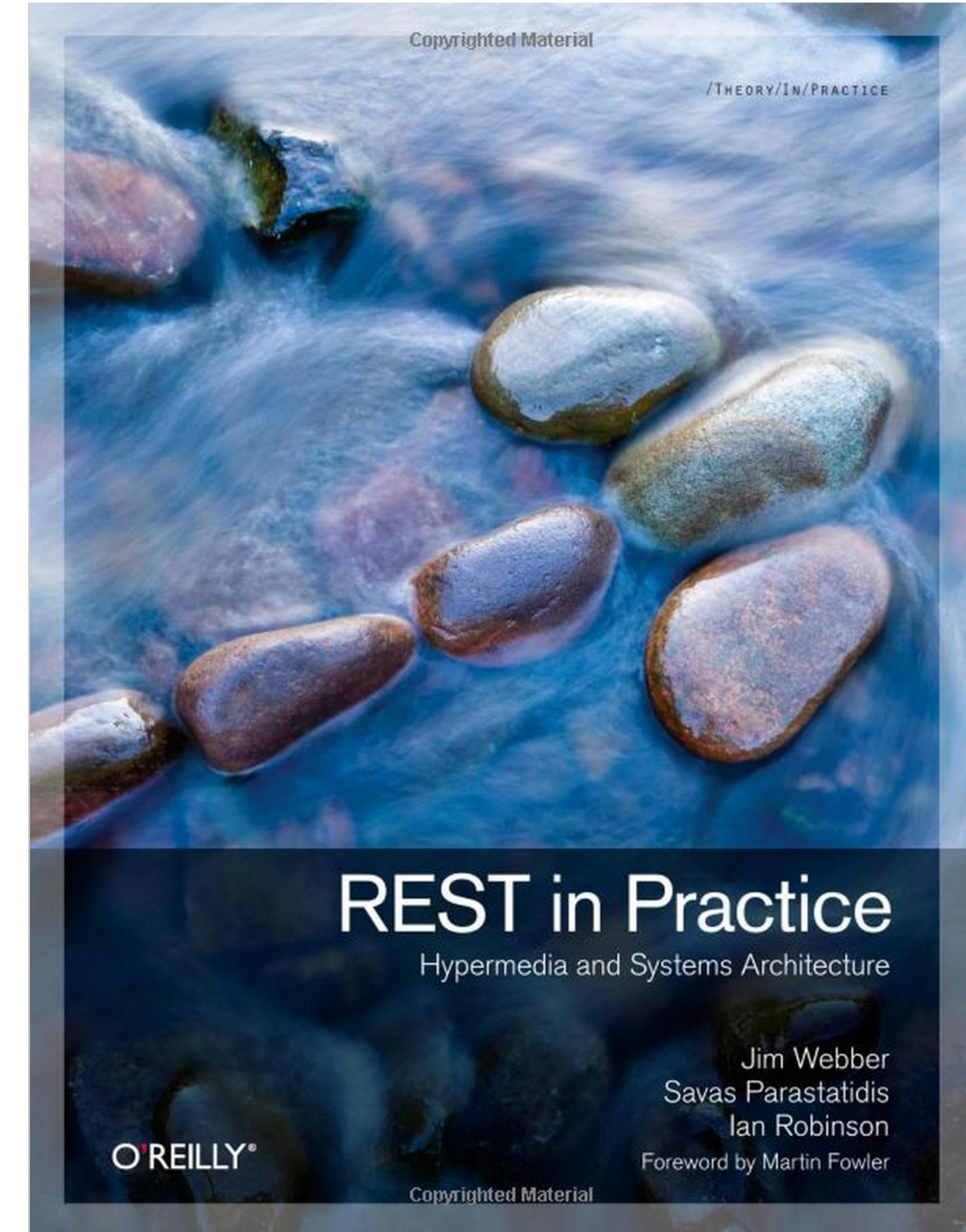


# REST : Representation State Transfer

---

# 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

# RESTful API

GET PUT POST DELETE

- 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.

