API & REST Fundamentals

Šta je API?

API (Application Programming Interface) je posrednik između dva softverska sistema koji omogućava njihovu komunikaciju. Najbolja analogija za razumevanje API-ja jeste konobar u restoranu. Konobar prima porudžbinu (zahtev) od gosta, prenosi je kuhinji (sistemu) i vraća gotovo jelo (odgovor) gostu. Gost ne mora da zna kako kuhinja funkcioniše – dovoljno je da komunicira preko konobara.

Zašto su API-ji važni?

API-ji predstavljaju "lepak" savremenog softverskog sveta.

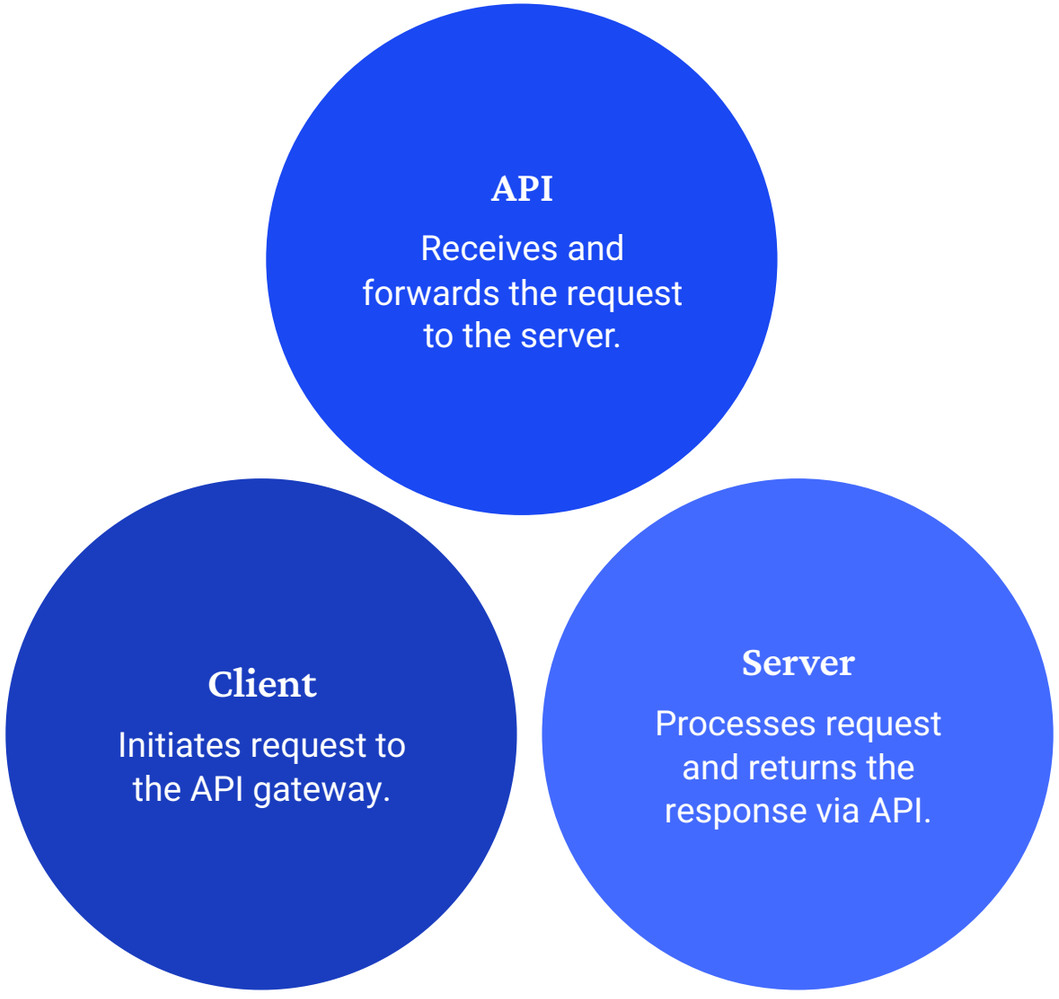
Oni omogućavaju:

- Povezivanje različitih aplikacija i servisa
- Ponovnu upotrebu postojećih funkcionalnosti
- Razdvajanje frontend-a i backend-a
- Brži razvoj softvera
- Skalabilnost sistema

Svakodnevni primer upotrebe API-ja jeste aplikacija za vremensku prognozu na telefonu – ona ne meri temperaturu sama, već "pita" API meteorološkog servisa i prikazuje dobijene podatke.

