

One Year GraphQL in Production

Radoslav Stankov

23/05/2018

Radoslav Stankov

@rstankov

blog.rstankov.com

github.com/rstankov

twitter.com/rstankov





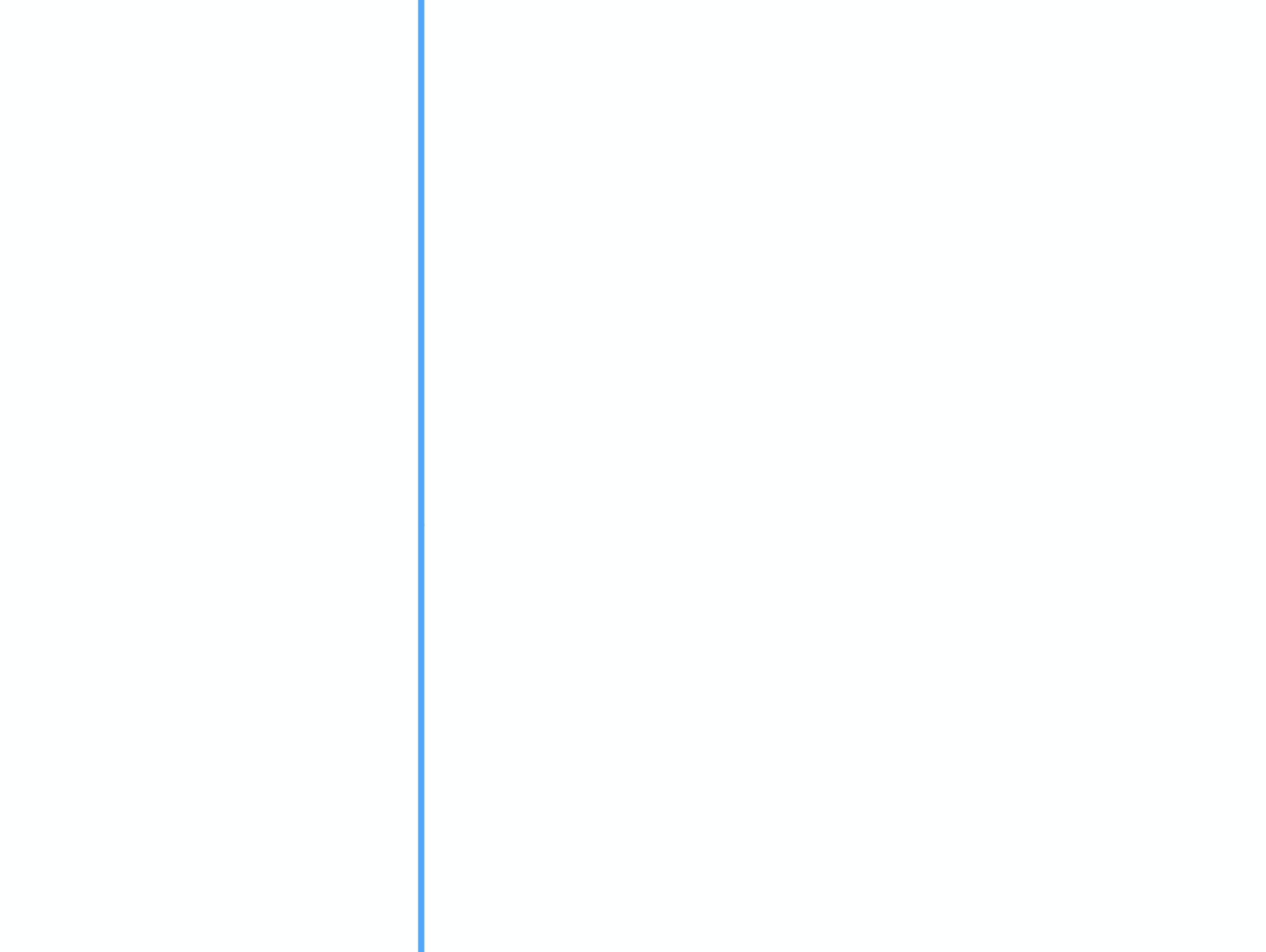
IE's
m



<https://speakerdeck.com/rstankov/one-year-graphql-in-production>



Product Hunt



early 2014

jQuery spaghetti

early 2014

October 2014

jQuery spaghetti
Backbone

early 2014

October 2014

February 2015

jQuery spaghetti

Backbone

React & Rails

early 2014

October 2014

February 2015

May 2015

jQuery spaghetti

Backbone

React & Rails

custom **Flux**

early 2014

jQuery spaghetti

October 2014

Backbone

February 2015

React & Rails

May 2015

custom **Flux**

December 2015

Redux

October 2014

Backbone

February 2015

React & Rails

May 2015

custom **Flux**

December 2015

Redux

January 2016

React-Router

February 2015

React & Rails

May 2015

custom Flux

December 2015

Redux

January 2016

React-Router

April 2016

Redux Ducks

December 2015

Redux

January 2016

React-Router

April 2016

Redux Ducks

Febuary 2017

GraphQL

early 2014

jQuery spaghetti

October 2014

Backbone

February 2015

React & Rails

May 2015

custom **Flux**

December 2015

Redux

January 2016

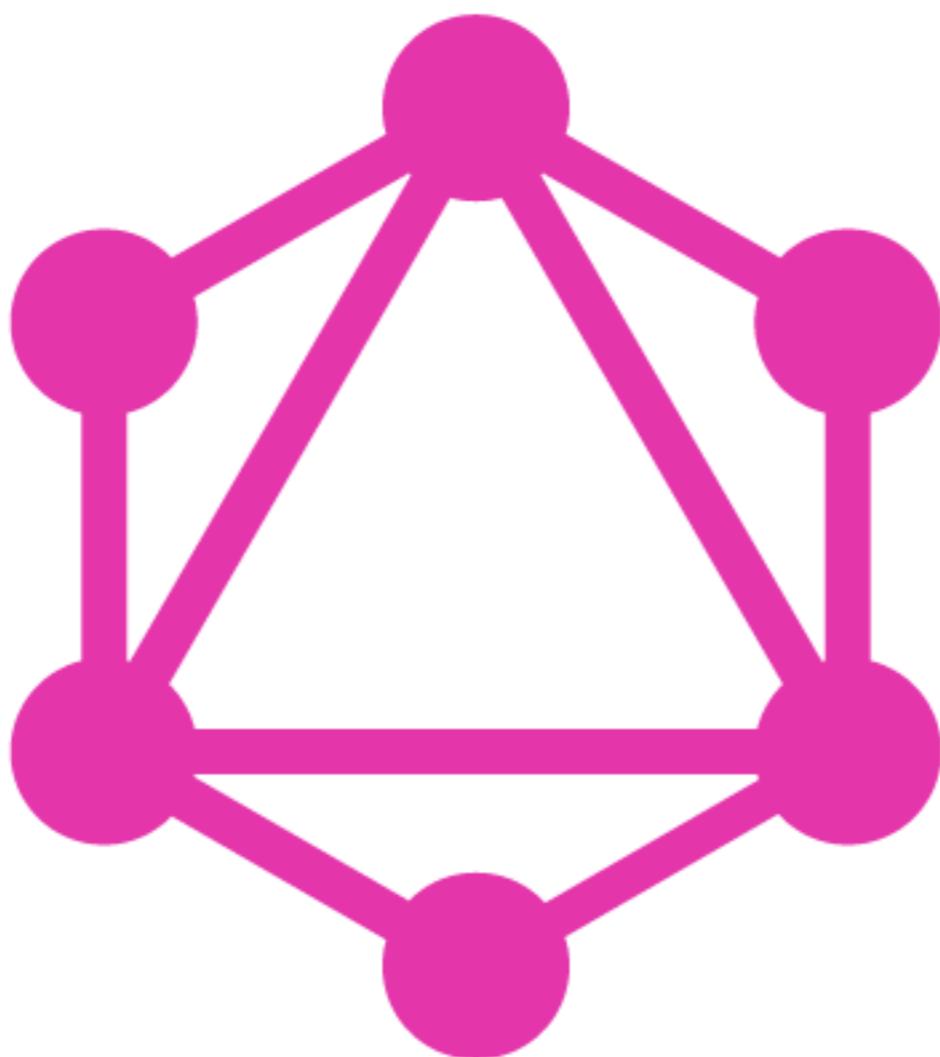
React-Router

April 2016

Redux Ducks

February 2017

GraphQL



<http://graphql.org/>



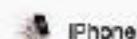
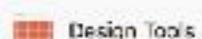
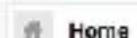
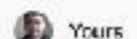
Discover your next favorite thing...

Ask Ship Makers Jobs ...



FREDS

Customize



Kevin Lou · Founder, Anytable

What are good alternatives to UserTesting?

I'm trying to run user research tests for a mobile app, and wanted to see what everyone else is using to do app user research.

TOP RECOMMENDATIONS

Feedback Tools Curated directory of tools for getting all the feedback ▲ 6

1 RECOMMENDED

Today

POPULAR NEWEST

**Screely**

Instantly turn screenshots into beautiful images

DESIGN TOOLS + 2

▲ 381 ▪ 29

**Visual Studio Live Share**

Real-time collaborative development

DEVELOPER TOOLS

▲ 300 ▪ 11

**Hive**

Supercharge your teamwork with analytics

PRODUCTIVITY + 3

▲ 457 ▪ 2

**CodementorX 2.0**

Hire world-class freelance developers for your team

PROMOTED

▲ 718 ▪ 6

**Hack Your Blog to the Top 5 in Google**

Authoritative & actionable blog marketing guide

SMM + 4

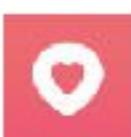
▲ 237 ▪ 10

**Contractbook**

Efficient contract management, automated through Zapier

PRODUCTIVITY + 5

▲ 130 ▪ 11

**Now**

The future of online dating, Now.

