GRAPHQL

Stanko Krtalic Rusendic

github.com/Stankec
@monorkin


Floating
Point

01

REST PROBLEMS

Representational state transfer

1 MINUTE AGO

STANKO

Hi

OTTO

STANKO

Cool!

17:56 · Delivered

OTTO

TYPE A MESSAGE

GIF

IN THEORY

●○○○○ bonbon 17:57 84%

OTTO THE BOT

1 MINUTE AGO

STANKO

Hi

OTTO

STANKO

Cool!

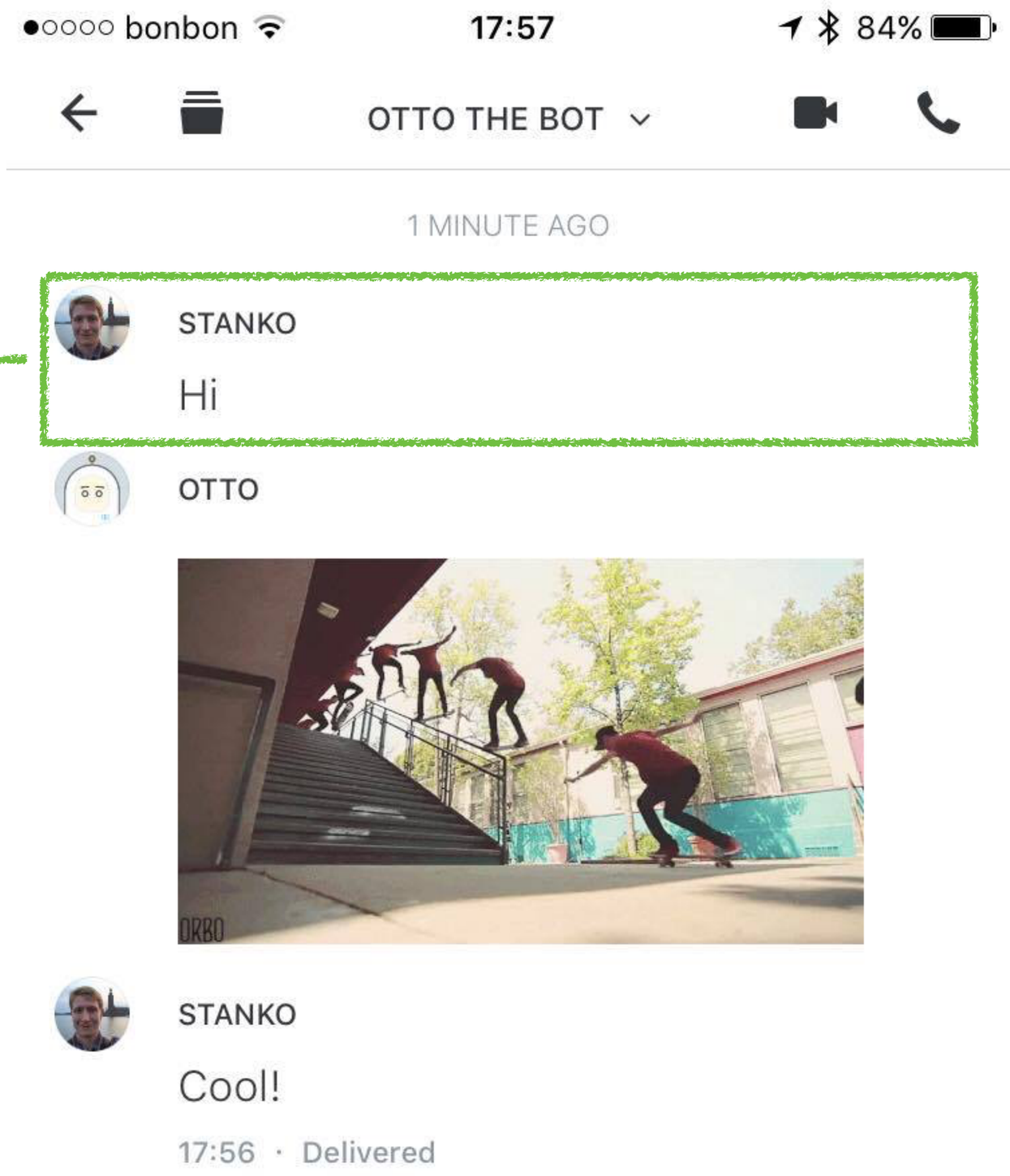
17:56 · Delivered

/api/conversations/53

```
{
  name: “Otto the bot”,
  participant_user_ids: [1, 1337],
  last_message_sent_at: 1337002,
  archived: false
}
```

