



# Uvod u GraphQL

## s primjerima korištenja u različitim tehnologijama

Nenad Crnko  
Zagreb, 7.9.2018.



# Sadržaj

Uvod (što je GraphQL i kako je nastao)

Usporedba s drugim sličnim tehnologijama

Primjeri korištenja u programskom jeziku  
JavaScript

Primjeri korištenja na relacijskoj bazi podataka  
MySQL pomoću programskog jezika PHP

Primjeri korištenja na dokumentno orijentiranoj  
bazi podataka MongoDB



# Uvod (Što je GraphQL i kako je nastao)



# Što je GraphQL

GraphQL predstavlja posebnu vrstu jezika upita pomoću koje aplikacija na strani klijenta na fleksibilan, ali istovremeno i optimiziran način, postavlja upite serveru, a server vraća rezultate obrade klijentu. Upiti se ne postavljaju izravno na bazu podataka nego na API podsustav.

U svojim mobilnim aplikacijama Facebook koristi ovu tehnologiju od 2012. godine.

GraphQL postaje javno dostupan 2015. godine



# Ključne karakteristike

GraphQL upit vraća uvijek točno tražene i predvidljive podatke te ništa više ili manje od toga. Pri tome aplikacija na strani klijenta upravlja time kakve će podatke dobiti, a ne server, zbog čega je brza i stabilna u radu.

Jednim GraphQL upitom može se dobiti velika količina podataka zajedno s vezama između podataka (prednost u odnosu na višestruke REST API pozive). Zato korištenje ove tehnologije može biti vrlo djelotvorno i na sporim (mobilnim) vezama.

GraphQL upiti organizirani su prema objektima i poljima kojima se pristupa preko jedinstvene adrese.



# Ključne karakteristike

Razvoj novih GraphQL upita je jednostavniji, jer ne utječe na djelovanje starih upita, koji se i dalje mogu normalno izvoditi. Također, prekid mogućnosti korištenja starijih verzija upita je jednostavnija nego kod drugih tehnologija.

Tehnologija je dobro provjerena u praksi, a nakon što je postala javno dostupna moguće je napraviti njezinu implementaciju u različitim okruženjima i programskim jezicima: Python, PHP, Java, JavaScript, C# / .NET, Go, Ruby, ...



# Osnovni primjer „Hello world”

Upit:

```
query {  
  hello  
}
```

Odgovor:

```
{  
  "data": {  
    "hello": "Hello world!"  
  }  
}
```



## Malo složeniji primjer upita

Upit na polje i objekt s navođenjem naziva operacije (obavezno samo kod većeg broja operacija, ali korisno kod traženja grešaka):

```
query HeroNameAndFriends {
  hero {
    # upit na polje
    name
    # upit na objekt
    friends {
      name
    }
  }
}
```





# Rezultati malo složenijeg primjera upita

Odgovor (iz „Star Wars” baze podataka):

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },

```



# Rezultati malo složenijeg primjera upita

```
{  
  {  
    {  
      {  
        {  
          {  
            {  
              {  
                {  
                  {  
                    {  
                      {  
                        {  
                          {  
                        }  
                      }  
                    }  
                  }  
                }  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```



# Korištenje argumenata u upitima

Zadavanje upita navođenjem argumenata:

```
query {  
  # korištenje dva argumenta od  
  # kojih je drugi enumerator  
  human(id: "1000") {  
    name  
    height(unit: FOOT)  
  }  
}
```



# Rezultati upita s argumentima

```
{  
  "data": {  
    "human": {  
      "name": "Luke Skywalker",  
      "height": 5.6430448  
    }  
  }  
}
```



# Korištenje aliasa u upitima

Rješavanje problema ponavljanja polja u upitima:

```
query {  
  empireHero: hero(episode: EMPIRE) {  
    name  
  }  
  jediHero: hero(episode: JEDI) {  
    name  
  }  
}
```



# Rezultati upita s aliasima

```
{
  "data": {
    "empireHero": {
      "name": "Luke Skywalker"
    },
    "jediHero": {
      "name": "R2-D2"
    }
  }
}
```



# Korištenje fragmenata za skupove polja

Isti fragment koristi se na više mjesta u upitu:

```
query {  
  leftComparison: hero(episode: EMPIRE) {  
    ...comparisonFields  
  }  
  rightComparison: hero(episode: JEDI) {  
    ...comparisonFields  
  }  
}  
fragment comparisonFields on Character {  
  name  
  appearsIn  
  friends {  
    name  
  }  
}
```



# Rezultati upita s korištenjem fragmenata

```
{
  "data": {
    "leftComparison": {
      "name": "Luke Skywalker",
      "appearsIn": [
        "NEWHOPE",
        "EMPIRE",
        "JEDI"
      ],
      "friends": [
        {
          "name": "Han Solo"
        }, ...
      ]
    },
  },
}
```





# Rezultati upita s korištenjem fragmenata

```
"rightComparison": {
  "name": "R2-D2",
  "appearsIn": [
    "NEWHOPE",
    "EMPIRE",
    "JEDI"
  ],
  "friends": [
    {
      "name": "Luke Skywalker"
    }, ...
  ]
}
```



# Korištenje varijabli u upitima

Umjesto pisanja novog upita mijenja se samo varijabla:

```
query HeroNameAndFriends ($episode:
  Episode) {
  hero(episode: $episode) {
    name
    friends {
      name
    }
  }
}
```

```
{
  "episode": "JEDI"
}
```



# Rezultati upita s varijablama

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
        ...
      ]
    }
  }
}
```



# Podrazumijevane vrijednosti varijabli

Varijablama koje se koriste kod upita mogu se zadati i podrazumijevane vrijednosti (ako su sve podrazumijevane, može samo naziv upita):

```
query HeroNameAndFriends ($episode:
Episode = JEDI) {
  hero(episode: $episode) {
    name
    friends {
      name
    }
  }
}
```



# Korištenje direktiva u upitima

Omogućavaju dinamičku izmjenu strukture upita  
(moguće vrijednosti @include i @skip):

```
query Hero($episode: Episode, $withFriends:
  Boolean!) {
  hero(episode: $episode) {
    name
    friends @include(if: $withFriends) {
      name
    }
  }
}
```

```
{
  "episode": "JEDI",
  "withFriends": false
}
```



# Mutacije za ažuriranje podataka

Za ažuriranje se može koristiti bilo koja vrsta upita, ali se preporučuje korištenje mutacija (sličnost REST/GET):

```
mutation CreateReviewForEpisode($ep: Episode!,
  $review: ReviewInput!) {
  createReview(episode: $ep, review: $review) {
    stars
    commentary
  }
}
```

```
{
  "ep": "JEDI",
  "review": {
    "stars": 5,
    "commentary": "This is a great movie!"
  }
}
```



# Rezultati izvođenja mutacije

```
{
  "data": {
    "createReview": {
      "stars": 5,
      "commentary": "This is a great
movie!"
    }
  }
}
```

Važna razlika mutacija u odnosu na druge upite je da se one obavezno izvode slijedno, jedna iza druge.



## Sheme i tipovi

Iako GraphQL uvijek vraća očekivane rezultate na strani klijenta, moguće je precizno opisati skup mogućih podataka što dodatno olakšava pisanje upita. Također, omogućava jednostavniju provjeru ispravnosti samih upita.

To se postiže definiranjem dostupnih tipova za upit (polja i objekti), odnosno definiranjem sheme upita.

Po svojoj strukturi sheme GraphQL upita slične su samim upitima, ali su istovremeno nezavisne od programskog jezika zaduženog za implementaciju upita u pozadini.





# Primjer jednostavne definicije

Kod definiranja sheme mogu se koristiti različite vrste skalarnih tipova podataka kao u slijedećem primjeru:

```
type Character {  
  name: String!  
  appearsIn: [Episode]!  
}
```

Character	- tip GraphQL objekta
Name i appearsIn	- polja u tipu Character
String	- osnovni skalarnih tip podatka
String! <i>not nullable</i>	- uvijek vraća vrijednost ili pogrešku
[Episode]! <i>not nullable</i>	- polje objekata Episode



# Podrazumijevana vrijednost

I ovdje se može definirati podrazumijevana vrijednost.

```
type Starship {  
  id: ID!  
  name: String!  
  length(unit: LengthUnit = METER): Float  
}
```



# Dostupne skalarne vrijednosti

## **Int**

32-bitna cjelobrojna vrijednost s predznakom.

## **Float**

Broj s pokretnim zarezom i predznakom dvostruke preciznosti.

## **String**

Niz znakova prema UTF-8 standardnu kodiranja.

## **Boolean**

Logička vrijednost true ili false.

## **ID**

Jedinstveni identifikator objekta (u stvari isto niz znakova)



# Ostale mogućnosti

Moguće je definirati i vlastiti skalarni tip podatka:

```
scalar Date
```

Postoji i mogućnost definiranja vlastitih Enum tipova podataka s ograničenim skupom vrijednosti:

```
enum Episode {  
    NEWHOPE  
    EMPIRE  
    JEDI  
}
```



# Ostale mogućnosti

Pomoću znakova [ ] mogu se definirati vlastite liste:

```
myField: [String!]
myField: null // OK
myField: [] // OK
myField: ['a', 'b'] // OK
myField: ['a', null, 'b'] // error

myField: [String]!
myField: null // error
myField: [] // OK
myField: ['a', 'b'] // OK
myField: ['a', null, 'b'] // OK
```



# Definiranje apstraktnih tipova

Moguće je navođenjem oznake **interfaces**:

```
interface Character {  
    id: ID!  
    name: String!  
    friends: [Character]  
    appearsIn: [Episode]!  
}
```

Svi izvedeni tipovi moraju sadržati nabrojene osnovne dijelove.



# Korištenje apstraktnih tipova

```
type Human implements Character {  
    id: ID!  
    name: String!  
    friends: [Character]  
    appearsIn: [Episode]!  
    starships: [Starship]  
    totalCredits: Int  
}
```

```
type Droid implements Character {  
    id: ID!  
    name: String!  
    friends: [Character]  
    appearsIn: [Episode]!  
    primaryFunction: String  
}
```



# Korištenje apstraktnih tipova

```
query HeroForEpisode($ep: Episode!) {
  hero(episode: $ep) {
    name
    primaryFunction
  }
}
{
  "errors": [
    {
      "message": "Cannot query field
\"primaryFunction\" on type \"Character\". Did you
mean to use an inline fragment on \"Droid\"?",
      "locations": [
        {
          "line": 4,
          "column": 5
        }
      ]
    }
  ] ...
}
```





# Korištenje unija

Koristi se za vraćanje različitih vrsta objekata iz upita.  
Na primjer, sljedeća definicija:

```
union SearchResult = Human | Droid |  
Starship
```

Omogućava vraćanje rezultata za bilo koji od  
dozvoljenih tipova.



# Korištenje unija

```
{
  search(text: "an") {
    ... on Human {
      name
      height
    }
    ... on Droid {
      name
      primaryFunction
    }
    ... on Starship {
      name
      length
    }
  }
}
```



# Korištenje unija

```
{
  "data": {
    "search": [
      {
        "name": "Han Solo",
        "height": 1.8
      },
      {
        "name": "Leia Organa",
        "height": 1.5
      },
      {
        "name": "TIE Advanced x1",
        "length": 9.2
      }
      ...
    ]
  }
}
```



# Objekti kao ulazni tipovi podataka

Osim jednostavnih skalarnih vrijednosti kao ulazni tipovi podataka mogu se koristiti i objekti.

U tom slučaju u samoj definiciji treba **type** zamijeniti s **input**. Na primjer:

```
input ReviewInput {  
    stars: Int!  
    commentary: String  
}
```

Na temelju toga može se definirati slijedeći oblik mutacije:



# Objekti kao ulazni tipovi podataka

```
mutation CreateReviewForEpisode($ep:  
Episode!, $review: ReviewInput!) {  
  createReview(episode: $ep, review:  
$review) {  
    stars  
    commentary  
  }  
}
```

I koristiti kao:

```
{  
  "ep": "JEDI",  
  "review": {  
    "stars": 5,  
    "commentary": "This is a great movie!"  
  }  
}
```



# Objekti kao ulazni tipovi podataka

```
{
  "data": {
    "createReview": {
      "stars": 5,
      "commentary": "This is a great
movie!"
    }
  }
}
```



# Usporedbe s drugim sličnim tehnologijama (REST API)



# Usporedba s REST API

Umjesto većeg broja krajnjih točaka s fiksnom strukturom podataka, GraphQL omogućava korištenje samo jedne krajnje točke s različitim i precizno definiranim strukturama podataka potrebnim klijentu.

U jednom GraphQL upitu mogu se sa servera zatražiti vrlo složeni podaci s međusobno manje ili više složenim vezama, a pri tome je je sam oblik upita pregledan te prilično jednostavan za razumijevanje. Server vraća odgovor koji po svojoj strukturi točno odgovara postavljenom upitu.





# Usporedba s REST API

Način postavljanja upita serveru i vraćanja podataka sa servera bitno smanjuje količinu prometa po mreži (vrlo bitno kod sporijih mobilnih mreža za prijenos podataka). Nema „Overfetching” i „Underfetching”.

GraphQL Schema Definition Language povećava produktivnost je omogućava nezavisan rad na strani servera i na strani klijenta.



# Mogući načini korištenja na strani servera

1. Izgradnja potpuno novog sustava koji preko GraphQL upita pristupa bazi podataka.
2. Pojednostavljivanje pristupa postojećim (legacy) sustavima tako da se pomoću GraphQL upita „skriva” njihova kompleksnost za aplikacije klijente.
3. Hibridni pristup – integracija prethodnih točki 1 i 2. Ovakva fleksibilnost moguća je izradom različitih **resolver** funkcija na strani servera.



# Mogući načini korištenja na strani klijenta

Uz pomoć klijent orijentiranih GraphQL biblioteka kao što su:

Relay (<https://facebook.github.io/relay/> )

Apollo (<https://www.apollographql.com/> )

Pristup podacima iz aplikacije klijenta može se svesti na:

1. Opis zahtjeva za podacima
2. Prikaz podataka u korisničkom sučelju

Umjesto REST API (priprema HTTP zahtjeva, primanje i parsing podataka, lokalno spremanje podataka, prikaz u korisničko sučelju).



# Primjer korištenja u biblioteci Relay

