Introduction to GraphQL

Or How I Learned to Stop Worrying about REST APIs

Who Am I

Hafiz Ismail

@sogko on all social media
https://wehavefaces.net

What is this about?

What is this about?

I'll try to convince you that **GraphQL** helps to address some of the more *common headaches* developers faced when building a **REST API** -backed application.

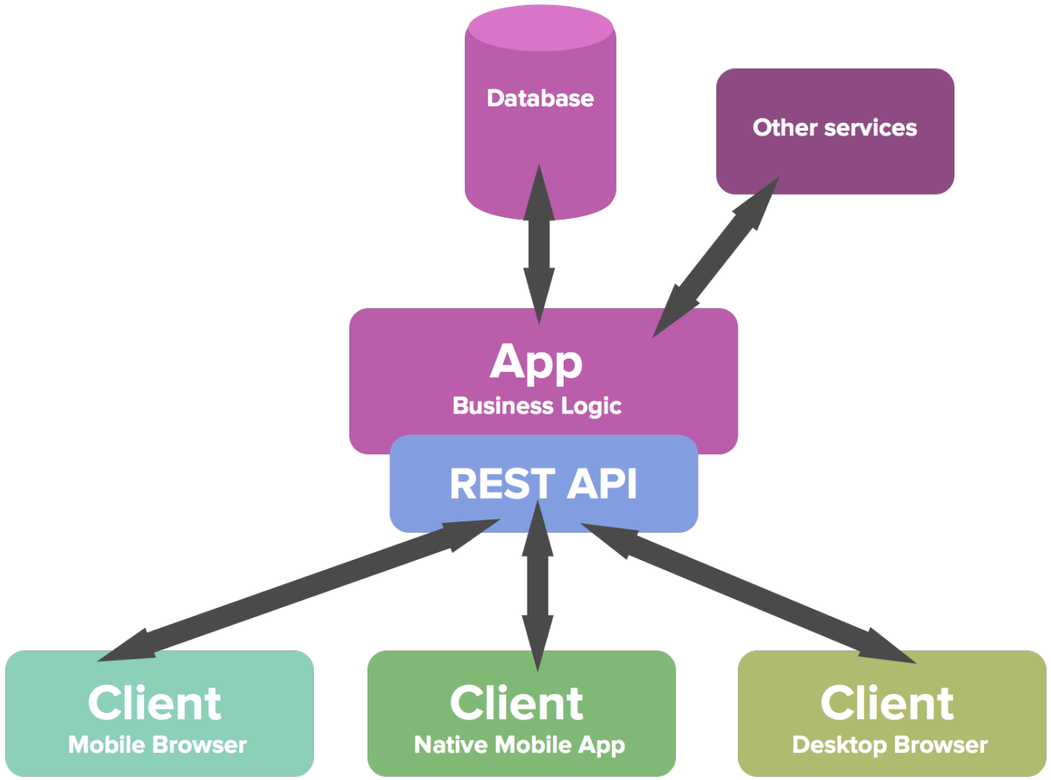
Who is this for?

Who is this for?

Developers, developers, developers

Let's start!