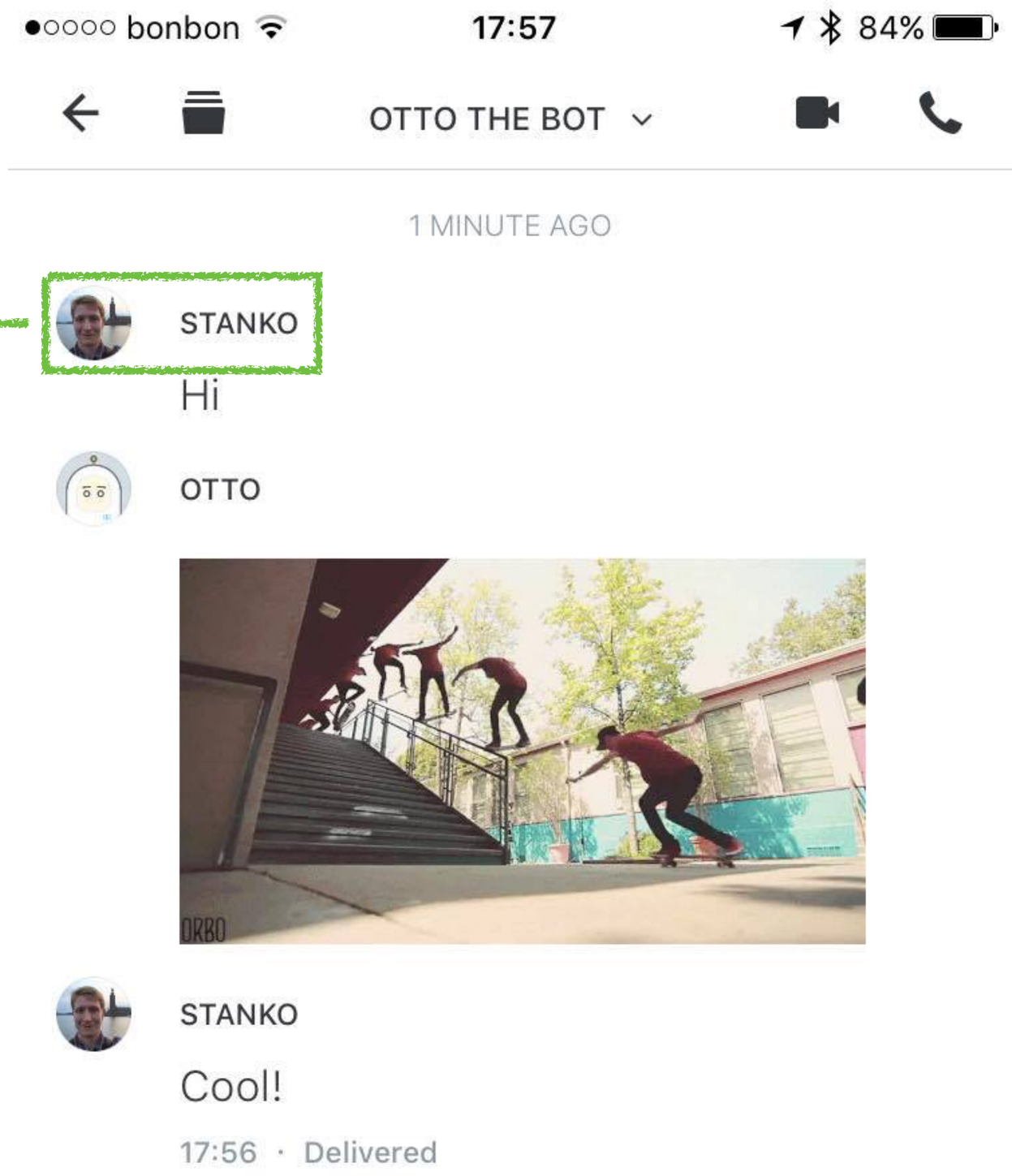
/api/conversations/53/message/1

```
{
  message_type: “plain”,
  body: “Hi”,
  sender_id: “1337”,
  created_at: 1337001,
  updated_at: 1337001,
  previous_version_ids: [],
  status: “delivered”,
  seen_by_participant_ids: [1]
}
```

