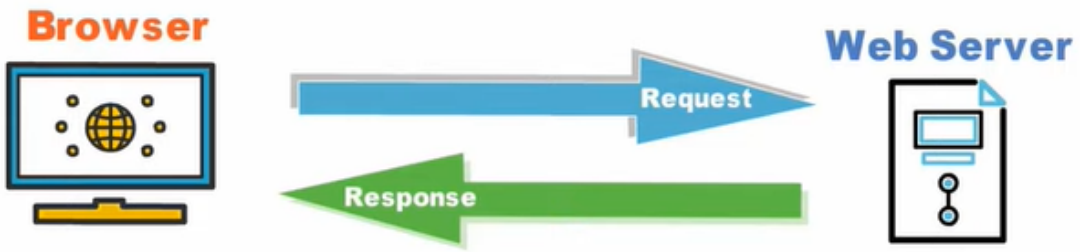
Giới thiệu: Hai phần Pydantic và Decorator sẽ được code trong quá trình sharing.
Trong **ứng dụng đặt hàng online**, thì bao gồm 2 thành phần chính: Client và Server.

- Client: Là những người dùng chuyên đưa ra yêu cầu **(Request)** và cần hệ thống xử lý yêu cầu, trả về kết quả **(Response)** Client mong muốn
- Server: Xử lý yêu cầu của Client và trả về kết quả.

Có thể hiểu: Dựa vào những nhu cầu của Client, coder sẽ xây dựng các hàm ở Server và cung cấp cho Client cách để gọi hàm và trả về kết quả như Client muốn

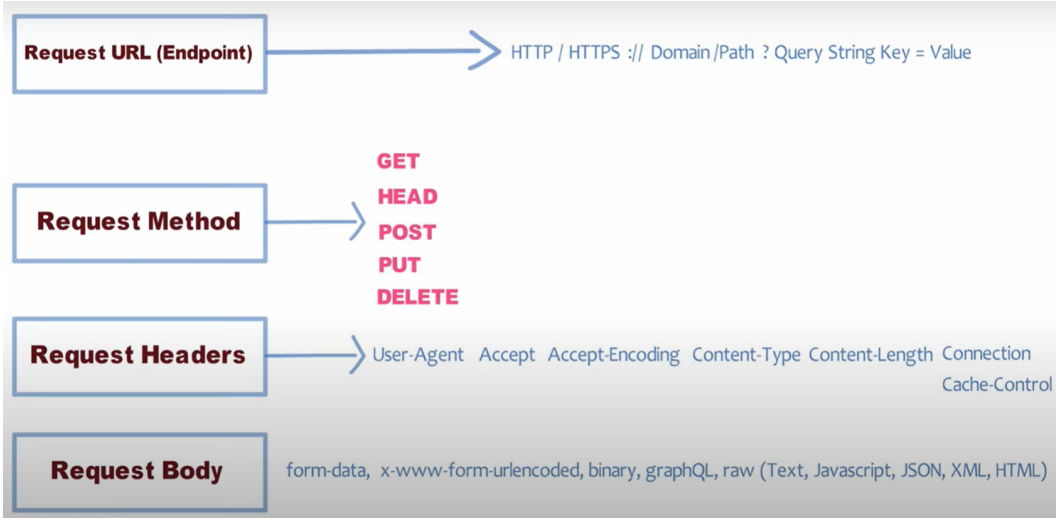
Phần 1. Lý thuyết về HTTP:

- HTTP là giao thức truyền tải siêu văn bản.
- Siêu văn bản là một văn bản chứa hyperlink để dẫn đến một trang siêu văn bản khác



- Client và Server giao tiếp với nhau qua từng HTTP Packet (gói HTTP).
- Trong HTTP Packet sẽ có 2 thành phần chính: HTTP Request, HTTP Response

Request:



- Request URL: Là đường dẫn mà client muốn request tới server, bao gồm:
  - HTTP/ : Tên giao thức request
  - Domain: Tên miền của server
  - Path: Tên API của server đó
  - Query: Là tên parameter truyền vào trên đường dẫn (optional)
- Request Method: Theo quy định của giao thức HTTP thì các phương thức trên có những tính chất sau:

	Request Body?	Response Body?	Safe	Idempotent	Cacheable
GET	Optional	Yes	Yes	Yes	Yes
HEAD	Optional	No	Yes	Yes	Yes
POST	Yes	Yes	No	No	Yes
PUT	Yes	Yes	No	Yes	No
DELETE	Optional	Yes	No	Yes	No

