200 ok !

Error handling in 💅 GraphQL 💅

Sasha Solomon

software engineer @ Twitter

previously @ Medium

@sachee

```
{
  user(username: "@ash") {
    id
    name
  }
}
```

```
{
  "data": {
    "user": {
      "id": "268314bb7e7e",
       "name": "Ash Ketchum"
    }
  }
}
```

errors array

errors by default are propagated in the

`errors array`

on a GraphQL response

```
{
  user(username: "@ash") {
    id
    name
  }
}
```

```
{
  "data": {
    "user": null
  },
  "errors": [
    { "path": [ "user" ],
      "locations": [ { "line": 2, "column": 3 } ],
      "extensions": {
        "message": "Object not found",
        "type": 2
      }
    }
  ]
}
```

all errors treated the same

hard to know where the error came from

hard for the client to know what errors to care about

what is an error?

Internal Server Error

Bad Gateway

Deleted User

Unavailable in Country

Suspended User

what is an error?

`Internal Server Error`

≠

`Suspended User`

what is an error?

Bad Gateway

≠

Unavailable In Country

“error” categories

Internal Server Error

Bad Gateway

errors

results

Deleted User

Unavailable in Country

Suspended User

errors

/graphql

500

```
"errors": [{
  "message":
    "Server Error"
}]
```

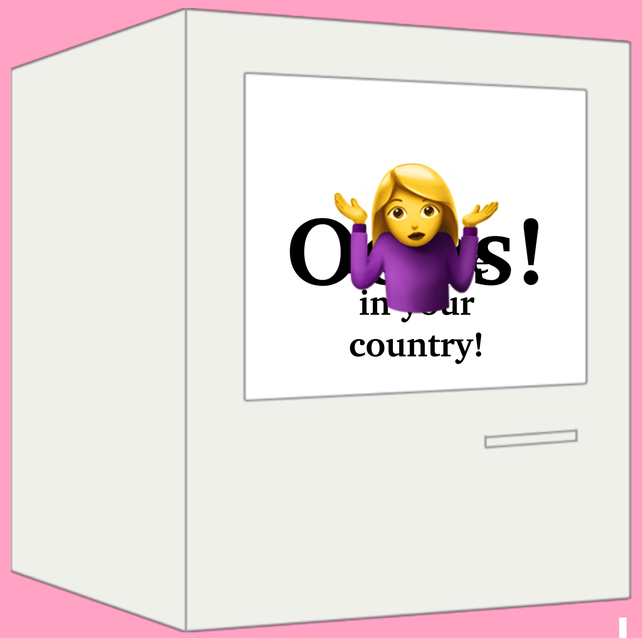
“error” categories

errors

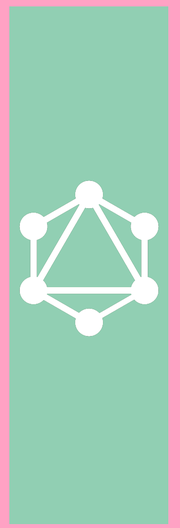
=

couldn't get requested data

results



/graphql



Suspended User

Unavailable in Country

“error” categories

results

=

got requested data

hold up

i asked for a User...

but i didn't get a User back

how is **that** the "requested data"?

we requested a User

but don't we care about these other states?

