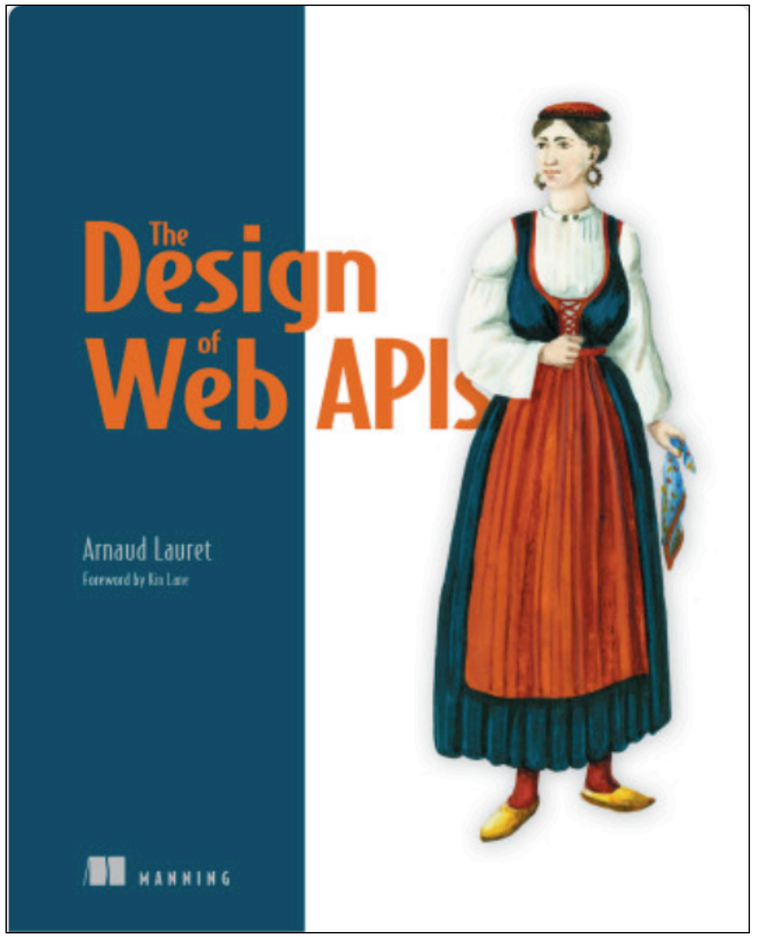
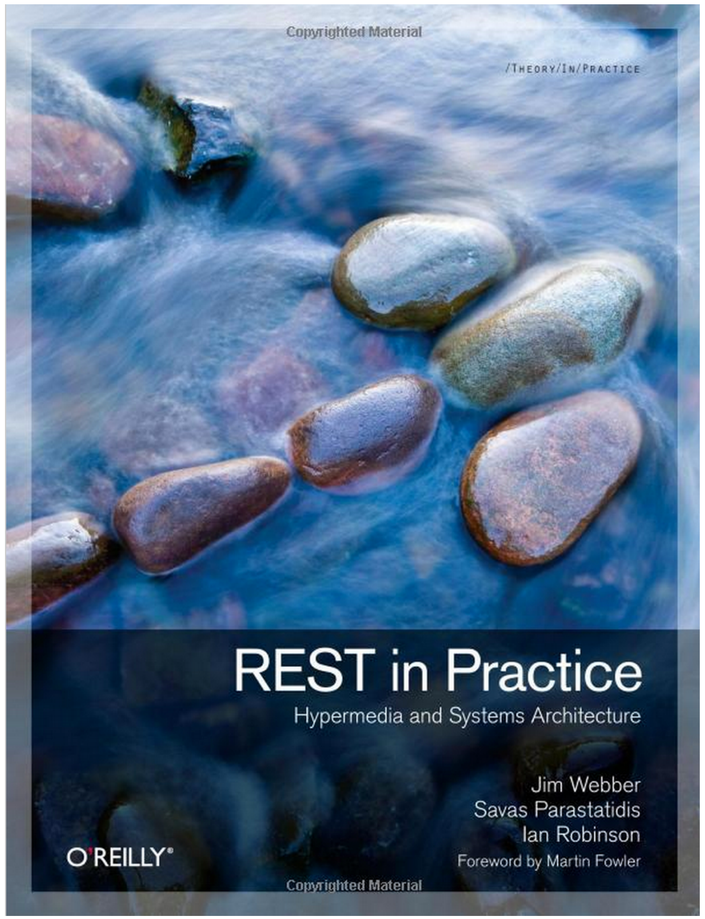
# APIs


Full Stack Web Development

# REST

**Representational State Transfer** (**REST**) is an architectural style that abstracts the architectural elements within a distributed hypermedia system.[1] REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.[2]REST has emerged as a predominant web API design model



# Examples - REST


- Twitter API
- Slack
- Github
- Foursquare
- IBM Watson
- Open Street Maps

- REST is an “Architectural Style” - enumerating an approach to building distributed systems.
- It embodies an approach that aims to maximize the infrastructure of http infrastructure deployed in the public internet, enabling secure, scalable distributed systems that do not require expensive, complex alternative infrastructure.


API list [...]