PHONE + 3

▲ 133 ▪ 9

Coinbase Institutional

Coinbase launched 2
NEW products

Get the best new products in
your inbox, every day

rstankov@gmail.com

SUBSCRIBE



\$7,500

awsactivate

SIGNUP NOW

Teams hiring now

GustoPayroll, Benefits, and HR
levels by your whole team **Credit Karma**Your credit scores should
be free. And now they are **InVision**The digital product design
platform **Creative Market**The world's marketplace
for design **HubSpot**Building the #1 trusted
marketing and CRM
software **Omni**An operating system for
your things

Today



Prisma

Build a GraphQL server with any database

DEVELOPER TOOLS

▲ 992

47



Coinbase Prime

A professional trading platform for cryptocurrencies

FINTECH + 3

▲ 494

15



Surface Hub 2

Meet Surface Hub 2, the future of collaboration

PRODUCTIVITY + 1

▲ 417

16



Google One

One simple way to get more out of Google

WEB APP + 2

▲ 407

6



Weekly UX Exercise

Receive challenges top companies use to interview designers

DESIGN TOOLS + 2

▲ 440

13

Today



Prisma

Build a GraphQL server with any database

DEVELOPER TOOLS

▲ 982

47

coinbase
Prime

Coinbase Prime

A professional trading platform for cryptocurrencies

FINTECH

+ 3

▲ 494

15



Surface Hub 2

Meet Surface Hub 2, the future of collaboration

PRODUCTIVITY

+ 1

▲ 417

16



Google One

One simple way to get more out of Google

WEB APP

+ 2

▲ 407

6



Weekly UX Exercise

Receive challenges top companies use to interview designers

DESIGN TOOLS

+ 2

▲ 440

11

```
posts {  
  id  
  name  
  tagline  
  votesCount  
  commentsCount  
  topics {  
    id  
    name  
  }  
  isVoted  
}
```

Today

 Prisma
Build a GraphQL server with any database
DEVELOPER TOOLS ▲ 982 47

 Coinbase Prime
A professional trading platform for cryptocurrencies
FINTECH + 3 ▲ 494 15

 Surface Hub 2
Meet Surface Hub 2, the future of collaboration
PRODUCTIVITY + 1 ▲ 417 16

 Google One
One simple way to get more out of Google
WEB APP + 2 ▲ 407 6

 Weekly UX Exercise
Receive challenges top companies use to interview designers
DESIGN TOOLS + 2 ▲ 440 11

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```

Today

 Prisma
Build a GraphQL server with any database
DEVELOPER TOOLS ▲ 982 47

 Coinbase Prime
A professional trading platform for cryptocurrencies
FINTECH + 3 ▲ 494 15

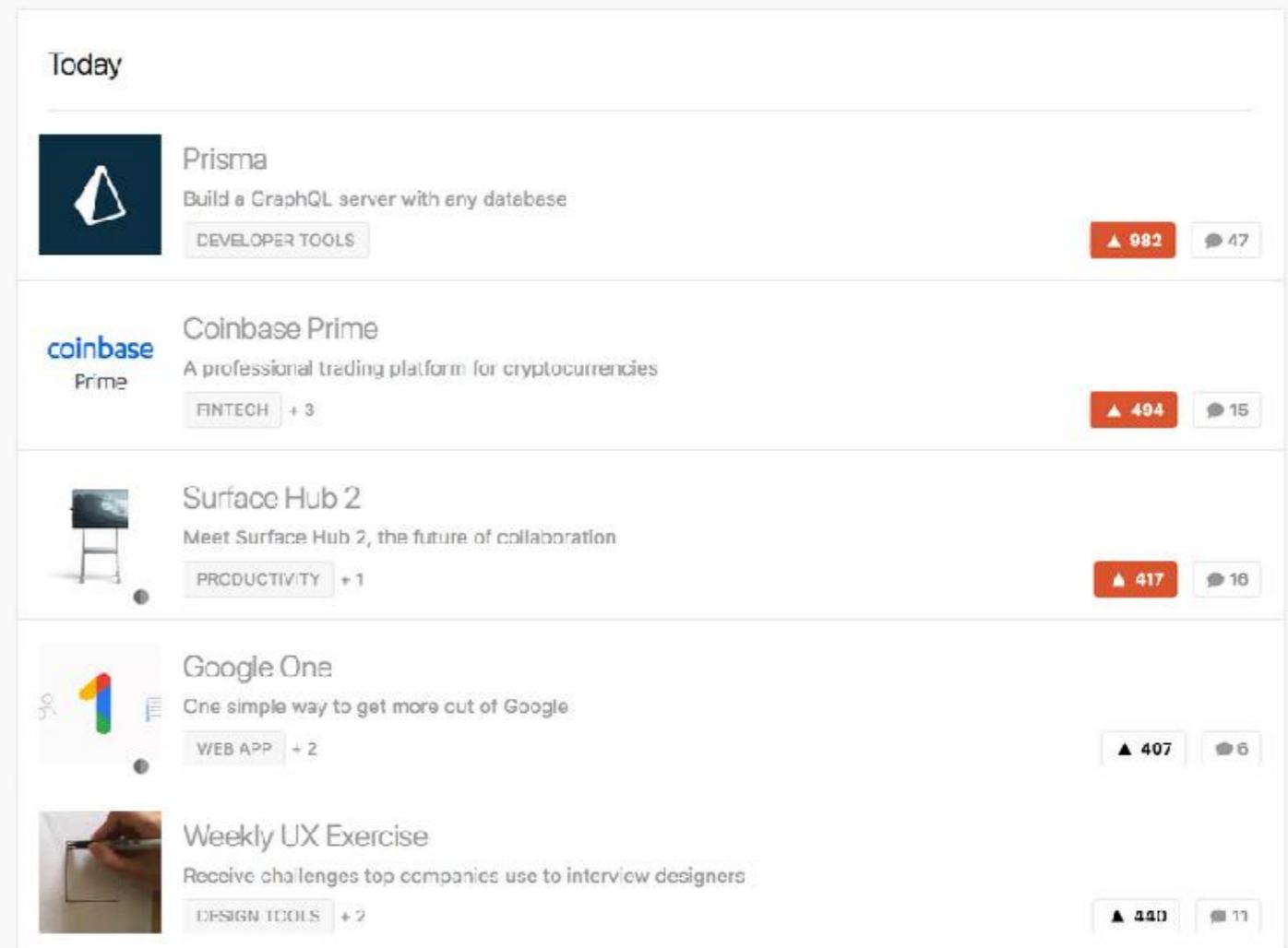
 Surface Hub 2
Meet Surface Hub 2, the future of collaboration
PRODUCTIVITY + 1 ▲ 417 16

 Google One
One simple way to get more out of Google
WEB APP + 2 ▲ 407 6

 Weekly UX Exercise
Receive challenges top companies use to interview designers
DESIGN TOOLS + 2 ▲ 440 11

