Live de Python #52

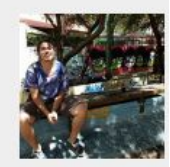
Usando Threads - Multiprocessamento #2

APOIE O CANAL
apoia.se/livedepython









Muito obrigado <3

Edimar Fardim
Eliabe Silva
Fabiano Teichmann
João Lugão
Maria boladona
Paulo Tadei
Segis Santos
Sérigo Passos
Willian Lopes

1º Sorteio da Live de Python

https://goo.gl/forms/wiNGNZaXboZ5GxC12

Nome:
Eduardo Mendes

Instituição:
Unicamp / Diebold Nixdorf

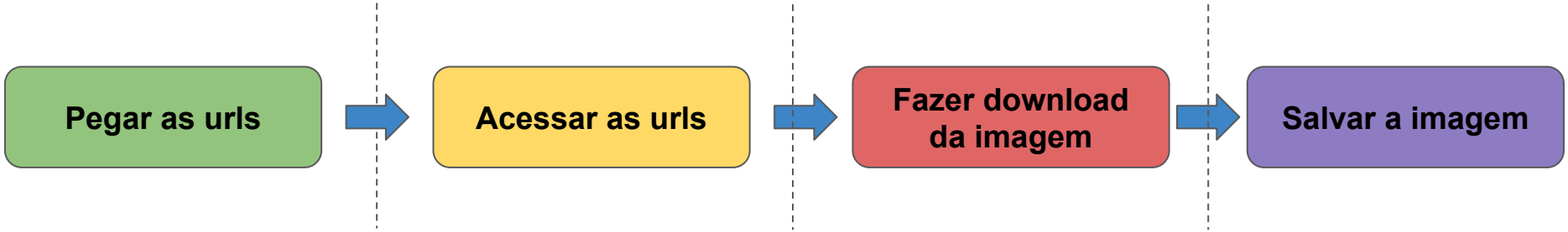
Contatos:
{facebook, github, gist
instagram, linkedin,
telegram, twitter}/dunossauro

Roteiro

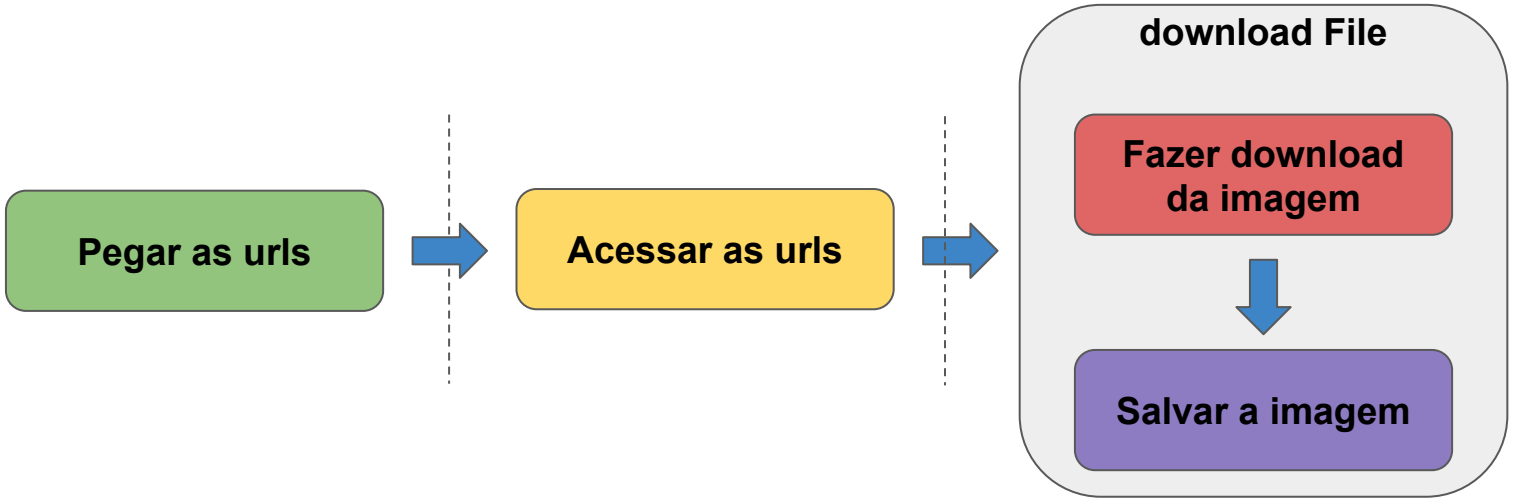
- Uma breve lembrança sobre a live 51
- Analisando nossa solução
- Projetando uma nova solução
  - Decomposição de dados
  - Queue e eventos
  - Threads
  - Pool