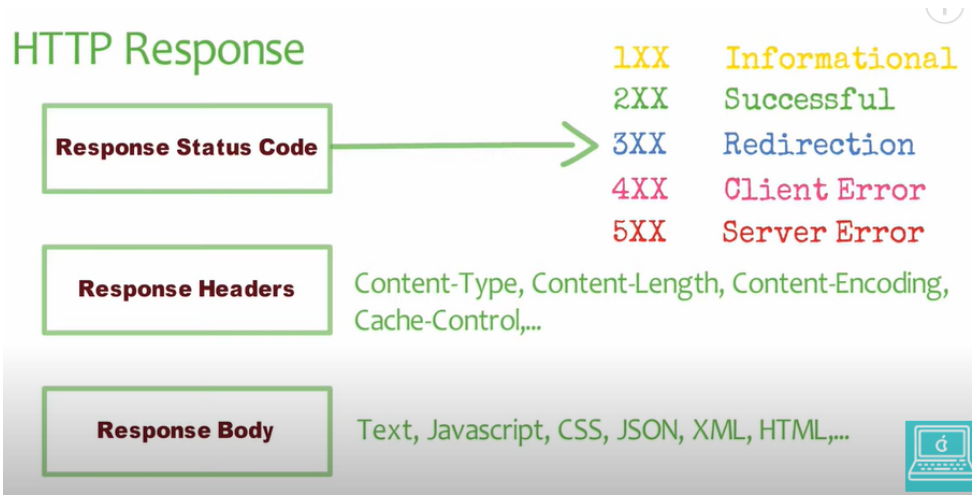
1. Request Body?: Là HTTP Request đó có đòi hỏi phải truyền một body request hay không?
2. Response Body?: HTTP Request đó có đòi hỏi server phải trả về một body response hay không?
3. Safe: HTTP Request đó có làm thay đổi data trong database hay không?
4. Idempotent: Khi request nhiều lần thì vẫn đảm bảo số lượng tài nguyên trong database không thay đổi
5. Cacheable: Có lưu cache lại

Và theo đó thì HTTP Protocol đã quy định các tính chất cho từng HTTP Method, từ đó người dùng lựa chọn HTTP Method cho từng chức năng của mình. Ví dụ:

1. METHOD GET: Dùng cho việc request một trang HTML từ server để show lên.
2. METHOD POST: Tạo mới thông tin trong database
3. METHOD PUT: Cập nhật thông tin trong database
4. METHOD DELETE: Xóa một trường thông tin trong database

- Request Header: Các thông tin của một gói HTTP, (sẽ giải thích nếu hỏi)
- Request Body: Truyền một file (text, json, javascript) tới server

Response:



- Response Status Code: Trạng thái trả về của server khi nhận được request từ client
- Response Header: Các thông tin của một gói HTTP, (sẽ giải thích nếu hỏi)
- Response Body: Trả về một file (text, HTML, JSON) tới client

Link youtube: https://www.youtube.com/watch?v=l1CnULyZLro

Phần 2. Một API trong FastAPI được code như thế nào?

- Một endpoint: gồm HTTP Method và đường dẫn

```python
@app.get("/")
def index():
    return {"name":"First Data"}
```

FastAPI dùng **decorator @** để chỉ định hàm index sẽ được gọi khi người dùng request Method GET tới đường dẫn "/"

- Request:
  - Parameter: Là parameter được ghi trên đường dẫn

```python
@app.get("/get-student")
def get_student(student_id:int):
    return students[student_id]
```

FastAPI cho phép đặt tham số ở hàm và sẽ tự biến nó về dạng parameter của đường dẫn -> /get-student?student_id=...

  - Body: Là một file json được gửi tới server

```python
class Student(BaseModel):
    name:str
    age:int
@app.put("/create-student")
def create_student(student_id:int, student:Student):
    students[student_id] = student
    return students
```

FastAPI dùng **Pydantic** để chuyển hóa từ một object sang một json text để gửi đi cho Client.

- Response:
  - Status code: Trả về status code và status message cho client biết

```python
from fastapi import FastAPI, HTTPException

app = FastAPI()

items = {"foo": "The Foo Wrestlers"}


@app.get("/items/{item_id}")
async def read_item(item_id: str):
    if item_id not in items:
        raise HTTPException(status_code=404, detail="Item not found")
    return {"item": items[item_id]}
```

  - Body: Trả về json body hoặc HTML.