`Blocked User`

`Deleted User`

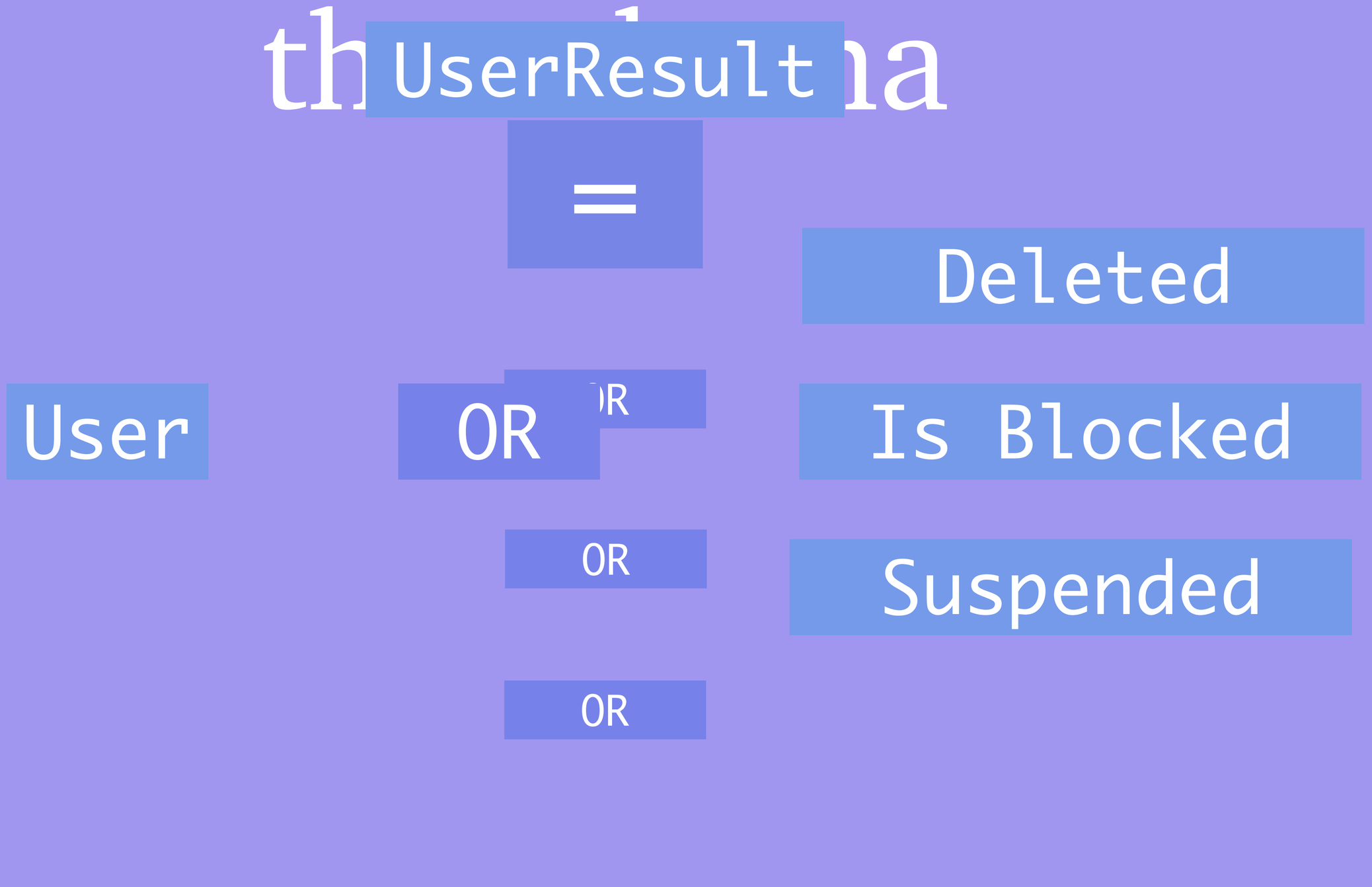
`Suspended User`

ask for what you care about

the schema

who doesn't love

 ***union types*** 


UserResult
=
Deleted
User
OR
Is Blocked
OR
Suspended
OR
OR

UserResult

User

Delet Message

Is Bloc Reason

Policy Violation Link

```graphql
type User {
  id: ID!
  name: String
}
```

```graphql
type Suspended {
  reason: String
}
type IsBlocked {
  message: String
  blockedByUser: User
}
type UnavailableInCountry {
  countryCode: Int
  message: String
}
```

```
"User, or reason we can't get one normally"
union UserResult = User | IsBlocked | Suspended
```

```
{
  userResult(username: "@ash") {
    __typename
    ... on User {
      id
      name
    }
    ... on IsBlocked {
      message
      blockedByUser {
        username
      }
    }
    ... on Suspended {
     reason
    }
}
```

```
{
  "data": {
    "userResult": {
      "__typename": "User",
      "id": "268314bb7e7e",
      "name": "Ash Ketchum"
    }
  }
}
```

```
{
  userResult(username: "@ash") {
    __typename
    ... on User {
      id
      name
    }
    ... on IsBlocked {
      message
      blockedByUser {
        username
      }
    }
    ... on Suspended {
     reason
    }
}
```

```
{
  "data": {
    "userResult": {
      "__typename": "IsBlocked",
