

Agenda

- 1 Flask + Jinja2 web aplikacija
- 2 Višeslojna arhitektura
- 3 Tok podataka
- 4 Primer arhitekture
- 5 React + Flask
- 6 Razdvojena višeslojna arhitektura
- 7 Tok podataka
- 8 Primer arhitekture

Flask + Jinja2 web aplikacija

Tehnološki stek

Web aplikacija zasnovana na Python Flask **mikro-okviru** i Jinja2 template-ing za generisanje HTML-a.

Arhitektura sistema

Flask **upravlja** HTTP rutama, kontrolerima i integracijom ekstenzija; Jinja2 **renderuje** dinamične stranice.
Tight-coupled (**backend i fronted u istom projektu!**)

Format odgovora

HTML stranice.

Gde se renderuje?

Na strani servera - *server-side rendered (SSR)*

Višeslojna arhitektura

Klijent (Browser)

Šalje zahteve, prima HTML odgovore i prikazuje HTML/CSS/JS.

Flask (Controller)

Obrađuje zahtjeve, izvršava logiku, komunicira sa bazom podataka...

Jinja2 (View)

Pisanje HTML dokumenata sa python-like sintaksom



MySQL / MongoDB (Data Storage)

Skladište za podatke



ORM / ODM (Data Access)

Veza između Flask-a i baze (npr. SQLAlchemy).

Tok podataka

1

Korisnik otvara URL; browser šalje zahtev ka Flask ruti.

2

Flask implementira poslovnu logiku i pristupa MySQL-u preko OR-a.

3

Podaci iz baze se vraćaju kao Python objekat.

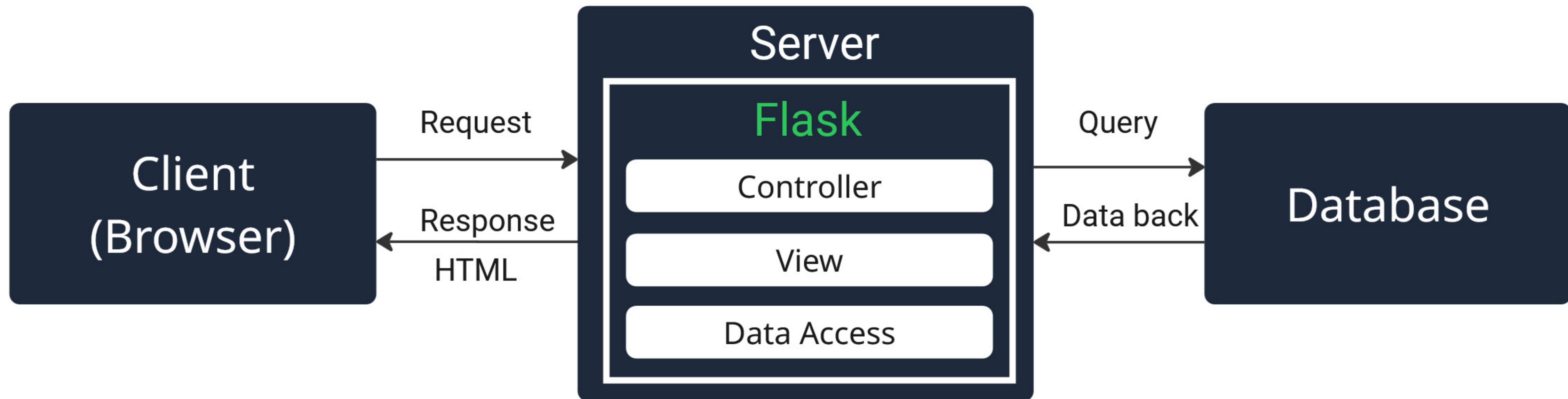
4

Jinja2 renderuje template sa kontekstom i generiše HTML.

5

Server šalje odgovor; browser prikazuje stranicu.

Arhitektura



Development server (Werkzeug)
or

Deployment server (WSGI)

React + Flask

Tehnološki stek

Web aplikacija zasnovana na Python Flask **mikro-okviru** za REST API i React biblioteci za izgradnju korisničkog interfejsa.

Arhitektura sistema

Flask **upravlja** HTTP API rutama, kontrolerima i poslovnom logikom; React **renderuje** korisnički interfejs na klijentu. Decoupled arhitektura (**frontend i backed u odvojenim projektima**).

Format odgovora

JSON podaci (API odgovori) koje React koristi za prikaz.

Gde se renderuje?

Na strani klijenta - *client-side rendered (CSR)*

Višeslojna arhitektura



Klijent (Presentation layer: Browser + React)

Šalje zahteve, prima JSON podatke i renderuje UI koristeći React komponente.



