




# FastAPI

# Deconstructed

~~~~~  
.....

>>>>>

Anatomy of a Modern ASGI Framework



# Hello, I'm Rafiqul Hasan

Lead Engineer @ bKash Limited

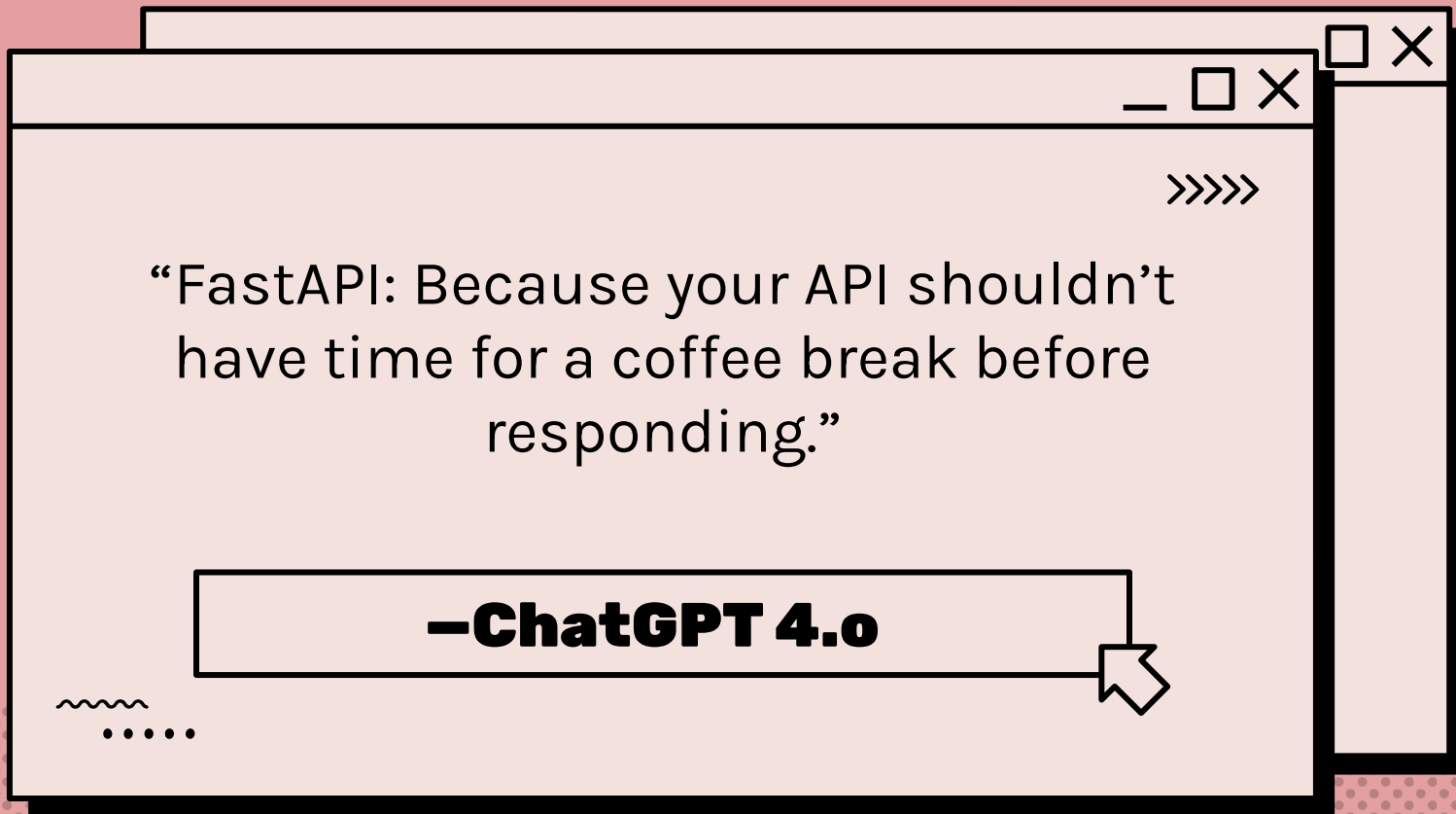
LinkedIn: [/in/rafiqulhasan](#)

Github: [/shopnilsazal](#)

X: [/shopnilsazal](#)

.....







# Table of contents



**01**

## Hello World

Let's start from the classic  
Hello World

**02**

## Components

Exploring the building  
blocks of FastAPI

**03**

## Journey

The lifecycle of a request

**04**

## The End

It might be a new  
beginning

# Hello World

.....