# A Collective List Of APIs. Build Something.

weather

All

GRID

LIST

▼

▲

D3 VISUALIZATION

LEAFLET

DRONEDEPLOY

THE CAT API

CAT FACTS

CRYPTONATOR

TICKSEL

CHARITY SEARCH

BITCOINAVERAGE

WETRANSFER

LIVECOIN

HOLIDAYS

CLIMACELL MICRO WEATHER

METaweather

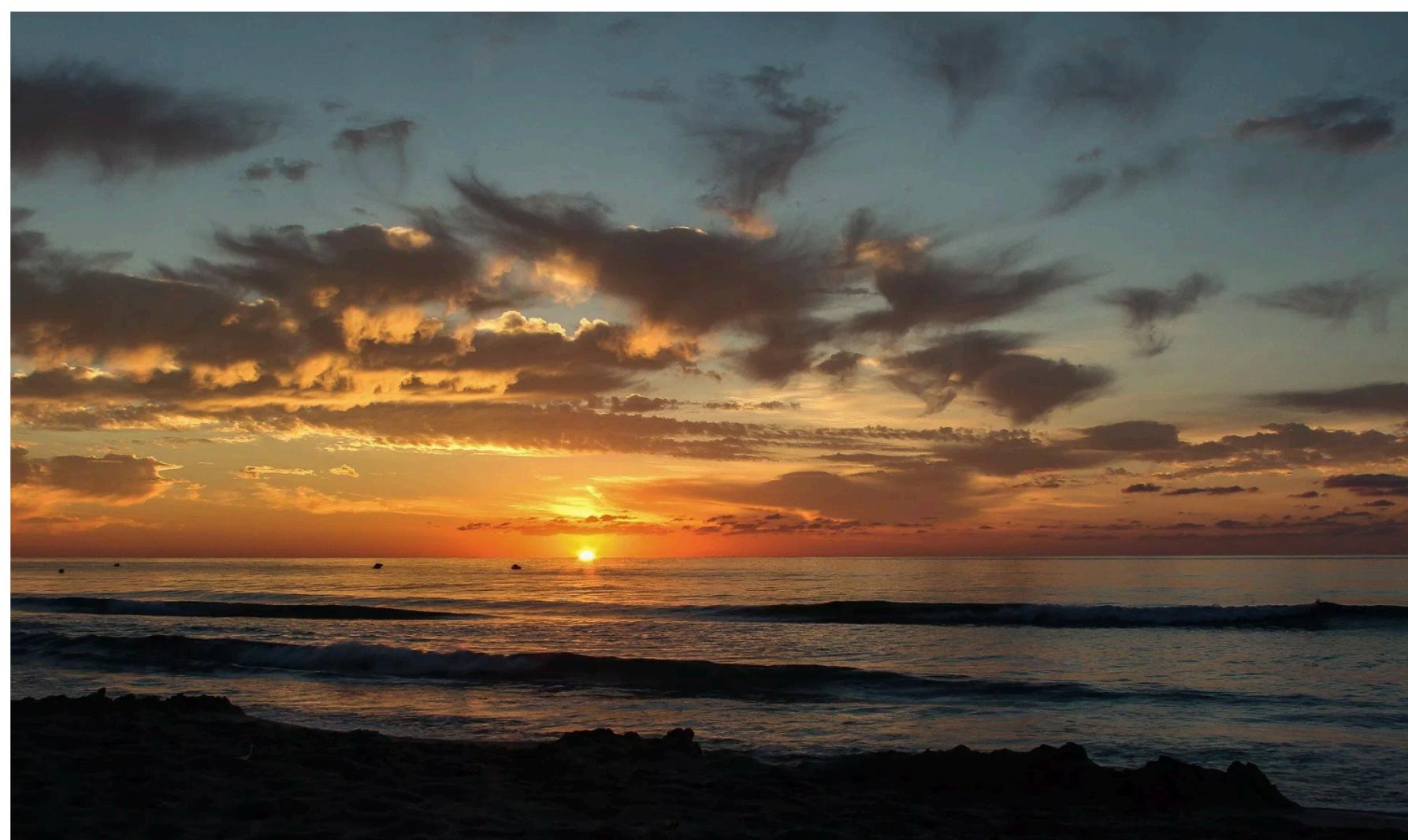
ODWEATHER

OPENWEATHERMAP

WEATHERBIT

YAHOO! WEATHER

<https://apilist.fun/>



## 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



Broken clouds

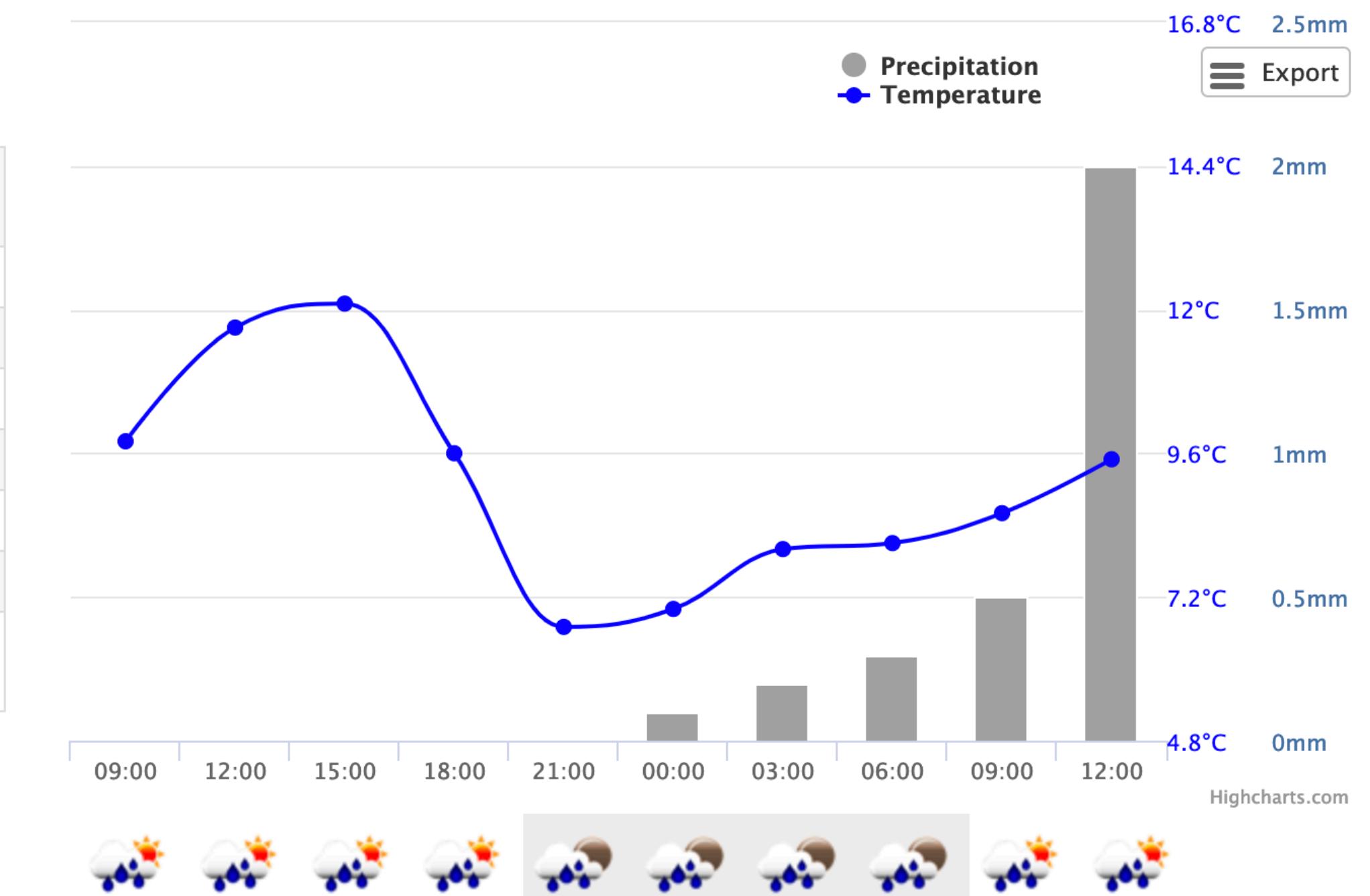
08:53 Feb 27 [Wrong data?](#)

|            |  |
|------------|--|
| 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.

Main Daily Hourly Chart Map

#### Weather and forecasts in Waterford, IE



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

We have combined our Weather services and **Satellite imagery** in one simple and fast **Agricultural API**. **Satellite images** (True & False color, NDVI, EVI), Weather data (Current and Forecast), Historical data, Soil temperature and moisture, Accumulated temperature and precipitation, etc.

[Try it right now!](#)

