

# FAST API FROM ZERO TO HERO

SEAMOS FAST HAGAMOS API

PYTHON CORUÑA 2023  
CAPÍTULO I



P Y T H O N   C O R U Ñ A   2 0 2 3

# ROBERTO GARCÍA

**BACKEND SOFTWARE ENGINEER & ARCHITECT**  
WALLBOX

**HEAD BACKEND & CO-FOUNDER**  
TELEBOWLING

**BACKEND TECH LEAD**  
IP GLOBAL

**HEAD BACKEND ENGINEER**  
VEGA GATE VR



# CONTENIDO

- ⬡ ASINCRONÍA
- ⬡ FAST API
- ⬡ CARACTERÍSTICAS
- ⬡ RECURSOS
- ⬡ SIGUIENTES PASOS

P Y T H O N   C O R U Ñ A   2 0 2 3



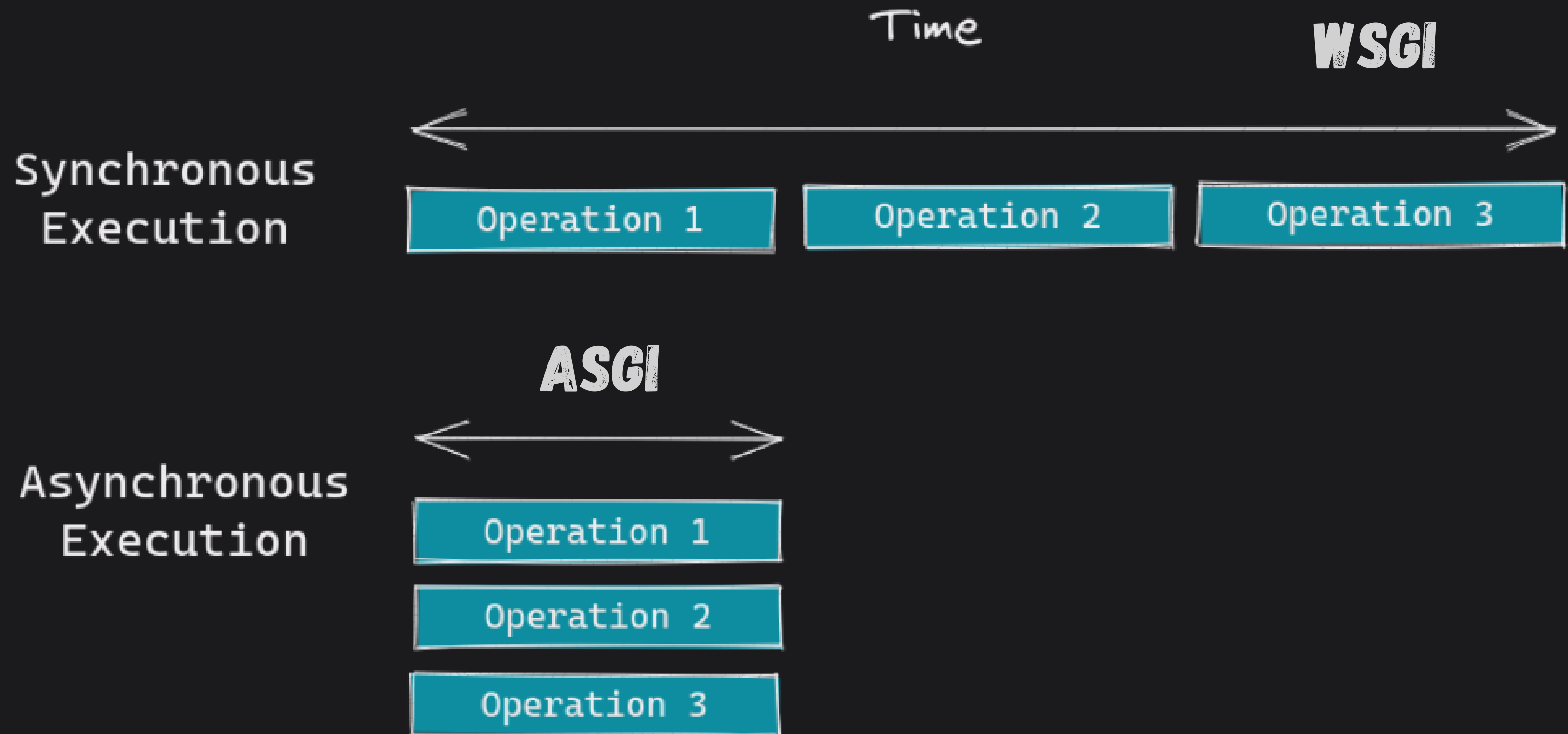
# INTRODUCCIÓN



# ¿QUE ES LA ASINCRONÍA?

“El diseño de software asíncrono amplía el concepto al construir código que permita a un programa solicitar que una tarea se realice al mismo tiempo que la tarea (o tareas) original, sin detenerse a esperar a que la primera se haya completado.

Cuando la tarea secundaria se completa, la original es notificada usando un mecanismo acordado, de tal forma que sepa que el trabajo se ha completado y que el resultado, si es que existe, está disponible.”



P Y T H O N   C O R U Ñ A   2 0 2 3



# ASYNCIO





**“Librería cuyo único fin es permitir al usuario escribir código concurrente haciendo uso de la sintaxis `async / await` la cual fue incorporada de forma nativa desde la versión ~3.5 de Python.”**