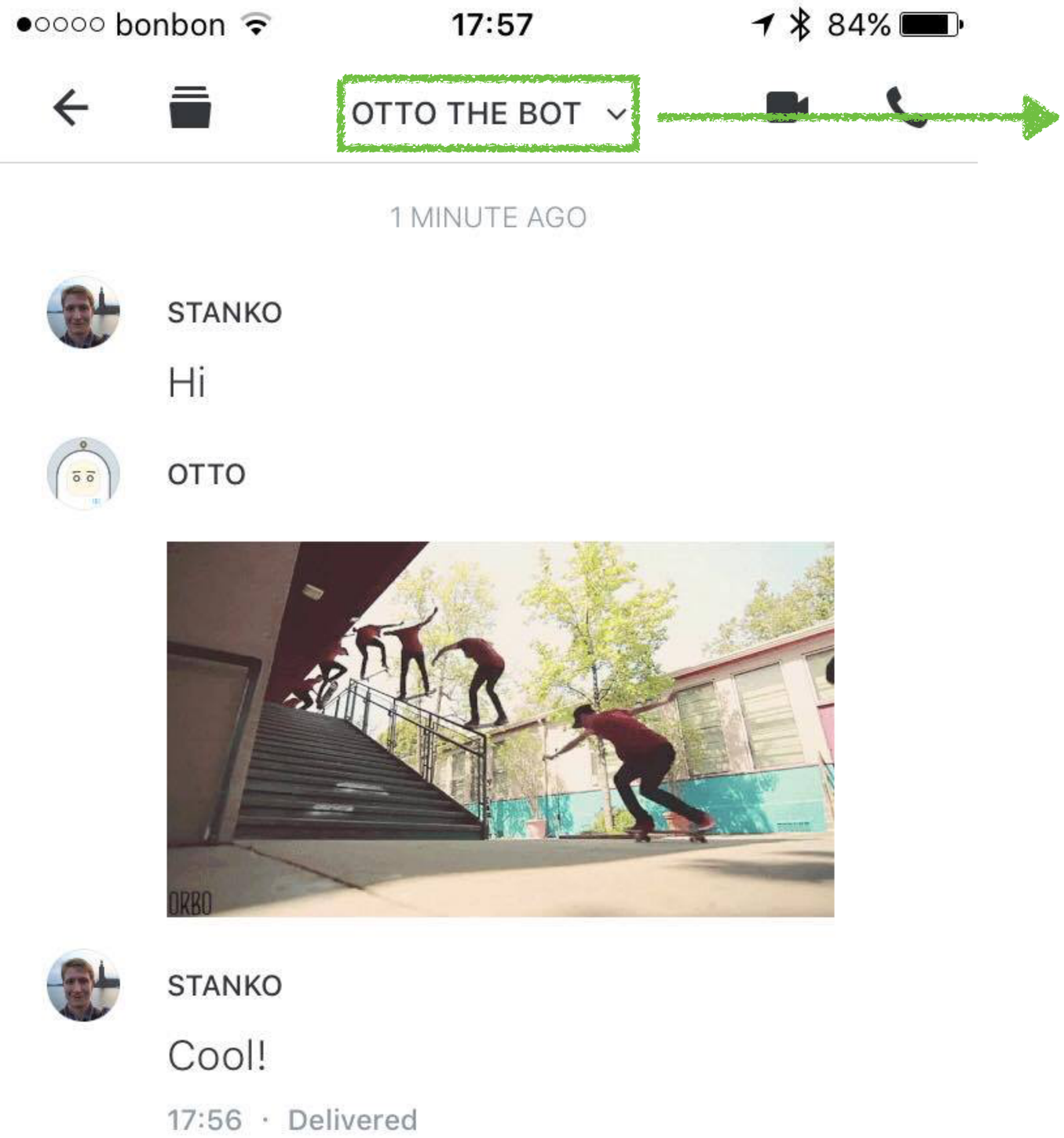


/api/users/1337

```
{
  first_name: “Stanko”,
  last_name: “Krtalic Rusendic”,
  avatar_url: “https://…”,
  last_online_at: 1337001,
  last_signin_ip: 133.10.45.99,
  sex: “male”,
  timezone: “GMT+1”,
  last_known_location: “16.0E45.0N”
}
```