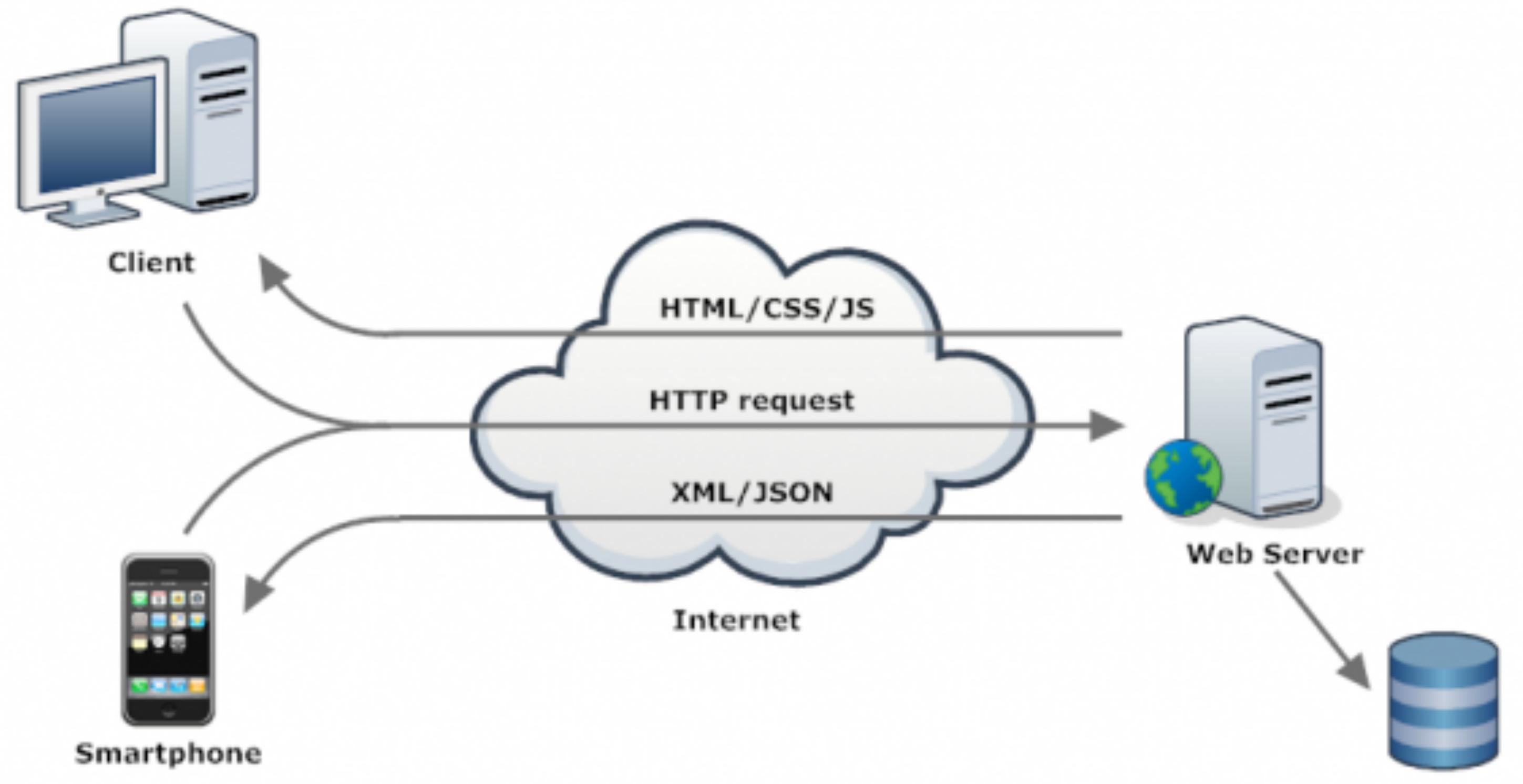
## Weather

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

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

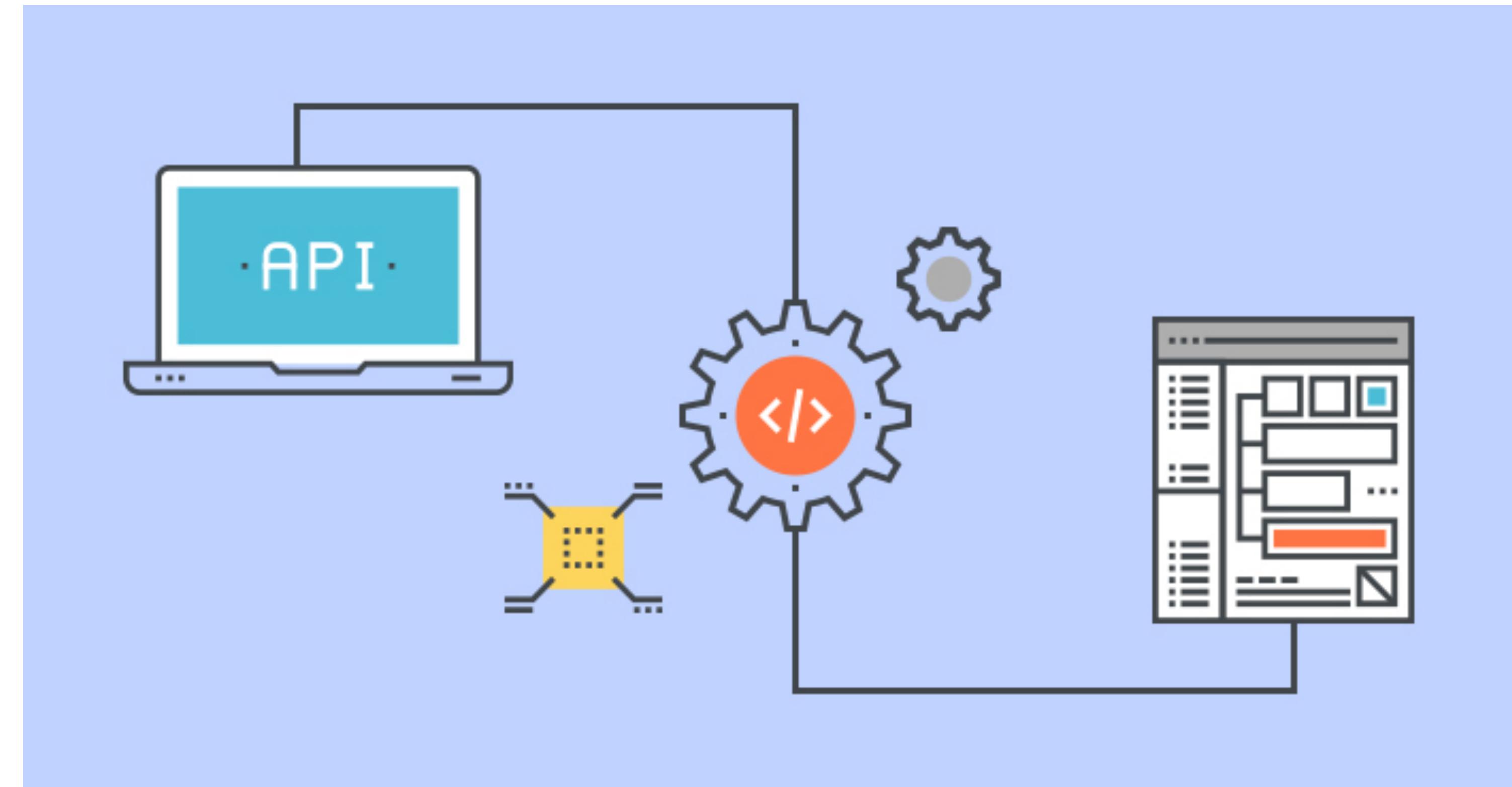
# REST Concepts

- **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.