POST /graphql

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```



POST /graphql

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```

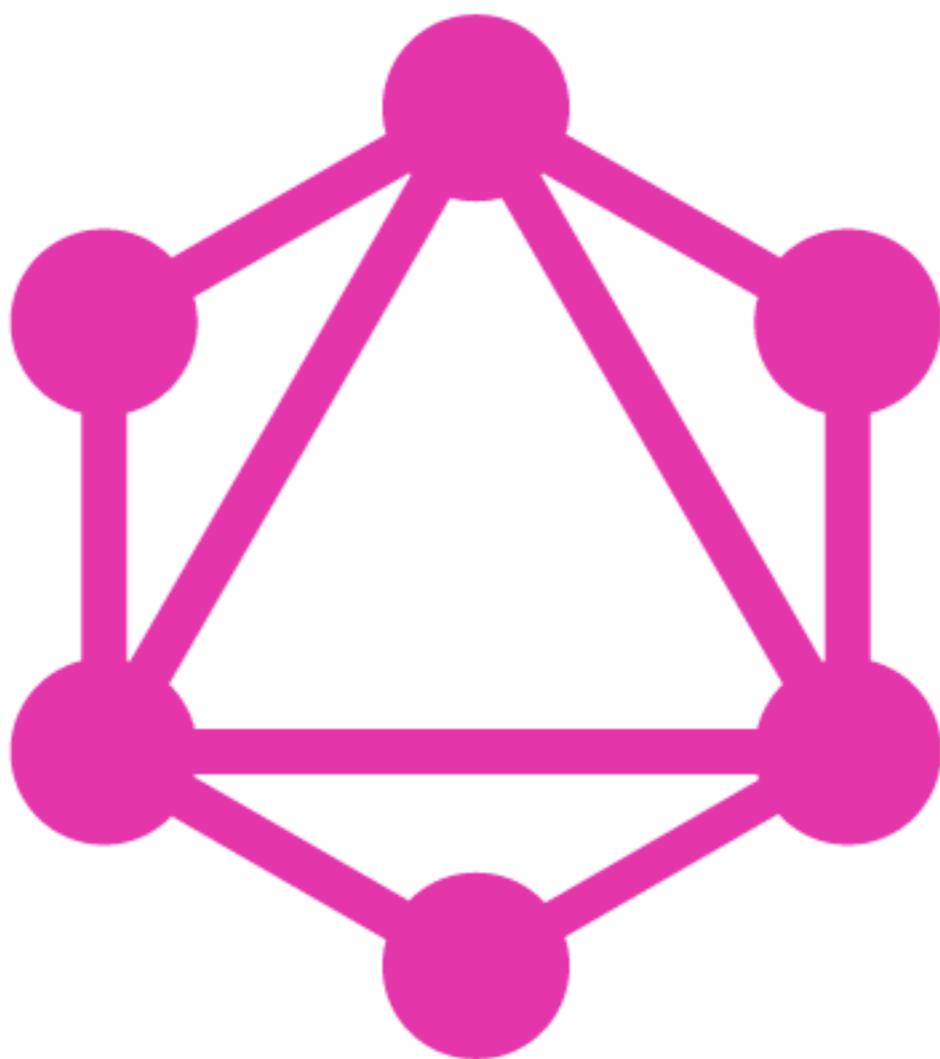


POST /graphql

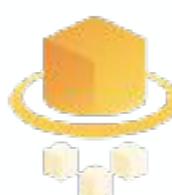
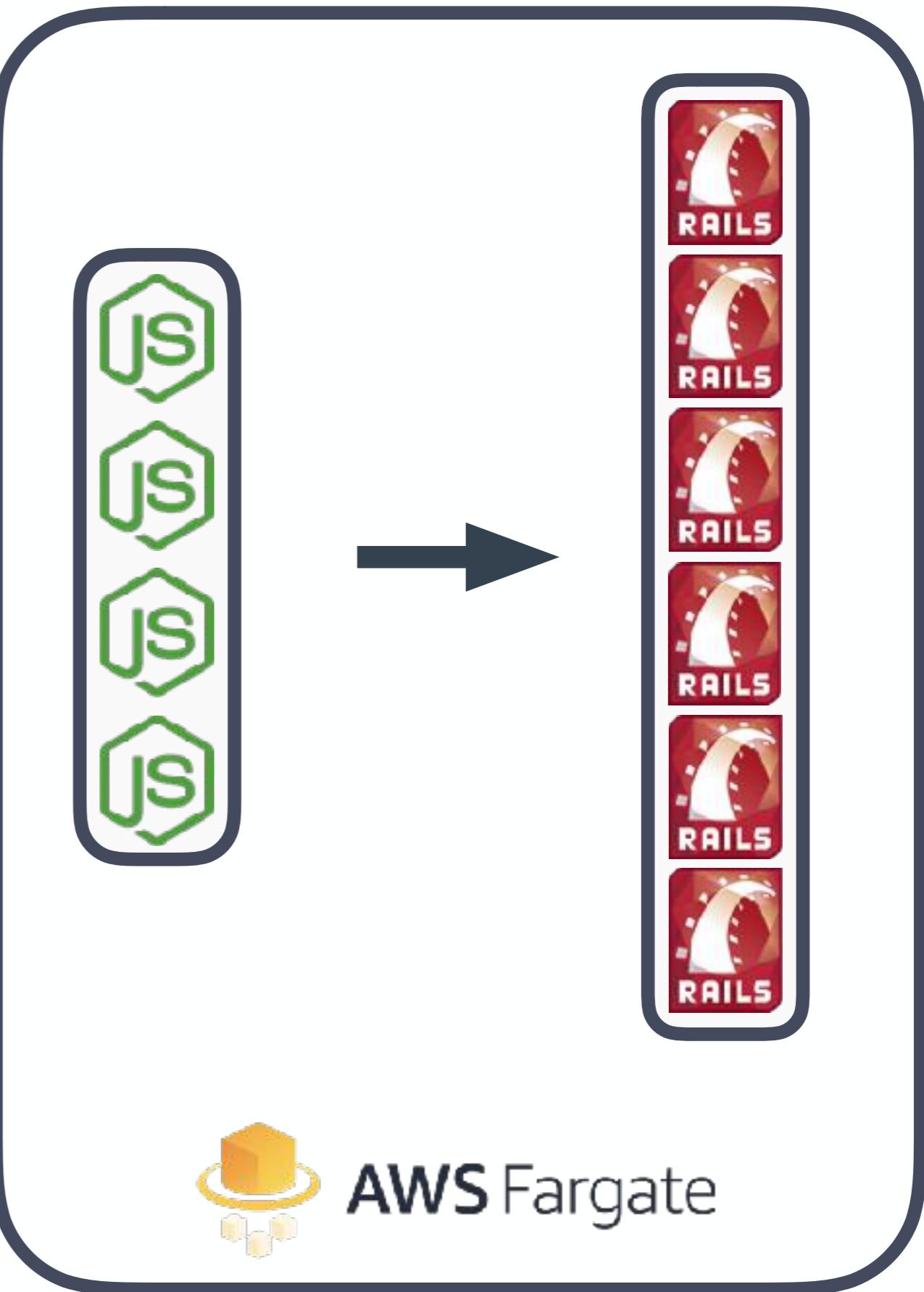
```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```



```
{  
  "data": {  
    "posts": {  
      "id": 1,  
      "name": "Google Duplex",  
      "tagline": "An AI assistant that can ta",  
      "votesCount": 2843,  
      "commentsCount": 45,  
      "topics": [{  
        "id": 1,  
        "name": "AI",  
      }],  
      "isVoted": true,  
    }  
  }  
}
```



<http://graphql.org/>



AWS Fargate



gem "graphql"

```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
  }
}
```

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
  }  
}
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do  
  name 'Post'  
  
  field :id, !types.ID  
  field :name, !types.String  
  field :tagline, !types.String  
end
```

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
  }  
}
```

```
Graph::Types::Query = GraphQL::ObjectType.define do  
  name 'Query'  
  
  field :post, !types[!Graph::Types::PostType] do  
    argument :date, !types.String  
    resolve -> (_obj, args, _ctx) {  
      Post.for_date(args[:date])  
    }  
  end  
end
```

```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
  }
}

Graph::Schema = GraphQL::Schema.define do
  query Graph::Types::Query
end
```

```
class GraphqlController < Frontend::BaseController
  def index
    render json: Graph::Schema.execute(query, variables: variables, context: context)
  end

  private

  def query
    params[:query]
  end

  def context
  {
    current_user: current_user,
  }
  end

  def variables
    convert_to_hash params[:variables]
  end

  def convert_to_hash(variables)
    # ...
  end
end
```

```
Rails.application.routes.draw do
  post '/graphql' => 'graphql#index', defaults: { format: :json }

  # ...
end
```

```
gem "graphql-rails"
```

```
Rails.application.routes.draw do
  post '/graphql' => 'graphql#index', defaults: { format: :json }

  mount GraphiQL::Rails::Engine, graphql_path: '/graphql', at: '/graphiql' if Rails.env.development?

# ...
end
```

GraphQL

Not Secure | ph.test:5000/graphiql?query=query%20%B%0A%20%20post(slug%3A%20"product-hunt")%20%B%0A%20%20%20%20%20%

GraphiQL

Prettify History

Query Post

Search Post...

No Description

IMPLEMENTATIONS

Node

Commentable

MetaTaggable

Topicable

Shareable

Reviewable

Votable

HasEmbeds

FIELDS

_id: ID!

alternatives{
first: Int
after: String
last: Int
before: String
}: PostConnection

alternatives_count: Int!

angellist_url: String

badges{
first: Int
after: String
last: Int
before: String

1 `query {`
2 `post(slug: "product-hunt") {`
3 `id`
4 `name`
5 `tagline`
6 `}`
7 `}`

}
 }
}
}

QUERY VARIABLES

GraphQL

Not Secure | ph.test:5000/graphiql?query=query%20%0A%20%20post(slug%3A%20"product-hunt")%20%0A%20%20%20%2...

GraphiQL

Prettify History

Query Post

Search Post...

No Description

IMPLEMENTATIONS

- Node
- Commentable
- MetaTaggable
- Topicable
- Shareable
- Reviewable
- Votable
- HasEmbeds

FIELDS

- _id: ID!
- alternatives{
 - first: Int
 - after: String
 - last: Int
 - before: String}
- PostConnection
- alternatives_count: Int!
- angellist_url: String
- badges{
 - first: Int
 - after: String
 - last: Int
 - before: String}

1 query {
2 post(slug: "product-hunt") {
3 id
4 name
5 tagline
6 }
7 }

1 {
2 "data": [
3 "post": {
4 "id": "9",
5 "name": "Product Hunt",
6 "tagline": "Discover your next favorite thing 🎉"
7 }
8]
9 }

Query

QUERY VARIABLES

GraphQL

Not Secure | ph.test:5000/graphiql?query=query%20%B%0A%20%20post(slug%3A%20"product-hunt")%20%B%0A%20%20%20%20%20%

GraphiQL

Prettify History

Query Post

Search Post...

No Description

IMPLEMENTATIONS

- Node
- Commentable
- MetaTaggable
- Topicable
- Shareable
- Reviewable
- Votable
- HasEmbeds

FIELDS

_id: ID!

alternatives{
first: Int
after: String
last: Int
before: String
}: PostConnection

alternatives_count: Int!

angellist_url: String

badges{
first: Int
after: String
last: Int
before: String

```
1 query {  
2   post(slug: "product-hunt") {  
3     id  
4     name  
5     tagline  
6   }  
7 }
```

```
1 {  
2   "data": [  
3     "post": {  
4       "id": "9",  
5       "name": "Product Hunt",  
6       "tagline": "Discover your next favorite thing 🎉"  
7     }  
8   ]  
9 }
```

Result

QUERY VARIABLES

The screenshot shows a GraphQL playground interface with the following details:

Left Panel (Query Editor):

```
1 query {
2   post(slug: "product-hunt") {
3     id
4     name
5     tagline
6   }
7 }
```

Right Panel (Results):

```
1 { data: [ pcst: { id: "9", name: "Product Hunt", tagline: "Discover your next favorite thing 🎉" } ] }
```

Bottom Left (Query Variables):

There are no visible query variables.

Bottom Right (Documentation Overlay):

A light green overlay provides documentation for the **Post** type:

- No Description**
- IMPLEMENTS**
 - Node
 - Commentable
 - MetaTaggable
 - Topicable
 - Shareable
 - Reviewable
 - Votable
 - HasEmbeds
- FIELDS**
 - _id: ID!**
 - alternatives(**
 - first: Int**
 - after: String**
 - last: Int**
 - before: String****): PostConnection**
 - alternatives_count: Int!**
 - angellist_uri: String**
 - badges(**
 - first: Int**
 - after: String**
 - last: Int**
 - before: String****): BadgeConnection**

Central Content Area:

Automatic Documentation

A screenshot of a GraphQL playground interface. The left panel shows a code editor with a query:

```
query {
  post(slug: "product-hunt") {
    id
    name
    tagline
  }
}
```

The right panel shows the resulting JSON response:

