# MLOPS

November 29, 2021

The **app process** is a combination of servers, APIs, operating systems, and databases that come together to power an app. What the end-user sees and with which he interacts has been established by coding interaction between the framework and the API.

*Visit a library to read a book. You are the app. The shelves, carts of books, and the building containing all this is the framework. The search for a book and the seat you take to read it is the API.*

A **library** is a group of code used to organize or simplify tasks. Constants, classes, functions, and procedures within a certain language reduce the need to write more code. The library is where you'll go to get information.

The **API** is the programming interface allowing you to interact with another application or operating system. It pulls from sources of code within the framework to work with other sources of code or applications of the code. The information pulled communicated through the API.

A **framework** is a generic structure that provides a skeleton architecture with which specific software can be implemented. The abstraction allows for common design patterns to be easily reused while still allowing the specific details to be left to the developers. Reusing common design patterns means having the general structure for solving similar sorts of problems.

**API** is the visible face of the app, while the **framework** and **library** are the inner workings of the app.

**REST** is an anagram for Representional State Trasnfer. *REST* is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web.What that means is an API will transfer to your API a representation of the state of the resource. As an example, your API contacts Facebook's API to request a user. REST forwards that user's name, profile, pictures, the number of friends and followers, posts, and much more. Your API might use REST API if you were an online shopping website like Shopify. Your end-user would sign in with Facebook thus giving your website the consumer's information.

**FastAPI** is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.

**Swagger** is an Interface Description Language for describing RESTful APIs expressed using JSON.

An **interface description language** or interface definition language (IDL), is a generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language. IDLs describe an interface in a language-independent way, enabling communication between software components that do not share one language, for example, between those written in C++ and those written in Java.

**The Web Server Gateway Interface** is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language. a **Calling convention** is an implementation-level (low-level) scheme for how subroutines receive parameters from their caller and how they return a result.

**ASGI** is a spiritual successor to WSGI, the long-standing Python standard for compatibility between web servers, frameworks, and applications.

*What's wrong with WSGI?* WSGI applications are a single, synchronous callable that takes a request and returns a response; this doesn't allow for long-lived connections, like you get with long-poll HTTP or WebSocket connections. WSGI applications are a single, synchronous callable that takes a request and returns a response; this doesn't allow for long-lived connections, like you get with long-poll HTTP or WebSocket connections. ASGI is structured as a single, asynchronous callable. It takes a scope, which is a dict containing details about the specific connection, send, an asynchronous callable, that lets the application send event messages to the client, and receive, an asynchronous callable which lets the application receive event messages from the client.

*SYNCHRONE*

```
| -------- A -------- |
                       |-------- B -------- |
```

*ASYNCHRONE*

```
| -------- A -------- |
      | -------- B -------- |
```