```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def hello(name: str = "World"):
    return {"message": f"Hello {name}!"}
```



# Run it

.....



```
$ uvicorn main:app
```

```
$ hypercorn main:app
```

```
$ granian --interface asgi main:app
```





# Components



## ASGI

The Protocol

## Uvicorn

ASGI Server

## Starlette

The Base App

## Pydantic

Data Validation

## Depends

Dependency Injection

## OpenAPI

Automatic API Doc

# Basic ASGI

.....





# ASGI Specification

>>>>

.....



## Scope

A dictionary containing connection's metadata.



## Events

- receive
- send



## Implementation

- Granian
- Hypercorn
- Uvicorn

# ASGI App

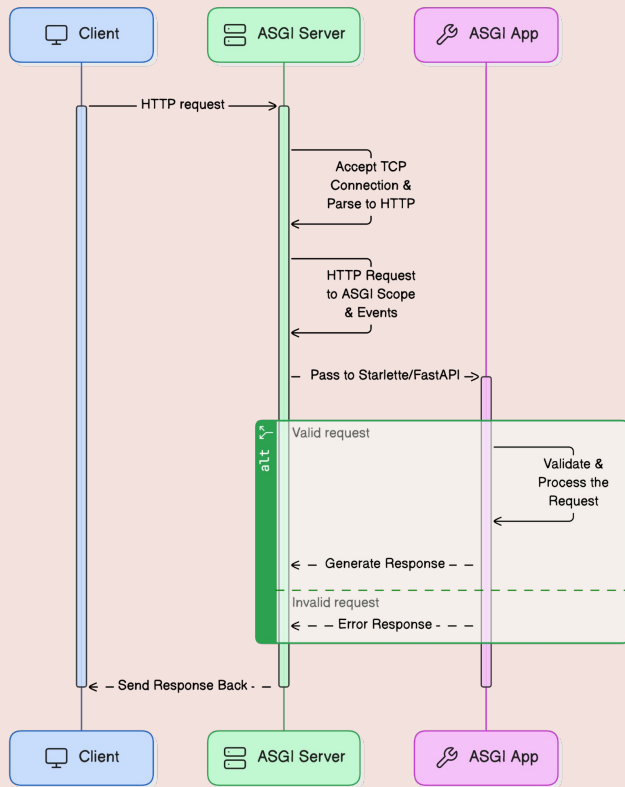
.....



```
async def app(scope, receive, send):  
    ...
```



# ASGI Flow



# Starlette

.....



```
from starlette.applications import Starlette
from starlette.responses import JSONResponse
from starlette.routing import Route

# A simple route handler
async def hello(request):
    return JSONResponse({'message': 'Hello, World!'})

# Defining the routes
routes = [
    Route('/hello', hello),
]

# Creating the Starlette app
app = Starlette(debug=True, routes=routes)
```



# ASGI Server

.....



```
$ uvicorn app:app --host 127.0.0.1 --port 8000
```



# CURL Request

.....

```
$ curl http://127.0.0.1:8000/hello
```



# HTTP Request

```
GET /hello HTTP/1.1  
Host: 127.0.0.1:8000  
User-Agent: curl/7.64.1  
Accept: */*
```



# ASGI Scope

.....