Typical architecture of a web application using REST API

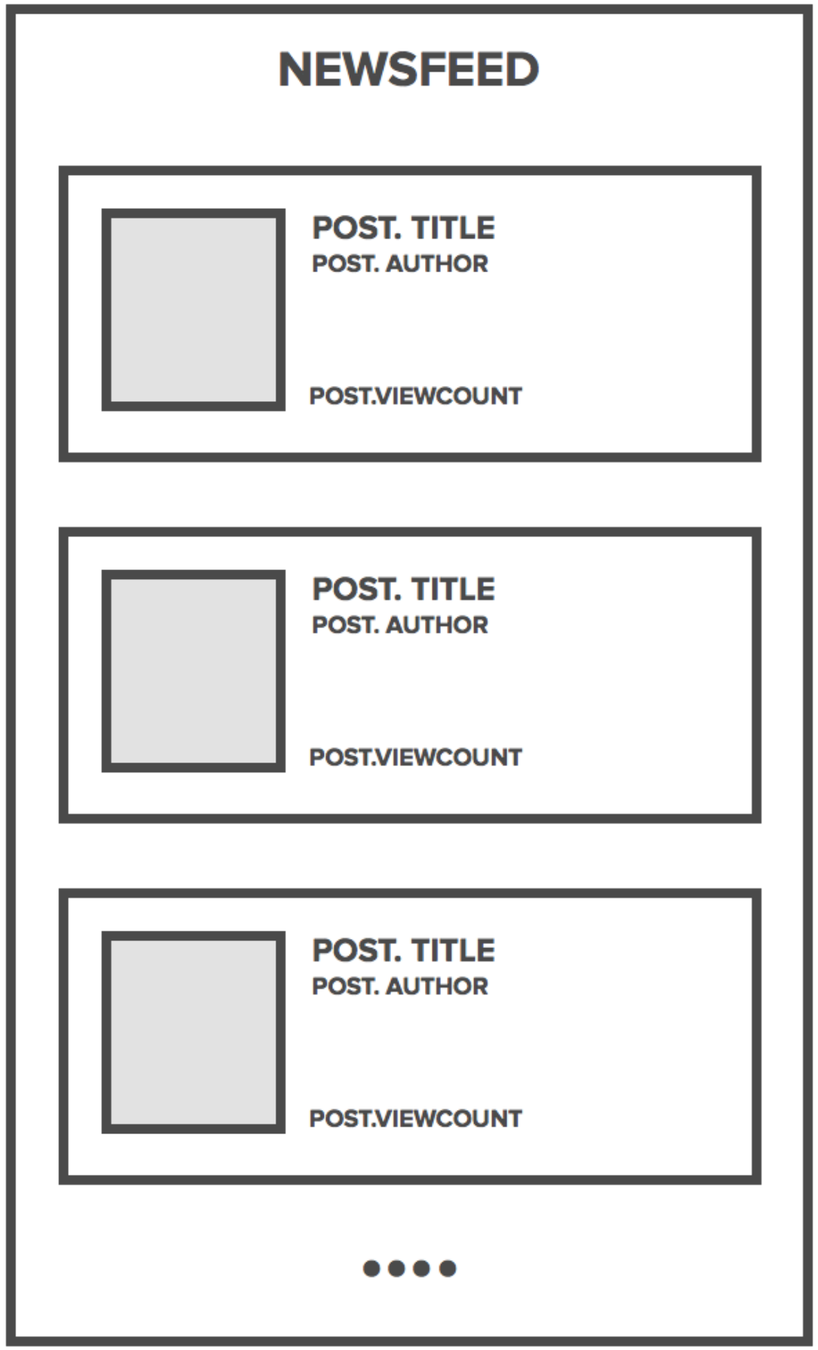


Understanding common issues that developers face

How?

Let's try to design and build a REST application together!

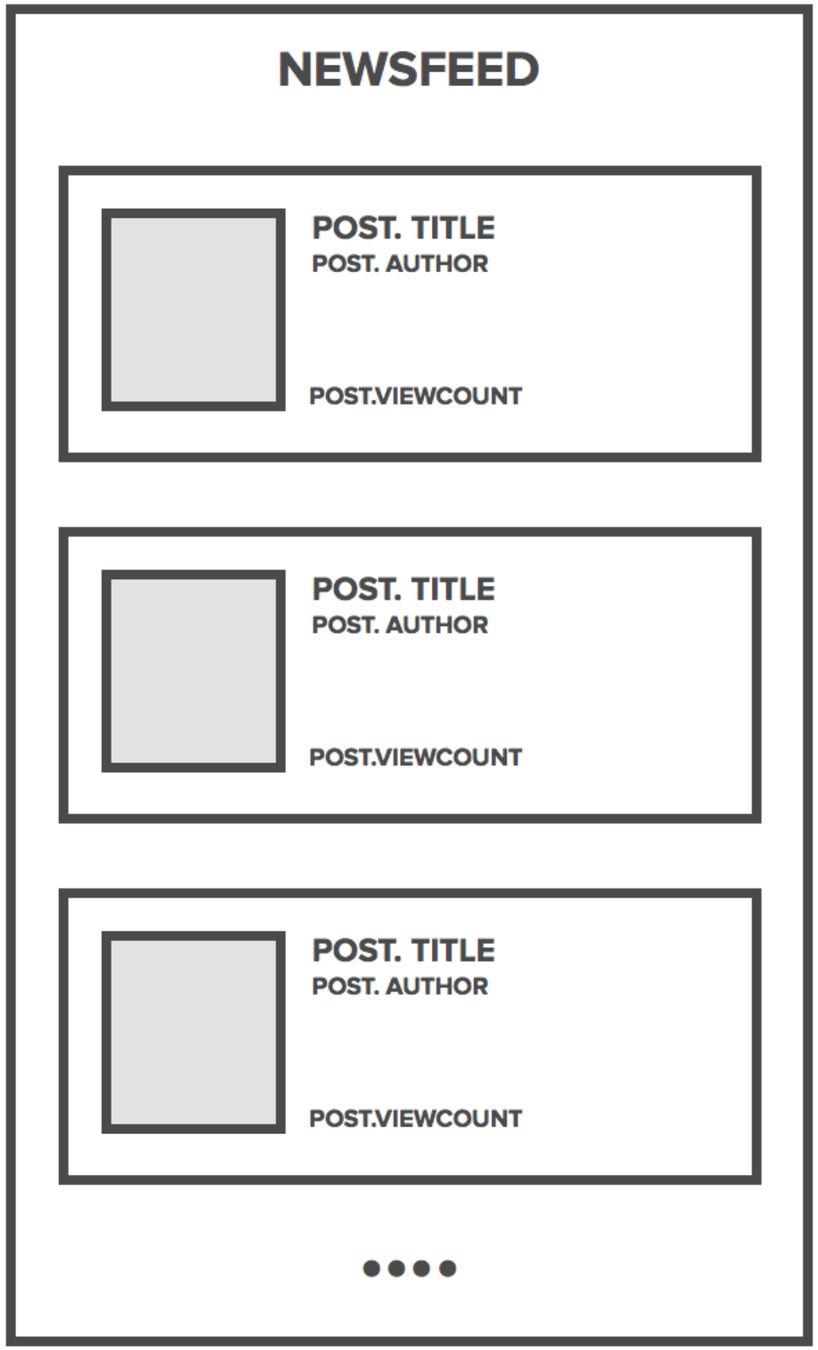
Yay!



