Rescuing legacy codebases with GraphQL

@nettofarah

Netto Farah

@nettofarah

Eng. Manager at

Segment

Budweiser

IFTTT

Context

- Millions of users
- Billions of API calls every day
- Website, iOS app, Android app

Tech Stack (at the time)

- Seasoned Rails 3 monolith app
- APIs v1, v2, v3, dev_api…
- Challenging to deploy/iterate/run tests
- sole web dev

Challenge:

Build an entirely new product

With a **9 months** deadline 😱

We knew we needed
to make some changes

Majestic Monoliths

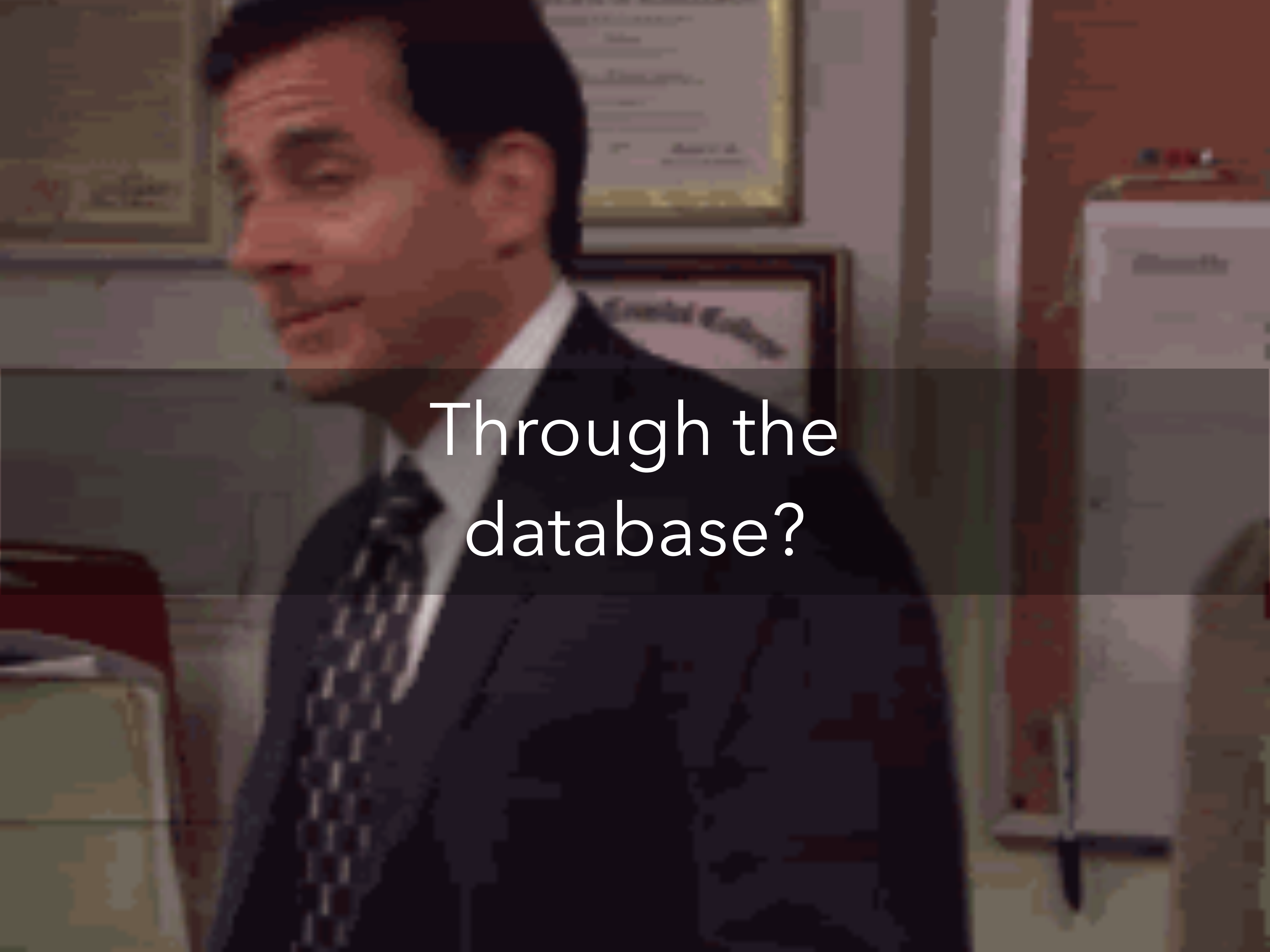
vs

Micro-services

Not a binary decision

A **hybrid** approach:
Rich API + **specific** clients

How can we make our frontend and backend apps **communicate**?



Through the database?

Why are **database-driven** integrations **tempting**?

Why are **database-driven** integrations **challenging**?

What about **APIs**?

Challenges with **Traditional** APIs

Multiple use cases

🚲 🚗 🚀

Different access pattern

📱 🖥️

Ambiguity

Solved with documentation
or conventions

👌🏽

[🇵🇭,🇺🇸,🇵🇹..🇨🇰] => ✅

👌🏽

[🇧🇷] =>

ask me later…

How can we
solve a few of these
challenges with APIs ?

Types

Ability to load **just** what we **need**

Always get **predictable** results

You know where
I’m going with this, right?

