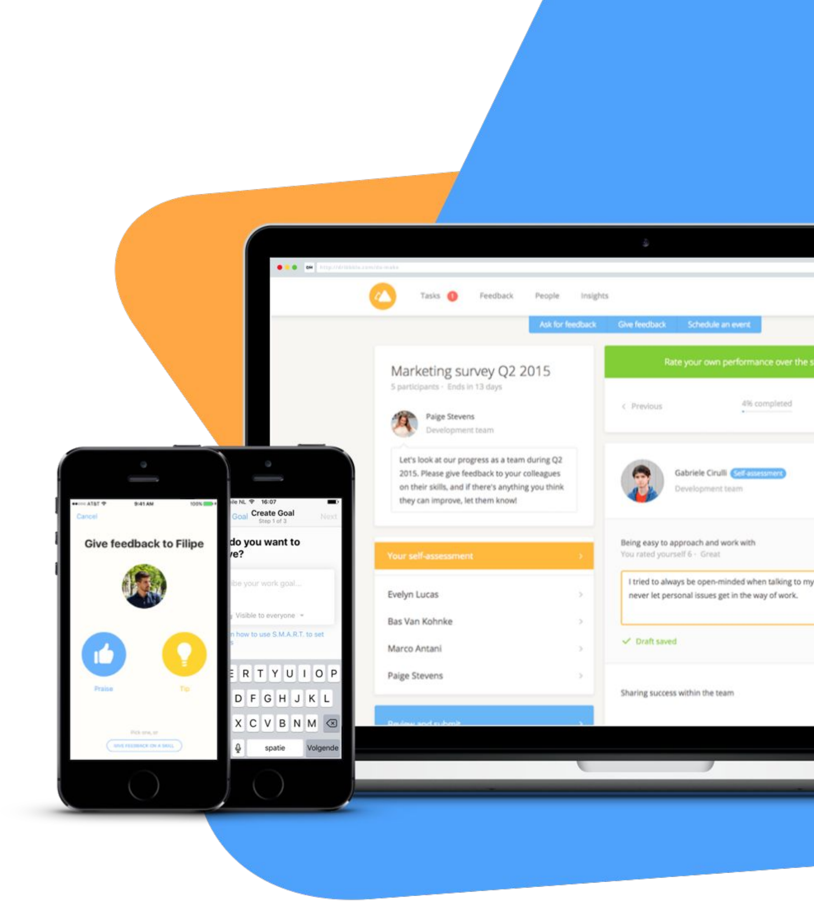
Making mistakes with **GraphQL**

Bertrand Dubaut - Thiago von Sydow
Impraise | Lisbon GraphQL
02/02/2018

Impraise

Who are we?



Let's look at our progress as a team during Q2 2015. Please give feedback to your colleagues on their skills, and if there's anything you think they can improve, let them know!

Bertrand Dubaut

Lead backend developer
Github, Slack & Twitter: ***@bdubaut***

Thiago von Sydow

Senior backend developer
Github: ***@thiago-sydow***

GraphQL @ *Impraise*

Cool. but why?

Why GraphQL?

Why take the gamble?

- Multiple clients to support with different requirements
- Endpoints became overloaded with options
- Some endpoints were duplicated
- Specific client/customer requests that led to “single use case” endpoints -> we needed flexibility

I HAVE NO IDEA WHAT I'M DOING
DANGER!
PICK ONE

Returning HTTP error codes with empty responses

Just read the spec and apply it.

- Both can cohabit for a little bit while the clients adapt:
  - Keep returning the HTTP error code
  - Return the normal GraphQL response with the “errors” entry
- When a significant % of users have updated their clients, remove the HTTP error codes from the response, and just return a “200 OK”.

Choosing a collection pattern early

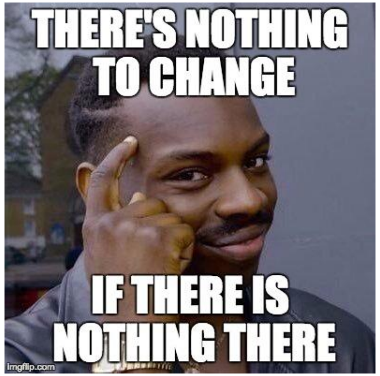
Decide early if you want to be relay-compatible

- Clients already using the id (database key)
  - Can’t *just* change to Global IDs
  - Clients have to update to use a new id field. Only after we can implement Global IDs
- Multiple fields returning collections but with different pagination patterns
  - Use savy timing for deprecation, and *communicate* about it to all concerned teams

Poor early schema design

...but if you really want to change it

- Remove all unused nodes from the graph



- Add deprecation warnings to all nodes that do not make sense, and build up the schema from there

Poor early schema design

Spend time as a team building your initial GraphQL schema

- Stop thinking in endpoints ASAP
- Check out these **Github** threads:
  - *facebook/graphql#175*
  - *facebook/graphql#134*

The Graph is your friend!