```
{
  "data": [
    {
      "post": {
        "id": "9",
        "name": "Product Hunt",
        "tagline": "Discover your next favorite thing 😎"
      }
    }
  ]
}
```

The right panel also contains a sidebar with documentation for the `Post` type:

- Search Post...
- No Description
- IMPLEMENTATION
- Node
- Commentable
- MetaTaggable
- Topicable
- Shareable
- Reviewable
- Votable
- HasEmbeds

FIELDS

- `id: ID!`
- `alternatives{
 first: Int
 after: String
 last: Int
 before: String
}: PostConnection`
- `alternatives_count: Int!`
- `angellist_url: String`
- `badges{
 first: Int
 after: String
 last: Int
 before: String`

QUERY VARIABLES

GraphQL Benefits

(backend developer edition)

 simplifies communication with frontend developers

 no extra controllers

 no extra routes

 build-in data serialization

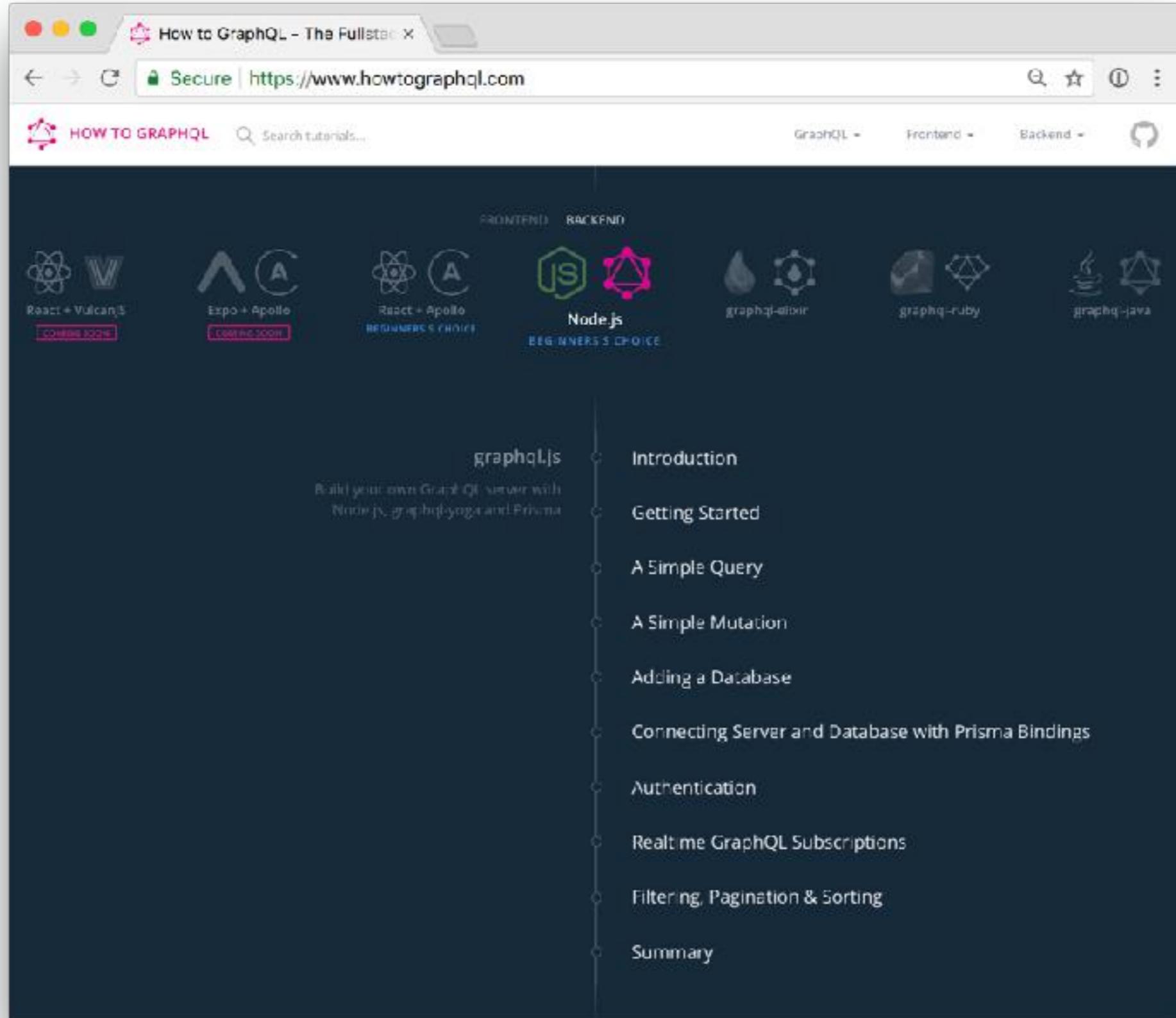
 build-in documentation

 ecosystem of tools

The screenshot shows a web browser window for the URL <https://www.howtographql.com/>. The page displays a navigation bar with links for "GraphQL", "Frontend", and "Backend". Below the navigation is a search bar and a "HOW TO GRAPHQL" logo. The main content area features several sections: "React + Vulcan.js" (coming soon), "Expo + Apollo" (coming soon), "React + Apollo" (beginner's choice), "Node.js" (beginner's choice), "graphql-elixir", and "graphql-ruby". A large central section is titled "React + Apollo" and describes it as "Learn how to get started with React and Apollo Client on the frontend". To the right of this text is a vertical list of ten tutorial chapters:

- Introduction
- Getting Started
- Queries: Loading Links
- Mutations: Creating Links
- Routing
- Authentication
- More Mutations and Updating the Store
- Filtering: Searching the List of Links
- Realtime Updates with GraphQL Subscriptions
- Pagination
- Summary

<https://www.howtographql.com/>



<https://www.howtographql.com/>

The screenshot shows a web browser window with the title "Building a GraphQL Server with Ruby" and the URL "https://www.howtographql.com/graphql-ruby/0-introduction/". The page content is about GraphQL fundamentals, specifically the introduction chapter. It features a sidebar with navigation links for "GRAPHQL FUNDAMENTALS" and "GRAPHQL RUBY TUTORIAL".

GRAPHQL FUNDAMENTALS

- Introduction
- GraphQL is the better REST
- Core Concepts
- Big Picture (Architecture)
- Clients
- Server
- +3 more chapters

GRAPHQL RUBY TUTORIAL

- Introduction
- Getting Started
- Queries
- Mutations
- Authentication
- Connecting Nodes
- Error Handling

How To GraphQL

Search tutorials...

WRITTEN BY

Radoslav Stankov
Developer @ Product Hunt

Radoslav is a developer for more than a decade and the organizer of React.Sofia meetup. He believes that frontend and backend are equally important and GraphQL is the best way to connect them.

graphql-ruby Tutorial - Introduction

Motivation

Ruby is a general purpose programming language optimized for programmer happiness. One of its most popular frameworks for building web applications is called Ruby On Rails.

The Ruby ecosystem was one of the first to adopt GraphQL. A couple of popular Ruby on Rails applications like GitHub and Shopify are using it in production already.

In this chapter you'll learn how to build your very own GraphQL server using the following technologies:

- Ruby On Rails: the most popular library for building applications in Ruby
- GraphQL Gem: the most popular library for building GraphQL applications
- GraphiQL: An in-browser IDE for exploring GraphQL, which comes bundled with GraphQL Gem

What is a GraphQL Server?

A GraphQL server should be able to:

- Receive requests following the GraphQL format, for example:

```
{ "query": "query { allUsers { id } }" }
```

- Connect to any necessary databases or services responsible for storing/fetching the actual data.
- Return a GraphQL response with the requested data, such as this:

<https://www.howtographql.com/>

Structure



Project

- ✓  Product Hunt
- ✓  app
 - >  controllers
 - ✓  graph
 - >  mutations
 - >  resolvers
 - >  types
 - >  utils
 - 📄 mutation.rb
 - 📄 query.rb
 - 📄 schema.rb
 - >  helpers
 - >  mailers
 - >  models
 - >  services
 - >  views
 - >  workers
 - >  bin
 - >  config
 - >  db
 - >  frontend
 - >  lib
 - >  log
 - >  public
 - >  spec
 - >  tmp
- 📄 README.md

Product Hunt
app
controllers
graph
mutations
resolvers
types
utils
mutation.rb
query.rb
schema.rb
helpers
mailers
models
services
views
workers
bin
config
db
frontend
lib
log
public
spec
tmp
README.md

Stats

-  51 root query fields
-  136 types
-  132 mutations
-  130 resolvers
-  7 developers

```
Product Hunt
  app
    controllers
  graph
    mutations
  resolvers
    types
    utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
bin
config
db
frontend
fb
log
public
spec
tmp
README.md
```

```
class Graph::Resolvers::Posts::PostsResolver < Graph::Resolvers::SearchResolver
  scope { Post.featured.by_credible_votes }

  OrderType = GraphQL::EnumType.define do
    name 'PostOrder'

    value 'NEWEST'
    value 'POPULAR'
  end

  option :order, type: OrderType, default: 'POPULAR'
  option :date, type: types.Int, with: :apply_date_filter
  option :query, type: types.String, with: :apply_query_filter

  # ...

  private

  def apply_order_with_newest(scope)
    scope.order_by_date
  end

  def apply_order_with_popular(scope)
    scope.order_by_votes
  end

  def apply_date_filter(scope, value)
    scope.for_date(value)
  end

  def apply_query_filter(scope, value)
    scope.for_query(query)
  end

  # ...
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
    utils
  mutation.rb
query.rb
  schema.rb
  helpers
  mailers
  models
  services
  views
  workers
bin
config
db
frontend
fb
log
public
spec
tmp
README.md
```

```
Graph::Query = GraphQL::ObjectType.define do
  name 'Query'

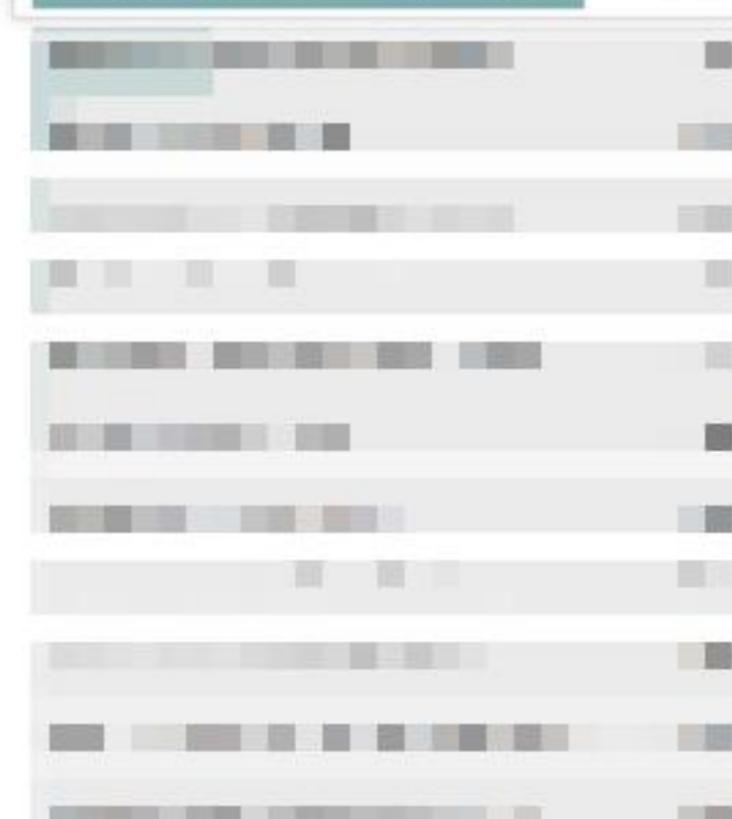
  connection :posts, Graph::Types::PostType.connection_type,
    function: Graph::Resolvers::Posts::PostsResolver