Goal: A newsfeed SPA

Mobile-first

REST API for data fetching



Designing the REST API

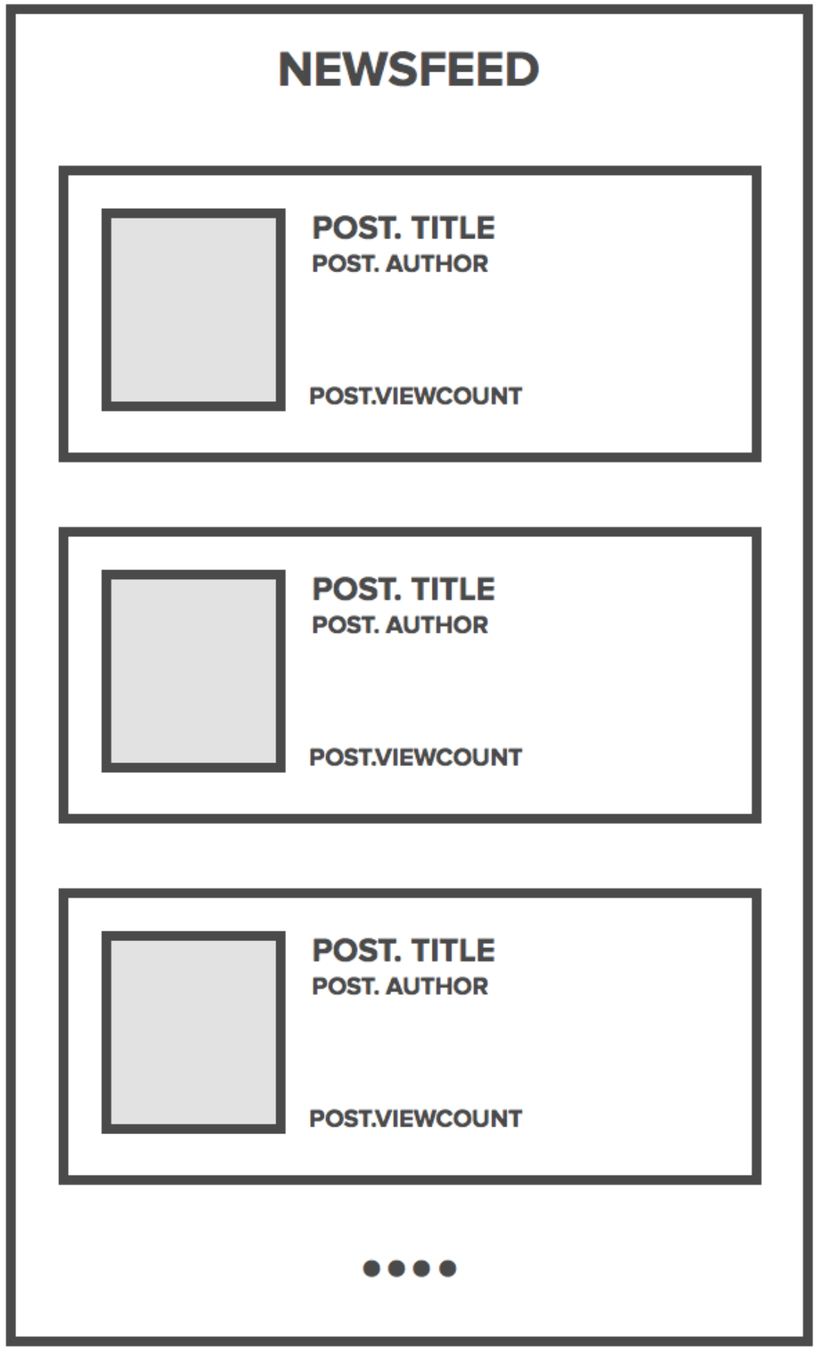
Two **resources**

- Users
- Posts

```
POST /posts
GET /posts/1
PUT /posts/1
DELETE /posts/1
...
POST /users
GET /users/1
PUT /users/1
DELETE /users/1
...
```

Someone said: Let's achieve strict REST! We can do this!

