

Rest API 101

\$whoami

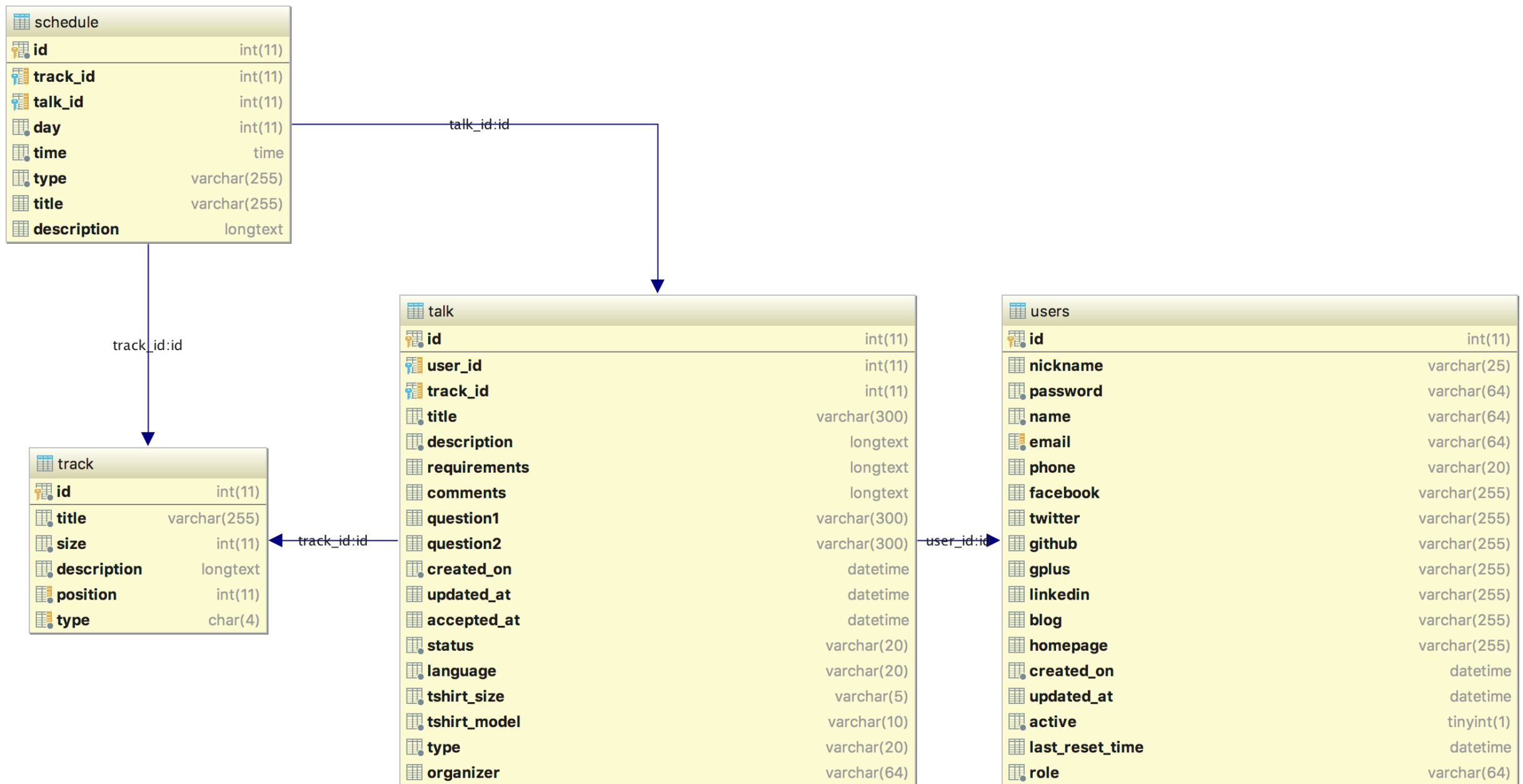
Arnas Petruškevičius
works @tesonet




Contents

- Planning Endpoints
- Requests
- Responses
- Relationships
- Errors

