One Year GraphQL in Production

Radoslav Stankov

23/05/2018

Radoslav Stankov

@rstankov

blog.rstankov.com
github.com/rstankov
twitter.com/rstankov

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


Product Hunt

early 2014

jQuery spaghetti

early 2014

jQuery spaghetti

October 2014

Backbone

early 2014 — **jQuery** spaghetti

October 2014 — **Backbone**

February 2015 — **React** & **Rails**

early 2014 — **jQuery** spaghetti

October 2014 — **Backbone**

February 2015 — **React** & **Rails**

May 2015 — 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**

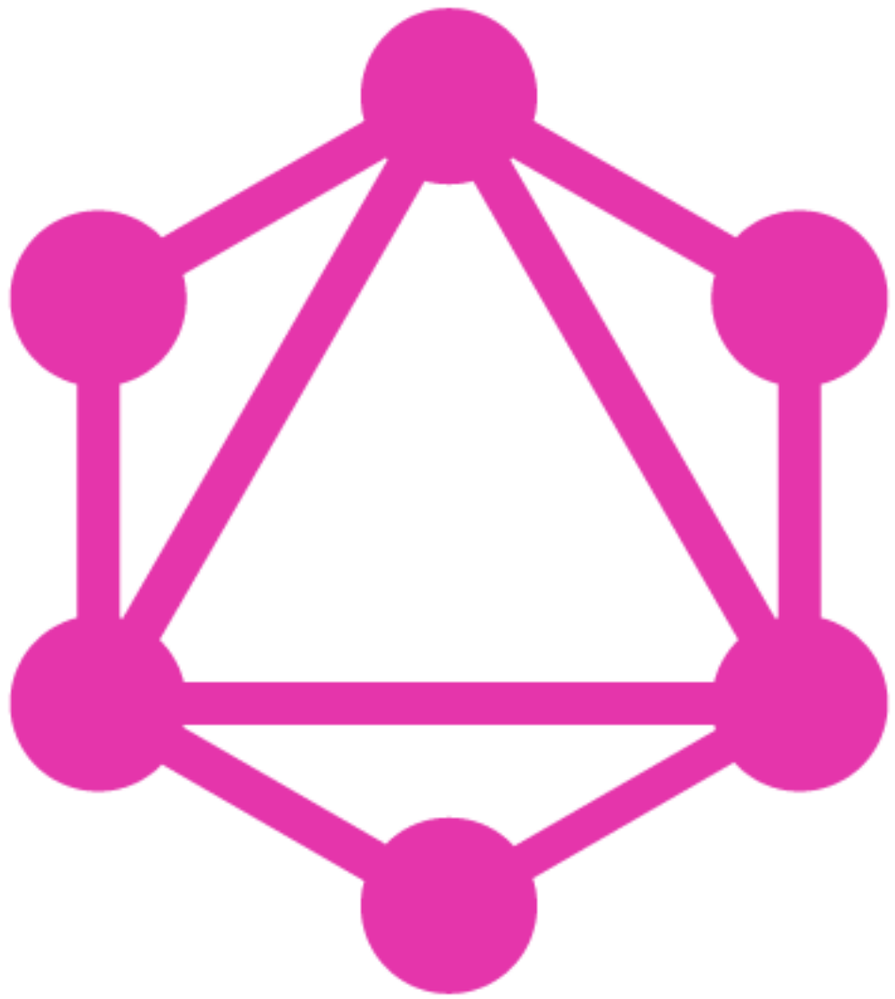
May 2015 — custom **Flux**

December 2015 — **Redux**

January 2016 — **React-Router**

April 2016 — **Redux Ducks**

Febuary 2017 — **GraphQL**



http://graphql.org/

FEEDS Customize

Yours
Home
Tech
Games
Books
APIs
Design Tools
iPad
iPhone
Mac
Open Source
Task Management
Text Editors
All Topics

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 7

CodementorX 2.0 PROMOTED
Hire world-class freelance developers for your team
▲ 718 6

Hack Your Blog to the Top 5 in Google
Authoritative & actionable blog marketing guide
SALES + 4 ▲ 237 10

Contractbook
Efficient contract management, automated through Zapier
PRODUCTIVITY + 5 ▲ 130 11

Now
The future of online dating, Now.
IPHONE + 3 ▲ 133 9

Teams hiring now

Gusto
Payroll, Benefits, and HR loved 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

