Creating Solid APIs

EuroPython 2018

Rivo Laks · 2018-07-27

Background

What is an API?

What is an API?

- *application programming interface*

What is an API?

- *application programming interface*
- *application **programmer** interface*

What is an API?

- *application programming interface*
- *application **programmer** interface*
- API is **user interface** for developers

What Makes an API Good?

What Makes an API Good?

- Documentation

What Makes an API Good?

- Documentation
- Familiarity

What Makes an API Good?

- Documentation
- Familiarity
- Lack of friction

Documentation

- Often overlooked

Documentation

- Often overlooked
- Gives the first impression

Documentation

- Often overlooked
- Gives the first impression
- The effort is worth it!

Creating Awesome Docs

What Should Go In There?

What Should Go In There?

- How do I access it?

What Should Go In There?

- How do I access it?
  - Do I need developer account?

What Should Go In There?

- How do I access it?
  - Do I need developer account?
  - Root URL, etc

What Should Go In There?

- How do I access it?
  - Do I need developer account?
  - Root URL, etc
- Authentication info

What Should Go In There?

- General encodings, formats, etc

What Should Go In There?

- General encodings, formats, etc
- Pagination, versioning, etc

What Should Go In There?

- General encodings, formats, etc
- Pagination, versioning, etc
- Common errors

What Should Go In There?

- General encodings, formats, etc
- Pagination, versioning, etc
- Common errors
- **Code for getting started**

The Endpoints

The Endpoints

- URL & operations

The Endpoints

- URL & operations
- Request/response data

The Endpoints

- URL & operations
- Request/response data
- Optional parameters

The Endpoints

- URL & operations
- Request/response data
- Optional parameters
- Permissions etc

Keep it Fresh!

Keep it Fresh!

- Obsolete docs are the worst

Keep it Fresh!

- Obsolete docs are the worst
- **Always autogenerate!**

Keep it Fresh!

- Obsolete docs are the worst
- **Always autogenerate!**
- Usually code » schema » docs

Schema & Autogeneration

Schema & Autogeneration

- OpenAPI, Swagger, etc

Schema & Autogeneration

- OpenAPI, Swagger, etc
- Use your tools

Schema & Autogeneration

- OpenAPI, Swagger, etc
- Use your tools
- Combine docs & code examples

Schema & Autogeneration

- OpenAPI, Swagger, etc
- Use your tools
- Combine docs & code examples
- Client libs autogeneration

Standardize!

JSON API

http://jsonapi.org/

one potential standard to use

JSON API

http://jsonapi.org/

one potential standard to use

GraphQL *is another option*

Standardize Structure

Standardize Structure

Responses have predictable, familiar structure

```
GET https://example.com/api/v1/projects
```

```
{
  "links": {
    "next": "https://example.com/api/v1/projects?cursor=cD0yMDE4L",
    "prev": null
  },
  "data": [...],
  "included": [...]
}
```

```
"data": [
  {
    "type": "project",
    "id": "289",
    "links": {
      "self": "https://example.com/api/v1/projects/289"
    },
    "attributes": {
      "created": "2018-06-28T22:52:08.690486Z",
      "name": "Allison-Patterson",
      "description": "aggregate collaborative models"
    },
    "relationships": {...}
  },
  ...
],
```

```
"data": [{ ...
    "relationships": {
      "created_by": {
        "data": {
          "type": "user",
          "id": "199"
        }
      },
      "epics": {
        "data": [
          {
            "type": "epic",
            "id": "3101"
          }
        ],
      }
    } }, ...
],
```

```
"included": [
  {
    "type": "epic",
    "id": "3101",
    "attributes": {
      "created": "2018-06-28T22:50:45.885691Z",
      "name": "Ergonomic background extranet"
    },
    "links": {
      "self": "https://example.com/api/v1/epics/3101"
    }
  },
  {
    "type": "user",
    "id": "199",
    "attributes": {...}
  }
]
```

Impressions?