  # ...
end
```

Monitoring







Frontend::GraphQLController#index

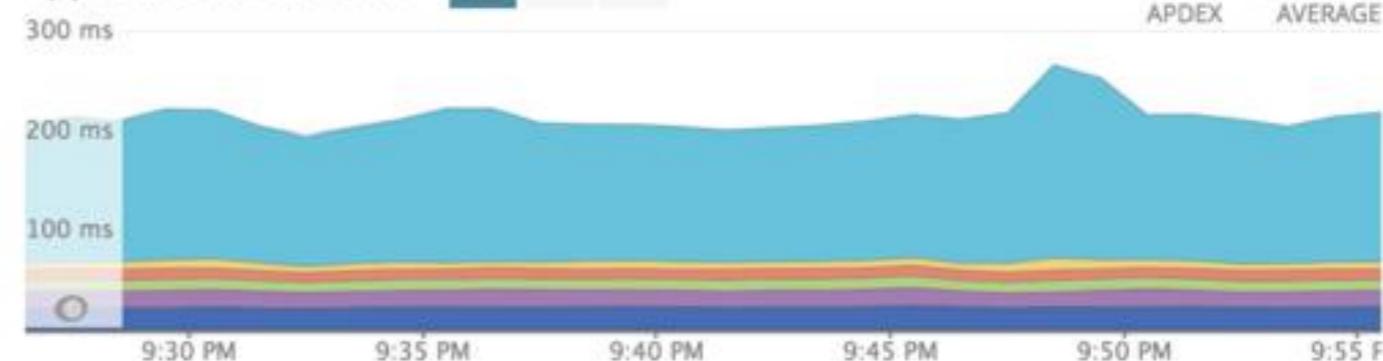
Track as key transaction

App performance

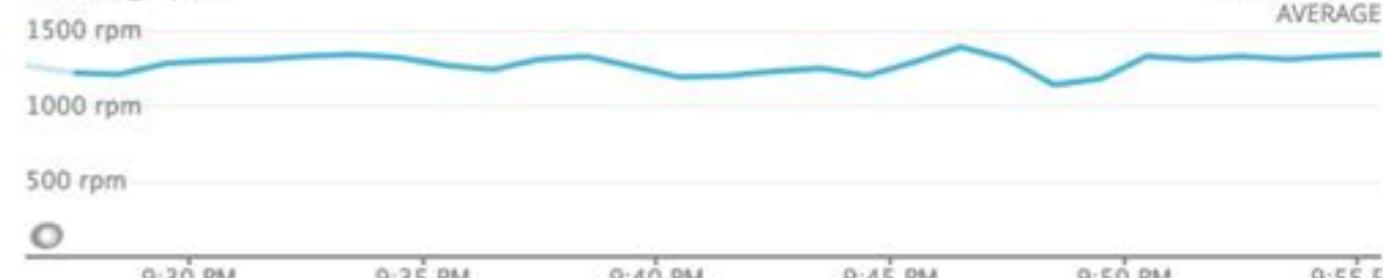
Historical performance

Map Beta

App server breakdown



Throughput



Breakdown table

Category	Segment	% Time	Avg calls (per txn)	Avg time (ms)
GraphQL	validate	10.6	1.0	22.5
Database	Redis get	8.0	11.4	16.9
GraphQL	lex	5.1	1.0	10.7

Type Web

Most time consuming

Frontend::GraphqlController#index



Track as key transaction

App performance

Historical performance

Map Beta

App server breakdown

300 ms

200 ms

100 ms

9:30 PM

9:35 PM

9:40 PM

9:45 PM

9:50 PM

9:55 F

0.93 APDEX

212 ms AVERAGE

validate

Redis get

lex

Post/topics

CommentEdge/node

Other

Throughput

1,280 rpm AVERAGE

1500 rpm

1000 rpm

500 rpm



9:30 PM

9:35 PM

9:40 PM

9:45 PM

9:50 PM

9:55 F

Breakdown table

Category	Segment	% Time	Avg calls (per txn)	Avg time (ms)
GraphQL	validate	10.6	1.0	22.5
Database	Redis get	8.0	11.4	16.9
GraphQL	lex	5.1	1.0	10.7
GraphQL	Post/topics	4.9	8.31	10.4
GraphQL	CommentEdge/node	3.5	1.17	7.44

GraphQL	Comment/user	1.9	1.99	4.05
Database	ActiveRecord Post find	1.7	1.5	3.54
Controller	Frontend::GraphqlController#index	1.4	1.0	3.01
Database	ActiveRecord Post select	1.4	1.63	3.03
GraphQL	UpcomingPage/popular_subscribers	1.3	0.125	2.74
Database	ActiveRecord Topic find	1.2	0.744	2.46
Database	ActiveRecord ShipSubscription find	1.2	0.669	2.61
GraphQL	Topic/related_topics	1.1	0.966	2.31
GraphQL	UserEdge/node	1.1	1.8	2.35
GraphQL	Section/cards	1.0	0.11	2.14
GraphQL	UserConnection/edges	1.0	0.301	2.05
GraphQL	TopicConnection/edges	1.0	4.38	2.14
GraphQL	analyze	1.0	2.0	2.08
Middleware	ActionDispatch::Routing::RouteSet#call	0.9	1.0	1.96
External	Net::HTTP[ph-files.imgix.net]: GET	0.9	0.0004	1.92
GraphQL	Post/meta	0.8	0.106	1.62
GraphQL	JobsCard/jobs	0.8	0.102	1.63
GC	GC Execution	0.7	0.104	1.47
Database	ActiveRecord ProductMedia find	0.7	1.02	1.4
Database	ActiveRecord ProductMaker find	0.7	1.09	1.59
GraphQL	Post/product_links	0.7	0.648	1.58
GraphQL	Comment/replies	0.7	0.48	1.47
GraphQL	CommentConnection/edges	0.7	0.841	1.4
GraphQL	Query/post	0.7	0.123	1.42

```
✓ Product Hunt
  ✓ app
    > controllers
  ✓ graph
    > mutations
    > resolvers
    > types
    > utils
    ⏺ mutation.rb
    ⏺ query.rb
    ⏺ schema.rb
  > helpers
  > mailers
  > models
  > services
  > views
  > workers
> bin
> config
> db
> frontend
> lib
> log
> public
> spec
> tmp
⠀ README.md
```

```
Graph::Schema = GraphQL::Schema.define do
  #...

  use GraphQL::Tracing::NewRelicTracing

  #...
end
```

Windows

A fatal exception error has occurred at 0020:0011E3D in DxD UMM(01) +
00010E36. The current application will be terminated.

- * Press any key to terminate the current application.
- * Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue




NoMethodError Frontend::GraphQLController#index

● undefined method `value' for nil:NilClass | ruby

ISSUE # PRODUCTION-WEB-2J6 EVENTS 3 USERS 1 ASSIGNEE
 Resolve
 Ignore
 Star
 Copy
 Share
Link Issue Tracker
[Details](#) [Comments \(0\)](#) [User Feedback \(0\)](#) [Tags](#) [Events](#) [Merged](#) [Similar Issues](#)
Event [8eca54c507ae4e64bbbc53629caa2f78](#)

May 11, 2018 11:55:02 AM UTC | [JSON \(101.0 KB\)](#)
 K
 Older
 Newer
 >

All Environments

LAST 24 HOURS



LAST 30 DAYS



FIRST SEEN

When: 17 hours ago

May 11, 2018 11:51:49 AM UTC

Release: n/a

LAST SEEN

When: 17 hours ago

May 11, 2018 11:55:02 AM UTC

Release: n/a

MESSAGE

NoMethodError: undefined method `value' for nil:NilClass

EXCEPTION (most recent call first)

App Only
Full
Raw
NoMethodError

undefined method `value' for nil:NilClass

app/services/slate/html_converter.rb in inline at line 83

80. end

Tags





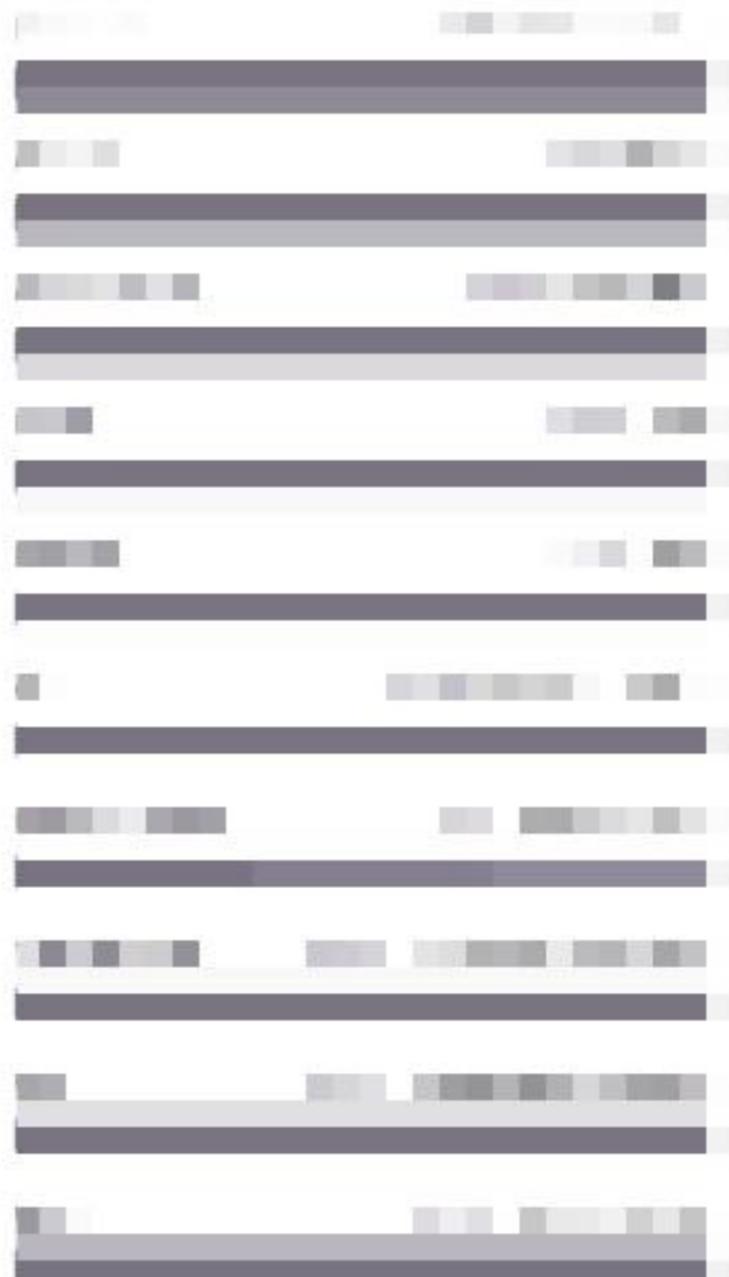
NoMethodError

undefined method `value' for nil:NilClass

