koje smo do sada koristili u prethodnim projektima, veoma brzo bi naleteli na problem. Naime, servis koji bi komunicirao sa serverom ne bi radio.

Do sada smo se uzdali u šablon gde je HTTP GET zahtev na *http://localhost:3000/api/ENTITET* vraćao objekat sa poljima:

- **results**, što je predstavljalo listu *ENTITET* objekta;
- **count**, što je sadržalo informacije o broju datih entiteta na serveru.

Spram ovog šablona smo pravili *ENTITETList* klasu koja je sadržala results i count polje i računali smo da će odgovor sa servera biti u datom obliku. To jeste bio slučaj kod vina, restorana i dokumenata, ali nije slučaj sa serverom koji koristimo u Skijaniji.

Iako smo do sada radili sa odgovorom koji sadrži **count** i **results**, treba napomenuti da ovo ne mora da bude uvek slučaj. Informacija o ukupnom broju elemenata se može dobiti na više načina i ne postoji pravilo. Pored objekta koji sadrži **count** i **results**, neka česta rešenja podrazumevaju da se informacija o ukupnom broju elemenata:

- Postavlja u zaglavlje HTTP odgovora (*header response*), te da je neophodno onda pročitati vrednost koja je smeštena u dato zaglavlje;

- Dobavlja sa potpuno odvojenim HTTP zahtevom na URL koji nudi tu informaciju (npr. http://localhost:3000/api/restaurants/count);
- Ne šalje.

Kao razvijač front-end aplikacija, vi ćete dobiti dokumentovan API servera iz kog ćete moći da izvučete informaciju kako da dođete do određenog podatka, poput ukupnog broja elemenata, te ne morate da nagađate koji pristup je korišten.

Generalno, vaš posao kao front-end programer jeste da analizirate API servera i prilagodite se njegovoj strukturi. U idealnom slučaju, API će biti dobro dokumentovan te nećete morati mnogo da istražujete. Pored toga, jedan jednostavan način kako možete makar deo APIa upoznati, jeste upotrebom veb-čitača, što ćemo mi da koristimo.

Kako bi otkrili strukturu podataka koje server za *Skijanija* projekat vraća, ukucajte svaki URL koji se pojavljuje u tekstu zadatka (gde ćete parametre URLa poput ID-a zameniti sa nekim konkretnim brojem). Datu strukturu namapirajte na vaš front-end, tako da vam radi servis za komunikaciju sa serverom.

Tipovi servera

Jedna tema koju ćemo površno preći, i koja predstavlja veoma bitnu veštinu koju treba da razvijate u budućnosti, jeste snalaženje sa različitim tipovima servera. Pod različitim tipovima servera ovde mislimo konkretno na dva tipa servera, i to su:

- Serveri lokalni za preduzeće, što predstavlja back-end sistema koji firma za koju radite proizvodi;
- Eksterni serveri, kojim upravlja drugo preduzeće.

Prvi tip je nešto što mi radimo na kursu, i predstavlja komunikaciju sa našim serverom sa kojim ste upoznati. Ovde ne bi trebalo da bude mnogo problema, pošto ćete uvek imati ljude koji su razvijali taj back-end sistem sa kojim se možete konsultovati.

Međutim, problem nastaje kod drugih tipova servera. Vaša aplikacija će možda imati potrebu da prikaže vremensku prognozu za određeni dan (kao što imamo slučaju u *Skijanija* projektu). Možda će trebati da kontaktira IMDB da bi prikazao neke filmove, ako pravite aplikaciju čija je to tema (kao što ćemo videti sledeći termin).

Mnogi servisi sa kojim ste upoznati, IMDB, YouTube, Google Maps, servis za vremensku prognozu, nude javno dostupne API-e, putem kojih možete da dobavljate podatke sa datih servisa. Kako bi se zaštitili od zloupotrebe, javno dostupni API zahteva nešto što se zove *API Key*, što predstavlja specijalan kod koji treba dostaviti sa svakim zahtevom. API Key možete dobiti registracijom na određeni servis, gde ćete periodično morati da zatražite novi API Key kada vam stari istekne.

Za potrebe *Skijanija* zadatka smo obezbedili podatke o vremenskoj prognozi za skijališta, ali dobar napredni zadatak iz oblasti čitanja dokumentacije i snalaženja sa eksternim servisima jeste da iskoristite *Weather API* (https://openweathermap.org/api) da dobavite ove podatke sa njihovog servisa.