TYPES +

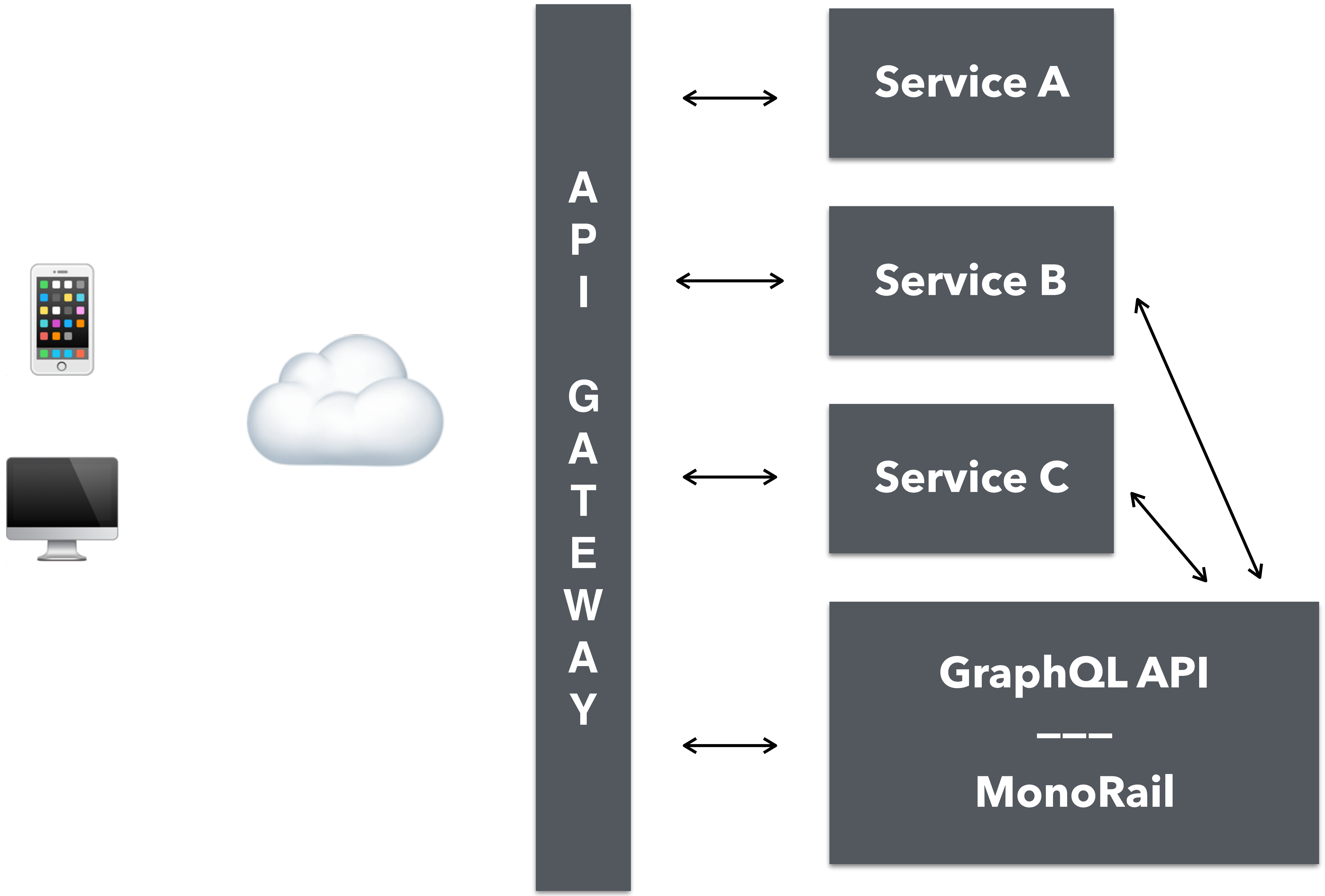
PREDICTABLE RESULTS +

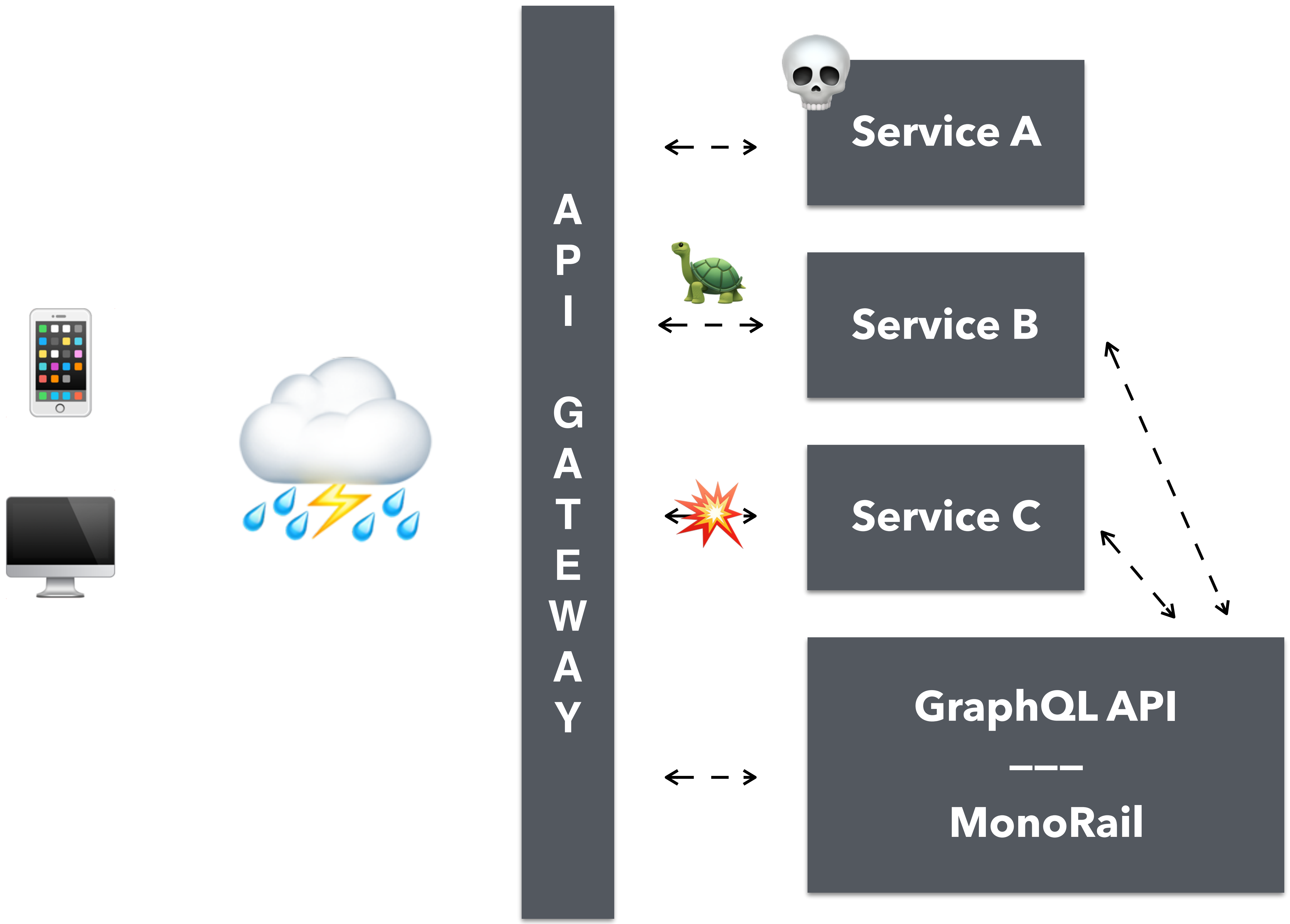
COMPOSABLE QUERIES

= GraphQL ❤️