▲ 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

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

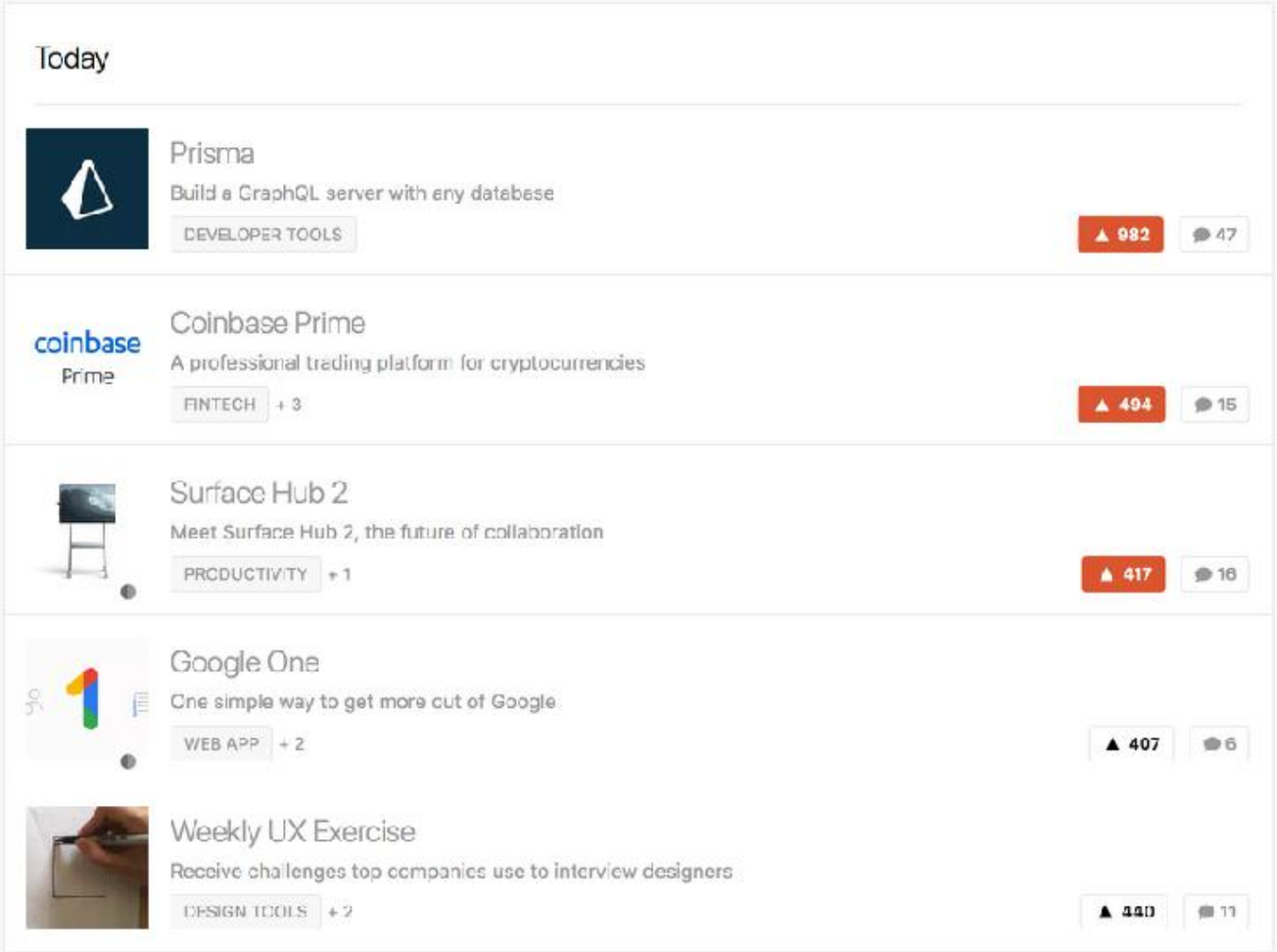
Weekly UX Exercise

Receive challenges top companies use to interview designers

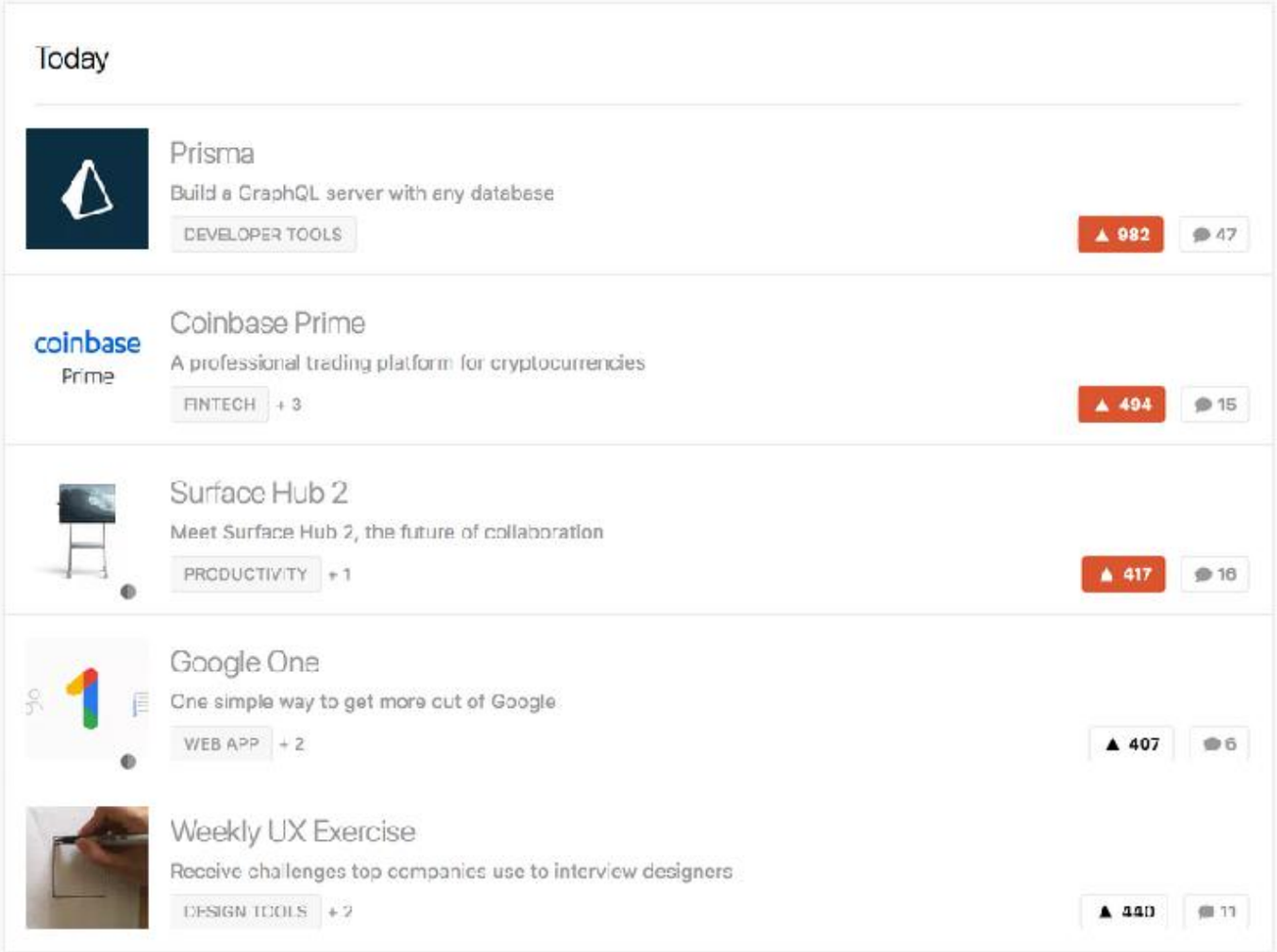
DESIGN TOOLS + 2

▲ 440 · 11

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

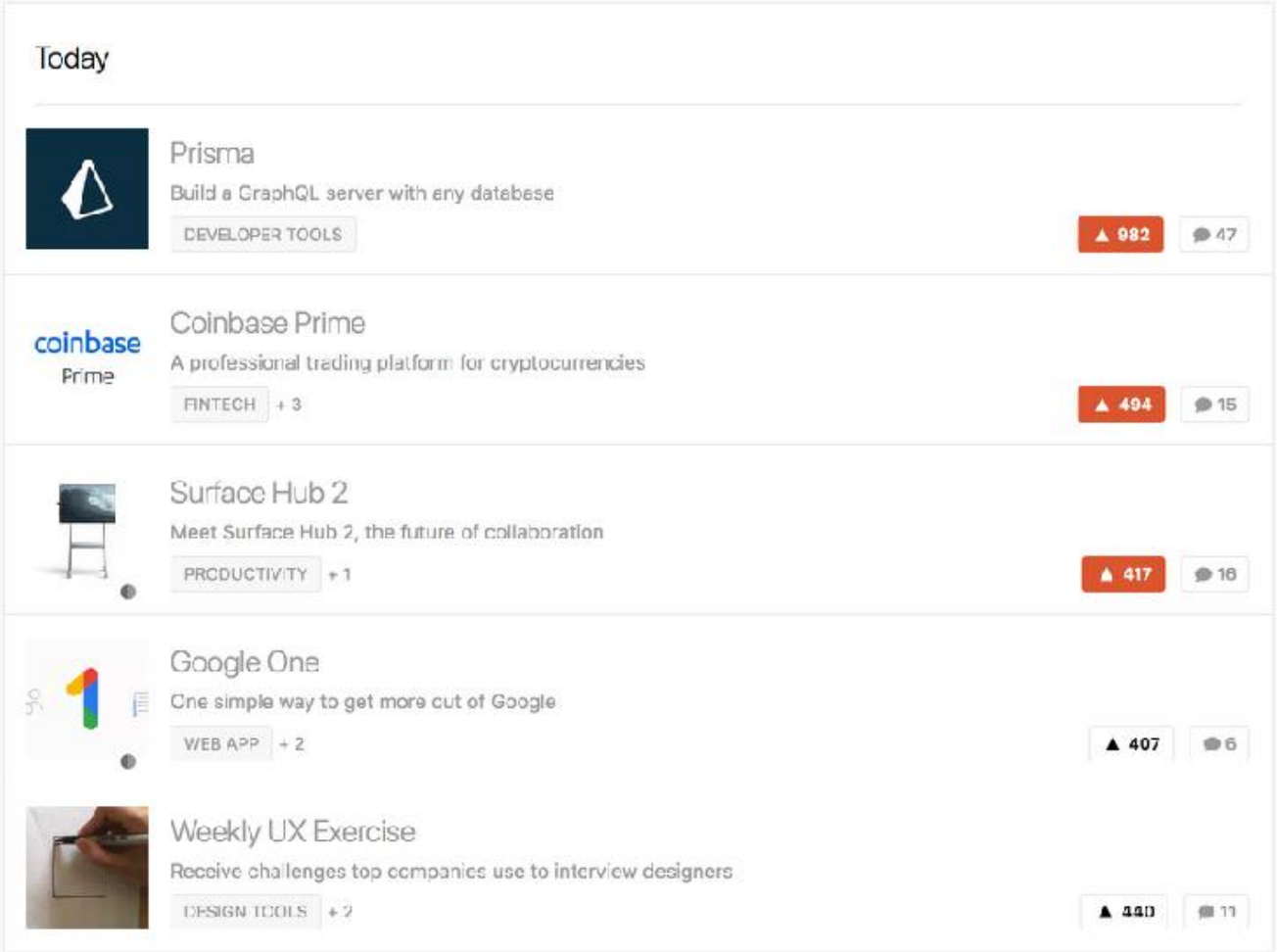


```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
    votesCount
    commentsCount
    topics {
      id
      name
    }
    isVoted
  }
}
```



POST /graphql

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



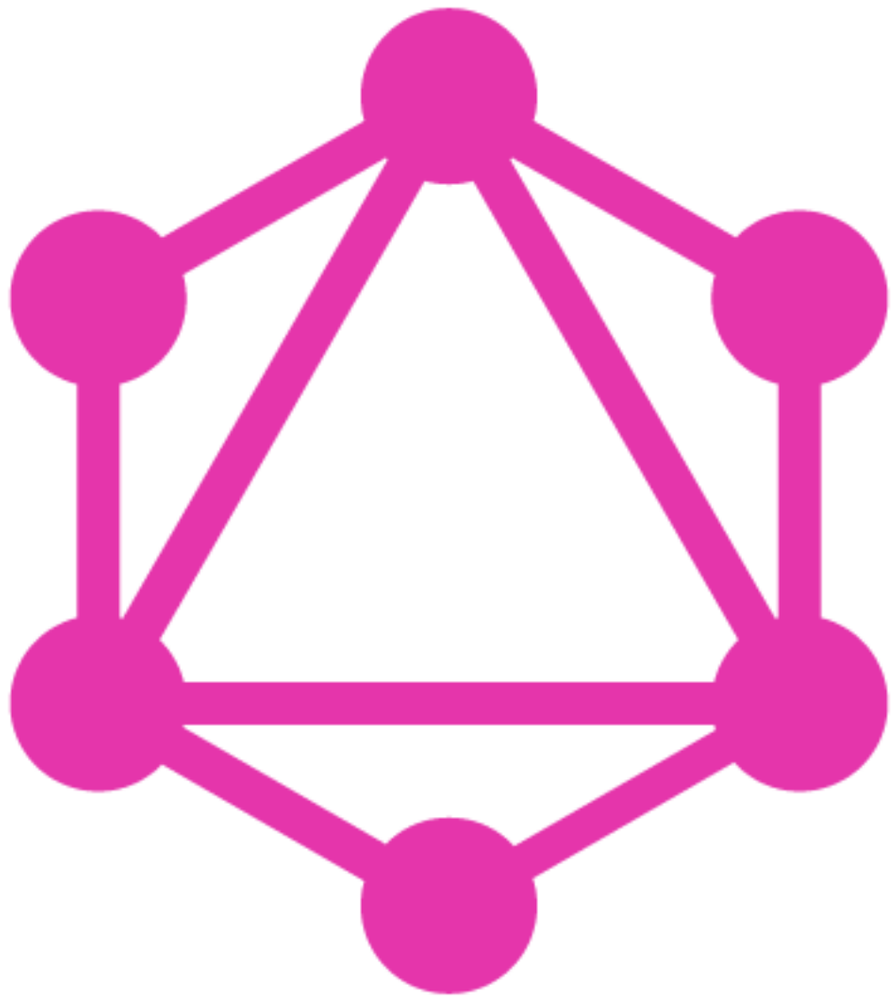
POST /graphql

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

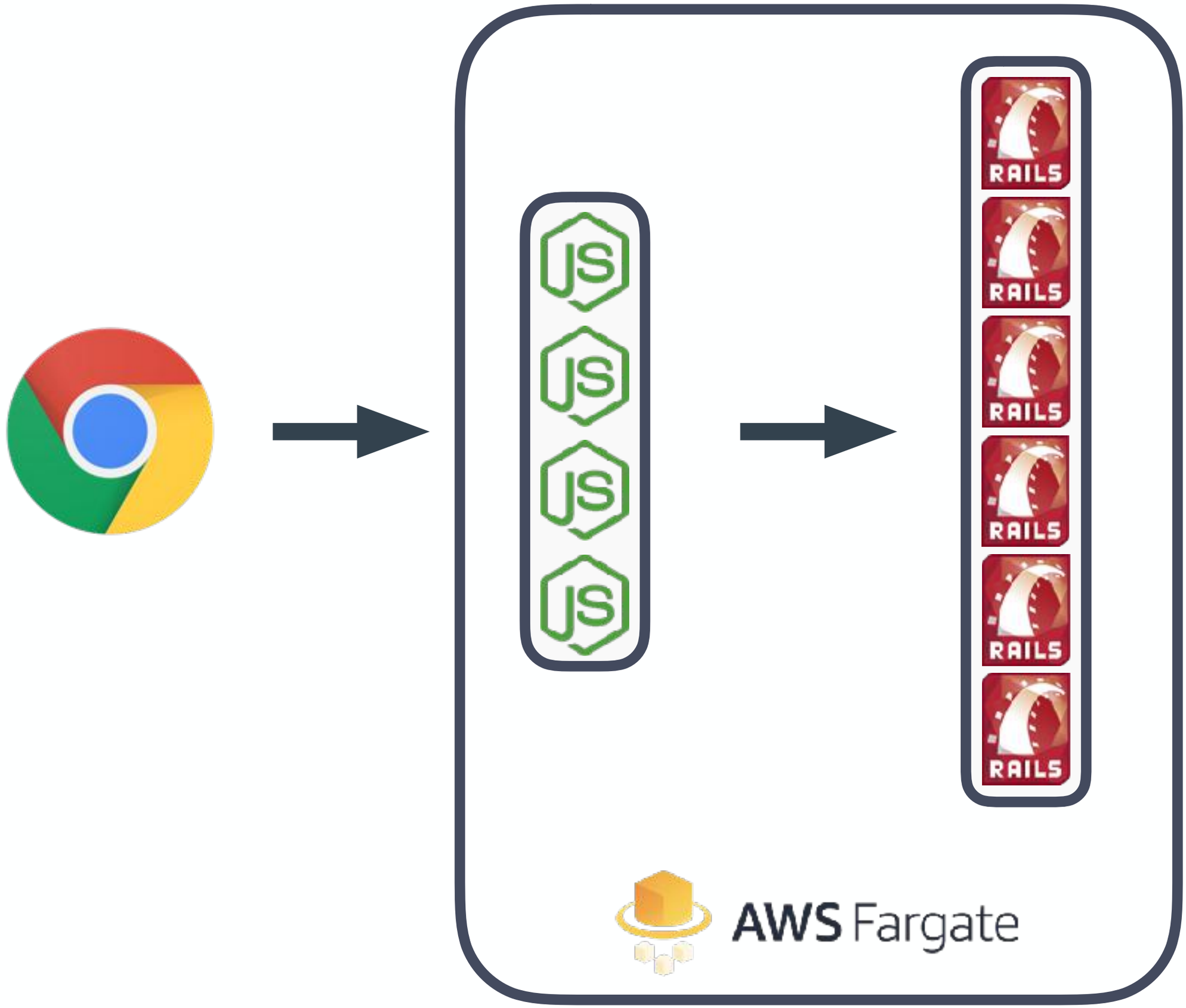
POST /graphql

```
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/


JS
JS
JS
JS
RAILS
RAILS
RAILS
RAILS
RAILS
RAILS
AWS Fargate


RAILS

```
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
```

```ruby
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
```

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

```
gem "graphiql-rails"
```

```ruby
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
```

GraphiQL

Not Secure ph.test:5000/graphiql?query=query%20%7B%0A%20%20post(slug%3A%20"product-hunt")%20%7B%0A%20%20%20%20%2...

GraphiQL ▶ Prettify History

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

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

QUERY VARIABLES

Query **Post**

Search Post...

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_url: String

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

GraphiQL

Not Secure ph.test:5000/graphiql?query=query%20%7B%0A%20%20post(slug%3A%20"product-hunt")%20%7B%0A%20%20%20%20%2...

GraphiQL ▶ Prettify History

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

Query

QUERY VARIABLES

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

Query **Post**

Search Post...

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_url: String

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

GraphiQL

Not Secure ph.test:5000/graphiql?query=query%20%7B%0A%20%20post(slug%3A%20"product-hunt")%20%7B%0A%20%20%20%20%2...

GraphiQL Prettify History

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

QUERY VARIABLES

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

Result

Query **Post**

Search Post...

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_url: String

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

GraphiQL

Not Secure ph.test:5000/graphiql?query=query%20%7B%0A%20%20post(slug%3A%20"product-hunt")%20%7B%0A%20%20%20%20%2...

GraphiQL ▶ Prettify History

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

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

QUERY VARIABLES

Query **Post**

Search Post...

No Description

IMPLEMENTS

Node
Commentable
MetaTaggable
Topicable
Shareable
Reviewable
Votable
HasEmbeds

Automatic Documentation

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
```

GraphiQL

Not Secure ph.test:5000/graphiql?query=query%20%7B%0A%20%20post(slug%3A%20"product-hunt")%20%7B%0A%20%20%20%20%2...

GraphiQL ▶ Prettify History

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

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

QUERY VARIABLES

Query **Post**

Search Post...

