INFINUM

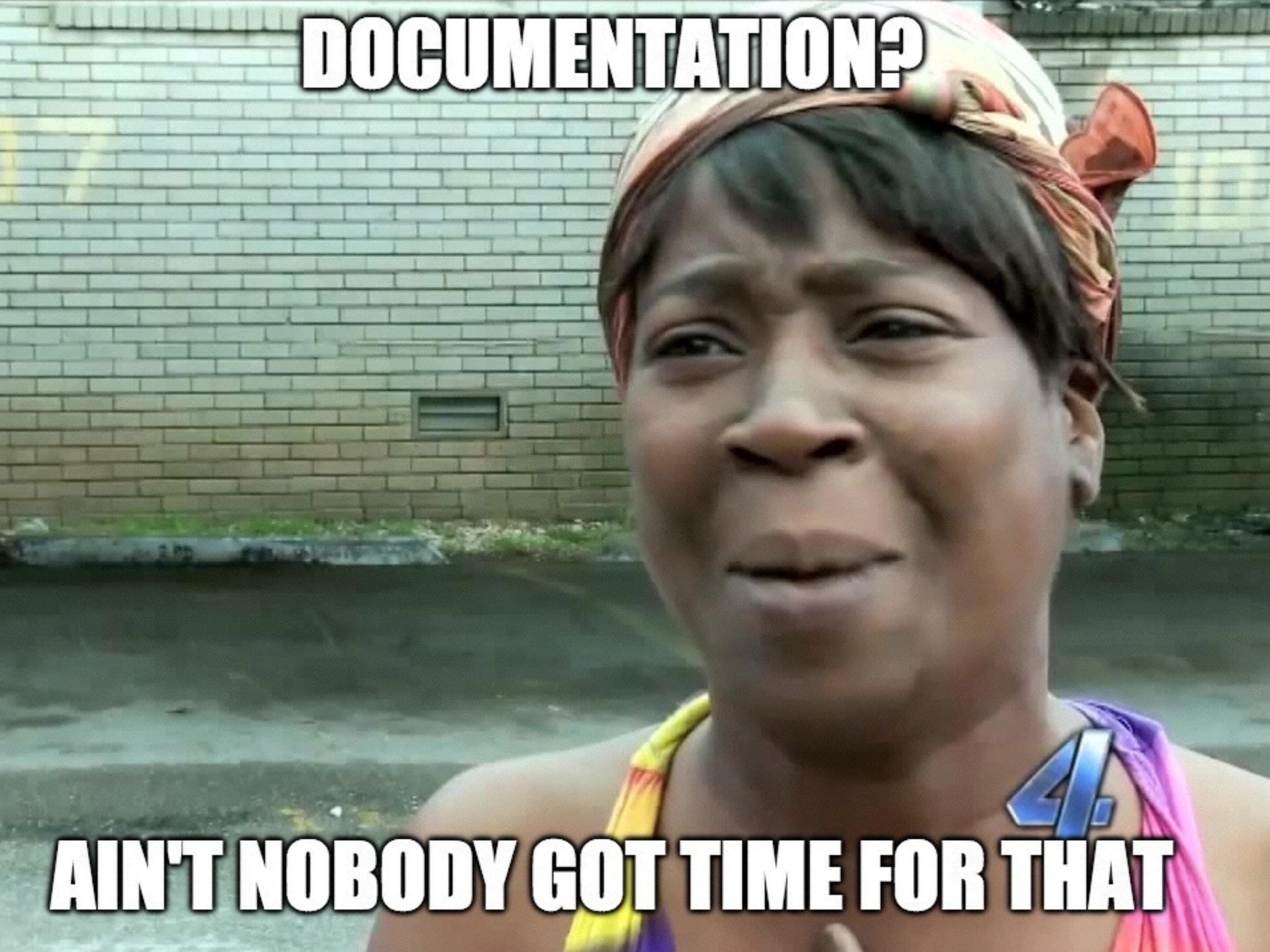
Dox

MELITA KOKOT
MELITA.KOKOT@INFINUM.HR, @MELITAKOKOT

ONE DOES NOT SIMPLY

WRITE AN API WITHOUT DOCUMENTATION

Maintaining docs is a pain.


DOCUMENTATION?
AIN'T NOBODY GOT TIME FOR THAT

NOT SURE IF API IS WRONG

OR IF DOCUMENTATION IS WRONG

WHAT IF WE AUTOMATE THIS?

