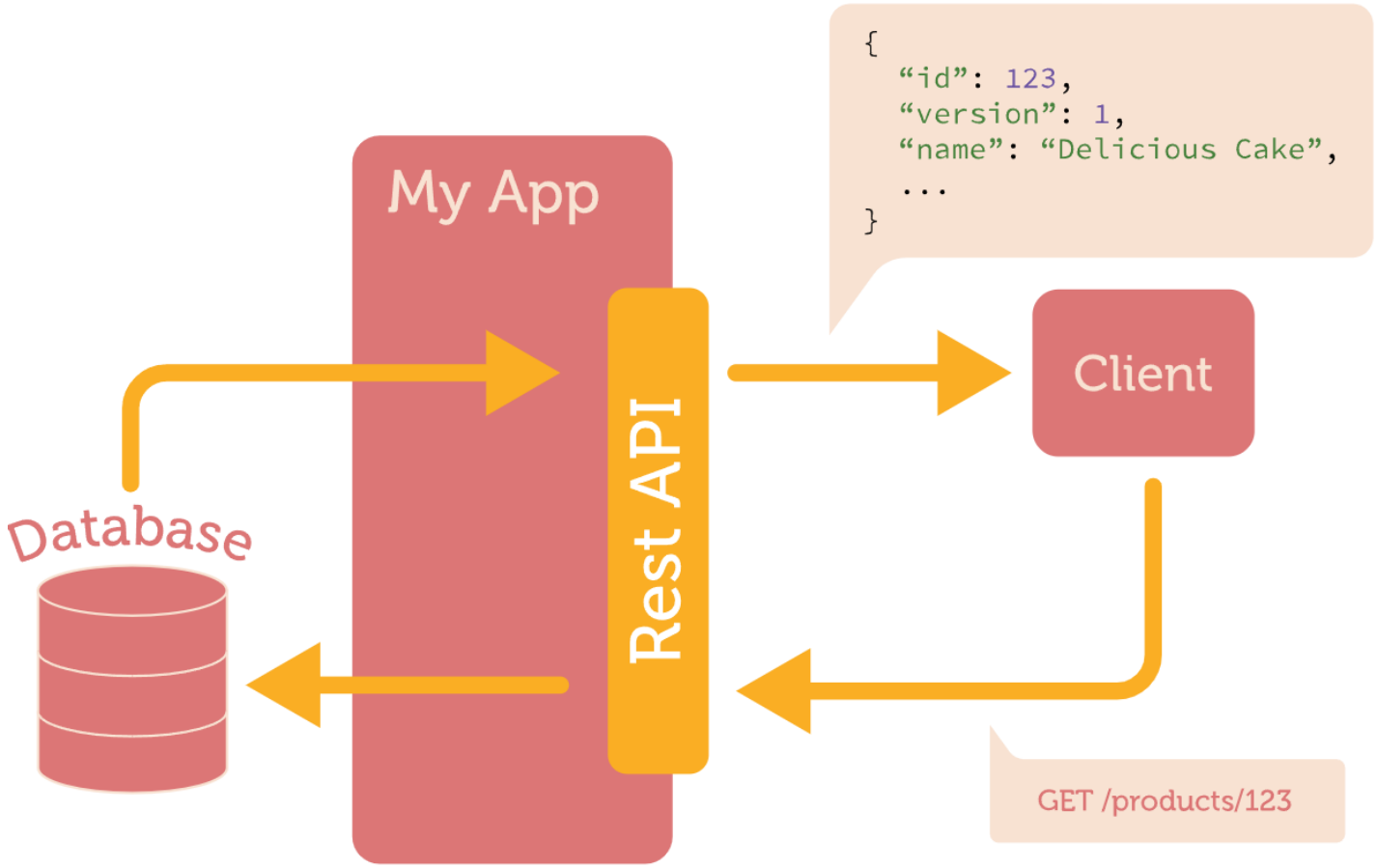
GraphQL

Jaime Riascos

REST API

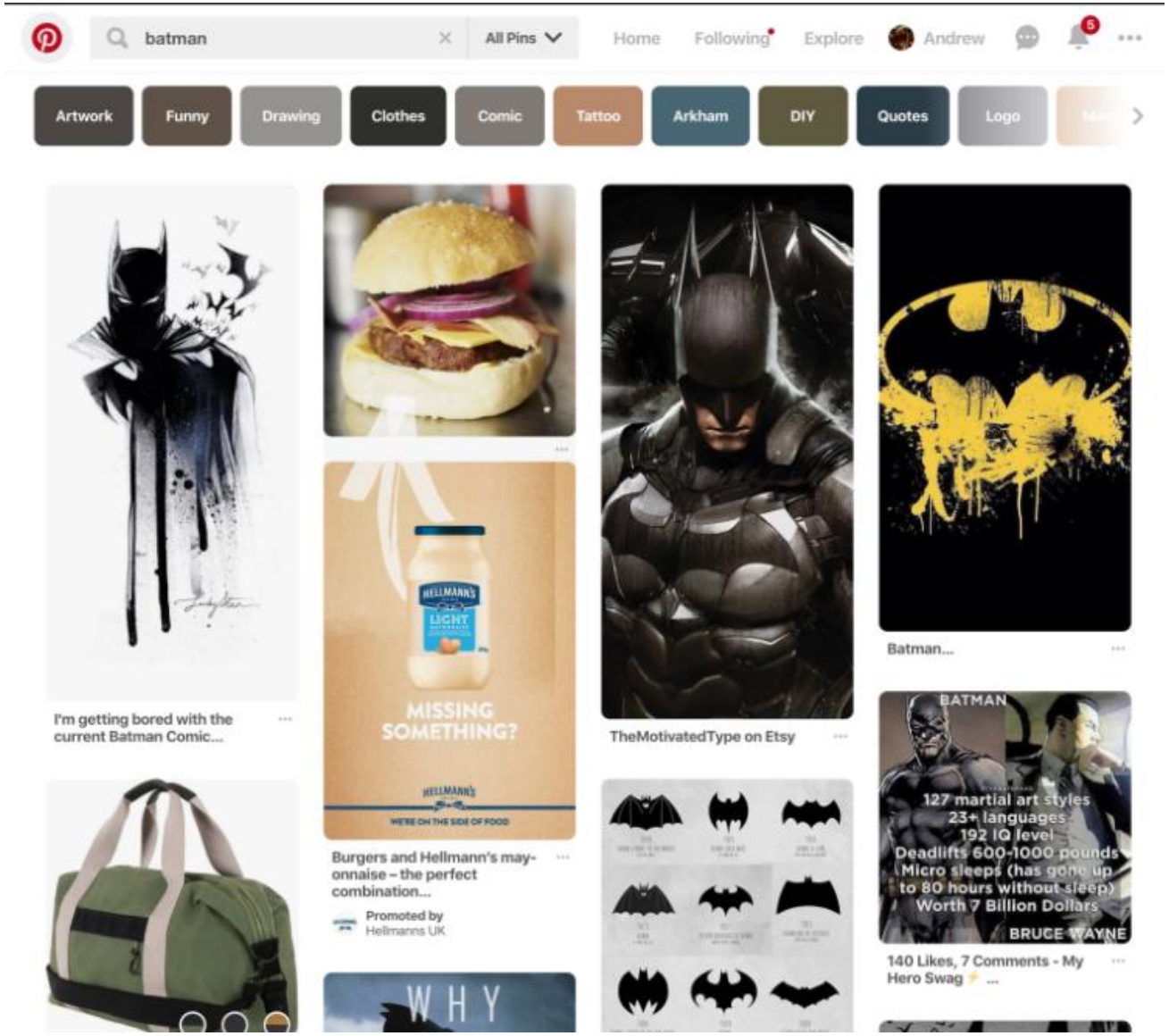

{
"id": 123,
"version": 1,
"name": "Delicious Cake",
...
}
My App
Rest API
Client
Database
GET /products/123

Problemas?

- Búsqueda Excesiva(Over-fetching)
  - /productos?campo=nombre&campo=descripción&campo=variantes[*].precio
- Búsqueda Insuficiente (Under-fetching)
  - /products?expand=productType&expand=variants[*].price.taxRate
- Cambios y evolución de la API
  - Control de versiones
  - Desaprobación
  - Mantenimiento

Problemas?

- Quiero obtener el nombre y descripción de cada imagen de una REST API de un website.