Let's see what happen



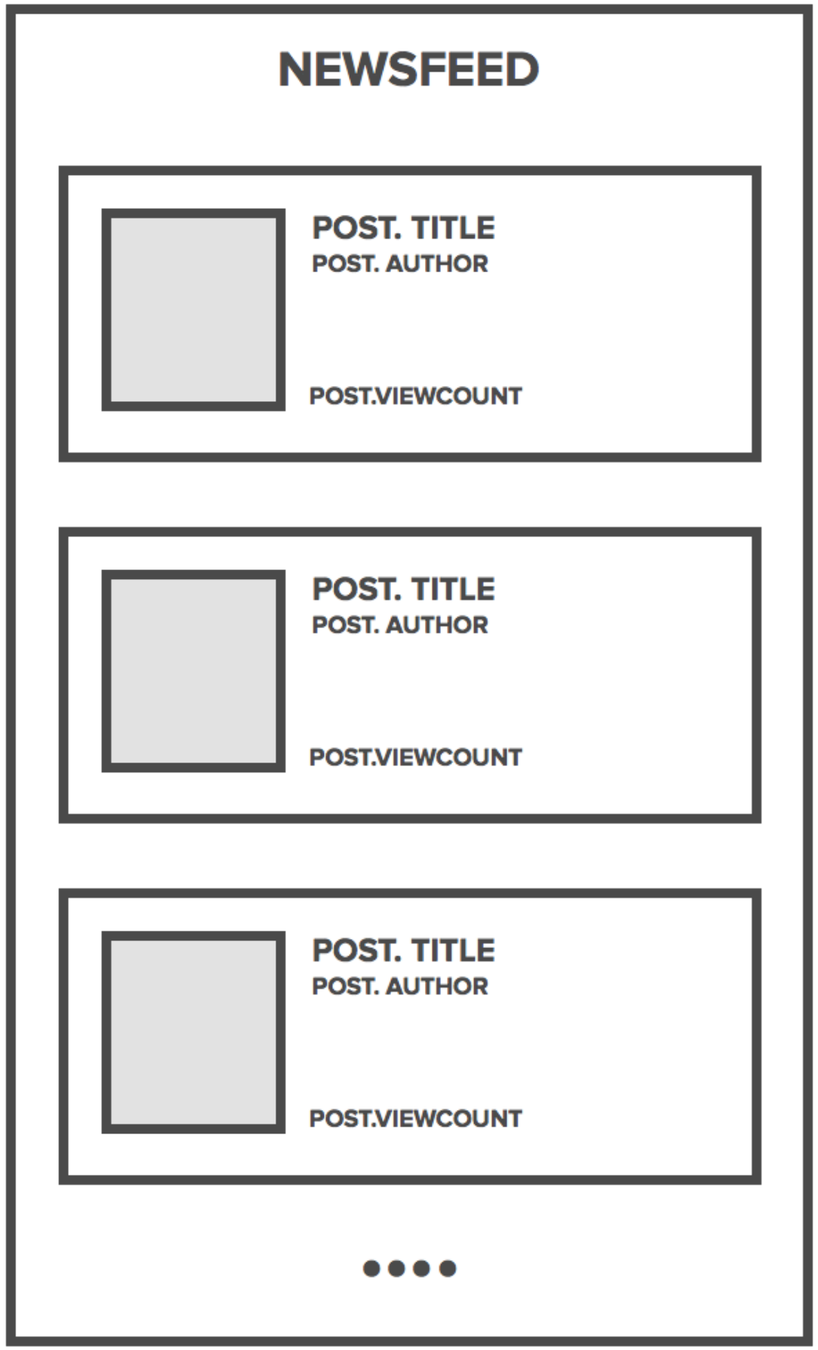
Render newsfeed

```
GET /posts?limit=10
{
  "posts": [
    {
        "id": 1,
        "title": "Hello world!",
        "author": 10,
        "viewCount": 23,
        "likedCount": 3,
        "likedBy": [1, 3],
    },
    ...
  ]
}

_
```

Great!

