The future of REST in Drupal

@e0ipso | Mateu Aguiló

Lullabot

What degree of REST holiness?

60%

Of the sites **will not need** most of this.

≈80%

Of the decoupled traffic **will need this**.

What to expect

From this talk

- What are the problems
- What are the solutions to those
- How Drupal gets there

When you leave the room

You should know that…

There are missing pieces for REST

There are known solutions for those

Contrib is going to fill the gap

I hope you learn…

To use an API more efficiently

To generate a flexible API

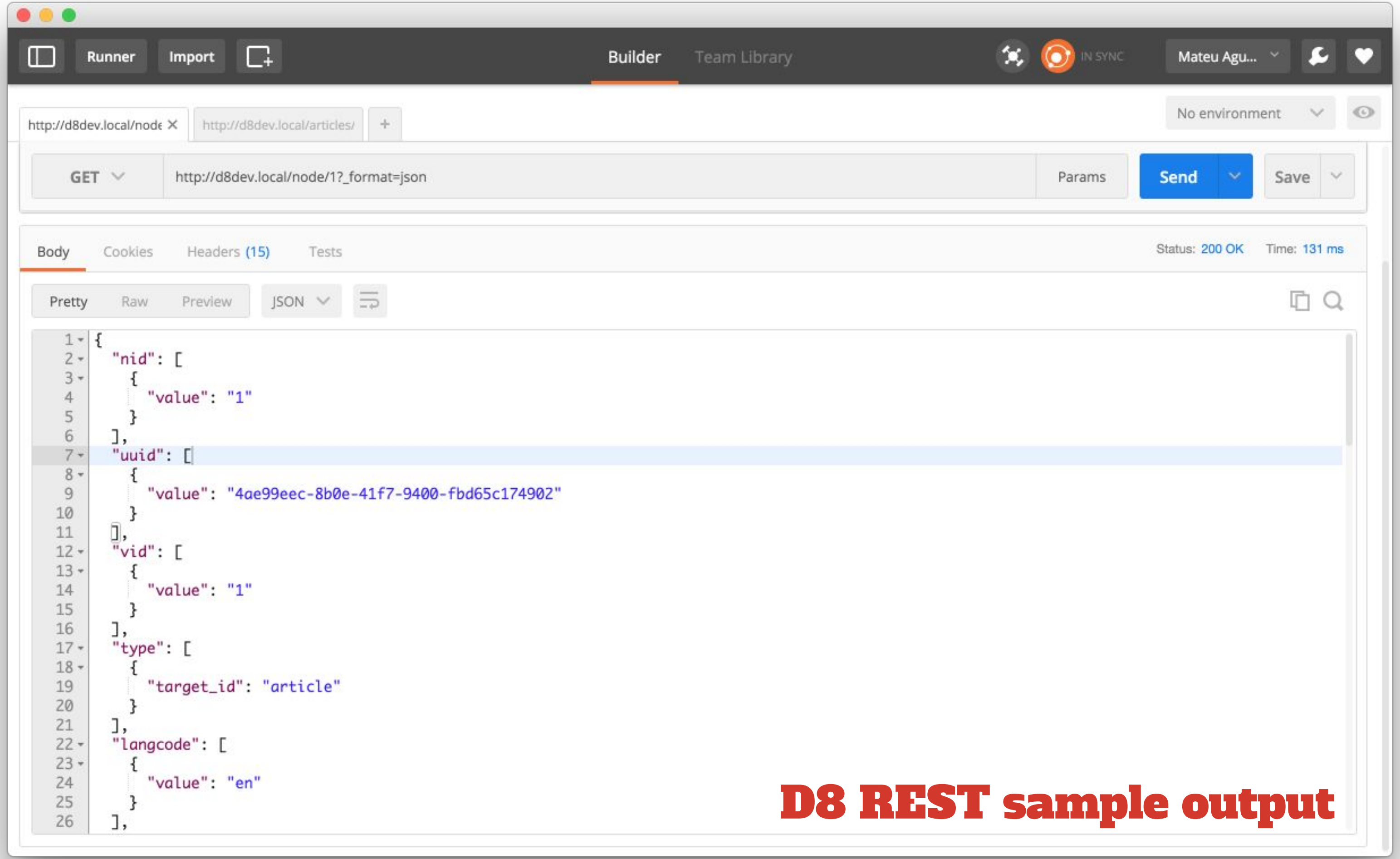
About JSONAPI, GraphQL, Collection JSON, …

You are unique snowflakes

REST in Drupal 8

Is truly awesome and flexible.