Python documentation

⬡ COROUTINE

⬡ TASK

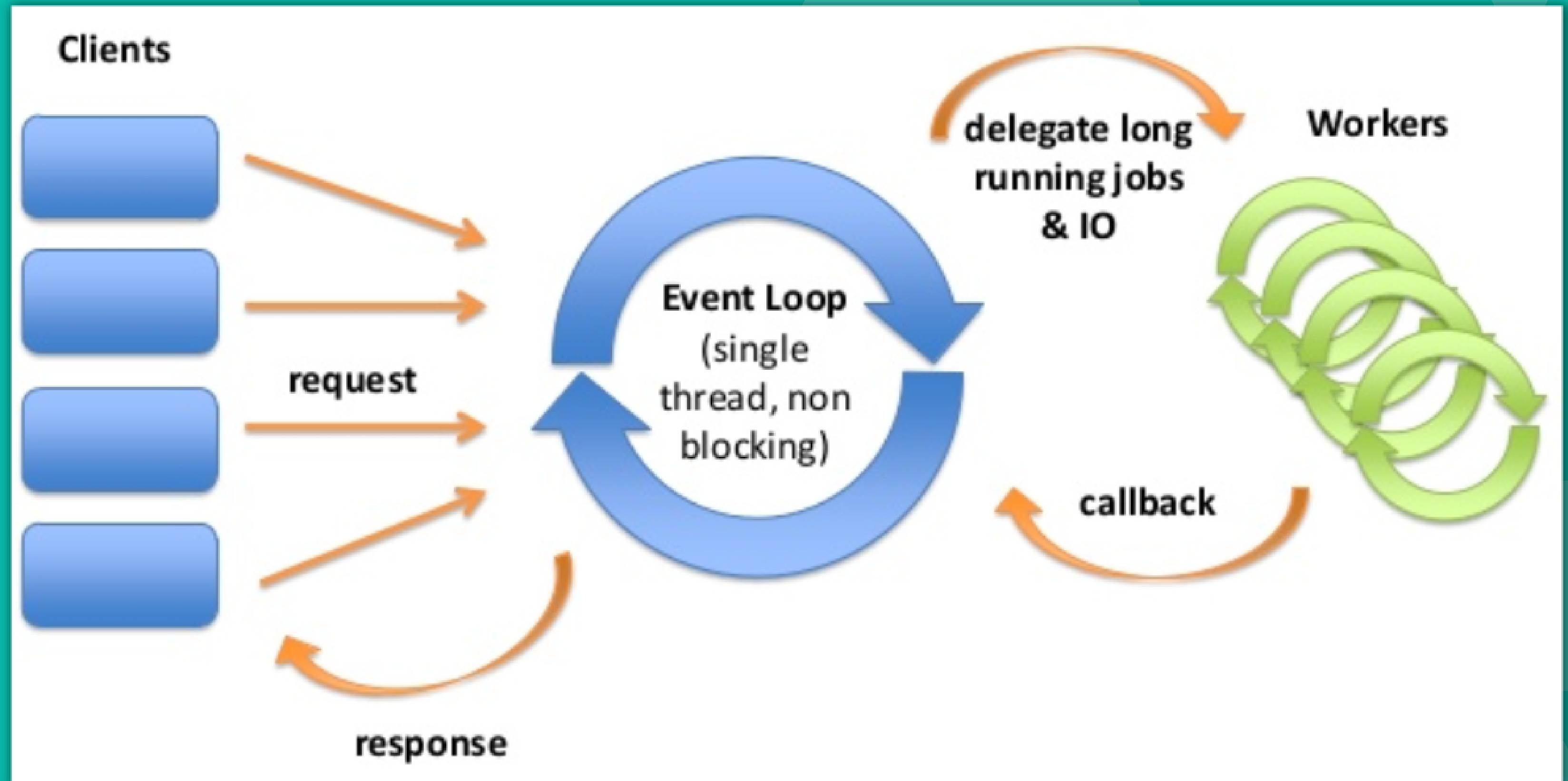
⬡ AWAITABLE

⬡ FUTURE

asyncio







P Y T H O N   C O R U Ñ A   2 0 2 3



# COROUTINES / AWAITABLE





## CLASES

```
import asyncio

# Function
async def awaitable_function():
    await asyncio.sleep(1)
    return 'Hello, Python Coruña!'

# Generator
async def awaitable_generator():
    for i in range(3):
        await asyncio.sleep(1)
        yield i
```

## FUNCIONES

```
import asyncio
from typing import List

class CartItem:
    def __init__(self, price: float):
        self.price = price

class Cart:
    def __init__(self, items: List[CartItem]):
        self.items = items

    async def __aiter__(self) -> CartItem:
        for idx, cartItem in enumerate(self.items):
            await asyncio.sleep(1)
            yield cartItem
```



**“Use async IO when you can; use threading when you must.”**



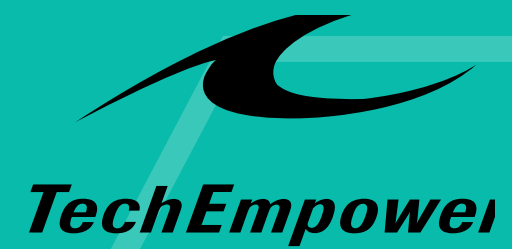
# FAST API

**“Is a modern, high-performance web framework for building APIs with Python based on standard type hints fully centered in concurrent proccessing with asyncio out-of-the-box.”**

 Sebastián Ramírez

 tiangolo

# PERFORMANCE



- ⬡ **Principalmente ideado para el desarrollo de REST API**
- ⬡ **Usa Starlette (HTTP) e integra Pydantic (Model / DTO)**
- ⬡ **Es un micro-framework rápido y ligero**
- ⬡ **Intuitivo, robusto y escalable**