Oh wait, we need to get **`author's name`** and **`avatar URL`**

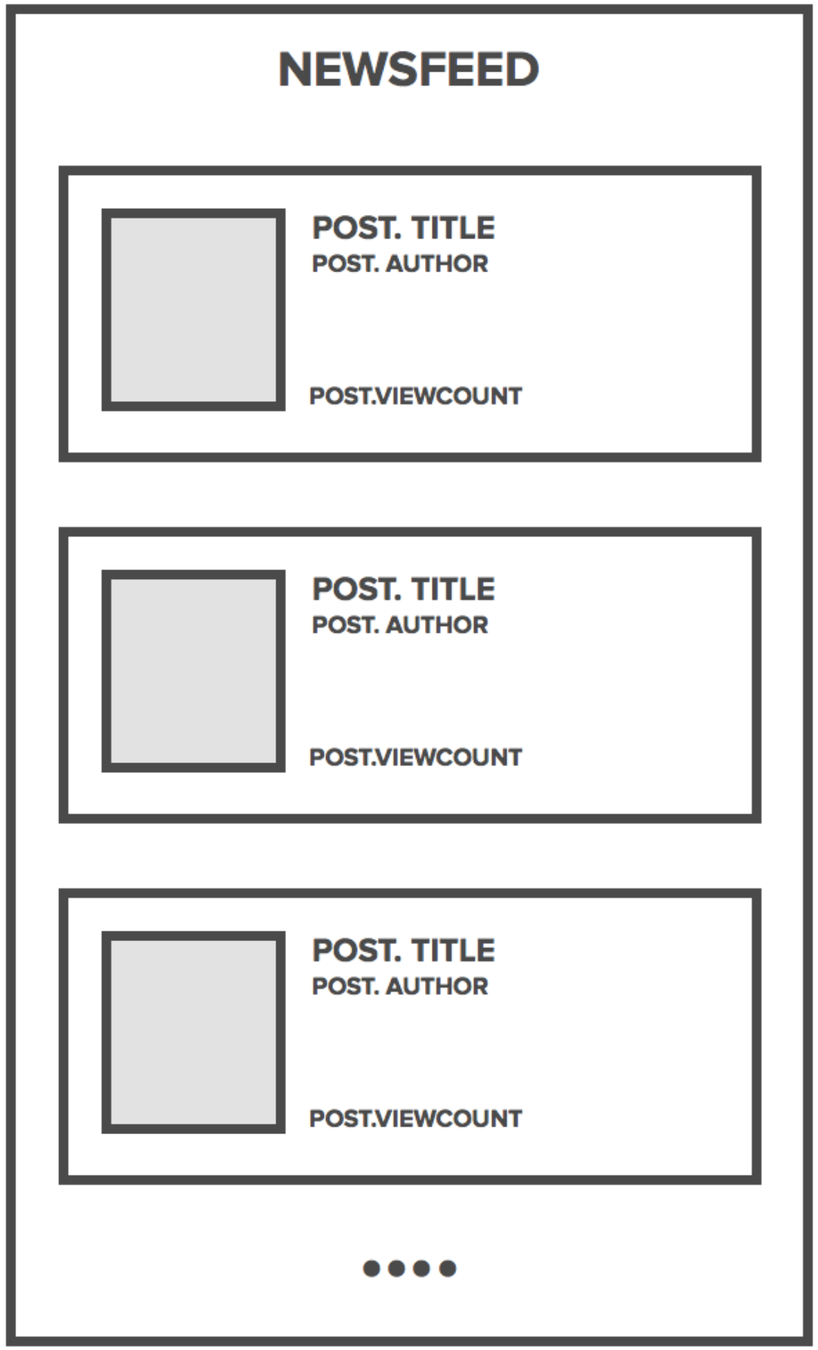


Render newsfeed

```
GET /posts?limit=10
GET /users/10
{
  "user": {
        "id": 10,
        "name": "John Doe",
        "nickname": "Johnny",
        "age": 23,
        "avatar_url": "/avatar/10.jpg"
    }
}

_
```

So we make another request to get the author for the first post...



Render newsfeed

```
GET /posts?limit=10
GET /users/10
GET /users/20
{
  "user": {
        "id": 20,
        "name": "Emily Sue",
        "nickname": "M",
        "age": 25,
        "avatar_url": "/avatar/20.jpg"
    }
}
```

Wait, so we have to do a separate request for each post to get information for its author?