```
import {graphql} from 'react-relay';
```

```
graphql`  
  query MyQuery {  
    viewer {  
      id  
    }  
  }  
`;  
`;
```

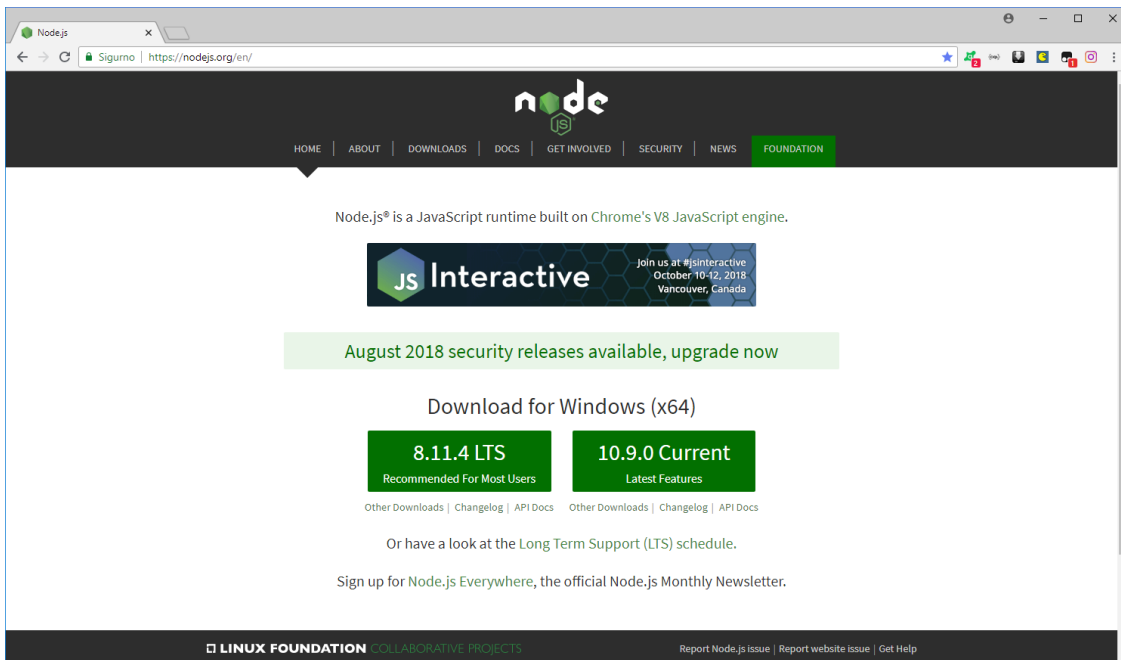


# Primjeri korištenja u programskom jeziku JavaScript



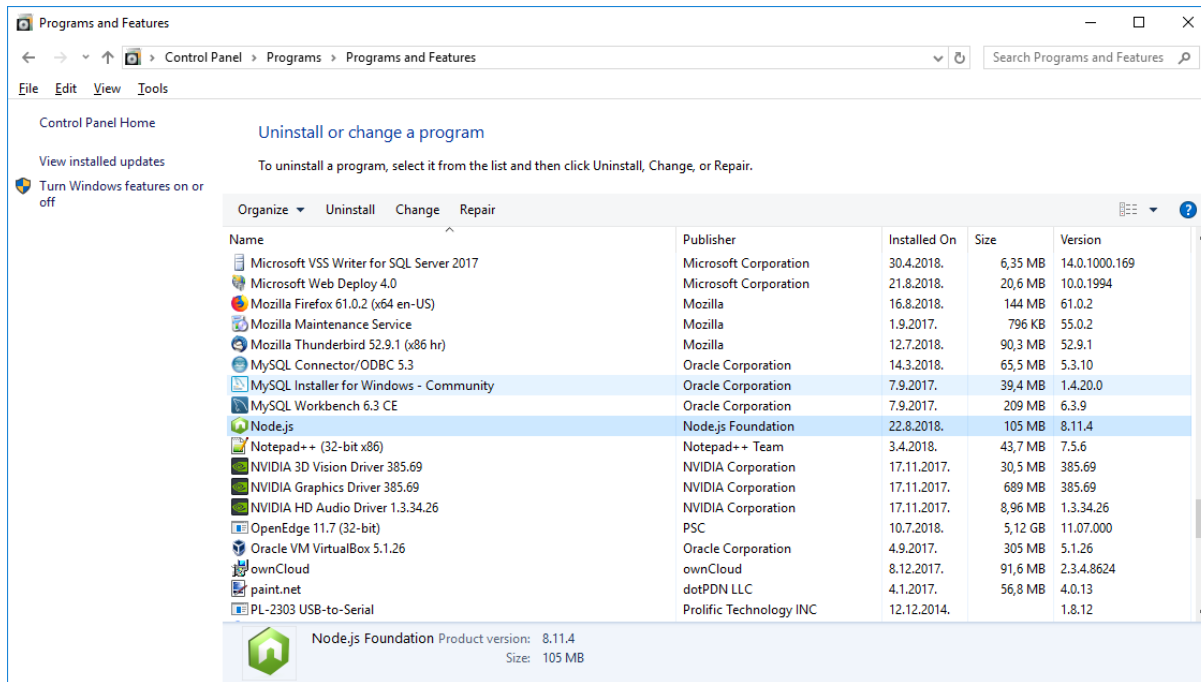
# JavaScript – priprema Windows računala

Preuzimanje paketa **node.js**, ako ne postoji na računalu  
- <https://nodejs.org/en/>



# JavaScript – priprema Windows računala

Instalacija paketa **node.js** izvodi se iz preuzete EXE datoteke.



The screenshot shows the Windows 'Programs and Features' control panel window. The title bar reads 'Programs and Features'. The address bar shows the path: 'Control Panel > Programs > Programs and Features'. The main content area is titled 'Uninstall or change a program' and includes the instruction: 'To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.' Below this is a table of installed programs. The 'Node.js' entry is highlighted in blue. At the bottom of the window, a detailed view for Node.js is shown, indicating it is from Node.js Foundation, version 8.11.4, and 105 MB in size.

Name	Publisher	Installed On	Size	Version
Microsoft VSS Writer for SQL Server 2017	Microsoft Corporation	30.4.2018.	6,35 MB	14.0.1000.169
Microsoft Web Deploy 4.0	Microsoft Corporation	21.8.2018.	20,6 MB	10.0.1994
Mozilla Firefox 61.0.2 (x64 en-US)	Mozilla	16.8.2018.	144 MB	61.0.2
Mozilla Maintenance Service	Mozilla	1.9.2017.	796 KB	55.0.2
Mozilla Thunderbird 52.9.1 (x86 hr)	Mozilla	12.7.2018.	90,3 MB	52.9.1
MySQL Connector/ODBC 5.3	Oracle Corporation	14.3.2018.	65,5 MB	5.3.10
MySQL Installer for Windows - Community	Oracle Corporation	7.9.2017.	39,4 MB	1.4.20.0
MySQL Workbench 6.3 CE	Oracle Corporation	7.9.2017.	209 MB	6.3.9
Node.js	Node.js Foundation	22.8.2018.	105 MB	8.11.4
Notepad++ (32-bit x86)	Notepad++ Team	3.4.2018.	43,7 MB	7.5.6
NVIDIA 3D Vision Driver 385.69	NVIDIA Corporation	17.11.2017.	30,5 MB	385.69
NVIDIA Graphics Driver 385.69	NVIDIA Corporation	17.11.2017.	689 MB	385.69
NVIDIA HD Audio Driver 1.3.34.26	NVIDIA Corporation	17.11.2017.	8,96 MB	1.3.34.26
OpenEdge 11.7 (32-bit)	PSC	10.7.2018.	5,12 GB	11.07.000
Oracle VM VirtualBox 5.1.26	Oracle Corporation	4.9.2017.	305 MB	5.1.26
ownCloud	ownCloud	8.12.2017.	91,6 MB	2.3.4.8624
paint.net	dotPDN LLC	4.1.2017.	56,8 MB	4.0.13
PL-2303 USB-to-Serial	Prolific Technology INC	12.12.2014.		1.8.12

# JavaScript – osnovni primjer

Programski kod primjera

```
var { graphql, buildSchema } =  
require('graphql');
```

```
// Construct a schema, using GraphQL  
schema language
```

```
var schema = buildSchema(`  
  type Query {  
    hello: String  
  }  
`);
```





# JavaScript – osnovni primjer (nastavak)

```
// The root provides a resolver function
for each API endpoint
var root = {
  hello: () => {
    return 'Hello world!';
  },
};
// Run the GraphQL query '{ hello }' and
print out the response
graphql(schema, '{ hello }',
root).then((response) => {
  console.log(response);
});
```



# JavaScript – osnovni primjer (izvođenje)

```
C:\WINDOWS\system32\cmd.exe
C:\xampp\htdocs\GraphQL>node JSstart.js
{ data: { hello: 'Hello world!' } }
C:\xampp\htdocs\GraphQL>
```



# JavaScript – dodatni primjer

Koristi se dodatni web Framework za Node.js dostupan na adresi: <https://expressjs.com/>

Koraci:

1. Inicijalizacija projekta
2. Instalacija frameworka
3. Pisanje programskog koda
4. Izvođenje primjera



# JavaScript – inicijalizacija projekta

```
C:\WINDOWS\system32\cmd.exe
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (graphql)
version: (1.0.0)
description:
entry point: (JSStart.js)
test command: hello
git repository:
keywords:
author:
license: (ISC)
About to write to C:\xampp\htdocs\GraphQL\package.json:

{
  "name": "graphql",
  "version": "1.0.0",
  "description": "",
  "main": "JSStart.js",
  "scripts": {
    "test": "hello"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes) y
C:\xampp\htdocs\GraphQL>
```



# JavaScript – inicijalizacija projekta

## >npm init

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and save it as a dependency in the package.json file.



# JavaScript – inicijalizacija projekta

Press ^C at any time to quit.

package name: (graphql)

version: (1.0.0)

description:

entry point: (JSStart.js)

test command: hello

git repository:

keywords:

author:

license: (ISC)

About to write to

C:\xampp\htdocs\GraphQL\package.json:



# JavaScript – inicijalizacija projekta

```
{  
  "name": "graphql",  
  "version": "1.0.0",  
  "description": "",  
  "main": "JSStart.js",  
  "scripts": {  
    "test": "hello"  
  },  
  "author": "",  
  "license": "ISC"  
}
```

Is this ok? (yes) y



# JavaScript – instalacija frameworka

```
C:\WINDOWS\system32\cmd.exe
C:\xampp\htdocs\GraphQL>npm install express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN graphql@1.0.0 No description
npm WARN graphql@1.0.0 No repository field.

+ express@4.16.3
added 50 packages in 1.751s

C:\xampp\htdocs\GraphQL>
```





# JavaScript – instalacija frameworka

```
>npm install express
```

```
npm notice created a lockfile as package-lock.json. You should commit this file.
```

```
npm WARN graphql@1.0.0 No description
```

```
npm WARN graphql@1.0.0 No repository field.
```

```
+ express@4.16.3
```

```
added 50 packages in 1.751s
```



# JavaScript – programski kod

```
var express = require('express');
var graphqlHTTP = require('express-graphql');
var { buildSchema } = require('graphql');

// Construct a schema, using GraphQL
// schema language
var schema = buildSchema(`
  type Query {
    hello: String
  }
`);
```



# JavaScript – programski kod

```
for each
API endpoint
var root = {
  hello: () => {
    return 'Hello world!';
  },
};
var app = express();
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
})));
```



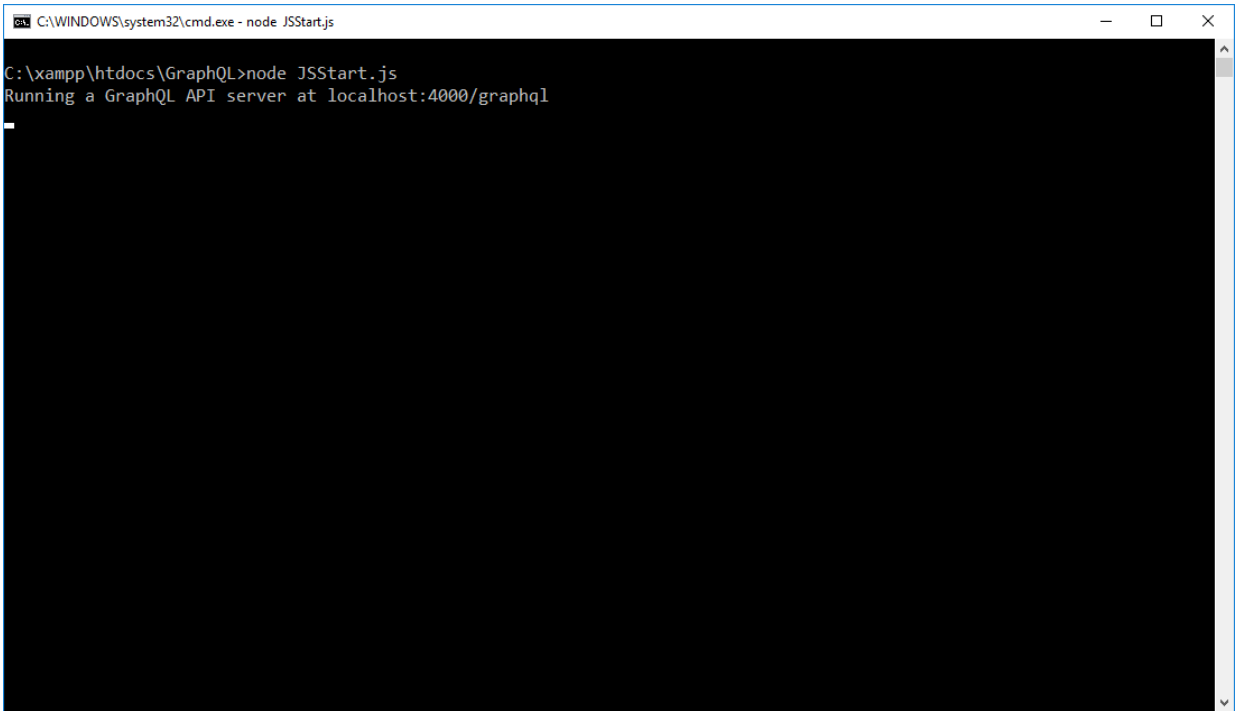
# JavaScript – programski kod

```
app.listen(4000);  
console.log('Running a GraphQL API server  
at  
localhost:4000/graphql');
```



# JavaScript – pokretanje primjera

> node JSStart.js

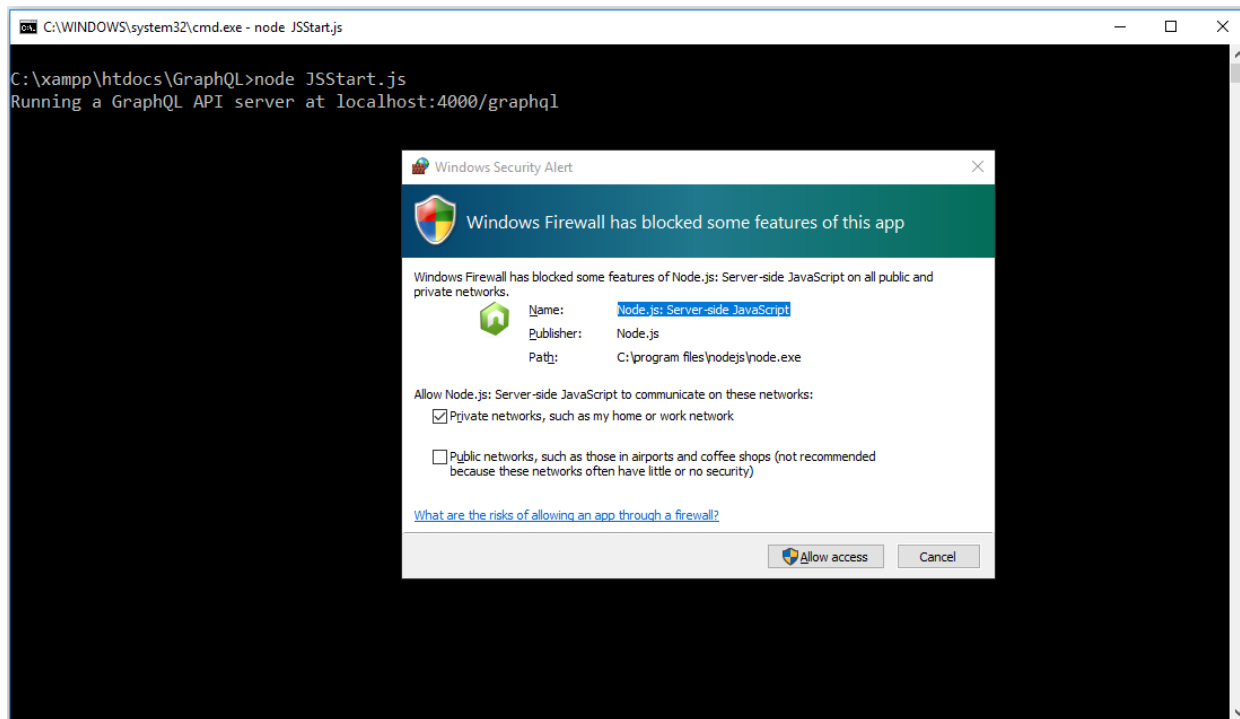


```
C:\WINDOWS\system32\cmd.exe - node JSStart.js  
C:\xampp\htdocs\GraphQL>node JSStart.js  
Running a GraphQL API server at localhost:4000/graphql
```



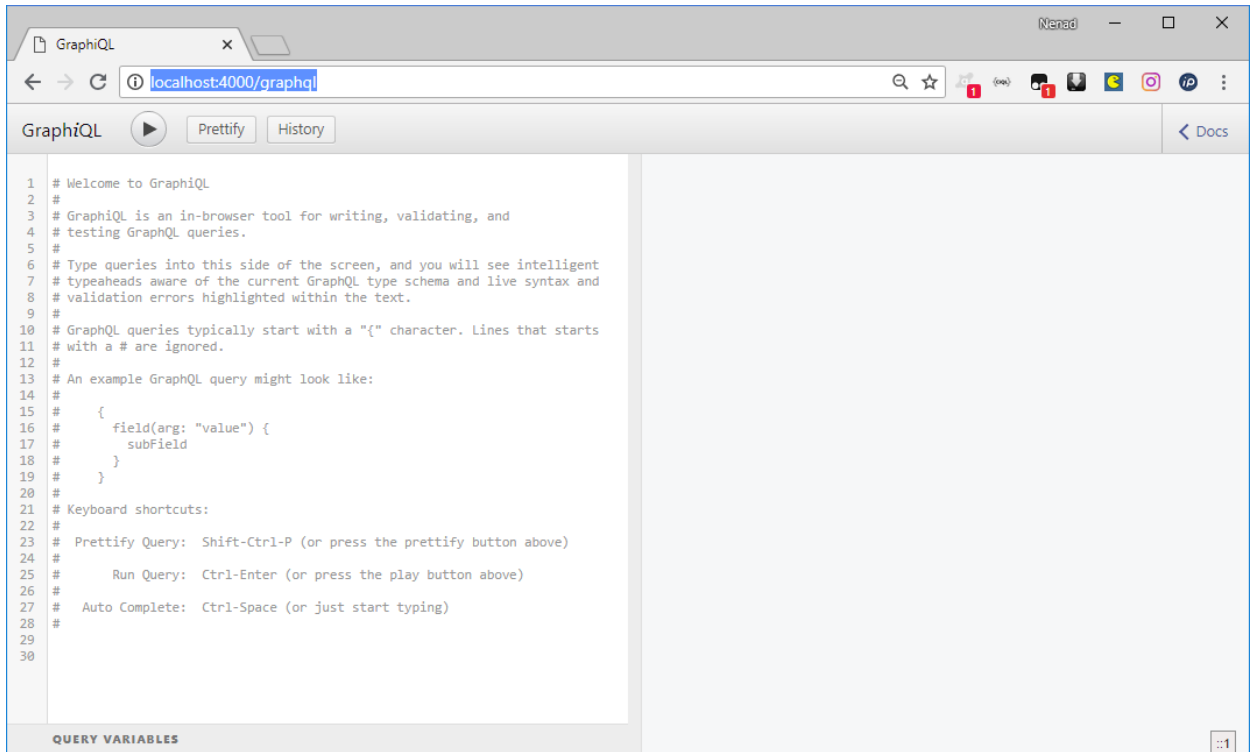
# JavaScript – izvođenje (Firewall dozvola)

## Pokretanje: node JSStart.js



# JavaScript – izvođenje u Chrome pregledniku

Upisuje se adresa: **localhost:4000/graphql**

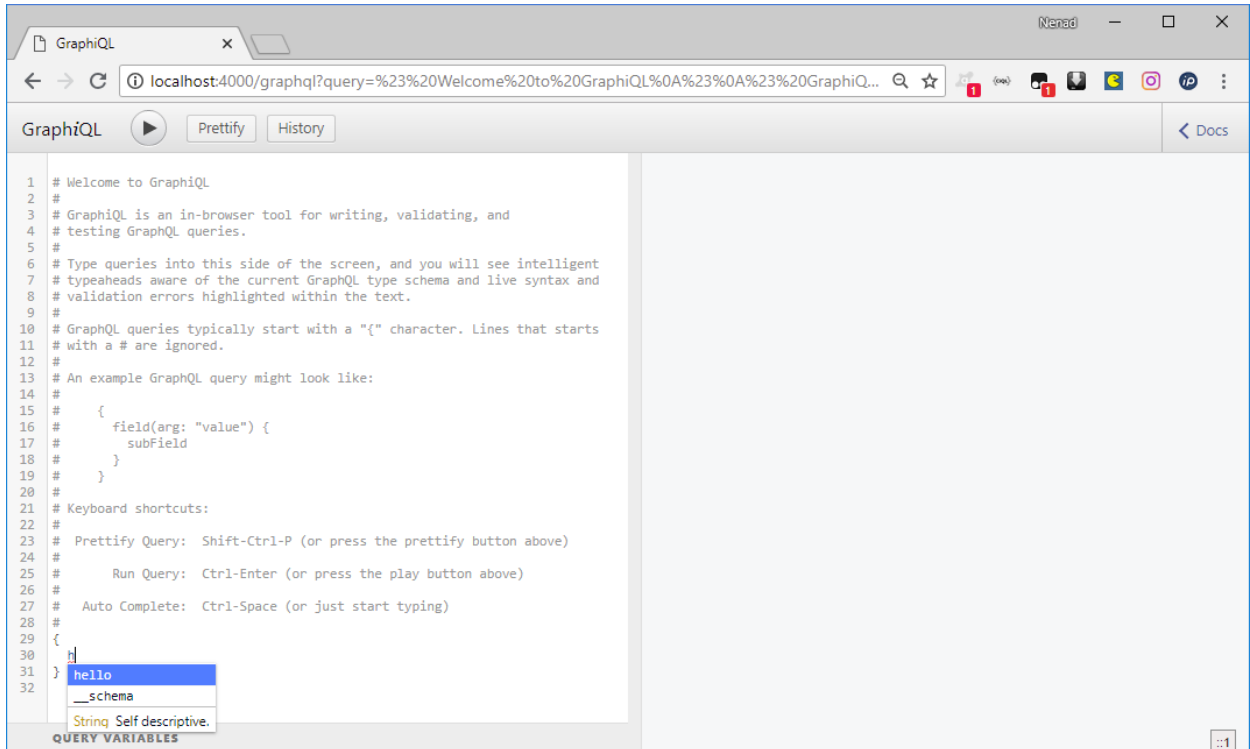