```
app/services/slate/html_converter.rb in inline at line 83
80. end
81.
82. def inline(node)
83.   href = node.attributes['href'].value
84.   Slate::Inline.new(type: 'link', data: { href: href }, nodes:
block_nodes(node.children))
85. end
86.

app/services/slate/html_converter.rb in block in block_nodes at line 69 +
app/services/slate/html_converter.rb in map at line 65 +
app/services/slate/html_converter.rb in block_nodes at line 65 +
app/services/slate/html_converter.rb in block in document_nodes at line 45 +
Called from: nokogiri/xml/node_set.rb in block in each
app/services/slate/html_converter.rb in map at line 43 +
app/services/slate/html_converter.rb in document_nodes at line 43 +
app/services/slate/html_converter.rb in call at line 36 +
app/services/slate/html_converter.rb in call at line 25 +
app/services/slate.rb in from_html at line 7 +
app/services/comments/create_form.rb in body= at line 25 +
Called from: mini_form/model.rb in public_send
app/graph/mutations/create_comment.rb in perform at line 17 +
app/graph/resolvers/mutation.rb in call at line 26 +
app/graph/resolvers/mutation.rb in call at line 3 +
Called from: graphql/batch/setup.rb in block (2 levels) in instrument_field
app/graph/middleware/marginalia.rb in call at line 5 +
```

Tags



Notifications

You're receiving updates because you are [subscribed to workflow notifications](#) for this project.

[Unsubscribe](#)

Which query causes
this issue? 🤔

Product Hunt

app

controllers

graph

mutations

resolvers

types

utils

mutation.rb

query.rb

schema.rb

helpers

mailers

models

services

views

workers

bin

config

db

frontend

fb

log

public

spec

tmp

README.md

```
class Frontend::GraphqlController < Frontend::BaseController
  before_action :ensure_query

  def index
    render json: Graph::Schema.execute(query, variables: variables, context: context)
  rescue => e
    handle_error e
  end

  private

  def handle_error(e)
    if Rails.env.development?
      logger.error e.message
      logger.error e.backtrace.join("\n")

      response = {
        data: nil
      errors: [
        message: e.message,
        extensions: {
          backtrace: e.backtrace,
        }
      ],
    }

      render json: response, status: 500
    elsif Rails.env.test?
      p e.message
      p e.backtrace
    else
      Raven.capture_exception(e, extra: { query: query })
    end
  end

  #
end
```

Product Hunt

app

controllers

graph

mutations

resolvers

types

utils

mutation.rb

query.rb

schema.rb

helpers

mailers

models

services

views

workers

bin

config

db

frontend

fb

log

public

spec

tmp

README.md

```
class Frontend::GraphqlController < Frontend::BaseController
  before_action :ensure_query

  def index
    render json: Graph::Schema.execute(query, variables: variables, context: context)
  rescue => e
    handle_error e
  end

  private

  def handle_error(e)
    if Rails.env.development?
      logger.error e.message
      logger.error e.backtrace.join("\n")

      response = {
        data: nil
      errors: [
        message: e.message,
        extensions: {
          backtrace: e.backtrace,
        }
      ],
    }

      render json: response, status: 500
    elsif Rails.env.test?
      p e.message
      p e.backtrace
    else
      Raven.capture_exception(e, extra: { query: query })
    end
  end

  #
end
```



NoMethodError

undefined method `value' for nil:NilClass

app/services/slate/html_converter.rb in **inline** at line 83

```
80. end
81.
82. def inline(node)
83.   href = node.attributes['href'].value
84.   Slate::Inline.new(type: 'link', data: { href: href }, nodes:
block_nodes(node.children))
85. end
86.
```

app/services/slate/html_converter.rb in **block in block_nodes** at line 69

app/services/slate/html_converter.rb in **map** at line 65

app/services/slate/html_converter.rb in **block_nodes** at line 65

app/services/slate/html_converter.rb in **block in document_nodes** at line 45

Called from: nokogiri/xml/node_set.rb in **block in each**

app/services/slate/html_converter.rb in **map** at line 43

app/services/slate/html_converter.rb in **document_nodes** at line 43

app/services/slate/html_converter.rb in **call** at line 36

app/services/slate/html_converter.rb in **call** at line 25

app/services/slate.rb in **from_html** at line 7

app/services/comments/create_form.rb in **body=** at line 25

Called from: mini_form/model.rb in **public_send**

app/graph/mutations/create_comment.rb in **perform** at line 17

app/graph/resolvers/mutation.rb in **call** at line 26

app/graph/resolvers/mutation.rb in **call** at line 3

Called from: graphql/batch/setup.rb in **block (2 levels) in instrument_field**

app/graph/middleware/marginalia.rb in **call** at line 5

Tags



Notifications

You're receiving updates because you are subscribed to **workflow notifications** for this project.

[Unsubscribe](#)

```
query
  mutation CreateComment($commentsThreadRepliesCursor:
    String = "", $input: CreateCommentInput!) {
      response: createComment(input: $input) {
        node {
          _id
          id
          subject {
            _id
            id
            ... on Commentable {
              _id
              id
              comments_count
              __typename
            }
            __typename
          }
          ...CommentsThreadComment
          __typename
        }
        errors {
          field
          messages
          __typename
        }
        __typename
      }
    }

  fragment CommentsThreadComment on Comment {
    _id
    id
    is_sticky
    replies(first: 5, after: $commentsThreadRepliesCursor)
    {
      edges {
        node {
          _id
          id
          ...CommentsThreadItemComment
          __typename
        }
        __typename
      }
      pageInfo {
        endCursor
        hasMore
      }
    }
  }
}
```



Authentication & Authorization

Product Hunt

Secure https://www.producthunt.com

Discover your next favorite thing... Ask Ship Jobs ...

LOG IN SIGN UP

FEEDS

- Home
- Tech
- Games
- Books
- Artificial Intelligence
- Developer Tools
- Home
- Productivity
- Touch Bar Apps
- Wearables
- All Topics

Customize Your Feed

Login to Join The Community

Product Hunt is a community to share and geek out about the latest products, books and games. Join us :)

LOG IN WITH TWITTER LOG IN WITH FACEBOOK LOG IN WITH ANGELIST

We'll never post to any of your accounts without your permission.

Yesterday

POPULAR NEWEST

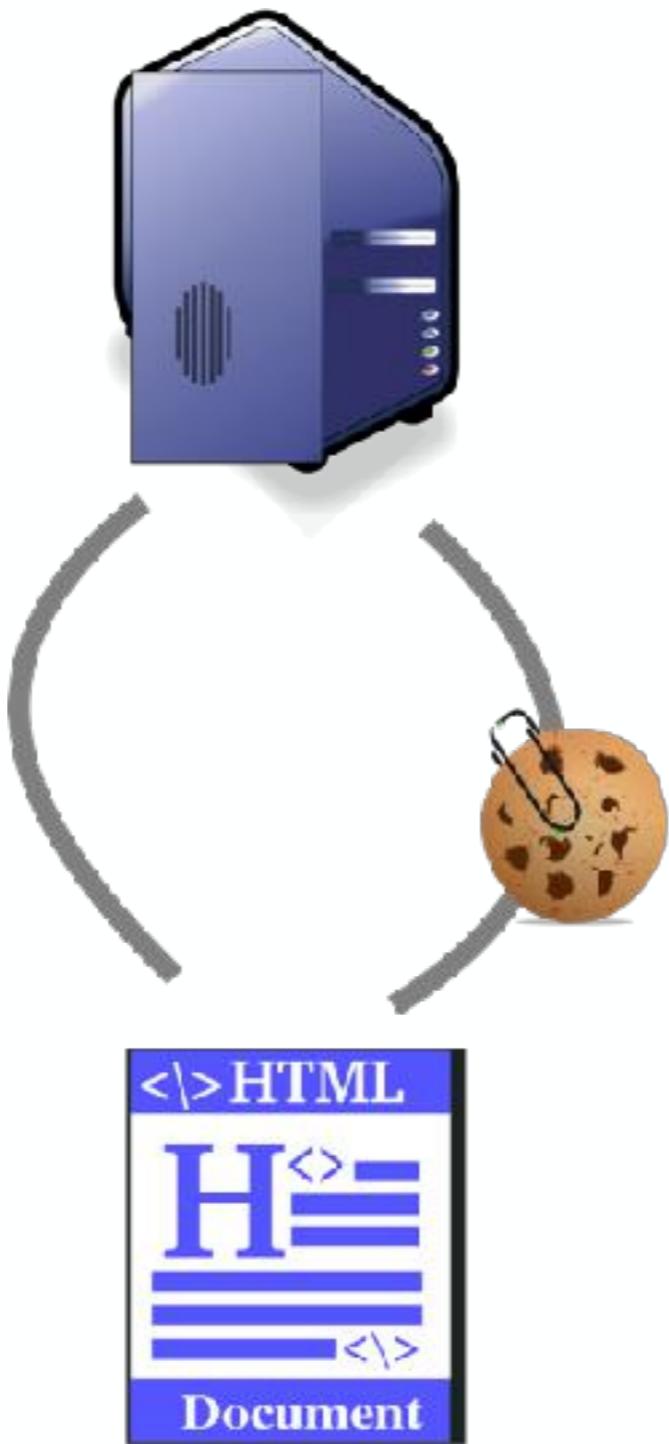
Public Market
Create sales-free eCommerce
CRYPTOCURRENCIES +2 ▲ 400 □ 63

Delux Designer
Customisable, shortcut oriented keyboard built for designers
DESIGN TOOLS +1 ▲ 354 □ 10

How HTTPS Works
We want to tell you how HTTPS works... in a comic! ▲ 278 □ 13