Hhnnggghhh 😖

Issue #1: Multiple round trips

This is no bueno

Issue #1: Multiple round trips

One possible solution:
- A new endpoint **`/newsfeed`**

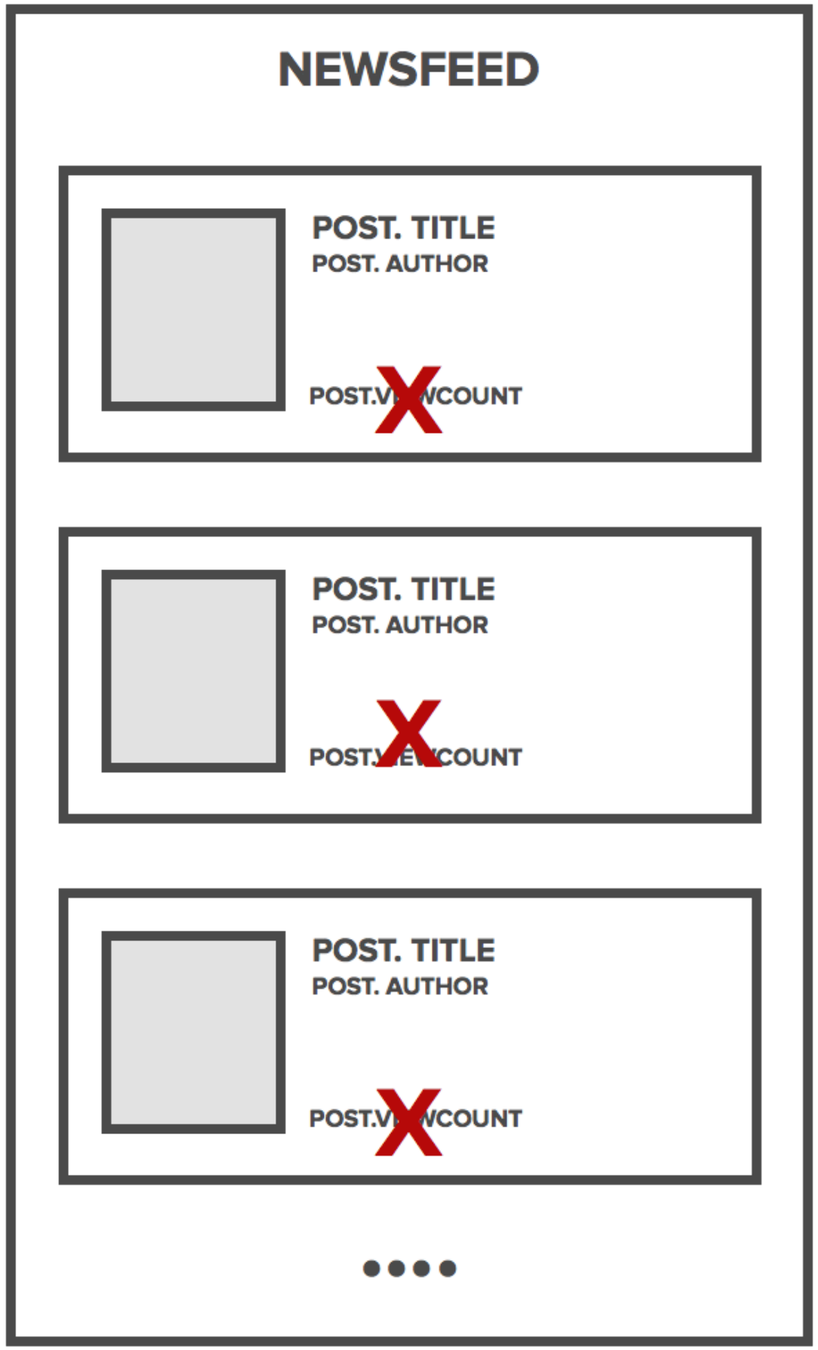
But now you have a singleton REST resource that is too tightly-coupled to your client UI.

You then tell yourself "it's not that bad. We're still REST-ish."

Eventually, you launch your app with its mobile client and it went viral! Yay!

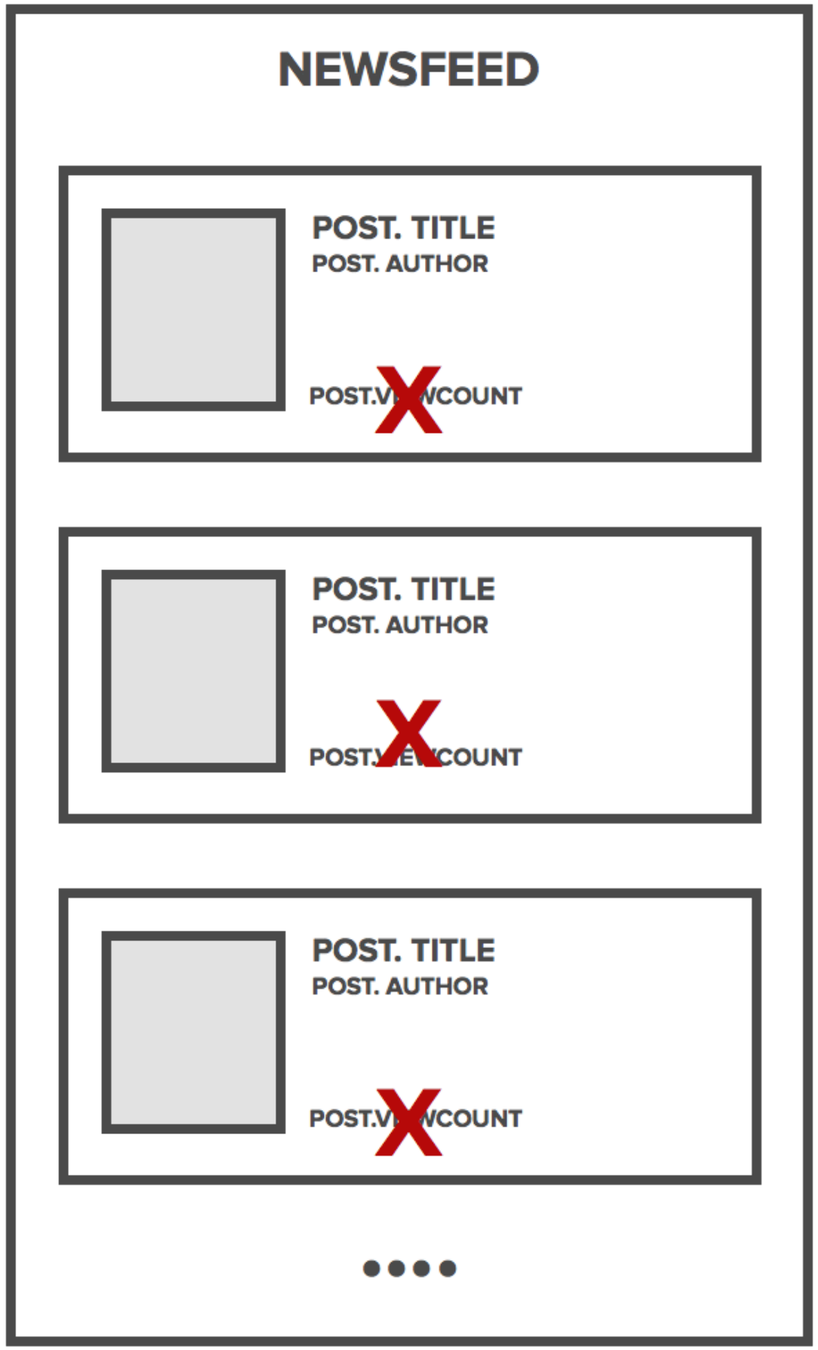
New requirement!