      "message": "User blocked: @ash",
      "blockedByUser": {
        "username": "@brock"
      }
    }
  }
}
```

```
{
  userResult(username: "@ash") {
    __typename
    ... on User {
      id
      name
    }
    ... on IsBlocked {
      message
      blockedByUser {
        username
      }
    }
    ... on Suspended {
     reason
    }
}
```

```
{
  "data": {
    "user": null
  },
  "errors": [
    {
      "message": "Internal server error",
      "path": [ "user" ],
      "locations": [ { "line": 2, "column": 3 } ]
    }
  ]
}
```

results are customizable for each entity

we know where the error came from

client decides what errors it cares about and what errors it can ignore

more complex system

What happens when our schema structure is more complex?

```
"User, or reason we can't get one normally"
union UserResult = User | IsBlocked | Suspended
```

```
type User {
  id: ID!
  name: String
}
```

```
type Suspended {
  reason: String
}
type IsBlocked {
  message: ID!
  blockedByUser: User
}
type UnavailableInCountry {
  countryCode: Int
  message: String
}
```

```
type User {
  id: ID!
  name: String
}
type Suspended {
  reason: String
}
type IsBlocked {
  message: ID!
  blockedByUser: User
}
type UnavailableInCountry {
  countryCode: Int
  message: String
}
```

