

# The Right API for the Job

Rob Allen

PHPDay, May 2022



A close-up photograph of a wooden board game. In the center, a red wooden block with a smooth, rounded top and a slightly irregular, blocky shape sits on a circular hole in the board. The board is made of light-colored wood with a visible grain. Several other identical wooden blocks, which are light-colored, are scattered around the board, some sitting in their respective holes and others on the surface. The lighting is soft, creating gentle shadows. The text "Fit for Purpose" is overlaid in a yellow, serif font across the middle of the image.

# Fit for Purpose



# API Architecture

The background of the slide is a photograph of the Pantheon in Rome at night. The building is illuminated with warm yellow lights, highlighting its massive concrete structure and the portico with its Corinthian columns. The sky is dark blue with some clouds. The title 'API Architecture' is overlaid in a large, yellow, serif font.

M·AGRIPPA·L·F·COS·TERTIUM·PATER



*APIs can be realised in any style  
but, which makes the most sense?*



# RPC APIs

# RPC APIs

- Call a function on a remote server

# RPC APIs

- Call a function on a remote server
- Common implementations: JSON-RPC, SOAP, gRPC

# RPC APIs

- Call a function on a remote server
- Common implementations: JSON-RPC, SOAP, gRPC
- Tends to require a schema (WSDL, ProtoBuf Definition)



# Ethereum JSON-RPC

## Request:

POST / HTTP/1.1

Host: localhost:8545

```
{  
  "jsonrpc": "2.0",  
  "id": 1,  
  "method": "net_peerCount",  
  "params": []  
}
```

# Ethereum JSON-RPC

Response:

```
{  
  "id":1,  
  "jsonrpc": "2.0",  
  "result": "0x2"  
}
```

# gRPC

Interact via PHP library:

```
$client = new RouteGuideClient('localhost:50051');  
  
$p = new RouteGuide\Point();  
$p->setLatitude(409146138);  
$p->setLongitude(-746188906);  
list($feature, $status) = $client->GetFeature($p)->wait();
```



# RESTful APIs

# RESTful APIs

- Operate on a representation of the state of a resource through HTTP verbs



# RESTful APIs

- Operate on a representation of the state of a resource through HTTP verbs
- HTTP native





# RESTful APIs

- Operate on a representation of the state of a resource through HTTP verbs
- HTTP native
- Uniform interface



# RESTful APIs

- Operate on a representation of the state of a resource through HTTP verbs
- HTTP native
- Uniform interface
- Hypermedia controls



# RESTful APIs

PUT /users/ba60c99fd3

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com"  
}
```

# RESTful APIs

PUT /users/ba60c99fd3

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com"  
}
```

# RESTful APIs

PUT /users/ba60c99fd3

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com"  
}
```

# RESTful APIs

HTTP/1.1 201 Created

ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com",  
  "_links": {  
    "self": "https://api.example.com/user/ba60c99fd3"  
  }  
}
```



# RESTful APIs

HTTP/1.1 201 Created

Etag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com",  
  "_links": {  
    "self": "https://api.example.com/user/ba60c99fd3"  
  }  
}
```

# RESTful APIs

HTTP/1.1 201 Created

ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com",  
  "_links": {  
    "self": "https://api.example.com/user/ba60c99fd3"  
  }  
}
```

# RESTful APIs

HTTP/1.1 201 Created

ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

Content-Type: application/json

```
{  
  "name": "Rob Allen"  
  "email": "rob@akrabat.com",  
  "_links": {  
    "self": "https://api.example.com/user/ba60c99fd3"  
  }  
}
```



# GraphQL APIs

# GraphQL APIs

- Retrieve only the data you need on consumer side

# GraphQL APIs

- Retrieve only the data you need on consumer side
- Reduce the number of calls to retrieve data with embedded resources



# GraphQL APIs

- Retrieve only the data you need on consumer side
- Reduce the number of calls to retrieve data with embedded resources
- Self-describing schema



# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```



# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```

# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```

# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```

# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```

# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```

# Queries

```
query {  
  author(name: "Ann McCaffrey") {  
    id, name  
    books(first: 5) {  
      totalCount  
      edges {  
        node {  
          id, title, datePublished  
        }  
      }  
    }  
  }  
}
```