- ⬡ **Abraza los estándares de la industria**
- ⬡ **Librería para construir apps de CLI ~ Typer**
- ⬡ **Curva de aprendizaje aceptable**
- ⬡ **Interacción con asyncio out-of-the-box**



# REQUERIMIENTOS

🔶 PYTHON 3.7+

🔶 PIP

🔶 PIPENV \*

🔶 POETRY \*

\* Recomendaciones

```
bash

# Full installation way
pip install "fastapi[all]" # FastAPI + uvicorn

# Step by step way
pip install fastapi # FastAPI
pip install "uvicorn[standard]"
```



P Y T H O N C O R U Ñ A 2 0 2 3

# HELLO WORLD

python

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/welcome")
async def root():
    return {"message": "Hello Python Coruña"}
```

```
$ uvicorn main:app --reload
```

```
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [28720]
INFO: Started server process [28722]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

P Y T H O N   C O R U Ñ A   2 0 2 3



# CARACTERÍSTICAS





# SIGUIENTES PASOS



# SIGUIENTES PASOS


- ⬡ SEAMOS FAST, HAGAMOS API - MEETUP
- ⬡ HEXAGONALMENTE HABLANDO - MEETUP
- ⬡ MV ¿QUE? - DE MV\* A HEXAGONAL - WORKSHOP
- ⬡ LET'S GO TO PRODUCTION - WORKSHOP



# RECURSOS








## Secure access for **everyone** But not just **anyone**

**Secure access for everyone. But not just anyone.**

Rapidly integrate authentication and authorization for web, mobile, and legacy applications so you can focus on your core business.