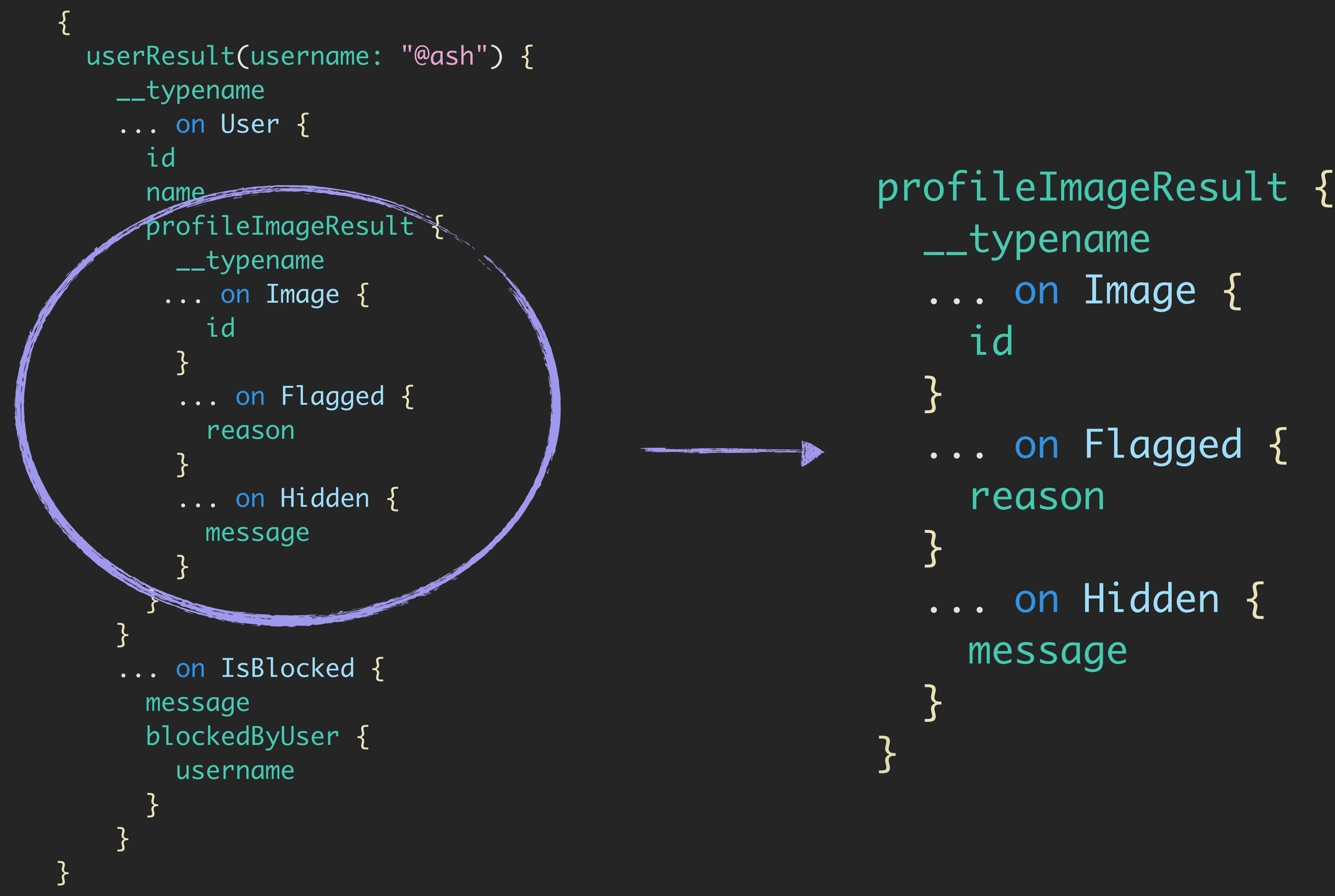
```
type ProfileImage {
  id: ID!
  value: String
}
type Flagged {
  reason: String
}
type Hidden {
  message: String
}
```

```
"Image, or reason we can't get one normally"
union ImageResult = Image | Flagged | Hidden
```

```
type User {
  id: ID!
  profileImageResult: ImageResult
}
type Suspended {
  reason: String
}
type IsBlocked {
  message: ID!
  blockedByUser: User
}
type UnavailableInCountry {
  countryCode: Int
  message: String
}
```

```
type ProfileImage {
  id: ID!
  value: String
}
type Flagged {
  reason: String
}
type Hidden {
  message: String
}
```

added additional Result Types for our new Object Type


{
userResult(username: "@ash") {
__typename
... on User {
id
name
profileImageResult {
__typename
... on Image {
id
}
... on Flagged {
reason
}
... on Hidden {
message
}
}
}
... on IsBlocked {
message
blockedByUser {
username
}
}
}
profileImageResult {
__typename
... on Image {
id
}
... on Flagged {
reason
}
... on Hidden {
message
}
}

```
{
  userResult(username: "@ash") {
    __typename
    ... on User {
      id
      name
      profileImageResult {
        __typename
       ... on Image {
          id
        }
        ... on Flagged {
          reason
        }
        ... on Hidden {
          message
        }
      }
    }
    ... on IsBlocked {
      message
      blockedByUser {
        username
      }
    }
}
```

```
{
  "data": {
    "userResult": {
      "__typename": "User",
      "id": "268314bb7e7e",
      "name": "Ash Ketchum",
      "profileImageResult": {
        "__typename": "Image",
        "id": "982642"
      }
    }
  }
}
```

```
{
  userResult(username: "@ash") {
    __typename
    ... on User {
      id
      name
      profileImageResult {
        __typename
       ... on Image {
          id
        }
        ... on Flagged {
          reason
        }
        ... on Hidden {
          message
        }
      }
    }
    ... on IsBlocked {
      message
      blockedByUser {
        username
      }
    }
}
```

```
{
  "data": {
    "userResult": {
      "__typename": "User",
      "id": "268314bb7e7e",
      "name": "Ash Ketchum",
      "profileImageResult": {
        "__typename": "Flagged",
        "reason": "Copyrighted image"
      }
    }
  }
}
```

