CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Introduction to API (Consumer)

Restaurant

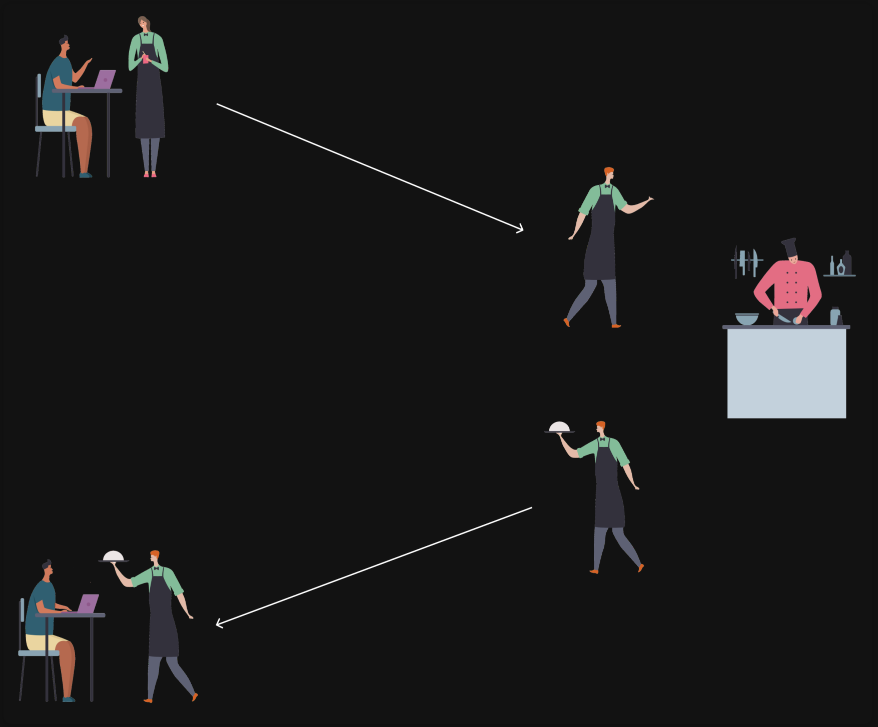


Image by macrovector

- The waiter comes to your table and gives you a menu
- You choose the item you would like from the menu
- The waiter takes your order to the kitchen
- The chef prepares your order and puts it away to be collected
- The waiter collects your completed order
- The waiter brings the order to your table

What is an API

An intermediary between a client and a server

The API

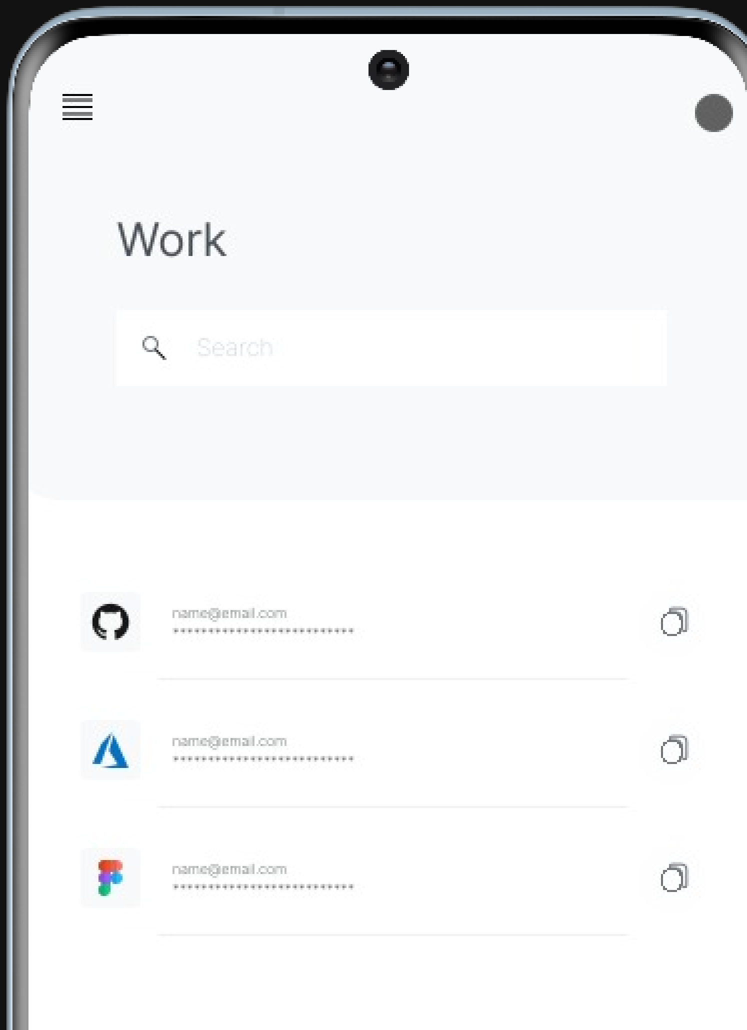
- Provides a client with a set of endpoints
- Allows the client to connect to an endpoint
- Takes the clients request to the server
- Lets the server processes the requests and create a response
- Takes the servers response and relays it to the client