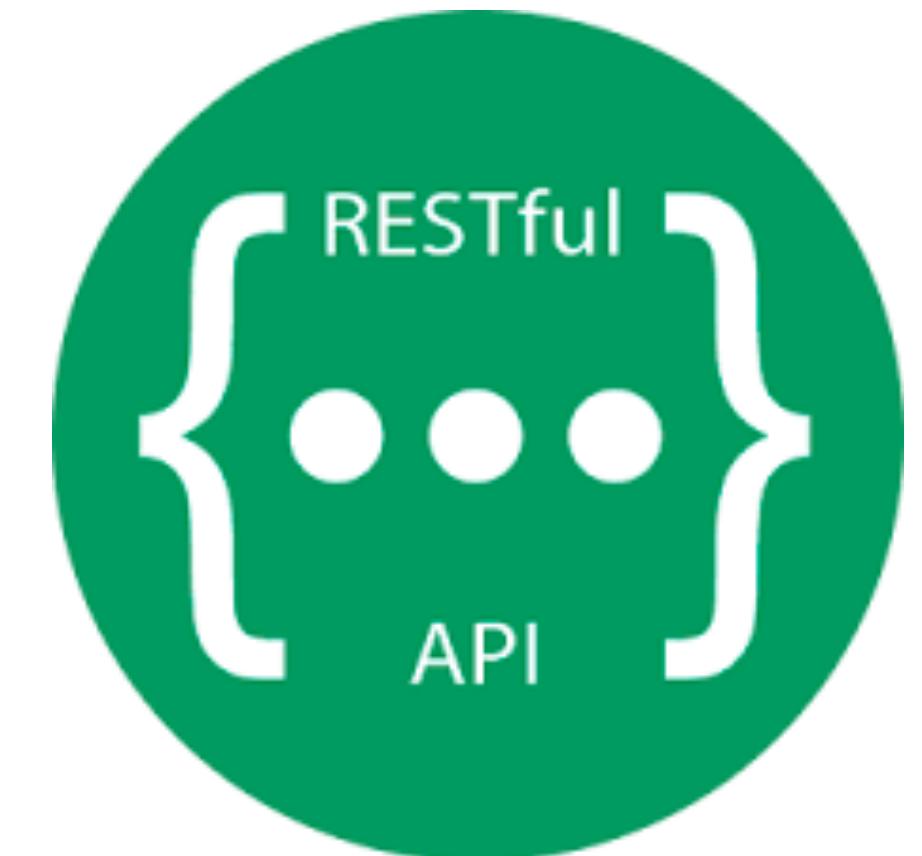


# Guiding Principles of REST

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

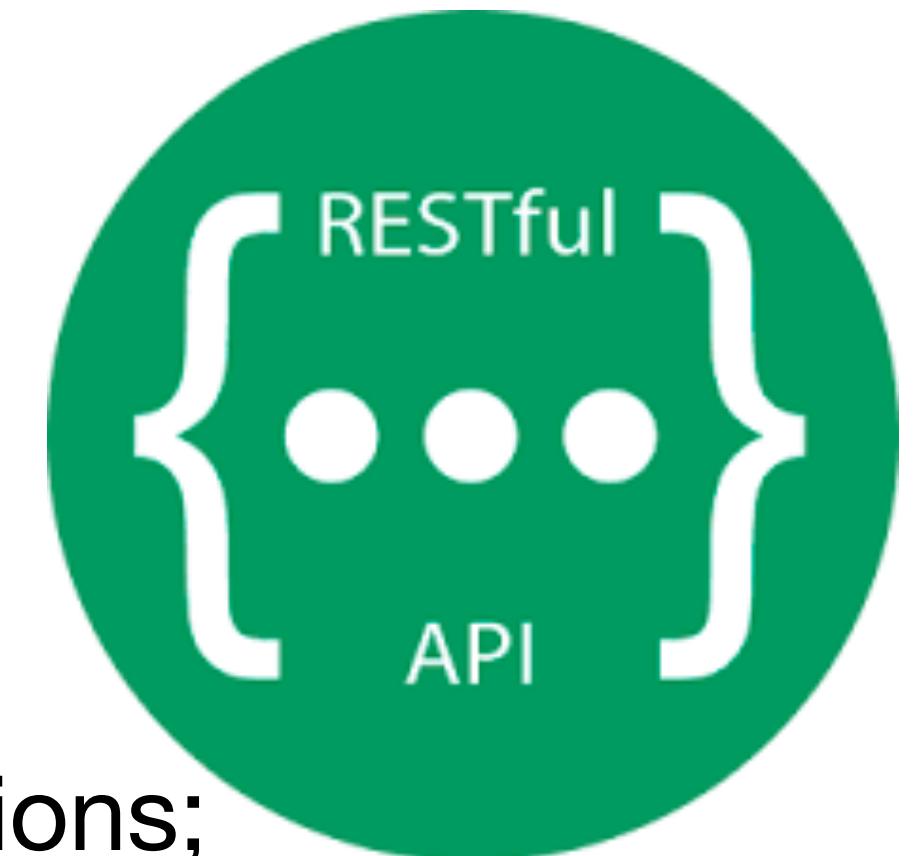


## Client–server - Stateless - Cacheable



- **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.