- Flexible output (code)
- Support for entities
- Custom routes (code)
- Media type negotiation
- Access control
- Authentication providers



GET http://d8dev.local/node/1?_format=json

Status: 200 OK Time: 131 ms

```
{
  "nid": [
    {
      "value": "1"
    }
  ],
  "uuid": [
    {
      "value": "4ae99eec-8b0e-41f7-9400-fbd65c174902"
    }
  ],
  "vid": [
    {
      "value": "1"
    }
  ],
  "type": [
    {
      "target_id": "article"
    }
  ],
  "langcode": [
    {
      "value": "en"
    }
  ],
```

D8 REST sample output

```
{
  "nid": [{"value": "1"}],
  "uuid": [{"value": "4ae99eec-8b0e-41f7-9400"}],
  "type": [{"target_id": "article"}],
  "langcode": [{"value": "en"}],
  "title": [{"value": "Test"}],
  "created": [{"value": "1450276200"}],
  "changed": [{"value": "1450276218"}],
  "body": [{
    "value": "<p>test</p>\r\n",
    "format": "basic_html",
    "summary": ""
  }]
}
```

(simplified) Response body

API design

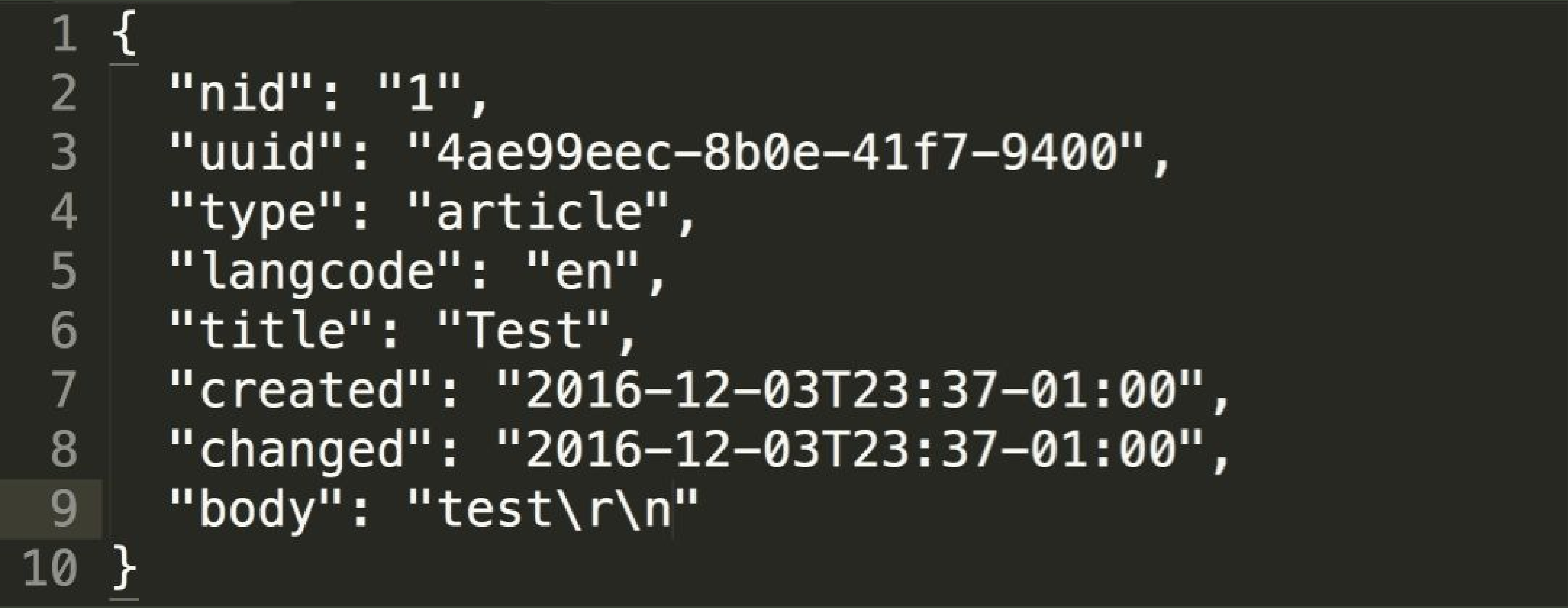
Problem 1: API design

WHAT

- Specify the format
- Decide what to expose
- Computed fields
- Preprocess fields