IN REALITY

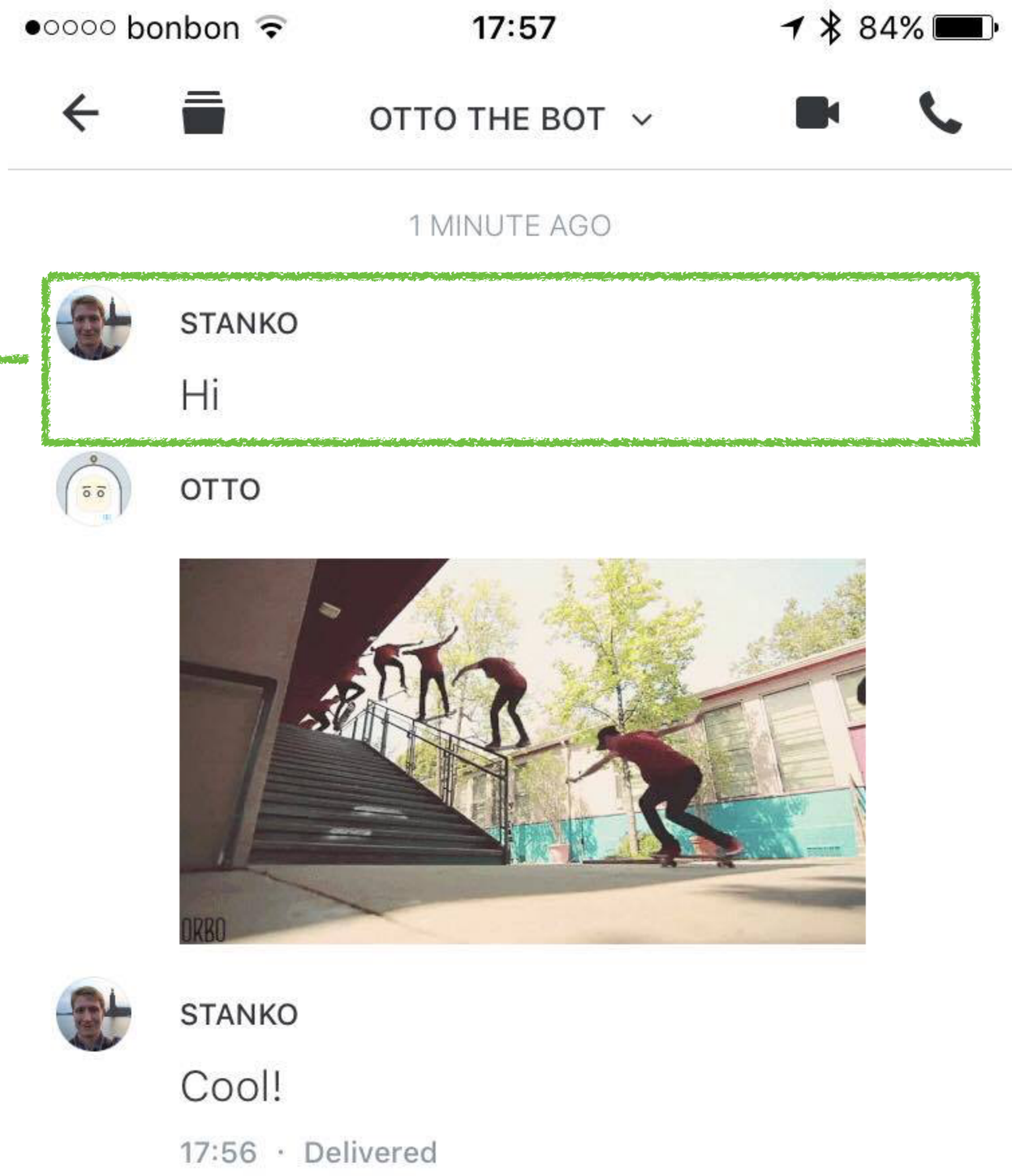


/api/conversations/53

```
{
  name: “Otto the bot”,
  participant_user_ids: [1, 1337],
  last_message_sent_at: 1337002,
  last_message_sent_by: {
    first_name: “Stanko”,
    last_name: “Krtalic Rusendic”
  }
  archived: false,
  last_message: {
    type: “plain”,
    body: “Cool!”,
    status: “delivered”
  }
}
```

/api/conversations/53/message/1

```
{
  message_type: “plain”,
  body: “Hi”,
  sender_id: “1337”,
  created_at: 1337001,
  updated_at: 1337001,
  previous_version_ids: [],
  status: “delivered”,
  seen_by_participants: [
    { first_name: “Otto” }
  ],
  sender: {
    first_name: “Stanko”,
    avatar_url: “https://…”
  }
}
```