 Auth0

robcharlwood/**asyncio-playground**



A little place for me to play with asyncio in Python 3.7

1

Contributor
 

0

Issues
 

0

Stars
 

0

Forks

---

**robcharlwood/asyncio-playground: A little place for me to play with asyncio in Python 3.7**

A little place for me to play with asyncio in Python 3.7 - GitHub - robcharlwood/asyncio-playground: A little place for me to play with asyncio in Python 3.7

 GitHub

python-injector/**injector**



Python dependency injection framework, inspired by Guice

23

Contributors
 

1k

Used by
 

917

Stars
 

71

Forks

---


**python-injector/injector: Python dependency injection framework, inspired by Guice**

Python dependency injection framework, inspired by Guice - GitHub - python-injector/injector: Python dependency injection framework, inspired by Guice

 GitHub


**HTTPX**

A next-generation HTTP client for Python.


 python-httpx.org

**Punk API: Brewdog's DIY Dog as an API**

Brewdog's DIY Dog as a searchable, filterable API

 punkapi.com

psincraian/**commandbus**



Command Bus Pattern in Python

1

Contributor
 

9

Used by
 

20

Stars
 

1

Fork

---

**psincraian/commandbus: Command Bus Pattern in Python**

Command Bus Pattern in Python. Contribute to psincraian/commandbus development by creating an



**Asynchronous Context Managers in Python**

You can create and use asynchronous context managers in asyncio programs by defining an object that implements the `__aenter__()` and `__aexit__()`...



## MagicStack/uvloop

Ultra fast asyncio event loop.



58 Contributors 72k Used by 9k Stars 540 Forks



### MagicStack/uvloop: Ultra fast asyncio event loop.

Ultra fast asyncio event loop. Contribute to MagicStack/uvloop development by creating an account on GitHub.

GitHub

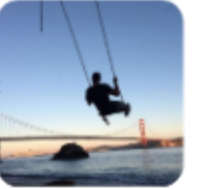


### ets-labs/python-dependency-injector: Dependency injection framework for Python

Dependency injection framework for Python. Contribute to ets-labs/python-dependency-injector development by creating an account on GitHub.

GitHub

## timofurrer/awesome-asyncio



A curated list of awesome Python asyncio frameworks, libraries, software and resources

54 Contributors 0 Issues 4k Stars 290 Forks



### timofurrer/awesome-asyncio: A curated list of awesome Python asyncio frameworks, libraries, software and resources

A curated list of awesome Python asyncio frameworks, libraries, software and resources - GitHub - timofurrer/awesome-asyncio: A curated list of awesome Python asyncio frameworks, libraries, softwar...

GitHub

## pypa/pipenv

Python Development Workflow for Humans.



402 Contributors 72k Used by 25 Discussions 24k Stars 2k Forks



### pypa/pipenv: Python Development Workflow for Humans.

Python Development Workflow for Humans. Contribute to pypa/pipenv development by creating an account on GitHub.

GitHub

## python-poetry/poetry

Python packaging and dependency management made easy



460 Contributors 601 Issues 377 Discussions 24k Stars 2k Forks



### python-poetry/poetry: Python packaging and dependency management made easy

Python packaging and dependency management made easy - GitHub - python-poetry/poetry: Python packaging and dependency management made easy

GitHub

## typer

Build great CLIs. Easy to code. Based on Python type hints.

### tiangolo/typer: Typer, build great CLIs. Easy to code. Based on Python type hints.

Typer, build great CLIs. Easy to code. Based on Python type hints. - GitHub - tiangolo/typer: Typer, build great CLIs. Easy to code. Based on Python type hints.

GitHub