Kako API funkcioniše?

Osnovni model komunikacije funkcioniše po principu **zahtev-odgovor**. Klijent (aplikacija) šalje **ZAHTEV (request)** ka API-ju koji služi kao posrednik. API prosleđuje zahtev serveru (backendu) koji obrađuje podatke i vraća **ODGOVOR (response)** putem API-ja nazad klijentu.



Ključni elementi ovog modela su **klijent** kao aplikacija koja traži podatke, **API** kao definisana "recepcija" kroz koju se komunicira i **server** kao backend koji obrađuje zahtev i vraća podatke.

Vrste API-ja prema dostupnosti

Javni / Otvoreni API-ji

Dostupni su svim programerima uz minimalna ograničenja. Često su besplatni uz ograničenje broja poziva.

Primeri: Google Maps API, Twitter API, OpenWeather API

Interni / Privatni API-ji

Koriste se samo unutar jedne organizacije za povezivanje internih sistema poput CRM-a i računovodstva.

Partnerski API-ji

Dele se samo sa specifičnim poslovnim partnerima.

Kompozitni API-ji

Kombinuju podatke iz više API-ja u jednom pozivu.

Otvoreni API i otvoreno računarstvo

Otvoreni API-ji prate istu filozofiju kao i **open source softver**.

Transparentnost

Javno dostupna dokumentacija

Sloboda korišćenja

Svako može da integriše API u svoje aplikacije

Interoperabilnost

Povezivanje različitih sistema

Zajednica

Programeri koji grade dodatne alate i biblioteke oko API-ja

> Otvoreni API predstavlja spoj **otvorenih standarda** i **otvorenog pristupa**.

Šta je REST?

REST (Representational State Transfer) **nije protokol**, već **arhitekturni stil** za dizajn mrežnih aplikacija. Definisao ga je **Roy Fielding 2000. godine** u svojoj doktorskoj disertaciji.

REST je danas dominantan jer:

Koristi HTTP

Standardni protokol

Jednostavan

Za razumevanje i implementaciju

Nezavisan

Od platforme i programskog jezika

Skalabilan

I pouzdan

Ključni principi REST-a

1 Klijent-server arhitektura

Razdvaja korisnički interfejs (frontend) od servera koji čuva podatke (backend).

2 Bezstanje (Stateless)

Server ne pamti stanje klijenta između zahteva. Svaki HTTP zahtev mora da sadrži sve potrebne informacije poput autentifikacije.

3 Keširanje (Cacheability)

Odgovori moraju da definišu da li mogu da se keširaju, čime se smanjuje opterećenje servera i ubrzava rad.

Jedinstveni interfejs (Uniform Interface)

Četvrti i najvažniji princip jeste **jedinstveni interfejs** koji čini srž REST-a.

Resursi (Resources)

Predstavljaju osnovni koncept – **sve je resurs**: korisnik, proizvod, porudžbina. Svaki resurs ima svoj URI (Uniform Resource Identifier).

Primeri URI-ja

`/korisnici` – lista svih korisnika

`/korisnici/123` – konkretan korisnik

Resursi se razmenjuju u standardnim formatima, najčešće JSON, a ponekad XML.

HTTP metode – "glagoli" REST API-ja

HTTP metode predstavljaju standardizovane operacije nad resursima.

