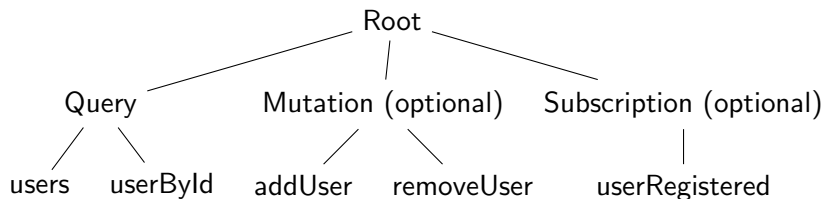# Rust and GraphQL for backend

Schema creation and basic features

Maciek Zwoliński

June 2022

Graphql Schema

```
struct User {
    id: usize,
    name: String,
    surname: String,
    age: u32,
    car_id: Optional<usize>,  // references car.id
}

enum Brand {
    Fiat,
    Ford,
    Opel,
}

struct Car {
    id: usize,
    brand: Brand,
    model: String,
    year: u32,
    owner_id: usize,  // references user.id
}
```

REST API

Overfetching

```
GET /users -> [
  {
    "id": 0,  // this is unused
    "name": "Will",
    "surname": "Smith",
    "age": 32,
    "car_id": null  // so is this
  },
  ...
]
```

Users:

▶ Will Smith, 32 years

▶ ...

REST API

Underfetching

```
GET /users -> [
  {
    "name": "Will",
    "car_id": 1,
    ... // unused
  },
  {
    "name": "Agnieszka",
    "car_id": 3,
    ... // unused
  },
  ...
]
GET /cars/1 -> {
  "brand": "Opel",
  "model": "Astra",
  "year": 1999  // unused
}
GET /cars/3 -> {
  "brand": "Ford",
  "model": "Focus",
  "year": 2010  // unused
}
```

Users:

▶ Will, Opel Astra

▶ Agnieszka, Ford Focus

▶ ...

GraphQL API
Declarative fetching

```
query {
  users {
    name,
    surname,
    age
  }
}
```

```
"users": [
  {
    "name": "Will",
    "surname": "Smith",
    "age": 32,
  },
  ...
]
```

Users:

▶ Will Smith, 32 years

▶ ...

```
query {
  users {
    name,
    car {
      brand,
      model
    }
  }
}
```

```
"users": [
  {
    "name": "Will",
    "car": {
      "brand": "Opel"
      "model": "Astra"
    }
  },
  ...
]
```

Users:

▶ Will, Opel Astra

▶ ...

REST API
Overfetching/Underfetching

```
GET /users_with_cars ?

GET /users?fields=name,surname ?

GET /users?cars=true&fields=name,surname
    &car_fields=brand,model ?
```

Server side crates:
- async-graphql
- juniper

both integrates well with popular backend crates like actix-web, axum, rocket, warp...

Client side crates:
- graphql_client
- cynic

you can choose from different designs of strong typing but subscriptions are not quite there yet