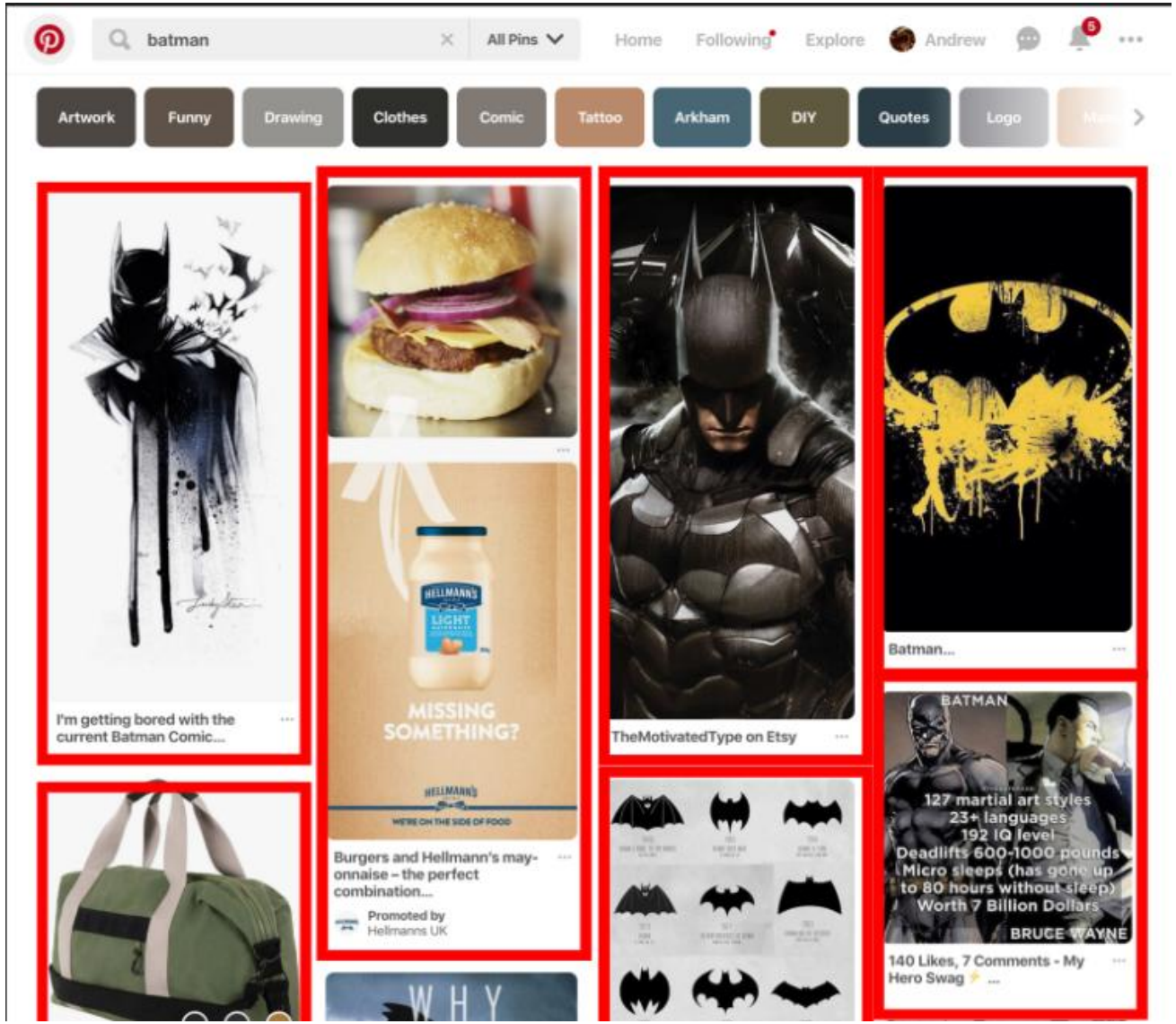


Problemas?

- Una llamada a la API para la recopilación
- Luego, una llamada por artículo para obtener más detalles

```
GET /results
GET /results/1234
GET /results/5678
GET /results/9012
```