Flask (Controller / API Layer)

Obrađuje zahtjeve, izvršava logiku, komunicira sa bazom podataka...



MySQL / MongoDB (Data Storage)

Skladište za podatke



ORM / ODM (Data Access)

Veza između Flask-a i baze (npr. SQLAlchemy).

Tok podataka

1

Korisnik otvara URL u React aplikaciji; browser učitava Reacte bundle; šalje se zahtev ka Flask API.

2

Flask obrađuje zahtev, izvršava poslovnu logiku i pristupa bazi (MySQL/MongoDB) preko ORM/ODM sloja.

3

Baza vraća podatke; Flask ih pretvara u JSON format.

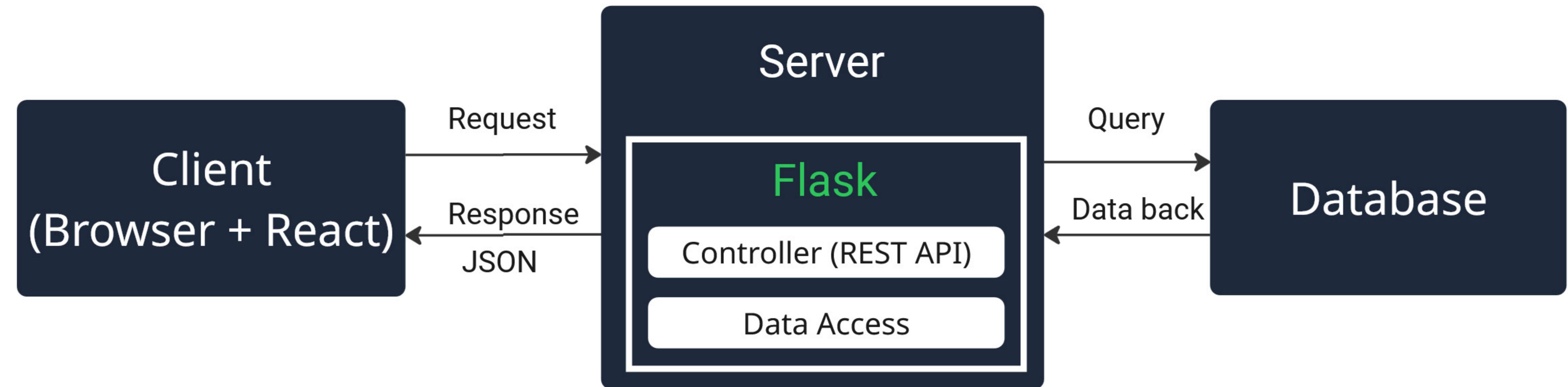
4

Flask šalje JSON odgovor React aplikaciji.

5

React ažurira stanje (state) i renderuje UI koristeći podatke iz JSON odgovora.

Arhitektura



Development server (Werkzeug)

or

Deployment server (WSGI)

Agenda

- 1 SOP (Same Origin Policy)
- 2 CORS (Cross-Origin Resource Sharing)
- 3 SQL baze podataka
- 4 NoSQL baze podataka
- 5 MongoDB
- 6 Redis
- 7 Rest API vs WebSocket

SOP (*Same Origin Policy*)

Šta je?

Bezbednosno pravilo u browseru koje sprečava da skripta koja se izvršava iz jednog origina direktno pristupa resursima sa drugog origina.

ORIGIN = PROTOKOL + DOMEN + PORT

Scenario

Frontend: <https://moja-stranica.com>

API: <https://moj-api.com>

1. Korisnik otvara <https://moja-stranica.com> - browser **učitava** HTML, CSS, JS
2. JS na toj stranici kreira zahtev ka <https://moj-api.com>
3. Browser formira **HTTP request** u koji dodaje zaglavje **Origin: https://moja-stranica.com**
4. Server prima zahtev i šalje odgovor npr. 200 OK
5. Browser blokira čitanje odgovora u JavaScriptu jer origin nije isti i server nije dao dozvolu (nije dodao zaglavje **Access-Control-Allow-Origin: https://moja-stranica.com**)

CORS (Cross-Origin Resource Sharing)

Šta je?

CORS je mehanizam na serverskoj strani koji omogućava kontrolisanu relaksaciju SOP-a.

Šta nije?

NIJE browser funkcija. **NIJE** front-end funkcija.