ADD API | Interviews | Blog | Sponsor Us | Login

A collective list of APIs. Build.

weather

Explore Apis

Free APIs
Cool APIs
Help Fix The Climate

API Categories

API
Analytics
Animals
Animation
Application Development
Applications
Astrology
Audio
Avatars
Big Data
Billing
Blogging
Bookmarks
Business
Charts
Chinese

By Last Updated

**Weatherbit**

Weather ... Read More

Weather

**Yahoo! Weather**

Weather ... Read More

Weather

**Climacell Micro Weather**

Historical, real-time and nowcast weather data ... Read More

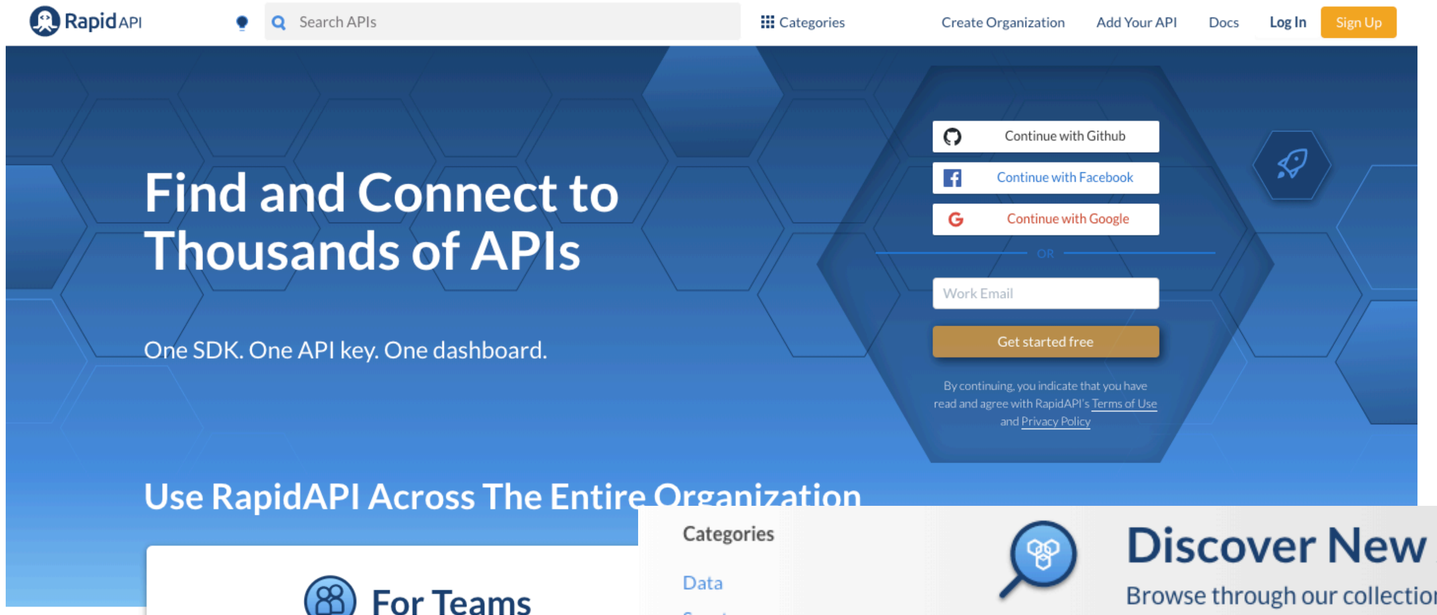
Weather

**Openweathermap**

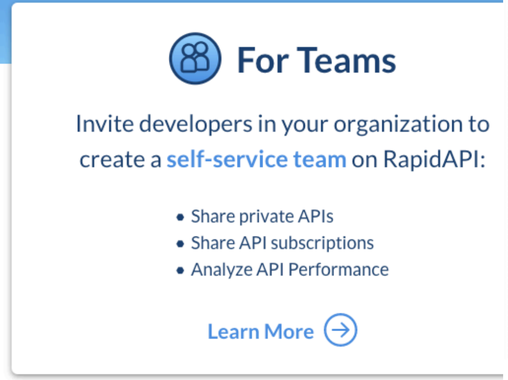
Weather ... Read More

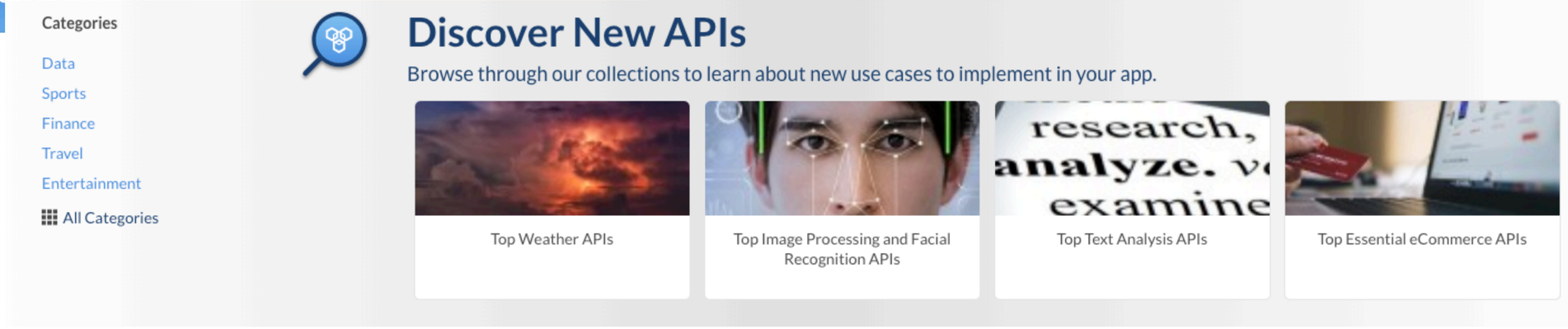
Weather

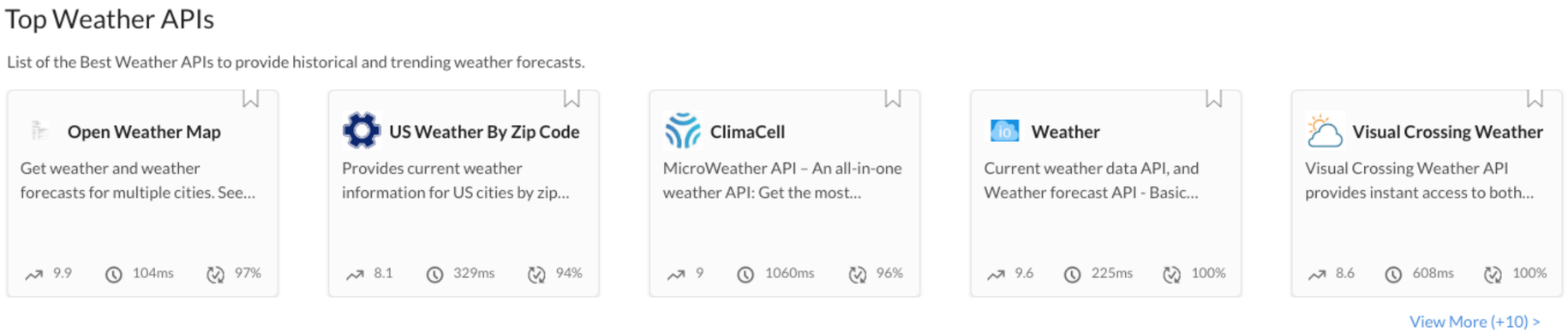
https://apilist.fun/




https://rapidapi.com/



https://blog.rapidapi.com/access-global-weather-data-with-these-weather-apis/

