Capítol 2: API

Aina Palacios

Aina Palacios

- Enginyera de Telecomunicacions especialitzada en Audiovisuals
- Màster en Tecnologías Avançades especialitzada en deep learning en Multimèdia!
- Experiència en programació web i machine learning.
- Mentora a IT Academy de **Vuejs**

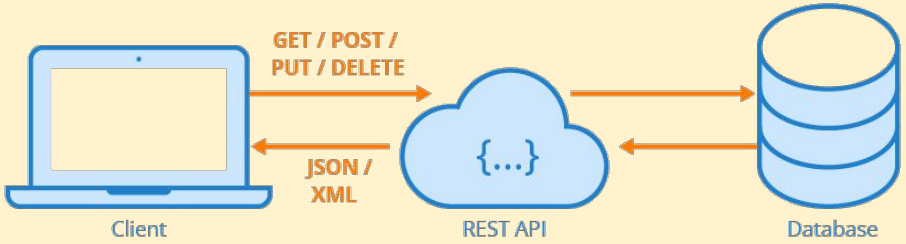
https://www.linkedin.com/in/ainapc/

ainaPali#2617

Què és una API?

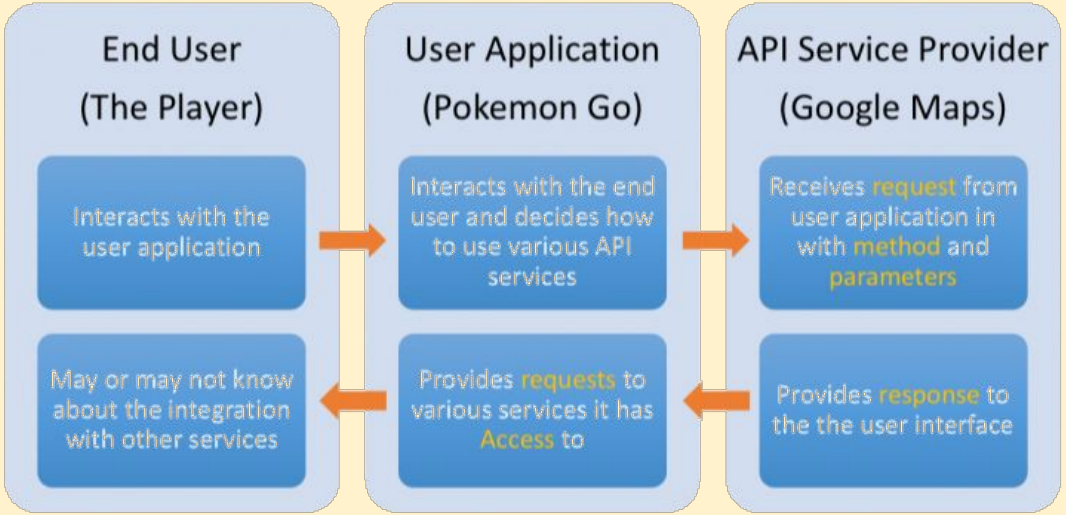
Application Programming Interface -> interfície que permet la intercomunicació entre dos sistemes. Els mètodes d'aquestes ja estan predefinides!

- És independent del llenguatge utilitzat
- Permeten l'entrada a un software que no ha de ser necessàriament de la teva propietat, i fer ús d'aquest sense haver-lo de tenir.
- No necessites saber com funciona l'altre software!



Per què serveix una API?

- Intercanvi de dades o funcionalitats entre dos sistemes diferents! (Ex. El temps)
- Pots integrar la funcionalitat al teu programa.
- Pots accedir a una base de dades
- Pots modificar aquesta base de dades!
- Pots esborrar dades!