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_url: String

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

GraphQL Benefits
(backend developer edition)

🚕 simplifies communication with frontend developers

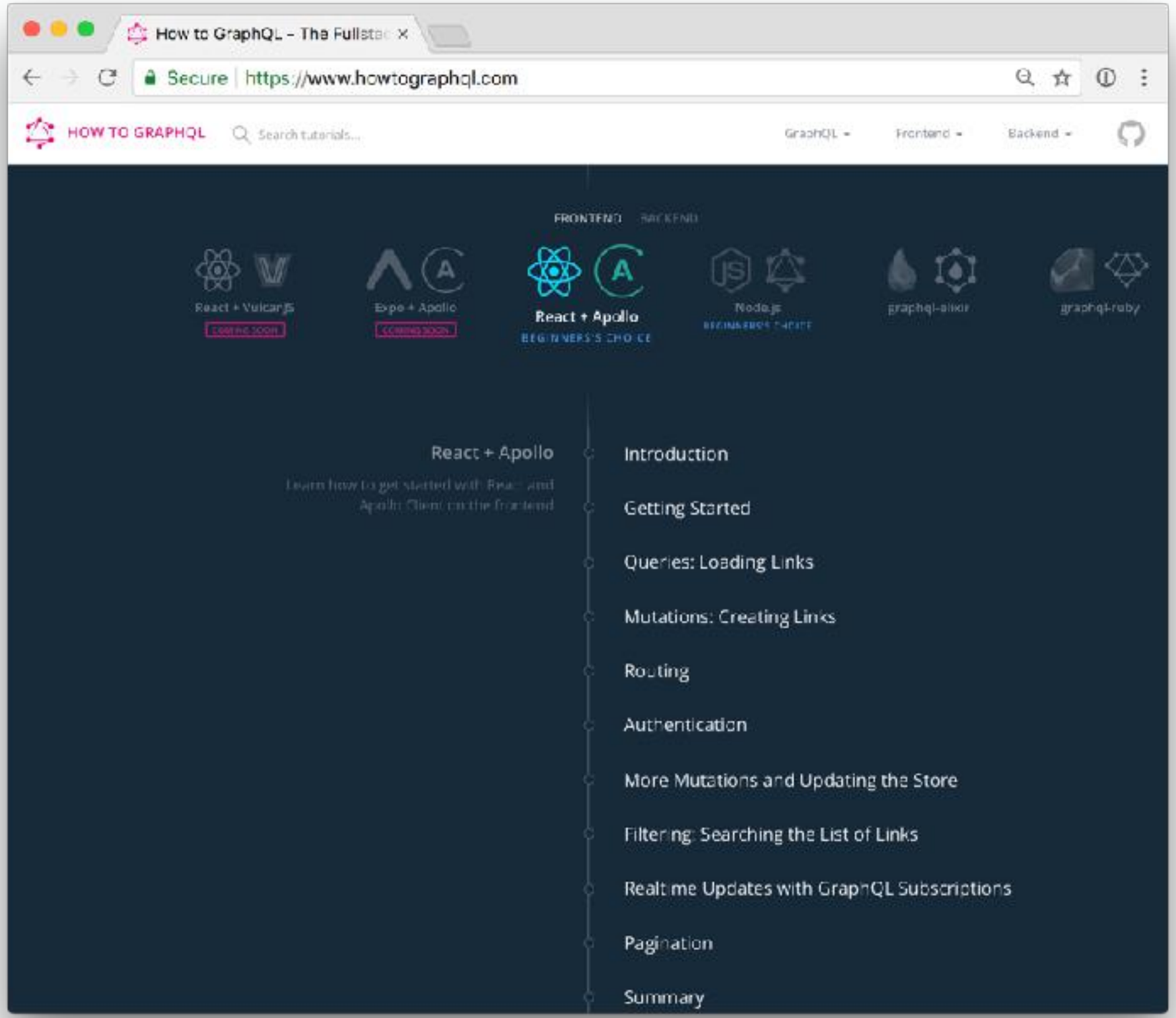
🚙 no extra controllers

🚗 no extra routes

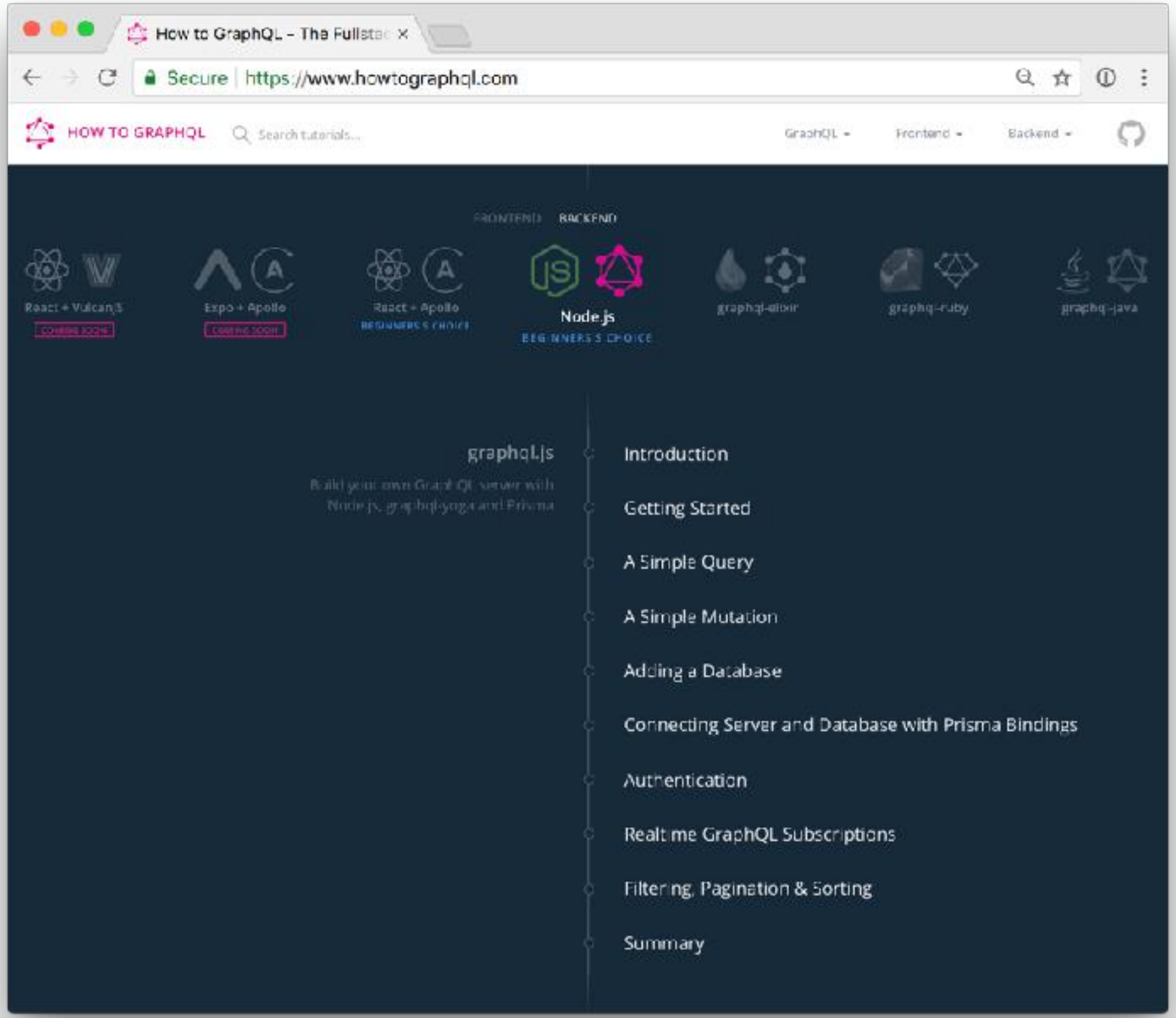
🚎 build-in data serialization

🚑 build-in documentation

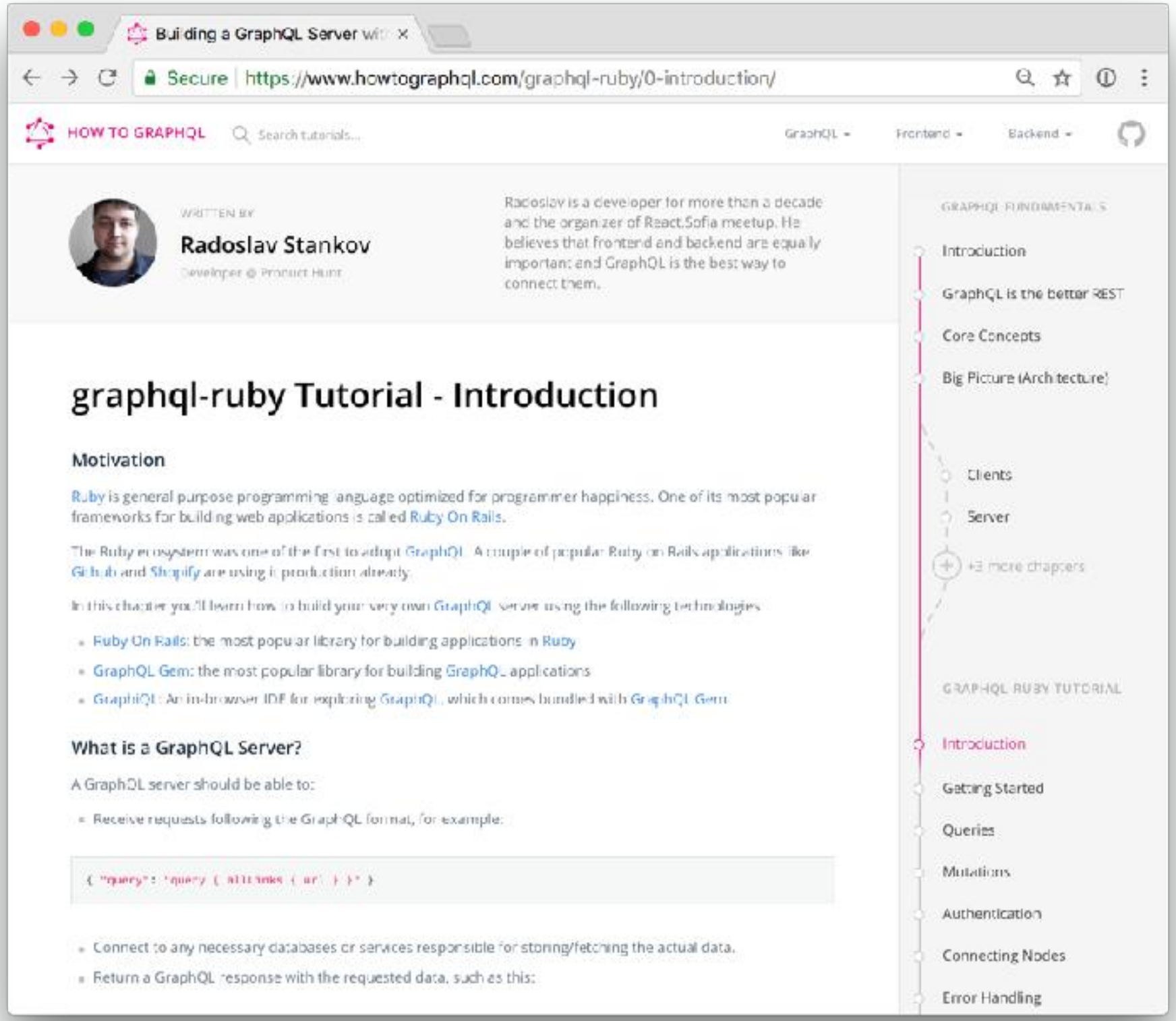
🚌 ecosystem of tools



https://www.howtographql.com/



https://www.howtographql.com/



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

Stats

🚕 51 root query fields

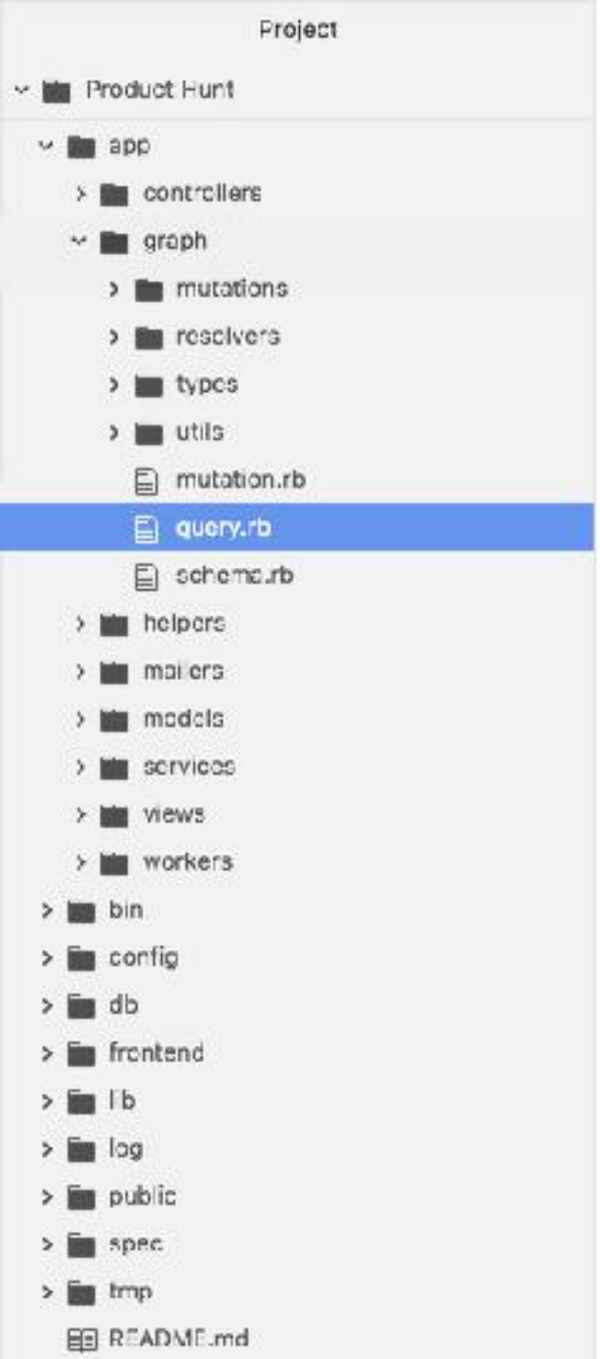
🚙 136 types

🚗 132 mutations

🚓 130 resolvers

👨‍💻 7 developers


```ruby
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
```