results are customizable for each entity

we know where the result came from

client decides what results it cares about and what results it can ignore

"errors" don't cause failures in nested queries

can tune how verbose we want results to be

more accurately represent our data

more than just two categories of "errors"

you can customize as much as you want

results

Medium

is using this functionality in production right now

is working on using this functionality in the future

thanks!

@sachee

@sachee

something went wrong

Internal Server Error

Bad Gateway

errors

results

Deleted User

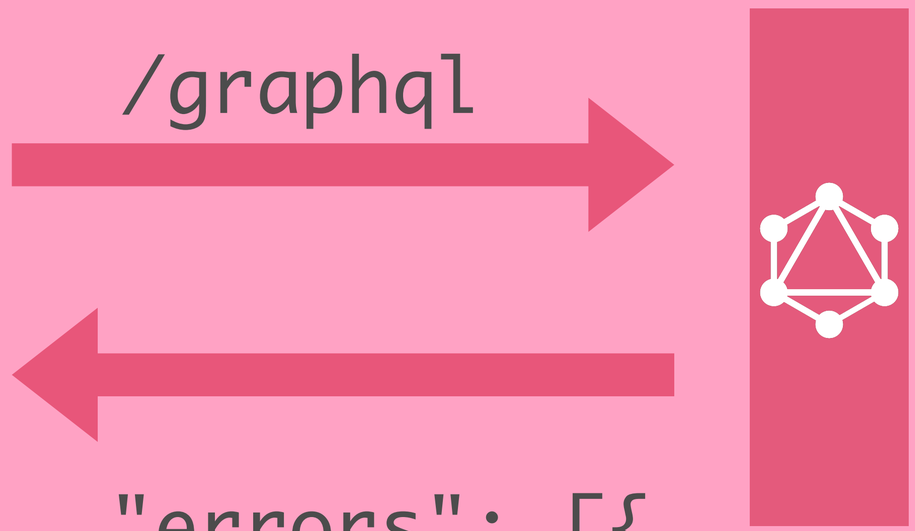
Unavailable in Country

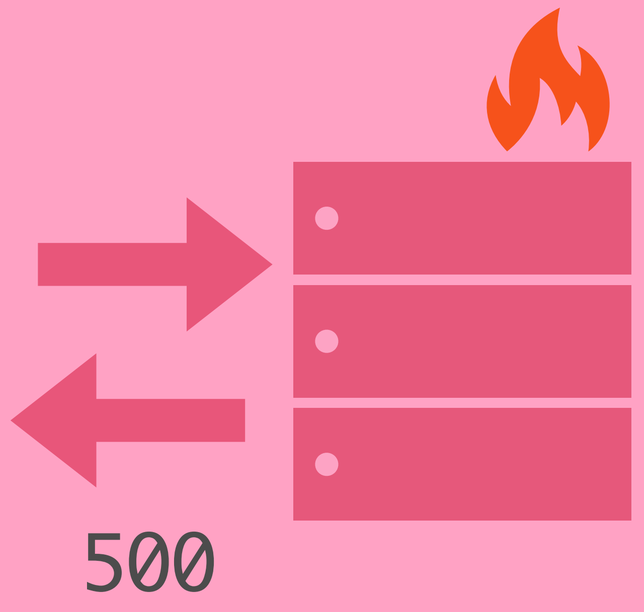
Suspended User

nothing went wrong

errors

/graphql

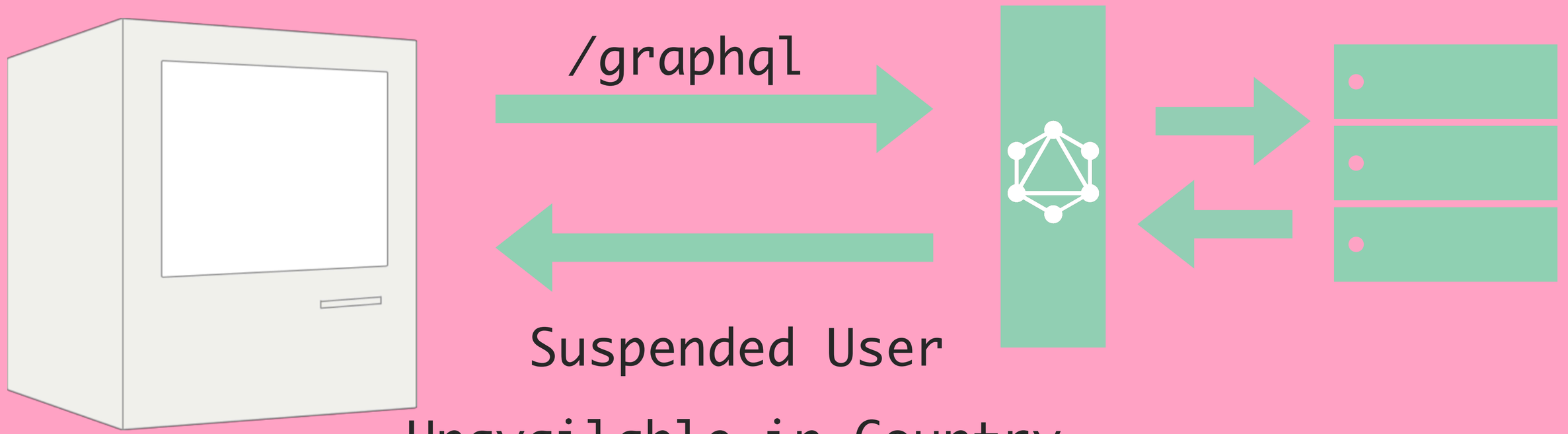




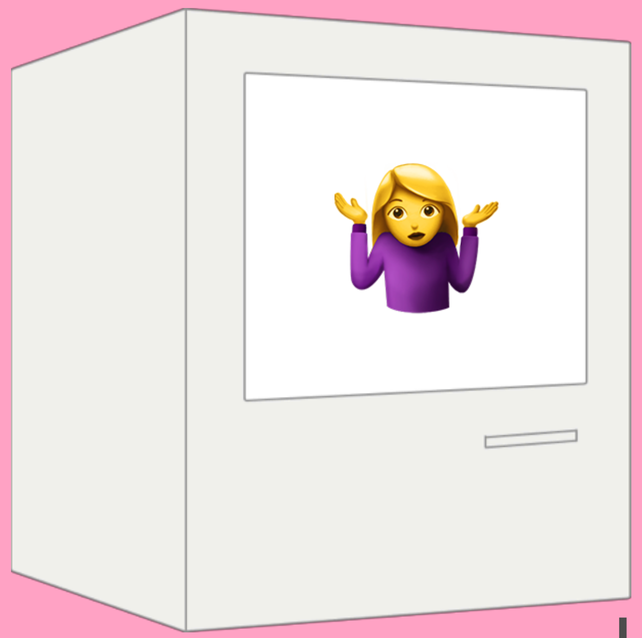
500

```
"errors": [{
  "message":
    "Server Error"
}]
```