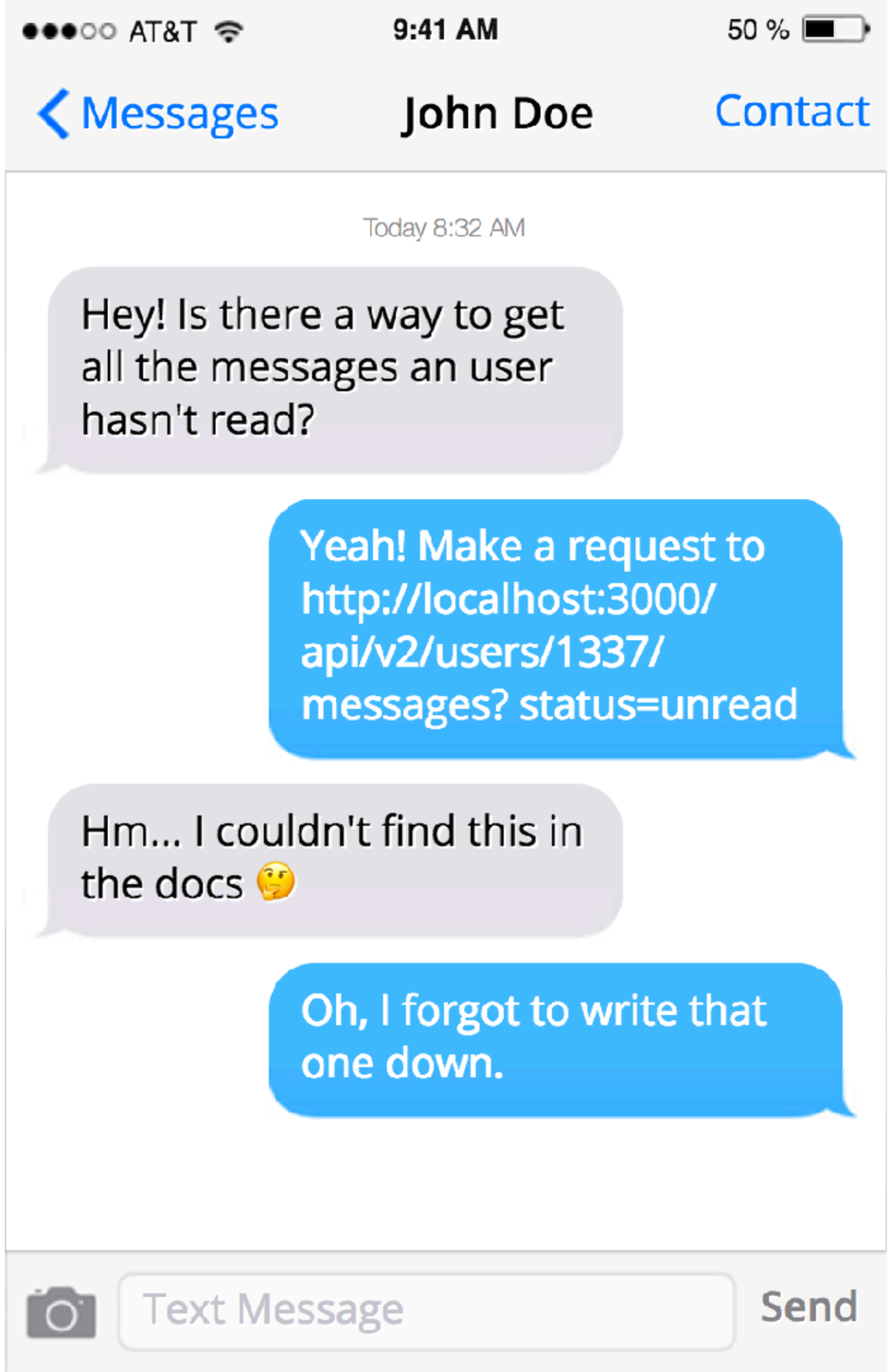


REST is pointless if your endpoints respond to a screen in your app

JSON::API
HAL

02

DOCUMENTATION


AT&T
9:41 AM
50 %
Messages
John Doe
Contact
Today 8:32 AM
Hey! Is there a way to get all the messages an user hasn't read?
Yeah! Make a request to http://localhost:3000/api/v2/users/1337/messages? status=unread
Hm... I couldn't find this in the docs 🤔
Oh, I forgot to write that one down.
Text Message
Send

Documentation is boring.

Documentation will always be lacking,
or at least lag behind the
implementation

Swagger
Apiary
API Blueprint
(documentation derived from tests)