Relembrando o problema

Fazer o download de um sprite dos primeiros 100 pokémons da pokeapi



Analisando nossa solução



Analisando nossa solução

Pegar as urls

```python
def download_file(name, url, *, path=path, type_='png'):
    """Faz o download de um arquivo."""
    response = get(url, stream=True)
    fname = f'{path}/{name}.{type_}'
    with open(fname, 'wb') as f:
        copyfileobj(response.raw, f)
    return fname


def get_sprite_url(url, sprite='front_default'):
    return get(url).json()['sprites'][sprite]


with timeit() as t:
    pokemons = get(urljoin(base_url, 'pokemon/?limit=100')).json()['results']
    images_url = {j['name']: get_sprite_url(j['url']) for j in pokemons}
    files = [download_file(name, url) for name, url in images_url.items()]
```

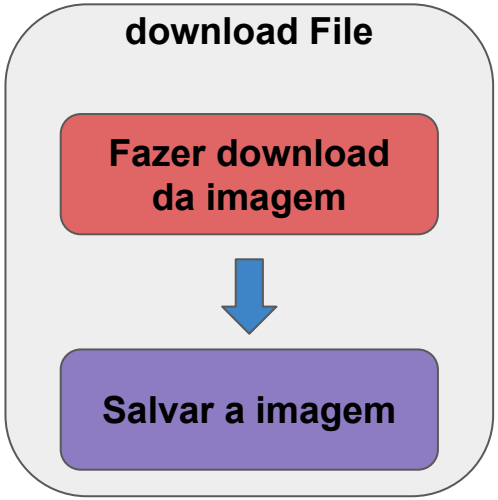
Analisando nossa solução

Acessar as urls

```
def download_file(name, url, *, path=path, type_='png'):
    """Faz o download de um arquivo."""
    response = get(url, stream=True)
    fname = f'{path}/{name}.{type_}'
    with open(fname, 'wb') as f:
        copyfileobj(response.raw, f)
    return fname


def get_sprite_url(url, sprite='front_default'):
    return get(url).json()['sprites'][sprite]


with timeit() as t:
    pokemons = get(urljoin(base_url, 'pokemon/?limit=100')).json()['results']
    images_url = {j['name']: get_sprite_url(j['url']) for j in pokemons}
    files = [download_file(name, url) for name, url in images_url.items()]
```

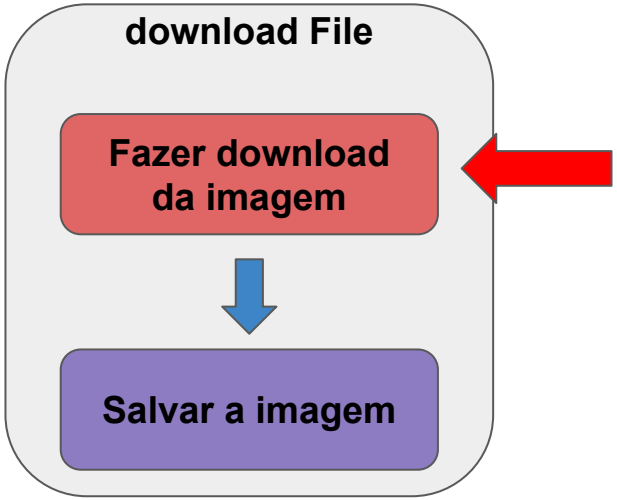
Analisando nossa solução

```python
def download_file(name, url, *, path=path, type_='png'):
    """Faz o download de um arquivo."""
    response = get(url, stream=True)
    fname = f'{path}/{name}.{type_}'
    with open(fname, 'wb') as f:
        copyfileobj(response.raw, f)
    return fname
```