Metoda	Operacija	Opis	Primer
GET	Dohvata resurs	Ne menja stanje sistema	`GET /proizvodi`
POST	Kreira novi resurs	Dodaje novi zapis	`POST /korisnici`
PUT	Zamenjuje ceo resurs	Potpuna zamena postojećeg	`PUT /korisnici/123`
PATCH	Delimično ažurira	Menja samo deo resursa	`PATCH /korisnici/123`
DELETE	Briše resurs	Uklanja zapis	`DELETE /korisnici/123`

Važno je razumeti da su HTTP metode standardizovane i nezavisne od toga šta resurs predstavlja.

HTTP statusni kodovi

Server u odgovoru uvek vraća statusni kod koji označava ishod zahteva.

2xx – Uspeh

- **200 OK** – Zahtev uspešno obrađen
- **201 Created** – Resurs uspešno kreiran putem POST metode

4xx – Greška klijenta

- **400 Bad Request** – Loš zahtev
- **401 Unauthorized** – Klijent nije autentifikovan
- **404 Not Found** – Resurs ne postoji

5xx – Greška servera

- **500 Internal Server Error** – Nešto pođe po zlu na serveru

JSON – jezik razmene podataka

JSON (JavaScript Object Notation) je najčešći format za razmenu podataka u REST API-jima. Primer JSON odgovora izgleda ovako:

```
{
  "id": 123,
  "ime": "Marko Marković",
  "email": "marko@example.com",
  "aktivan": true,
  "godine": 25
}
```

JSON je omiljen jer je **čitljiv ljudima i mašinama**, nezavisan je od programskog jezika, lagan je u poređenju sa XML-om i podržan je u svim modernim programskim jezicima.

Povezivanje sa protokolima i standardima

REST API je direktna primena HTTP protokola, što je ključno za razumevanje.

HTTP metode

Definišu način komunikacije

URI

Obezbeđuje adresiranje resursa

Statusni kodovi

Standardizovani odgovori

JSON/XML

Standardizovani formati podataka

> REST ne izmišlja nove načine komunikacije – on koristi postojeće HTTP standarde na **dosledan način**.

API Design & Best Practices

Dobre i loše prakse u dizajnu API-ja

Dobar REST API (RESTful) koristi pravilno imenovane rute. Primeri dobrih praksi uključuju `GET /proizvodi` za listu svih proizvoda, `GET /proizvodi/123` za pojedinačni proizvod, `POST /porudzbine` za kreiranje nove porudžbine i `GET /korisnici/456/porudzbine` za prikaz porudžbina konkretnog korisnika.

✅ Dobar API (RESTful)

- `GET /proizvodi`
- `GET /proizvodi/123`
- `POST /porudzbine`
- `GET /korisnici/456/porudzbine`

❌ Loš API (nije RESTful)

- Glagoli u URI-ju: `GET /getProduct` ili `POST /createOrder`
- Nestandardne forme: `GET /productinfo?id=123`
- Pogrešne HTTP metode: `POST /updateProduct/123` umesto PUT metode

Osnovno pravilo: URI treba da imenuje resurse (imenice), a HTTP metoda definiše akciju (glagol).

Dokumentacija API-ja

Dokumentacija je ključna za upotrebljivost API-ja jer programerima objašnjava kako da ga koriste.

Dobra dokumentacija sadrži:

Endpoint-i

Spisak svih endpoint-a (ruta)

HTTP metode

Dozvoljene HTTP metode za svaki endpoint

Parametri

Očekivane parametre i formate podataka

Primeri

Primer zahteva i odgovora

Statusni kodovi

Statusne kodove i značenje grešaka

Autentifikacija

Način autentifikacije

OpenAPI Specification (bivši Swagger)

OpenAPI je jezik za opisivanje API-ja koji je razumljiv i ljudima i mašinama.

Šta definiše?

- **Dostupne rute (endpoint-e)**
- **Dozvoljene HTTP metode**
- **Očekivane parametre i JSON šemu**
- **Autentifikaciju**

Zašto je važan?

- Omogućava **automatsko generisanje dokumentacije**
- Omogućava **automatsko generisanje klijentskog koda** za različite programske jezike
- **Testiranje API-ja**
- Služi kao **"ugovor" između klijenta i servera**

Primer alata koji koriste OpenAPI standard su **Swagger UI** i **Swagger Editor**.

