

### **REST vs GraphQL**

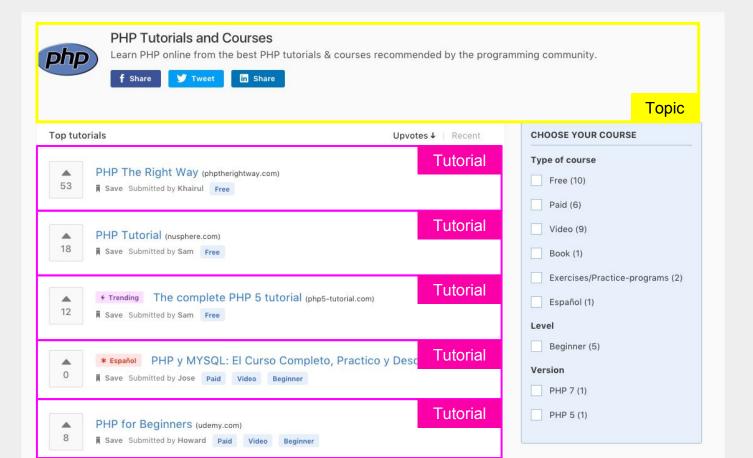
by Pawan Kumar Software Engineer at SquareBoat



# The REST Way



#### **Topic Page**





#### **REST-ful API**

GET - /topics/php

GET - /topics/php/tutorials

GET - /topics/php/related\_tutorials

or

GET - /topics/php?include=tutorials,related\_tutorials

or

GET - /topics/php?include=tutorials(name,description,vote\_count)



#### **REST-ful API Issues**

- Response is Implicit
- Too many Round trips
- No way to get limited fields
- No way to manipulate nested resources



# **GraphQL Way**



### What is GraphQL?

- Query Language
- Spec

### What GraphQL is not?

- A Database
- Library
- Language Specific



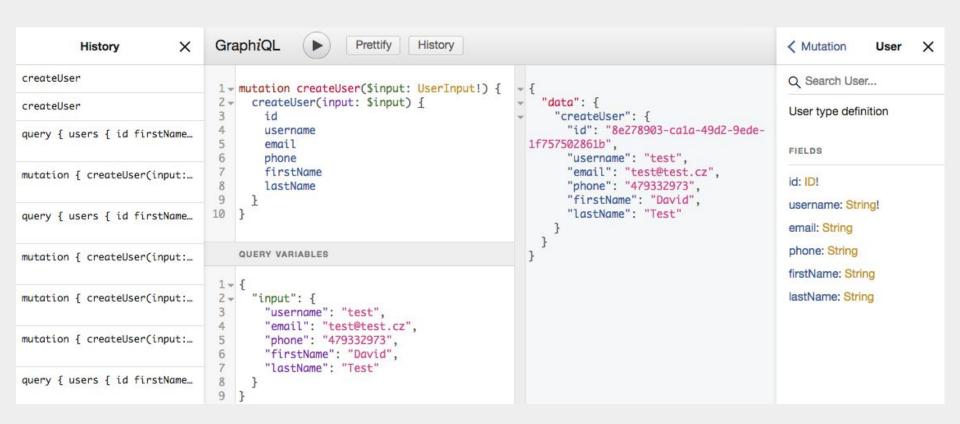
### The Query

```
{
    users {
      id
      name
    }
}
```

```
"users":[
      "id":1,
      "name": "Swapnil Banga"
   },
{
      "id":2,
      "name": "Pawan Kumar"
   },
{
      "id":3,
      "name": "Rajat Patel"
```



#### **GRAPHIQL**





# **Type System**



### **GraphQL Types**

```
type Query {
    users: [User]
    user(id: Int!): User
type Task {
   id: Int!
   title: String
    status: String
    created_at: String
type User {
    id: Int!
    name: String
    created_at: String
    tasks: [Task]
```



### Resolvers



#### Resolvers

```
Query: {
    users: function(root, args, context, info){
      return new User()
        .fetchAll()
        .then(function(users){
          var nodes = users.models.map(function(user){
            return {
                id: this.user.get('id'),
                name: this.user.get('name'),
                created at: this.user.get('created at'),
          });
          return nodes;
});
}
```



## **Mutations**



#### **Mutations**

creating a user

```
input userInput {
  name: String
}

type Mutation {
  createUser (input: userInput): User
}
```

```
Mutation: {
    // create a new user
    createUser(root, { input }) {
        return new User({ name: input.name }).save().then(function(user) {
            var node = new UserTransformer(user).transform();
            return node;
        });
    },
}
```



#### **Mutations**

updating a user

```
type Mutation {
  updateUser (id:Int!, input: userInput): User
}
```

```
Mutation: {
    updateUser(parent, { id, input }) {
        return new User({ id: id }).fetch().then(function(user){
            return user.save(input, { patch: true }).then(function(user) {
                var node = new UserTransformer(user).transform();
                return node;
                });
        });
    })
}
```



## **Validations**



#### **Validations**

```
Mutation: {
 // create a new user
 createUser(parent, { input }) {
   let validation = new Validator(input, {
     name: 'required min:3'
   });
   if(validation.fails()) throw new ValidationException(validation.errors);
   return new User({ name: input.name }).save().then(function(user) {
     var node = new UserTransformer(user).transform();
     return node;
   });
```



### **Formatting errors**

```
routes.use('/graphql', bodyParser.json(), graphqlExpress({
  schema,
 formatError: function(error){
   return {
     message: .get(error, 'message', 'Something went wrong'),
     code: .get(error, 'originalError.code', 400),
     errors: .get(error, 'originalError.errors', [])
  formatResponse: function(response, options){
    if("errors" in response){
     let errors = response.errors;
     response = .omit(response, 'data');
     response = _.omit(response, 'errors');
     response = errors[0];
   return response;
}));
```



### N+1 Problem



### **Dataloader**

https://github.com/facebook/dataloader



## **GraphQL Tools**



https://github.com/apollographql/graphql-tools



# Thank you