We already have this ...

```
RSpec.describe 'Pokemons', type: :request do

  let(:pikachu) { create(:pokemon) }

  describe 'GET /pokemons/:id' do
    it 'gets a pokemon’ do
      get pokemon_path(pikachu), as: :json
      expect(response).to have_http_status(200)
    end
  end
end
```

... and with a bit more info ...

test

```ruby
RSpec.describe 'Pokemons', type: :request do
  include Documentation::Pokemons::Api

  let(:pikachu) { create(:pokemon) }

  describe 'GET /pokemons/:id' do
    include Documentation::Pokemons::Show

    it 'gets a pokemon' do
      get pokemon_path(pikachu), as: :json
      expect(response).to have_http_status(200)
    end
  end
end
```

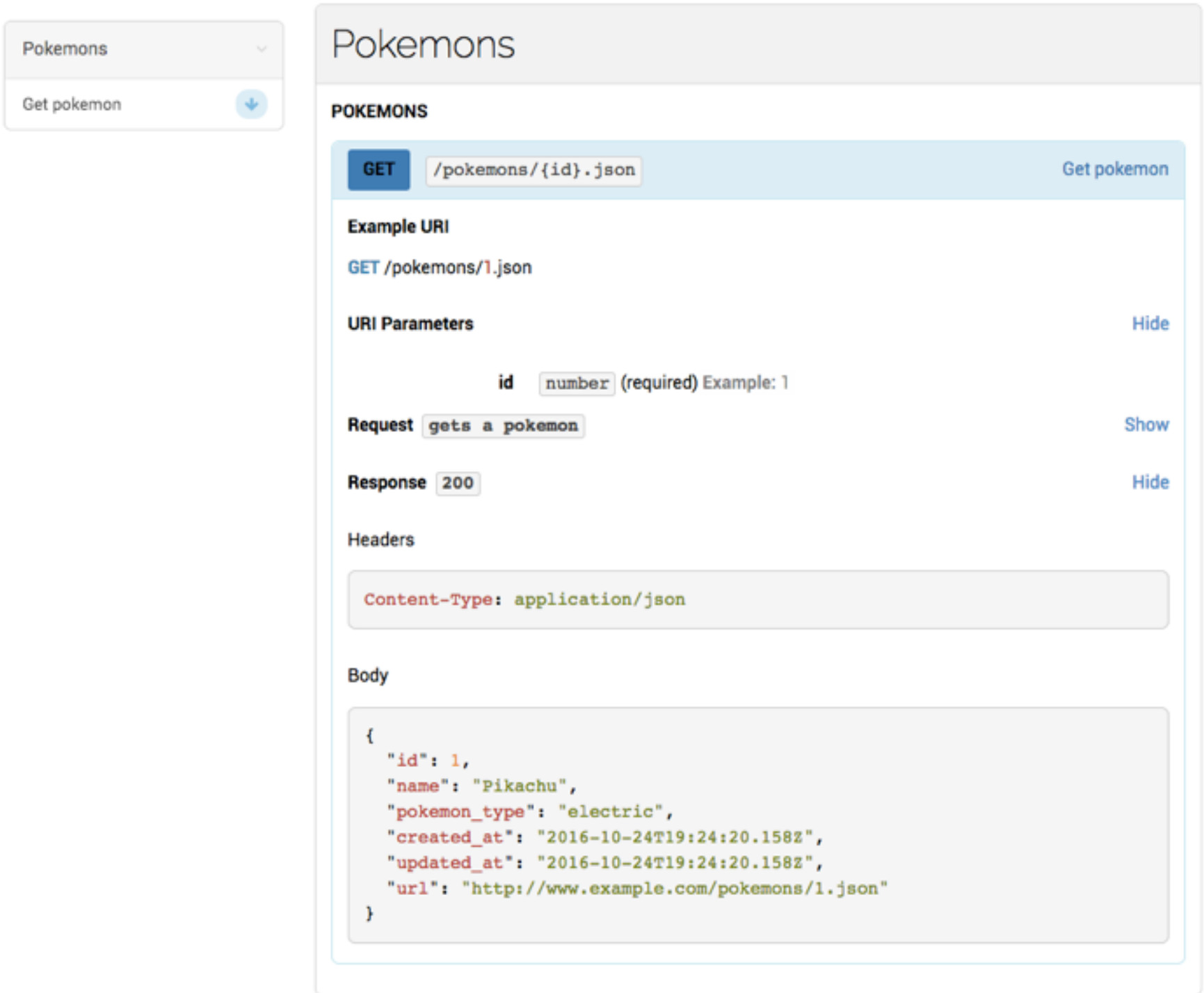
doc descriptor

```ruby
module Documentation
  module Pokemons
    extend Dox::DSL::Syntax

    document :api do
      resource 'Pokemons' do
        endpoint '/pokemons'
        group 'Pokemons'
      end
    end

    document :show do
      action 'Get pokemon'
    end
  end
end
```

... and one command ...

```
$ generate_documentation
```

... we could get something like this!