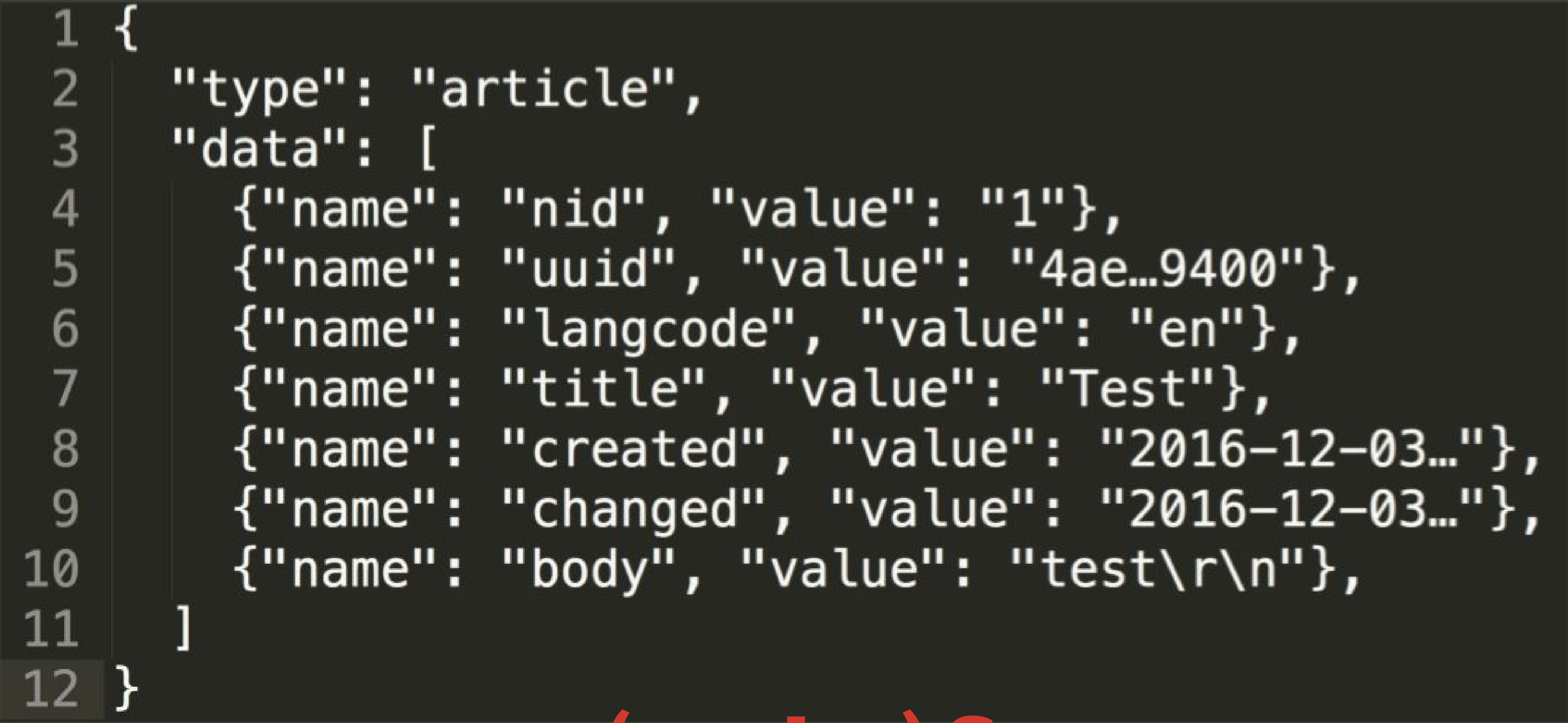
WHY

- Readability / *Writeability*
- Replaceability
- Meet existing API design
- Existing tools



```json
{
    "nid": "1",
    "uuid": "4ae99eec-8b0e-41f7-9400",
    "type": "article",
    "langcode": "en",
    "title": "Test",
    "created": "2016-12-03T23:37-01:00",
    "changed": "2016-12-03T23:37-01:00",
    "body": "test\r\n"
}
```