Here comes your product designer with changes

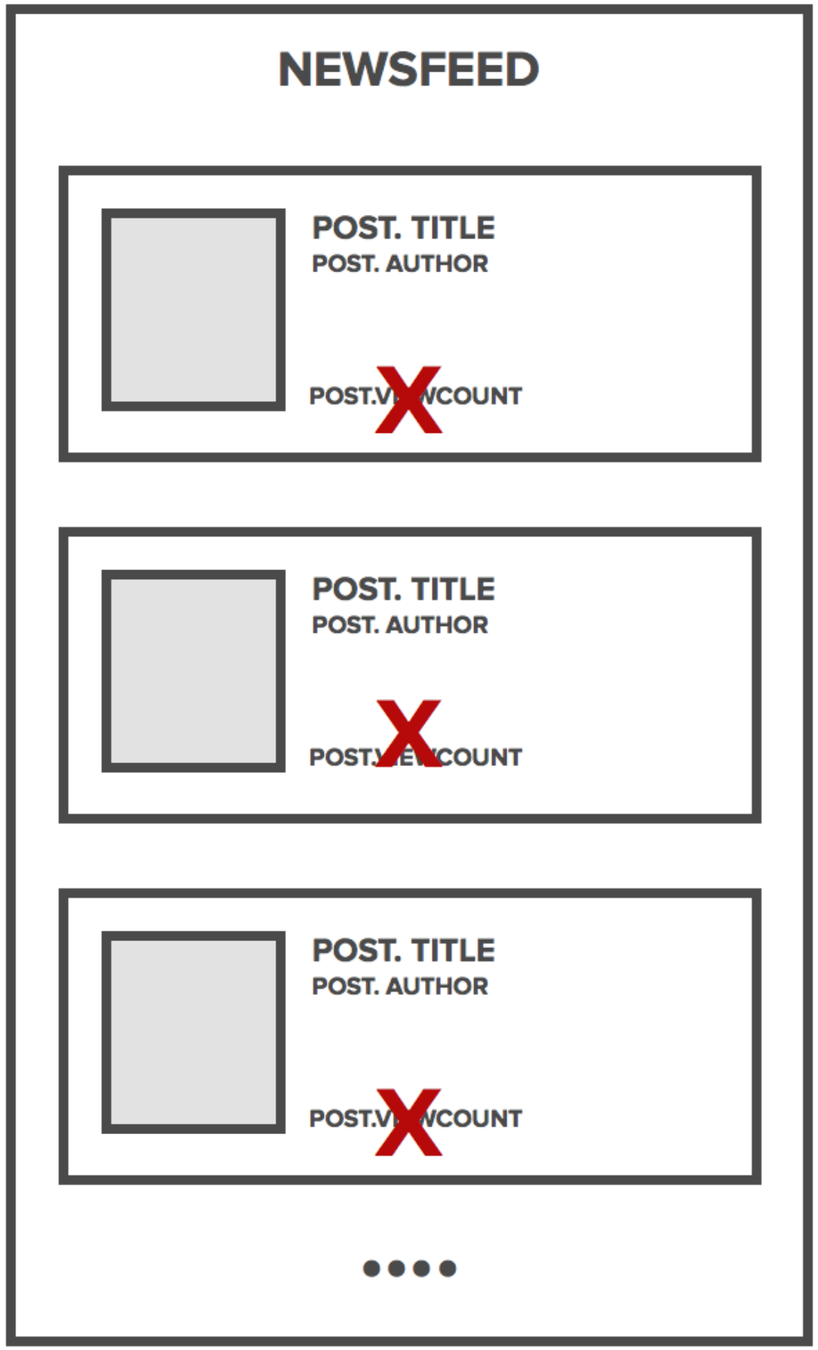


New requirement!

- It has been a year since you first launch your mobile client, and you have several versions of the client out in the wild.
- Your product designer said: "We need to stop showing the **`view_count`** because of reasons"
- What do you do now?



- Removing the **`view_count`** field from **`/newsfeed`** is not an option.
- Older version of your mobile client depends on it. (What if they crash? #nilpointerreference)
- So newer version of the mobile client **does not need** the **`view_count`** field, but to cater to the older versions, the **`/newsfeed`** still need to return it.