Require manual labor

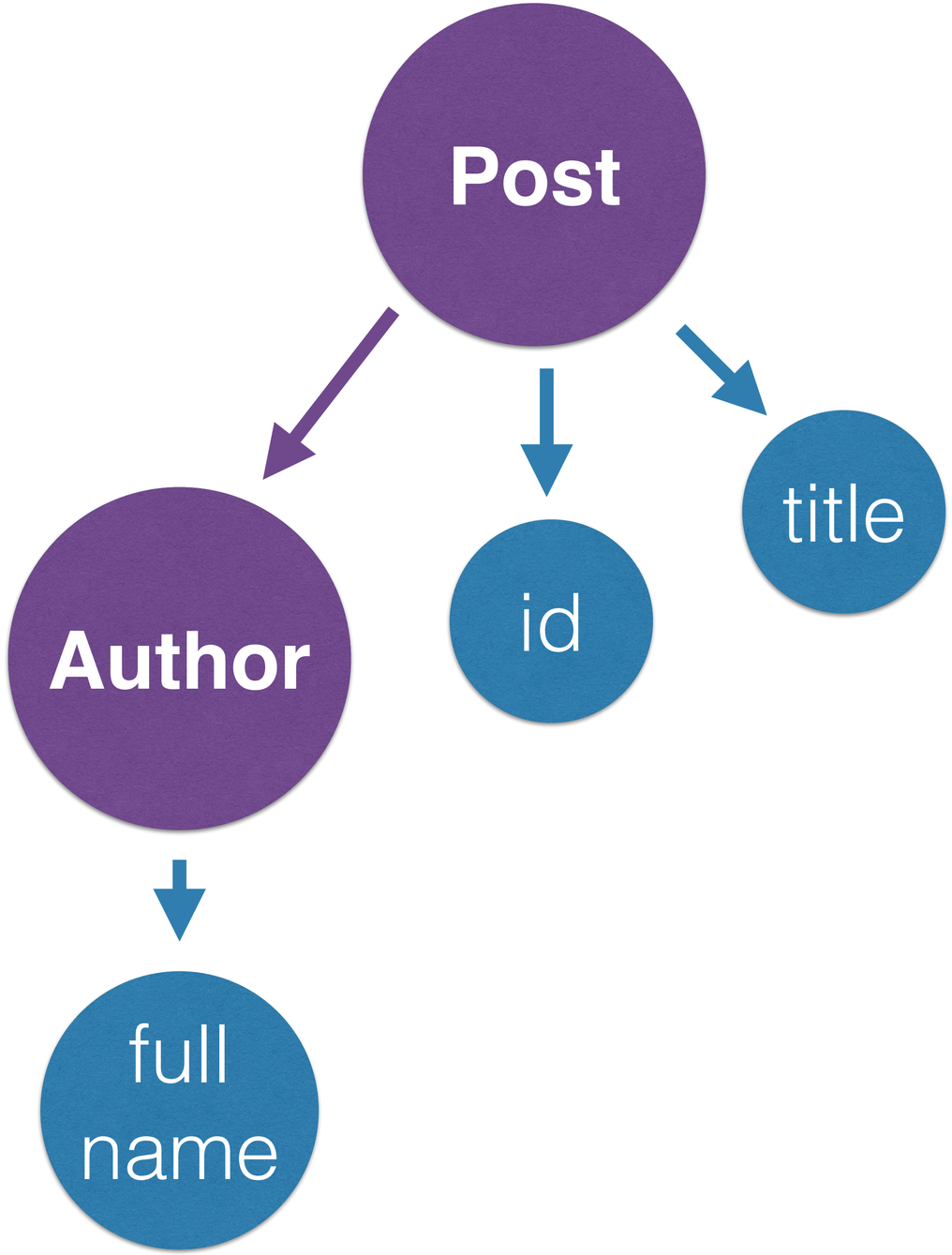
Laborious

03

GRAPHQL

Think of your resources as endpoints of a graph

```
query {
  posts {
    id
    title
    created_at
    author {
      full_name
    }
  }
}
```



```
SELECT "posts".id, "posts".title, "posts".body, "posts".created_at
FROM "posts"
ORDER BY "posts".created_at DESC
```

```
query {
  posts {
    id
    title
    created_at
    author {
      full_name
    }
  }
}
```

```
{
  "data": {
    "posts": [
      {
        "id": "1",
        "title": "Lilies of the Field",
        "created_at": "2017-02-25T18:45:29Z",
        "author": {
          "full_name": "Stanko Krtalic Rusendic"
        }
      },
```

String
Boolean
Int
Float
ID (String or Int)

Post

A blog post

FIELDS

author: Author!

body: String!

comment_count: Int!

comments: [Comment!]

created_at: DateTime!

id: ID!

title: String!