Alati za rad sa API-jima

Za testiranje i razvoj API-ja koriste se različiti alati.

Alat	Tip	Opis
Curl	Komandnolinijski	Alat za slanje HTTP zahteva, npr. `curl https://api.example.com/korisnici/123`
Postman	Grafički (freemium)	Grafički alat za testiranje API-ja
Insomnia	Open source	Open source alternativa Postman-u
Swagger UI	Dokumentacija	Automatski generisana interaktivna dokumentacija
Hoppscotch	Open source / Web	Open source alat za testiranje API-ja direktno u pregledaču

Zaštita API-ja

API-ji se štite od zloupotreba kroz **autentifikaciju i autorizaciju**.

API ključ

Jednostavan token za identifikaciju aplikacije.

OAuth 2.0

Standard za delegiranje pristupa, poznat po funkcionalnosti "Prijavi se preko Google-a".

JWT (JSON Web Token)

Token koji sadrži informacije o korisniku.

Rate Limiting

Ograničava broj zahteva po vremenskom intervalu za zaštitu od preopterećenja.

API Gateway

Sloj ispred API-ja koji upravlja saobraćajem.

Verzionisanje API-ja

Osnovni princip razvoja API-ja jeste da se **nikada ne menja API na način koji ruši postojeće klijente**.

Rešenje je verzionisanje, na primer:

```
https://api.example.com/v1/korisnici
https://api.example.com/v2/korisnici
```

Stari klijenti i dalje koriste verziju **v1**, dok novi mogu da pređu na **v2**.

Načini implementacije

1 **U URI-ju**

`/v1/`, `/v2/`

2 **U HTTP header-u**

`Accept: application/vnd.example.v1+json`

3 **Query parametar**

`?version=1`

API i ekosistemi

Kada su API-ji otvoreni, nastaju ekosistemi.

Google Maps API

Koriste hiljade aplikacija

Twitter API

Izgrađen čitav ekosistem klijenata i alata

Stripe API

Platforma za online plaćanja

Ovo je direktna paralela sa open source ekosistemima – kao što open source alati grade ekosisteme, tako i otvoreni API-ji omogućavaju gradnju dodatnih slojeva i integracija od strane zajednice.

Praktični primer – API za blog

Zamislimo API za blog sistem.

Resurs "članci"

Metoda	Ruta	Opis
GET	`/clanci`	Lista svih članaka
POST	`/clanci`	Kreiranje novog članka
GET	`/clanci/42`	Pojedinačni članak
PUT	`/clanci/42`	Zamena celog članka
PATCH	`/clanci/42`	Delimično ažuriranje
DELETE	`/clanci/42`	Brisanje članka

Resurs "komentari"

Metoda	Ruta	Opis
GET	`/clanci/42/komentari`	Prikaz komentara određenog članka
POST	`/clanci/42/komentari`	Dodavanje novog komentara

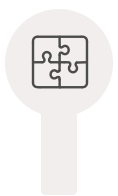
API i obrazovanje

Razumevanje API-ja izuzetno je važno za studente.

Učenjem o API-jima studenti:

Razumevanje komunikacije

Razumevaju kako savremeni softver komunicira



Korišćenje gotovih servisa

Mogu da koriste gotove servise umesto da sve prave od nule

Industrijske veštine

Razvijaju veštine tražene u industriji poput integracije sistema

Cloud arhitekture

Stiču osnovu za rad sa mikroservisima i cloud arhitekturama

Tehnička dokumentacija

Uče kako da čitaju i pišu tehničku dokumentaciju

> **API nije samo tehnički koncept – to je način razmišljanja o povezivanju sistema.**

Ograničenja i odgovornost

Otvoreni API-ji donose brojne prednosti – omogućavaju inovacije i integracije, smanjuju vreme razvoja i grade ekosisteme. Međutim, oni zahtevaju i odgovorno korišćenje.