(but it will get messy)

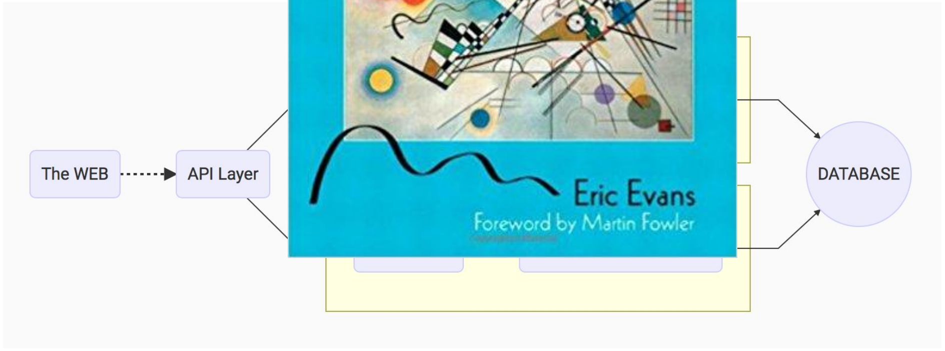
A few good things we’ve done

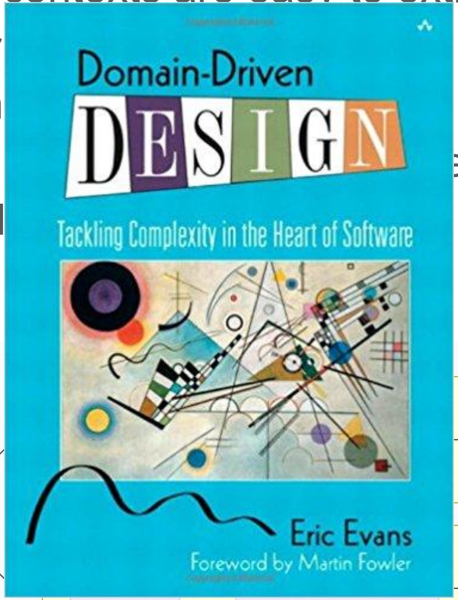
We can’t make only mistakes (can we?)

Domain Driven Design

Allowing the monolith to scale

- Bounded contexts are easy to extract to
  - Inter
  - Stan
  - And ernal API I




Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Domain Driven Design

Allowing the monolith to scale

- A place for everything, and everything in its place
  - Easier debugging
  - Easier testing (isolation)
  - Less moving pieces
  - Easier team scaling

How we approach things now

Trying to be neat AF

- Have a **clear separation** between the API layer and the domain layer
- As few Rails as needed. Be a Ruby developer first (language > framework)
- The graph (API layer) is **based** on the **designs**

Field authorization

Leveraging instrumentation

```ruby
module Graph
  class FieldAuthorization
    def instrument(_type, field)
      [...]
      resolve_proc = authorization_proc(field)

      field.redefine do
        resolve(resolve_proc)
      end
    end

  private

    def authorization_proc(field)
      [...]
      ->(obj, args, ctx) {
          [...]
          policy = permission[:policy_class].new(ctx[:current_user], promise_result)
          raise GraphQL::ExecutionError, :forbidden unless policy.send(permission[:action])
          [...]
        end

        resolved
      }
    end
  end
end
```

Field authorization

Leveraging instrumentation

As simple as writing:

```ruby
field :organization do
  type OrganizationType
  description "Lookup an `Organization` by id"
  access_permission(policy_class: OrganizationsPolicy, action: :load?)
```

Questions?

Any feedback is also welcome

Oh, and if you want to make GraphQL mistakes with us:

Impraise (*Amsterdam* or *Lisboa*) is looking for :

- **Ruby** developers
- **Front-end** developers (React, Apollo, Redux)
- **DevOps** animals
- **Mobile** developers (iOS and Android)

Send us your CVs

at bertrand@impraise.com

or thiago@impraise.com

Obrigado!

Check us out at impraise.com