Prethodni scenario + CORS

- 4.1. Server proverava da li sme da dozvoli ovom originu pristup (na osnovu konfiguracije)
- 4.2. Ako sme -> server u odgovor dodaje zaglavljje **Access-Control-Allow-Origin: <https://moja-stranica.com>**
5. Browser sada dozvoljava da se pročita odgovor jer je uključeno zaglavljje sa dozvolom

Napomena: *Koraci se odnose na Simple CORS zahtev, u slučaju Complex CORS prvo ide preflight (OPTIONS) gde se proverava dozvola za glavni zahtev koji treba da se pošalje i tek nakon toga ide glavni zahtev!*

SQL baze podataka

1

Relacione baze podataka, podaci se čuvaju u **tabelama** (redovi i kolone), relacije između tabela se uspostavljaju pomoću **primarnih** i **stranih** ključeva

2

Postoji striktna fiksna šema. Podaci moraju da odgovaraju unapred definisanoj šemi.

3

Koriste SQL jezik za upite, manipulaciju i definiciju podataka.

4

Dobar za struktuirane podatke, kompleksne JOIN upite, ACID svojstva...

5

Primeri: MySQL, PostgreSQL, Microsoft SQL Server...

NoSQL baze podataka

1

Nerelacione baze podataka gde se podaci čuvaju u razlicitim formatima i modelima.

2

Dinamicna i fleksibilna šema. Npr. dokumenti u kolekciji mogu imati razlicita polja.

3

Ne postoji standardizovani jezik za upite.

4

Idealan kada se šeme često menjaju, velike kolicine nestrukturiranih podataka...

5

Primeri: MongoDB (dokument), Redis (Key-value), Neo4j (graf)...

MongoDB



NoSQL dokument-baza podataka

Podaci se čuvaju kao dokument u JSON/ BSON formatu.



Database

Sadrži više **kolekcija**.



Collection

Sadrži više **dokumenata**



Document

Jedan zapis u JSON formatu

*Dokument može imati jedan skup **polja**, dok drugi dokument u istoj kolekciji može izgledati **potpuno drugačije**.*

Redis (*Remote Dictionary Server*)



NoSQL key value baza podataka

In-memory (podatke čuva u RAM-u umesto na disku) -> izuzetno brza



Key-Value Store

Podaci se čuvaju kao parovi **ključ - vrednost**



Podržava više tipova podataka

String, List, Set, Sorted Set, Hash, Stream, Bitmap



Komande

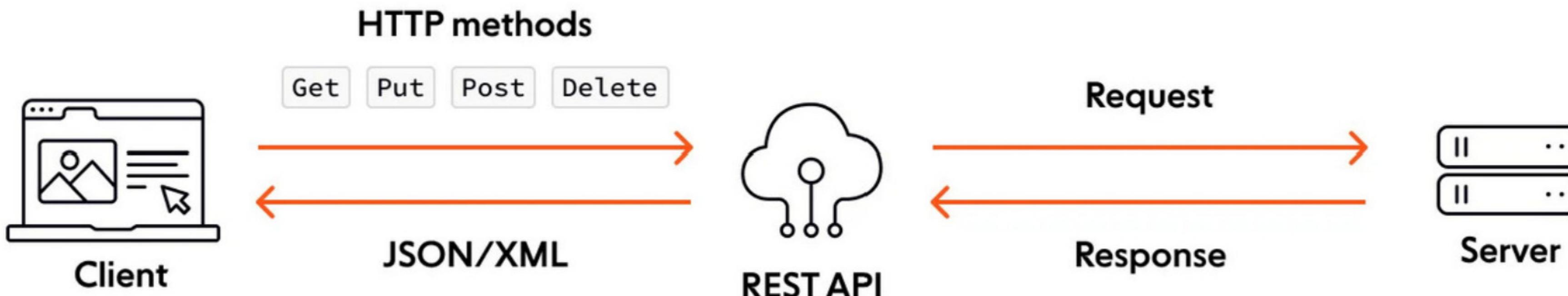
GET, SET, INCR, EXPIRE, EXISTS... (<https://redis.io/docs/latest/commands/>)

Koristi se za keširanje, čuvanje sesija, rate limiting, queue sistemi...

Rest API vs WebSocket

	REST API	WebSocket
ŠTA JE?	Način dizajniranja API	Protokol koji omogućava komunikaciju
TIP KOMUNIKACIJE	Request-response jednosmerno	Dvosmerna (full-duplex) veza bazirana na događajima
PROTOKOL	HTTP	WebSocket
VEZA	Nova veza za svaki zahtev	Ista veza
STANJE	Stateless (server ne pamti stanje između zahteva)	Stateful
UPOTREBA	CRUD operacije, API pozivi	Real-time aplikacije, chat, notifikacije, online igre

Rest API



WebSocket

