APIs

What are APIs?

- Application Programming Interface
- APIs are everywhere
- Contract for pieces of software to talk to each other
- Structured request and response

Analogy

Consuming APIs

- APIs can expose:
  - Data
    - Facebook
    - Google maps
  - Algorithms
    - Face detection (https://cloud.google.com/vision/docs/detecting-faces)
    - Geo-location
  - Infrastructure
    - Amazon Web services

What do (most) APIs look like

- The REST specification is the most widely used style.
- Developed by Roy Fielding in 2000.

REST or RESTFUL API

- Representational State Transfer
- Architecture style for designing networked applications
- Relies on a stateless client-server protocol (HTTP)
- Treats server objects (e.g. entry in database) as resources that can be created/destroyed
- Can be used by almost any programming language (http requests)

REST or RESTFUL API

- Most REST APIs deliver data in JSON format
  - Very widely used
  - Very simple grammar
  - Human readable
  - Concise
  - Lots of tools to produce/parse it

The REST specification

- Client-server: clients don’t care about storage, servers don’t care about look & feel.
- Stateless: servers don’t track state (e.g. progress) between requests.
- Cacheable: it must be clear what requests can be cached.
- Layered system: clients don’t care if they connect to root server or some intermediary.
- Code on demand:* servers can send executable code to clients.
- Uniform Interface: consistency in how resources are retrieved and manipulated.

REST Behaviour

- GET: read data from a specified resource
- POST: non-idempotent **create** or update (submit data to be processed)
- PUT: idempotent create or **update**
- DELETE: delete

- Other (rarely used):
- HEAD: Same as get, without returning body
- OPTIONS: Returns the supported HTTP methods
- PATCH: Update partial resources

POST vs PUT

- If you call PUT 100 times, you get one object
  - Can only use this when client knows full path to instance.
  - Can only use this when object is completely specified.
- If you call POST 100 times, you get 100 objects
  - Use this when path/id is automatically generated.
  - Can use this when object is partially specified.

REST responses

- Response codes are also already built into the HTTP standard
- 200 OK
- 201 Created
- 204 No Content

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Error

Code Examples

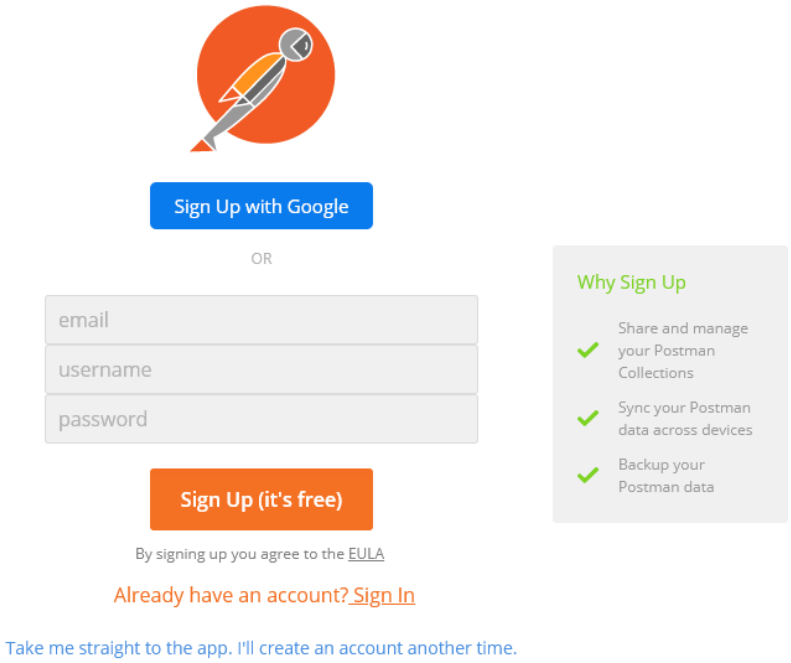
- https://bitbucket.org/positivecomputing/recipes.git

Authentication

- How does a user let your app access their Facebook data?
  - Without giving away their password?
  - Without giving away too many rights?
  - Without hackers impersonating app?
  - Without hackers impersonating user?
  - And be able to revoke access at any time?

- Many APIs require authentication (free or paid)

Authentication



OAuth

- Delegated authorization framework for REST/APIs.
  - Apps obtain limited access (scopes) to a user's data without giving away user's password.
  - Decouples authentication from authorization

- Analogy
  - If you have a hotel key card, you can get access to your room.
  - How do you get a hotel key card?
  - You have to do an authentication process at the front desk to get
  - it. After authenticating and obtaining the key card, you can access resources across the hotel.
- OAuth:
  - App requests authorization from User
  - User authorizes App and delivers proof
  - App presents proof of authorization to server to get a Token
  - Token is restricted to only access what the User authorized for the specific App

Example

- FacebookForum in the recipes repo
  - Extends SimpleForum, so authors are identified by Facebook accounts
  - Most extras are separated into FBPS (Facebook Permission System) for reuse