Informacioni sistemi 2

GraphQL

Uvod

- **Cilj: Uvođenje GraphQL-a kao alternative za REST**
- **Teme koje će se pokriti:**
  - Osnove Node.js-a i MongoDB-a
  - Uvod u GraphQL
  - Kako GraphQL funkcioniše sa Node.js-om i MongoDB-om

Node.js

- **Node.js** je okruženje za izvršavanje JavaScript-a na serverskoj strani, koje omogućava korišćenje JavaScript-a za razvoj backend aplikacija.
- Ključne karakteristike
  - **Jednonitno izvršavanje**: Node.js koristi jednonitnu arhitekturu, ali podržava **asinhroni pristup** i **neblokirajuće operacije.**
  - **Event-Driven**: Node.js koristi događajima vođenu arhitekturu koja omogućava da aplikacija odgovara na događaje, kao što su korisnički zahtevi ili dolazni podaci, bez zastoja.
  - **Veliki ekosistem**: Pomoću **npm** (Node Package Manager), omogućen pristup hiljadama paketa i modula koji ubrzavaju razvoj aplikacija.

Node.js

- Iako je Node.js okruženje za izvršavanje JavaScript-a, često se umesto JavaScripta koristi TypeScript
- Zašto TypeScript za Node.js?
  - **Statistička tipizacija**
    - Sprečava greške otkrivanjem problema tokom razvoja, pre izvršavanja koda.
  - **Veća pouzdanost i održivost koda**
    - Tipovi olakšavaju rad sa velikim aplikacijama i čine kod čitljivijim i lakšim za održavanje.
  - **Bolja podrška u IDE-ovima**
    - Napredne mogućnosti autokompletiranja i detekcije grešaka, ubrzavaju razvoj i smanjuju greške.

Node.js

- Kreiranje I inicijalizacija aplikacije
  1. Kreiranje aplikacije: npm init –y
  2. Instaliranje svih paketa koji su nam potrebni za rad u typescript:
     npm install -D typescript ts-node @types/node @types/express @types/graphql nodemon
  3. Kreirati fajl tsconfig.json I kopirati :
  4. Kreirati folder src I u njemu fajl index.ts
  5. U package.json, u delu ”script” dodati:

     "start": "ts-node src/index.ts"

```
{
"compilerOptions": {
"target": "ES2019",
"module": "commonjs",
"strict": true,
"esModuleInterop": true,
"resolveJsonModule": true,
"outDir": "./dist"
},
"include": ["src"],
"exclude": ["node_modules"]
}
```

- Nakon ovoga, aplikaciju pokrećemo komandom npm start

MongoDB

- MongoDB je dokumentno orijentisana baza podataka zasnovana na NoSQL pristupu.

Karakteristika	MongoDB (NoSQL)	Relaciona baza
Model podataka	Dokumentno orijentisan (BSON format slican JSON formatu)	Relacioni model (Tabele sa kolonama i redovima)
Struktura podataka	Fleksibilna, nema unapred definisanu šemu	Fiksna šema, unapred definisane tabele i kolone
Veza izmedju podataka	Podaci često ugrađeni u dokumentima	Koristi eksplicitne relacije (strani ključevi)
Upiti	Upiti preko JSON-slične sintakse, koristi agregacione funkcije	SQL jezik za standardne upite i manipulaciju podacima

MongoDB

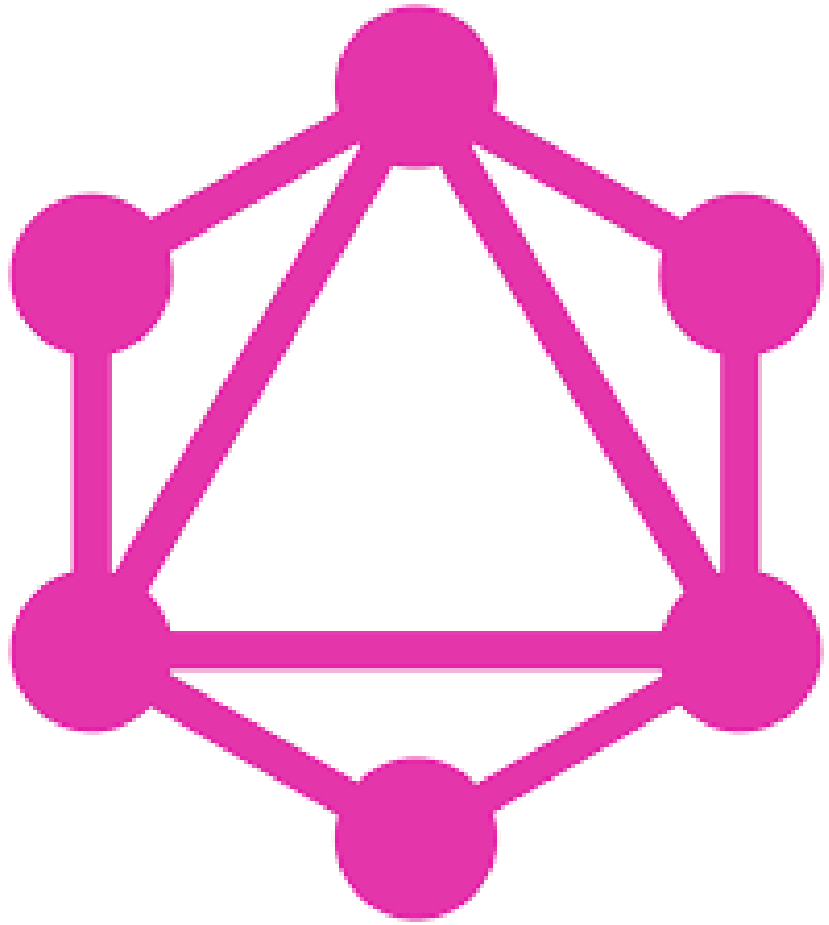
- Izgled jednog dokumenta


```
{
  "user": "User1",
  "status": "denied",
  "items": [
    {
      "article": "Cheesecake",
      "amount": 3,
      "price": 1500
    }
  ]
}
```


