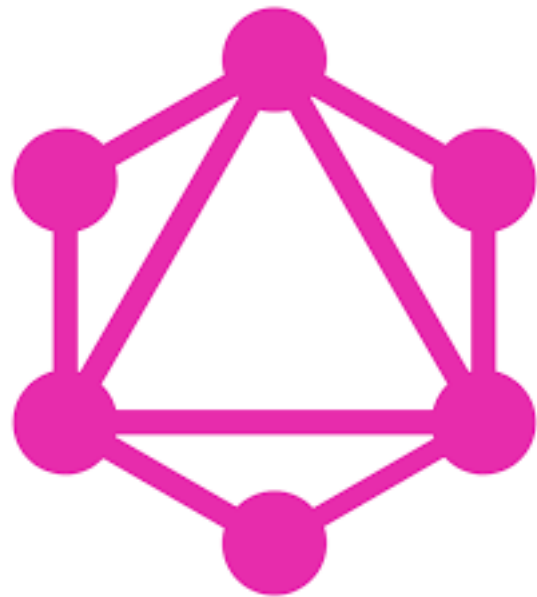
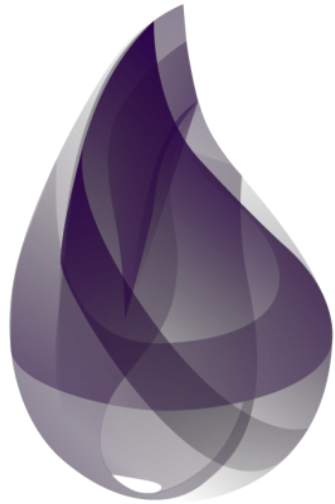


Absinthe - GraphQL demo with Elixir

Stack



+



City

Meetup

Speaker

Wrocław

Elixir Wrocław n3

Bartłomiej

...

...

...

...

Kraków

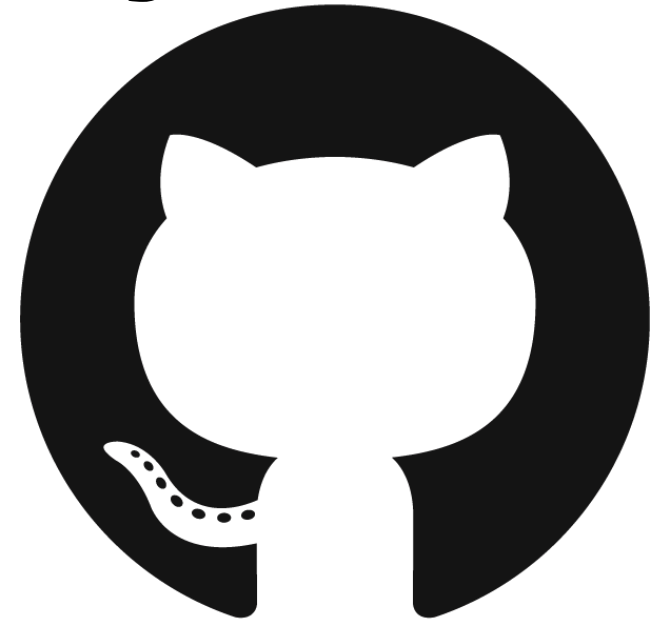
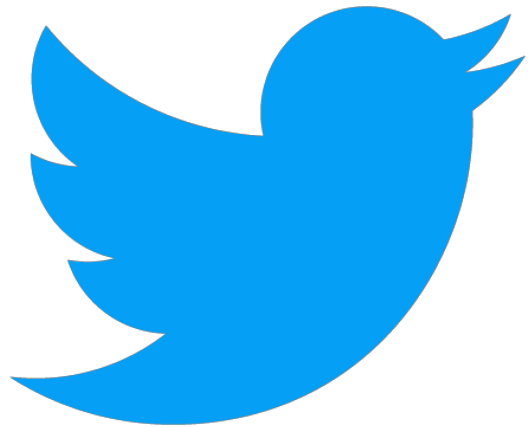
...

...

GraphQL?

- język zapytań dla API
- ~2012
- zapewnia elastyczność
- statyczne typy (schemat)
- niezależny od transportu (może być HTTP, WebSocket i inne)
- introspection
- wbudowana dokumentacja (jako kod)

GraphQL - kto tego używa?



Więcej: <https://graphql.org/users/>

Ale po co mi to?

REST

GET /cities

GET /cities/*/meetups

GET /cities/*/meetups/*/speakers

GET /meetups ?