O = n + 1

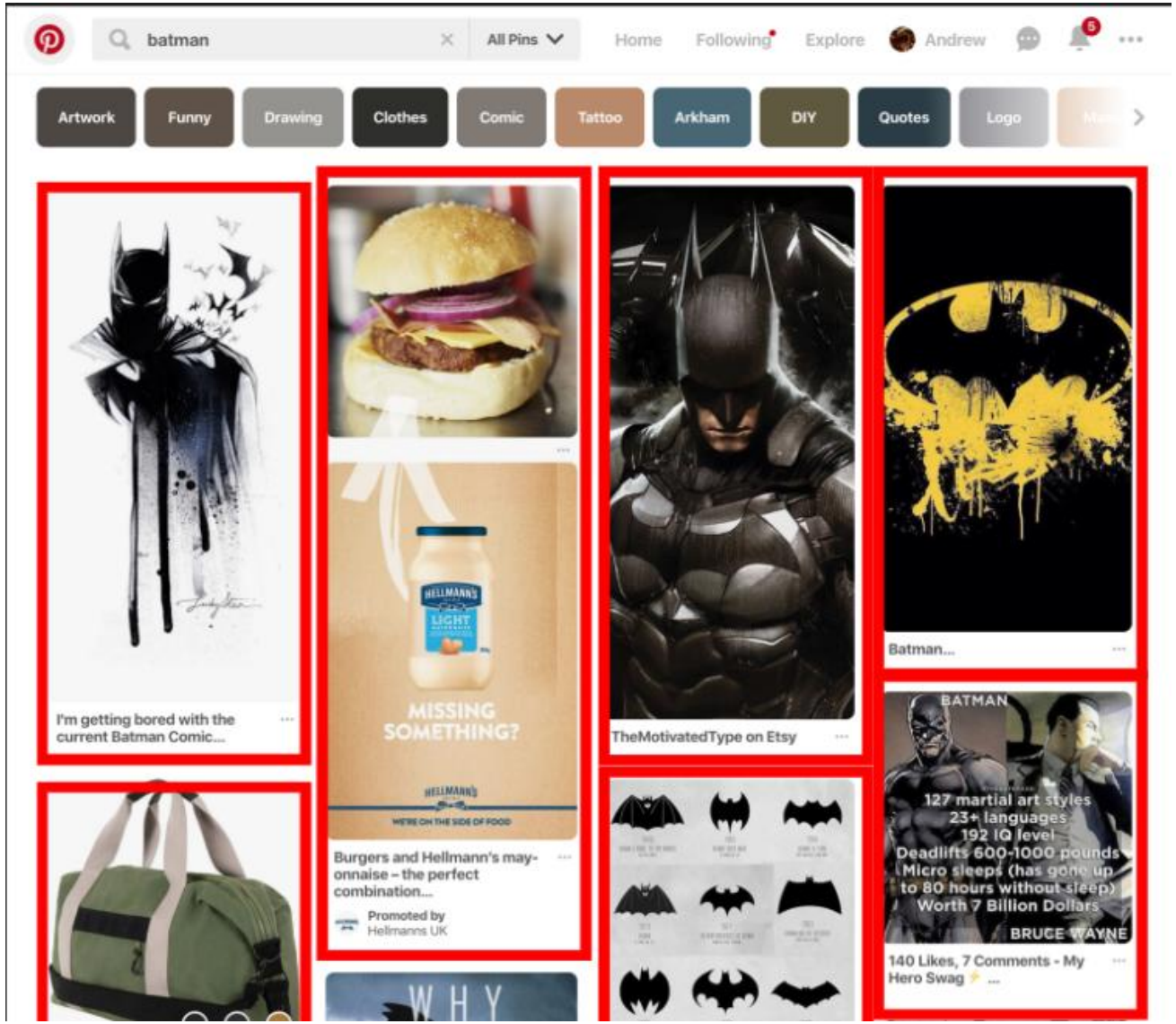


Problemas?

- GraphQL can bundle lots of data into one query instead of many

```
POST /graphQl

{
    pins(last: 10) {
        imgUrl
        name
        likes
    }
}
```

O = 1



GraphQL

GraphQL aborda estos problemas de raíz:

- Un solo endpoint: Toda la comunicación ocurre a través de un único punto de acceso (ej. /graphql). Se acabaron los cientos de endpoints.
- El cliente tiene el control: El cliente envía una consulta con la forma exacta de los datos que necesita. El servidor responde con un JSON que coincide con esa forma.
- Evolución sin versiones: Añadir nuevos campos o tipos al esquema no rompe las aplicaciones existentes. Los clientes antiguos siguen pidiendo los mismos datos y los nuevos pueden empezar a pedir los nuevos campos