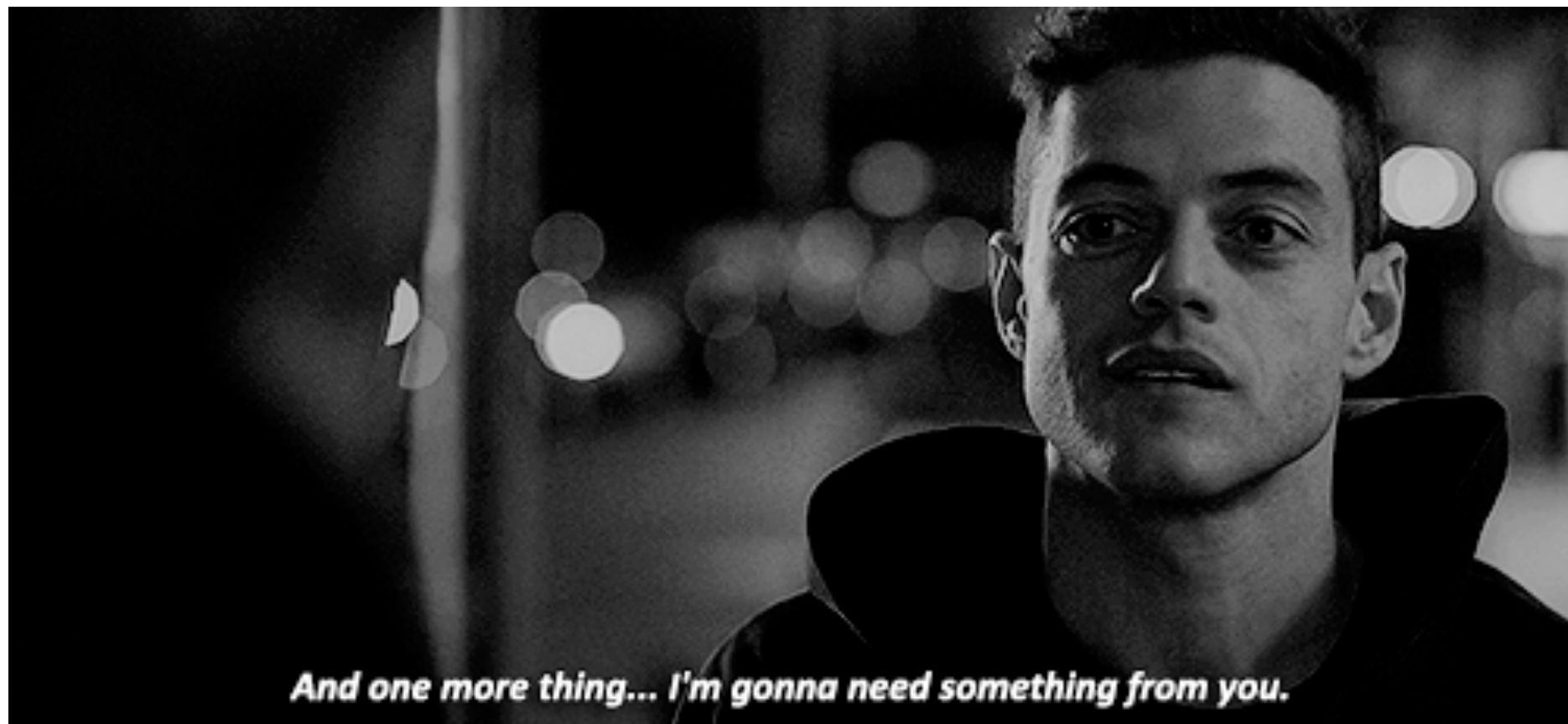
Example app: NTA API



Basic rules

- Plural vs singular: Plural
- Verb vs noun: Noun
- No content negotiation with file extensions
- Auto increment 