Custom response

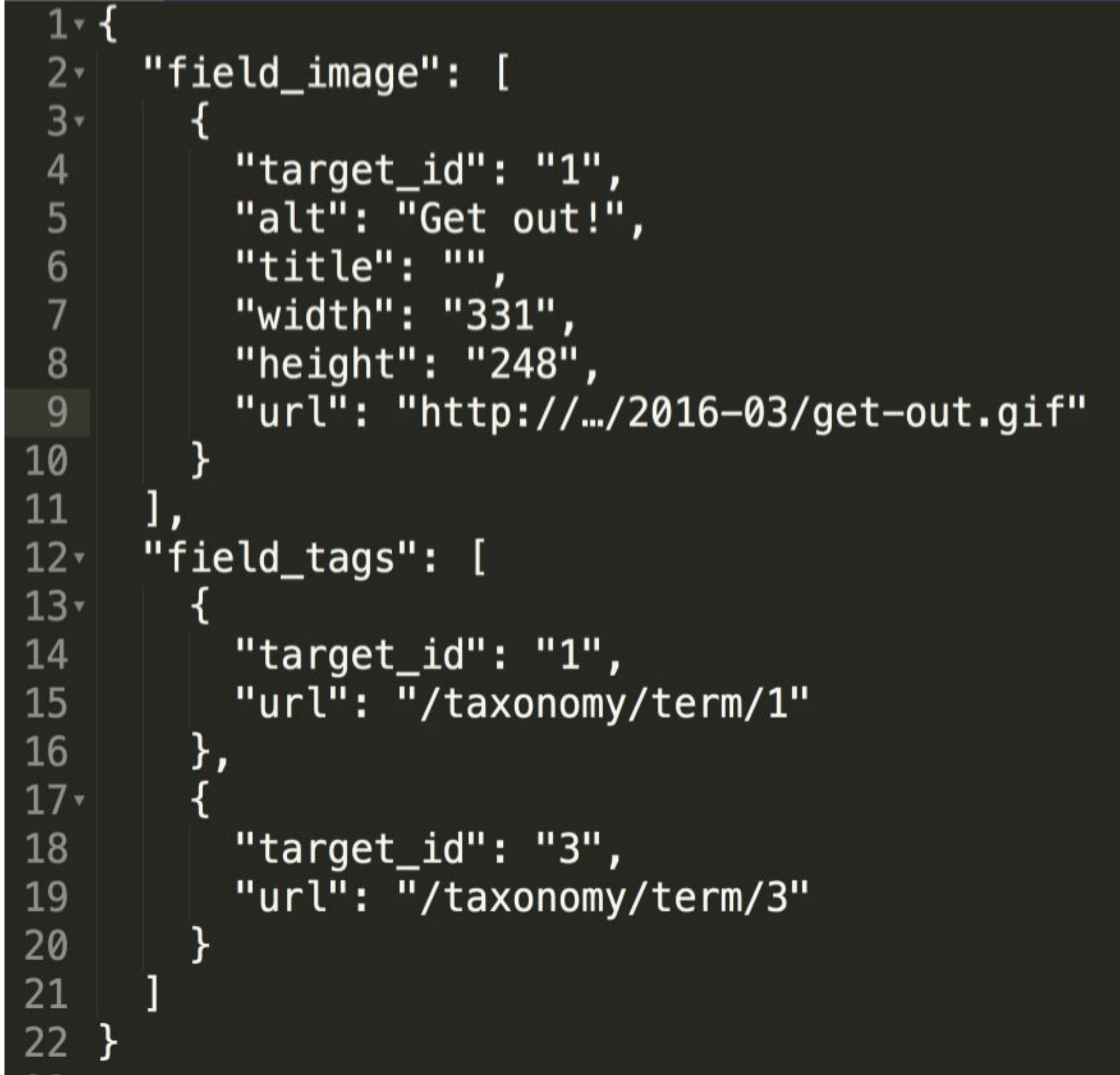


```
{
  "type": "article",
  "data": [
    {"name": "nid", "value": "1"},
    {"name": "uuid", "value": "4ae…9400"},
    {"name": "langcode", "value": "en"},
    {"name": "title", "value": "Test"},
    {"name": "created", "value": "2016-12-03…"},
    {"name": "changed", "value": "2016-12-03…"},
    {"name": "body", "value": "test\r\n"},
  ]
}
```

