**API Workshop**

**SLIDE 1 INTRO**

I am Giovanni (Gio) and have **attended Le Wagon 1 year ago**. **Before attending** the bootcamp was a banker and had **no idea** what an API is or how to use one

Today, on the other hand, I am here to try **explaining what a API is** and how they make the **life** of a developer (or an advanced user of the web) **easier**

A Web API to be precise, as there are many types of APIs but we will be talking about this breed today

Two questions before we start:

1. How many of you have any programming knowledge?
2. How many of you know roughly what an API is? Anyone wanna try explaining it to classmates + make an example

**SLIDE 2**

**SLIDE 3**

An API is an interface through which **programs** (and by extension developers writing those programs) can **interact** and **exchange information** **with applications**

**SLIDE 4**

Taking a metaphor from the real world to explain this better:

Phone is application

What will be the interface?

**SLIDE 5**

The ***user* interface** will be the rotor

**SLIDE 6**

But remember we mentioned an API is built to **allow machines** (programs, scripts call them as you like) to use it, and can you imagine our program (wall-e in this example) **using a rotor** with its large fingers?

**SLIDE 7**

It will be **hard**. **Because humans and machines are not alike, and do not have the same requirements**.

For humans, **sight** is the most natural way of interpreting things, for machines the opposite: seeing its crazy hard (at least up until now – with ML advances this is changing too).

A **tool built for a human will not be machine-friendly**, and the other way around, an API will not be human-friendly

A **web service meant to be used by final consumer will usually have both a human-friendly interface (GUI) and a program-friendly interface (an API)** – *and we will see this later*. So why would such a service have also an API? After all, we live in a world which is dominated by humans not machines (for now at least)…

**SLIDE 8**

**Developers!** A special breed of humans (with the superpower of knowing how to talk to machines) who will write code and programs that make use of APIs. So humans do **not make direct use of APIs, but they write code that does**. Why do developers even use APIs? Because it **simplifies their job** and allows them to implement cool things in their applications without the need to reinvent the wheel

This principle is a very important one in software development. It is called ***DRY***. Do not repeat your code but reuse it. In the same way if someone has already built an application doing something, use it, do not build it again. A large part of a **developer’s job** is this as well, piping together parts that work well alone to create a new cool product

**SLIDE 9**

**SLIDE 10**

Let’s take another real life example: what does this interface expose?

**SLIDE 11**

**SLIDE 12**

A developer will create a **program** (the plug) **consuming** whatever the API exposes.

Web APIs like plugs, **follow a standard**: it doesn’t matter what the program actually does or who built it and it what language, the consumption mechanism will be the same

**SLIDE 13**

For **RESTful Web APIs** (which are the great majority of all) this standard used for exchanges **is HTTP**, the communication protocol at the base of the web

Does any of you know how does the web work behind the scene? Explain it?

Exchanges of data through the HTTP protocol are the **basis of the web**: visiting a website causes an **HTTP request to be sent out by a user’s computer (the client)** which is followed by **a response from the computer behind the website (the server)** that sends back an **HTML** file with the information you have requested. HTML is used when you are surfing the web because this is a format that can be graphically represented by your browser on the screen and hence it is easy to interpret for the human eye

Similarly, also using an **API entails an exchange**: a request is sent out by the client (which this time is not a user but an application or a developer) and a response is sent back from the server. This time, though, data will be sent in a different format which is easier to interpret for the machine (like **JSON or XML**) because as I mentioned before humans and machines will have different requirements

Who knows what JSON is?

* **GOOGLE SEARCH HARRODS**
* [**https://serpapi.com/**](https://serpapi.com/)

ANY QUESTIONS?

**SLIDE 14**

* Link

On this website you have a collection of all the **most famous web APIs** if you want to take a look

**SLIDE 15**

Let’s take the first one from the list. Does anyone know what does the Google Maps API do?

**SLIDE 16**

It allows developers to **implement google’s maps on their page** and do all sorts of things with it like customisation

**PLUG IS AIRBNB (showcase results, customise markers)**

* <https://snazzymaps.com/>

This API is free up to something like 100k monthly requests – what do they obtain from it?