DOX

API docs generator for rails + rspec

TOP DEFINITION

dox

Personal information about people on the Internet, often including real name, known aliases, address, phone number, SSN, credit card number, etc.

"Someone dropped Bob's dox and the next day, ten pizzas and three tow trucks showed up at his house."

#lulz #owned #hacker #social engineering #ruin

Page not found.

It will be mine. Oh yes. It will be mine.

Back to RubyGems.org →

HOW DOES IT WORK?

Define descriptors using DSL

```ruby
module Documentation
  module Pokemons
    extend Dox::DSL::Syntax

    document :api do
      resource 'Pokemons' do
        endpoint '/pokemons'
        group 'Pokemons'
      end
    end

    document :show do
      action 'Get pokemon'
    end
  end
end
```

Documentation::Pokemons::Api

Documentation::Pokemons::Show

Include modules ...

```ruby
RSpec.describe 'Pokemons', type: :request do
  include Documentation::Pokemons::Api

  let(:pikachu) { create(:pokemon) }

  describe 'GET /pokemons/:id' do
    include Documentation::Pokemons::Show

    it 'gets a pokemon' do
      get pokemon_path(pikachu), as: :json
      expect(response).to have_http_status(200)
    end
  end
end
```

... which generate some meta data

```ruby
RSpec.describe 'Pokemons', type: :request, :apidoc, resource_name: 'Pokemons',
  resource_group: 'Pokemons', resource_endpoint: '/pokemons' do

  let(:pikachu) { create(:pokemon) }

  describe 'GET /pokemons/:id', action_name: 'Get Pokemon' do
    it 'gets a pokemon' do
      get pokemon_path(pikachu), as: :json
      expect(response).to have_http_status(200)
    end
  end
end
```

Run tests with DOX formatter

```
$ rspec spec/controllers --tag apidoc -f Dox::Formatter --out spec/
apispec.md
```

API BLUEPRINT

A powerful high-level API description language for web APIs.

https://apiblueprint.org/

Apispec.md

```
<!-- include(/Users/mel/dev/dox-demo/spec/documentation/api.md) -->

# Group Pokemons

## Pokemons [/pokemons]

### Get pokemon [GET /pokemons/{id}.json]

+ Parameters
    + id: `1` (number, required)

+ Request gets a pokemon (application/json)
+ Response 200 (application/json)

        {
          "id": 1,
          "name": "Pikachu",
          "pokemon_type": "electric",
          "created_at": "2016-10-24T19:24:20.158Z",
          "updated_at": "2016-10-24T19:24:20.158Z",
          "url": "http://www.example.com/pokemons/1.json"
        }
```