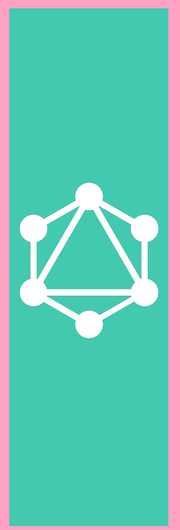
results

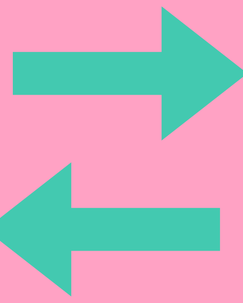


results



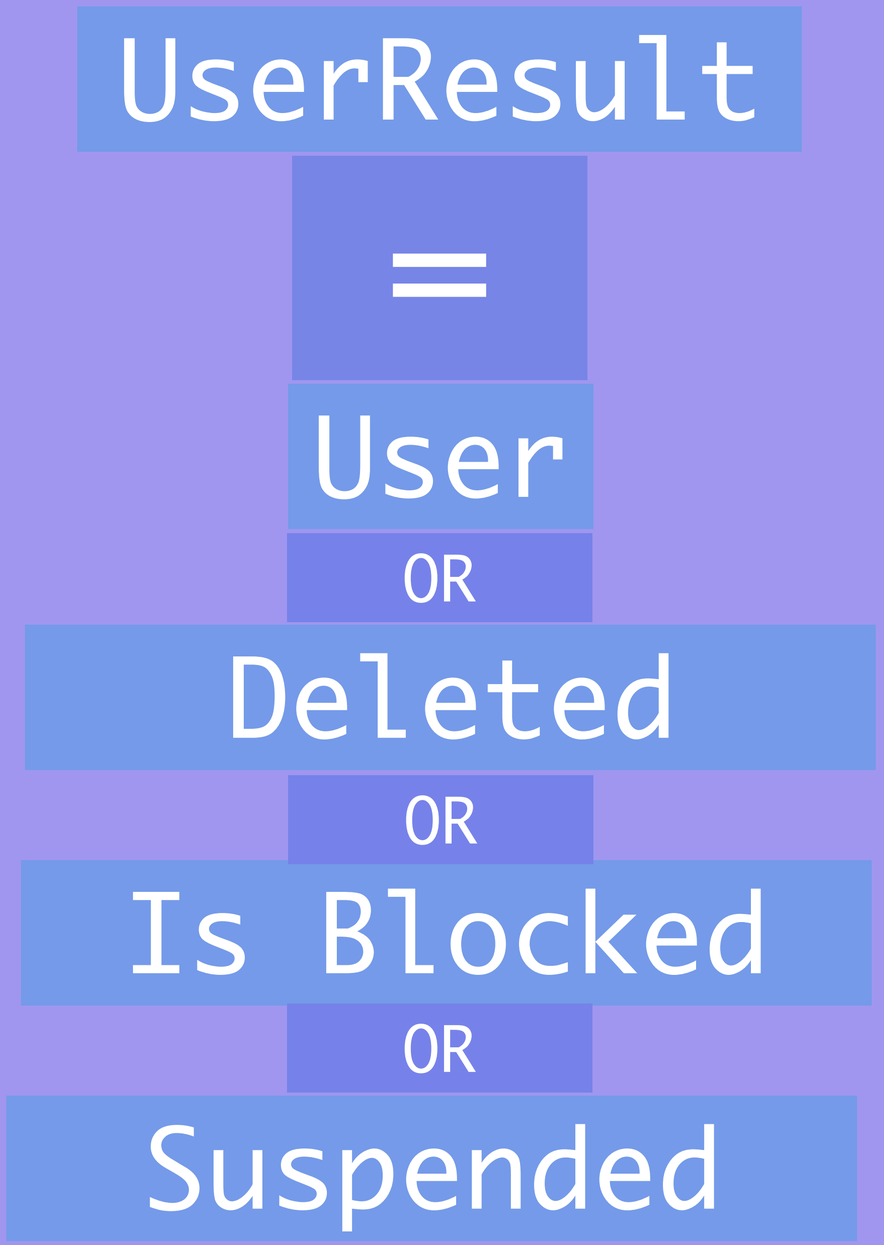
/graphql





Suspended User

Unavailable in Country


UserResult
=
User
OR
Deleted
OR
Is Blocked
OR
Suspended

UserResult

User

Deleted

Message

Is Blocked

Reason

Suspended

Policy Violation Link