## Client–server - Stateless - Cacheable



- **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.

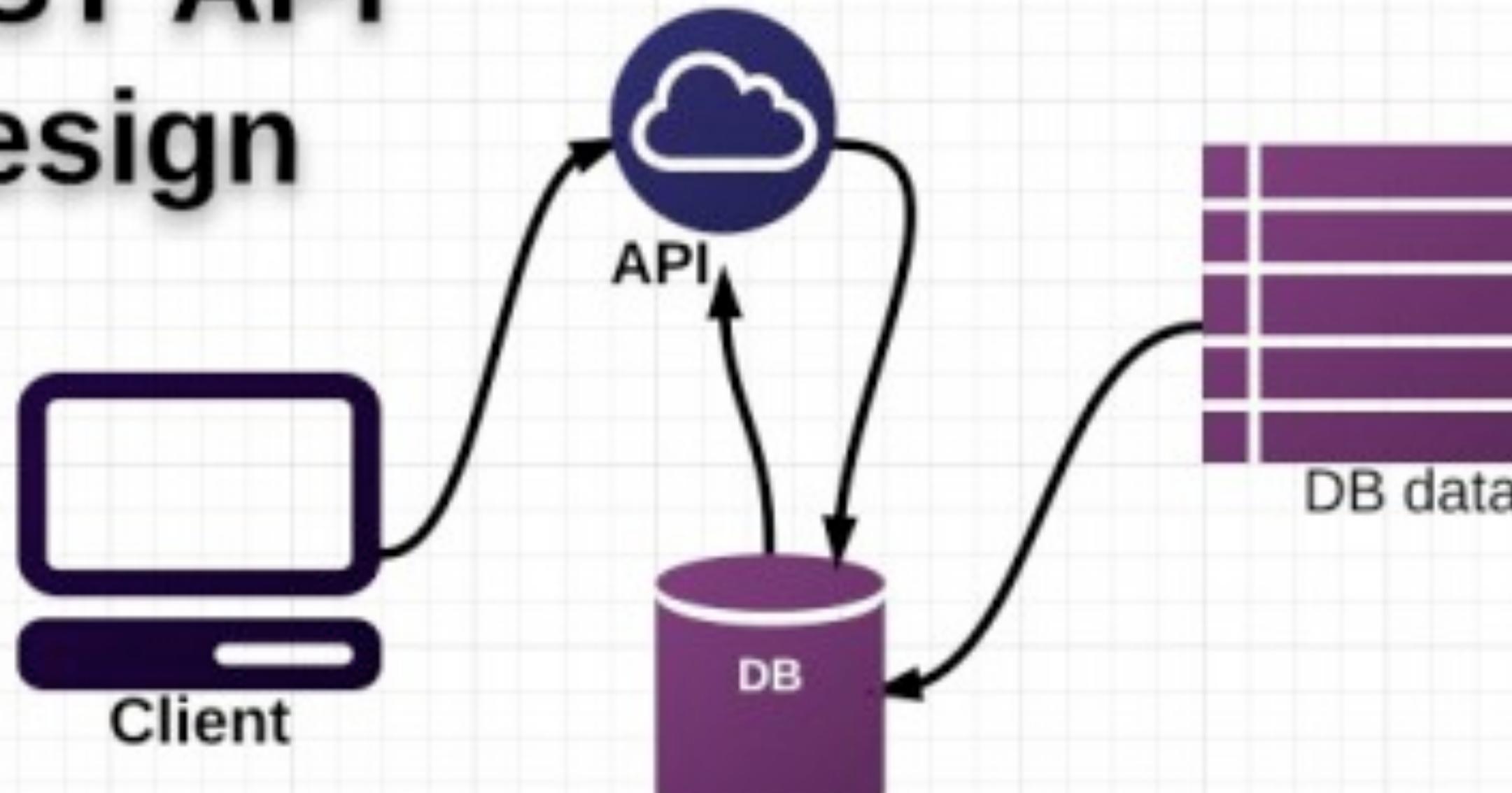
## Or more simply ... REST is The Web Used Correctly

- 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”

(see [http://www.slideshare.net/deimos/stefan-tilkov-pragmatic-intro-to-rest?qid=bf0300ad-6e68-4faa-bac7-0f77424ec093&v=qf1&b=&from\\_search=1](http://www.slideshare.net/deimos/stefan-tilkov-pragmatic-intro-to-rest?qid=bf0300ad-6e68-4faa-bac7-0f77424ec093&v=qf1&b=&from_search=1))