```
scope = {  
    "type": "http",  
    "http_version": "1.1",  
    "method": "GET",  
    "path": "/hello",  
    "query_string": b"",  
    "headers": [  
        (b"host", b"127.0.0.1:8000"),  
        (b"user-agent", b"curl/7.64.1"),  
        (b"accept", b"*/*"),  
    ],  
    "client": ("127.0.0.1", 12345),  
    "server": ("127.0.0.1", 8000),  
}
```





# Receive Event

```
request_event = {  
    "type": "http.request",  
    "body": b"", # Request body  
    "more_body": False, # Whether more data will be sent  
}
```



# ASGI Middleware

```
class ASGIMiddleware:
    def __init__(self, app: ASGIApp) → None:
        self.app = app

    async def __call__(self, scope, receive, send) → None:
        if scope["type"] ≠ "http":
            return await self.app(scope, receive, send)

        async def send_wrapper(message) → None:
            # ... Do something
            await send(message)

        await self.app(scope, receive, send_wrapper)
```



# Send Event

```
response_start_event = {  
    "type": "http.response.start",  
    "status": 200, # HTTP status code  
    "headers": [  
        (b"content-type", b"application/json"),  
    ],  
}
```



# Send Event

```
response_body_event = {  
    "type": "http.response.body",  
    "body": b'{"message": "Hello, World!"}',  
    "more_body": False,  
}
```



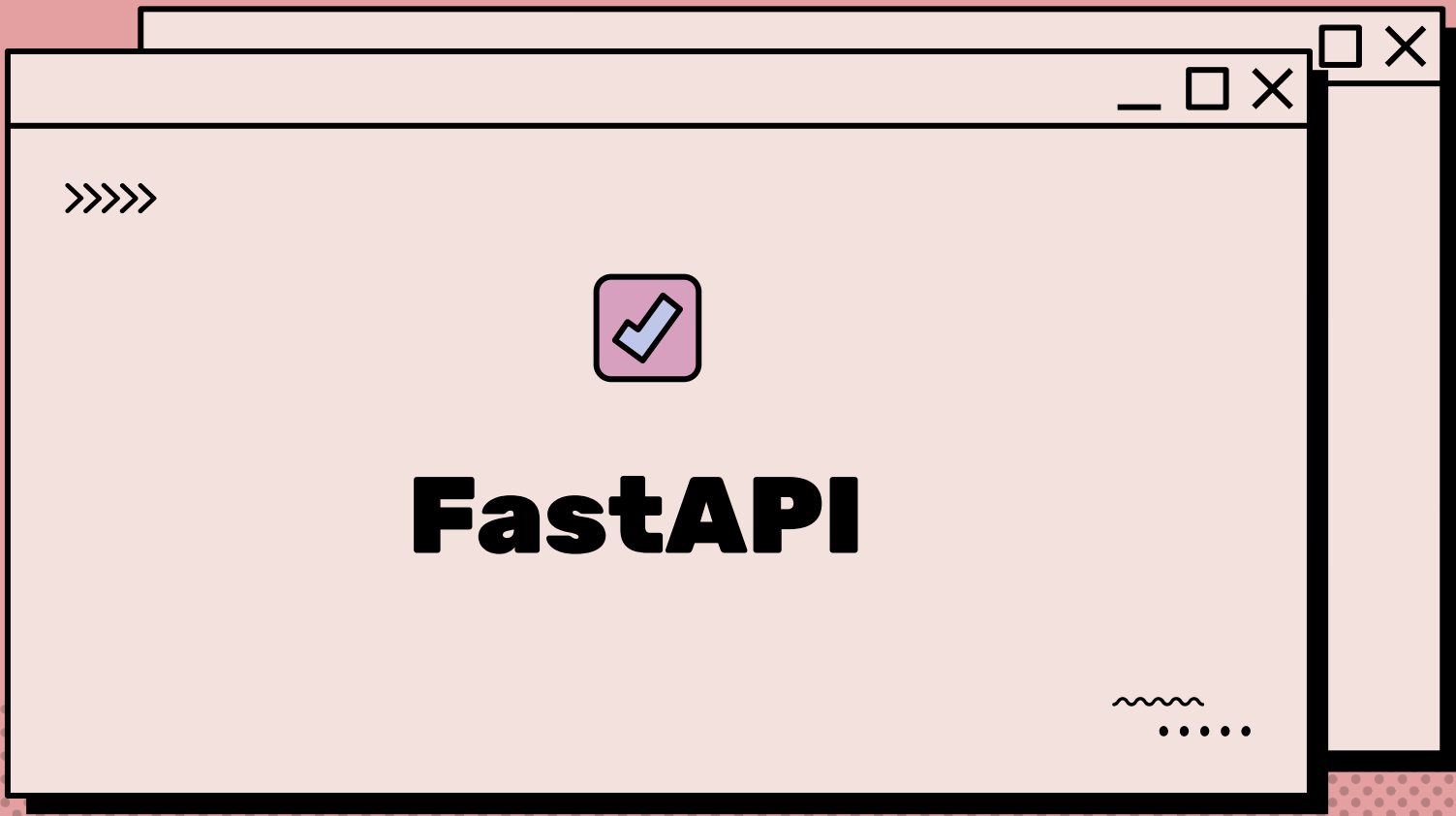
# Response

.....