Flexibility

Configurable fields:

```
GET https://example.com/api/v1/projects
GET https://example.com/api/v1/projects \
    ?included=comments
GET https://example.com/api/v1/projects \
    ?included=comments&fields[project]=name,comments
```

Pagination

List responses have `next` / `prev` links

```
{
  "links": {
    "next": "https://example.com/api/v1/projects?cursor=cD0yMDE4L",
    "prev": null
  },
  "data": [...],
}
```

Pagination

List responses have `next` / `prev` links

```
{
  "links": {
    "next": "https://example.com/api/v1/projects?cursor=cD0yMDE4L",
    "prev": null
  },
  "data": [...],
}
```

Cursor-based pagination FTW (but YMMV).

There is More ...

- Filtering
- Ordering

Errors

Errors

```
POST https://example.com/api/v1/projects
```

```
{
  "errors": [
    {
      "title":  "Invalid Attribute",
      "detail": "Name must contain at least three letters.",
      "source": { "pointer": "/data/attributes/name" },
      "status": "422"
    }
  ]
}
```

Special Cases

For when you have LOTS of data

Special Cases

For when you have LOTS of data

- *out-of-band* approach

```
GET https://example.com/api/v1/datasets/123
```


```
{
  "data": {
    "type": "dataset",
    "id": "123",
    "attributes": {
      "name": "CIFAR10 dataset",
    },
    "links": {
      "data_tgz": "https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz",
      "self": "https://example.com/api/v1/datasets/123"
    }
  }
}
```

Standardization Matters

- the specific standard isn't that important
- GraphQL, etc are also good options

Authentication & Authorization

Token Authentication

Clients send HTTP header, ala

```
Authorization: Token 9944b09199c62bcf9418
```

OAuth 2.0

OAuth 2.0

- For creating platforms

OAuth 2.0

- For creating platforms
- Complex, but solves many issues

OAuth 2.0

- For creating platforms
- Complex, but solves many issues
- Many packages, e.g. *Django OAuth Toolkit*, *OAuthLib*

Versioning

Versioning Schemes

Versioning Schemes

- `AcceptHeaderVersioning` *(DRF)*

```
GET /projects HTTP/1.0
Accept: application/json; version=1.0
```

Versioning Schemes

- `AcceptHeaderVersioning` *(DRF)*

```
GET /projects HTTP/1.0
Accept: application/json; version=1.0
```

- `URLPathVersioning` *(DRF)*

```
GET /v1/projects HTTP/1.0
Accept: application/json
```

Versioning Schemes Cont.

- Integers (`v1`) vs dates (`2018-07-27`)?

Versioning Schemes Cont.

- Integers (`v1`) vs dates (`2018-07-27`)?
  - Dates are less emotional

Versioning Schemes Cont.

- Integers (`v1`) vs dates (`2018-07-27`)?
  - Dates are less emotional
- Make upgrades easy

Version Transformers

Version Transformers

- » » Requests into **newer** version » »
  - Core code is for **latest** version
- « « Responses into **older** version « «

Version Transformers

- » » Requests into **newer** version » »
  - Core code is for **latest** version
- « « Responses into **older** version « «
- Won't work for big, breaking changes

Client's Perspective

The Scenario

- Let's try speech recognition...

The Scenario

- Let's try speech recognition...
- ... using AWS and GCP

Getting Started

- Documentation

Getting Started

- Documentation
  - Quite easy to find
  - A bit overwhelming

Getting Started

- Documentation
  - Quite easy to find
  - A bit overwhelming
- Code examples

Comprehensive Clients

- Install Python client
  - GCP: `google-cloud` package
  - AWS: `boto3` package

Comprehensive Clients

- Install Python client
  - GCP: `google-cloud` package
  - AWS: `boto3` package
- Authenticate

Comprehensive Clients

- Install Python client
  - GCP: `google-cloud` package
  - AWS: `boto3` package
- Authenticate
- Thorough docs

- Amazon

```
import boto3

client = boto3.client('transcribe')
response = client.start_transcription_job(...)
```

- Amazon

```
import boto3

client = boto3.client('transcribe')
response = client.start_transcription_job(...)
```

- Google

```
from google.cloud import speech

client = speech.SpeechClient()
results = client.recognize(...)
```

In Summary

- Invest in documentation
- Embrace standards *(e.g. JSON API)*
- Use automation
- Reduce friction

Thanks!

Rivo Laks · @rivolaks · rivolaks.com

tinyurl.com/ep18api