We built a GraphQL API
on top of our monolith

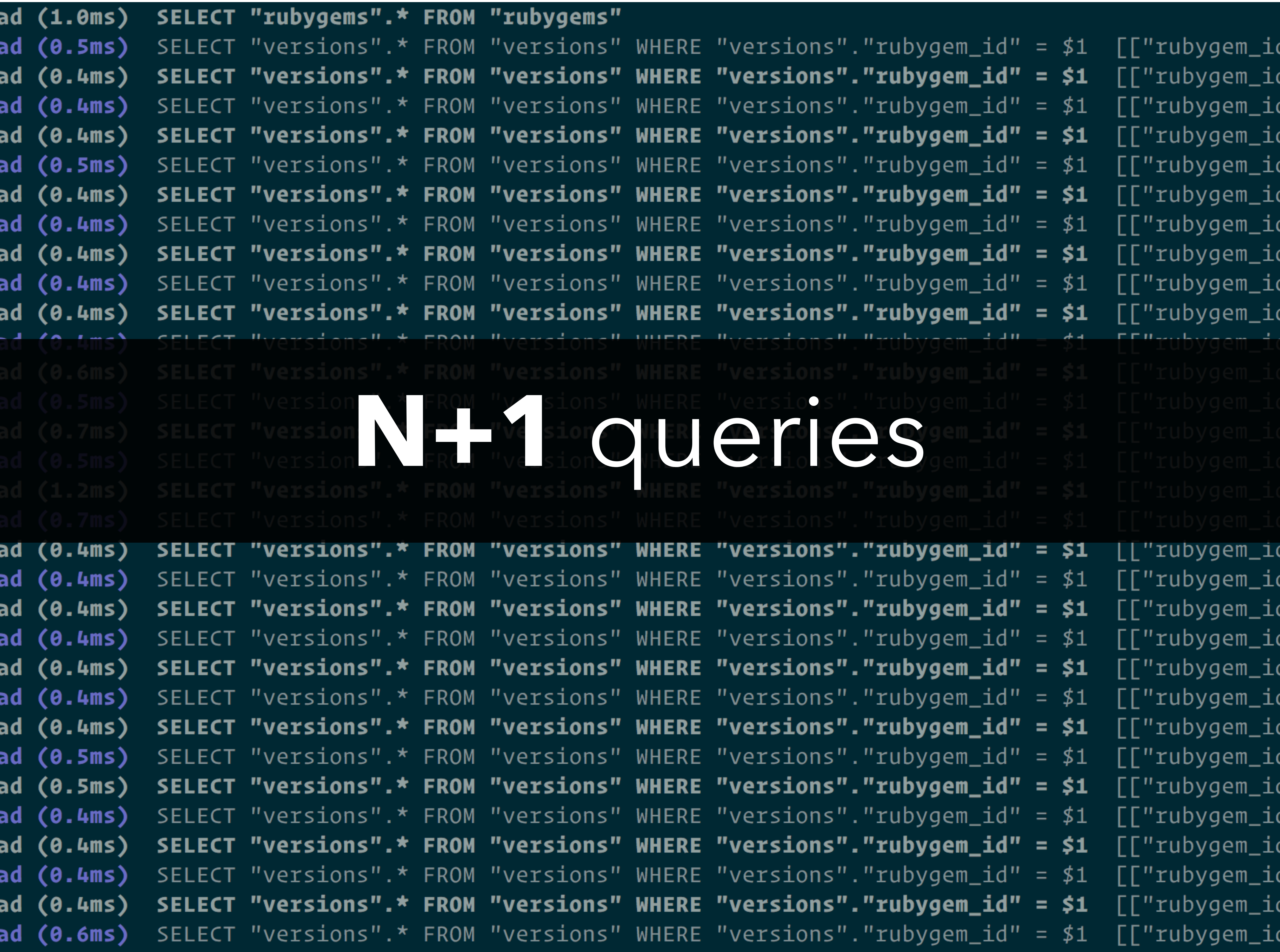
GraphQL API
as an **integration** layer
for multiple (not so micro) **services**