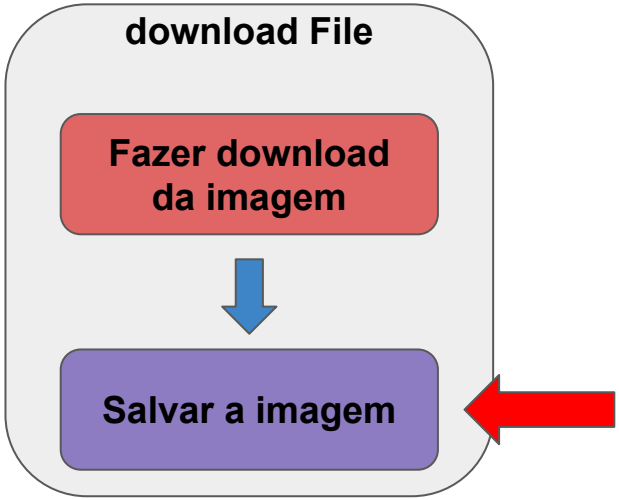
Analisando nossa solução

```python
def download_file(name, url, *, path=path, type_='png'):
    """Faz o download de um arquivo."""
    response = get(url, stream=True)
    fname = f'{path}/{name}.{type_}'
    with open(fname, 'wb') as f:
        copyfileobj(response.raw, f)
    return fname
```



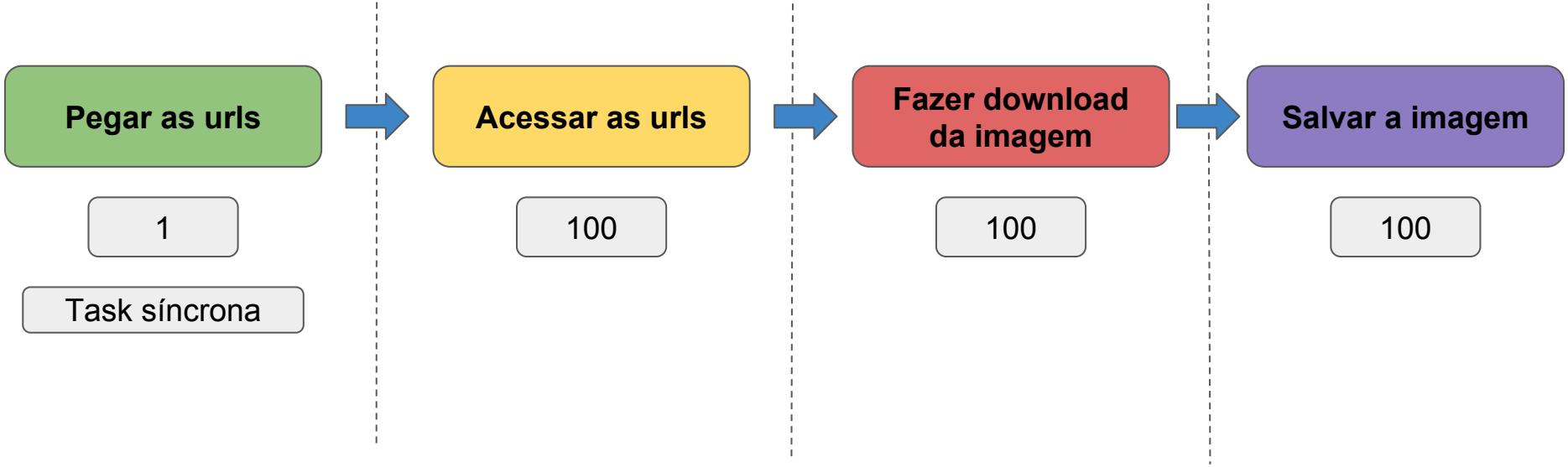
Analisando nossa solução

```python
def download_file(name, url, *, path=path, type_='png'):
    """Faz o download de um arquivo."""
    response = get(url, stream=True)
    fname = f'{path}/{name}.{type_}'
    with open(fname, 'wb') as f:
        copyfileobj(response.raw, f)
    return fname
```