```
1 # Welcome to GraphQL
2 #
3 # GraphQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeahead aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subfield
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 # Prettify Query: Shift-Ctrl-P (or press the prettify button above)
24 #
25 # Run Query: Ctrl-Enter (or press the play button above)
26 #
27 # Auto Complete: Ctrl-Space (or just start typing)
28 #
29
30
```

QUERY VARIABLES

# JavaScript – izvođenje u Chrome pregledniku

## Upis i odabir dostupnih naredbi: hello



```
1 # Welcome to GraphQL
2 #
3 # GraphQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subField
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Prettify Query: Shift-Ctrl-P (or press the prettify button above)
24 #
25 #   Run Query: Ctrl-Enter (or press the play button above)
26 #
27 #   Auto Complete: Ctrl-Space (or just start typing)
28 #
29 {
30 }
31 }
32 }
```

hello  
\_\_schema  
String! Self descriptive.

QUERY VARIABLES



# JavaScript – izvođenje u Chrome pregledniku

## Izvođenje i prikaz rezultata

```
1 # Welcome to GraphQL
2 #
3 # GraphQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeahead aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subfield
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Prettify Query: Shift-Ctrl-P (or press the prettify button above)
24 #
25 #   Run Query: Ctrl-Enter (or press the play button above)
26 #
27 #   Auto Complete: Ctrl-Space (or just start typing)
28 #
29 # {
30 #   hello
31 # }
32
```

```
{
  "data": {
    "hello": "Hello world!"
  }
}
```

QUERY VARIABLES

# Primjeri korištenja na relacijskoj bazi podataka MySQL pomoću programskog jezika PHP



# PHP – priprema računala za korištenje

Podrazumijeva se na je na računalu instaliran programski jezik PHP i baza podataka MySQL ili MariaDB (npr. XAMPP - [www.apachefriends.org](http://www.apachefriends.org) )

Dodatna PHP biblioteka za podršku GraphQL upita dostupna je na adresi:

<https://github.com/webonyx/graphql-php>

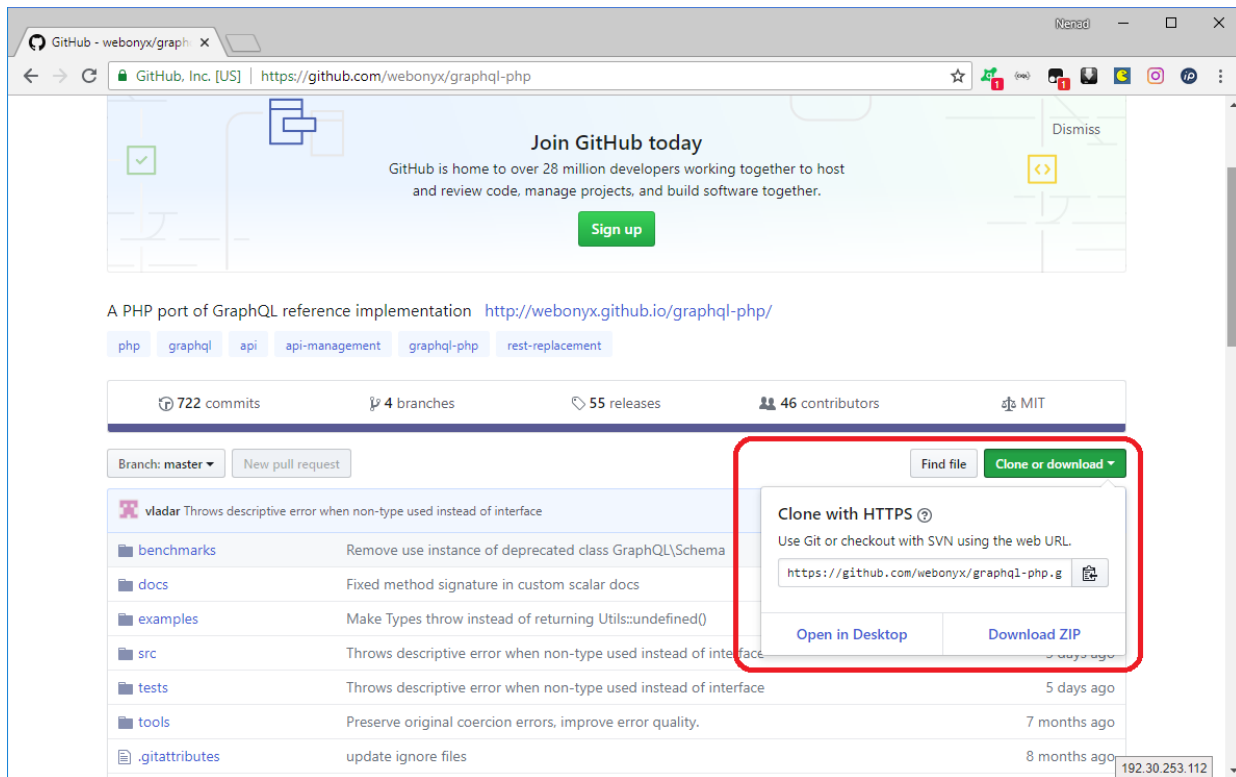
Dva glavna oblika instalacije:

1. Korištenje gumba **Clone or Download**
2. Korištenje alata **Composer**



# PHP – priprema računala za korištenje

## Korištenje gumba **Clone or Download**



The screenshot shows a web browser window displaying the GitHub repository page for 'webonyx/graphql-php'. The page features a 'Join GitHub today' banner, a 'Sign up' button, and a list of tags including 'php', 'graphql', 'api', 'api-management', 'graphql-php', and 'rest-replacement'. Below the tags, repository statistics are shown: 722 commits, 4 branches, 55 releases, 46 contributors, and MIT license. A dropdown menu for 'Clone or download' is open, showing the 'Clone with HTTPS' option, the repository URL 'https://github.com/webonyx/graphql-php.git', and buttons for 'Open in Desktop' and 'Download ZIP'. A red box highlights this dropdown menu.

GitHub - webonyx/graphql-php

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

A PHP port of GraphQL reference implementation <http://webonyx.github.io/graphql-php/>

php graphql api api-management graphql-php rest-replacement

722 commits 4 branches 55 releases 46 contributors MIT

Branch: master New pull request

Find file Clone or download

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

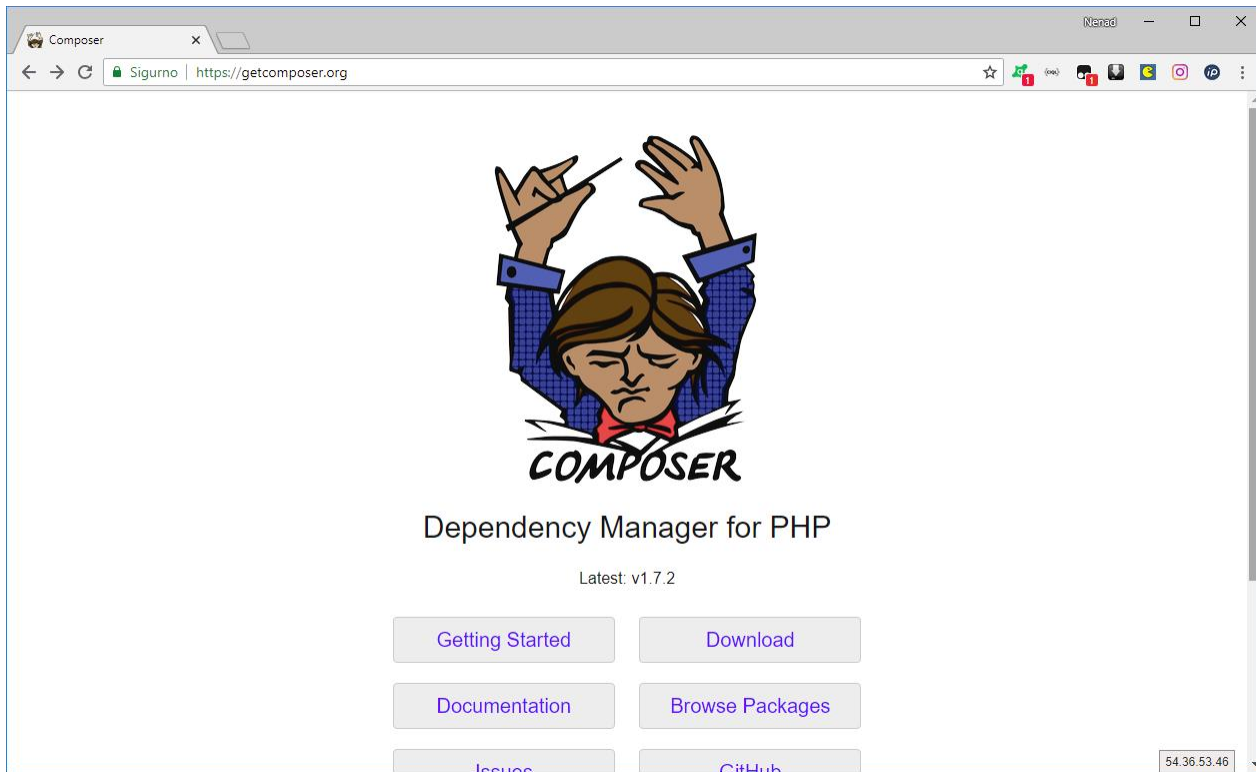
<https://github.com/webonyx/graphql-php.git>

Open in Desktop Download ZIP

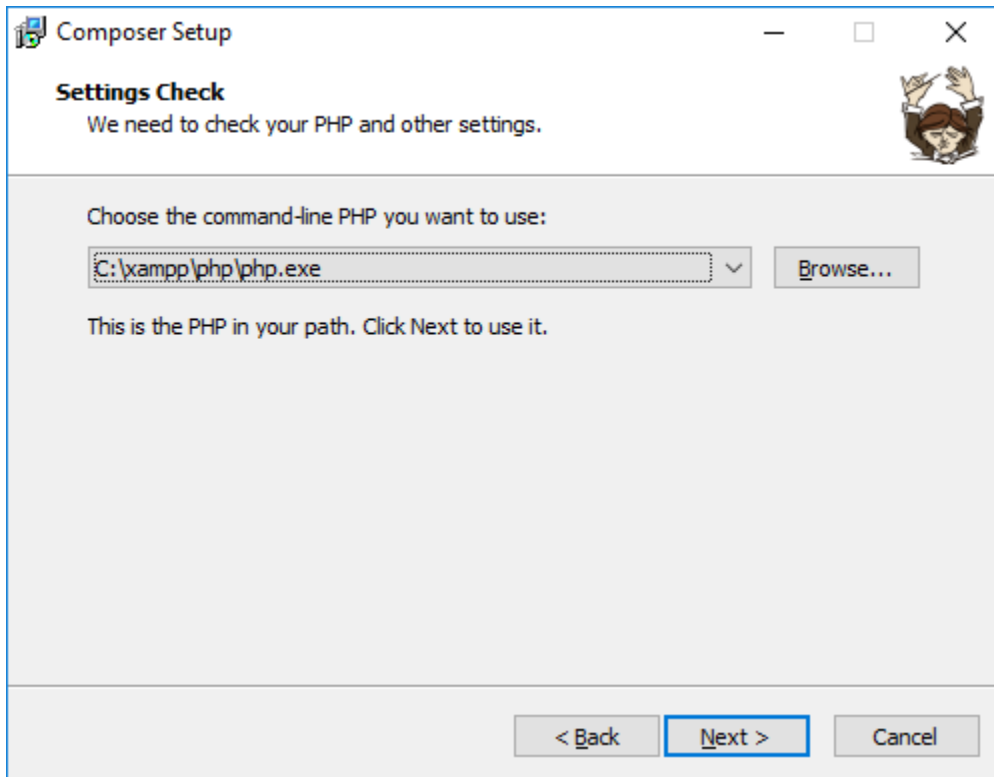
vladar	Throws descriptive error when non-type used instead of interface	
benchmarks	Remove use instance of deprecated class GraphQL\Schema	
docs	Fixed method signature in custom scalar docs	
examples	Make Types throw instead of returning Util::undefined()	
src	Throws descriptive error when non-type used instead of interface	
tests	Throws descriptive error when non-type used instead of interface	5 days ago
tools	Preserve original coercion errors, improve error quality.	7 months ago
.gitattributes	update ignore files	8 months ago

# PHP – priprema računala za korištenje

## Korištenje alata **Composer** ([getcomposer.org](https://getcomposer.org))



# PHP – priprema računala za korištenje Composer – instalacija i povezivanje s PHP-om



# PHP – priprema računala za korištenje

Preuzimanje biblioteke pomoću alata Composer na temelju datoteke **composer.json**

```
{  
    "require": {  
        "webonyx/graphql-php": "^0.12.5"  
    }  
}
```

**> composer require webonyx/graphql-php**



# PHP – priprema računala za korištenje

Preuzimanje biblioteke

> **composer require webonyx/graphql-php**

```
C:\WINDOWS\system32\cmd.exe

C:\xampp\htdocs\GraphQLPHP>composer require webonyx/graphql-php
Using version ^0.12.5 for webonyx/graphql-php
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
 - Installing webonyx/graphql-php (v0.12.5): Loading from cache
webonyx/graphql-php suggests installing react/promise (To leverage async resolving on React PHP platform)
webonyx/graphql-php suggests installing psr/http-message (To use standard GraphQL server)
Writing lock file
Generating autoload files