Requests



Pitfalls

- Don't define Content-Type
- Don't send POST with query_params
- PUT Vs. PATCH

Create

- POST /users

Read

- GET /users
- GET /users/1
- GET /users/1/talks

Update

- PUT /users/1
- PATCH /users/1

Delete

- DELETE /users
- DELETE /users/{X}
- DELETE /users/{X}/talks

Response

Pitfalls

- Hide X-Powered-By
- Return created object
- No 200 and error
- Don't Overuse 404, 400
- Don't straight cast from SQL

HTTP status codes

- 2xx -> success
- 3xx -> redirects
- 4xx -> client errors
- 5xx -> server errors

Content structure

No Right answer

Twitter style

```
// GET /users/1
{
  "id": 1,
  "nickname": "foobar",
  "email": "foo@bar.com"
}

// GET /users
[
  {
    "id": 1,
    "nickname": "foobar",
    "email": "foo@bar.com"
  },
  {
    "id": 2,
    "nickname": "barfoo",
    "email": "bar@foo.com"
  }
]
```

Pros

- Minimal

Cons

- Pagination, metadata -> `_(ツ)_/`

Facebook style

```
// GET /users/1
{
  "id": 1,
  "nickname": "foobar",
  "email": "foo@bar.com"
}

// GET /users
{
  "data": [{
    "id": 1,
    "nickname": "foobar",
    "email": "foo@bar.com"
  },
  {
    "id": 2,
    "nickname": "barfoo",
    "email": "bar@foo.com"
  }]
}
```

Pros

- Collections has space for pagination & metadata
- minimal response

Cons

- single items metadata only in item resource

JSON-API

Pros

- Standardised
- Well thought out

Cons

- Verbose

```
// GET /users/1
{
  "data": [{
    "type": "users",
    "id": "1",
    "attributes": {
      "name": "Arnas"
    },
    "links": {
      "self": "api.nta.lt/users/1"
    }
  }]
}
```

HATEOAS

unless you have hypermedia then it is not REST *Martin Fowler*

```
{  
  "id": 2,  
  //....  
  "_links": [{  
    "rel": "self",  
    "url": "localhost/users/2"  
  }]  
}
```

Relationships

N + 1 problem

GET /tracks/X/talks

5 tracks -> 6 requests

N + 1 problem

GET /tracks/1

Foreign Key arrays

```
{  
  "title": "Digital",  
  "size": 5,  
  "description": "description",  
  "position": 1,  
  "type": "TALK",  
  "_links": {  
    "talks": [1, 2, 3]  
  }  
}
```

GET /talks?id[]=1&id[]=2&id[]=3

N + 1 problem

GET /tracks

Compound Documents

```
{
  "data": [{
    "id": 1,
    "title": "Digital",
    //...
    "_links": {
      "users": [1, 2, 3]
    }
  },
  {
    "id": 2,
    "title": "Analog",
    //...
    "_links": {
      "users": [3]
    }
  }
],
  "_linked": {
    "users": [{
      "id": 1
    }, {
      "id": 2
    }, {
      "id": 3
    }
  ]
}
```

N + 1 problem

GET /tracks?include=users

Embedded Documents

```
{
  "data": [{
    "id": 1,
    "title": "Digital",
    //...
    "users": [{
      "id": 1
    }, {
      "id": 2
    }, {
      "id": 3
    }]
  }]
}
```


Errors



```
{
  "errors": [{
    "code": "225",
    "source": { "pointer": "/data/attributes/password" },
    "title": "Passwords must contain a letter, number, and punctuation
character.",
    "detail": "The password provided is missing a punctuation character."
  },
  {
    "code": "226",
    "source": { "pointer": "/data/attributes/password" },
    "title": "Password and password confirmation do not match."
  }
]
```

What I've missed

- Authentication
- Documentation
- Versioning
- Pagination

References

- Build APIs You Won't Hate
- Json-API