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

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

  # ...
end
```

Monitoring


New
Relic®

Heroku Add-ons

My Apps Add-ons Documentation Support

New Relic APM

Maps Alerts New Tools Help producthunt 2

Applications Service maps Key transactions Alerts

APPS
Product Hunt

TIME PICKER
Last 30 minutes ending now

SERVERS
All servers

MONITORING

- Overview
- Service maps
- **Transactions**
- Databases
- External services
- Instances
- Ruby VMs

EVENTS

- Error analytics
- Errors
- Alerts
- Deployments
- Thread profiler

REPORTS

- SLA
- Availability
- Capacity
- Scalability
- Web transactions
- Database
- Background jobs

Type Web

Most time consuming

Frontend::GraphqlController#index	**71.2%**

Frontend::GraphqlController#index

Track as key transaction

App performance | Historical performance | Map Beta

App server breakdown

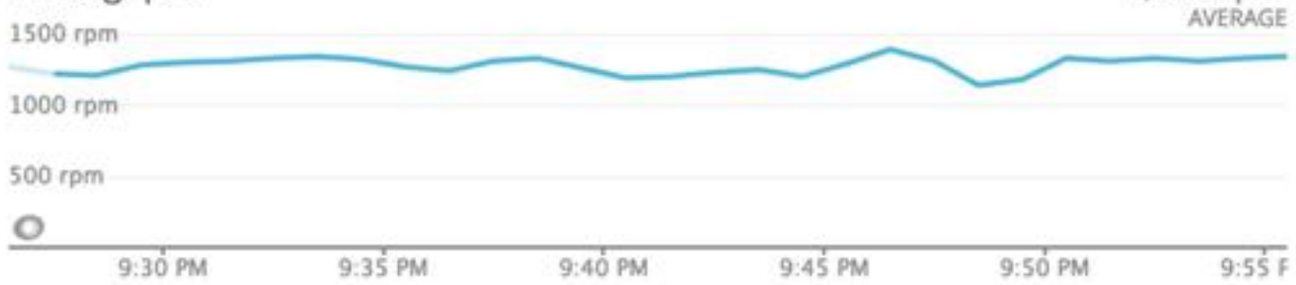
0.93 APDEX

212 ms AVERAGE

validate | Redis get | lex | Post/topics | CommentEdge/node | Other

Throughput

1,280 rpm AVERAGE



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	71.2%

Frontend::GraphqlController#index

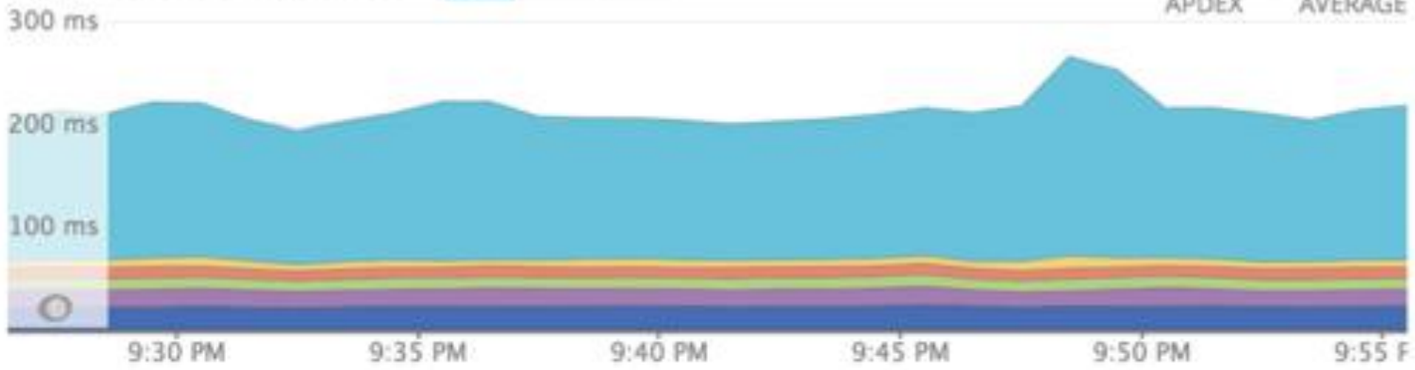
Track as key transaction

App performance | Historical performance | Map Beta

App server breakdown

0.93 APDEX

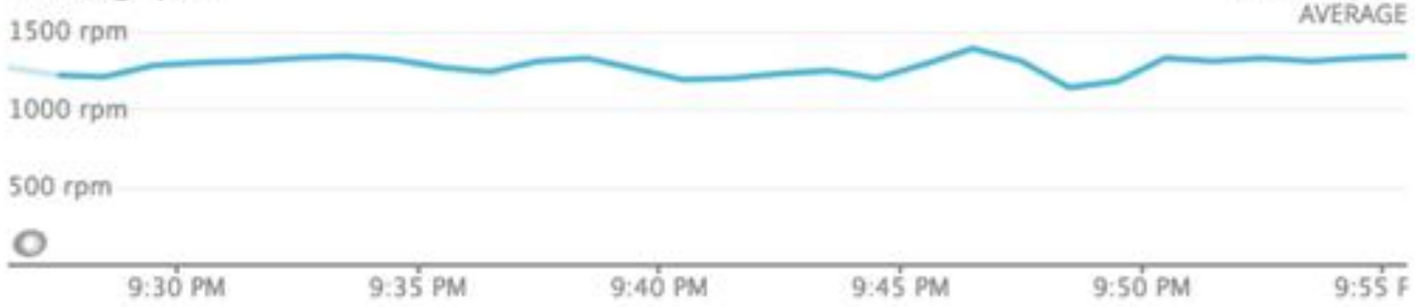
212 ms AVERAGE



validate | Redis get | lex | Post/topics | CommentEdge/node | Other

Throughput

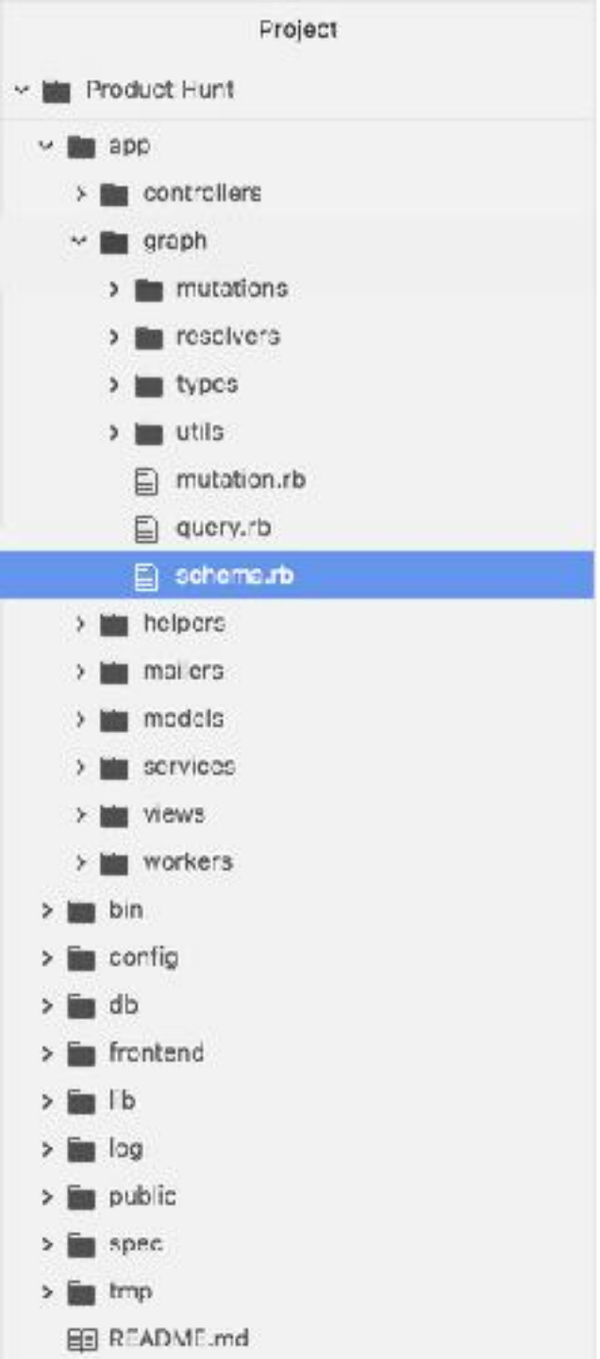
1,280 rpm AVERAGE



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




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

  use GraphQL::Tracing::NewRelicTracing

  #...
end
```

Windows

A fatal exception 0E has occurred at 0A20:CAB11E3D in VXD VMM(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


SENTRY

production-web

Issues | Overview | User Feedback | Releases | Settings

NoMethodError Frontend::GraphqlController#index

undefined method `value' for nil:NilClass | ruby

ISSUE #	EVENTS	USERS	ASSIGNEE
PRODUCTION-WEB-2J6	3	1	

Resolve | Ignore | 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)

|< | Older | Newer | >|

TAGS

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
```

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

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? 🤔