Important Words

- Client
- Server
- Request
- Response

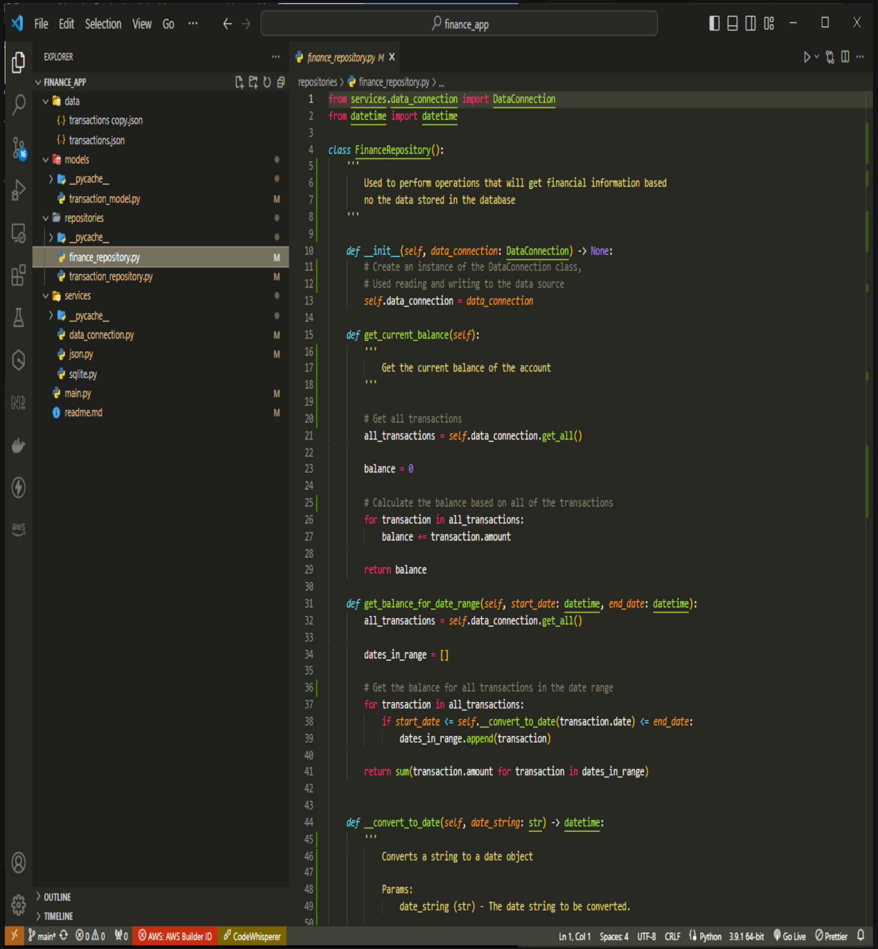
Client

- Usually the user facing application
  - Mobile app
  - Website
- Can also be a server connects to other services
- Sends a `request` to the API
- Performs operations based on the `response` of an API call



Server

- The application that processes requests
  - A backend Python application that does some soft of computation
  - A Python applicatoin that connects to a SQL database
- Produces the `response` that will be sent to the client
- Can be a client to other services



```python
from services.data_connection import DataConnection
from datetime import datetime

class FinanceRepository():
    '''
        Used to perform operations that will get financial information based
        no the data stored in the database
    '''

    def __init__(self, data_connection: DataConnection) -> None:
        # Create an instance of the DataConnection class,
        # Used reading and writing to the data source
        self.data_connection = data_connection

    def get_current_balance(self):
        '''
            Get the current balance of the account
        '''

        # Get all transactions
        all_transactions = self.data_connection.get_all()

        balance = 0

        # Calculate the balance based on all of the transactions
        for transaction in all_transactions:
            balance += transaction.amount

        return balance

    def get_balance_for_date_range(self, start_date: datetime, end_date: datetime):
        all_transactions = self.data_connection.get_all()

        dates_in_range = []

        # Get the balance for all transactions in the date range
        for transaction in all_transactions:
            if start_date <= self.__convert_to_date(transaction.date) <= end_date:
                dates_in_range.append(transaction)

        return sum(transaction.amount for transaction in dates_in_range)


    def __convert_to_date(self, date_string: str) -> datetime:
        '''
            Converts a string to a date object

            Params:
                date_string (str) - The date string to be converted.
```