(another) Custom response

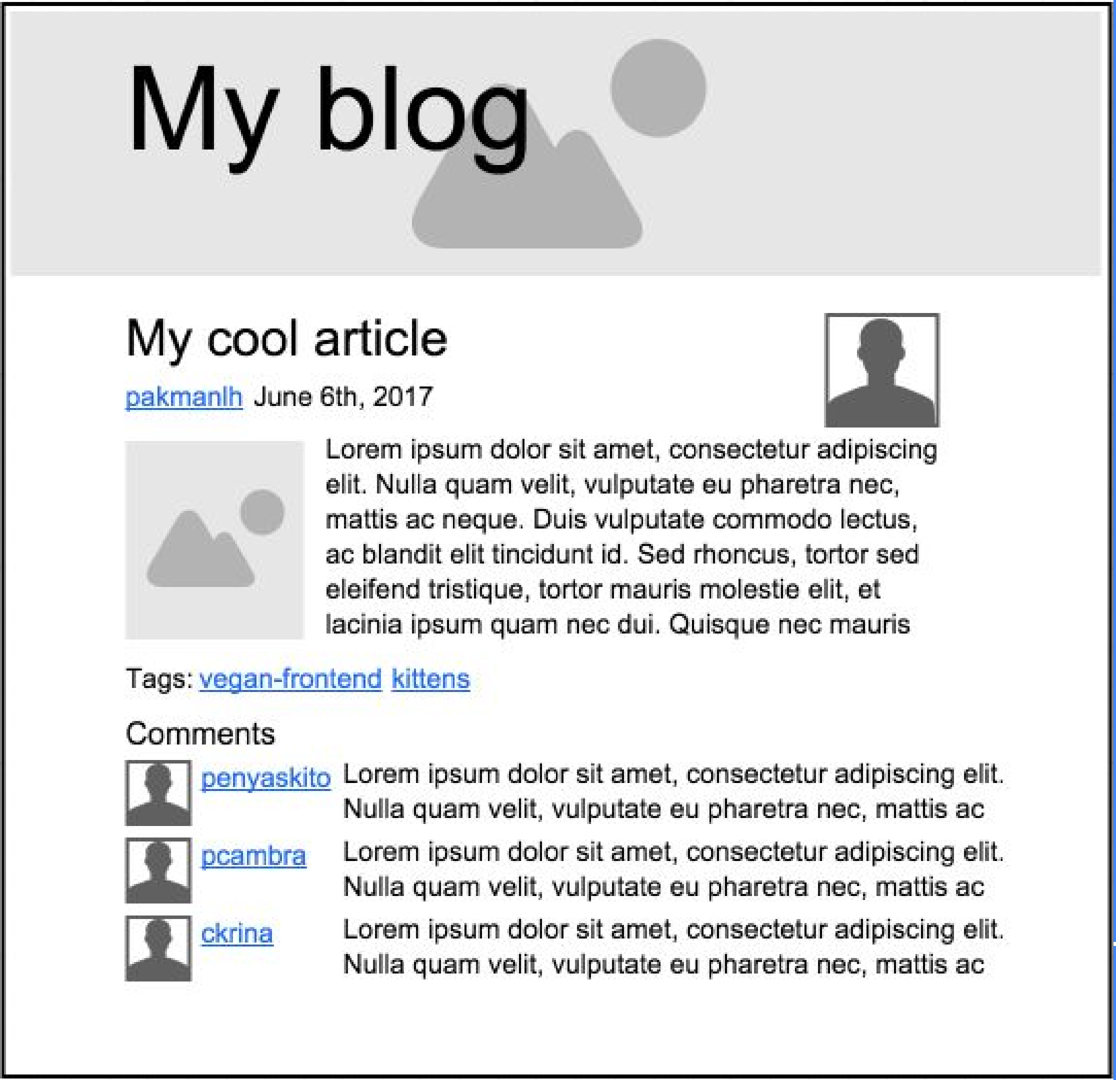
Relationships

From one resource to another



```
{
  "field_image": [
    {
      "target_id": "1",
      "alt": "Get out!",
      "title": "",
      "width": "331",
      "height": "248",
      "url": "http://…/2016-03/get-out.gif"
    }
  ],
  "field_tags": [
    {
      "target_id": "1",
      "url": "/taxonomy/term/1"
    },
    {
      "target_id": "3",
      "url": "/taxonomy/term/3"
    }
  ]
}
```