```
gem 'omniauth'  
gem 'a-twitter'  
gem 'omniauth-angellist'  
gem 'omniauth-facebook'
```





- ✓ Product Hunt
- ✓ app
 - ✓ controllers
 - ✓ graph
 - > mutations
 - > resolvers
 - > types
 - > utils
 - mutation.rb
 - query.rb
 - schemaurb
- > helpers
- > mailers
- > models
- > services
- > views
- > workers
- > bin
- > config
- > db
- > frontend
- > lib
- > log
- > public
- > spec
- > tmp
- README.md

```
class Frontend::GraphqlController < Frontend::BaseController
  before_action :ensure_query

  def index
    render json: Graph::Schema.execute(query, variables: variables, context: context)
  rescue => e
    handle_error e
  end

  private

  # ...

  def context
    {
      current_user: current_user,
      request: request,
    }
  end

  # ...
end
```

```
Product Hunt
  app
    controllers
    graph
      mutations
      resolvers
      types
      utils
    mutation.rb
  query.rb
  schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Query = GraphQL::ObjectType.define do
  name 'Query'

  field :viewer, Graph::Types::ViewerType do
    resolve ->(_obj, _args, ctx) { ctx[:current_user] }
  end

  # ...
end
```

```
query {  
  viewer {  
    id  
    username  
    // ...  
  }  
}
```



```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
    utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Schema = GraphQL::Schema.define do
  # ...

  authorization Graph::Utils::AuthorizationStrategy

  instrument :field, Graph::Utils::FallbackInstrumenter

  # ...
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
  utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
bin
config
db
frontend
fb
log
public
spec
tmp
README.md
```

```
class Graph::Utils::AuthorizationStrategy
  def initialize(ctx)
    @current_user = ctx[:current_user]
  end

  def allowed?(gate, value)
    Authorization.can? @current_user, gate.role, value
  end
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
  types
  utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
bin
config
db
frontend
fb
log
public
spec
tmp
README.md
```

```
Graph::Types::SurveyType = GraphQL::ObjectType.define do
  name 'Survey'

  authorize Authorization::READ

  field :id, !types.ID
  field :title, !types.String

  # ...

  field :questions, function: Graph::Resolvers::Surveys::QuestionsResolver.new
  field :answers, authorize: Authorization::MANAGE_FIELD, fallback: false, function: Graph::Resolvers::AnswersResolver.new

  field :canManage, function: Graph::Resolvers::CanResolver.new(Authorization::MANAGE_FIELD)
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
  types
  utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::SurveyType = GraphQL::ObjectType.define do
  name 'Survey'

  authorize Authorization::READ

  field :id, !types.ID
  field :title, !types.String

  # ...

  field :questions, function: Graph::Resolvers::Surveys::QuestionsResolver.new
  field :answers, authorize: Authorization::MANAGE_FIELD, fallback: [], function:
    field :canManage, function: Graph::Resolvers::CanResolver.new(Authorization::)
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
  types
  utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
bin
config
db
frontend
fb
log
public
spec
tmp
README.md
```

```
Graph::Types::SurveyType = GraphQL::ObjectType.define do
  name 'Survey'
```

```
    authorize Authorization::READ
```

```
module Authorization
  READ = :read
  MANAGE = :manage
  MANAGE_FIELD = { parent_role: MAINTAIN }

  #
end
```

```
surveys::QuestionsResolver.new
  field :parent, type: GraphQL::String
  resolve Surveys::QuestionResolvers::ParentField, fallback: []
  function Surveys::QuestionResolvers::ParentFunction
    parent_resolver: Surveys::QuestionResolvers::ParentResolver
    args: Surveys::QuestionResolvers::ParentArgs
    context: Surveys::QuestionResolvers::ParentContext
    description: Surveys::QuestionResolvers::ParentDescription
    type: Surveys::QuestionResolvers::ParentType
  end
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
  types
  utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::SurveyType = GraphQL::ObjectType.define do
  name 'Survey'

  authorize Authorization::READ

  field :id, !types.ID
  field :title, !types.String

  # ...

  field :questions, function: Graph::Resolvers::Surveys::QuestionsResolver.new
  field :answers, authorize: Authorization::MANAGE_FIELD, fallback: false, function: Graph::Resolvers::AnswersResolver.new

  field :canManage, function: Graph::Resolvers::CanResolver.new(Authorization::MANAGE_FIELD)
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
  types
  utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::SurveyType = GraphQL::ObjectType.define do
  name 'Survey'

  authorize Authorization::READ

  field :id, !types.ID
  field :title, !types.String

  # ...

  field :questions, function: Graph::Resolvers::Surveys::QuestionsResolver.new
  field :answers, authorize: Authorization::MANAGE_FIELD, fallback: []
  field :canManage, function: Graph::Resolvers::CanResolver.new(Authorization::MANAGE_FIELD)

end
```

```
Product Hunt
  app
    controllers
  graph
    mutations (selected)
    resolvers
    types
    utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
class Graph::Mutations::CollectionAddPost < Graph::Resolvers::Mutation
  input :post_id, !types.ID

  node :collection, type: Collection
  authorize: Authorization::MANAGE
  returns Graph::Types::PostType

  def perform
    post = Post.find inputs[:post_id]
    CollectionPosts.add collection, post
    post
  end
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations (selected)
    resolvers
    types
    utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
class Graph::Mutations::CollectionAddPost < Graph::Resolvers::Mutation
  input :post_id, !types.ID

  node :collection, type: Collection
  authorize: Authorization::MANAGE

  returns Graph::Types::PostType

  def perform
    post = Post.find inputs[:post_id]
    CollectionPosts.add collection, post
    post
  end
end
```



Performance

Today



Prisma

Build a GraphQL server with any database

DEVELOPER TOOLS

▲ 992

47



Coinbase Prime

A professional trading platform for cryptocurrencies

FINTECH + 3

▲ 494

15



Surface Hub 2

Meet Surface Hub 2, the future of collaboration

PRODUCTIVITY + 1

▲ 417

16



Google One

One simple way to get more out of Google

WEB APP + 2

▲ 407

6



Weekly UX Exercise

Receive challenges top companies use to interview designers

DESIGN TOOLS + 2

▲ 440

13

Today



Prisma

Build a GraphQL server with any database

DEVELOPER TOOLS

▲ 982

47

coinbase
Prime

Coinbase Prime

A professional trading platform for cryptocurrencies

FINTECH

+ 3

▲ 494

15



Surface Hub 2

Meet Surface Hub 2, the future of collaboration

PRODUCTIVITY

+ 1

▲ 417

16



Google One

One simple way to get more out of Google

WEB APP

+ 2

▲ 407

6



Weekly UX Exercise

Receive challenges top companies use to interview designers

DESIGN TOOLS

+ 2

▲ 440

11

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```

Today

 Prisma
Build a GraphQL server with any database
DEVELOPER TOOLS ▲ 982 47

 Coinbase Prime
A professional trading platform for cryptocurrencies
FINTECH + 3 ▲ 494 15

 Surface Hub 2
Meet Surface Hub 2, the future of collaboration
PRODUCTIVITY + 1 ▲ 417 16

 Google One
One simple way to get more out of Google
WEB APP + 2 ▲ 407 6

 Weekly UX Exercise
Receive challenges top companies use to interview designers
DESIGN TOOLS + 2 ▲ 440 11

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```

```
posts(date: '2018-05-05') {  
  id  
  topics {  
    id  
  }  
  isVoted  
}
```

```
[{  
  id: 1,  
  topics: [{ id: 1 }],  
  isVoted: true  
, {  
  id: 2,  
  topics: [{ id: 2 }],  
  isVoted: true  
, {  
  id: 3,  
  topics: [{ id: 2 }],  
  isVoted: true  
}]
```

```
[{  
    id: 1,  
    topics: [{ id: 1 }],  
    isVoted: true  
}, {  
    id: 2,  
    topics: [{ id: 2 }],  
    isVoted: true  
}, {  
    id: 3,  
    topics: [{ id: 2 }],  
    isVoted: true  
}]
```

SELECT * FROM posts WHERE DATE(featured_at) = {date}

SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_id}

SELECT * FROM votes WHERE post_id={post.id} and user_id={user_id}

SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_id}

SELECT * FROM votes WHERE post_id={post.id} and user_id={user_id}

SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_id}

SELECT * FROM votes WHERE post_id={post.id} and user_id={user_id}





```
[{  
    id: 1,  
    topics: [{ id: 1 }],  
    isVoted: true  
}, {  
    id: 2,  
    topics: [{ id: 2 }],  
    isVoted: true  
}, {  
    id: 3,  
    topics: [{ id: 2 }],  
    isVoted: true  
}]
```

[...]