# Queries

```
"data": {  
  "author": {  
    "id": "MXxBdXRob3J8ZjA",  
    "name": "Ann McCaffrey",  
    "books": {  
      "totalCount": 6,  
      "edges": [  
        {  
          "node": {  
            "id": "MXxCb29rfGYwNzU",  
            "title": "Dragonflight"  
          }  
        }  
      ],  
    },  
  },  
}
```

# Queries

```
"data": {  
  "author": {  
    "id": "MXxBdXRob3J8ZjA",  
    "name": "Ann McCaffrey",  
    "books": {  
      "totalCount": 6,  
      "edges": [  
        {  
          "node": {  
            "id": "MXxCb29rfGYwNzU",  
            "title": "Dragonflight"  
          }  
        }  
      ],  
    },  
  },  
}
```

# Queries

```
"data": {  
  "author": {  
    "id": "MXxBdXRob3J8ZjA",  
    "name": "Ann McCaffrey",  
    "books": {  
      "totalCount": 6,  
      "edges": [  
        {  
          "node": {  
            "id": "MXxCb29rfGYwNzU",  
            "title": "Dragonflight"  
          }  
        }  
      ],  
    },  
  },  
}
```

# Queries

```
"data": {  
  "author": {  
    "id": "MXxBdXRob3J8ZjA",  
    "name": "Ann McCaffrey",  
    "books": {  
      "totalCount": 6,  
      "edges": [  
        {  
          "node": {  
            "id": "MXxCb29rfGYwNzU",  
            "title": "Dragonflight"  
          }  
        },  
      ],  
    },  
  },  
}
```

# Mutations

```
mutation {  
  createAuthor(  
    name:"Mary Shelley", dateOfBirth: "1797-08-30"  
  ) {  
    returning {  
      id, name  
    }  
  }  
}
```

# Mutations

```
mutation {  
  createAuthor(  
    name:"Mary Shelley", dateOfBirth: "1797-08-30"  
  ) {  
    returning {  
      id, name  
    }  
  }  
}
```

# Mutations

```
mutation {  
  createAuthor(  
    name: "Mary Shelley", dateOfBirth: "1797-08-30"  
  ) {  
    returning {  
      id, name  
    }  
  }  
}
```

# Mutations

```
mutation {  
  createAuthor(  
    name:"Mary Shelley", dateOfBirth: "1797-08-30"  
  ) {  
    returning {  
      id, name  
    }  
  }  
}
```



# Mutations

Response:

```
"data": {  
  "createAuthor": {  
    "returning": [  
      {  
        "id": "e3388cbea4e840a",  
        "name": "Mary Shelly",  
      }  
    ]  
  }  
}
```





Which to pick?

Rob Allen ~ @akrabat



Lamborghini or Ferrari?





Lamborghini or Truck?



# Considerations

- What is it to be used for?
- Response customisation requirements
- HTTP interoperability requirements
- Binary protocol?

# Response customisation

- GraphQL is a query-first language
- REST tends towards less customisation
- With RPC you get what you're given!

(None will fix your database layer's ability to efficiently retrieve the data requested!)



# Performance

- REST and RPC puts server performance first
- GraphQL puts client performance first



# Caching

- GraphQL and RPC can only cache at application layer
- REST can additionally cache at HTTP layer



# Data Transfer

## GraphQL:

```
query {  
  avatar(userId: "1234")  
}  
  
{  
  "data": {  
    "avatar": "(base64 data)"  
    "format": "image/jpeg"  
  }  
}}
```

## RPC:

```
POST /api  
{  
  "method": "getAvatar",  
  "userId": "1234"  
}  
  
{  
  "result": "(base64 data)"  
}
```

# Data Transfer

REST:

```
GET /user/1234/avatar  
Accept: image/jpeg
```

```
HTTP/1.1 200 OK  
{jpg image data}
```

REST:

```
GET /user/1234/avatar  
Accept: application/json
```

```
HTTP/1.1 200 OK  
{"data": "(base64 data)"}
```



# Versioning

- RPC, GraphQL and REST can all version via evolution as easily as each other



# Versioning

- RPC, GraphQL and REST can all version via evolution as easily as each other
- GraphQL is very good for deprecation of specific fields



# Design considerations

It's *always* hard!



# Design considerations

It's *always* hard!