Otras ventajas

- Esquema fuertemente tipado
  - Se puede compartir con el cliente y el servidor
  - Puede validar solicitudes en tiempo de compilación
  - Completar automáticamente consultas y mutaciones
  - Se puede burlar fácilmente
- No under-fetching u over-fetching
  - Reduce tiempos de la llamada
- Comunidad comprometida
  - Este proyecto ha existido por un tiempo
  - Y durará un tiempo también

Arquitectura y Flujo de Datos

- Un servidor GraphQL se sitúa entre tus clientes y tus fuentes de datos (backend).
- **Cliente:** Envía una consulta GraphQL al servidor (HTTP POST).
- **Servidor GraphQL (Engine):** Parsea, valida y ejecuta la consulta.
- **Resolvers:** Son funciones que obtienen los datos de las fuentes subyacentes.
- **Fuentes de Datos:** Base de datos SQL/NoSQL, API REST, microservicio, etc.
- **Respuesta:** El servidor ensambla los resultados en un JSON y lo devuelve.

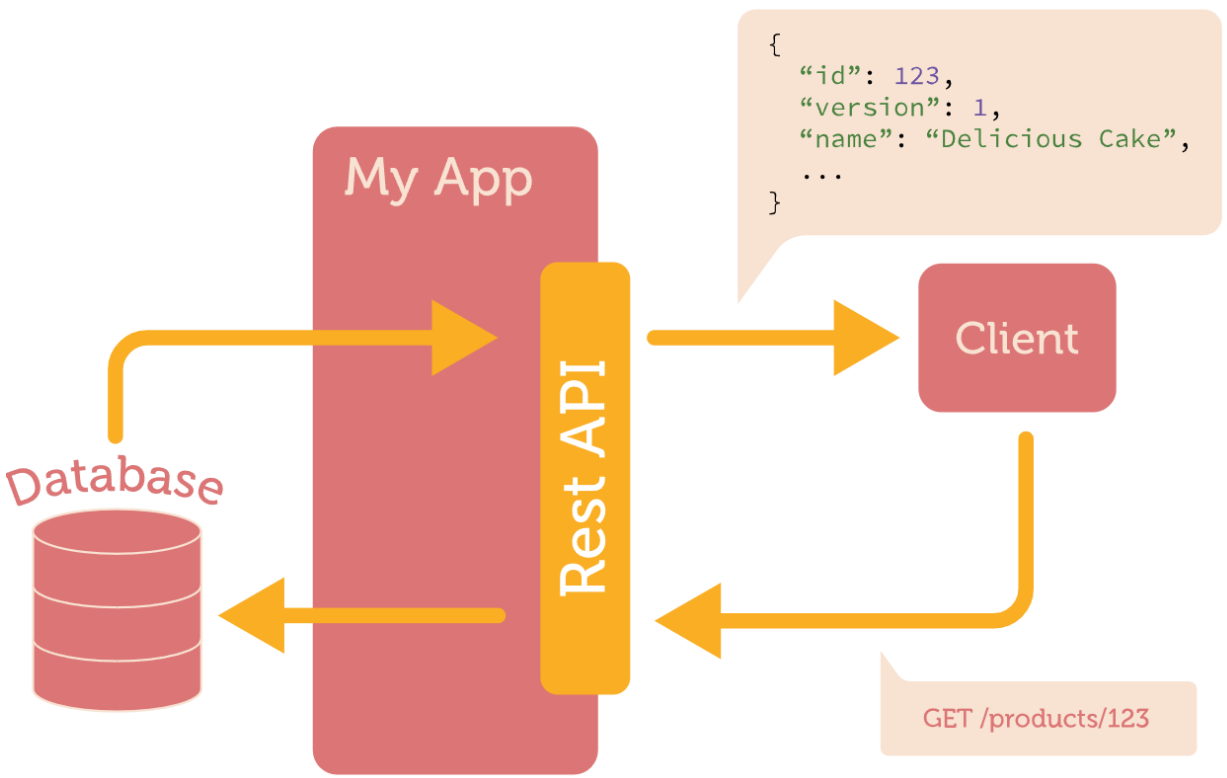
El esquema

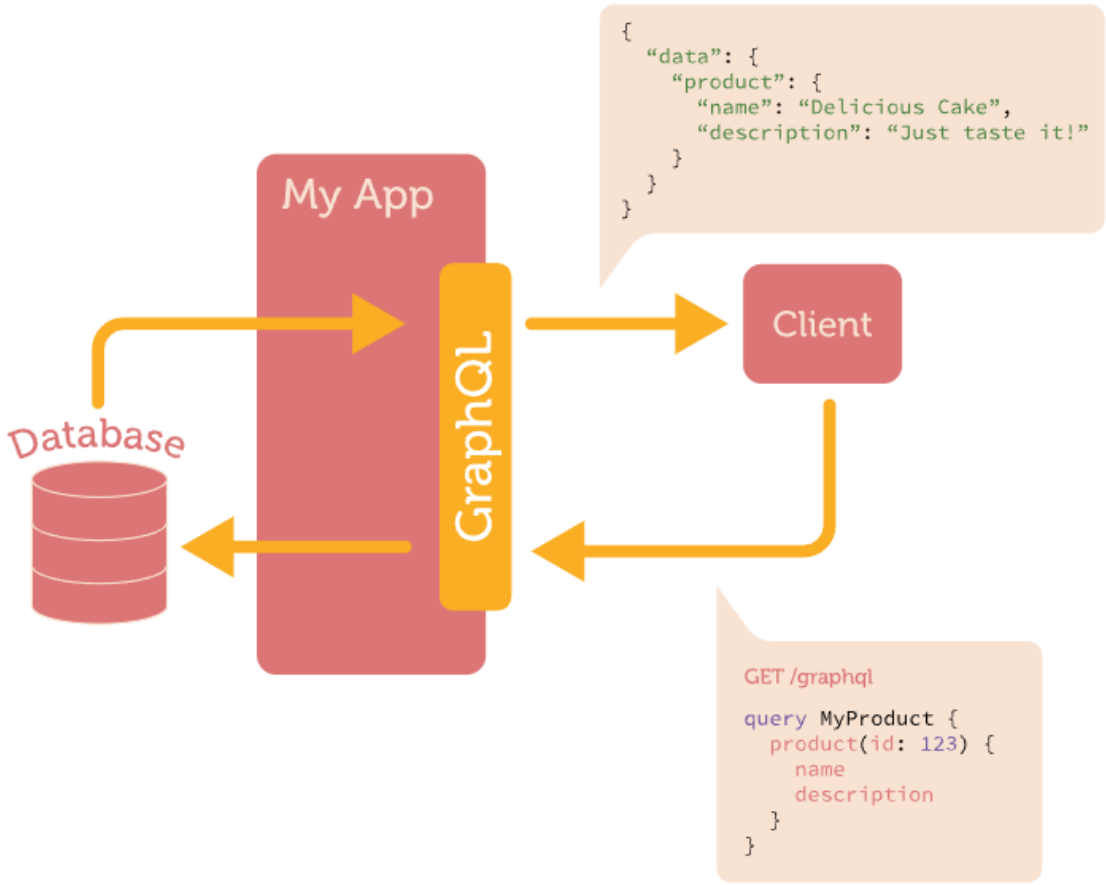
El Esquema es el contrato formal y fuertemente tipado entre el cliente y el servidor, escrito en SDL (Schema Definition Language).

```
type Query {
  articulo(id: ID!): Articulo
  articulos(limit: Int = 10): [Articulo!]!
}

type Articulo {
  id: ID!
  titulo: String!
  contenido: String
  autor: Usuario!
  comentarios: [Comentario!]
}
```