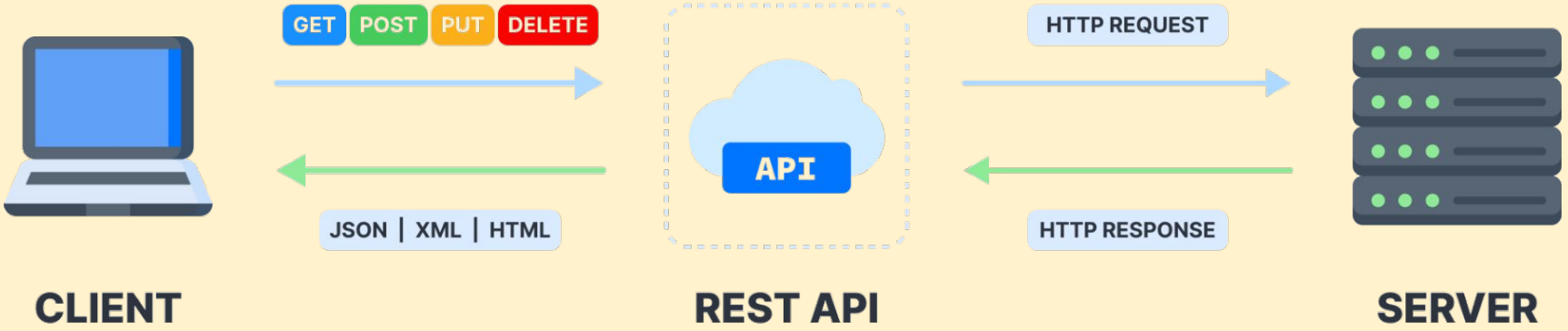
Elements bàsics d'una API

- **Accés**: Autorització a comunicar-se amb l'API -> KEY
- **Request**: Dada o servei que es demana a l'API. Conté dues parts:
  - **Mètode/Method:** Pregunta que li fas a l'API
  - **Paràmetres:** Informació extra
- **Response:** Resposta o servei que et proporciona l'API depenent de la teva Request

Operacions més comuns: CRUD

Operació	Què fa?
Create -> POST	Enviar dades al server
Read -> GET	Rebre dades del server
Update -> PUT	Modificar dades
Delete -> DELETE	Eliminar dades

REST API Model



Podem accedir a la informació a partir d'un URL públic.

Status Codes

Quan l'API ens retorna el **response** sol està acompanyat d'un codi que ens aporta info extra:

https://developer.mozilla.org/es/docs/Web/HTTP/Status

API Endpoints

Quan nosaltres estem cridant l'API, aquesta pot tenir diferents rutes que ens permeten accedir a diferent informació. Aquestes rutes s'anomenen endpoints. Mirem alguns exemples:

- Per accedir per exemple a la informació del client, podem utilitzar diferents endpoints:
  - Mètode HTTP serà **GET**
  - Per accedir a tots els clients, l'endpoint serà **/customers**
  - Per accedir a la informació d'un client en específic, podem tenir un altre endpoint com ara **/customers/<custoemer_id>**
- Podem tenir un mateix endpoint per diferents mètodes:
  - Amb endpoint **/customers** i el mètode **GET** ens retornen la llista de clients.
  - Amb endpoint **/customers** i el mètode **PUT** podem crear un nou client

Accedir a una API!

Preparar l'entorn

Per poder fer HTTP request, python ens proporciona una llibreria:

$**python3 -m pip install requests**

Les API REST segueixen el protocol que hem vist, però cada API serà diferent i tindrà uns endpoints diferents.

Nosaltres utilitzarem el següent api per començar:

https://kinduff.github.io/dog-api/

APIs d'interès:

https://cataas.com/#/

https://jsonplaceholder.typicode.com/

https://swapi.dev/

Fixeu-vos que si busqueu directament el URL en el vostre navegador, us farà un GET del URL

GET dogs Facts

```
In [6]: import requests # Importem la llibreria

        api_url = "https://dog-api.kinduff.com/api/facts" # Aquesta és l'URL de la nostra API. Endpoint /api/facts

        response = requests.get(api_url) # Fem un request a la api amb un get
        response.json() #Obtenim el JSON de la resposta!

Out[6]: {'facts': ['An estimated 1,000,000 dogs in the U.S. have been named as the primary beneficiaries in their owner's
        will.'],
         'success': True}
```

```
response.status_code

200
```

```
response.headers["Content-Type"]

'application/json;charset=utf-8'
```

CRUD a una API

Per poder practicar tots els mètodes HTTP farèm servir l'API jsonplaceholder, que simula dades falses.