**Marketing, advertisement, data, even money**

An important point is that an **API offers access for the publisher** on others’ applications which can be used in different ways

This means as a developer you always need to **be careful** about what APIs you decide to use as they can expose your website to **security threats**

**SLIDE 17**

Another famous API is Twilio which allows developers to **send automatic texts upon some event**

**Before** needed to **implement, maintain and update with all phone carriers**

Today, against **payment of xyz, Twilio deals with that for you** and makes it very easy to send texts to all carriers even if things change overnight

**SLIDE 18**

**PLUG IS UBER (send text when registration or car arrives)**

**BOTH CASES, LIFE OF DEVELOPER WAS MUCH EASIER AND COULD BUILD COOL THINGS ON APIs**

**SLIDE 19**

Let’s see a case study of how would a developer come about using an API

**SLIDE 20  
SLIDE 21  
SLIDE 22**

* **Content manager could go on maps every time, type the address and get the coordinates**

**SLIDE 23**

As the engineer on the product, **build a way through which the program behind the scenes saves the coordinates** when the content manager saves location

* **Like a dev on the job, search on google “how to transform an address into coordinates programming”**

**OFC THERE IS AN API ALLOWING A PROGRAM TO DO THAT AUTOMATICALLY**

**SLIDE 24  
SLIDE 25  
SLIDE 26**

**A tool through which we want to call the API with an address like this**

**SLIDE 27**

**Get this JSON as response**

**SLIDE 28  
SLIDE 29**

**HTTP *GET* request**

**SLIDE 30**

As explained before **browsing web is making HTTP request to website server to GET HTML file**

Hence, **an HTTP request to the geocoding API server to GET JSON file**

* **BROWSE TO MAPS API LINK FROM SLIDE**

**HOWEVER THIS WILL CAUSE ACCESS DENIED ERROR BECAUSE GOOGLE NEEDS TO KNOW WHO IS ASKING (LIKE WHEN YOU LOGIN ON WEBSITES)**

* Add at the end of the link ***“&key={YOUR\_API\_KEY}”***

Majority of APIs **need setup** before you can use them. To help you in the setup and usage of an API, **publishers offer documentation** (which are like instructions manual)

**SLIDE 31  
SLIDE 32**

As a developer will **build a tool like this** which the content manager will fill every time

**Behind the scenes our website server will call the API just like we did, open the JSON response and store the coordinates it founds in our database**

**SLIDE 33**

When visiting venue page will see them there

**SLIDE 34  
SLIDE 35  
SLIDE 36  
SLIDE 37**

AGAIN

**SLIDE 38  
SLIDE 39  
SLIDE 40**

No, we need the map

**SLIDE 41**

**2 APIs: CLIENT SIDE MAP + SERVER SIDE GEOCODING**

ANY QUESTIONS?

**SLIDE 42  
SLIDE 43**

Assume we want to implement this new feature

We know that Twilio does this

**SLIDE 44  
SLIDE 45**

Setup API following documentation **SLIDE 46**

* **Terminal mkdir, touch, cd, open text editor**
* **Copy paste ruby code from slide with appropriate creds**
* **Save variable message = client.messages.create**
* **Puts message.sid, .status, .uri**

**SLIDE 47**

* **Show message received on phone**

What **Ruby did** for us is basically an **HTTP POST request to the API endpoint with that information, and Twilio sent the message**

* **Navigate to API URI to show status**

<https://api.twilio.com/>{message.uri}

ANY QUESTIONS?

**SLIDE 48  
SLIDE 49**

So now we will take a look at **webhooks**, which are an interesting, powerful **variation of APIs**

**SLIDE 50**

You all know what the F5 key does right? It refreshes a page (i.e**. sends a new HTTP request**)

**SLIDE 51**

Imagine **adding feature on Timeout to display refreshed list of attendees from Meetup**

We could do it by **telling our program to call the API every 10 secs**

**UNEFFICIENT, LOADS OF REQUESTS (WHEN THEY ARE PAID NOT WANT THIS).**

Integration manager at Meetup will call you to complain server overload

**SLIDE 52**

*“WE WILL LET YOU KNOW”* principle

**SLIDE 53**

**USE WEBHOOK: OPPOSITE OF API**

**Create an API endpoint for your app** that meetup server will call when new attendee

**SLIDE 54  
SLIDE 55**

***FROM YOUR PERSPECTIVE ONE IS API AND ONE WEBHOOK (YOUR MACHINE IN ONE CASE IS CLIENT AND IN ONE IS SERVER)*  
BUT IN THE END THEY ARE THE SAME THING (A CLIENT SENDING AN HTTP REQUEST TO A SERVER)**

**SLIDE 56  
SLIDE 57  
SLIDE 58  
SLIDE 59  
SLIDE 60  
SLIDE 61  
SLIDE 62  
SLIDE 63**

* **Create form**
* **Test form**
* **Create trello board and lists**
* **Create integration on Typeform**
* **Test integration**
* **Show Trello API call doc**
* **Show webhook on requestbin**

**SLIDE 64  
SLIDE 65**

ANY QUESTIONS?