API GATEWAY
Service A
Service B
Service C
GraphQL API
———
MonoRail


API GATEWAY
Service A
Service B
Service C
GraphQL API
MonoRail

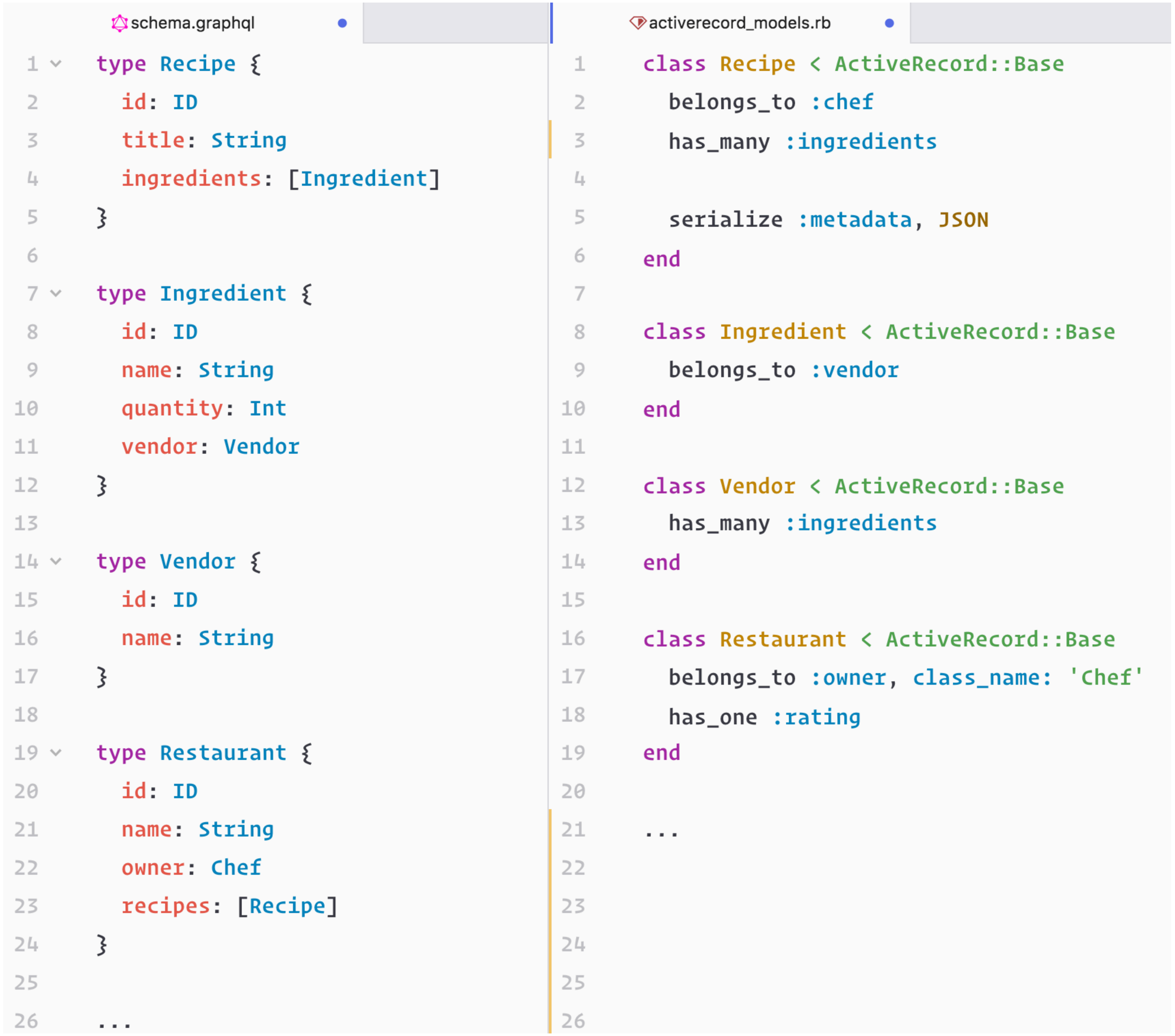
GraphQL (and Rails) in **production**

Challenge #1



N+1 queries

```
ad (1.0ms)  SELECT "rubygems".* FROM "rubygems"
ad (0.5ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.5ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.6ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.5ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.7ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.5ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (1.2ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.7ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.5ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.5ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.4ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
ad (0.6ms)  SELECT "versions".* FROM "versions" WHERE "versions"."rubygem_id" = $1  [["rubygem_i
```



schema.graphql

```graphql
type Recipe {
  id: ID
  title: String
  ingredients: [Ingredient]
}

type Ingredient {
  id: ID
  name: String
  quantity: Int
  vendor: Vendor
}

type Vendor {
  id: ID
  name: String
}

type Restaurant {
  id: ID
  name: String
  owner: Chef
  recipes: [Recipe]
}

...
```

activerecord_models.rb

```ruby
class Recipe < ActiveRecord::Base
  belongs_to :chef
  has_many :ingredients

  serialize :metadata, JSON
end

class Ingredient < ActiveRecord::Base
  belongs_to :vendor
end

class Vendor < ActiveRecord::Base
  has_many :ingredients
end

class Restaurant < ActiveRecord::Base
  belongs_to :owner, class_name: 'Chef'
  has_one :rating
end

...
```

```
query {
  recipes {
    title
    ingredients {
      name
      vendor { name }
    }
  }
}
```

```
SELECT "recipes".* FROM “recipes"

SELECT "ingredients".* FROM "ingredients" WHERE "ingredients"."recipe_id" = 1
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 1
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 2
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 3
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 4

SELECT "ingredients".* FROM "ingredients" WHERE "ingredients"."recipe_id" = 2
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 5
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 6

SELECT "ingredients".* FROM "ingredients" WHERE "ingredients"."recipe_id" = 3
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 7
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 8
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 9

SELECT "ingredients".* FROM "ingredients" WHERE "ingredients"."recipe_id" = 4
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 10
SELECT "vendors".* FROM "vendors" WHERE "vendors"."id" = 11
```

```sql
SELECT "ingredients".* FROM "ingredients" WHERE
"ingredients"."recipe_id" = 1

SELECT * FROM "vendors" WHERE "id" = 1
SELECT * FROM "vendors" WHERE "id" = 2
SELECT * FROM "vendors" WHERE "id" = 3
SELECT * FROM "vendors" WHERE "id" = 4
```

How do people
usually solve this problem?