Render HTML with Aglio

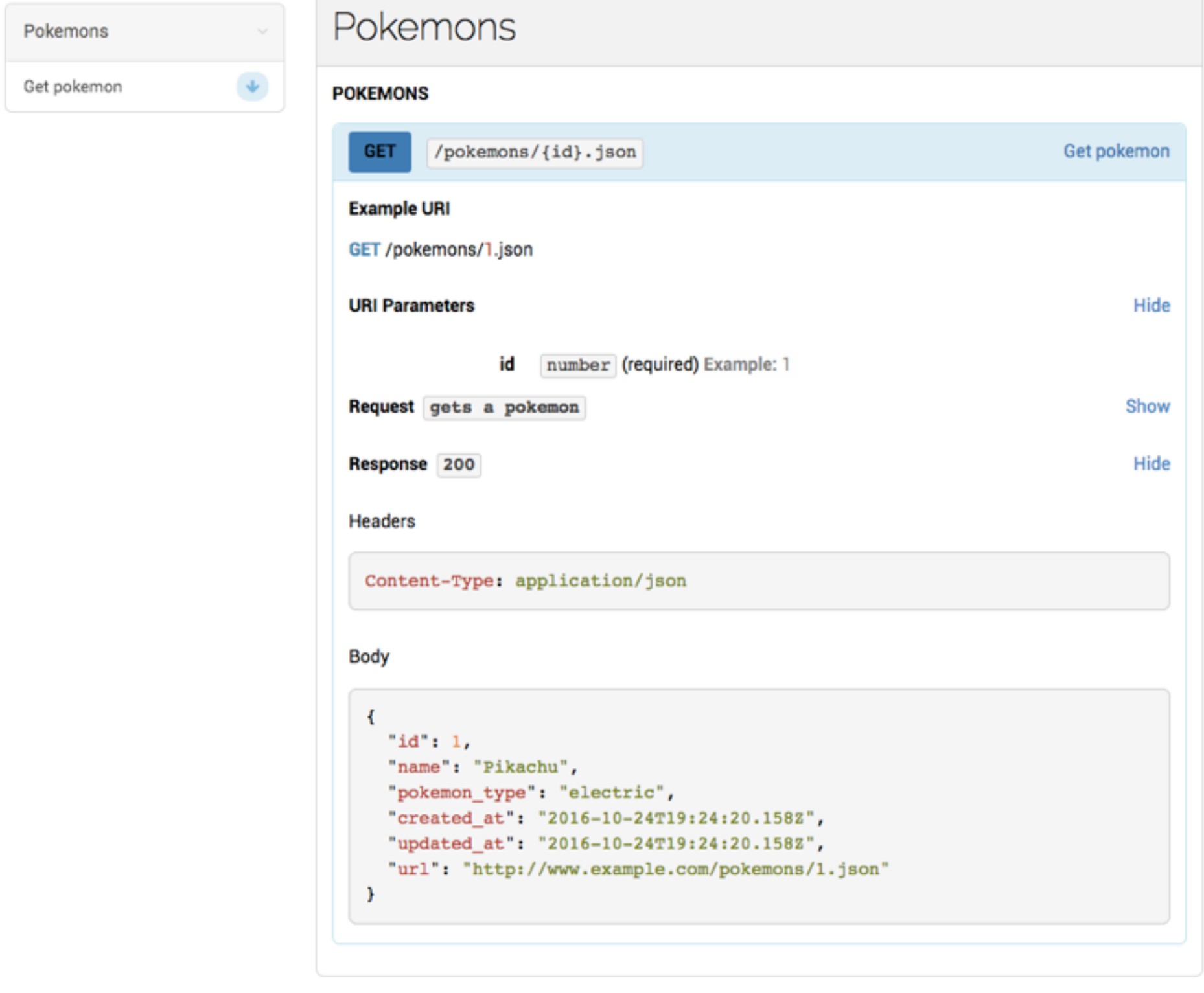
```
$ aglio -i spec/apispec.md -o public/api/docs/index.html
```

AGLIO

An API Blueprint renderer that supports multiple themes and outputs static HTML

https://github.com/danielgtaylor/aglio

And that's it!


Pokemons
Get pokemon
Pokemons
POKEMONS
GET
/pokemons/{id}.json
Get pokemon
Example URI
GET /pokemons/1.json
URI Parameters
Hide
id
number
(required) Example: 1
Request
gets a pokemon
Show
Response
200
Hide
Headers
Content-Type: application/json
Body
{
"id": 1,
"name": "Pikachu",
"pokemon_type": "electric",
"created_at": "2016-10-24T19:24:20.158Z",
"updated_at": "2016-10-24T19:24:20.158Z",
"url": "http://www.example.com/pokemons/1.json"
}

ADVANCED EXAMPLES

Add query params to URI template

```
index_params = { query: { type: :string, required: :optional, value: 'pika' } }

 document :index do
   action 'Get pokemons' do
     path '/pokemons?query={query}'
     params index_params
   end
 end
```

Add enumerations description

```
document :api do
   resource 'Pokemons' do
     endpoint '/pokemons'
     group 'Pokemons'
     desc 'pokemons_desc.md'
   end
 end
```

```
### Enums

**pokemon_type:**
- electric
- fire
- water
```

Pokemons ¶

POKEMONS

ENUMS

pokemon_type:

- electric
- fire
- water

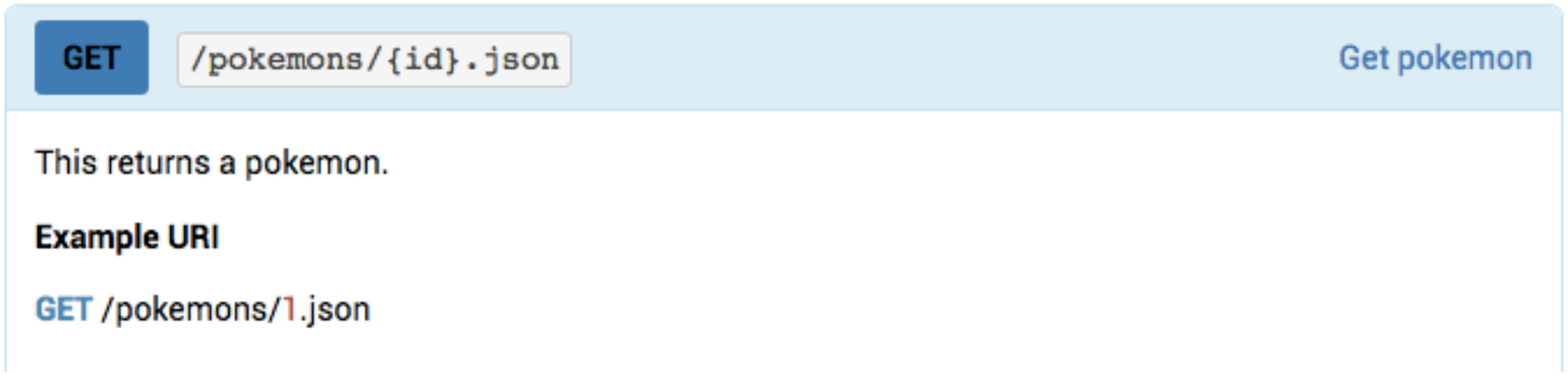
GET `/pokemons/{id}.json` Get pokemon

Example URI

GET /pokemons/1.json

Add action desc

```
document :show do
   action 'Get pokemon' do
     desc 'This returns a pokemon.'
   end
 end
```



Override everything

```
show_params = { id: { type: :integer, required: :required, value: 144, desc: 'pokemon id' } }

 document :show do
   action 'Get pokemon' do
     desc 'This returns a pokemon.'
     path '/pokemons/{id}'
     verb 'GET'
     params show_params
   end
end
```

Skip some examples with :nodoc

```ruby
describe 'GET /pokemons/:id' do
  include Documentation::Pokemons::Show

  it 'gets a pokemon' do
    get pokemon_path(pikachu), as: :json
    expect(response).to have_http_status(200)
  end

  it 'returns not found', :nodoc do
    get pokemon_path(123), as: :json
    expect(response).to have_http_status(404)
  end
end
```

Otherwise ...

GET `/pokemons/{id}` Get pokemon

This returns a pokemon.

Example URI

GET /pokemons/144

URI Parameters Hide

id `integer` (required) Example: 144

Request `gets a pokemon` Show

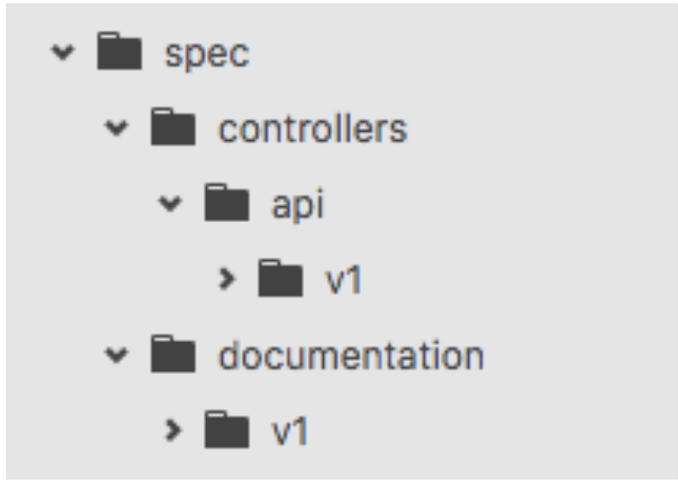
Response `200` Show

Request `returns not found` Show

Response `404` Show

API versioning

- use namespaces for tests and descriptors
- custom shell script



Custom shell script

```
#!/usr/bin/env bash

bundle exec rspec spec/controllers/v1 --tag apidoc -f Dox::Formatter --order defined --out
public/api/docs/v1/apispec.md

aglio --include-path / -i public/api/docs/v1/apispec.md -o public/api/docs/v1/index.html
```