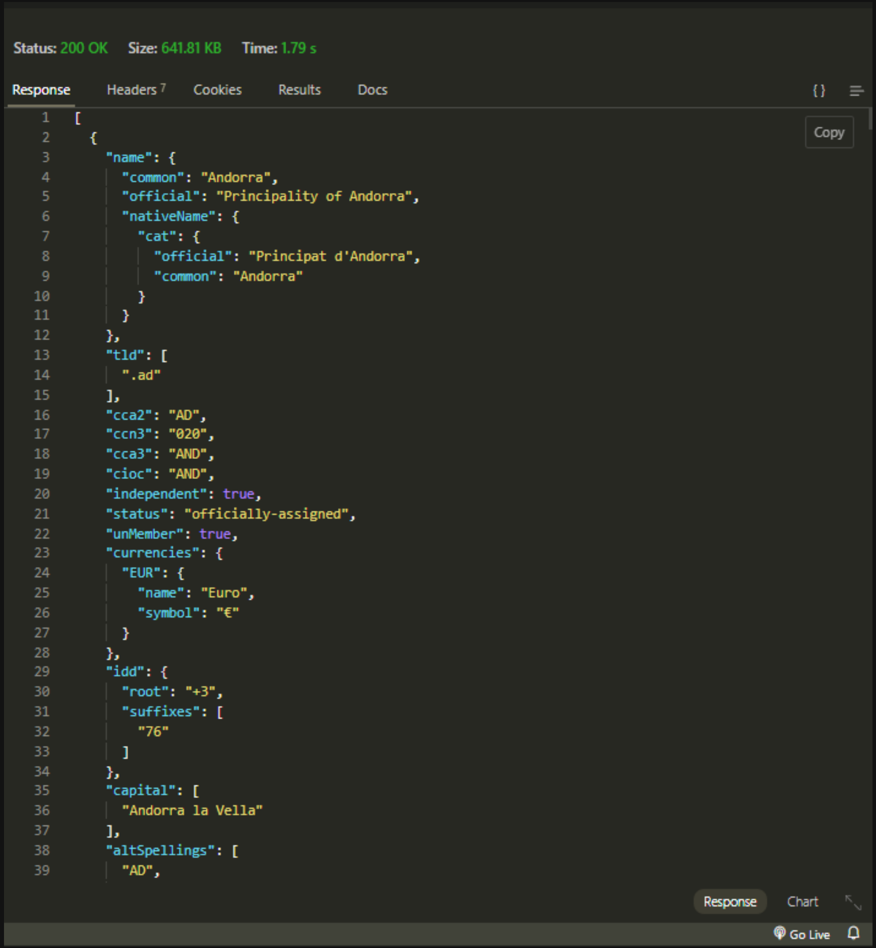
Request

- Sent by the `client`
- Contains key informations required to make an API call
  - URL - *web link*
  - Headers - *Information about the request*
  - Method - *Type of request being performed*
  - Params/Query - *Additional url paramters*
  - Body - *Data to be sent*

```
async loadProducts(search?: SearchQuery){
    await await axios({
      method: 'GET',
      baseURL: 'https://localhost:8000',
      url: 'api/products',
      params: {
        "limit": 20,
        "query": "graphics cards"
      }
    })
    .then((response) => {
      this.products = response.data.products;
    })
    .catch((error) => {
      alert("Error loading products")
    });
  }
```

Response

- Sent by the `server`
- Contains information about the processesd `request`
  - Headers - *Information about the response*
  - Status - *Outcomes of the request*
  - Body - *Data outputs*



```json
[
  {
    "name": {
      "common": "Andorra",
      "official": "Principality of Andorra",
      "nativeName": {
        "cat": {
          "official": "Principat d'Andorra",
          "common": "Andorra"
        }
      }
    },
    "tld": [
      ".ad"
    ],
    "cca2": "AD",
    "ccn3": "020",
    "cca3": "AND",
    "cioc": "AND",
    "independent": true,
    "status": "officially-assigned",
    "unMember": true,
    "currencies": {
      "EUR": {
        "name": "Euro",
        "symbol": "€"
      }
    },
    "idd": {
      "root": "+3",
      "suffixes": [
        "76"
      ]
    },
    "capital": [
      "Andorra la Vella"
    ],
    "altSpellings": [
      "AD",
```