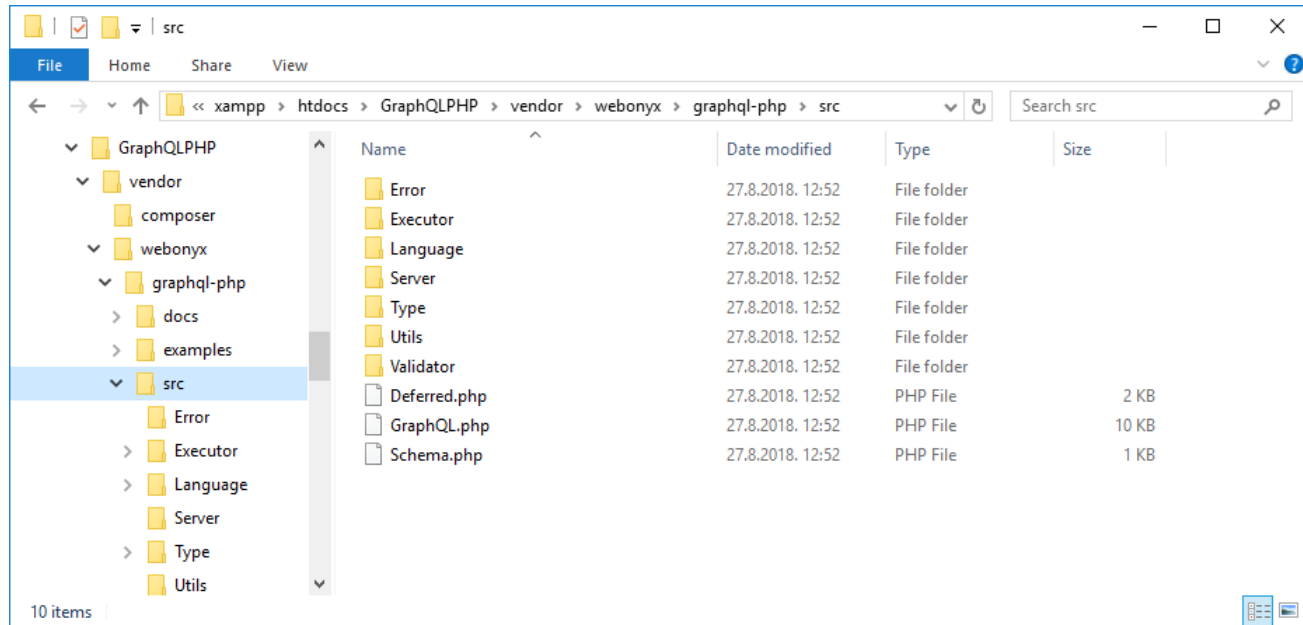
C:\xampp\htdocs\GraphQLPHP>
```





# PHP – priprema računala za korištenje

## Rezultati preuzimanja biblioteke



# PHP – priprema računala za korištenje

## Dodatni alat za Chrome: ChromeiQL

chrome web-trgovina

Pretražite trgovinu

Proširenja

Teme

KATEGORIJE

Sve

ZNAČAJKE

Radi izvan mreže

Od Googlea

Besplatno

Dostupno za sve uređaje

Funkcionalno

OČJENE

★★★★★

★★★★☆

★★★☆☆

★★☆☆☆

★☆☆☆☆

Pravila o privatnosti

ChromeiQL

nudi ermancelen

★★★★★ (23) | [Alati za razvojne programere](#) | Broj korisnika: 8.968

PREGLED RECENZIJE PODRŠKA SRODNO

DODANO U CHROME

Kompatibilno s vašim uređajem

**Chrome app wrapper for the GraphQL tool**

Simple extension wrapper around the great GraphQL IDE allowing it to work with any GraphQL endpoint of your choosing.

As long as you are authorized to send requests to the endpoint from your current browser session (cookies are set etc.) you are good to use ChromeiQL.

New in 1.0

==

\* Upgraded GraphQL to version 0.11.3 with history, scroll, copy, paste, undo and redo.

[Web-lokacija](#)

[Prijavi zlouporabu](#)

**Dodatne informacije**

Verzija: 1.0

Ažurirano: 21. kolovoza 2017.

Veličina: 681KB

Izik: Engleski

```
1 query whereState($address: String!) {
2   country { ISO: $address }
3   name {
4     en
5     de
6   }
7   program_id
8   location {
9     latitude
10    longitude
11  }
12 }
13 }
14 }
```

```
{
  "data": {
    "country": {
      "ISO": "DE"
    },
    "name": {
      "en": "Federal States",
      "de": "Staat"
    },
    "program_id": "NOLINER",
    "location": {
      "latitude": "50.95",
      "longitude": "10.98"
    }
  }
}
```

QUERY VARIABLES

```
{
  "address": "5. 8. 8. 8"
}
```

Search locales...

Obtain including of location data available for a group of address

name

city city

continent continent

country country

location location

postal postal

registered\_country registered\_country

unpublished unpublished



# PHP – priprema računala za korištenje

## Dodatni alat za Chrome: ChromeiQL

The screenshot shows the Chrome browser's extension management page. The address bar shows 'Chrome | chrome://extensions'. The page title is 'Proširenja'. A search bar contains 'Pretražite proširenja'. The 'Način za razvojne programere' toggle is turned off. The 'ChromeiQL' extension is highlighted with a red box. It has a blue 'Ukloni' button and a blue toggle switch. Other extensions include 'Avast Online Security', 'Avast SafePrice', 'Desktop for Instagram', and 'Grammar and Spelling checker by Ginger'.

Extension Name	Description	Buttons	Toggle
Avast Online Security	Avast Browser Security and Web Reputation Plugin.	Detalji, Ukloni	On
Avast SafePrice	Pronađite najbolje cijene, ponude i kupone tijekom kupnje putem interneta uz usporedbu cijena i kupone koje omogućuje Avast.	Detalji, Ukloni	Off
ChromeiQL	Chrome app wrapper for the GraphQL tool	Detalji, Ukloni	On
Desktop for Instagram	Browse web mobile Instagram site directly from your Desktop (Pc / Mac)	Detalji, Ukloni	On
Google dokumenti izvanmrežno	Izvršavajte zadatke izvanmrežno uz proizvode iz skupine Google dokumenata.	Detalji, Ukloni	On
Grammar and Spelling checker by Ginger	Improve your English communication with Ginger's #1 spelling and grammar checker!	Detalji, Ukloni	Off



# PHP – osnovni primjer

```
<?php
```

```
require_once __DIR__ . './autoload.php';
```

```
use GraphQL\Type\Definition\ObjectType;
```

```
use GraphQL\Type\Definition\Type;
```

```
use GraphQL\Type\Schema;
```

```
use GraphQL\GraphQL;
```

```
try {  
    $queryType = new ObjectType([  
        'name' => 'Query',  
        'fields' => [  
            'echo' => [  
                'type' => Type::string(),  
                'args' => [  

```



# PHP – osnovni primjer

```
'message' =>
['type' => Type::string()],
'resolve' => function ($root, $args) {
    return $root['prefix'] .
        $args['message'];
}
],
],
]);
```



# PHP – osnovni primjer

```
$mutationType = new ObjectType([
    'name' => 'Calc',
    'fields' => [
        'sum' => [
            'type' => Type::int(),
            'args' => [
                'x' => ['type' => Type::int()],
                'y' => ['type' => Type::int()],
            ],
            'resolve' => function ($root, $args) {
                return $args['x'] + $args['y'];
            },
        ],
    ],
]);
```



# PHP – osnovni primjer

```
$schema = new Schema([
    'query' => $queryType,
    'mutation' => $mutationType,
]);

$rawInput
    =file_get_contents('php://input');
$input = json_decode($rawInput, true);
$query = $input['query'];
$variableValues
    =isset($input['variables']) ?
$input['variables'] : null;
$rootValue = ['prefix' => 'You said: ];
$result = GraphQL::executeQuery($schema,
    $query, $rootValue, null,
    $variableValues);
$output = $result->toArray();
```



# PHP – osnovni primjer

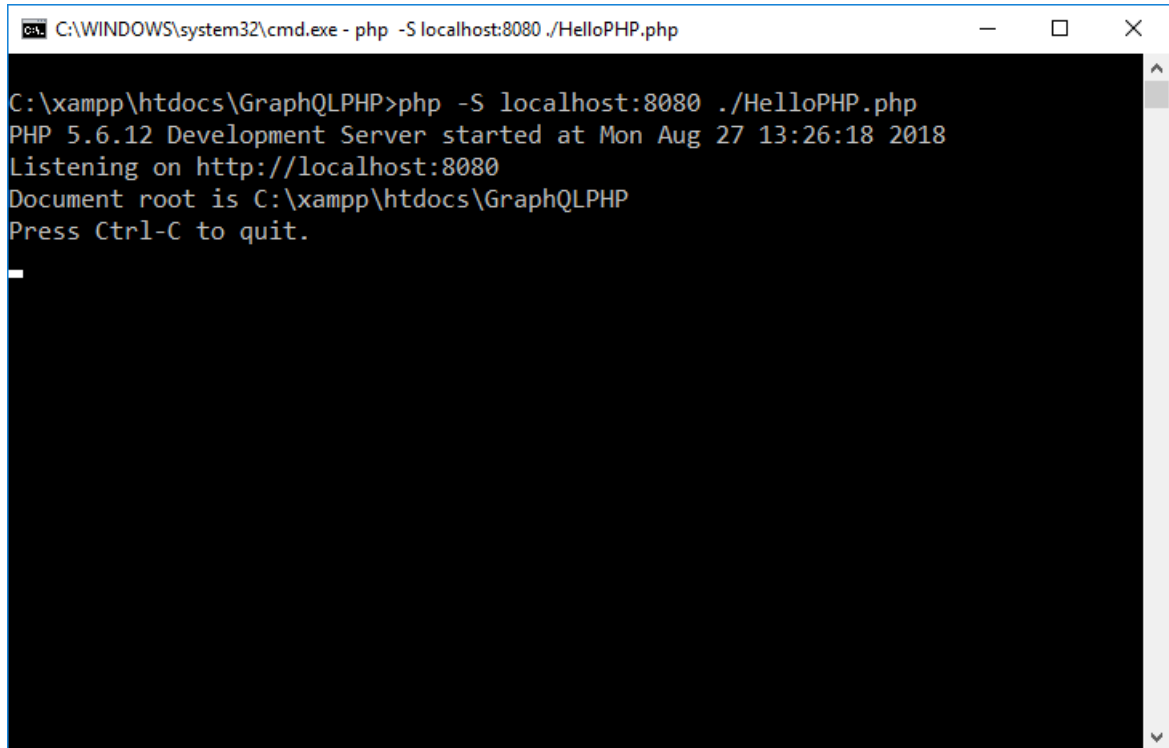
```
} catch (\Exception $e) {  
    $output = [  
        'error' => [  
            'message' => $e->getMessage()  
        ]  
    ];  
}  
header('Content-Type: application/json;  
charset=UTF-8');  
echo json_encode($output);
```





# PHP – izvođenje primjeraHelloPHP.php

> php -S localhost:8080 ./HelloPHP.php



```
C:\WINDOWS\system32\cmd.exe - php -S localhost:8080 ./HelloPHP.php

C:\xampp\htdocs\GraphQLPHP>php -S localhost:8080 ./HelloPHP.php
PHP 5.6.12 Development Server started at Mon Aug 27 13:26:18 2018
Listening on http://localhost:8080
Document root is C:\xampp\htdocs\GraphQLPHP
Press Ctrl-C to quit.
```



# PHP – izvođenje primjeraHelloPHP.php

Koristi se preglednik **Chrome** i dodatak **ChromeiQL**

Za korištenje primjera treba upisati istu adresu kao i kod pokretanje primjera u PHP-u:

**http://localhost:8080**

U lijevi dio prozora upisuje se naredba:

```
query {  
  echo(message: "Hello World")  
}
```



# PHP – izvođenje primjeraHelloPHP.php

Chrome web-trgovina - P X chrome-extension://fkia X

ChromeQL | chrome-extension://fkiamalpiddkjmcmjfbieiclmej/chromeiql.html

GraphiQL | Prettify | http://localhost:8080 | Set endpoint | < Docs

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subField
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Run Query:  Ctrl-Enter (or press the play button above)
24 #
25 #   Auto Complete:  Ctrl-Space (or just start typing)
26 #
27 #
28 #
29 query {
30   echo(message: "Hello World")
31 }
32 |
33
```

```
{
  "data": {
    "echo": "You said: Hello World"
  }
}
```

QUERY VARIABLES



# PHP – izvođenje u ChromeiQL

## Rezultat izvođenja

```
{  
  "data": {  
    "echo": "You said: Hello World"  
  }  
}
```



# PHP – pogreška kod izvođenja

## Primjer upita

```
query {  
  echo(messages: "Hello World")  
}
```

## I obrade pogreške

```
{  
  "errors": [{  
    "message": "Unknown argument \"messages\" on  
field \"echo\" of type \"Query\". Did you mean  
\"message\"?",  
    "category": "graphql",  
    "locations": [{  
      "line": 30,  
      "column": 8  
    }]  
  }]  
}
```



# PHP – pogreška kod izvođenja

The screenshot shows the GraphiQL browser extension interface. The left pane contains a GraphQL query with a syntax error. The right pane displays the resulting JSON error response.

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subField
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Run Query:  Ctrl-Enter (or press the play button above)
24 #
25 #   Auto Complete:  Ctrl-Space (or just start typing)
26 #
27
28
29 query {
30   echo(messages: "Hello World")
31 }
32
33
```

```
{
  "errors": [
    {
      "message": "Unknown argument \"messages\" on field
      \"echo\" of type \"Query\". Did you mean \"message\"?",
      "category": "graphql",
      "locations": [
        {
          "line": 30,
          "column": 8
        }
      ]
    }
  ]
}
```

QUERY VARIABLES



# PHP – primjer korištenja mutacije

## Primjer mutacije

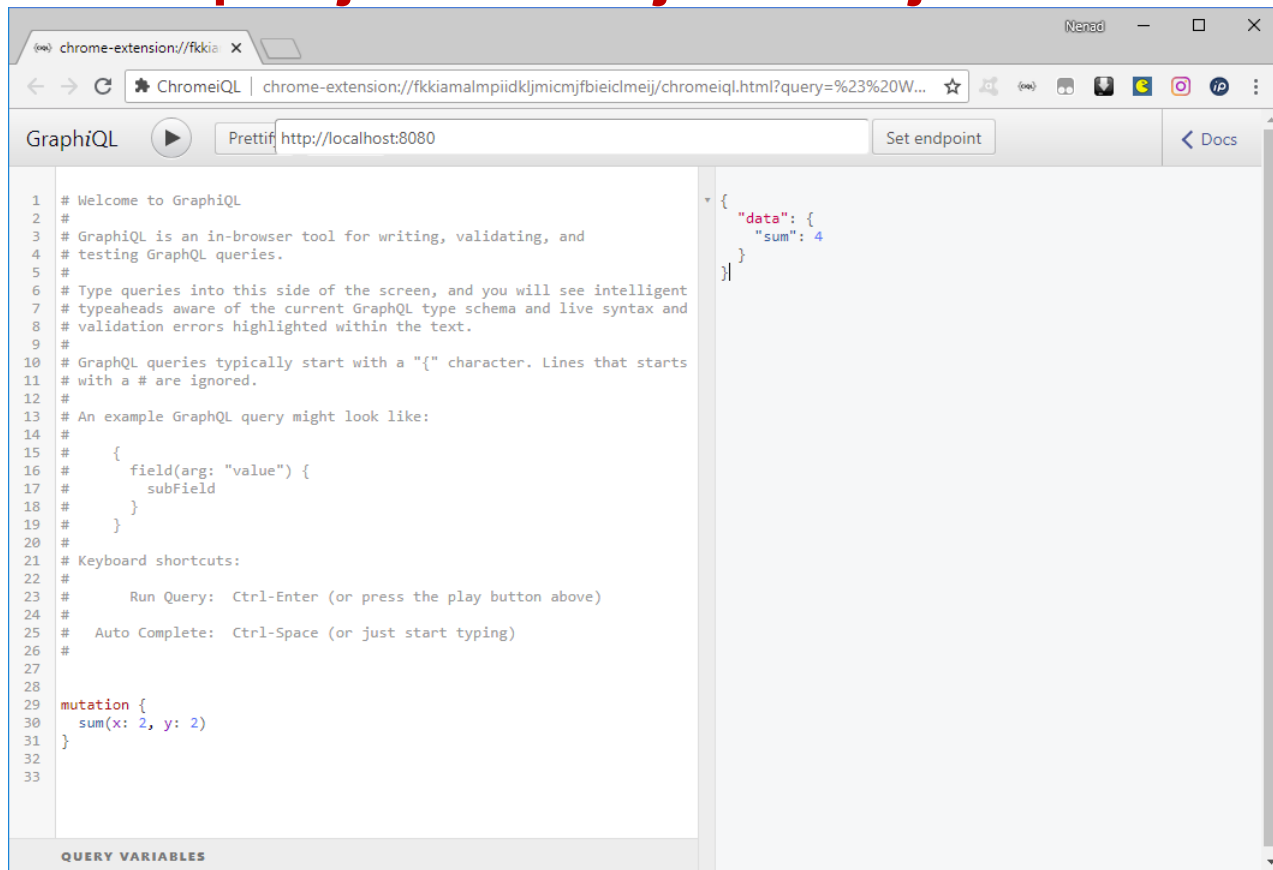
```
mutation {  
    sum(x: 2, y: 2)  
}
```

## Rezultat izvođenja mutacije

```
{  
  "data": {  
    "sum": 4  
  }  
}
```



# PHP – primjer korištenja mutacije



```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeaheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subField
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Run Query:  Ctrl-Enter (or press the play button above)
24 #
25 #   Auto Complete:  Ctrl-Space (or just start typing)
26 #
27
28
29 mutation {
30   sum(x: 2, y: 2)
31 }
32
33
```