Round trips



My blog

My cool article

pakmanlh June 6th, 2017

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris

Tags: vegan-frontend kittens

Comments

penyaskito Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac

pcambra Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac

ckrina Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac

Extremely simple example

Your site is way more complex than that.

Problem 2: Round trips

- Trip **1**: GET articles/12
    - Trip **2**: GET articles/12 => tags/34
    - Trip **3**: GET articles/12 => tags/88
    - Trip **4**: GET articles/12 => users/88
        - Trip **5**: GET articles/12 => users/88 => images/5
    - Trip **6**: GET articles/12/comments
    - Trip **7**: GET articles/12 => comment/2
        - Trip **8**: GET articles/12 => comment/2 => user/8
            - Trip **9**: GET articles/12 => comment/2 => user/8 => image/9
    - Trip **10**: GET articles/12 => comment/7 [...]

Problem 3: Versioning

Your API keeps moving, your released versions are stable.

- **Golden rule**: do not change the API. Add a new version.
- Support different consumers
- Ease the deploy process
- Consumer locks to a specific version

Discovering content

— *"What are the available articles?"*
— *"Give me the seasons associated to show 23"*
— *"List all the users younger than 23"*

- Do not memorize UUIDs
- Find content by its attributes
- Make lists of content
- Similar to *\Drupal::entityQuery()*

Problem 4: Listing / finding

- We need a way to query the API for information.
- Potentially any field that the API outputs should be used as a filter.
    - *Return a list of animals having the feed field “herbivore”.*
- We need to support nested filtering. Things like:
    - *Give me all the articles written by authors having this role.*
    - *Query for bands with at least one band member from Mallorca.*
- Filtering using operators:
    - Young authors: **`filter[author.age][value]=35&filter[author.age][operator]=<`**
    - In-style glasses: **`filter[year][value][]=1980-01-01&filter[year][value][]=1989&filter[year][operator][]=BETWEEN`**

A query language for the URL

2MB

Is not a good response size for a JSON object

Problem 5: Response bloat

The response is too big!

- Big JSON (lots of kB)
- Long transport time
- Heavy to parse in consumer
- Browser memory

Problem 6: Discoverability

"I enjoy writing documentation for HTTP APIs, but above all I love keeping the documentation up to date when the project requires changes."

— No one ever

(not even at a DrupalCamp Spain party)

Enough with the problems!

No! There are more issues!

Decoupling is not easy

- Image styles
- Presentation logic in backend
- Data source aggregation
- Content type negotiation
- Aggregation queries
- …

The solutions!

(aka consumer dictatorship)

1. API design

Solution to the API design problem.
This is a Drupal-only problem

Use the Drupal 8 guts

Use a combination of configuration and custom Normalizers.

2. Round trips

We have too many round trips. We want a single response with all the information!

Allow the consumer to request relationships to be embedded in the original response.

Ex: My list of includes for articles/12

- Whatever the **tags** are
- Whatever the **author** is
- ...

3. Versioning

Don’t change the structure I’m relying on!

Just create different versions. The consumer uses the one they need.

Ex: all these may return differently

- api.example.org/v1.7/articles/12
- api.example.org/v1.6/articles/12
- ...

4. Listing / finding

The query language for the URL

Define a query language & use Drupal’s \Drupal::entityQuery()

The query language is used for the consumer to ask what they want to get.

The back end interprets that and uses entityQuery to find the results.

5. Response bloat

Do not return everything! Use sparse fieldsets.

Use the defined query language to select the fields you want.

Ex: Only get these fields for the article

```
…&fields=title,body,
author.name,tags.label,
tags.link,comments.author.
image
```

6. Discovery

Do not write docs, get them auto generated. Introspect your data.

You have described your data in Drupal already. Use it!

- Field definitions contain a lot of information.
- Config entities have a YAML schema.
- Typed data is a good data introspection tool.