DataLoader

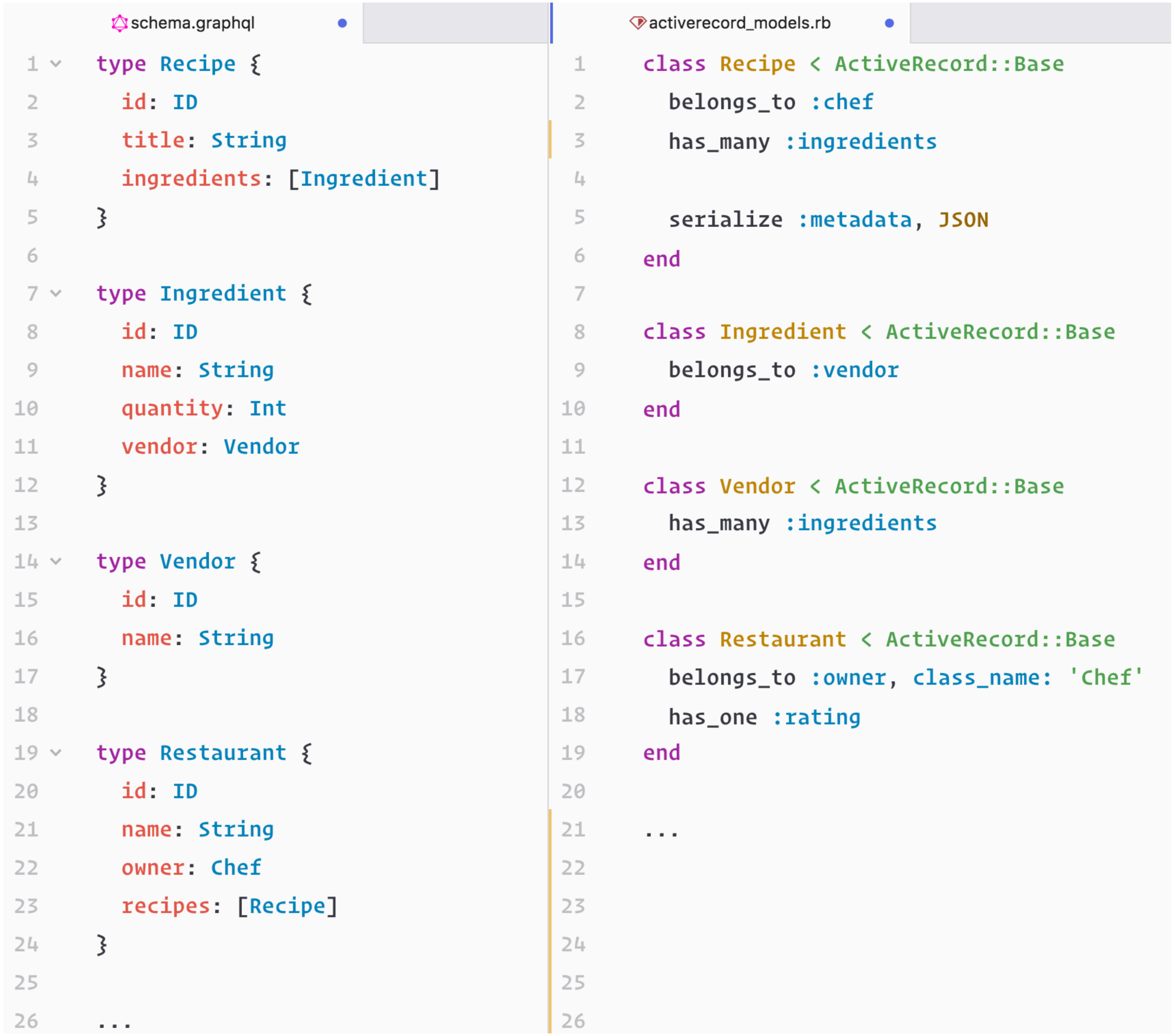
but that's a javascript only tool

😔

GraphQL-Batch

```
resolve -> (obj, args, context) do
  Loader.for(Product).load(args["id"]).then do |p|
    Loader.for(Image).load(p.image_id)
  end
end
```

Let’s take a second look at our data models



```
type Recipe {
  id: ID
  title: String
  ingredients: [Ingredient]
}

type Ingredient {
  id: ID
  name: String
  quantity: Int
  vendor: Vendor
}

type Vendor {
  id: ID
  name: String
}

type Restaurant {
  id: ID
  name: String
  owner: Chef
  recipes: [Recipe]
}

...
```

```
class Recipe < ActiveRecord::Base
  belongs_to :chef
  has_many :ingredients

  serialize :metadata, JSON
end

class Ingredient < ActiveRecord::Base
  belongs_to :vendor
end

class Vendor < ActiveRecord::Base
  has_many :ingredients
end

class Restaurant < ActiveRecord::Base
  belongs_to :owner, class_name: 'Chef'
  has_one :rating
end

...
```

```ruby
Recipe.all.includes({
  ingredients: 'vendor'
})
```

```
query {
  recipes {
    title
    ingredients {
      name
      vendor
    }
  }
}
```

```
Recipe.all.includes({
  ingredients: 'vendor'
})
```

```sql
SELECT "recipes".* FROM "recipes"

SELECT "ingredients".* FROM “ingredients”
 WHERE “ingredients"."recipe_id"
 IN (1, 2, 3, 4)

SELECT "vendors".* FROM “vendors"
 WHERE “vendors"."id"
 IN (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
```

github.com/nettofarah/graphql-query-resolver

📉 ~60% **reduction**

in database IOPS

=

💰💰💰

Selectively choosing our Database **environment**

```ruby
if includes_mutation?(query)
  DatabaseSelection.use_main_database do
    GraphQL.execute_query(query)
  end
else
  DatabaseSelection.use_readonly_replica do
    GraphQL.execute_query(query)
  end
end
```