# REST API Design

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



# Rest Practices

- 1: Give Everything an 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>

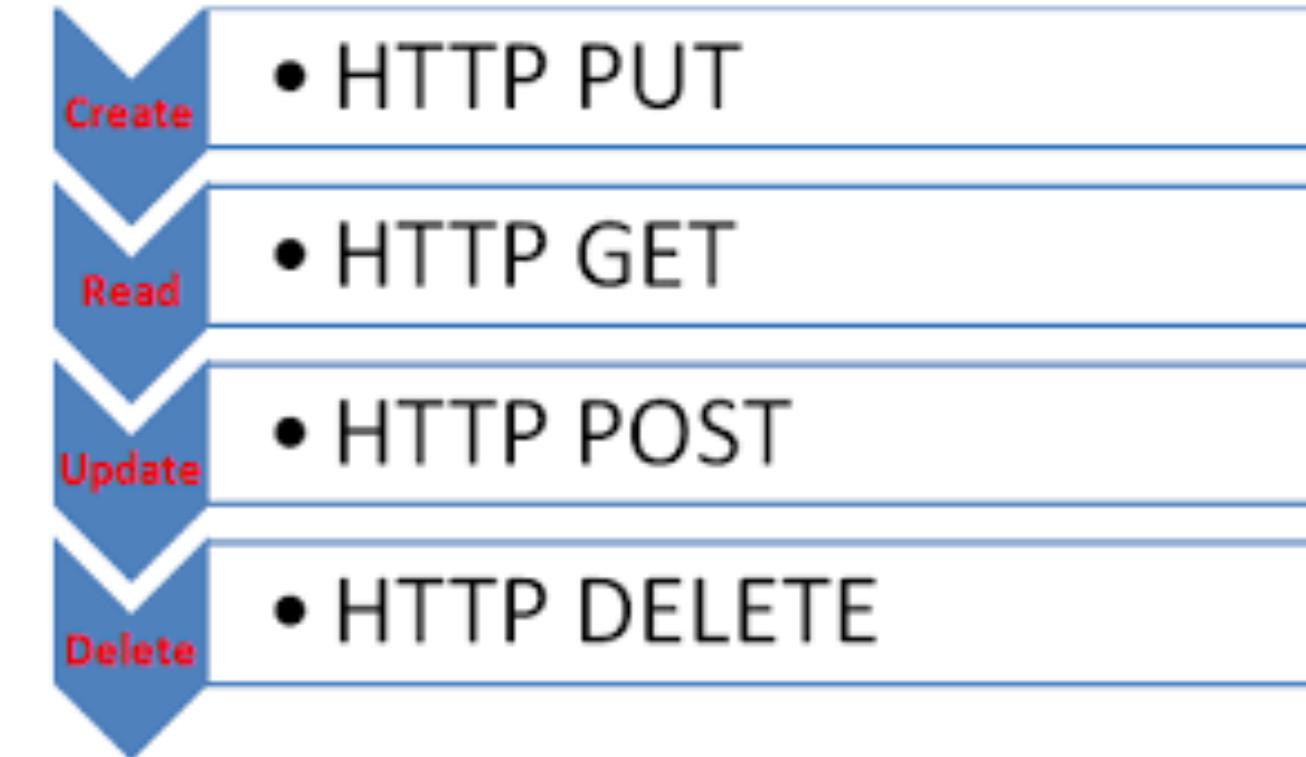


## 2: Link Things Together

```
<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

```
{
```

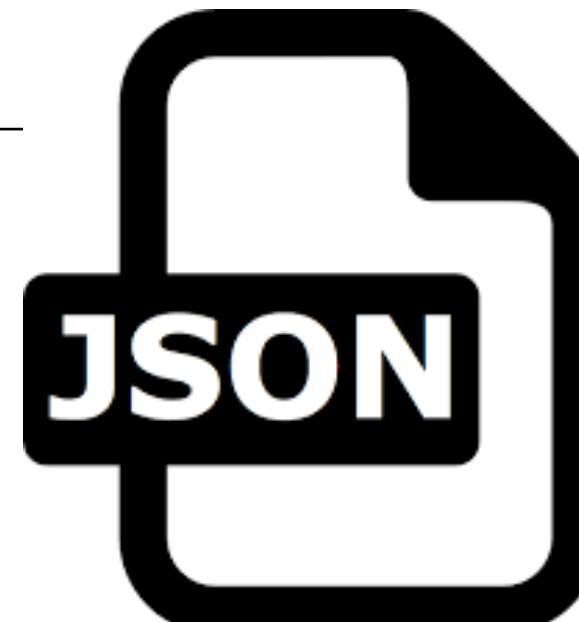
```
    "firstName" : "fred",
```

```
    "lastName" : "simpson",
```

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