⚠️ Ne preopterećuj server

⏱️ Poštuj limite

Rate limiting postoji s razlogom

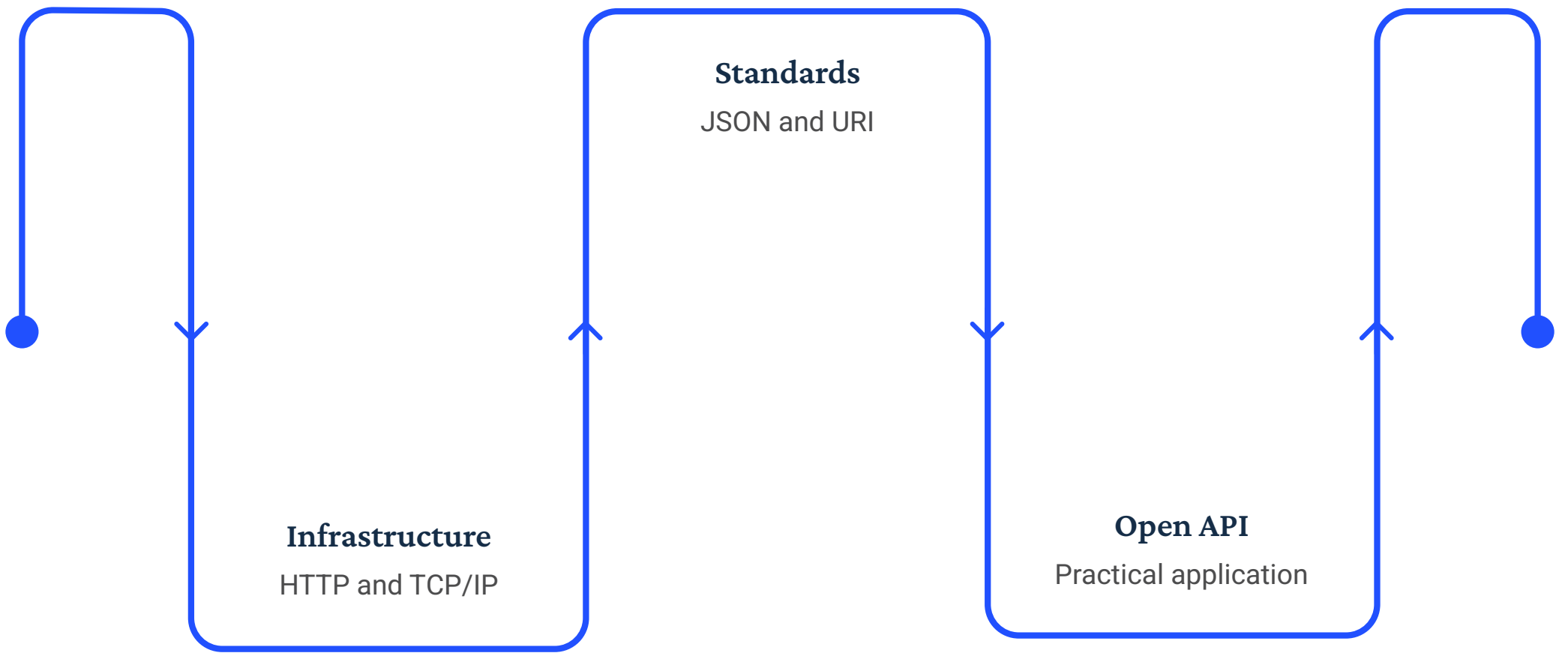
🔒 Zaštiti podatke

Ne šalji osetljive podatke kroz javne API-je

📋 Razumi uslove

API može imati svoja pravila

Otvoreni API-ji kao nastavak otvorenih protokola i standarda



Protokoli poput HTTP i TCP/IP čine infrastrukturu, standardi poput JSON i URI definišu jezik komunikacije, a otvoreni API donosi primenu u praksi.

Povezivanje sistema

Deljenje funkcionalnosti

Gradnja na tuđim rešenjima

Interoperabilnost u praksi

> **API nije cilj sam po sebi, već sredstvo za izgradnju povezanog digitalnog sveta.**