That sounds like a lot of work

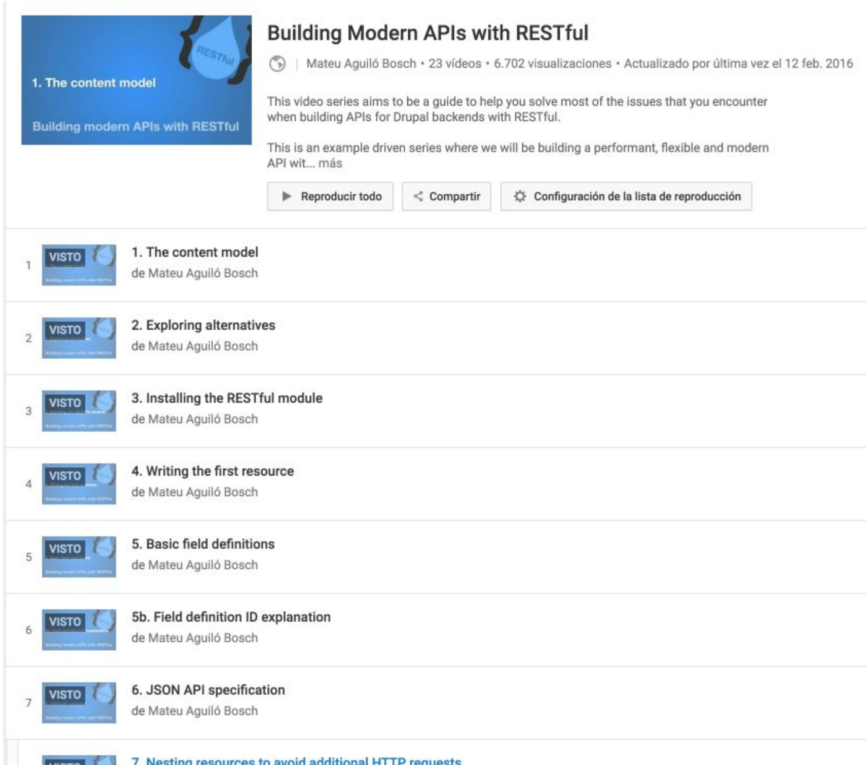
(contrib is still catching up)

You can use the **`RESTful`** module with `7.x-2.x` **today** in Drupal 7.

Watch the video tutorials, and solve these problems.

D8 version in the works.

… if you need it NOW (23/04/2016)



Building Modern APIs with RESTful

Mateu Aguiló Bosch • 23 vídeos • 6.702 visualizaciones • Actualizado por última vez el 12 feb. 2016

This video series aims to be a guide to help you solve most of the issues that you encounter when building APIs for Drupal backends with RESTful.

This is an example driven series where we will be building a performant, flexible and modern API wit... más

1. The content model
Building modern APIs with RESTful
RESTful
Reproducir todo
Compartir
Configuración de la lista de reproducción

1 VISTO 1. The content model
de Mateu Aguiló Bosch

2 VISTO 2. Exploring alternatives
de Mateu Aguiló Bosch

3 VISTO 3. Installing the RESTful module
de Mateu Aguiló Bosch

4 VISTO 4. Writing the first resource
de Mateu Aguiló Bosch

5 VISTO 5. Basic field definitions
de Mateu Aguiló Bosch

6 VISTO 5b. Field definition ID explanation
de Mateu Aguiló Bosch

7 VISTO 6. JSON API specification
de Mateu Aguiló Bosch

7. Nesting resources to avoid additional HTTP requests

Examples!

JSON API (jsonapi.org)

- Sparse fieldsets
- Includes
- Solves circular dependencies
- Easy to read
- Format for errors
- Extensibility
- One of the few stable standards