Queries

Mutations

Query-Mutations

```ruby
# Query method - given an id it returns the coresponding user
def get_user(id)
end

# Mutation method - given the inputs it creates a new user, but returns nothing
def create_user(first_name, last_name, email)
end

# Query-Mutation - given the inputs it returns the modified user object
def update_user(first_name, last_name, email)
end

```

Queries

~~Mutations~~

Query-Mutations

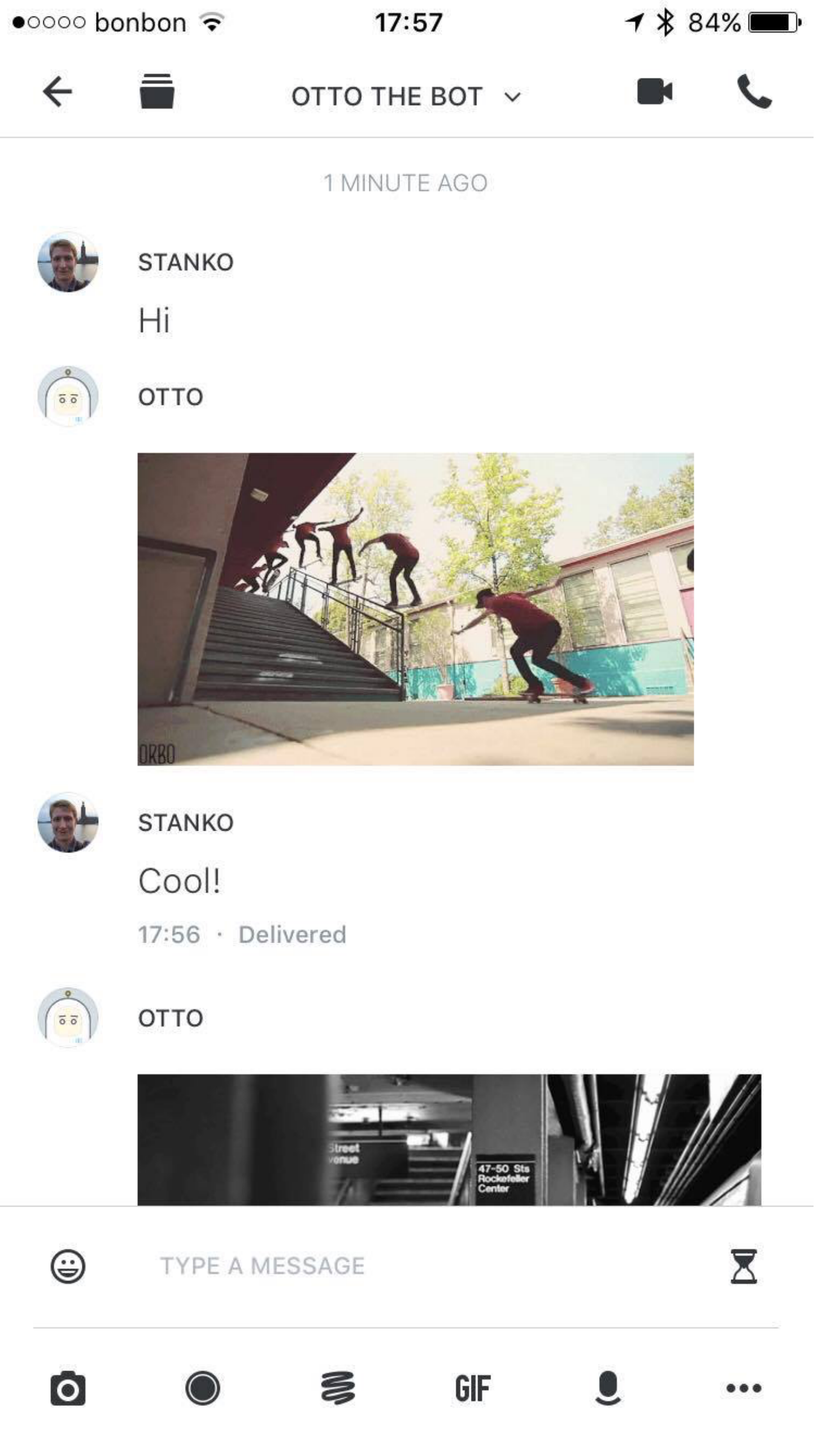
```
query {
  post(id: 1){
    id
    title
    created_at
    author {
      full_name
    }
    comments {
      author {
        first_name
      }
      body
    }
  }
}
```

```
{
  "data": {
    "post": {
      "id": "1",
      "title": "Lilies of the Field",
      "created_at": "2017-02-25T18:45:29Z",
      "author": {
        "full_name": "Stanko Krtalic Rusendic"
      },
      "comments": [
        {
          "author": {
            "first_name": "Dario"
          },
          "body": "It was summer… and it was hot. Rachel was there… A
lonely grey couch…”OH LOOK!” cried Ned, and then the kingdom was his
forever. The End."
        },
        {
          "author": {
            "first_name": "Tomislav"
          },
          "body": "Raspberries? Good. Ladyfingers? Good. Beef? GOOD!"
        },
```

```
mutation {
  createComment(input: {
    postId: 1
    authorId: 1
    body: "This was created using GraphQl 🙌"
  })
  {
    post {
      comments {
        body
      }
    }
  }
}
```

```
{
  "data": {
    "createComment": {
      "post": {
        "comments": [
          {
            "body": "This was created with GraphQL 🙌"
          }
        ]
      }
    }
  }
}
```