# Top 3 Free Weather APIs to Access Global Weather Data in 2019

December 27, 2017 By Janet Wagner 3 Comments

Table of Contents [show]

## TL;DR

1. AccuWeather
2. DarkSky
3. OpenWeatherMap
4. More Weather APIs

## Current weather and forecasts in your city

### Weather in Waterford, IE

Main | Daily | Hourly | Chart | Map

9 °C

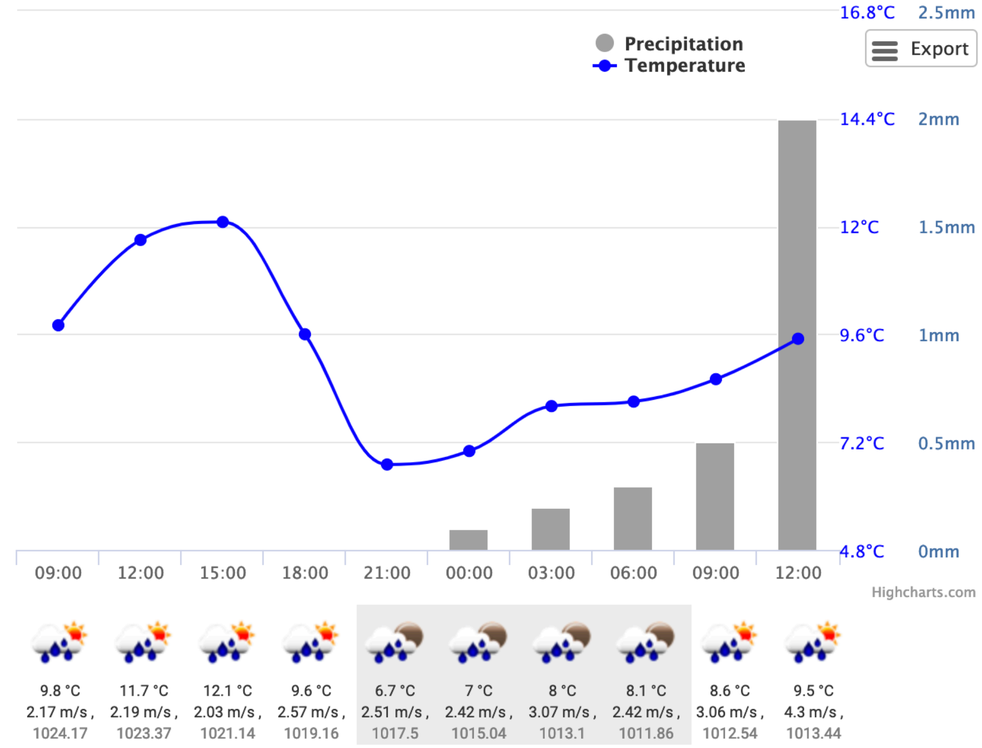
Broken clouds

08:53 Feb 27 Wrong data?

| | |
|---|---|
| Wind | Light breeze, 2.2 m/s, South-southwest ( 207 ) |
| Cloudiness | Broken clouds |
| Pressure | 1024 hpa |
| Humidity | 88 % |
| Rain | 0.045 mm |
| Sunrise | 07:18 |
| Sunset | 18:04 |
| Geo coords | [52.26, -7.11] |

The weather forecast is displayed in accordance with your local time. Please pay attention to it when you will watch the weather in another time zone.