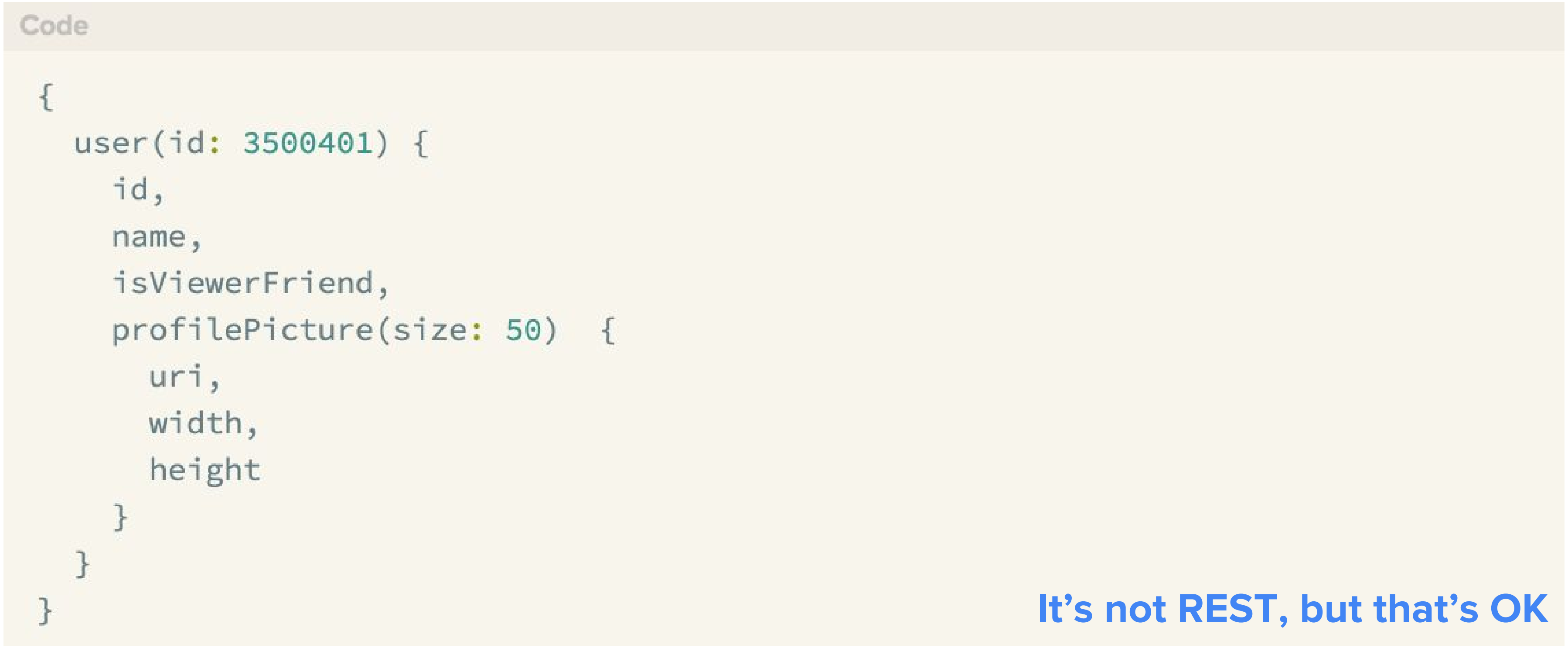
- It doesn’t have an opinion about: versioning and discovery

```
HTTP/1.1 200 OK
Content-Type: application/vnd.api+json

{
  "data": [{
    "type": "articles",
    "id": "1",
    "attributes": {
      "title": "JSON API paints my bikeshed!",
      "body": "The shortest article. Ever."
    },
    "relationships": {
      "author": {
        "data": {"id": "42", "type": "people"}
      }
    }
  }],
  "included": [
    {
      "type": "people",
      "id": "42",
      "attributes": {
        "name": "John"
      }
    }
  ]
}
```

```
GET /articles?include=author&fields[articles]=title,body,author&fields[people]=name
HTTP/1.1
```

GraphQL request


Code

```
{
  user(id: 3500401) {
    id,
    name,
    isViewerFriend,
    profilePicture(size: 50)  {
      uri,
      width,
      height
    }
  }
}
```

It’s not REST, but that’s OK

GraphQL response

Code

```
{
  "user" : {
    "id": 3500401,
    "name": "Jing Chen",
    "isViewerFriend": true,
    "profilePicture": {
      "uri": "http://someurl.cdn/pic.jpg",
      "width": 50,
      "height": 50
    }
  }
}
```

There are many more!

- Siren
- Collection+JSON
- Uber
- HAL
- JSON-LD (+ Hydra)
- Mason

- Falcor