```
[{  
    id: 1,  
    topics: [find topics for post 1],  
    isVoted: [is post 1 voted]  
}, ...]
```

```
[{  
    id: 1,  
    topics: [find topics for post 1],  
    isVoted: [is post 1 voted]  
}, {  
    id: 2,  
    topics: [find topics for post 2],  
    isVoted: [is post 2 voted]  
}, ...]
```

```
[{  
    id: 1,  
    topics: [find topics for post 1],  
    isVoted: [is post 1 voted]  
}, {  
    id: 2,  
    topics: [find topics for post 2],  
    isVoted: [is post 2 voted]  
}, {  
    id: 3,  
    topics: [find topics for post 3],  
    isVoted: [is post 3 voted]  
}]
```



[find topics for post 1, 2, 3]
[is post 1, 2, 3 voted]

```
[{
  id: 1,
  topics: [find topics for post 1],
  isVoted: [is post 1 voted]
}, {
  id: 2,
  topics: [find topics for post 2],
  isVoted: [is post 2 voted]
}, {
  id: 3,
  topics: [find topics for post 3],
  isVoted: [is post 3 voted]
}]
```

```
SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_ids}
SELECT * FROM votes WHERE post_id IN {post_ids} and user_id={user_id}
```

```
[{
  id: 1,
  topics: [{ id: 1 }],
  isVoted: true
}, {
  id: 2,
  topics: [{ id: 2 }],
  isVoted: true
}, {
  id: 3,
  topics: [{ id: 2 }],
  isVoted: true
}]
SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_ids}
SELECT * FROM votes WHERE post_id IN {post_ids} and user_id={user_id}
```

```
gem "graphql-batch"
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
    utils
    mutation.rb
    query.rb
    schema.rb
  helpers
  mailers
  models
  services
  views
  workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Schema = GraphQL::Schema.define do
  # ...

  use GraphQL::Batch

  # ...
end
```

```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
    votesCount
    commentsCount
    topics {
      id
      name
    }
    isVoted
  }
}
```

```
query {  
  posts(date: '2018-05-05') {  
    id  
    name  
    tagline  
    votesCount  
    commentsCount  
    topics {  
      id  
      name  
    }  
    isVoted  
  }  
}
```

Project

- ✓ Product Hunt
- ✓ app
 - > controllers
 - ✓ graph
 - > mutations
 - > resolvers
 - ✓ topics
 - > utils
 - mutation.rb
 - query.rb
 - schemerb
 - > helpers
 - > mailers
 - > models
 - > services
 - > views
 - > workers
- > bin
- > config
- > db
- > frontend
- > lib
- > log
- > public
- > spec
- > tmp

```
class Graph::Resolvers::Posts::TopicsResolver < GraphQL::Function
  type !types[!Graph::Types::TopicType]

  def call(post, _args, _ctx)
    TopicsLoader.for.load(post)
  end

  class TopicsLoader < GraphQL::Batch::Loader
    def perform(posts)
      ::ActiveRecord::Associations::Preloader.new.preload(posts, :topics)

      posts.each do |post|
        fulfill post, post.topics
      end
    end
  end
end
```

```
✓ Product Hunt
  ✓ app
    > controllers
  ✓ graph
    > mutations
    > resolvers
  > types
    > mutation.rb
    > query.rb
    > schema.rb
  > helpers
  > mailers
  > models
  > services
  > views
  > workers
> bin
> config
> db
> frontend
> lib
> log
> public
> spec
> tmp
  README.md
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

  field :id, !types.ID
  field :name, !types.String
  field :tagline, !types.String
  field :topics, function: Graph::Resolvers::Posts::TopicsResolver.new
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
      mutation.rb
      query.rb
      schema.rb
    helpers
    mailers
    models
    services
    views
    workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

  field :id, !types.ID
  field :name, !types.String
  field :tagline, !types.String
  field :topics, function: Graph::Resolvers::Posts::TopicsResolver.new
end
```

```
Product Hunt
  app
    controllers
    graph
      mutations
      resolvers
    types
      mutation.rb
      query.rb
      schema.rb
    helpers
    mailers
    models
    services
    views
    workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

  field :id, !types.ID
  field :name, !types.String
  field :tagline, !types.String
  field :topics, function: Graph::Resolvers::Posts::TopicsResolver.new
end
```

https://github.com/Shopify/graphql-batch/blob/master/examples/association_loader.rb

<https://gist.github.com/RStankov/48070003a31d71a66f57a237e27d5865>

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
      mutation.rb
      query.rb
      schema.rb
    helpers
    mailers
    models
    services
    views
    workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

  field :id, !types.ID
  field :name, !types.String
  field :tagline, !types.String
  field :topics, function: Graph::Resolvers::AssociationResolver.new(:topics),
end
```

```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
    votesCount
    commentsCount
    topics {
      id
      name
    }
    isVoted
  }
}
```

```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
    votesCount
    commentsCount
    topics {
      id
      name
    }
    isVoted
  }
}
```

```
Product Hunt
  app
    controllers
  graph
    mutations
  resolvers
    types
    utils
      mutation.rb
      query.rb
      schema.rb
  helpers
  mailers
  models
  services
  views
  workers
bin
config
db
frontend
lib
log
public
spec
tmp
README.md
```

```
class Graph::Resolvers::Posts::HasVotedResolver < GraphQL::Function
  type !types.Boolean

  def call(post, _args, ctx)
    return false unless ctx[:current_user]
    VotesLoader.for(ctx[:current_user]).load(post)
  end

  class VotesLoader < GraphQL::Batch::Loader
    def initialize(user)
      @user = user
    end

    def perform(posts)
      voted_posts_ids = @user.votes.where(post_id: posts.map(&:id)).pluck(:post_id)
      posts.each do |post|
        fulfill post, voted_posts_ids.include?(post.id)
      end
    end
  end
end
```

```
Product Hunt
app
  controllers
graph
  mutations
  resolvers
  types
utils
  mutation.rb
  query.rb
  schema.rb
helpers
mailers
models
services
views
workers
bin
config
db
frontend
fb
log
public
spec
tmp
README.md
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

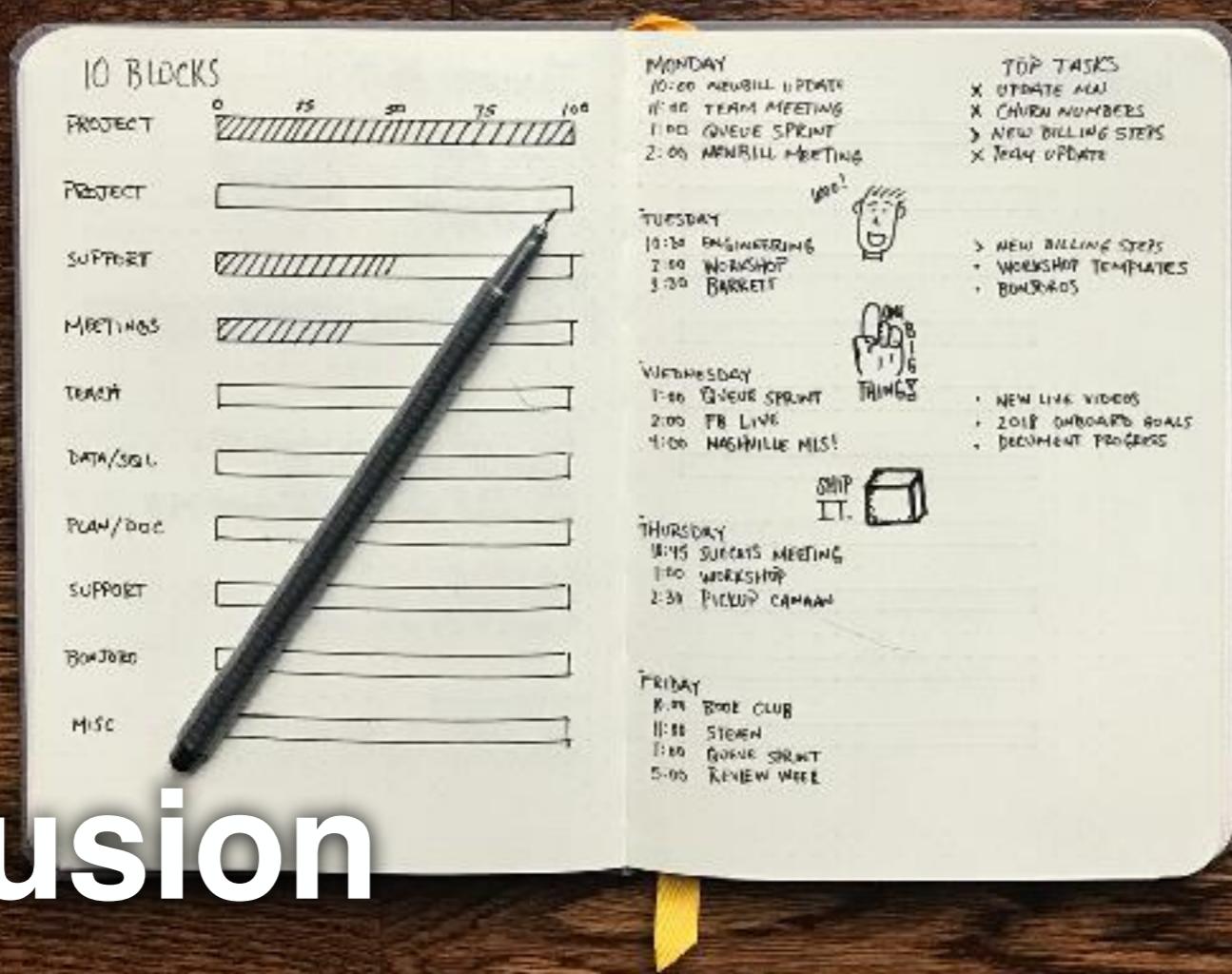
  field :id, !types.ID
  field :name, !types.String
  field :tagline, !types.String
  field :topics, function: Graph::Resolvers::Posts::TopicsResolver.new
  field :isVoted, function: Graph::Resolvers::Posts::HasVotedResolver.new
end
```

```
Product Hunt
  app
    controllers
  graph
    mutations
    resolvers
    types
      mutation.rb
      query.rb
      schema.rb
    helpers
    mailers
    models
    services
    views
    workers
  bin
  config
  db
  frontend
  lib
  log
  public
  spec
  tmp
  README.md
```

```
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

  field :id, !types.ID
  field :name, !types.String
  field :tagline, !types.String
  field :topics, function: Graph::Resolvers::Posts::TopicsResolver.new
  field :isVoted, function: Graph::Resolvers::Posts::HasVotedResolver.new
end
```

Conclusion







Thanks 😎





<https://speakerdeck.com/rstankov/one-year-graphql-in-production>