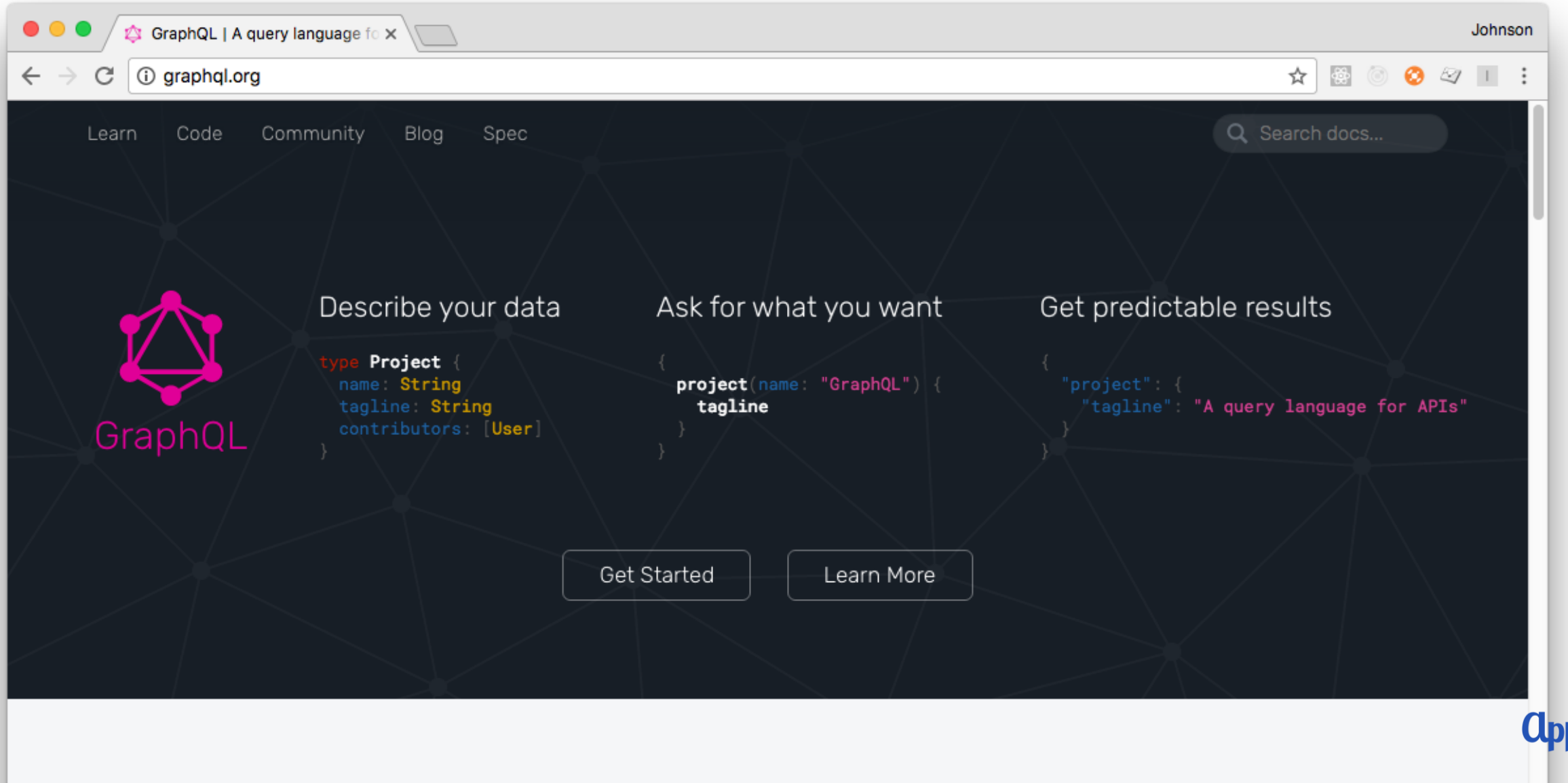
```
type Usuario {
  id: ID!
  nombre: String!
  email: String!
}

type Comentario {
  id: ID!
  texto: String!
  autor: Usuario!
}
```

El esquema


{
"id": 123,
"version": 1,
"name": "Delicious Cake",
...
}
My App
Rest API
Client
Database
GET /products/123


{
"data": {
"product": {
"name": "Delicious Cake",
"description": "Just taste it!"
}
}
}
My App
GraphQL
Client
Database
GET /graphql
query MyProduct {
product(id: 123) {
name
description
}
}