CONFIGURATION

Configuration options

```
Dox.configure do |config|
  config.header_file_path = Rails.root.join('spec/documentation/api.md')
  config.desc_folder_path = Rails.root.join('spec/documentation/descriptions')
end
```

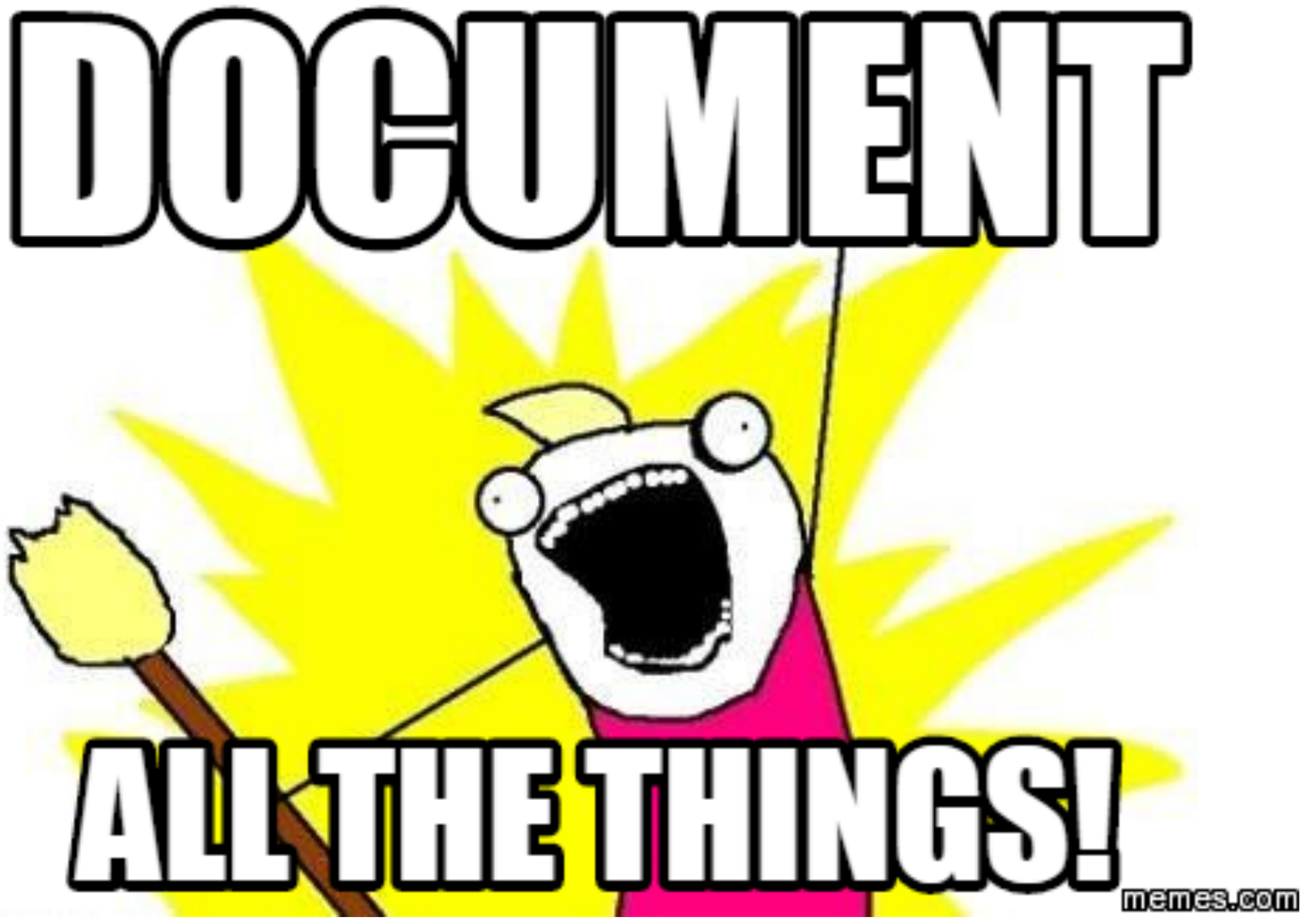
CONCLUSION

SIMPLE

- when the defaults are enough
- minimalistic documentation

HIGHLY EXTENSIBLE

- when you want more control
- rich documentation with custom descriptions


DOCUMENT
ALL THE THINGS!
memes.com

INFINUM

Questions? Ideas?

Visit infinum.co or find us on social networks:

infinum.co infinumco infinumco infinum