☠️ Eliminated contention locks

Lessons

#1 Figure out **batching** as **early** as you can

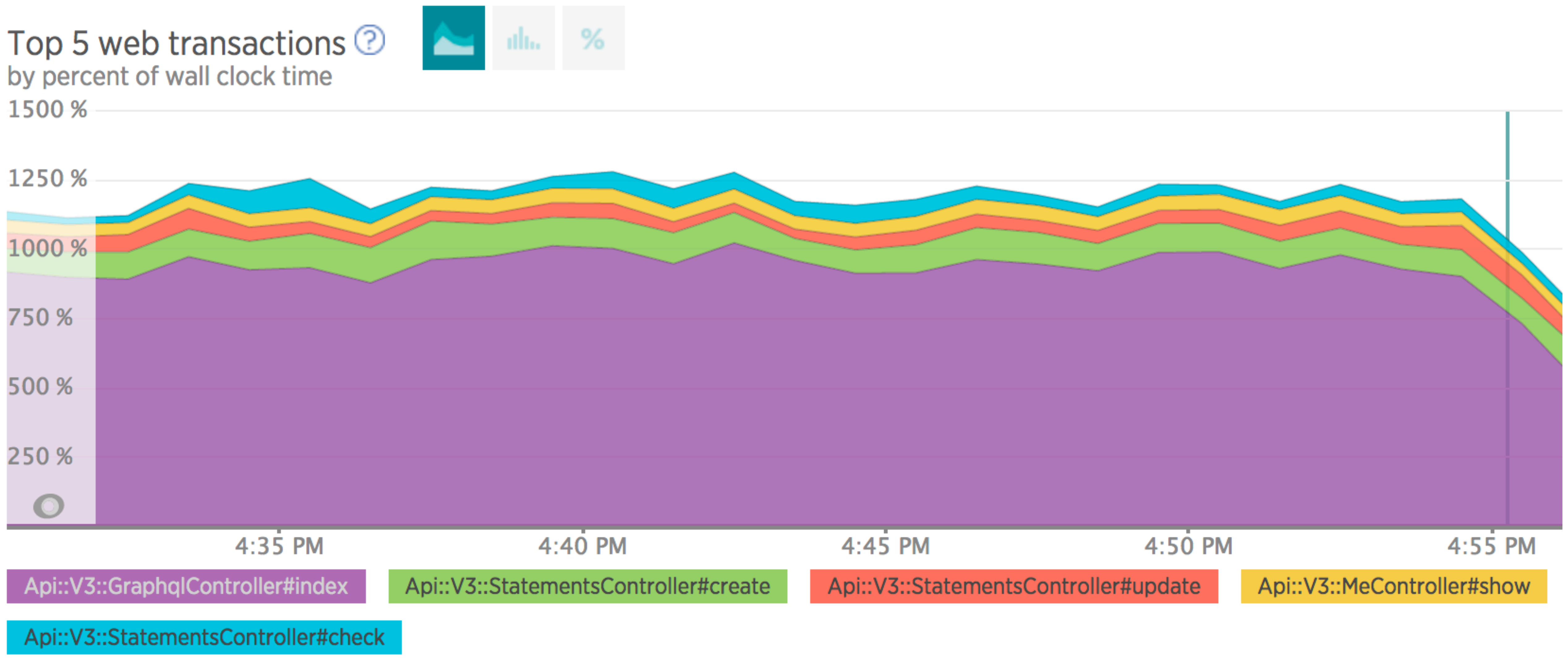
#2 **Leverage** GraphQL **types**

Challenge #2

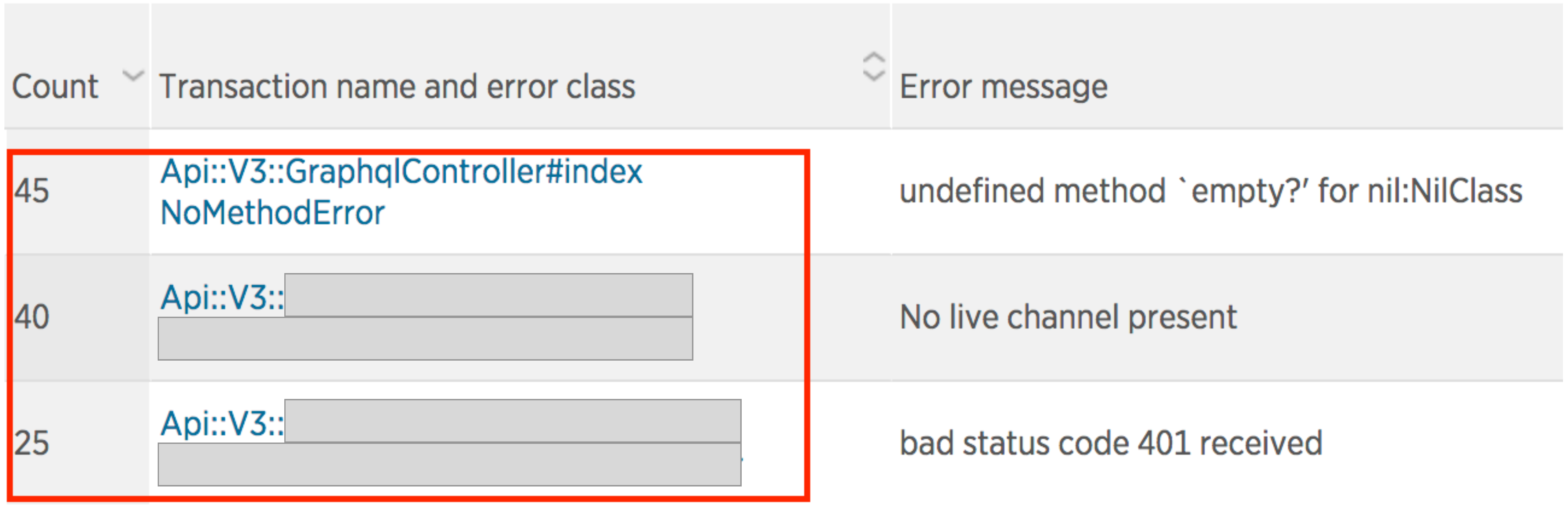
Monitoring and **Errors**

This is not really useful

🙄



What’s up with my errors? 🤔



Count	Transaction name and error class	Error message
45	Api::V3::GraphqlController#index NoMethodError	undefined method `empty?’ for nil:NilClass
40	Api::V3::[redacted]	No live channel present
25	Api::V3::[redacted]	bad status code 401 received