```
    "password" : "secret"
```

```
}
```



```
GET /donors/1234
```

Host: example.com

Accept: application/xml

```
<donor>
```

```
    <firstName> "fred" </firstName>
```

```
    <lastName> "simpson" </lastName>
```

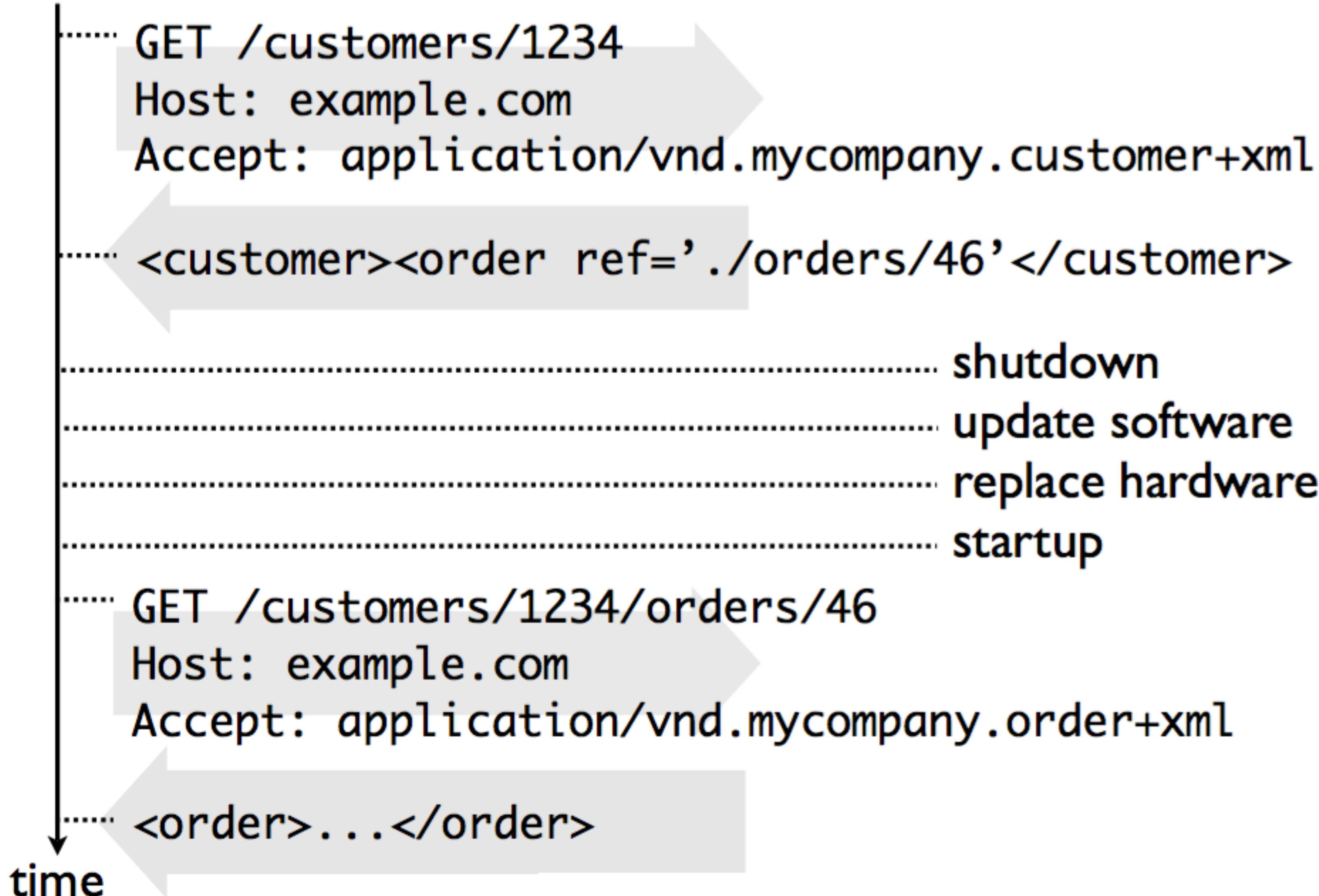
```
    <email> "fred@simpson.com" </email>
```

```
    <password> "secret" </password>
```

```
</donor>
```



## 5: Communicate Stateless



| OrderManagementService   |
|--|
| + getOrders()<br>+ submitOrder()<br>+ getOrderDetails()<br>+ getOrdersForCustomers()<br>+ updateOrder()<br>+ addOrderItem()<br>+ cancelOrder() |

| CustomerManagementService   |
|---|
| + getCustomers()<br>+ addCustomer()<br>+ getCustomerDetails()<br>+ updateCustomer()<br>+ deleteCustomer() |



| «interface» Resource         |
|------------------------------|
| GET<br>PUT<br>POST<br>DELETE |

|                                     |
|-------------------------------------|
| /orders                             |
| GET - list all orders               |
| PUT - unused                        |
| POST - add a new order              |
| DELETE - unused                     |
| /orders/{id}                        |
| GET - get order details             |
| PUT - update order                  |
| POST - add item                     |
| DELETE - cancel order               |
| /customers                          |
| GET - list all customers            |
| PUT - unused                        |
| POST - add new customer             |
| DELETE - unused                     |
| /customers/{id}                     |
| GET - get customer details          |
| PUT - update customer               |
| POST - unused                       |
| DELETE - delete customer            |
| /customers/{id}/orders              |
| GET - get all orders for customer   |
| PUT - unused                        |
| POST - add order                    |
| DELETE - cancel all customer orders |

Identify resources & design URIs

Select format (Json)

Identify method semantics

Select response codes