```
{  
  "message": "Hello, World!"  
}
```





# Pydantic

.....



```
from fastapi import FastAPI, Depends
from pydantic import BaseModel

app = FastAPI()

# Define a Pydantic model for data validation
class Item(BaseModel):
    name: str
    price: float
    is_offer: bool = None

# Route with Pydantic validation and dependency injection
@app.post("/items/")
async def create_item(item: Item):
    return {
        "item": item,
        "db": db # injected dependency in the response
    }
```



# Request

.....



```
$ curl -X POST "<http://127.0.0.1:8000/items/>" -H  
"Content-Type: application/json" -d '{"name":  
"Table", "price": 150.0}'
```





# Response

.....



```
{  
  "item": {  
    "name": "Table",  
    "price": 150.0,  
    "is_offer": null  
  }  
}
```



# Error Response

```
{
  "detail": [
    {
      "loc": ["body", "name"],
      "msg": "field required",
      "type": "value_error.missing"
    }
  ]
}
```



# Depends

.....



```
from fastapi import Depends

# Define a dependency
async def get_db():
    db = {"connection": "db-connection"}
    try:
        yield db
    finally:
        # Cleanup logic, like closing the DB connection
        pass

# Use the dependency in a route
@app.get("/items/")
async def read_items(db=Depends(get_db)):
    return {"db": db}
```



# API Doc

.....



GET

/hello Hello

▼

POST

/items Create Item

▲

Parameters

Try it out

No parameters

Request body required

application/json ▼

Example Value | Schema

```
{  "name": "string",  "price": 0,  "is_offer": true}
```

Responses

| Code | Description                                                                                                                                                                        | Links    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| 200  | Successful Response <div><div>Media type</div><div>application/json ▼</div><div>Controls Accept header.</div><div>Example Value   Schema</div><div><pre>"string"</pre></div></div> | No links |
| 422  | Validation Error <div><div>Media type</div><div>application/json ▼</div></div>                                                                                                     | No links |



# API Doc

Search...

GET Hello

POST Create Item

## Hello

### QUERY PARAMETERS

name string (Name)  
Default: "World"

### Responses

> 200 Successful Response

> 422 Validation Error

## Create Item

REQUEST BODY SCHEMA: application/json

name string (Name)  
price number (Price)  
is\_offer boolean (Is Offer)

### Responses

> 200 Successful Response

> 422 Validation Error

....



GET /hello

### Response samples

200 422

Content type  
application/json

null

Copy

POST /items

### Request samples

Payload

Content type  
application/json

```
{
  "name": "string",
  "price": 0,
  "is_offer": true
}
```

Copy

### Response samples

200 422

Content type  
application/json



# Summary



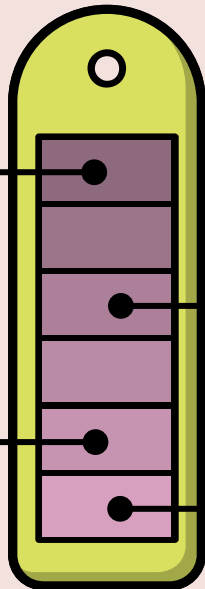
....

## ASGI

ASGI is the backbone of modern Python web frameworks.

## Starlette

Starlette is the core web framework handling routing and middlewares.



## Uvicorn

Uvicorn is your fast, async ASGI server.

## FastAPI

FastAPI extends Starlette with data validation via Pydantic, dependency injection & auto docs.



# Thanks!

Does anyone have any questions?

shopnilsazal@gmail.com  
rafiqul.dev

**CREDITS:** This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**