1.2.1 GET

```
import requests

api_url = "https://jsonplaceholder.typicode.com/posts/1"

response = requests.get(api_url)
response.json()
```

```
{'userId': 1,
 'id': 1,
 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit',
 'body': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas t
otam\nnostrum rerum est autem sunt rem eveniet architecto'}
```

POST

Si ens interessa afegir dades a la base de dades, podem fer un POST. No només cridarem a la url, sinó que a més li hem d'entregar la informació que necessita, en format objecte.

```
api_url = "https://jsonplaceholder.typicode.com/todos"
todo = {"userId": 1, "title": "Buy milk", "completed": False}
response = requests.post(api_url, json=todo)
response.json()
```

```
{'userId': 1, 'title': 'Buy milk', 'completed': False, 'id': 201}
```

```
response.status_code
```

```
201
```

PUT

```python
api_url = "https://jsonplaceholder.typicode.com/todos/10"
response = requests.get(api_url)
print("\nResposta a l'id 10 abans de fer el put: ")
print(response.json())


todo = {"userId": 1, "title": "Wash car", "completed": True}
response = requests.put(api_url, json=todo)
print("\n\nResposta a l'id 10 després de fer el put: ")

print(response.json())

print("\n\nSatatus code ")
print(response.status_code)
```

```
Resposta a l'id 10 abans de fer el put:
{'userId': 1, 'id': 10, 'title': 'illo est ratione doloremque quia maiores aut', 'completed': True}


Resposta a l'id 10 després de fer el put:
{'userId': 1, 'title': 'Wash car', 'completed': True, 'id': 10}


Satatus code
200
```

DELETE

```
api_url = "https://jsonplaceholder.typicode.com/todos/10"
response = requests.delete(api_url)
response.json()
```

```
{}
```

```
response.status_code
```

```
200
```

Existeixen molts mètodes, però els que hem vist són els principals. Cada API conté els seus propis endpoints amb els seus mètodes i tots estan anomenats als documents de la mateixa API.

QUERY PARAMETERS

També ens podem trobar que la nostra API conté query params. És informació extra per la nostra request:

```
api_url = "https://dog-api.kinduff.com/api/facts" #Volem cridar a https://dog-api.kinduff.com/api/facts?number=5
params = {
    "number": 5
}
response = requests.get(api_url, params = params)

for res in response.json()['facts']:
    print("\n " + res)
```

```
 At about 6 inches, the Chihuahua is the shortest breed.

 Dogs are natural pack animals.

 About 12 percent of the air that a dog breathes goes into a special area in the back of the nose that is dedicate
d to smelling.

 Why are dogs' noses so wet? Dogs' noses secrete a thin layer of mucous that helps them absorb scent. They then li
ck their noses to sample the scent through their mouth.

 President Lyndon Johnson had two beagles named Him and Her.
```

Crear una API

Amb Python no només podem accedir a la informació d'una API, també podem crear la nostra pròpia!