- Za integraciju sa Node.js
  - Paket mongoose
  - Elegantna ODM biblioteka za integraciju MongoDB u JavaScript

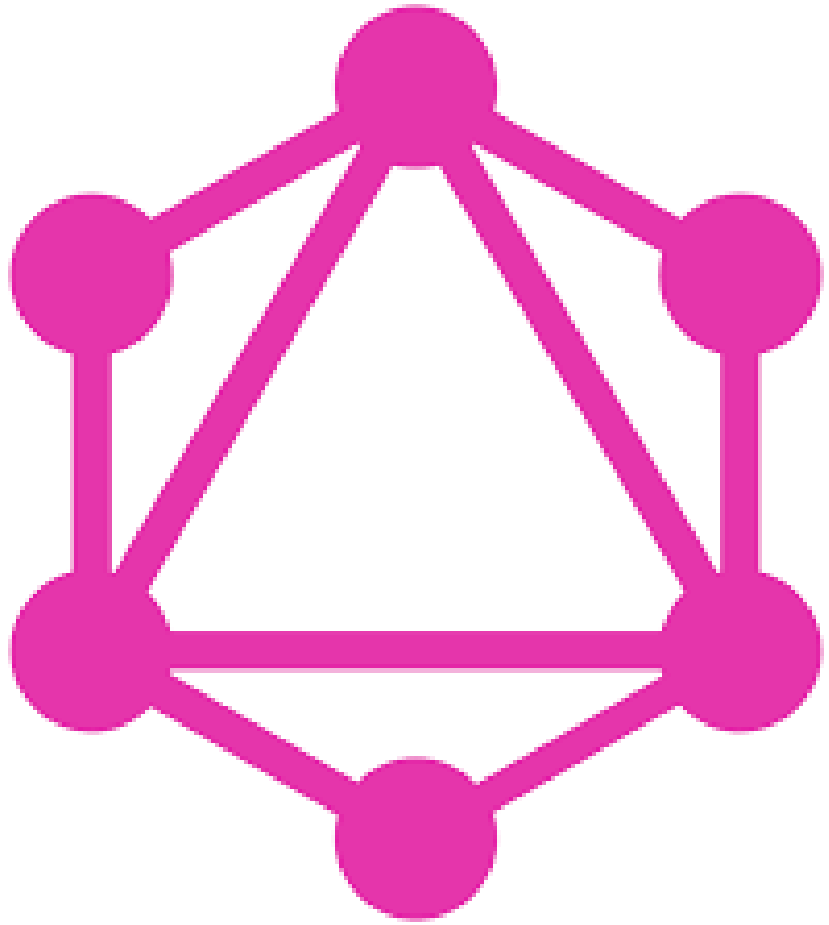
GraphQL

- Šta je GraphQL?
  - API query jezik razvijen od strane Facebook-a.
  - Nezavistan od programskih jezika i specifične baze podataka
  - Koriste se podaci koje mi želimo
- Kako GraphQL radi?
  - Klijent zahteva specifične podatke u jednom API pozivu
  - Klijentu se vraćaju isključivo podaci koje je tražio

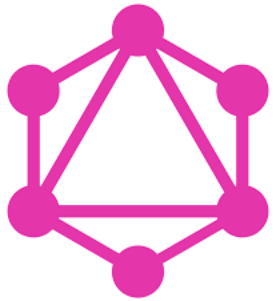


GraphQL

- Ključne karakteristike
  - Precizni upiti - omogućava klijentima da zahteva samo potrebne podatke
  - Fleksibilnost - prilagođava se različitim potrebama klijenata.
  - Upravljanje relacijama - omogućava lako povezivanje podataka iz različitih entiteta.
  - Single Endpoint - sve upite obavlja preko jednog API endpoint-a.