Lesson

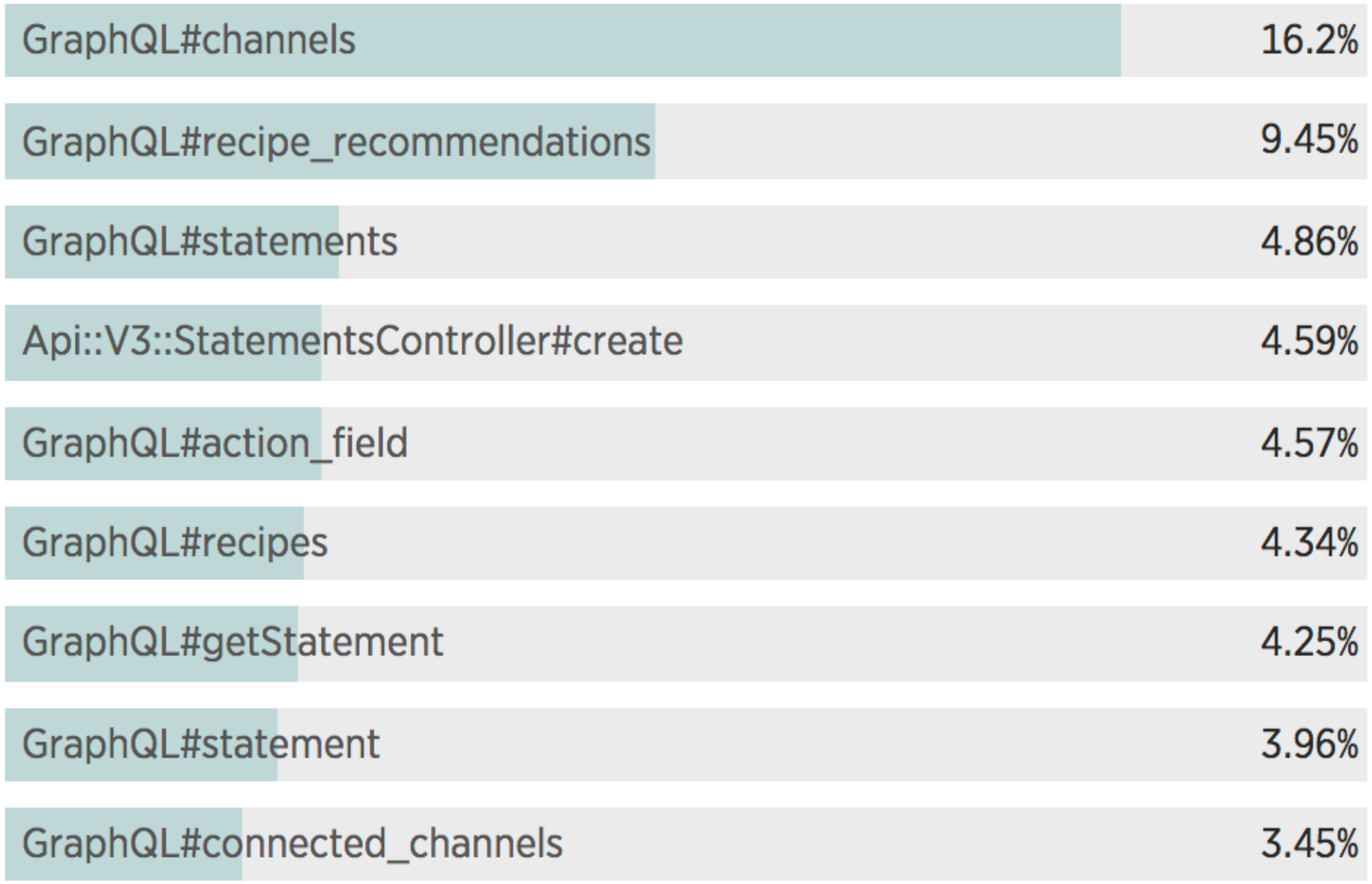
#2 **Leverage** GraphQL **types**

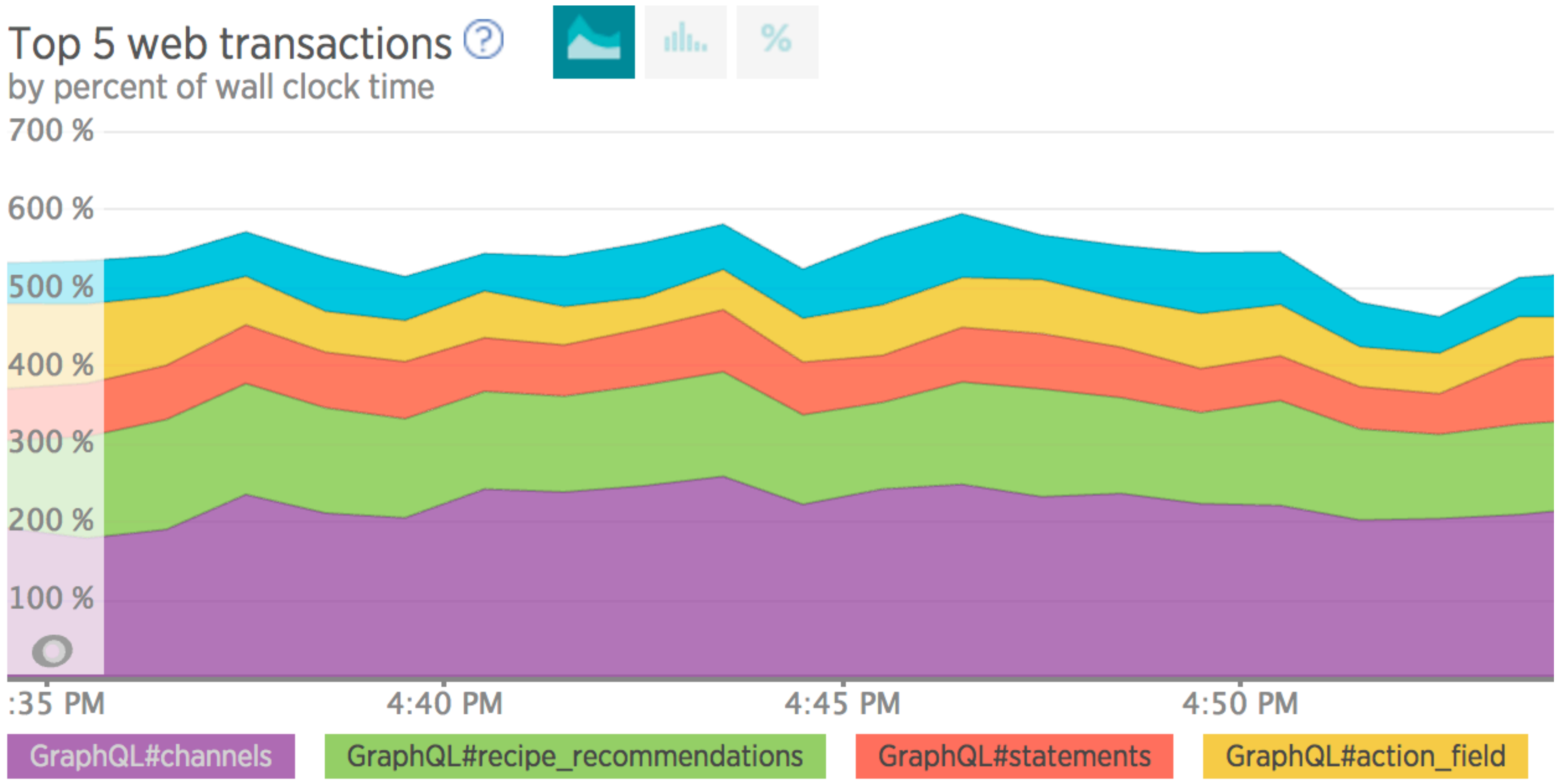
(again)