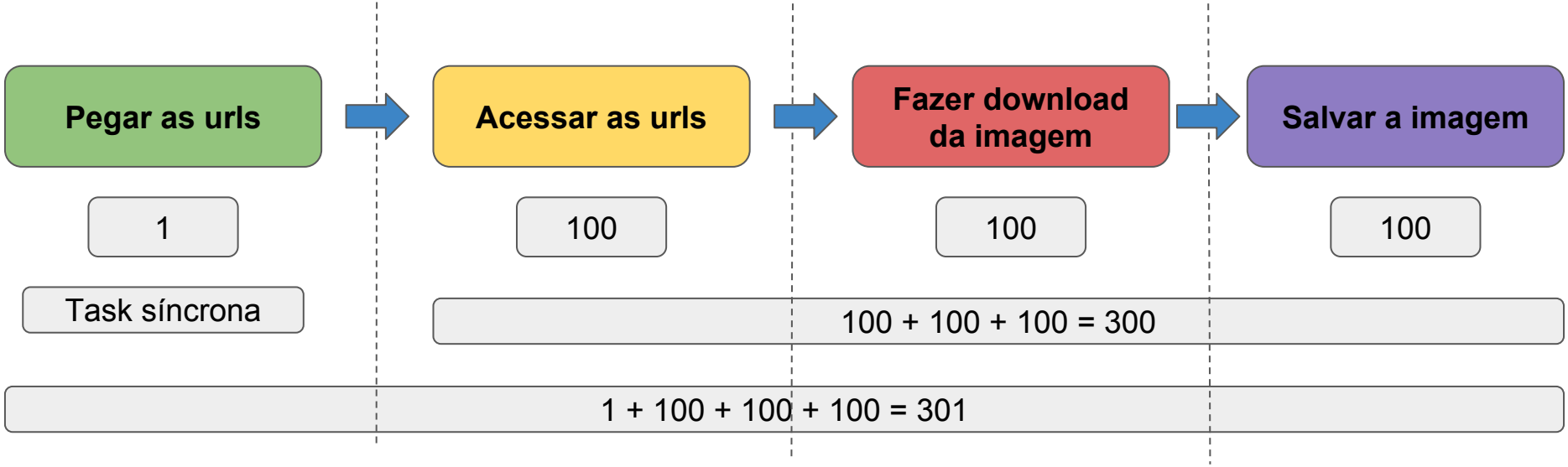
Decomposição

A decomposição inside em pegar um gargalo do processamento e decompor ele em tarefas menores. Vamos chamar tarefas de ‘tasks’ e coisas que resolvem as tasks de workers.