```
{
  "data": {
    "sum": 4
  }
}
```

QUERY VARIABLES



# Primjer povezan s MySQL bazom podataka

Koristi se demo baza podataka dvojice autora (Patrick Crews i Giuseppe Maxia) opisana je na adresi:

<https://dev.mysql.com/doc/employee/en/employees-introduction.html>

A sam sadržaj baze podataka može se preuzeti s adrese:

[https://github.com/datacharmer/test\\_db](https://github.com/datacharmer/test_db)

Po svojoj internoj strukturi baza podataka je prilično jednostavna (sastoji se od svega šest tablica i dva pogleda), ali zato sadrži popriličan broj slogova u tablicama – oko 3 000 000.

Zbog takve veličine može se koristiti za druge vrste primjera poput optimizacije upita na bazu podataka i slično.



# Demo baza - preuzimanje

The screenshot shows a web browser window displaying the GitHub repository page for 'datacharmer/test\_db'. The browser's address bar shows the URL 'https://github.com/datacharmer/test\_db'. Below the browser window, a summary bar indicates '21 commits', '1 branch', '0 releases', and '3 contributors'. The main content area shows a list of files and folders with their commit messages and dates. The files listed include 'images', 'sakila', 'Changelog', 'README.md', 'employees.sql', 'employees\_partitioned.sql', 'employees\_partitioned\_5.1.sql', 'load\_departments.dump', 'load\_dept\_emp.dump', 'load\_dept\_manager.dump', 'load\_employees.dump', and 'load\_salaries1.dump'. The commit messages range from 'Initial deployment' to 'Made load files editor-friendly'. The dates range from '3 years ago' to '5 months ago'. A small tooltip with the IP address '192.30.253.112' is visible in the bottom left corner of the browser window.

GitHub - datacharmer/test\_db

GitHub, Inc. [US] | https://github.com/datacharmer/test\_db

A sample MySQL database with an integrated test suite, used to test your applications and database servers

21 commits   1 branch   0 releases   3 contributors

Branch: master   New pull request   Find file   Clone or download

datacharmer Merge pull request #8 from tedwa5/master   Latest commit 0b3c979 on 6 Apr

images	Initial deployment	3 years ago
sakila	Add description for Sakila data	3 years ago
Changelog	Initial deployment	3 years ago
README.md	Update README.md	5 months ago
employees.sql	Added missing semicolon	3 years ago
employees_partitioned.sql	Added missing semicolon	3 years ago
employees_partitioned_5.1.sql	Added missing semicolon	3 years ago
load_departments.dump	Made load files editor-friendly	3 years ago
load_dept_emp.dump	Made load files editor-friendly	3 years ago
load_dept_manager.dump	Made load files editor-friendly	3 years ago
load_employees.dump	Made load files editor-friendly	3 years ago
load_salaries1.dump	Made load files editor-friendly	3 years ago

192.30.253.112



# Demo baza – pregled količine podataka

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length
departments	InnoDB	10	Compact	9	1820	16.0 KiB	0.0 bytes	16.0 KiB
dept_emp	InnoDB	10	Compact	331570	36	11.5 MiB	0.0 bytes	5.5 MiB
dept_manager	InnoDB	10	Compact	24	682	16.0 KiB	0.0 bytes	16.0 KiB
employees	InnoDB	10	Compact	299246	50	14.5 MiB	0.0 bytes	0.0 bytes
salaries	InnoDB	10	Compact	1910497	35	64.6 MiB	0.0 bytes	0.0 bytes
titles	InnoDB	10	Compact	442070	46	19.6 MiB	0.0 bytes	0.0 bytes



# Demo baza – pregled strukture

The screenshot displays the phpMyAdmin interface for a MySQL database named 'employees'. The left sidebar shows the database structure tree with 'employees' selected. The main panel shows the 'Strukturu' (Structure) tab for the 'employees' database. A table list is visible, and the 'employees' table is selected. Below the table list, there is a form to create a new table.

Tablica	Aktivnost	Redaka	Vrsta	Uspoređivanje	Veličina	Prepunjenje
current_dept_emp	Pretraživanje	Strukturu	Traži	Umetni	Ispusti	-0 Prikaz ---
departments	Pretraživanje	Strukturu	Traži	Umetni	Ispustni	-9 InnoDB latin1_swedish_ci 32 kB -
dept_emp	Pretraživanje	Strukturu	Traži	Umetni	Ispustni	-331,570 InnoDB latin1_swedish_ci 17 MB -
dept_emp_latest_date	Pretraživanje	Strukturu	Traži	Umetni	Ispusti	-0 Prikaz ---
dept_manager	Pretraživanje	Strukturu	Traži	Umetni	Ispustni	-24 InnoDB latin1_swedish_ci 32 kB -
employees	Pretraživanje	Strukturu	Traži	Umetni	Ispustni	-299,245 InnoDB latin1_swedish_ci 14.5 MB -
salaries	Pretraživanje	Strukturu	Traži	Umetni	Ispustni	-1,918,497 InnoDB latin1_swedish_ci 64.6 MB -
titles	Pretraživanje	Strukturu	Traži	Umetni	Ispustni	-442,870 InnoDB latin1_swedish_ci 19.6 MB -
8 tablice	Zbroj				InnoDB utf8_general_ci	115.8 MB 0 B

Below the table list, there is a form to create a new table:

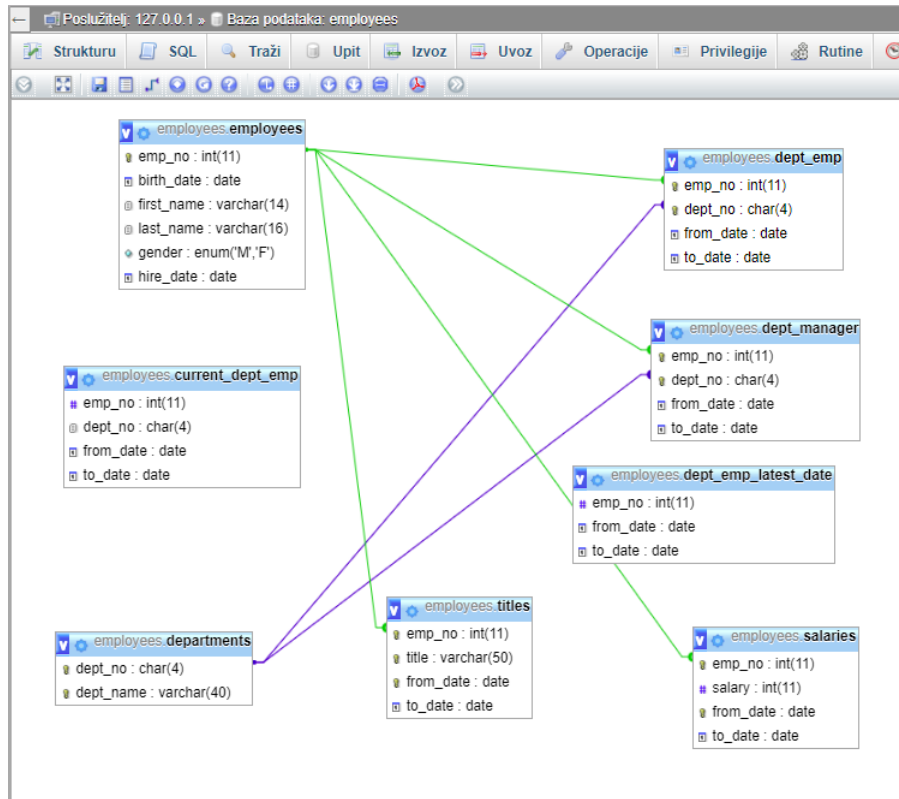
Označi sve    S odabirom:

Prikaz ispisa    Rječnik podataka

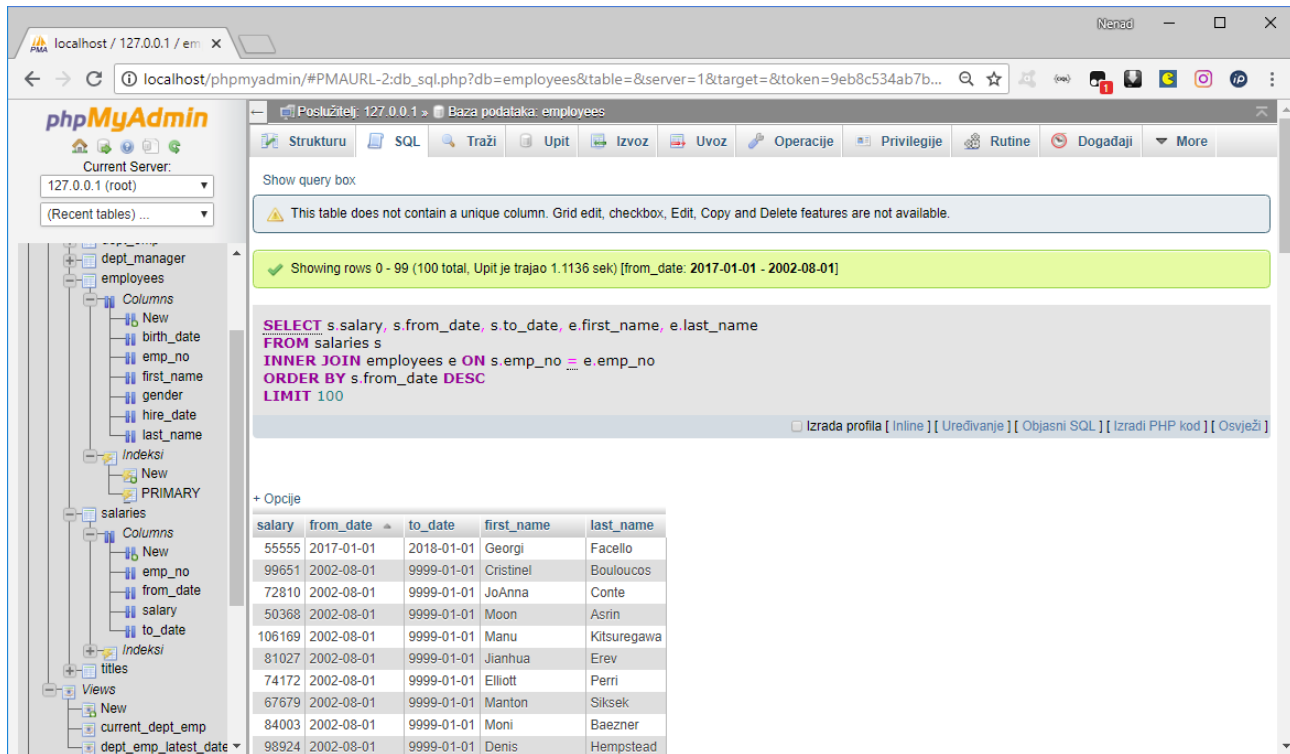
Naziv:     Number of columns:



# Demo baza – pregled strukture



# Demo baza – pregled podataka za primjer



The screenshot shows the phpMyAdmin interface for a MySQL database named 'employees'. The left sidebar displays the database structure, including tables like 'dept\_manager', 'employees', 'salaries', and 'titles'. The main area shows a SQL query and its results.

**SQL Query:**

```
SELECT s.salary, s.from_date, s.to_date, e.first_name, e.last_name
FROM salaries s
INNER JOIN employees e ON s.emp_no = e.emp_no
ORDER BY s.from_date DESC
LIMIT 100
```

**Query Results:**

salary	from_date	to_date	first_name	last_name
55555	2017-01-01	2018-01-01	Georgi	Facello
99651	2002-08-01	9999-01-01	Cristinel	Bouloucos
72810	2002-08-01	9999-01-01	JoAnna	Conte
50368	2002-08-01	9999-01-01	Moon	Asrin
106169	2002-08-01	9999-01-01	Manu	Kitsuregawa
81027	2002-08-01	9999-01-01	Jianhua	Erev
74172	2002-08-01	9999-01-01	Elliott	Perri
67679	2002-08-01	9999-01-01	Manton	Siksek
84003	2002-08-01	9999-01-01	Moni	Baezner
98924	2002-08-01	9999-01-01	Denis	Hempstead



# UseMySQL.php – korištenje baze u PHP-u

```
<?php
```

```
require_once __DIR__ . './autoload.php';
```

```
require_once __DIR__ . './_config.php';
```

```
use GraphQL\Type\Definition\ObjectType;
```

```
use GraphQL\Type\Definition\Type;
```

```
use GraphQL\Type\Schema;
```

```
use GraphQL\GraphQL;
```

```
try {
```



# UseMySQL.php – korištenje baze u PHP-u

```
$queryType = new ObjectType([
    'name' => 'Query',
    'fields' => [
        'echo' => [
            'type' => Type::string(),
            'args' => [
                'emp_no' => ['type' => Type::string()],
            ],
            'resolve' => function ($root, $args) {
                $query = "select s.salary, s.from_date,
                    s.to_date, e.first_name,
                    e.last_name
                    from salaries s
                    inner join employees e on s.emp_no
                    = e.emp_no
                    where s.emp_no = '" . $args['emp_no'] . "'
                    order by s.from_date desc limit 1";
```





# UseMySQL.php – korištenje baze u PHP-u

```
$result = mysql_query($query);
while($rec = mysql_fetch_array($result)) {
    $salary = $rec['salary'];
    $period = $rec['from_date'] . '
              - ' . $rec['to_date'];
    $employee = $rec['first_name'] . ' ' .
                $rec['last_name'];
}

return $root['prefix'] . $salary . '
(' . $period . ') ' . $employee;
    }
    ],
],
]);
```



# UseMySQL.php – korištenje baze u PHP-u

```
$mutationType = new ObjectType([
    'name' => 'Calc',
    'fields' => [
        'add' => [
            'type' => Type::string(),
            'args' => [
                'emp' => ['type' =>
                    Type::string()],
            'from' => ['type' => Type::string()],
            'to' => ['type' =>
                Type::string()],
            'salary' => ['type' =>
                Type::int()],
            ],
        ],
```



# UseMySQL.php – korištenje baze u PHP-u

```
'resolve' => function ($root, $args) {
    $query = "insert into salaries (emp_no,
salary,
                from_date, to_date)
                values ('" . $args['emp'] . "'," .
                $args['salary'] . "," .
                $args['from'] . "','" .
                $args['to'] . "')";
    $result = mysql_query($query);
    if ($result == 1)
        return "Added salary for " . $args['emp'];
    else
        return "Error adding salary";
    },
    ],
    ]);
```



# UseMySQL.php – korištenje baze u PHP-u

```
$schema = new Schema([  
    'query' => $queryType,  
    'mutation' => $mutationType,  
]);
```

```
$rawInput = file_get_contents('php://input');  
$input = json_decode($rawInput, true);  
$query = $input['query'];  
$variableValues = isset($input['variables']) ?  
$input['variables'] : null;
```



# UseMySQL.php – korištenje baze u PHP-u

```
$rootValue = ['prefix' => 'Last salary: '];
$result = GraphQL::executeQuery($schema, $query,
    $rootValue, null, $variableValues);
$output = $result->toArray();
} catch (\Exception $e) {
    $output = [
        'error' => [
            'message' => $e->getMessage()
        ]
    ];
}
header('Content-Type: application/json;
charset=UTF-8');
echo json_encode($output);

?>
```



# config.php – korištenje baze u PHP-u

```
<?php
    $con = mysql_connect('localhost','root','')
        or die(mysql_error());
    if (!$con) {
        echo "Unable to connect to DB: " .
mysql_error();
        exit;
    }
    if (!mysql_select_db("employees")) {
        echo "Unable to select mydbname: " .
mysql_error();
        exit;
    }
    mysql_query("SET NAMES 'utf8'");
    mysql_query("COLLATE 'utf8_general_ci'");
```



# PHP – primjer upita na bazu

## Primjer upita

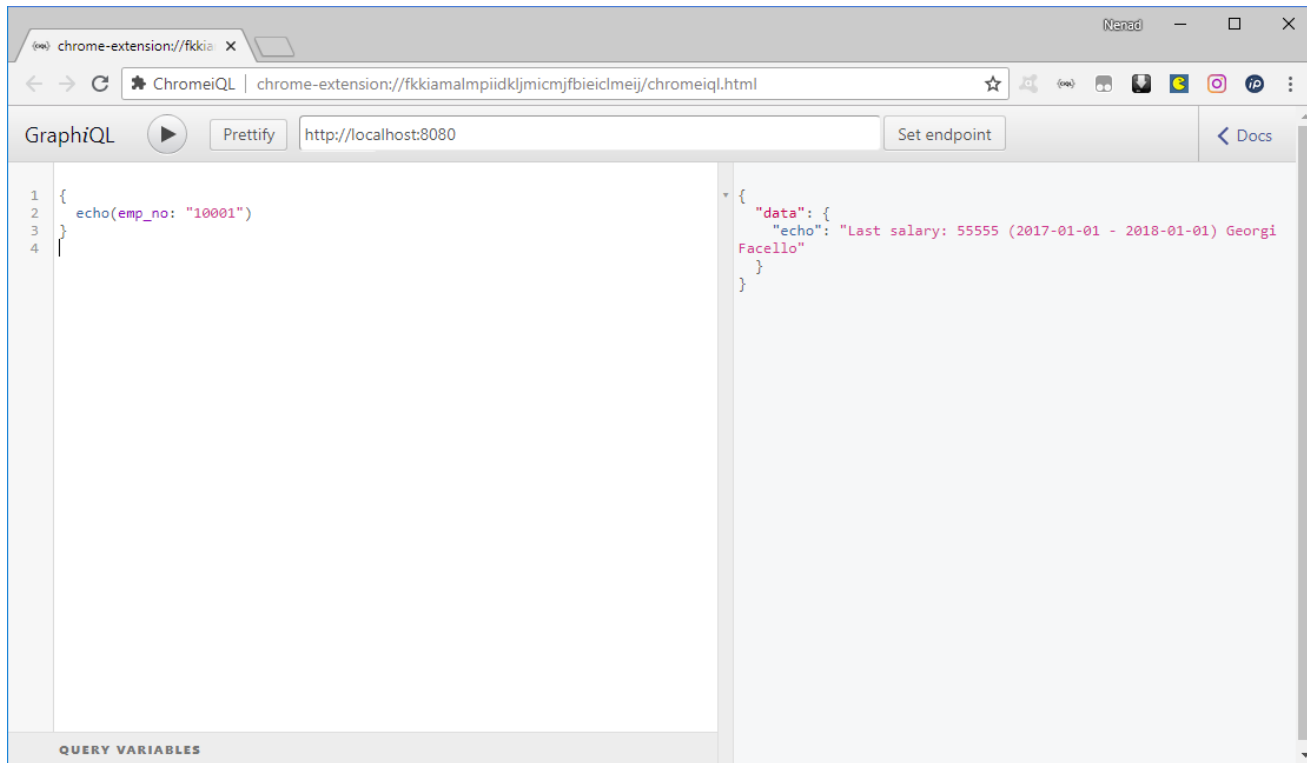
```
{  
    echo (emp_no: "10001")  
}
```

## Rezultat izvođenja upita

