REST Web Service

* Web Service
  + A service is a black box piece of software you interact with
    - Black box – you do not know how it works
    - If I gave you API endpoints you cannot even tell me what language it written in
* Web Service vs Web Application
  + Web service is designed to consumed/used by other machines/services
    - Humans do not directly use a web service. Not Human friendly
  + Web Application is a completed application for a human to use
    - Nice completed website
* REST Basics
  + REST uses HTTP or HTTPS
    - The only protocols you can use
* HTTP Basics
  + HyperText Transfer protocol
  + Main protocol of the internet
  + Built on top of TCP
  + Built on requests and responses
    - Every Request gets a response
  + HTTP Request
    - Header
      * Meta Information
    - Body
      * Content
    - Verb
      * Get, put post, delete
    - URL (Request Line)
      * Where you are sending your request
    - HTTP version
      * What version HTTP you are using
  + HTTP Response
    - Header
    - Body
    - Status code
      * 100’s information
      * 200’s successes
      * 300’s redirects
      * 400’s Client side error
      * 500’s Server side error
* Idempotent
  + An operation that can be perform 1 to infinitely many times and you get the same result
  + Idempotent HTTP
    - GET
    - DELETE
    - PUT
      * POST is not idempotent
* RESTful Web Service
  + RESTful web service is an architectural style of building a web service
  + A RESTful web service adheres to these 6 constraints
    - Client-server architecture
      * The RESTful web service is decoupled from the front end
        + Your web service is front-end agnostic
        + Your API could be used to build 3 separate front-ends just fine
    - Stateless
      * There are no sessions in the backend
      * You cannot store session information on the backend
        + You can use things like API keys or JWTs for security
    - Cacheable
      * You can optionally cache information on the front-end or back-end
        + Imagine 30 different users keep querying for /associates

You can caches the results on the backend or front end so you do not have to go to the database each time

* + - Layered System
      * Your REST services can stack on top of each other
      * Your REST API can call another REST API that calls another REST API and that is all good.
    - Code On demand (optional)
      * Optionally a RESTful web service can return executable code
        + \*\*You get back a callback function from an endpoint which helps you process information your server
    - Uniform Interface
      * Unified REST conventions for making HTTP requests and endpoints
        + Identification of Resources through the uri

/associates

/associates/2

* + - * + Self-Descriptive messages

A single HTTP request has all the information needed to process it

You do not need to send follow HTTP requests for your initial request to be properly processed

* + - * + Manipulation of Resources through representations

You return a representation of the object not the object itself

JSON (99%)

XML (.9%)

Plain text(.1%)

* + - * + HATEOAS

HyperMedia as the Engine of the Application state

An HTTP response will also have a bunch of links to similar related resources

This is a way for someone consuming your API to find the enpoints

REST technically does not require documentation