```ruby
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
```


```ruby
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

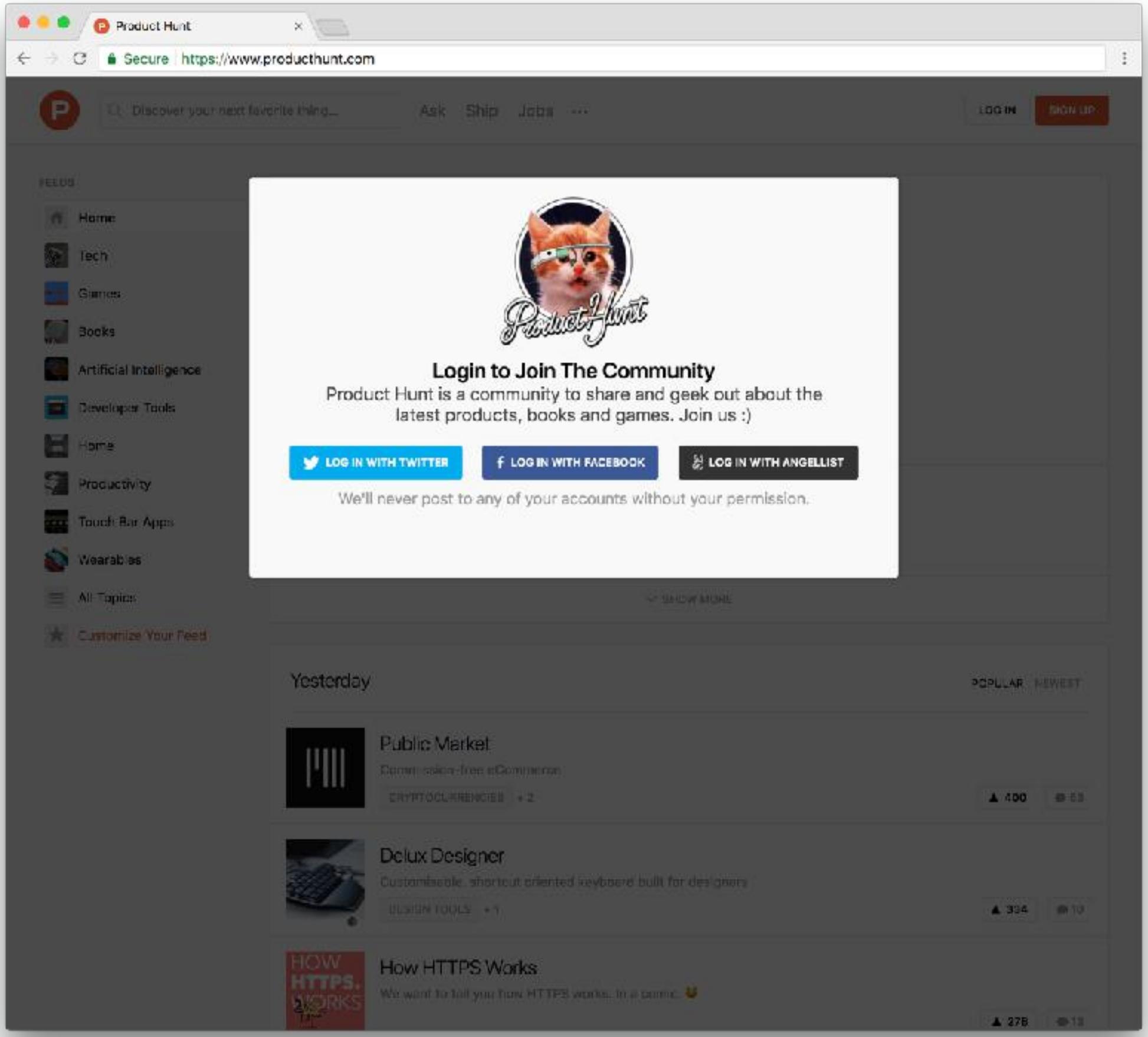
ADDITIONAL DATA

Formatted | Raw

query

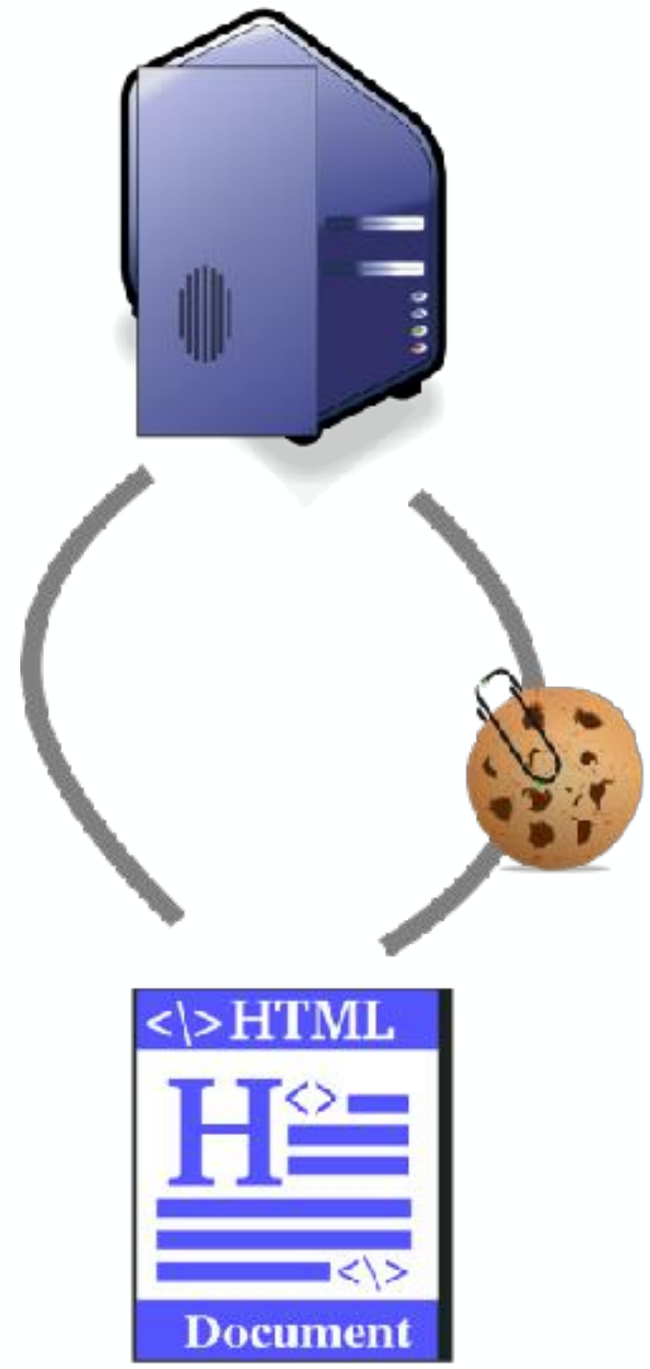
```
mutation CreateComment($commentsThreadRepliesCursor:
String = "", $input: CreateCommentInput!) {
  response: createComment(input: $input) {
    node {
      _id
      id
      subject {
        _id
        id
        ... on Commentable {
          _id
          id
          comments_count
          __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
```

Authentication & Authorization

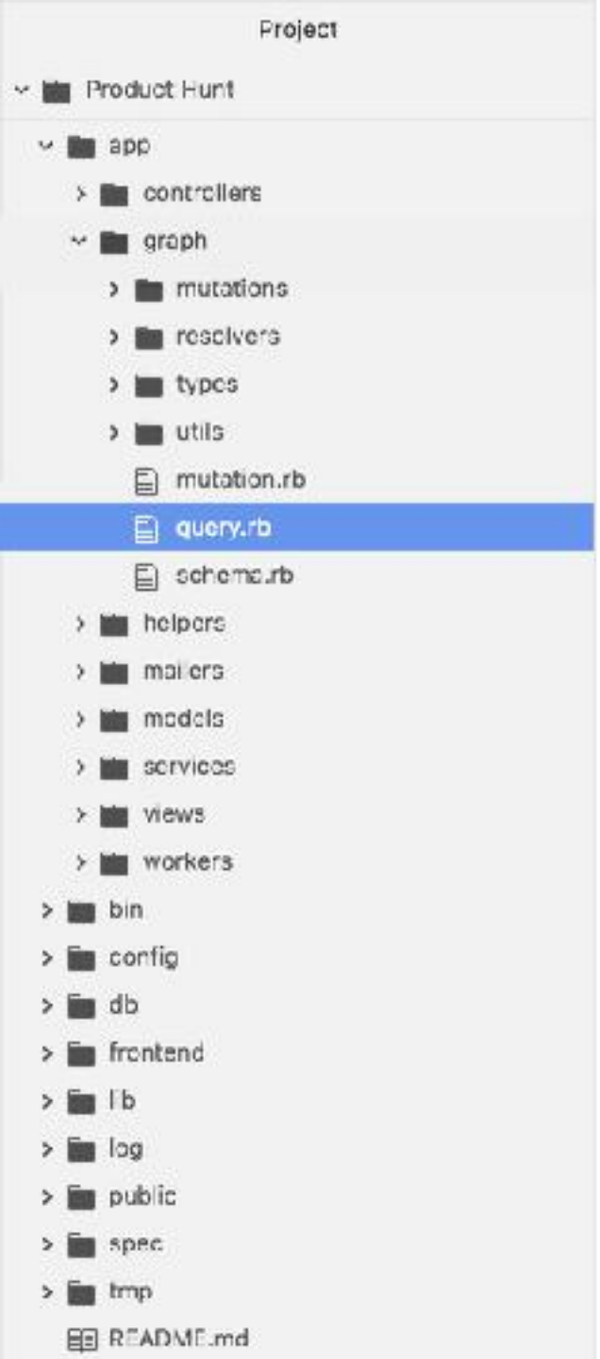

Product Hunt
Secure https://www.producthunt.com
Discover your next favorite thing...
Ask
Ship
Jobs
LOG IN
SIGN UP
FEEDS
Home
Tech
Games
Books
Artificial Intelligence
Developer Tools
Home
Productivity
Touch Bar Apps
Wearables
All Topics
Customize Your Feed
Login to Join The Community
Product Hunt is a 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 ANGELLIST
We'll never post to any of your accounts without your permission.
SHOW MORE
Yesterday
POPULAR
NEWEST
Public Market
CRYPTOCURRENCIES + 2
▲ 400
Delux Designer
DESIGN TOOLS + 1
▲ 334
How HTTPS Works
▲ 278

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

SESAME STREET .org


HTML
Document

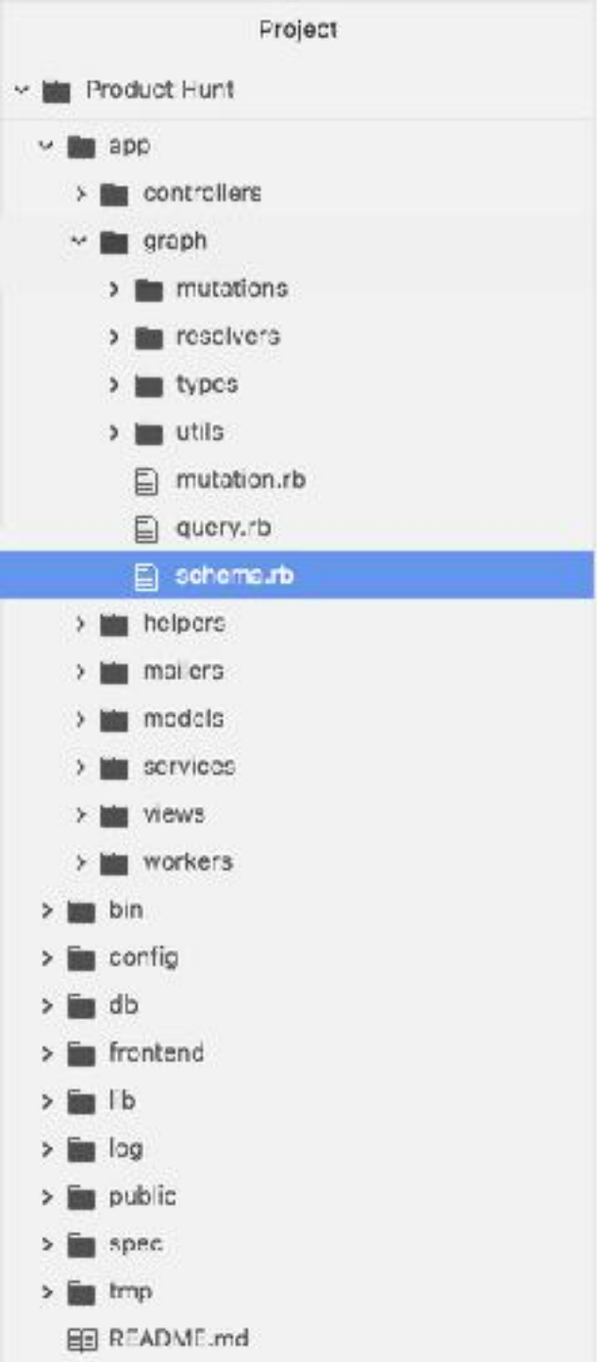

```ruby
class Frontend::GraphqlController < Frontend::BaseController
  before_action :ensure_query

  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
```



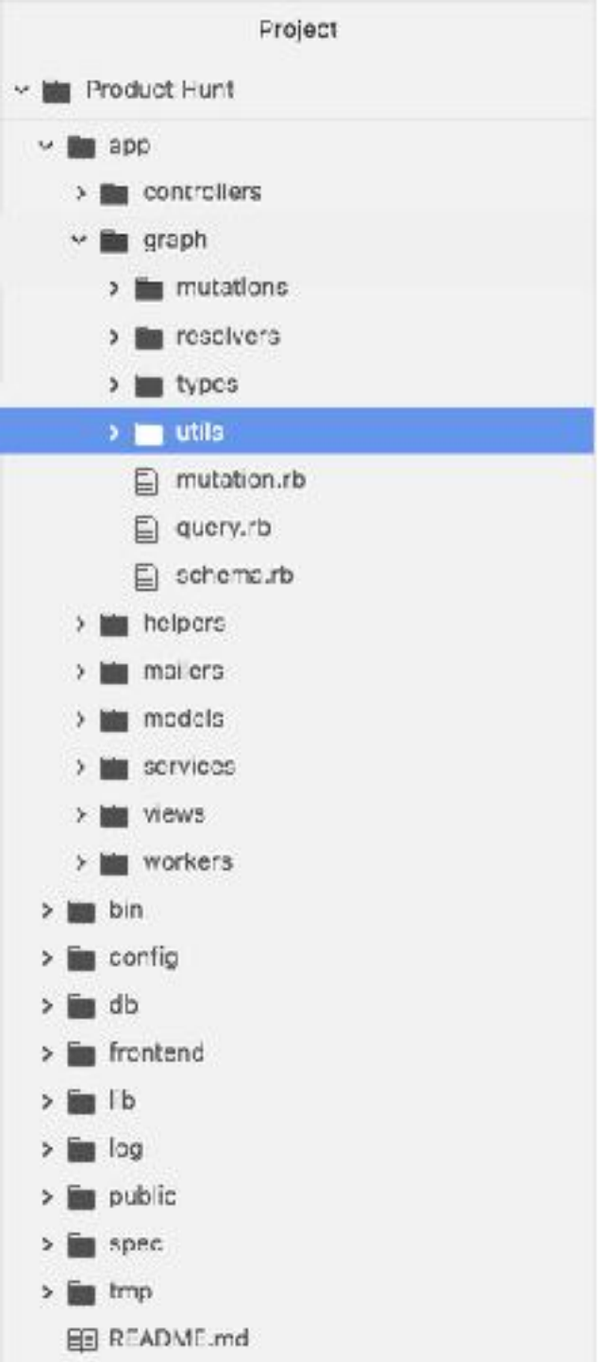

```ruby
Graph::Query = GraphQL::ObjectType.define do
  name 'Query'

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

  # ...
end
```