Restaurant API

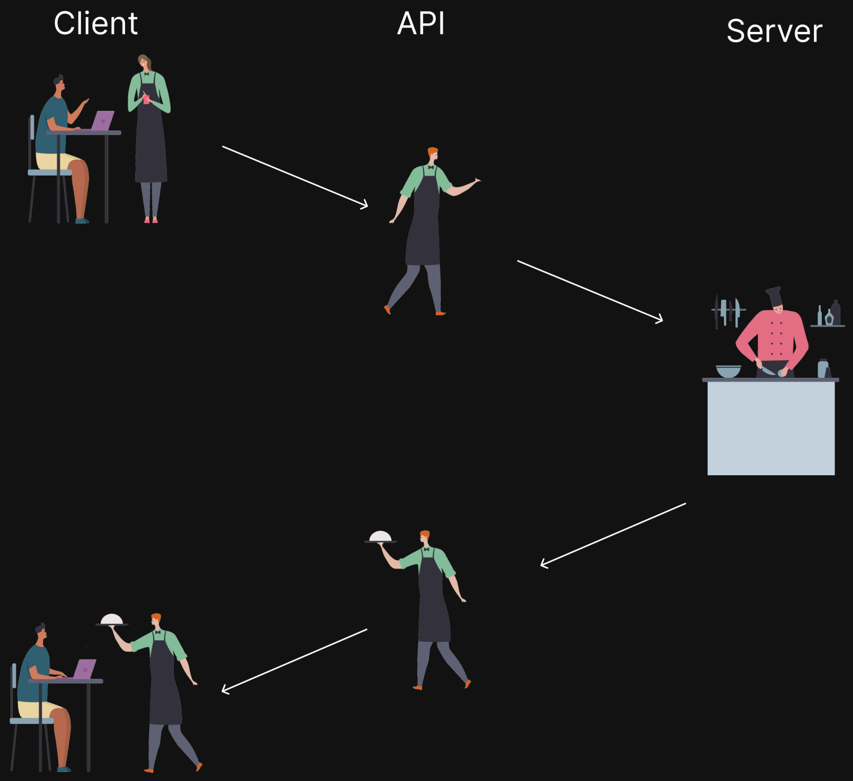


Image by macrovector

Roles

- Patron is the client
- Waiter is the API
- Chef is the Server

Key Details

- Client and server never talk directly
- Client doesn't know what the server does
- Client doesn't know where the server is
- Server doesn't know who the client is
- Server doesn't know what the client will do

Benefits of an API

- Abstraction
  - Client doesn’t care about the process, they just want the result
  - Server doesn’t care about what the client will do with the results
  - Client will never know the full depth of the process
  - Server can abstract its full complexity and hide details about itself
- Scalability
  - Client can grow independently
  - Server can grow independently
  - More servers can be dynamically deployed

Benefits for Software Engineers

Consuming APIs

- Allows you to work with large teams
- Independent scaling of your service
- Lets you focus on the important aspects of your application
- They are easier to swap out than native code

Producing APIs

- Lets you abstract information about your code
- Lets you scale your backend service independently
- Lets other people/teams use a defined set of operations from your service

Benefits for Data Scientists

- Lets you focus on analysis
- Gain access to a wider range of data sets
- Common means of receiving data in an engineering team
- Lets you access streaming data

More Resources

Types of APIs - YouTube Video

REST API - YouTube Video

Python and APIs - Blog

API Testing

Postman - Industry standard

Thunder Client - Lightweight VS Code Exension

JSON Formatter and Validator

JSON Formatter & Validator

Python Package

Requests

Installing Requests

Open the VS Code Terminal and run the following command

```
pip install requests
```

Disclaimer

Before working with an API, make sure you understand what you can and cannot do with the APIs, some APIs will require you to include branding on your apps or websites, some APIs don’t want you to save the data and so on.

It’s best to make sure that you work within the rules of the API

You must also be careful with services where you need to enter credit card information, if you go over your allocated limit, you might be charged per request going forward

Easy to use APIs

REST Countries

Random Data API

Pokeapi

Require Authentication

OpenWeather

Spoonacular

News API

API Sites

Public APIs - Curated

Rapid API - Marketplace (Be careful here)

LET’S GO!