-
-
-
-
-

GraphQL

SDL

```
1 type City {  
2   id: ID!  
3   name: String!  
4   meetups: [Meetup!]!  
5 }
```

Query Language

```
1 query {  
2   cities {  
3     id  
4     name  
5   }  
6 }
```


GraphQL - jeden graf

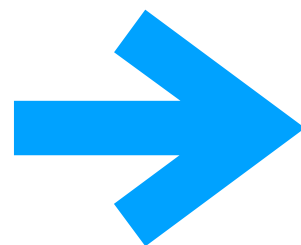
POST / api

```
1 query {  
2   cities {  
3     id  
4     name  
5     meetups {  
6       id  
7       name  
8     }  
9   }  
10 }
```

GraphQL - jeden graf

POST / api

```
1 query {  
2   cities {  
3     id  
4     name  
5     meetups {  
6       id  
7       name  
8     }  
9   }  
10 }
```



```
{  
  "data": {  
    "cities": [  
      {  
        "name": "Wrocław",  
        "meetups": [  
          {  
            "name": "Some Meetup 1",  
            "id": "3"  
          },  
          {  
            "name": "Some Meetup 1",  
            "id": "2"  
          },  
          {  
            "name": "Elixir Wrocław",  
            "id": "1"  
          }  
        ],  
        "id": "1"  
      }  
    ]  
  }  
}
```

Or...

```
id
```

```
name
```

```
meet
```

```
id
```

```
title
```

```
}
```

```
}
```

```
}
```

```
id
```

```
name
```

```
likes
```

```
description
```

```
location
```

```
speakers
```

```
...
```



Cannot query field "title" on type "Meetup".

Operacje

- Query
- Mutation
- Subscription

Typy w GraphQL

- int, string, boolean...
- listy
- typy wyliczeniowe
- interfejsy
- unie
- obiekty
- custom scalar types

NULL?

Demo

APOLLO Client side



- Apollo, Relay
- Wbudowane cache z normalizacją
- Śledzenie zmian w obrębie całej aplikacji
- Codegen
- Tooling!

Absinthe





- Definicja schematu
- Wykonanie zapytania

DSL

```
@desc "City"
object :city do
  @desc "Unique Id"
  field(:id, non_null(:id))

  @desc "City name ex. Wroclaw"
  field(:name, non_null(:string))

  @desc "Meetups in given city"
  field :meetups, non_null(list_of(non_null(:meetup))) do
    resolve fn city, _, _ ->
      {:ok, Meetups.list_city_meetups(city)}
    end
  end
end
end
```

DSL

```
@desc "City"
object :city do
  @desc "Unique Id"
  field(:id, non_null(:id))

  @desc "City name ex. Wroclaw"
  field(:name, non_null(:string))

  @desc "Meetups in given city"
  field :meetups, non_null(list_of(non_null(:meetup))) do
    resolve fn city, _, _ ->
      {:ok, Meetups.list_city_meetups(city)}
    end
  end
end
```

```
import graphene

class Person(graphene.ObjectType):
  first_name = graphene.String()
  last_name = graphene.String()
  full_name = graphene.String()

  def resolve_full_name(self, info):
    return '{} {}'.format(self.first_name, self.last_name)
```


SDL (od 1.5)

```
1  type City {  
2      id: ID!  
3      name: String!  
4      meetups: [Meetup!]!  
5  }
```




Integracja z Phoenixem

Wbudowane „IDE”

Query 1 

+ New Query

Name

Query 1 

URL

http://localhost:4000/api

Recent ▾






WS URL


GraphQL WS URL

Headers

+ Add

Standard ▾



GraphiQL 

History ▾

Save

```
1 query {  
2   cities {  
3     id  
4     name  
5     meetups {  
6       id  
7       name  
8     }  
9   }  
10 }
```

QUERY VARIABLES

Documentation Explorer 

 Search Schema...

A GraphQL schema provides a root type for each kind of operation.

ROOT TYPES

query: RootQueryType

Demo

Channels

- Subskrypcje
- Phoenix PubSub
- Klient JS - integracja z Apollo, Relay

Demo

Dataloader & Dataloader.Ecto

- Rozwiązuje N+1
- Batching
- Proste wczytywanie relacji przez wnioskowanie na podstawie Ecto schema

Demo

Co jeszcze?

- Analiza złożoności zapytań
- asynchroniczne resolvery
- obsługa uploadu plików

Problemy

- Integracja z kanałami oficjalnie wspierana tylko w Apollo JS
- Upload plików we własny sposób, ponownie niezgodny z Apollo, wsparcie tylko w webowej wersji
- Brak schema stitching (planowany na 1.5)
- Czasem ciężkie do rozczytania komunikaty błędów

Skąd czerpać wiedzę?



Dzięki!

Pytania?