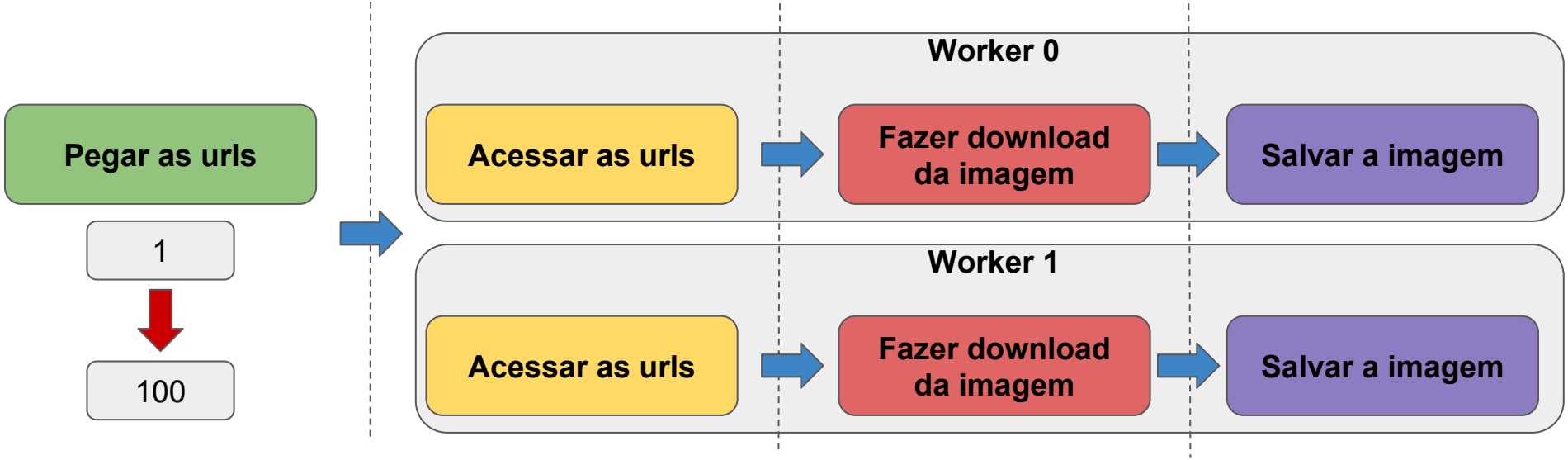
Decomposição

A decomposição inside em pegar um gargalo do processamento e decompor ele em tarefas menores. Vamos chamar tarefas de ‘tasks’ e coisas que resolvem as tasks de workers.

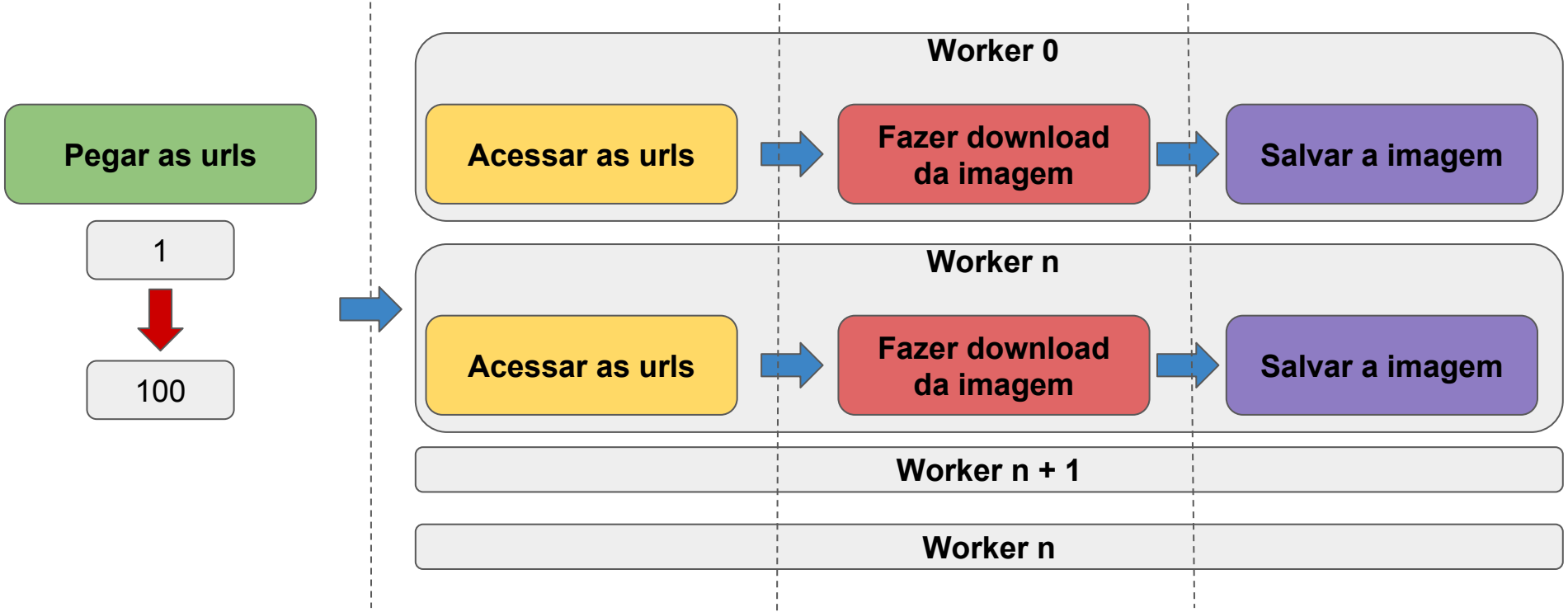


Decomposição

A decomposição inside em pegar um gargalo do processamento e decompor ele em tarefas menores. Vamos chamar tarefas de ‘tasks’ e coisas que resolvem as tasks de workers.