```
{  
    "data": {  
        "echo": "Last salary: 55555 (2017-01-01 -  
2018-01-01) Georgi Facello"  
    }  
}
```



# PHP – primjer upita na bazu



The screenshot shows a web browser window with a GraphQL client interface. The browser address bar shows the URL `chrome-extension://fkkiamalmpiidkjmjfbieidmej/chromeiql.html`. The client interface has a header with the text "GraphiQL", a "Prettify" button, an input field containing `http://localhost:8080`, a "Set endpoint" button, and a "Docs" link. The main area is split into two panes. The left pane contains a GraphQL query:

```
1 {
2   echo(emp_no: "10001")
3 }
4 |
```

The right pane shows the JSON response:

```
{
  "data": {
    "echo": "Last salary: 55555 (2017-01-01 - 2018-01-01) Georgi Facello"
  }
}
```

At the bottom left of the interface, there is a section labeled "QUERY VARIABLES".





# PHP – primjer mutacije na bazu

## Primjer mutacije

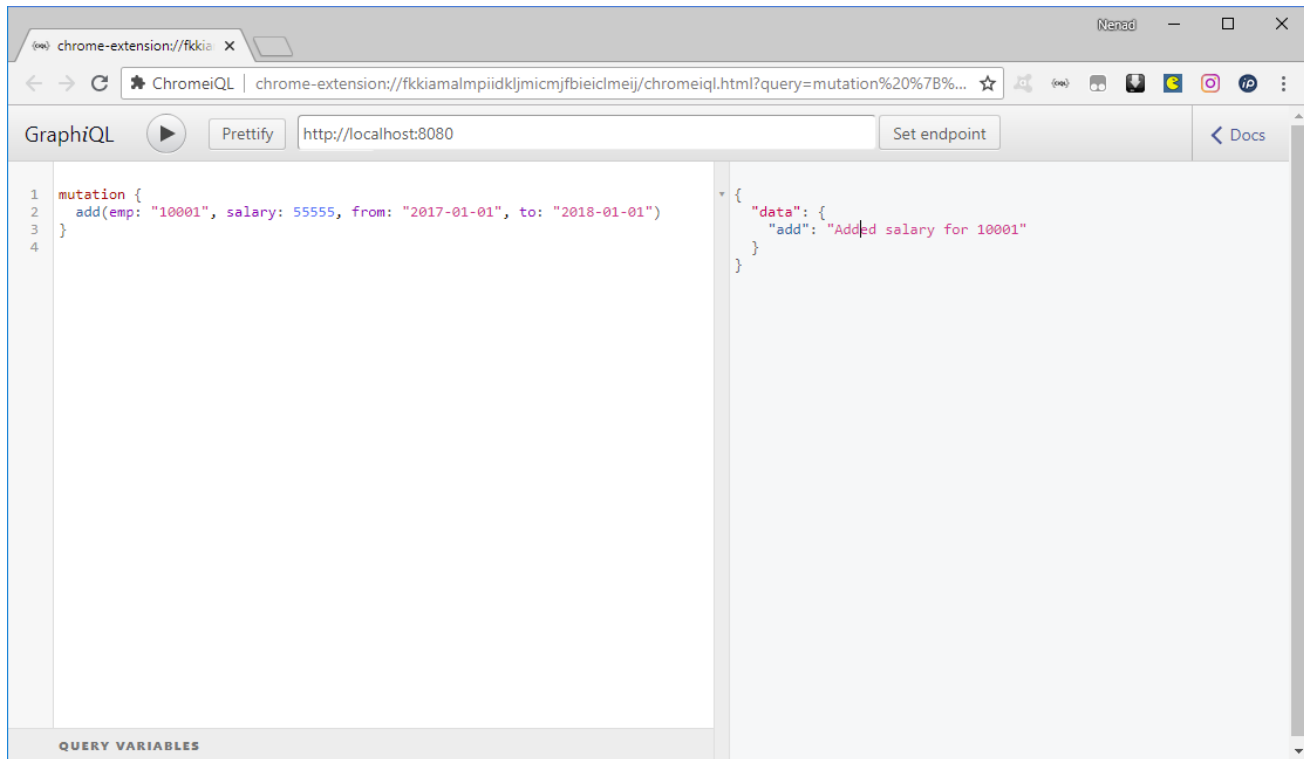
```
mutation {  
  add(emp: "10001", salary: 55555, from: "2017-  
01-01", to: "2018-01-01")  
}
```

## Rezultat izvođenja mutacije

```
{  
  "data": {  
    "add": "Added salary for 10001"  
  }  
}
```



# PHP – primjer mutacije na bazu



The screenshot shows a web browser window with a Chrome extension interface for GraphQL. The browser address bar shows the URL: `chrome-extension://fkkiamalmpiidkijmicmjfbieiclmeij/chromeiq.html?query=mutation%20%7B%20add(emp: "10001", salary: 55555, from: "2017-01-01", to: "2018-01-01")%20%7D`. The GraphQL client interface has a "Prettify" button and a "Set endpoint" button with the URL `http://localhost:8080`. The query editor on the left contains the following GraphQL mutation:

```
1 mutation {  
2   add(emp: "10001", salary: 55555, from: "2017-01-01", to: "2018-01-01")  
3 }  
4
```

The response editor on the right shows the JSON response:

```
{  
  "data": {  
    "add": "Added salary for 10001"  
  }  
}
```

At the bottom left of the interface, there is a section labeled "QUERY VARIABLES".



# PHP – Ponavljanje iste mutacije

## Primjer mutacije

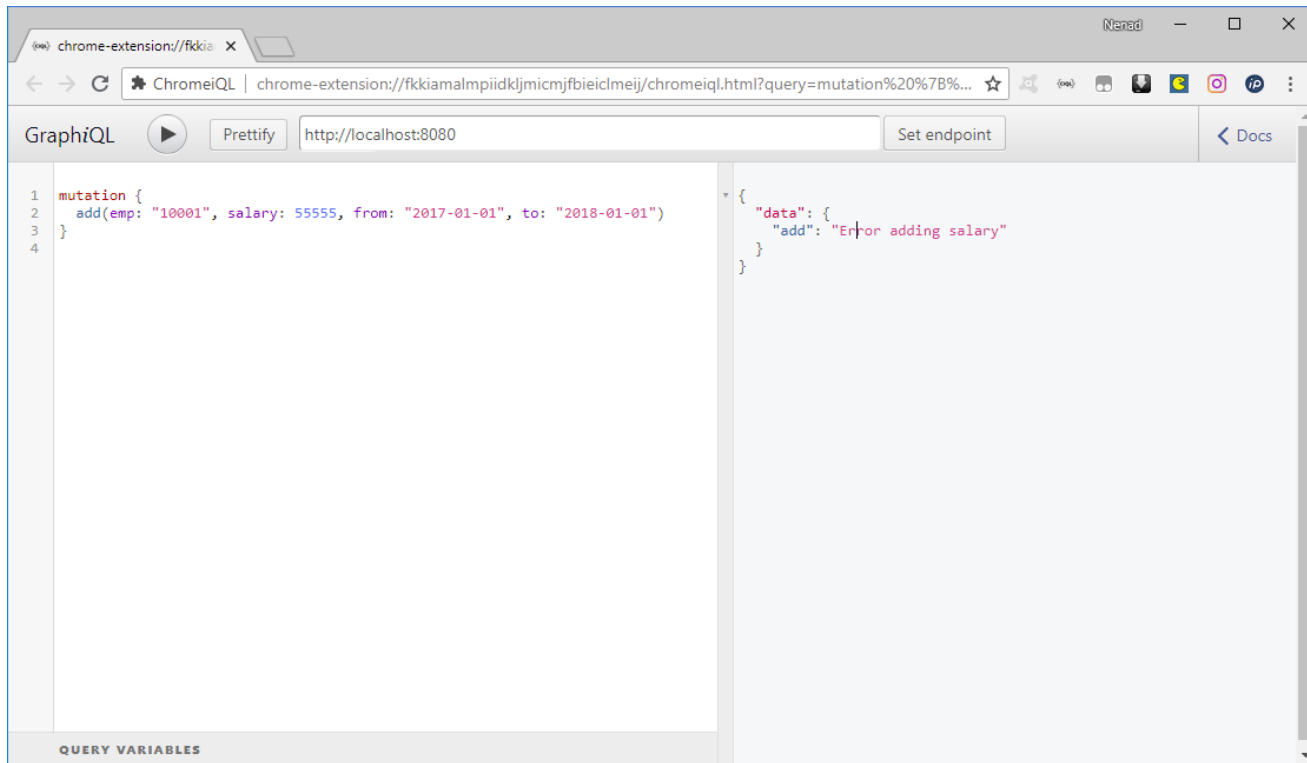
```
mutation {  
  add(emp: "10001", salary: 55555, from: "2017-  
01-01", to: "2018-01-01")  
}
```

## Rezultat izvođenja mutacije

```
{  
  "data": {  
    "add": "Error adding salary"  
  }  
}
```



# PHP – primjer mutacije na bazu



The screenshot shows a web browser window with a GraphQL client interface. The browser's address bar shows the URL: `chrome-extension://fkkiamalmpiidkljmjfbieidmeij/chromeiql.html?query=mutation%20%7B%...`. The GraphQL client interface has a "Prettify" button and a "Set endpoint" button with the value `http://localhost:8080`. The query editor on the left contains the following GraphQL mutation:

```
1 mutation {
2   add(emp: "10001", salary: 55555, from: "2017-01-01", to: "2018-01-01")
3 }
4
```

The response editor on the right shows the following JSON response:

```
{
  "data": {
    "add": "Error adding salary"
  }
}
```

At the bottom of the interface, there is a section labeled "QUERY VARIABLES".



# UseMySQL.php – izmijenjeni primjer

```
'name' => 'Query',  
  'fields' => [  
    'LastSalary' => [  
      'type' => Type::string(),  
      'args' => [  
        'emp_no' => ['type'  
          => Type::string()],  
      ],  
    ],
```



# UseMySQL.php – izmijenjeni primjer

```
'name' => 'Query',  
  'fields' => [  
    'FirstSalary' => [  
      'type' => Type::string(),  
      'args' => [  
        'emp_no' => ['type'  
          => Type::string()],  
      ],  
    ],
```

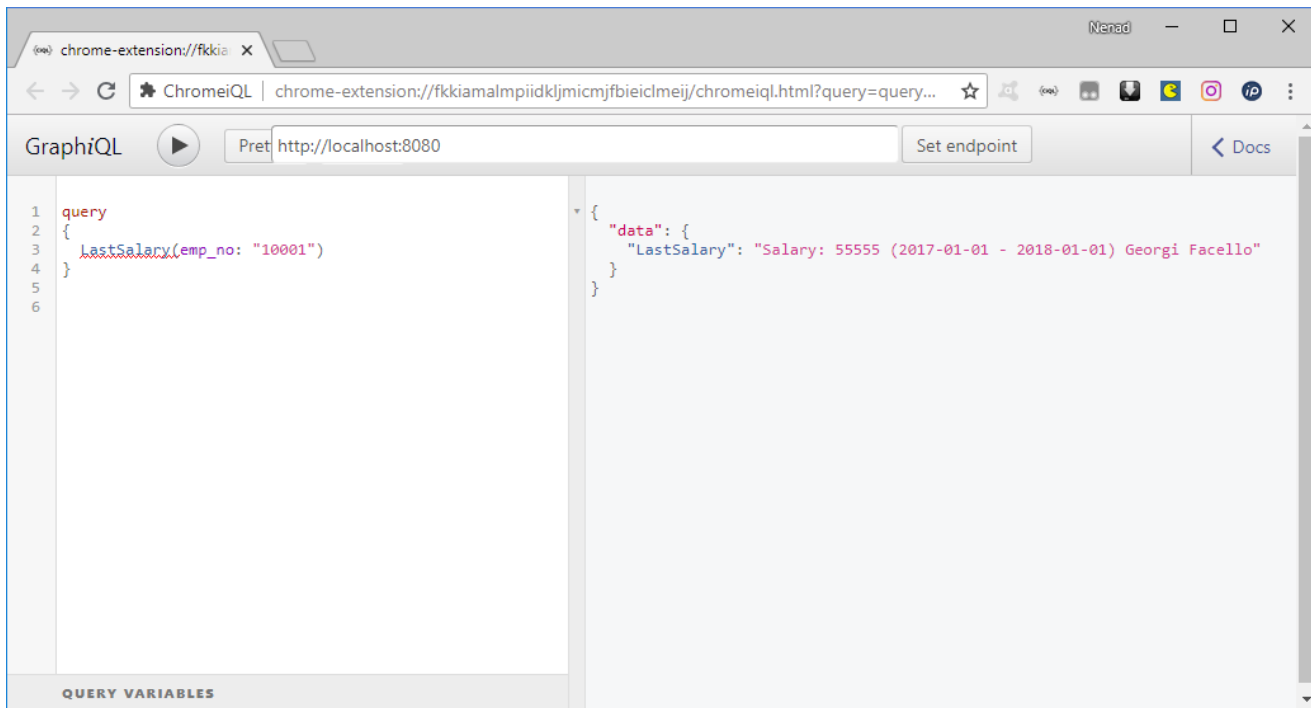


# UseMySQL.php – izmijenjeni primjer

```
$query = "select s.salary, s.from_date,  
s.to_date, e.first_name, e.last_name  
from salaries s  
inner join employees e on s.emp_no = e.emp_no  
where s.emp_no = '" . $args['emp_no'] . "'  
order by s.from_date asc limit 1";
```



# PHP – izvođenje izmijenjenog primjera



The screenshot shows the GraphQL Playground interface. The URL bar indicates the endpoint is `http://localhost:8080`. The query input on the left contains the following GraphQL query:

```
1 query
2 {
3   lastSalary(emp_no: "10001")
4 }
5
6
```

The response on the right is a JSON object:

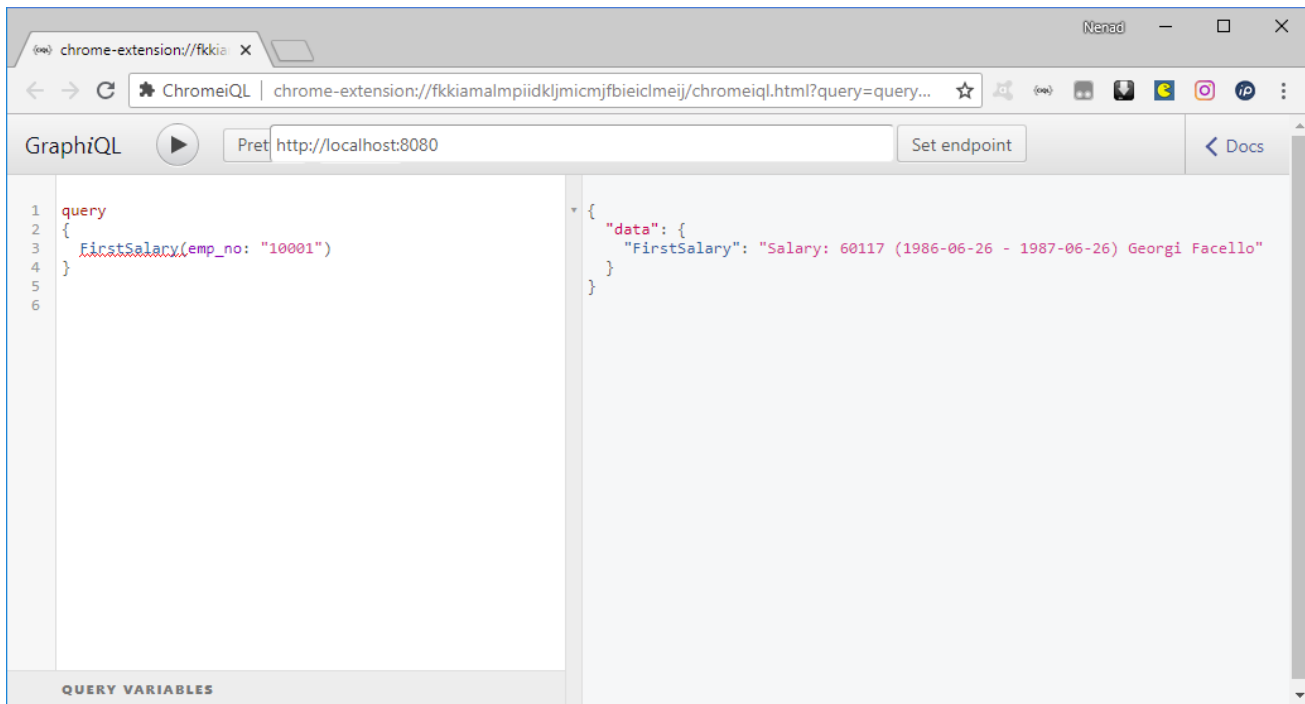
```
{
  "data": {
    "lastSalary": "Salary: 55555 (2017-01-01 - 2018-01-01) Georgi Facello"
  }
}
```

At the bottom left of the interface, there is a section labeled "QUERY VARIABLES".





# PHP – izvođenje izmijenjenog primjera



The screenshot shows a web browser window with a GraphQL IDE interface. The browser's address bar shows the URL: `chrome-extension://fkkiamalpiidkljmjcmjfbieicmeij/chromeiq.html?query=query...`. The IDE has a header with a play button, a text input containing `http://localhost:8080`, and a `Set endpoint` button. Below the header, the left pane contains a GraphQL query:

```
1 query
2 {
3   firstSalary(emp_no: "10001")
4 }
5
6
```

The right pane displays the JSON response:

```
{
  "data": {
    "firstSalary": "Salary: 60117 (1986-06-26 - 1987-06-26) Georgi Facello"
  }
}
```

At the bottom left of the IDE, there is a section labeled `QUERY VARIABLES`.



# Primjeri korištenja na dokumentno orijentiranoj bazi podataka MongoDB



## MongoDB priprema računala

MongoDB je dokumentno (JSON-like) orijentirana baza podataka slobodna za korištenje.

Omogućava korištenje u različitim programskim jezicima.

Instalacija za Windowse preuzima se s adrese

<https://www.mongodb.com/download-center?jmp=nav#community>



# MongoDB – preuzimanje i instalacija

The screenshot shows the MongoDB Download Center website. The browser address bar displays the URL: <https://www.mongodb.com/download-center?jmp=nav#community>. The page features a navigation menu with links for DOCS, LEARN, WHAT'S MONGODB?, and LOGIN. A prominent green button labeled "Get MongoDB" is visible. Below the navigation, a horizontal menu highlights "Community Server" among other options like Atlas, Enterprise Server, Ops Manager, Compass, Connector for BI, and Charts. The main content area is titled "Current Stable Release (4.0.1)" and includes links for "08/03/2018: Release Notes | Changelog" and "Download Source: tgz | zip". It offers download options for Windows, Linux, and OSX. A "Version:" dropdown menu is set to "Windows 64-bit x64". A green "DOWNLOAD (msi)" button is prominently displayed. Below this, there are links for "Binary: Installation Instructions | All Version Binaries". At the bottom of the page, a promotional message reads: "Deploy a free cluster in the cloud with MongoDB Atlas, our database service that gets you up and running in minutes." The browser's status bar at the bottom right shows the IP address 52.206.222.245.



# MongoDB – preuzimanje i instalacija

MongoDB 4.0.1 2008R2Plus SSL (64 bit) Service Customization

### Service Configuration

Specify optional settings to configure MongoDB as a service.

Install MongoD as a Service

Run service as Network Service user

Run service as a local or domain user:

Account Domain:

Account Name:

Account Password:

Service Name:

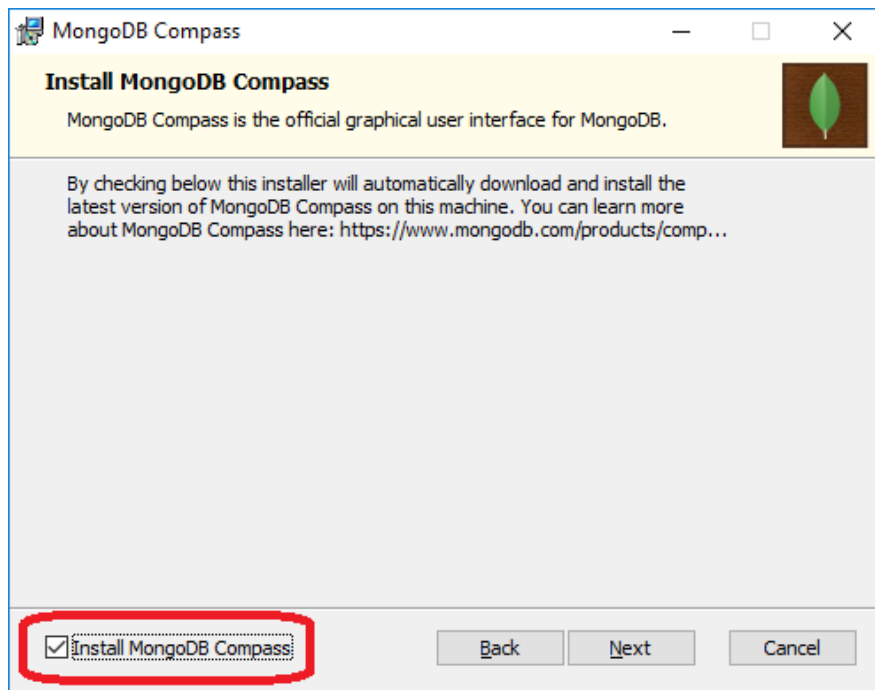
Data Directory:

Log Directory:

< Back    Next >    Cancel



# MongoDB – preuzimanje i instalacija



# MongoDB – uvod u korištenje

Nakon instalacije automatski su pripremljene tri lokalne baze podataka: **admin**, **config** i **local**, te jedna kolekcija dokumenata (**startup\_log** u bazi **local**).

Baza se može koristiti preko grafičkog korisničkog sučelja **MongoDB Compass Community**.

Za to se koristi podrazumijevana veza prema bazi podataka također automatski pripremljena kod instalacije baze podataka.



# MongoDB – uvod u korištenje

MongoDB Compass Community - Connect

Connect View Help

**CREATE FREE ATLAS CLUSTER**  
Includes 512 MB of data storage.  
[Learn more](#)

**New Connection**

★ Favorites

🕒 RECENTS

## Connect to Host

Hostname

Port

SRV Record

Authentication

Replica Set Name

Read Preference

SSL

SSH Tunnel

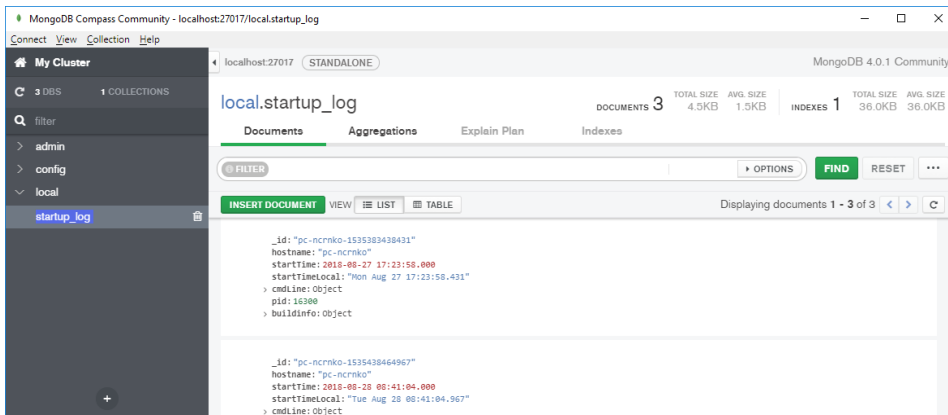
Favorite Name ⓘ

**CONNECT**





# MongoDB – pregled podataka



MongoDB Compass Community - localhost:27017/local.startup\_log

Connect View Collection Help

My Cluster localhost:27017 STANDALONE MongoDB 4.0.1 Community

3 DBS 1 COLLECTIONS

filter

- admin
- config
- local
- startup\_log

local.startup\_log

DOCUMENTS 3 TOTAL SIZE 4.5KB AVG. SIZE 1.5KB INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

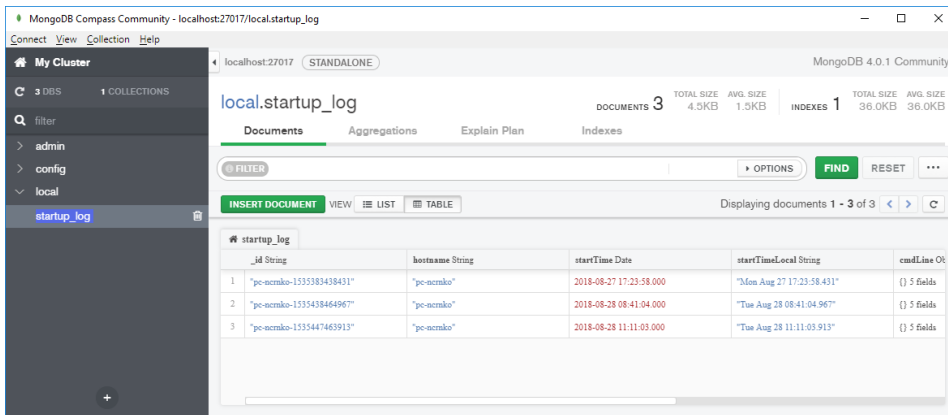
FILTER OPTIONS FIND RESET ...

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 3 of 3

```
{ "_id": "pc-ncrnko-153538438431",
  "hostname": "pc-ncrnko",
  "startTime": 2018-08-27 17:23:58.000,
  "startTimeLocal": "Mon Aug 27 17:23:58.431",
  "cmdLine": Object,
  "pid": 16300,
  "buildInfo": Object }

{ "_id": "pc-ncrnko-1535438464967",
  "hostname": "pc-ncrnko",
  "startTime": 2018-08-28 08:41:04.000,
  "startTimeLocal": "Tue Aug 28 08:41:04.967",
  "cmdLine": Object }
```



MongoDB Compass Community - localhost:27017/local.startup\_log

Connect View Collection Help

My Cluster localhost:27017 STANDALONE MongoDB 4.0.1 Community

3 DBS 1 COLLECTIONS

filter

- admin
- config
- local
- startup\_log

local.startup\_log

DOCUMENTS 3 TOTAL SIZE 4.5KB AVG. SIZE 1.5KB INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

FILTER OPTIONS FIND RESET ...

INSERT DOCUMENT VIEW LIST TABLE

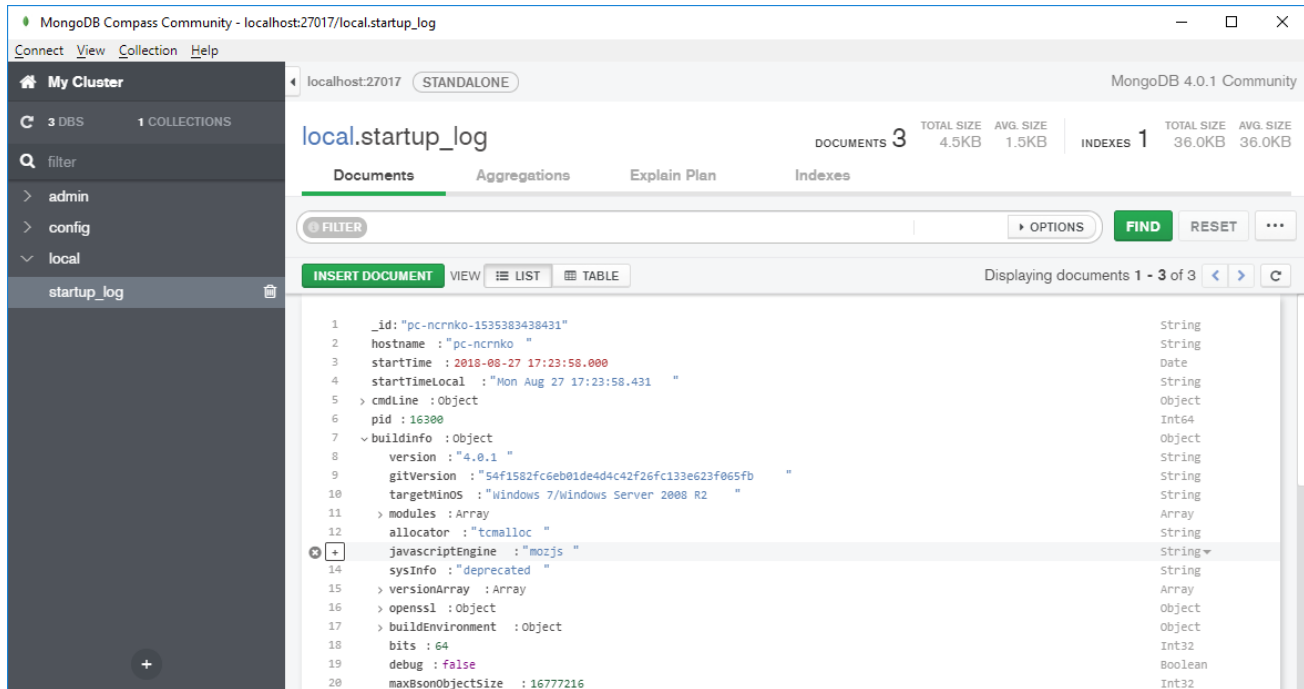
Displaying documents 1 - 3 of 3

startup\_log

	_id String	hostname String	startTime Date	startTimeLocal String	cmdLine Object
1	"pc-ncrnko-153538438431"	"pc-ncrnko"	2018-08-27 17:23:58.000	"Mon Aug 27 17:23:58.431"	{ } 5 fields
2	"pc-ncrnko-1535438464967"	"pc-ncrnko"	2018-08-28 08:41:04.000	"Tue Aug 28 08:41:04.967"	{ } 5 fields
3	"pc-ncrnko-1535447463913"	"pc-ncrnko"	2018-08-28 11:11:03.000	"Tue Aug 28 11:11:03.913"	{ } 5 fields



# MongoDB – unos/ažuriranje podataka



The screenshot shows the MongoDB Compass interface. The left sidebar displays the cluster structure: My Cluster, 3 DBS, 1 COLLECTIONS, and a tree view with folders for admin, config, and local, and a collection named startup\_log. The main area shows the local.startup\_log collection with 3 documents, 4.5KB total size, and 1 index. The document details are displayed in a table view, showing fields like \_id, hostname, startTime, and buildinfo.

Field	Value	Type
1	_id: "pc-ncrnko-1535383438431"	String
2	hostname: "pc-ncrnko "	String
3	startTime: 2018-08-27 17:23:58.000	Date
4	startTimeLocal: "Mon Aug 27 17:23:58.431 "	String
5	cmdLine: Object	Object
6	pid: 16300	Int64
7	buildinfo: Object	Object
8	version: "4.0.1 "	String
9	gitVersion: "54f1582fc6eb01de444c42f26fc133e623f0e5fb "	String
10	targetMinOS: "Windows 7/Windows Server 2008 R2 "	String
11	> modules: Array	Array
12	allocator: "tcmalloc "	String
13	javascriptEngine: "mozjs "	String
14	sysInfo: "deprecated "	String
15	> versionArray: Array	Array
16	> openssl: Object	Object
17	> buildEnvironment: Object	Object
18	bits: 64	Int32
19	debug: false	Boolean
20	maxBsonObjectSize: 16777216	Int32



# MongoDB – uvod u korištenje

Dokumenti u bazi podataka mogu se pregledavati u obliku popisa dokumenata ili tablice.

Različiti dijelovi dokumenta mogu se unositi / ažurirati preko korisničkog sučelja ili programskog koda.

Moguće je uvesti podatke iz drugih baza podataka. Na primjer, MySQL baza podataka employees. Primjer upita za izvoz podataka:

```
select e.emp_no, e.first_name, e.last_name,  
s.salary, s.from_date, s.to_date  
from salaries s  
inner join employees e on s.emp_no = e.emp_n  
order by 1 asc  
limit 100000
```



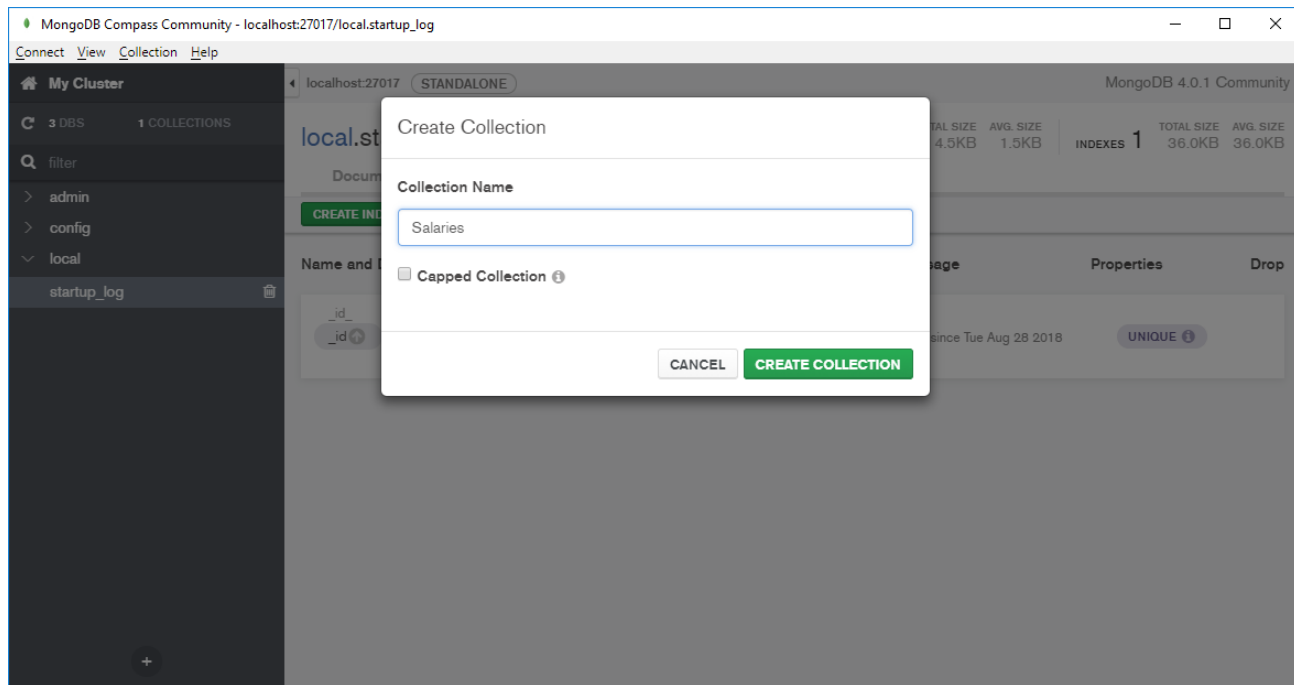
# MongoDB – uvoz podataka u MongoDB

Izvezeni podaci iz relacijske baze podataka pogodni za uvoz u MongoDB:

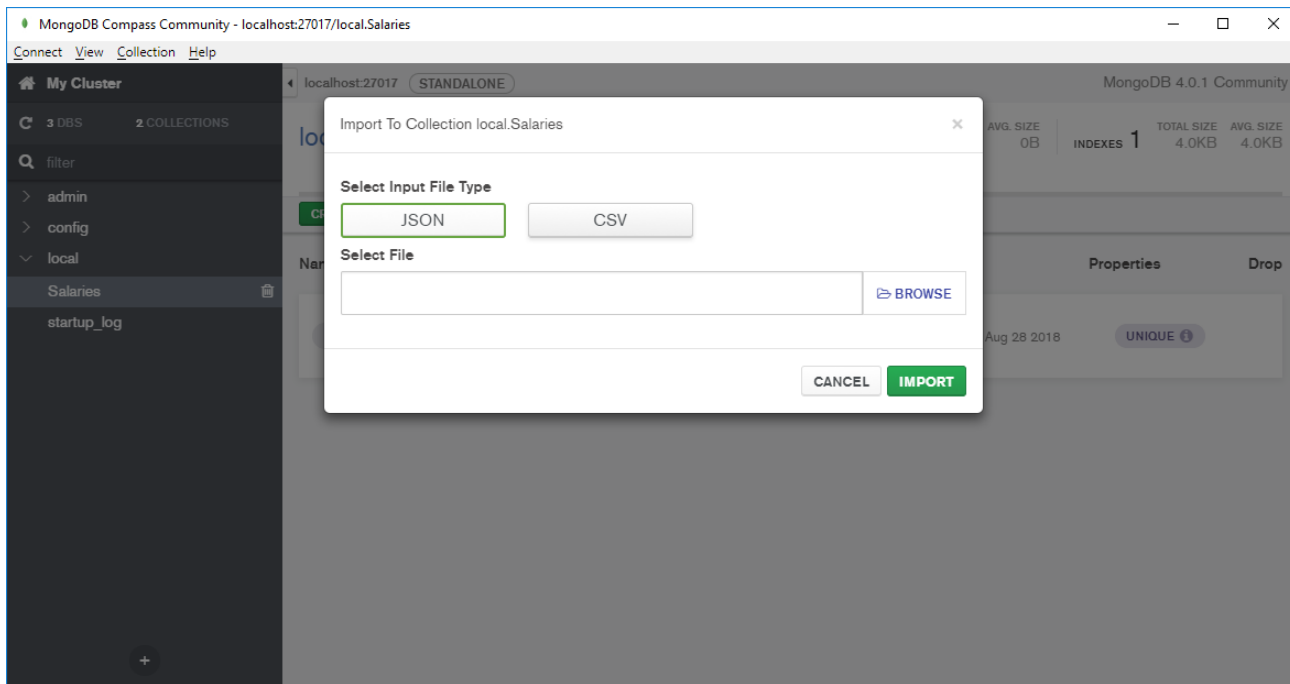
```
emp_no,first_name,last_name,salary,from_date,to_date
10001,Georgi,Facello,60117,1986-06-26,1987-06-26
10001,Georgi,Facello,62102,1987-06-26,1988-06-25
10001,Georgi,Facello,66074,1988-06-25,1989-06-25
10001,Georgi,Facello,66596,1989-06-25
10002,Bezalel,Simmel,65828,1996-08-03,1997-08-03
10002,Bezalel,Simmel,65909,1997-08-03,1998-08-03
10002,Bezalel,Simmel,67534,1998-08-03,1999-08-03
10002,Bezalel,Simmel,69366,1999-08-03,2000-08-02
10002,Bezalel,Simmel,71963,2000-08-02,2001-08-02
10002,Bezalel,Simmel,72527,2001-08-02,9999-01-01
```



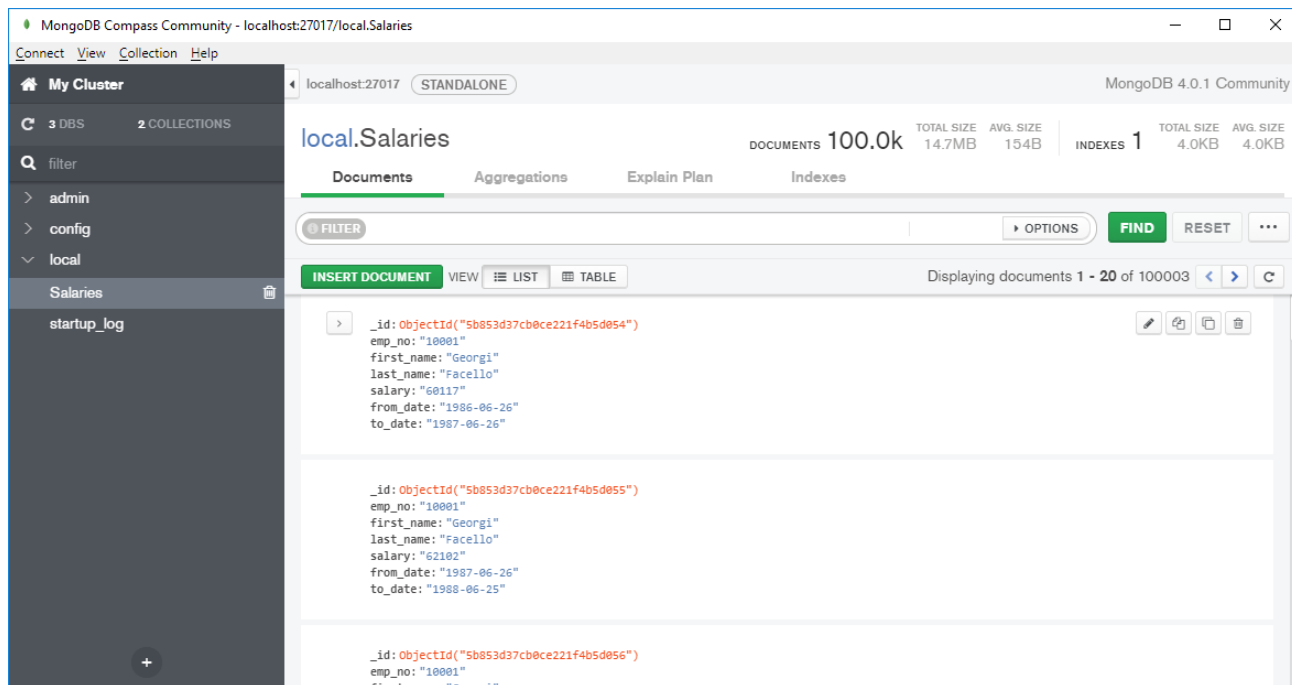
# Korak 1 – priprema nove kolekcije za uvoz



# Korak 2 – odabir polaznog formata datoteke



# Korak 3 – provjera podataka nakon uvoza



MongoDB Compass Community - localhost:27017/local.Salaries

Connect View Collection Help

localhost:27017 (STANDALONE) MongoDB 4.0.1 Community

local.Salaries

DOCUMENTS 100.0k TOTAL SIZE 14.7MB AVG. SIZE 154B INDEXES 1 TOTAL SIZE 4.0KB AVG. SIZE 4.0KB

Documents Aggregations Explain Plan Indexes

FILTER OPTIONS FIND RESET ...

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 20 of 100003

```
> {
  "_id": ObjectId("5b853d37cb0ce221f4b5d054"),
  "emp_no": "10001",
  "first_name": "Georgi",
  "last_name": "Facello",
  "salary": "60117",
  "from_date": "1986-06-26",
  "to_date": "1987-06-26"
}
```

```
{
  "_id": ObjectId("5b853d37cb0ce221f4b5d055"),
  "emp_no": "10001",
  "first_name": "Georgi",
  "last_name": "Facello",
  "salary": "62102",
  "from_date": "1987-06-26",
  "to_date": "1988-06-25"
}
```

```
{
  "_id": ObjectId("5b853d37cb0ce221f4b5d056"),
  "emp_no": "10001",
  "first_name": "Georgi"
}
```



# Mongoose – MongoDB framework

Za korištenje dokumentno orijentirane baze podataka MongoDB uz pomoć jezika upita GraphQL, potreban je još jedan dodatak – Mongoose.

Osim baze podataka MongoDB ma računalu treba biti instaliran i **Node.js**.

Mongoose je dostupan na web adresi:

<https://mongoosejs.com/>





# Mongoose – instalacija

>npm install mongoose

```
C:\WINDOWS\system32\cmd.exe
C:\xampp\htdocs\GraphQLMongo>npm install mongoose
npm WARN htdocs@1.0.0 No repository field.
npm WARN htdocs@1.0.0 No license field.

+ mongoose@5.2.12
added 24 packages in 3.958s

Update available 5.6.0 → 6.4.1
Run npm i npm to update

C:\xampp\htdocs\GraphQLMongo>
```



# MongoDB/Mongoose – GraphQL primjer

```
var express = require('express');
var graphqlHTTP = require('express-graphql');
var { buildSchema } = require('graphql');

var schema = buildSchema(`
  type Query {
    salary: String
  }
`);
```



# MongoDB/Mongoose – GraphQL primjer

```
var root = {
  salary: () => {
    var mongoose = require('mongoose'),
        Schema = mongoose.Schema;
    var db =
mongoose.connect('mongodb://localhost/local'
, { useNewUrlParser: true });

    var Salaries = new Schema({
      emp_no: String,
      first_name: String,
      last_name: String,
      salary: String,
      from_date: String,
      to_date: String
    });
```



# MongoDB/Mongoose – GraphQL primjer

```
var SalaryModel =
mongoose.model('Salaries', Salaries);

var query = SalaryModel.find({ emp_no:
'10001'}, function(err, Salaries)
{}).sort([[ 'from_date', 'ascending' ]]);

return query.then(function (retsalaries) {
  console.log(retsalaries);
  return 'First salary for ' +
retsalaries[0].first_name + " " +
retsalaries[0].last_name + " = " +
retsalaries[0].salary;
});
}
```



# MongoDB/Mongoose – GraphQL primjer

```
var app = express();
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
app.listen(4000);
console.log('Running a GraphQL API server
at localhost:4000/graphql');
```

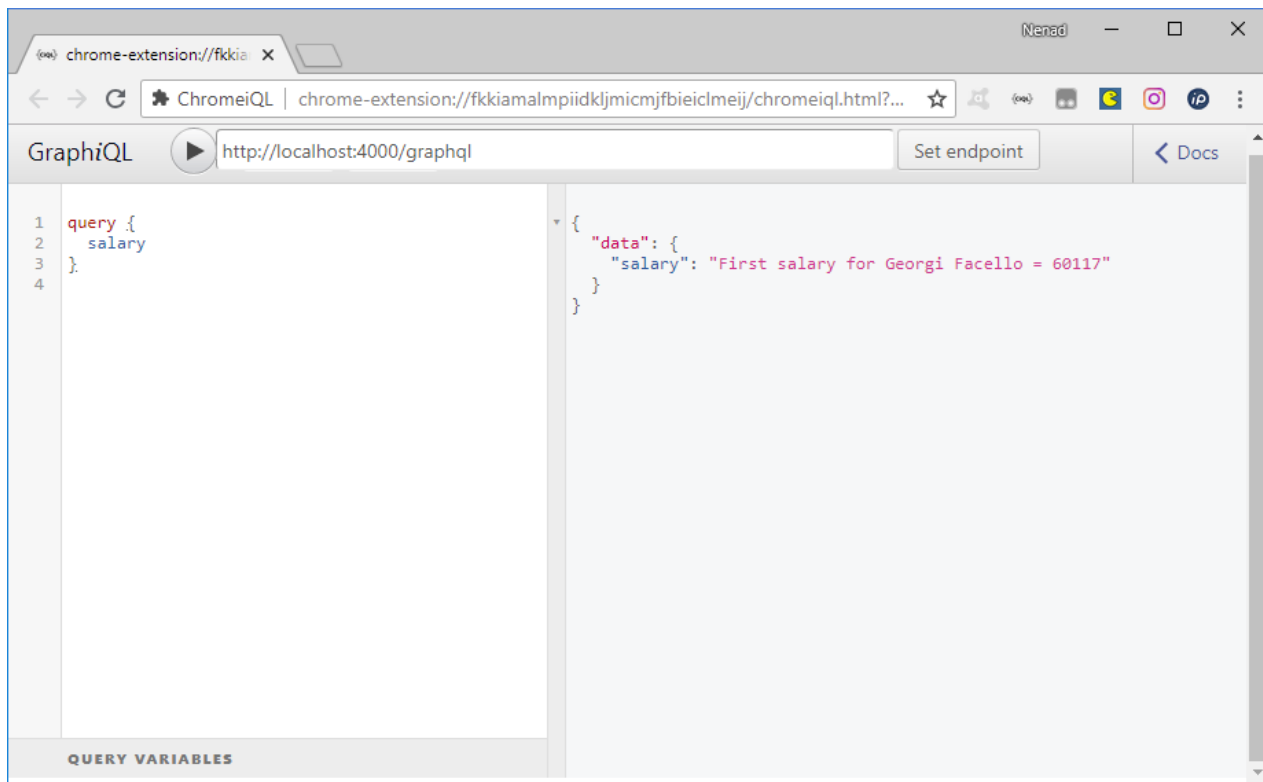


# MongoDB/Mongoose – izvodenje

```
C:\WINDOWS\system32\cmd.exe - node MongoExample.js
C:\xampp\htdocs\GraphQLMongo>node MongoExample.js
Running a GraphQL API server at localhost:4000/graphql
[ { _id: 5b8d5a2f0f185a31288f31dd,
  emp_no: '10001',
  first_name: 'Georgi',
  last_name: 'Facello',
  salary: '60117',
  from_date: '1986-06-26',
  to_date: '1987-06-26',
  __v: 0 } ]
```



# MongoDB/Mongoose – izvodenje



The screenshot shows the ChromeiQL GraphQL client interface. The browser address bar displays the URL `http://localhost:4000/graphql`. The interface is split into two main sections: a query editor on the left and a response viewer on the right.

**Query Editor (Left):**

```
1 query {  
2   salary  
3 }  
4
```

**Response Viewer (Right):**

```
{  
  "data": {  
    "salary": "First salary for Georgi Facello = 60117"  
  }  
}
```

At the bottom left of the interface, there is a section labeled "QUERY VARIABLES".



# MongoDB - primjer koda za ažuriranje baze

```
var SalaryModel =  
mongoose.model('Salaries', Salaries);
```

```
var record = new SalaryModel();  
record.emp_no = '10001';  
record.first_name = 'Georgi';  
record.last_name = 'Facello';  
record.salary = '60117';  
record.from_date = '1986-06-26';  
record.to_date = '1987-06-26';
```

```
record.save(function (err) { });
```





# Sažetak predavanja

Uvod u GraphQL s osnovnim primjerima korištenja neovisno o implementaciji. Dodatni materijali - cjelokupan StarWars primjer dostupan je na adresi:

[https://github.com/graphql/graphql-js/blob/master/src/\\_tests](https://github.com/graphql/graphql-js/blob/master/src/_tests)

Primjeri korištenja u programskom jeziku JavaScript.

Primjeri korištenja na relacijskoj bazi podataka MySQL pomoću programskog jezika PHP.

Primjeri korištenja na dokumentno orijentiranoj bazi podataka MongoDB.



## Komentari i pitanja?



[www.srce.unizg.hr](http://www.srce.unizg.hr)

Ovo djelo je dano na korištenje pod licencom Creative Commons *Imenovanje-Nekomercijalno* 4.0 međunarodna.

[creativecommons.org/licenses/by-nc/4.0/deed.hr](http://creativecommons.org/licenses/by-nc/4.0/deed.hr)

Srce politikom otvorenog pristupa široj javnosti osigurava dostupnost i korištenje svih rezultata rada Srca, a prvenstveno obrazovnih i stručnih informacija i sadržaja nastalih djelovanjem i radom Srca.

[www.srce.unizg.hr/otvoreni-pristup](http://www.srce.unizg.hr/otvoreni-pristup)