- **What if this keeps happening?**
  Newer clients would be requesting data that they essentially don't need anymore.
- Not that bad when you just start out, but in the long run, it'll be something nagging at you
- Sleepless nights are ahead for you.

Issue #2:
Overfetching of data

Issue #2: Overfetching of data

Wouldn't it be nice if there client receives only the data that it requires and had requested?

One possible way to go about this:

- Endpoint accepts parameter to specify the fields that you are interested in

Not a bad solution, but *yet* another thing to worry about.

From a humble SPA to a full-fledge product

You and your core team have build this complex API that serves a great purpose.

From a humble SPA to a full-fledge product

- Your CEO recently announced that he envisions that your product should have a **client for every device / platform imaginable**.
- iOS, Android, OSX, Windows, Linux
- Raspberry Pis, BBC micro:bits
- Cars
- Your mom's toaster

From a humble SPA to a full-fledge product

- New hires/developers join in.
- How do you quickly allow new developers to study your API
  - What resources are available,
  - What parameters are accepted
  - Which ones are required, which ones are not?

From a humble SPA to a full-fledge product

- If only you had used Swagger / RAML / API Blueprint when you started.
- Now you have to invest time/effort into it.
- Or probably you already did, bonus points for you 👍🏻

Issue #3: Documenting your API now becomes a thing.

Issue #3: Documenting your API now becomes a thing.

- More than just writing down the specs in a formal form so that it can be referenced
- How you allow one to **discover** and **explore** your API?
- Big enough concern that some parts of the community has banded together to create tools for this. *(Which is a great thing, yay open-source)*
- But *yet* another thing for you to worry about.

Had enough?

So how do we proceed from here?

Here's where GraphQL can help

What is GraphQL?

What is GraphQL?

GraphQL is a **data query language and runtime** designed and used at Facebook to request and deliver data to mobile and web apps since 2012

Source: http://graphql.org

What is GraphQL?

What you need to know:

- A GraphQL **query is a string** interpreted by a server that **returns data in a specified format.**

Which format? Anpther propriety format from FB aye?

What is GraphQL?

Here is an example query and its response:

```
{
  user(id: 3500401) {
    id,
    name,
    isViewerFriend,
    profilePicture(size: 50)  {
      uri,
      width,
      height
    }
  }
}
```

What is GraphQL?

Here is an example query and its response:

```
{
  user(id: 3500401) {
    id,
    name,
    isViewerFriend,
    profilePicture(size: 50)  {
      uri,
      width,
      height
    }
  }
}
```

```
{
  "user" : {
    "id": 3500401,
    "name": "Jing Chen",
    "isViewerFriend": true,
    "profilePicture": {
      "uri": "http://someurl.cdn/pic.jpg",
      "width": 50,
      "height": 50
    }
  }
}
```

Wait, so how does GraphQL address the issues previously raised?

Does it even lift, bruh?

Wait, so how does GraphQL address the issues previously raised?

Let me show you

Recap of the issues

Issue #1: Multiple round trips.
Issue #2: Overfetching of data.
Issue #3: Documenting your API now becomes a thing.

It's demo time!

😄

Review of demo

Hope nothing crashes

Review of demo

- How to do a query
  - **`curl`**
  - GraphiQL (https://github.com/graphql/graphiql)
    - Uses **`introspection`** queries to allow one to explore API

Review of demo

- Addressed the issues that we had previously raisedough time
    1. One query is all your probably need to render your UI fully
    2. Your clients would only get data that it is interested in.
    3. Built-in documentation and introspection as part as Schema definition.

What's next?

What's next?

If you're interested to learn more about **GraphQL**

- GraphQL : https://graphql.org
- #graphql
  - Twitter : https://twitter.com/search?q=graphql
  - Medium : https://medium.com/search?q=graphql
- https://wehavefaces.net #shamelessplug
  - A couple of introductory articles (Go + GraphQL + Relay)
  - More articles coming soon

What's next?

GraphQL libraries for many platforms available

- **`graphql-js`** (NodeJS)
- **`graphql-ruby`** (Ruby)
- **`graphene`** (Python)
- **`sangria`** (Scala)
- **`graphql-go` (Go/Golang)**

(*btdubs*, GraphQL is platform agnostic, yay)

The real reason why I'm here

Not for the money nor fame, but...

Looking for more contributors!

`graphql-go`

https://github.com/graphql-go/graphql

- 8 months year old baby
- Still at its infancy, but growing fast
- Very pleasant and chill community; constructive discussion always encouraged.
- Actively looking for more contributors (Currently at 15)

Looking for more contributors!

`graphql-go`

https://github.com/graphql-go/graphql

Ping me **@sogko** or **@chris-ramon**

Or better yet, dive right in and just **submit a PR!**

Very much encouraged

Thanks for listening

Feel free to come up and say hi

Slides will be up @ https://github.com/sogko/fossasia-2016-graphql-demo