### Weather and forecasts in Waterford, IE




https://openweathermap.org/

# Weather API



**Please sign up and use our fast and easy-to-work weather APIs for free. Look at our monthly subscriptions for more options than Free account can provide you. Read How to start first and enjoy using our powerful weather APIs.**

## Current weather data

API doc Subscribe

- Access current weather data for any location including over 200,000 cities
- Current weather is frequently updated based on global models and data from more than 40,000 weather stations
- Data is available in JSON, XML, or HTML format
- Available for Free and all other paid accounts

## 5 day / 3 hour forecast

API doc Subscribe

- 5 day forecast is available at any location or city
- 5 day forecast includes weather data every 3 hours
- Forecast is available in JSON and XML
- Available for Free and all other paid accounts

## 16 day / daily forecast

API doc Subscribe

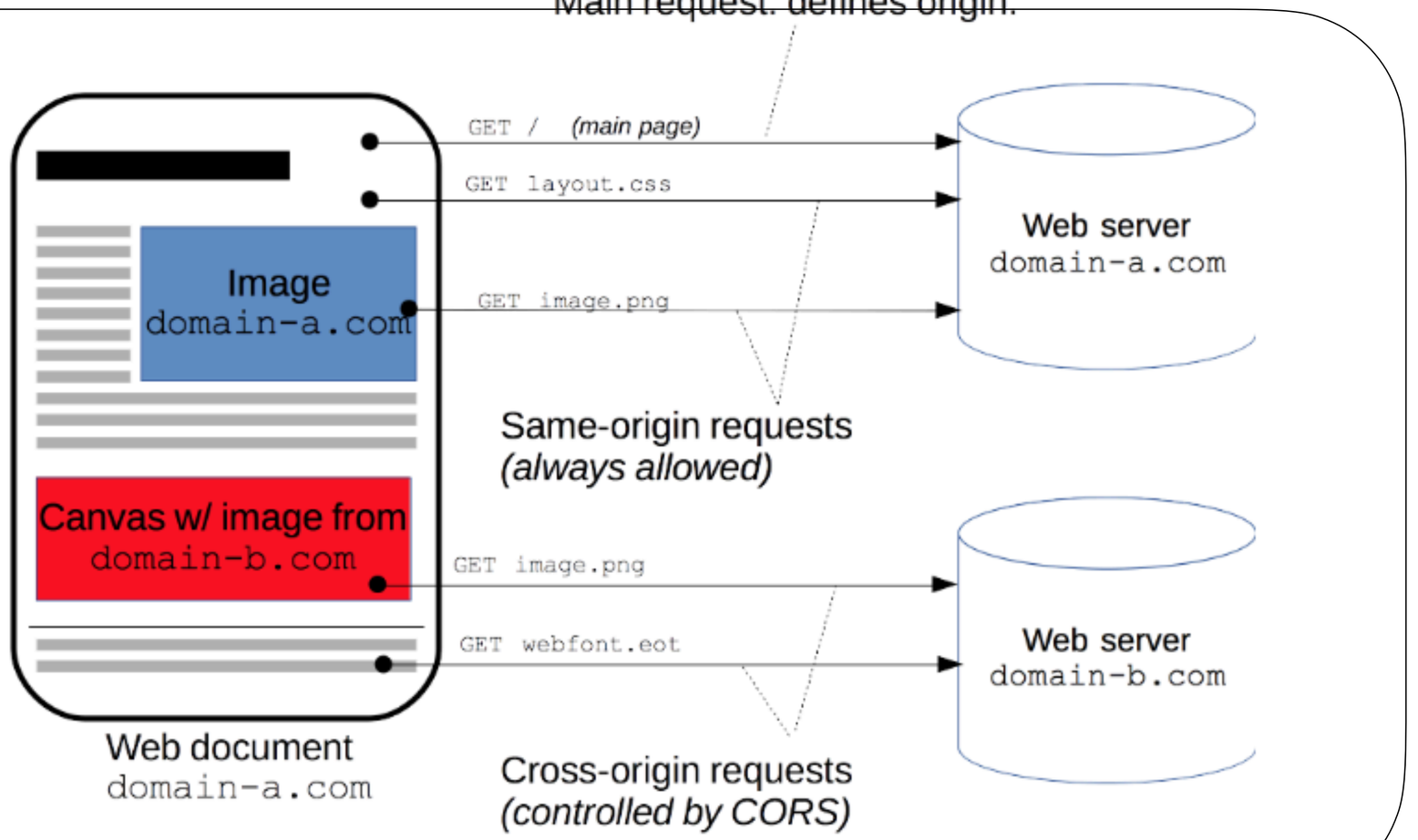
- 16 day forecast is available at any location or city
- 16 day forecast includes daily weather
- Forecast is available in JSON and XML
- Available for all paid accounts

## Weather

https://github.com/toddmotto/public-apis#weather

| API | Description | Auth | HTTPS | CORS |
|---|---|---|---|---|
| APIXU | Weather | `apiKey` | Yes | Unknown |
| ClimaCell Micro Weather | Historical, real-time and nowcast weather data | `apiKey` | Yes | Yes |
| Dark Sky | Weather | `apiKey` | Yes | No |
| MetaWeather | Weather | No | Yes | No |
| NOAA Climate Data | Weather and climate data | `apiKey` | Yes | Unknown |
| ODWeather | Weather and weather webcams | No | No | Unknown |
| OpenUV | Real-time UV Index Forecast | `apiKey` | Yes | Unknown |
| OpenWeatherMap | Weather | `apiKey` | No | Unknown |
| Storm Glass | Global marine weather from multiple sources | `apiKey` | Yes | Yes |
| Weatherbit | Weather | `apiKey` | Yes | Unknown |
| Yahoo! Weather | Weather | `apiKey` | Yes | Unknown |

An example of a cross-origin request: the front-end JavaScript code served from `https://domain-a.com` uses `XMLHttpRequest` to make a request for `https://domain-b.com/data.jso`




For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts.