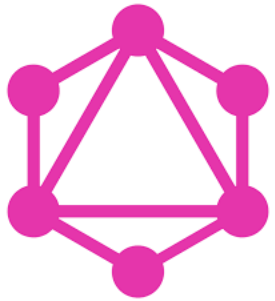


GraphQL



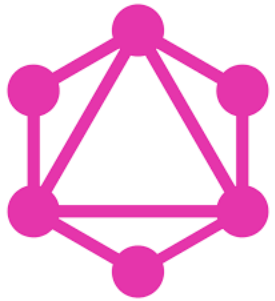
Karakteristika	REST	GraphQL
Struktura	Više endpoint-a za različite entitete	Jedan endpoint za sve upite
Prenos podataka	Vraća sve podatke entiteta, bez mogućnosti filtriranja	Vraća samo tražene podatke, prema specifikaciji klijenta
Optimizacija poziva	Veći broj poziva za različite resurse	Svi podaci se mogu preuzeti u jednom pozivu
Učinkovitost za klijente	Moguće je preuzimanje nepotrebnih podataka	Efikasno, jer klijent određuje koji podaci su potrebni
Preglednost u razvoju	Lakše započeti sa osnovnim CRUD operacijama	Može zahtevati više planiranja, ali fleksibilniji u dužem periodu

GraphQL



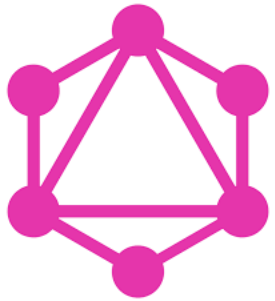
- Osnovni pojmovi:
  - **Šema** (Schema) - Definicija strukture podataka koje API pruža, sadrži sve tipove i njihove relacije.
  - **Upit** (Query) - Osnovna vrsta upita u GraphQL-u, omogućava klijentima da precizno definišu koje podatke žele da dobiju. Klijenti mogu selektivno tražiti polja, što smanjuje količinu nepotrebnih podataka.
  - **Mutacija** (Mutation) - Upit za manipulaciju podacima. Dok Query služi za čitanje podataka, Mutation služi za kreiranje, ažuriranje i brisanje podataka.

GraphQL



- Biblioteka Express GraphQL
  - npm install express express-graphql graphql
- Glavne karakteristike:
  - **Jednostavna Konfiguracija**: Postavljanje GraphQL endpoint-a u Express aplikaciji zahteva minimalnu konfiguraciju.
  - **Integracija sa Express-om**: Radi kao middleware za Express, što omogućava jednostavno ubacivanje GraphQL endpoint-a u postojeću Express aplikaciju.
  - **GraphiQL Podrška**: Omogućava pokretanje **GraphiQL** korisničkog interfejsa u pretraživaču, gde možete testirati i istraživati svoje GraphQL upite. GraphiQL je vrlo korisno sredstvo za razvoj.
  - **Prilagođavanje i Ekstenzibilnost**: Omogućava lako prilagođavanje rezolucije upita, dodavanje autentifikacije, logovanje i druge middleware funkcionalnosti.

GraphQL

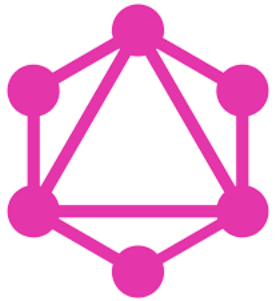


- Primer mutacije

```
mutation{
  addPerson(name:"User3",age:25){
    pk
  }
}
```

```
{
  "data": {
    "addPerson": {
      "pk": 3
    }
  }
}
```

GraphQL

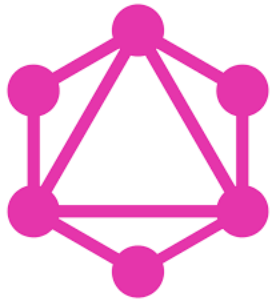


- Primer upita (bez parametara)

```
{
  persons{
    name
    age
  }
}
```


```
{
  "data": {
    "persons": [
      {
        "name": "User1",
        "age": 23
      },
      {
        "name": "User2",
        "age": 25
      },
      {
        "name": "User3",
        "age": 25
      }
    ]
  }
}
```

GraphQL



- Primer upita (sa parametrom)

```
{
  person(pk:1){
    name
  }
}
```

```
{
  "data": {
    "person": {
      "name": "User1"
    }
  }
}
```