# It's your choice



# Developer Experience



$$R = \frac{ad^2}{dx}$$

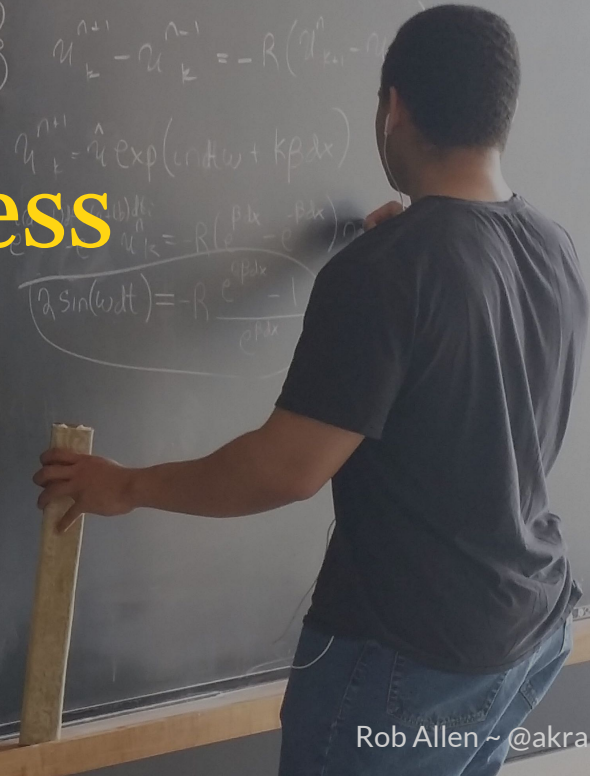
$$u_{k+1}^n = u_k^n + \frac{R}{2} \{ u_k^n - 2u_{k-1}^n + u_{k-2}^n \} - \frac{R}{2} \{ u_k^n - 4u_{k-1}^n + u_{k-2}^n \}$$

$$u(k\Delta x, n\Delta t) = u_k^n$$

$$\frac{\partial u}{\partial t} dt + \frac{\partial^2 u}{\partial t^2} \frac{dt^2}{2} + \frac{\partial^3 u}{\partial t^3} \frac{dt^3}{3!} + \frac{\partial^4 u}{\partial t^4} \frac{dt^4}{4!} + O(dt^5) =$$

$$\frac{\partial}{\partial x} \left\{ \frac{\partial^2 u}{\partial x^2} \frac{dx^2}{2!} + \frac{\partial^3 u}{\partial x^3} \frac{dx^3}{3!} (-6) + \frac{\partial^4 u}{\partial x^4} \frac{dx^4}{4!} (12) \right\} - \frac{R}{2} \left\{ \frac{\partial u}{\partial x} (dx) 2 + \frac{\partial^2 u}{\partial x^2} \frac{dx^2}{2!} (-6) + \frac{\partial^3 u}{\partial x^3} \frac{dx^3}{3!} (12) \right\} + \frac{\partial^4 u}{\partial x^4} \frac{dx^4}{4!} \Rightarrow$$

# Correctness



# Correctness

*RPC: Functions!*



# Correctness

*RPC*: Functions!

*REST*: HTTP matters!



# Correctness

*RPC*: Functions!

*REST*: HTTP matters!

*GraphQL*: Think in terms of relationships!



# Correctness

*RPC*: Functions!

*REST*: HTTP matters!

*GraphQL*: Think in terms of relationships!



# Errors



# Errors

Error representations must be first class citizens



# Errors

Error representations must be first class citizens





# Documentation

A young woman with long brown hair is sitting and reading a large, open book with a green cover. She is wearing a black top and a necklace. The background is a library or study with bookshelves and warm lighting. The word "Documentation" is overlaid in yellow text.

# Documentation

- API Reference



# Documentation

- API Reference
- Tutorials



# To sum up



*If you suck at providing a REST API,  
you will suck at providing a GraphQL API*

Arnaud Lauret, API Handyman



# Thank you!

<https://joind.in/talk/8cdd9>



# Photo credits

- Architecture: <https://www.flickr.com/photos/shawnstilwell/4335732627>
- Choose Pill: <https://www.flickr.com/photos/eclib/4905907267>
- Lamborghini & Ferrari: <https://akrab.at/3w0yFmg>
- Lamborghini & Truck: <https://akrab.at/3F4kAZk>
- '50s Computer: <https://www.flickr.com/photos/9479603@N02/49755349401>
- Blackboard: <https://www.flickr.com/photos/bryanalexander/17182506391>
- Crash Test: <https://www.flickr.com/photos/astrablog/4133302216>