Designing APIs

SECT.09.2026

- How does a web application make use of APIs?
- What makes a good API?

Application Programming Interface

~~Application Programming~~ Interface

Typically either the interface between two programs or a program and a third-party library.

an API written in Django vs. Django's Database API

Web APIs

Two or more programs communicating via the web.

Server program running on "backend" – written in Python, Node.JS, Go, Ruby, etc.

Script <- Server

APIs called from command line programs/data pipelines/etc.

Often read-only API, used to make data available/queryable.

Native Client <–> Server

Often two programs co-created: Windows/MacOS/iOS/Android App interacting with cloud data: Spotify, Google Drive, Notion, etc.

Native application can provide some guarantees, persist some data locally, use device-specific features.

Web App <–> Server

Frontend program written in **client-side JavaScript**. Constrained by browser: HTTP, limited local storage options, no filesystem, etc.

Loading slide...

Loading slide...

Loading slide...

Types of Web APIs

Simple HTTP endpoints

RESTful

Remote Procedure Call (RPC)

GraphQL

Example API

`GET /users/{username}/`

`GET /timeline/`

`GET /search/?q=term`

`POST /posts/`

`POST /posts/{id}/likes/`

`POST /users/{username}/followers/`

`DELETE /posts/{id}/`

`DELETE /posts/{id}/likes/`

`DELETE /users/{username}/followers/`

Loading slide...

RESTful API Design

REpresentational **S**tate **T**ransfer

An *architectural pattern*: defines how software is structured.

Fielding (1999): https://roy.gbiv.com/pubs/dissertation/rest_arch_style.htm

Key Concepts

Resources (Nouns)

- User, Post, Favorite
- each has a URI: /post/123/ /user/carl/

HTTP Verbs

- GET, POST, PUT, DELETE, HEAD

Representation

- JSON, XML, etc.

No client-side state

- No cookies!

Loading slide...

Loading slide...

Verbs

Create: POST (or PUT)

Read: GET

Update: PUT (or POST)

Delete: DELETE

PUT vs POST

`PUT /users/tony/` vs. `POST /users/`

`GET /posts/`

`GET /posts/{id}/`

`POST /posts/`

`DELETE /posts/{id}/`

`GET /posts/{id}/likes/`

`PUT /posts/{id}/likes/`

`DELETE /posts/{id}/likes/`

Benefits

- Leverage HTTP ecosystem (caching, client libraries, ease of use)
- Representation-agnostic: eases integration
- Encourages careful design: avoid duplicate resources, separation of concerns

Drawbacks

- Statelessness comes with challenges.
- Complexity of handling nested resources, can lead to **performance** issues.

Loading slide...

Loading slide...

REST Alternative: RPCs

Remote Procedure Calls

```
GET /api/v3/?func=GetTopSongs(10)&format=json
```

```
GET /api/v3/?func=SearchSongsByArtist(artistName:"prince")&format=xml
```

Loading slide...

GraphQL

Ask API to fill out a graph data structure for you.

In practice this leads to a very tight coupling.

It is useful as an orchestration layer (API of APIs), but beware backwards compatibility!

Anecdata: Most teams I know that introduced a GraphQL API have since replaced it.

Practical Considerations

Error Handling

Versioning

Authentication

Clients & Documentation

Loading slide...

Loading slide...

Loading slide...

Minimal Surface Area

Only expose (& test) what you need.

Much easier to add than remove from all APIs, web or otherwise.

Do not ignore *extensibility* though, consider how & where new behavior will fit in.

API Authentication

Type	Example	Pro	Con
query string	`?apikey=123`	trivial to implement	exposed in logs
header	`X-ApiKey: 123`	easy in most clients	still exposed client side
JSON Web Token	Authorization: Bearer	client-side security possible*	complexity

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

*not guaranteed: bad implementations abound

Loading slide...

Loading slide...

API Documentation

Many tools (JSON API, Django REST Framework, FastAPI) can generate documentation for you from docstrings or equivalent.

Like other doc generation tools, can ensure your documentation stays up to date.

Enough uniformity and even client libraries can be auto-generated. (JSON API)

Examples

Stripe: https://docs.stripe.com/api/errors?lang=python

Open States (FastAPI): https://v3.openstates.org/docs