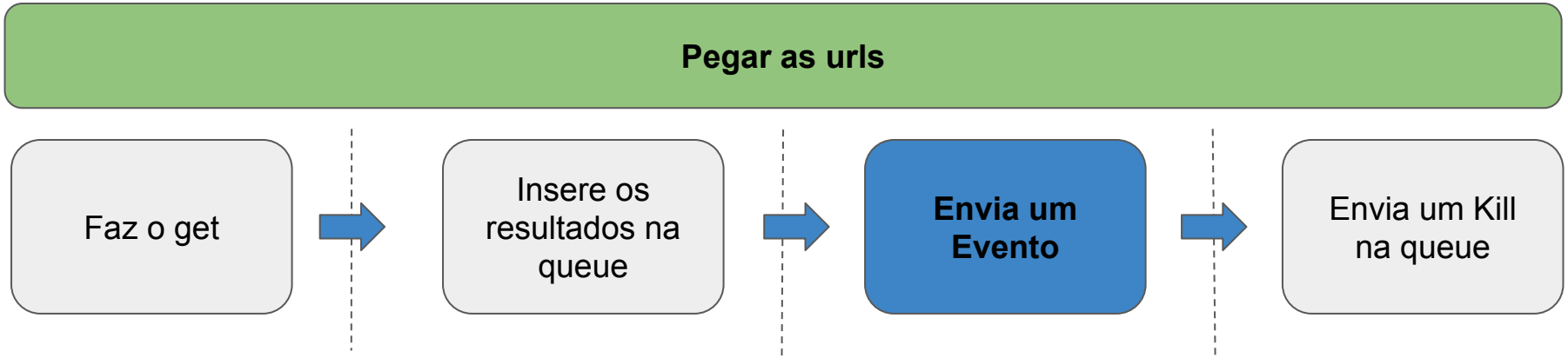


Decomposição



Queue e Evento

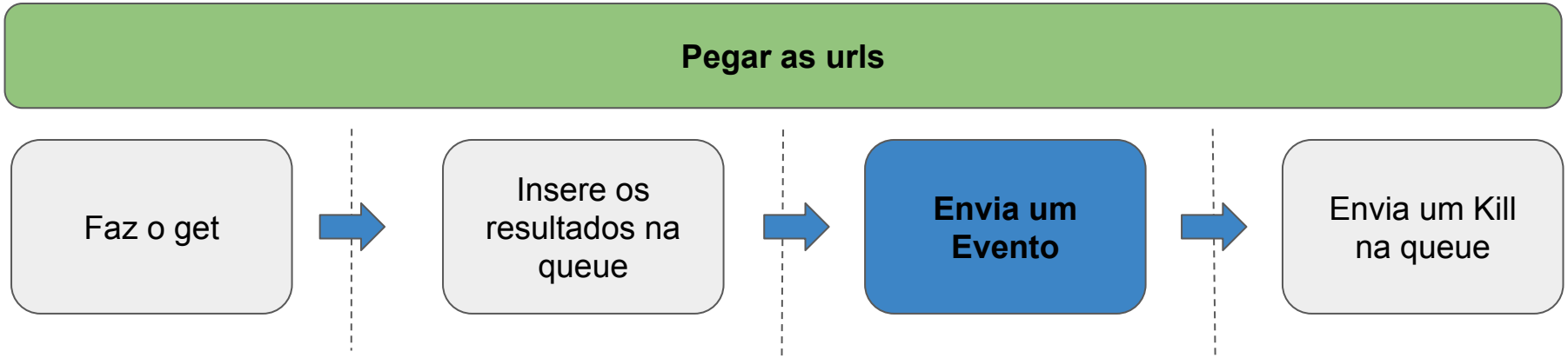
- Um evento é algo que avisa o worker que ele pode iniciar a resolver a task.
- Queue é uma fila onde os works vão consumir suas tasks