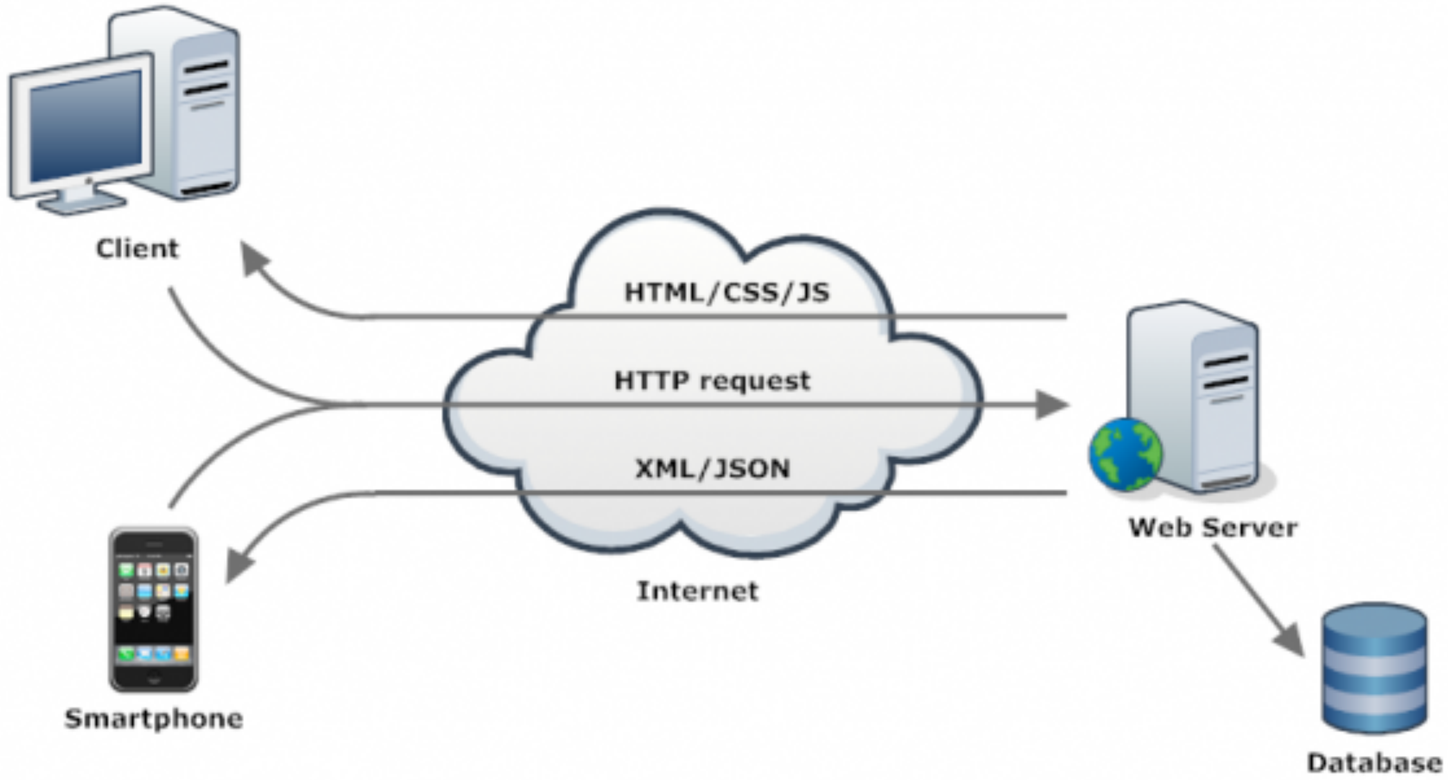
The CORS mechanism supports secure cross-origin requests and data transfers between browsers and servers.