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


AUTHORIZED

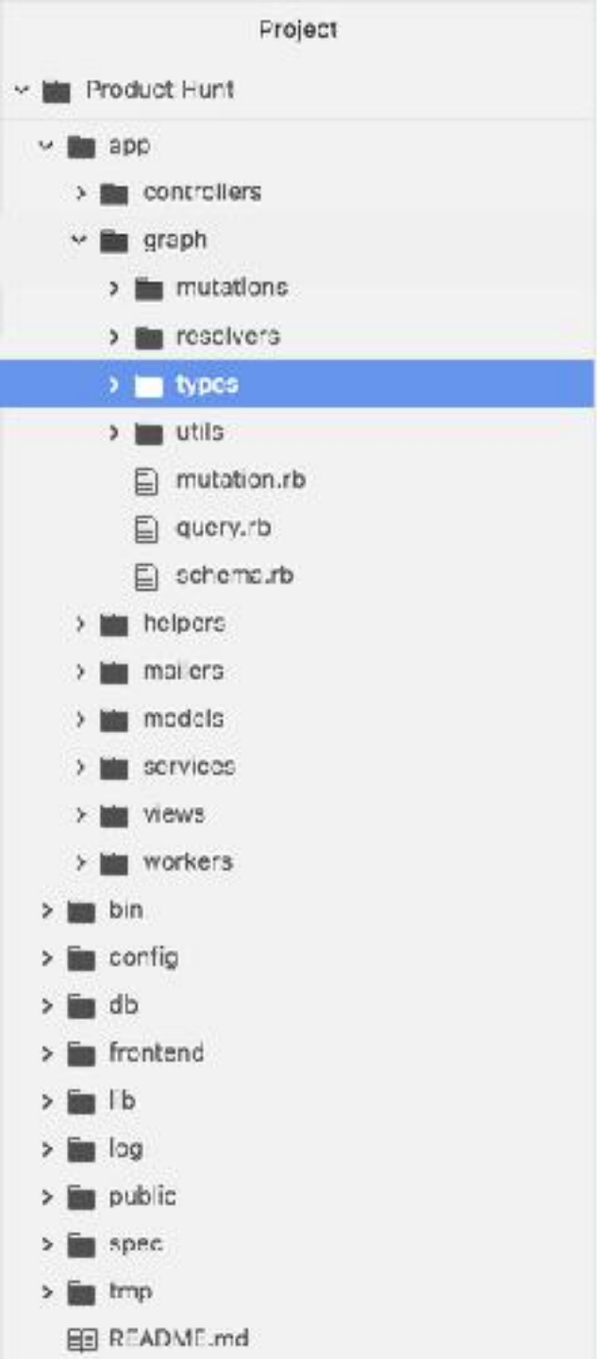



```ruby
Graph::Schema = GraphQL::Schema.define do
  # ...

  authorization Graph::Utils::AuthorizationStrategy

  instrument :field, Graph::Utils::FallbackInstrumenter

  # ...
end
```




```ruby
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
```


```ruby
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: [], functio

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

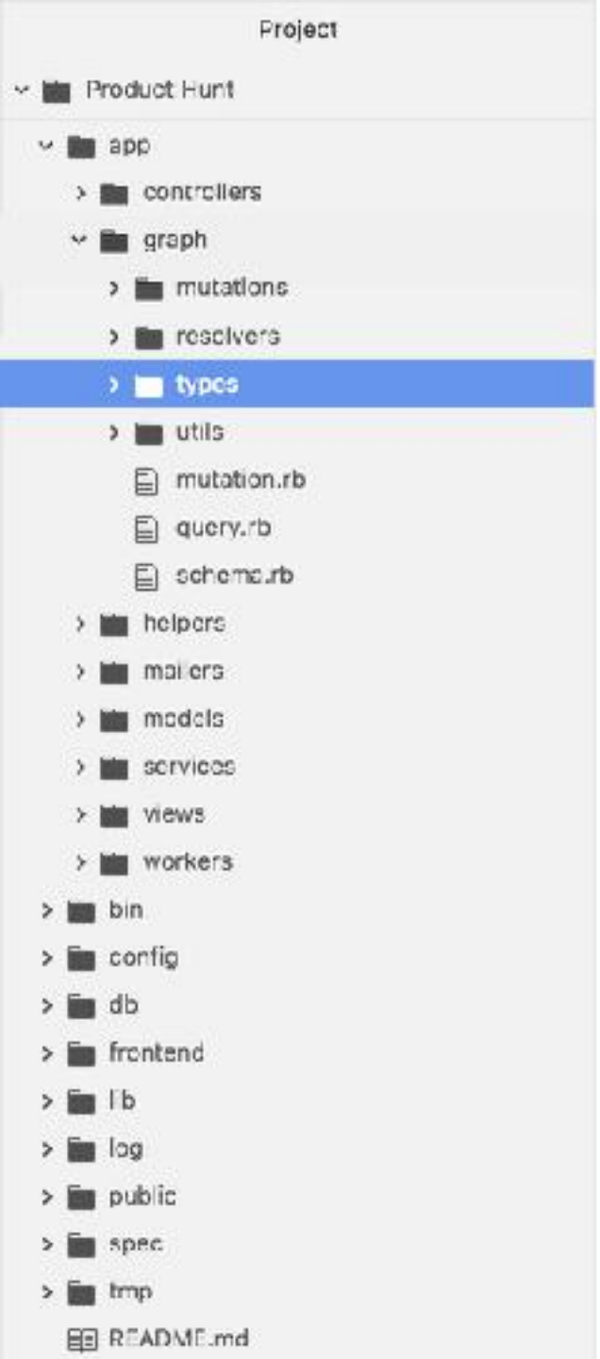



```ruby
Graph::Types::SurveyType = GraphQL::ObjectType.define do
  name 'Survey'

  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: [], functio

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


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

  authorize Authorization::READ
```

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

  # ...
end
```

```
urveys::QuestionsResolver.new
E_FIELD, fallback: [], functio

anResolver.new(Authorization::
```




```ruby
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: [], functio

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


```ruby
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: [], functio

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


```ruby
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
```


```ruby
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

▲ 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

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

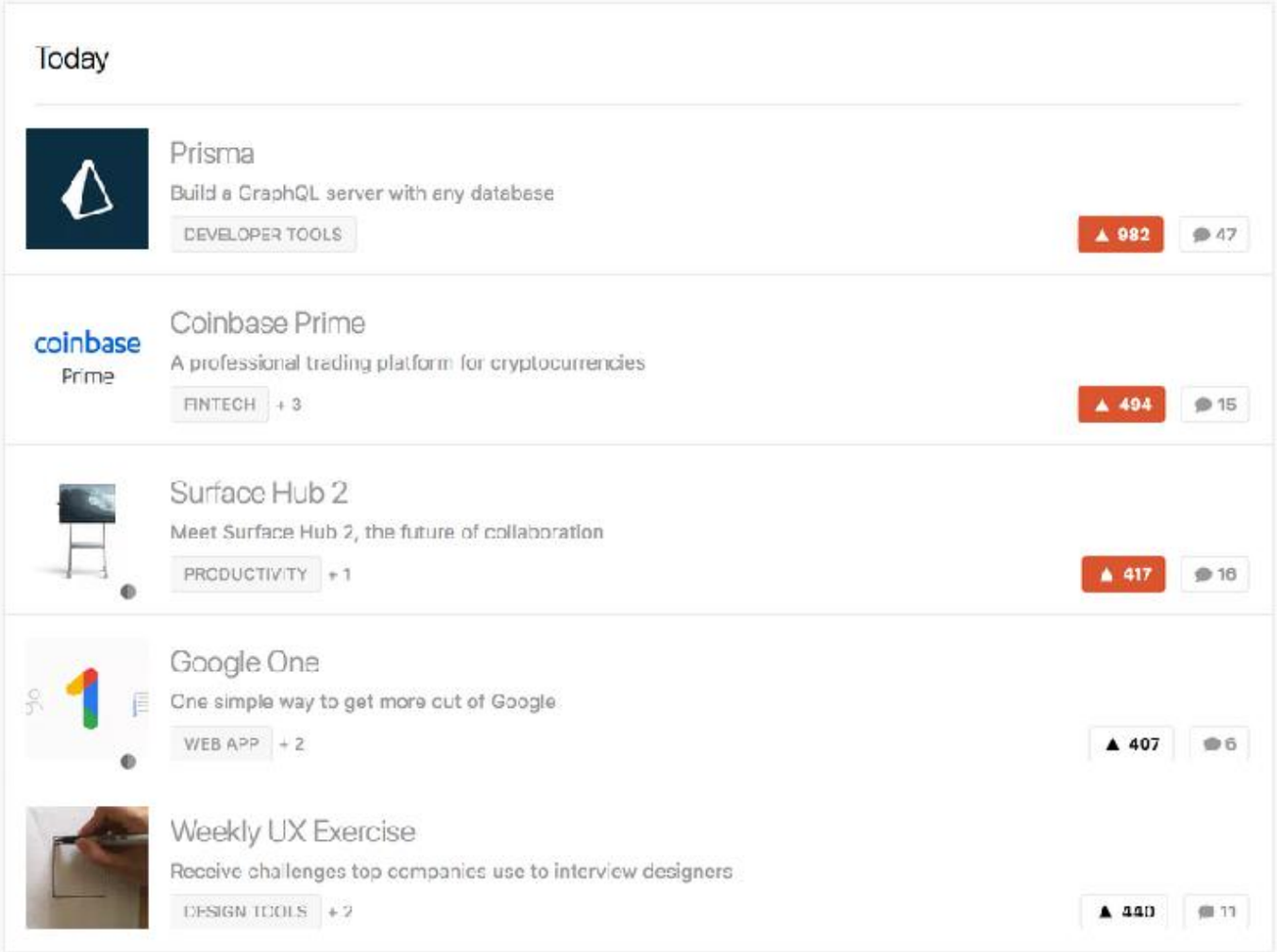
Weekly UX Exercise

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
  }
}
```



```
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
}]
```

```
[{                                  SELECT * FROM posts WHERE DATE(featured_at) = {date}
  id: 1,
  topics: [{ id: 1 }],                SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_id}
  isVoted: true                       SELECT * FROM votes WHERE post_id={post.id} and user_id={user_id}
}, {
  id: 2,
  topics: [{ id: 2 }],                SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_id}
  isVoted: true                       SELECT * FROM votes WHERE post_id={post.id} and user_id={user_id}
}, {
  id: 3,
  topics: [{ id: 2 }],                SELECT * FROM topics JOIN post_topics WHERE post_id IN {post_id}
  isVoted: true                       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]
```

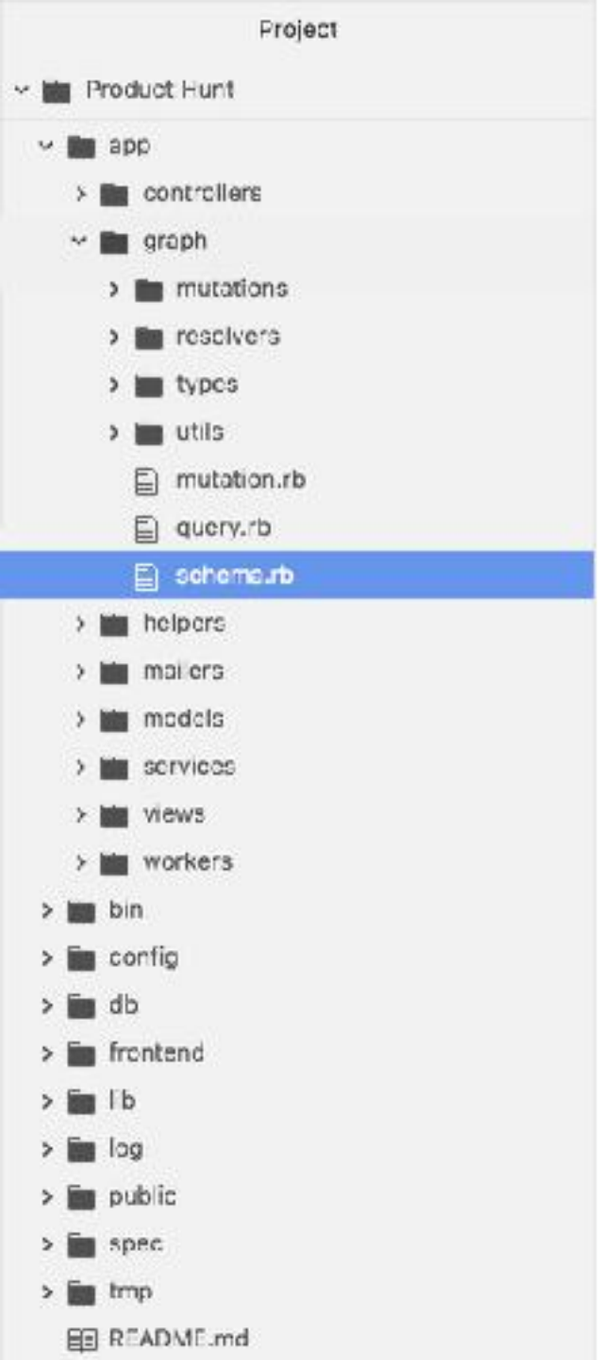
```
[{
  id: 1,
  topics: [find topics for post 1],
  isVoted: [is post 1 voted]
}, {
  id: 2,
  topics: [find topics for post 2],
  isVoted: [is post 2 voted]
}, {
  id: 3,
  topics: [find topics for post 3],
  isVoted: [is post 3 voted]
}]
```

```
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"
```