Back to the original example




```
query {
  conversation(id: 53) {
    name
    messages {
      author {
        fullName
        avatar {
          url
        }
      }
      body
      createdAt
      status
    }
  }
}
```

Everything is served from one endpoint

http://localhost/graphql

```
query {
  post_1: post(id: "1") {
    ...postFields
  },
  post_2: post(id: "2") {
    ...postFields
  }
  post_3: post(id: "3") {
    ...postFields
  }
}

fragment postFields on Post {
  title
  author {
    full_name
  }
  body
  comment_count
}
```

```
{
  "data": {
    "post_1": {↔},
    "post_2": {↔},
    "post_3": {
      "title": "I Sing the Body Electric",
      "author": {
        "full_name": "Stanko Krtalic Rusendic"
      },
      "body": "If we override the driver, we can get to the GB
application through the multi-byte XML alarm!\nWe need to index the 1080p
RAM interface!\nTry to transmit the AI bus, maybe it will connect the
digital interface!\nYou can't connect the interface without connecting
the solid state SQL driver!\nUse the optical EXE bus, then you can
compress the haptic alarm!\nThe PCI interface is down, input the open-
source capacitor so we can back up the SMTP feed!\nOverriding the pixel
won't do anything, we need to synthesize the virtual usb card!\nI'll
connect the 1080p RSS driver, that should bandwidth the GB array!",
      "comment_count": 5
    }
  }
}
```

```graphql
query HeroNameAndFriends($episode: Episode) {
  hero(episode: $episode) {
    name
    friends {
      name
    }
  }
}
```

VARIABLES

```json
{
  "episode": "JEDI"
}
```

```json
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        }
```

```
query Hero($episode: Episode, $withFriends: Boolean!)
  hero(episode: $episode) {
    name
    friends @include(if: $withFriends) {
      name
    }
  }
}
```

VARIABLES

```
{
  "episode": "JEDI",
  "withFriends": false
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2"
    }
  }
}
```

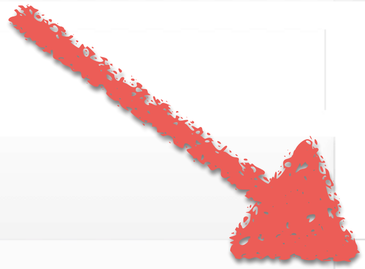
```
{
  search(text: "an") {
    __typename
    ... on Human {
      name
    }
    ... on Droid {
      name
    }
    ... on Starship {
      name
    }
  }
}
```

```
{
  "data": {
    "search": [
      {
        "__typename": "Human",
        "name": "Han Solo"
      },
      {
        "__typename": "Human",
        "name": "Leia Organa"
      },
      {
        "__typename": "Starship",
        "name": "TIE Advanced x1"
      }
    ]
```

GraphiQL

/localhost:3000/graphql

Method POST Edit HTTP Headers



```
{
  "data": {
    "post_1": {↔},
    "post_2": {↔},
    "post_3": {
      "title": "I Sing the Body Electric",
      "author": {
        "full_name": "Stanko Krtalic Rusendic"
      },
      "body": "If we override the driver, we can get to the GB
application through the multi-byte XML alarm!\nWe need to index the 1080p
RAM interface!\nTry to transmit the AI bus, maybe it will connect the
digital interface!\nYou can't connect the interface without connecting
the solid state SQL driver!\nUse the optical EXE bus, then you can
compress the haptic alarm!\nThe PCI interface is down, input the open-
source capacitor so we can back up the SMTP feed!\nOverriding the pixel
won't do anything, we need to synthesize the virtual usb card!\nI'll
connect the 1080p RSS driver, that should bandwidth the GB array!",
      "comment_count": 5
    }
  }
}
```

Documentation Explorer

Search the schema ...

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

ROOT TYPES

query: Query

mutation: Mutation

Schema definition language

```
type Character {
  name: String!
  appearsIn: [Episode]!
}

type Query {
  hero(episode: Episode): Character
  droid(id: ID!): Droid
}
```

```
enum Episode {
  NEWHOPE
  EMPIRE
  JEDI
}

union SearchResult = Human | Droid | Starship
```

```graphql
type Human implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  starships: [Starship]
  totalCredits: Int
}

type Droid implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  primaryFunction: String
}
```

```graphql
interface Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
}
```

GRAPHQL

Relay

Absinthe

This presentation is available at

github.com/stankec/lectures

Questions?