## Cross-Origin Resource Sharing (CORS)

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

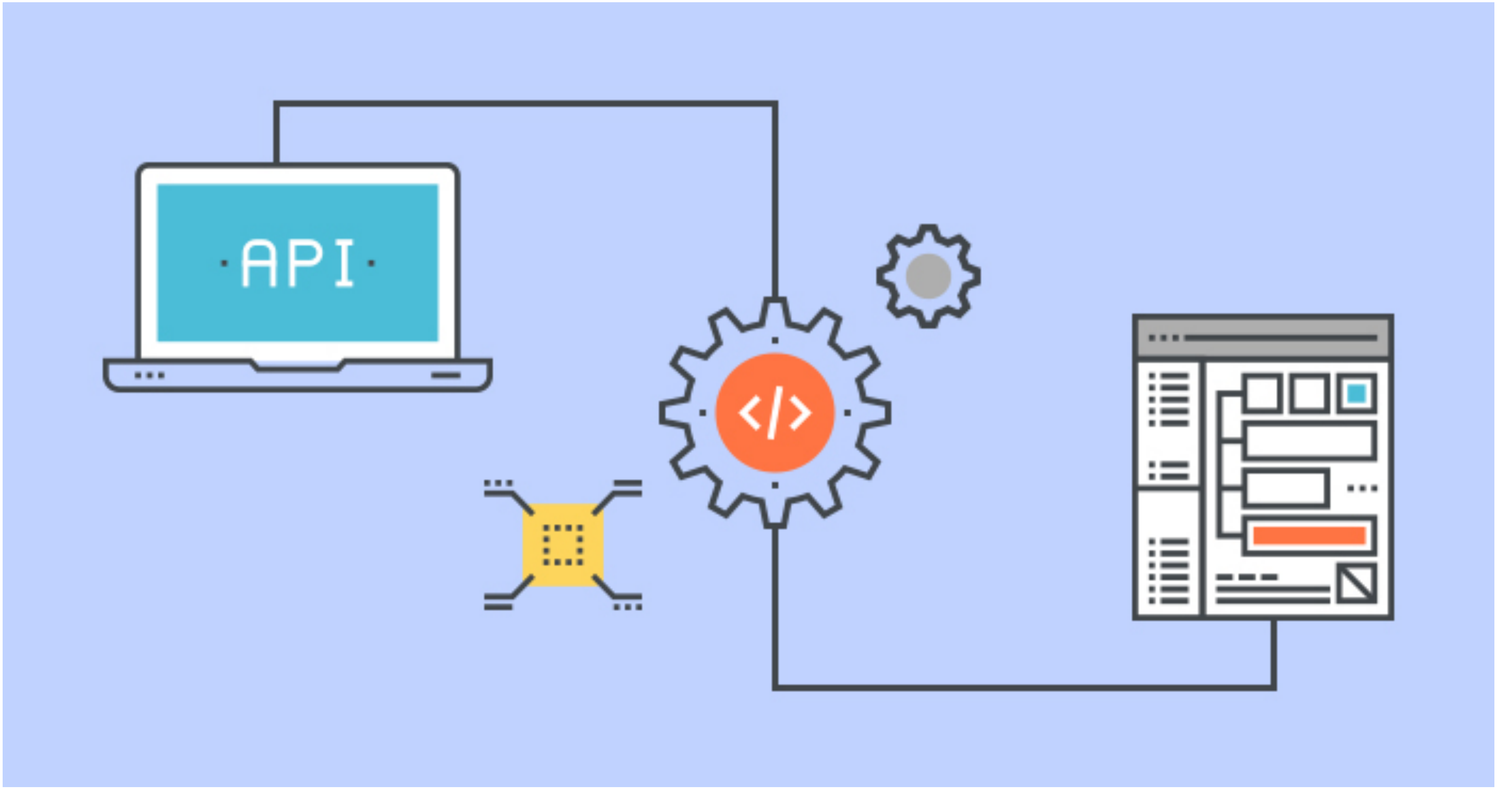
# REST Concepts

- **Resources** expose easily understood directory structure URIs.
- **Representations** transfer JSON or XML to represent data objects and attributes.
- **Messages** use HTTP methods explicitly (for example, GET, POST, PUT, and DELETE).
- **Stateless** interactions store no client context on the server between requests. State dependencies limit and restrict scalability. The client holds session state.

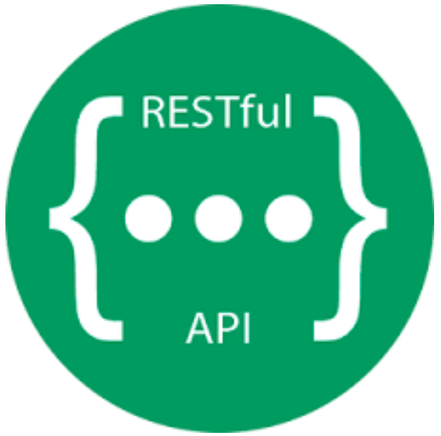


# Guiding Principles of REST

- Client–server
- Stateless
- Cacheable
- Uniform interface
- Layered system
- Code on demand (optional)



# Client–server - Stateless - Cacheable

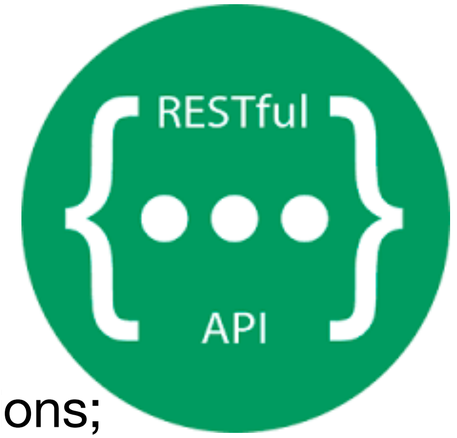



- **Client–server** – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

- **Stateless –** Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.

- **Cacheable** – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.

# Uniform Interface - Layered - Code-on-demand




- **Uniform interface** – REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

- **Layered system** – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.

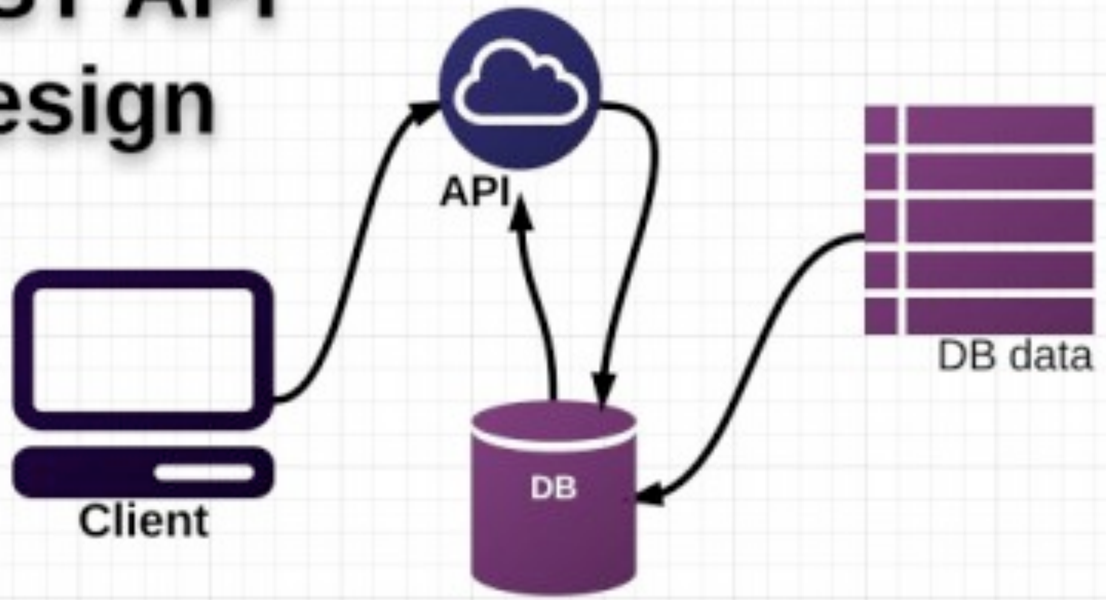
- **Code on demand (optional)** – REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