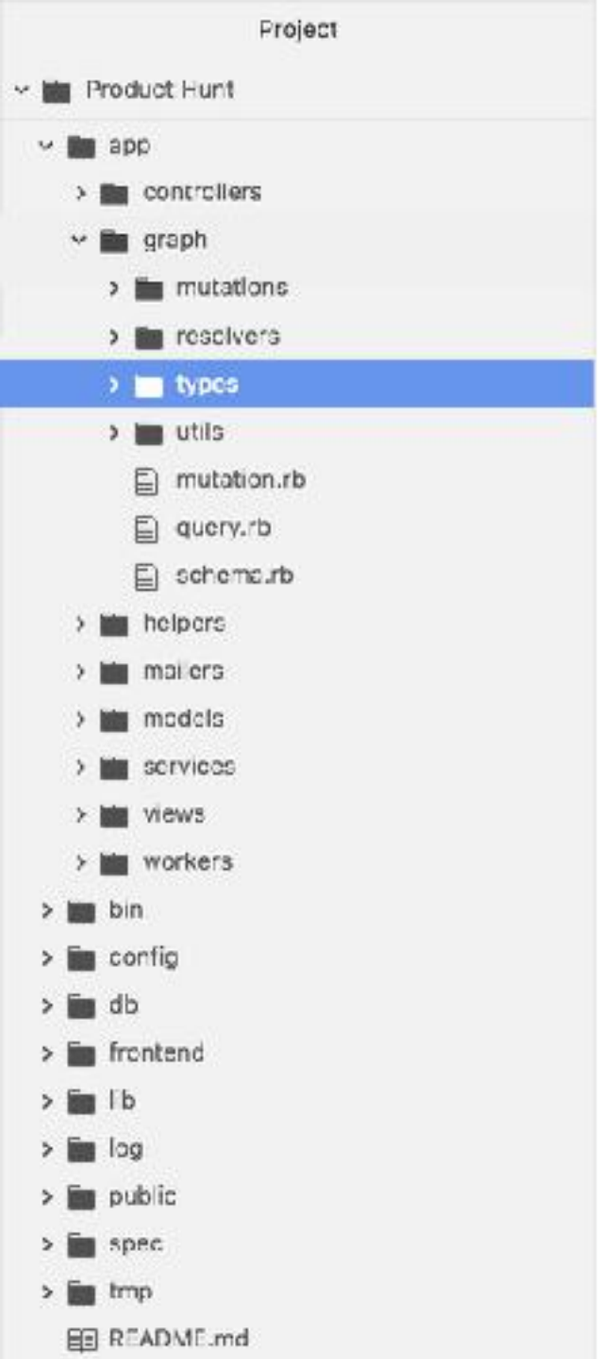



```ruby
Graph::Schema = GraphQL::Schema.define do
  # ...

  use GraphQL::Batch

  # ...
end
```

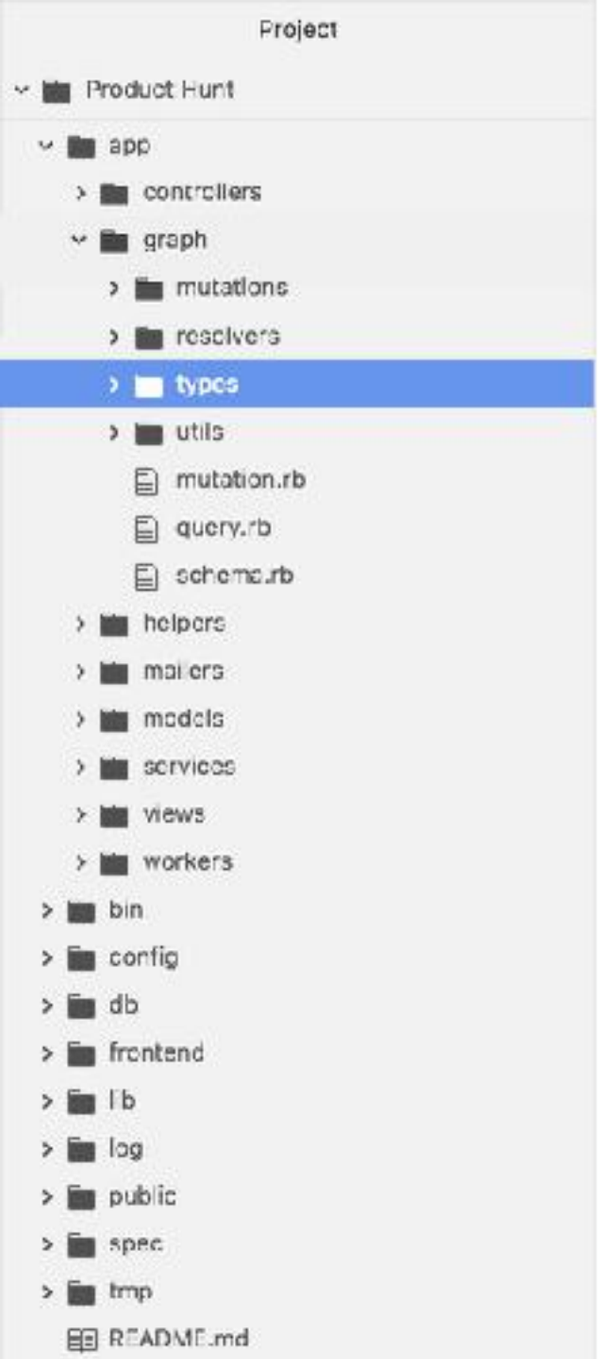
```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
    votesCount
    commentsCount
    topics {
      id
      name
    }
    isVoted
  }
}
```

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

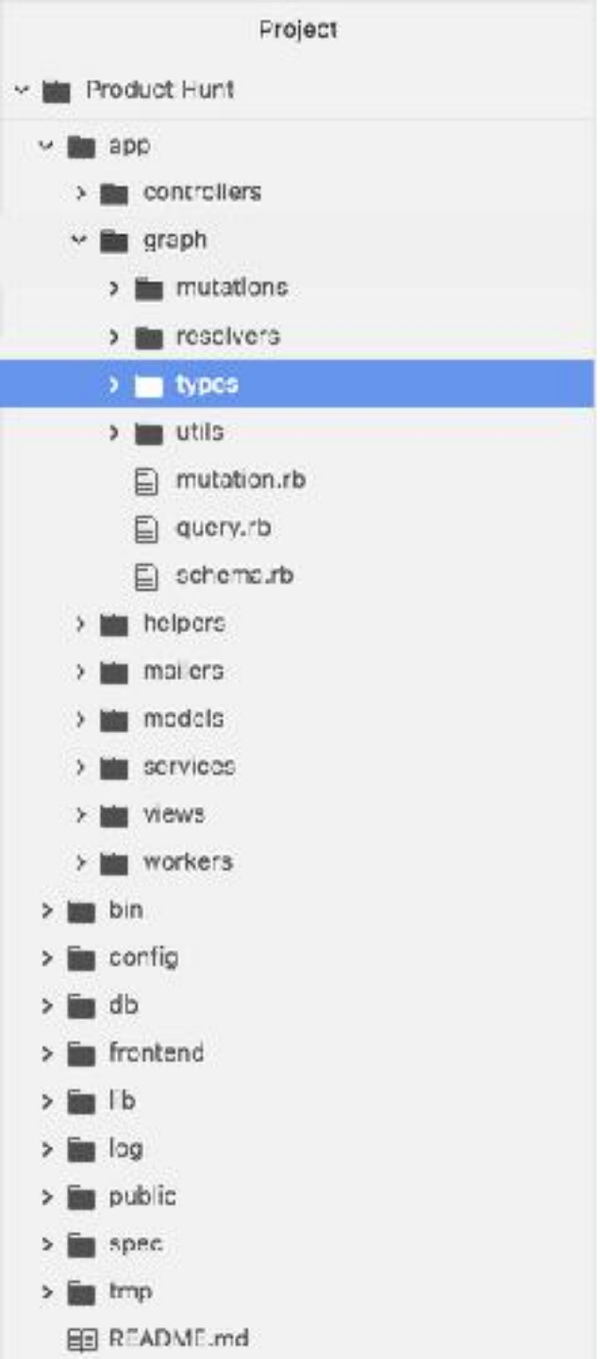

```ruby
class Graph::Resolvers::Posts::TopicsResolver < GraphQL::Function
  type !types[!Graph::Types::TopicType]

  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
```




```ruby
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
```



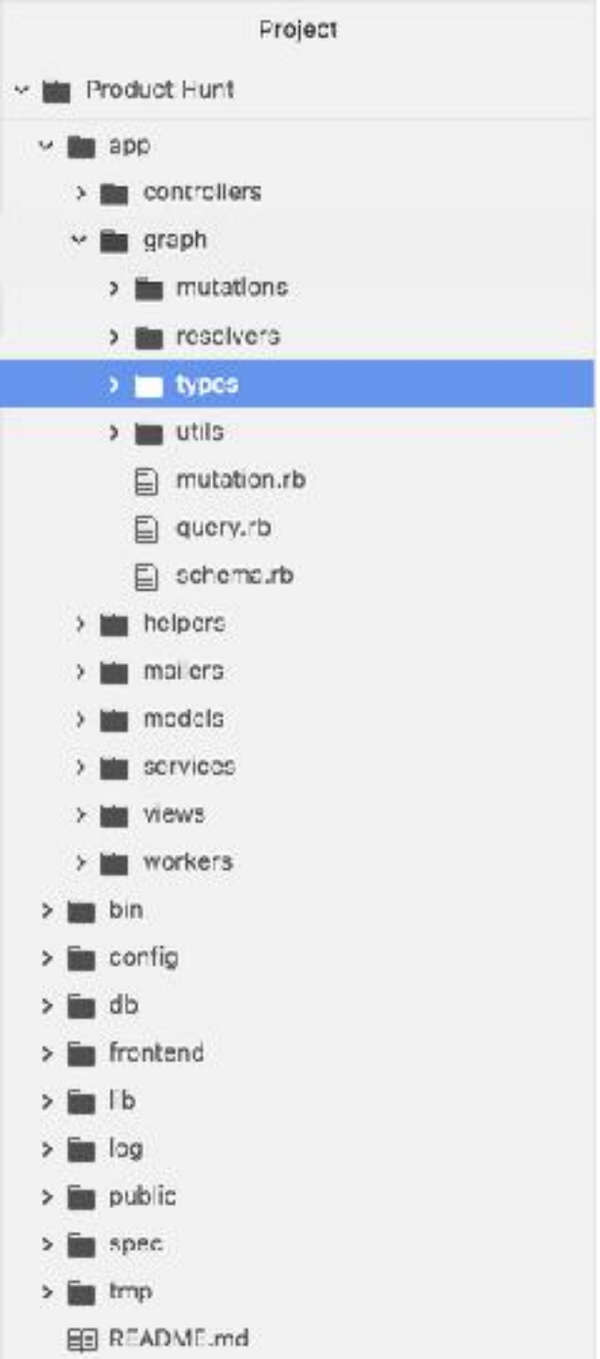

```ruby
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

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