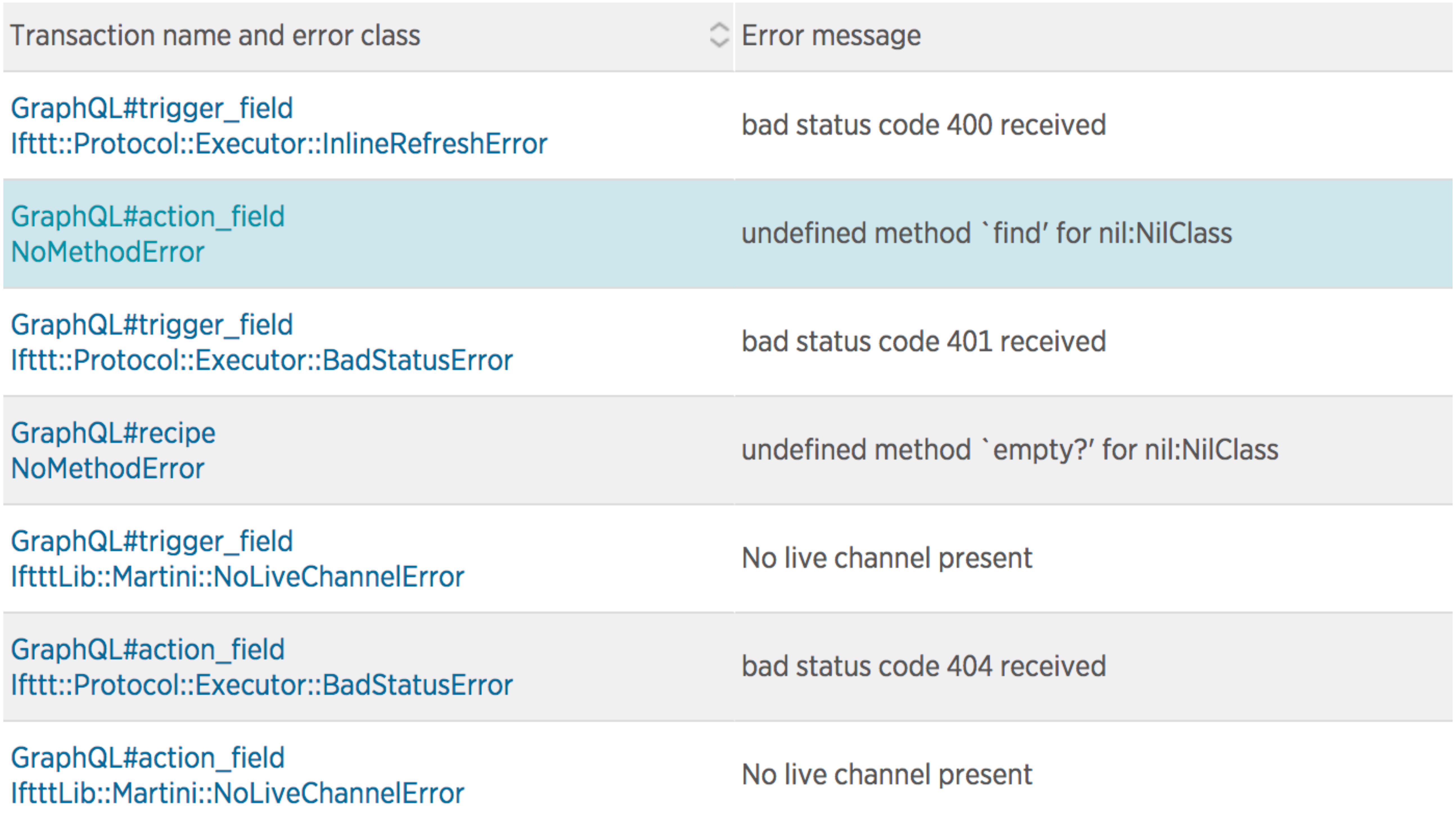
```
# At the query level
NewRelic::Agent.set_transaction_name(query_name)


# At the field level
new_resolver = -> (obj, args, ctx) {
  name = ["GraphQL/field/#{type.name}.#{field.name}"]

  NewRelic.trace_execution_scoped(name) do
    old_resolver.call(obj, args, ctx)
  end
}
```


GraphQL#channels 16.2%
GraphQL#recipe_recommendations 9.45%
GraphQL#statements 4.86%
Api::V3::StatementsController#create 4.59%
GraphQL#action_field 4.57%
GraphQL#recipes 4.34%
GraphQL#getStatement 4.25%
GraphQL#statement 3.96%
GraphQL#connected_channels 3.45%


Top 5 web transactions
by percent of wall clock time
700 %
600 %
500 %
400 %
300 %
200 %
100 %
:35 PM
4:40 PM
4:45 PM
4:50 PM
GraphQL#channels
GraphQL#recipe_recommendations
GraphQL#statements
GraphQL#action_field



Transaction name and error class	Error message
GraphQL#trigger_field Ifttt::Protocol::Executor::InlineRefreshError	bad status code 400 received
GraphQL#action_field NoMethodError	undefined method `find' for nil:NilClass
GraphQL#trigger_field Ifttt::Protocol::Executor::BadStatusError	bad status code 401 received
GraphQL#recipe NoMethodError	undefined method `empty?' for nil:NilClass
GraphQL#trigger_field IftttLib::Martini::NoLiveChannelError	No live channel present
GraphQL#action_field Ifttt::Protocol::Executor::BadStatusError	bad status code 404 received
GraphQL#action_field IftttLib::Martini::NoLiveChannelError	No live channel present

http://bit.ly/gql-rb-nr

#3 Proper **monitoring** is as **important** as good performance

#4 GraphQL is **awesome**

@nettofarah

nettofarah@gmail.com