# Or more simply … REST is The Web Used Correctly

- A system or application architecture
- ... that uses HTTP, URI and other Web standards “correctly”
- ... is “on” the Web, not tunnelled through it ... also called ““RESTful HTTP”

# REST API Design

| | |
|---|---|
| **GET** | /tasks - display all tasks |
| **POST** | /tasks - create a new task |
| **GET** | /tasks/{id} - display a task by ID |
| **PUT** | /tasks/{id} - update a task by ID |
| **DELETE** | /tasks/{id} - delete a task by ID |



# Rest Practices

1: Give Everything and ID

2: Link Things Together

3: Use Standard HTTP Methods

4: Allow for Multiple Representations

5: Communicate Statelessly

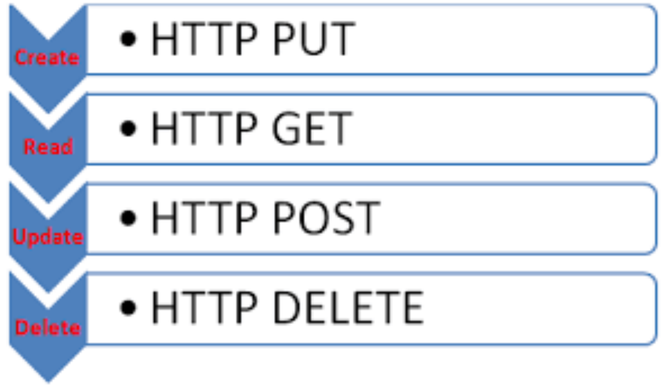
# 1: Give Every Thing and ID

- http://example.com/customers/1234
- http://example.com/orders/2007/10/776654
- http://example.com/products/4554
- http://example.com/processes/sal-increase-234

# 2: Link Things Together

```
<order self='http://example.com/orders/1234'>
  <amount>23</amount>
  <product ref='http://example.com/products/4554' />
  <customer ref='http://example.com/customers/1234' />
</order>
```

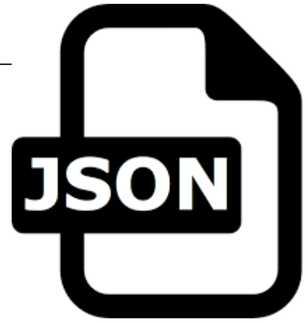
# 3: Use Standard HTTP Methods




| | |
|---|---|
| GET | retrieve information, possibly cached |
| PUT | Update or create with known ID |
| POST | Create or append sub-resource |
| DELETE | (Logically) remove |