```ruby
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

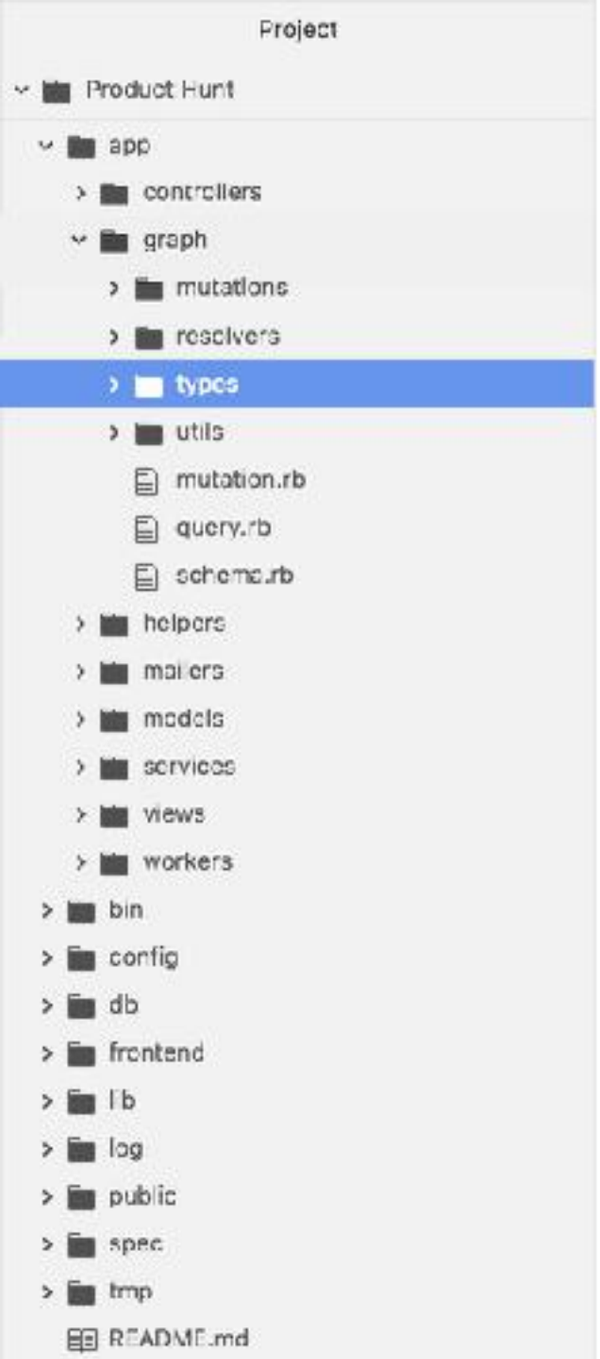



```ruby
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

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

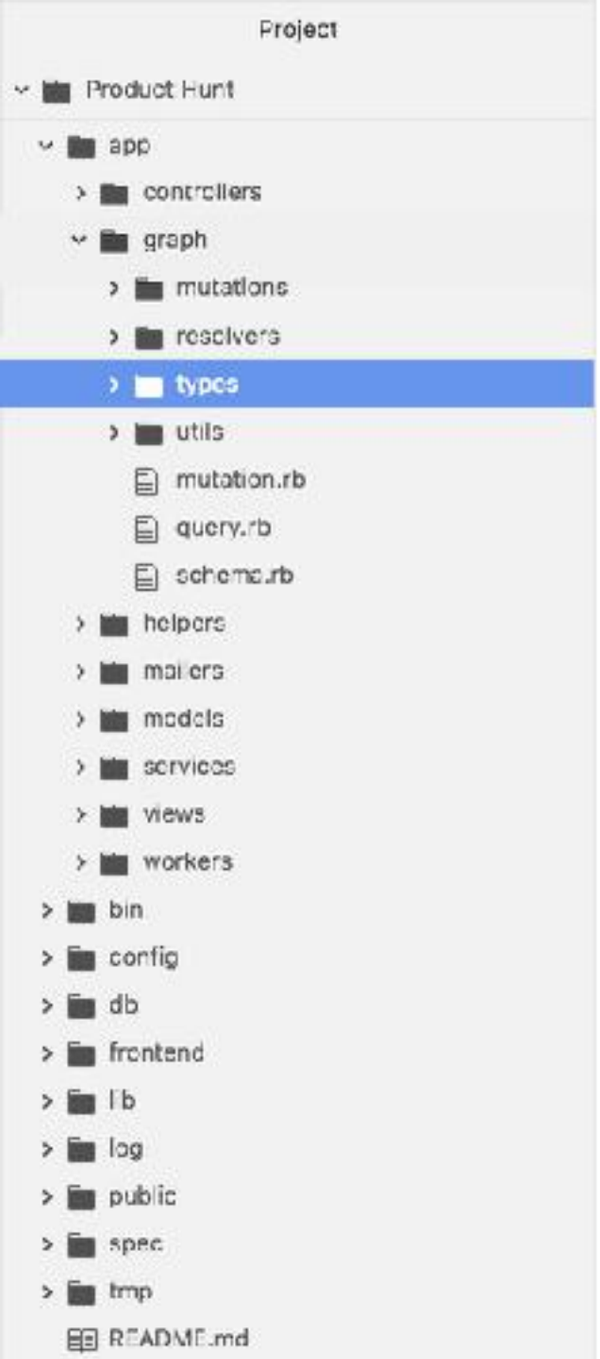
```
query {
  posts(date: '2018-05-05') {
    id
    name
    tagline
    votesCount
    commentsCount
    topics {
      id
      name
    }
    isVoted
  }
}
```

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


```ruby
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(:pos
      posts.each do |post|
        fulfill post, voted_posts_ids.include?(post.id)
      end
    end
  end
end
```

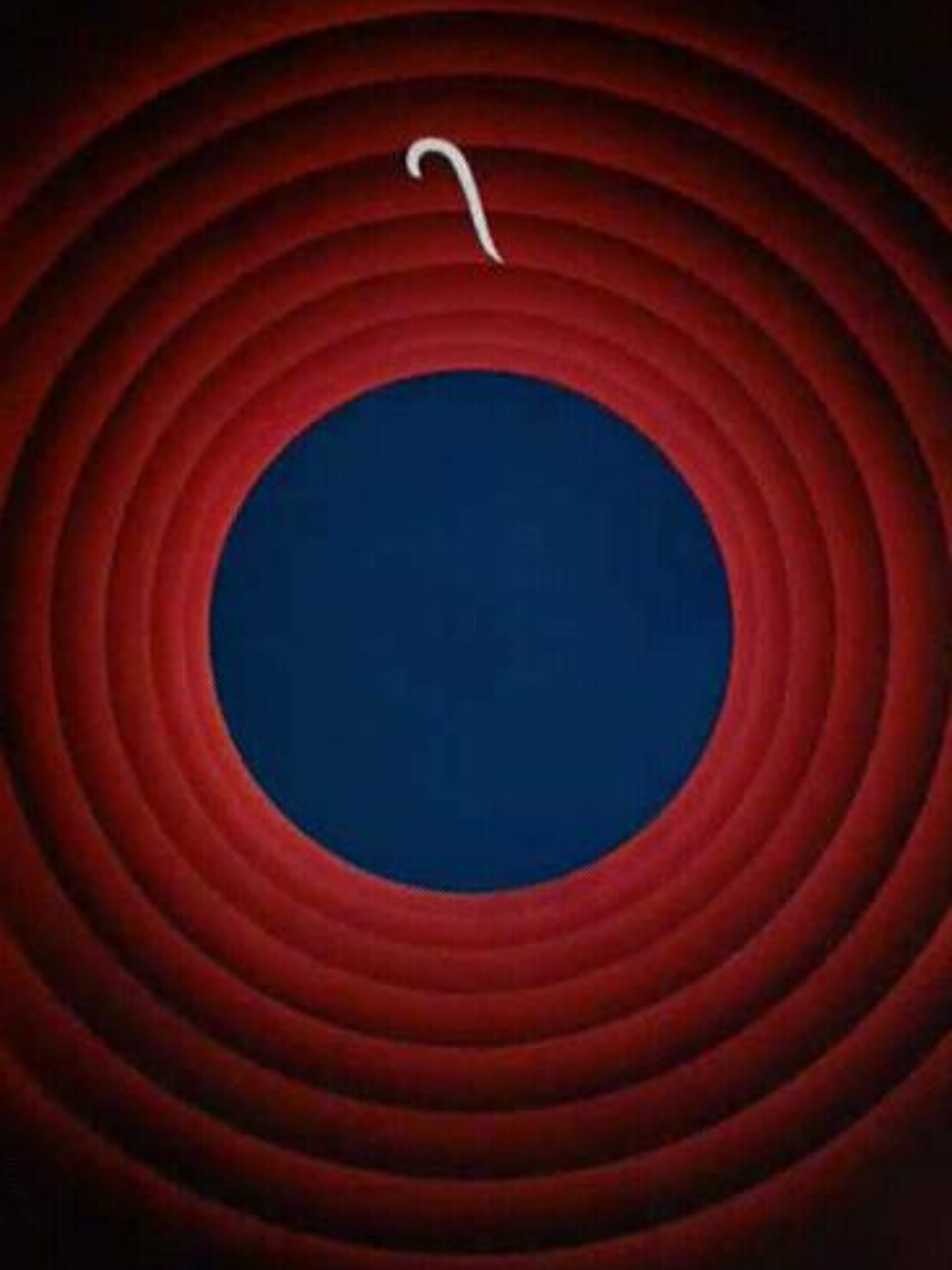


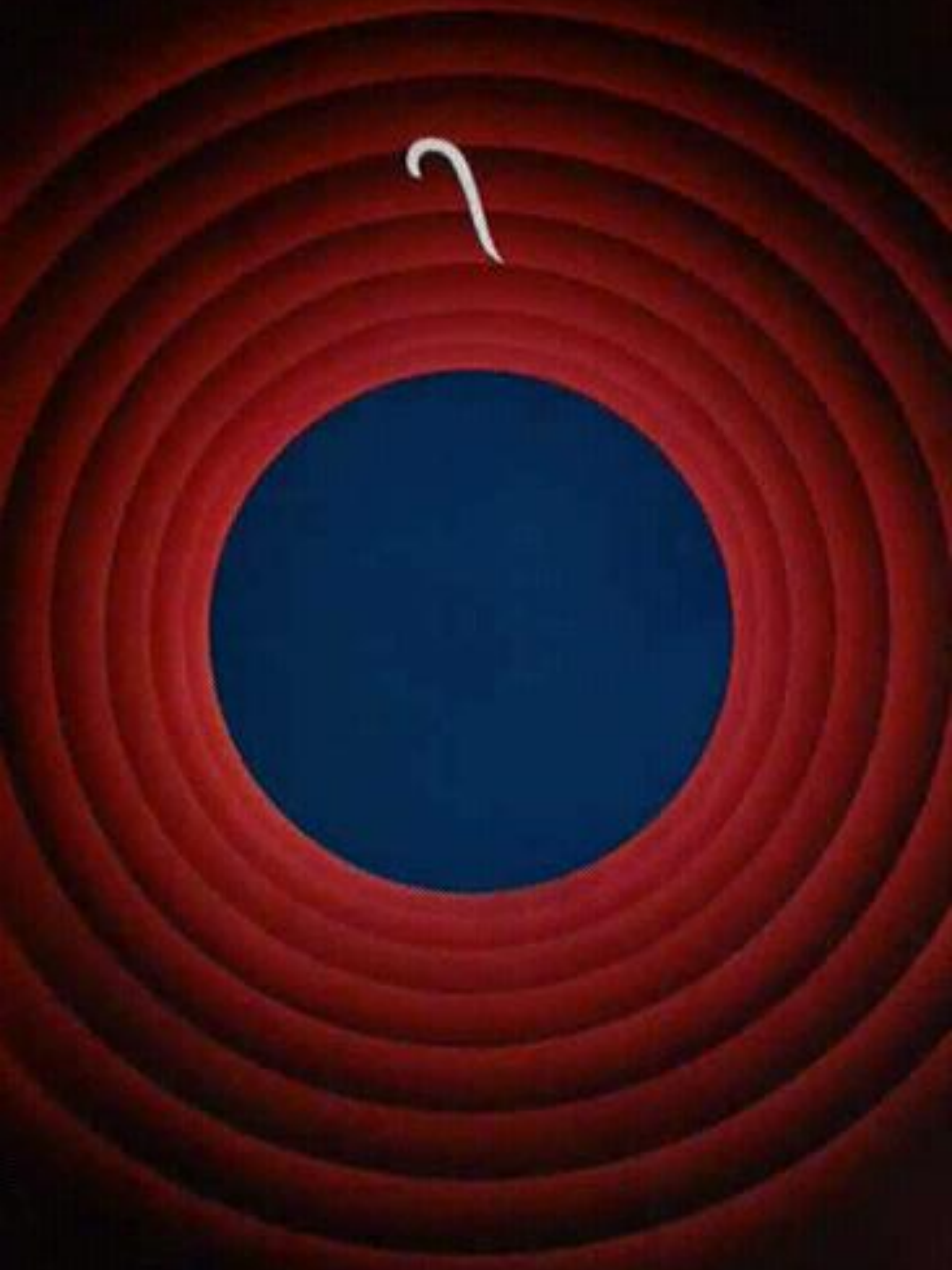

```ruby
Graph::Types::PostType = GraphQL::ObjectType.define do
  name 'Post'

  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
```




```
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**