Primers passos

El primer pas sempre serà preparar el nostre protocol. Això vol dir:

- Primerament, hem de tenir una base de dades. En aquest cas, utilitzarem CSV per facilitar les coses, però podem utilitzar el que vulguem -> JSON, XML, MongoDB, crides a SQL....
- Hem de definir els nostres Endpoints i Query Params
- Hem de baixar-nos les llibreries necessàries per crear aquesta API

	id_gos	nom	edat
0	101	Pitufina	8
1	102	Rex	10
2	103	Lolo	5
3	104	Coco	2
4	105	Nina	12
5	106	Milka	8

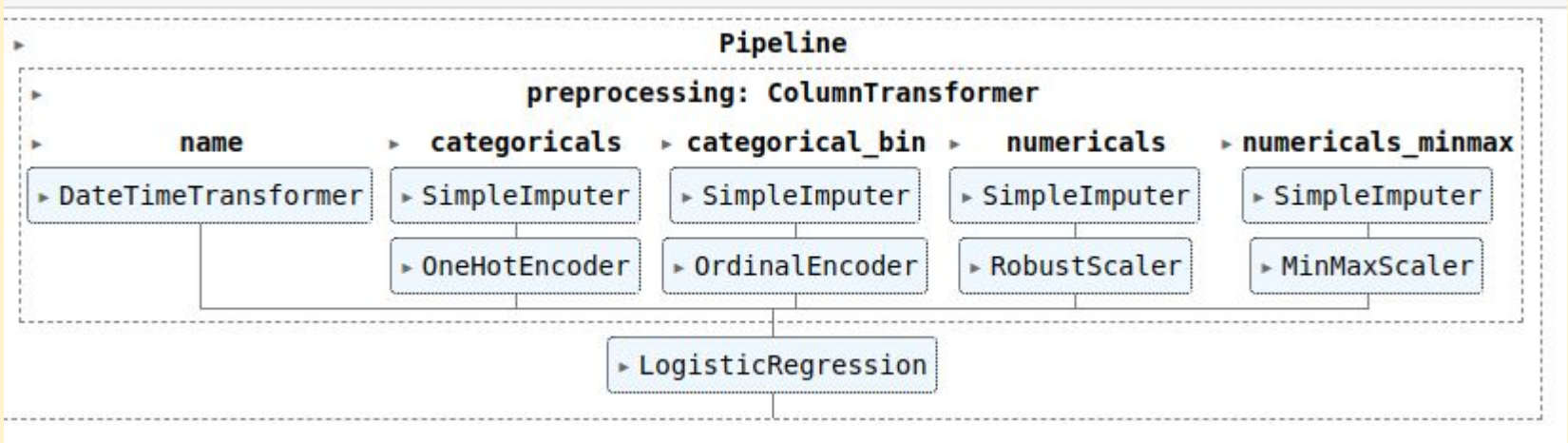
	id	Nom	Gossos
0	1	Aina	[101, 105]
1	2	Dani	[102]
2	3	Laia	[104]
3	4	Bego	[103]

Guardar el meu model

Guardar el nostre model!

Ara que ja hem vist com funcionen les APIs, com podem crear la nostra pròpia API en Flask, però com podem cridar al nostre model?

1. **Crear Pipeline**



CUSTOMTRANSFORM

El nostre CustomTransform no es pot guardar simplement. Aquesta Classe l'haurem de carregar com si fos una llibreria. Per fer-ho, crearem un arxiu python amb la classe:

```
from classes.customTransformers import CustomTransformer
```