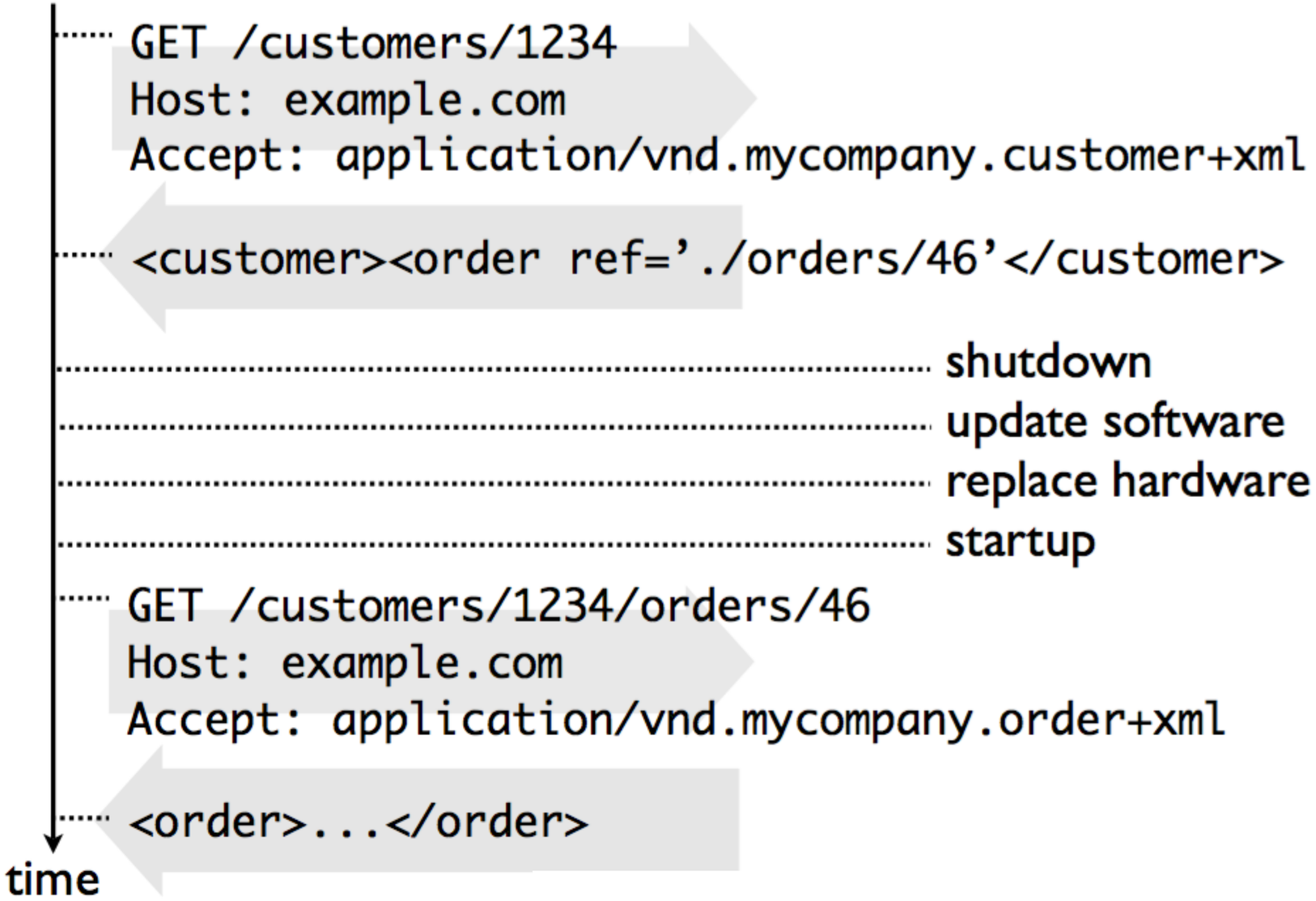
# 4: Allow for Multiple Representations

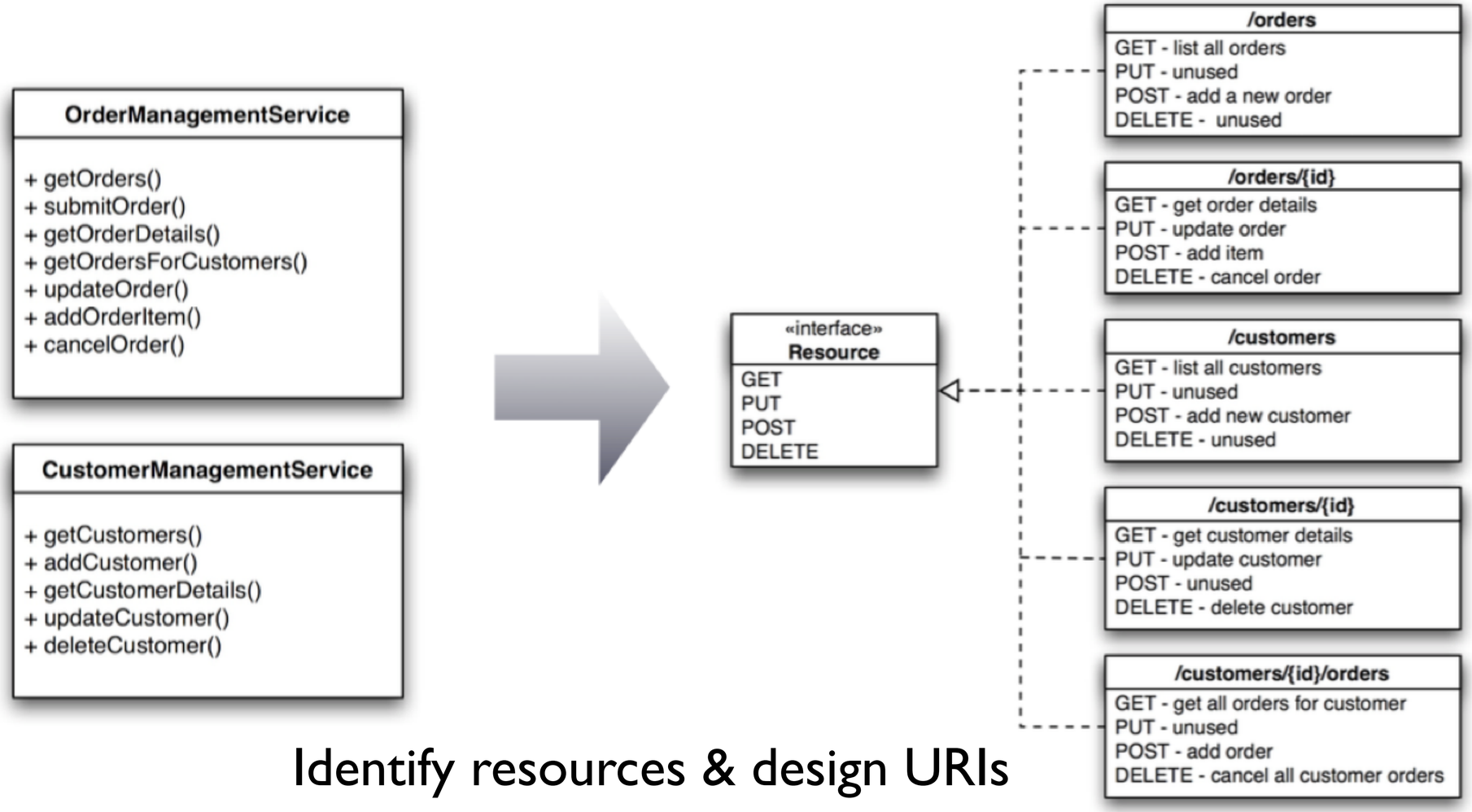
```
GET /donors/1234

Host: example.com

Accept: application/json

{

  "firstName" : "fred",

  "lastName"  : "simpson",

  "email"     : "fred@simpson.com",

  "password"  : "secret"

}
```




```
GET /donors/1234

Host: example.com

Accept: application/xml

<donor>

  <firstName> “fred” </firstName>

  <lastName> “simpson” </lastName>

  <email> “fred@simpson.com” </email>

  <password> “secret” </password>

</donor>
```

# 5: Communicate Statelessly






Identify resources & design URIs

Select format (Json)

Identify method semantics

Select response codes

# APIs


Full Stack Web Development