graphql.org

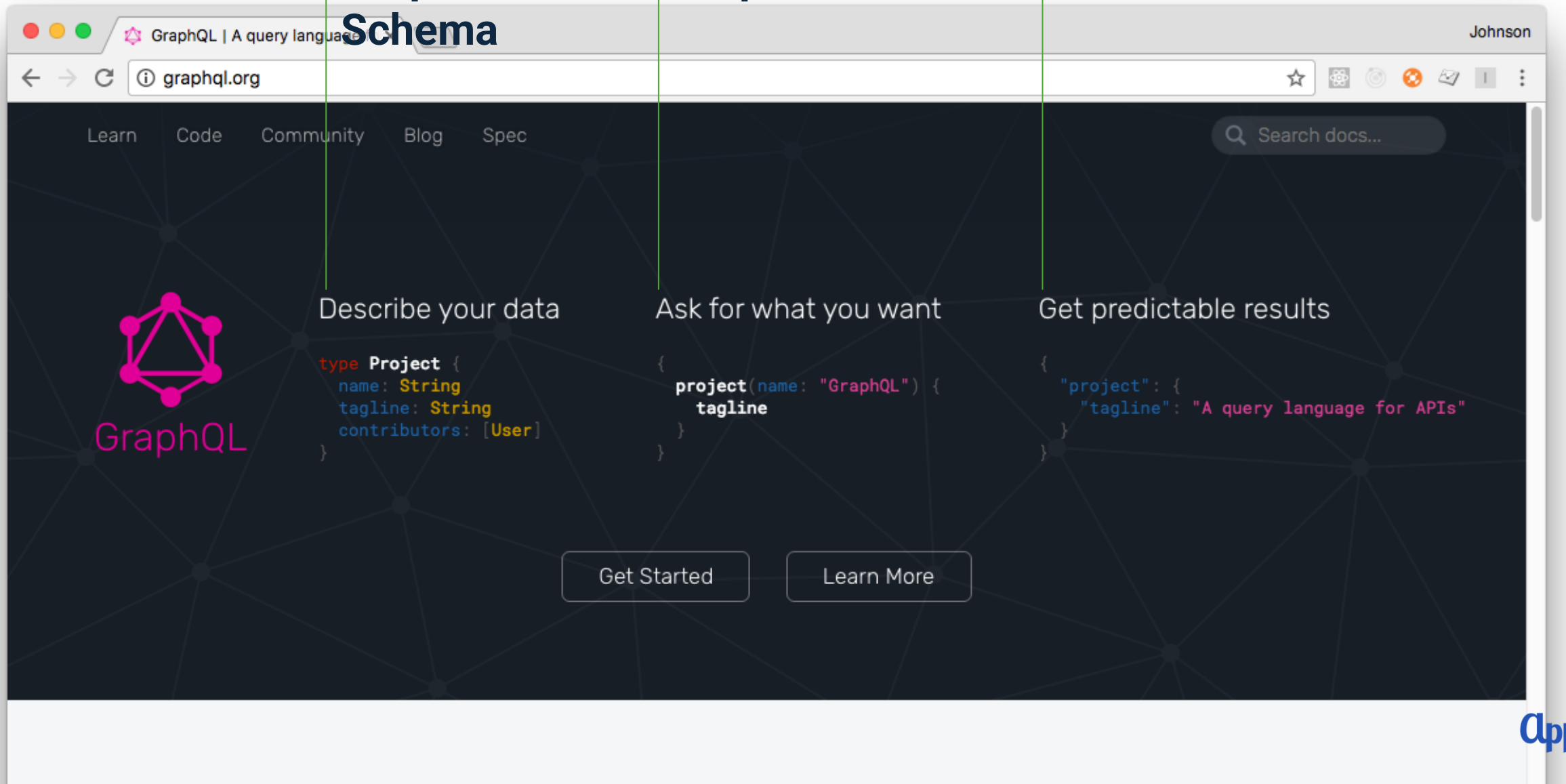


Appier

[Server]
API shape;
GraphQL Schema

[Client]
Query;
GraphQL

[Client]
Result



GraphQL

- Un lenguaje de consulta de datos
- Desarrollado por Facebook
- Utilizado internamente desde 2012
- Versión de código abierto publicada en julio de 2015
- Publicamente lanzado en agosto de 2015
  - Especificación: https://facebook.github.io/graphql
- GraphQL es un sustituto/alternativa a REST
- GrapQL utiliza un LENGUAJE DE CONSULTA (Query)
- GraphQL es un patrón, no una tecnología

GraphQL

Característica	GraphQL	REST
Endpoint	Único	Múltiples
Búsqueda de Datos	Definida por el cliente	Definida por el servidor
Over/Under-fetching	No ocurre	Problema común
Tipado	Fuerte (Esquema)	Débil (OpenAPI)
Versionado	No es necesario	Requerido (/v2)

GraphQL – Ecosistemas y herramientas

- **Servidores**
  - Apollo Server
  - GraphQL Yoga
  - Hasura
  - Graphene (Python)
- **Clientes**
  - Apollo Client
  - Relay
  - urql
  - fetch simple
- **Herramientas**
  - GraphiQL
  - Apollo Studio

https://slides.com/ajdaniel/graphql

https://olegilyenko.github.io/presentation-graphql-introduction/#/