Folder File

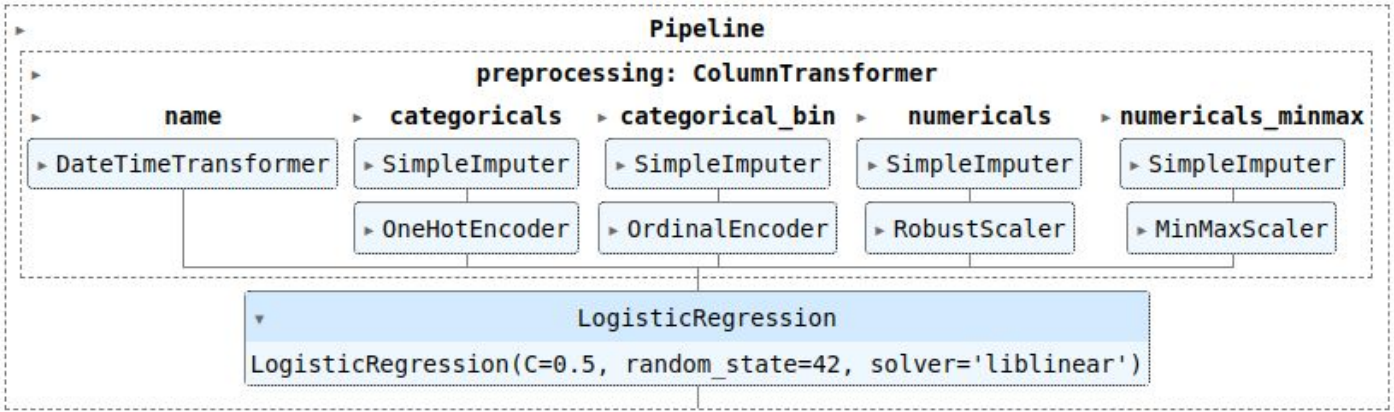
```
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np

class CustomTransformer(BaseEstimator, TransformerMixin):

    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        return np.c_[[self.get_title(x) for x in X]]

    def get_title(self, x):

        title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}

        value_title = 0
        for key, value in title_mapping.items():
            if key in x:
                value_title = value
                continue
        return value_title
```

https://rebeccabilbro.github.io/module-main-has-no-attribute/

2. Entrenar amb totes les dades

```
X = df.drop('Survived', axis = 1)
y = df.Survived
```

```
pipeline.fit(X,y)
```



3. Guardar el model

```
import joblib
joblib.dump(pipeline, 'best_model.pkl')
```

```
['best_model.pkl']
```

4. Comprovar el model!

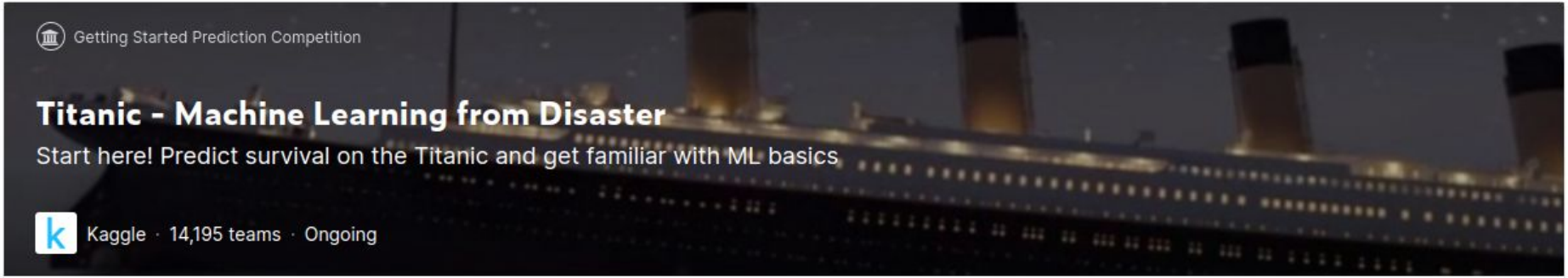
```
pipeline_loaded = joblib.load('best_model.pkl')
```

```
df = pd.read_csv('test.csv', index_col='PassengerId')
df.head()
```

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

```
df['Survived'] = pipeline_loaded.predict(df)
```

```
df['Survived'].to_csv('out.csv')
```



Overview Data Code Discussion Leaderboard Rules Team Submissions **Submit Predictions** ···

Submissions

All Successful Errors Recent

Submission and Description	Public Score
out.csv Complete · 7m ago · First submit	**0.77272**

Hora de crear una API pel model!

És el vostre torn!

Ets capaç de crear una classe POST?

PISTA:

1. Crear un endpoint
2. Decidir com accedir a les dades
3. Retornar el resultat!

Ja hem acabat!

Gràcies a tots!