Queue e Evento

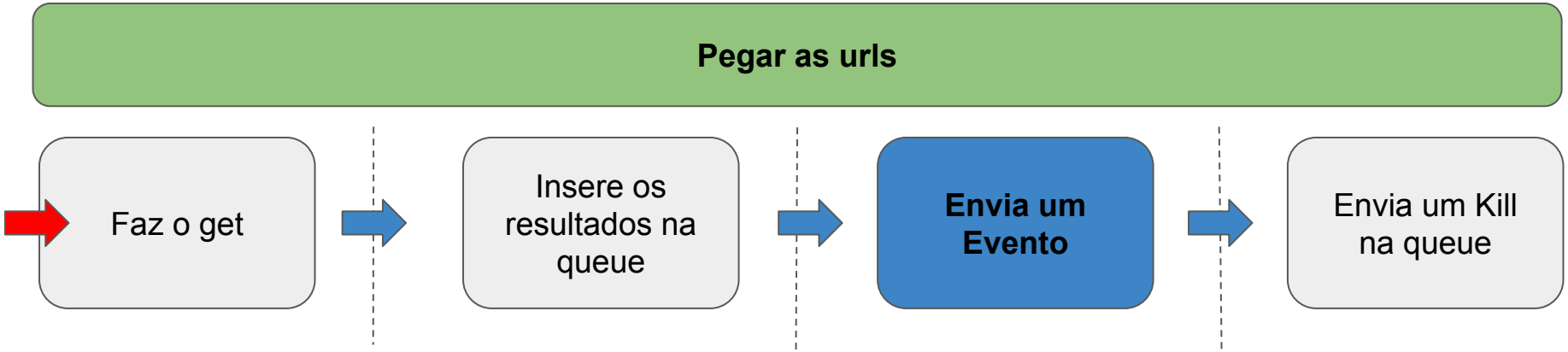
```python
from threading import Event
from queue import Queue

event = Event()
fila = Queue(maxsize=101)
```

- Um evento é algo que avisa o worker que ele pode iniciar a resolver a task.
- Queue é uma fila onde os works vão consumir suas tasks



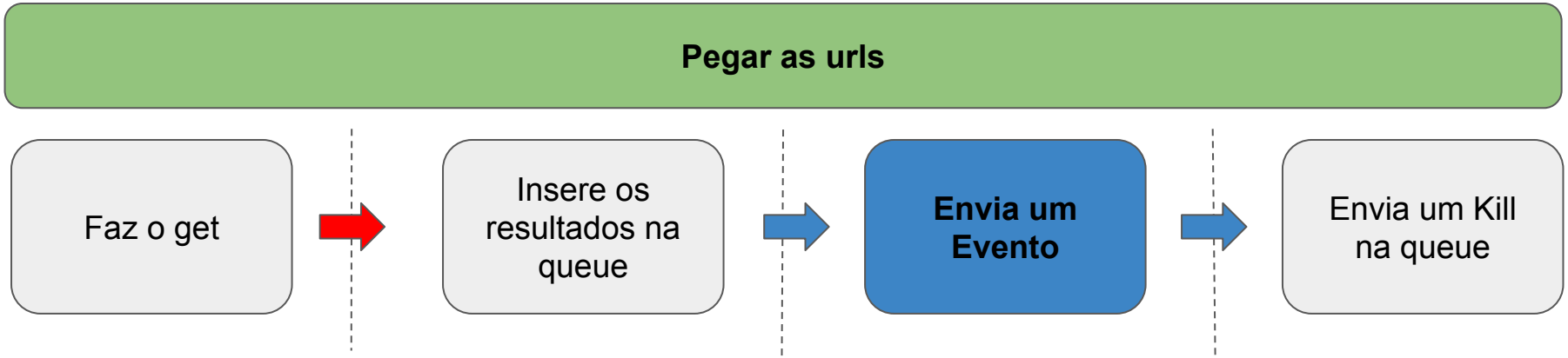
Queue e Evento

```
def get_urls():
    pokemons = get(urljoin(base_url, 'pokemon/?limit=100')).json()['results']
    [fila.put(pokemon) for pokemon in pokemons]
    event.set()
    fila.put('Kill')
```



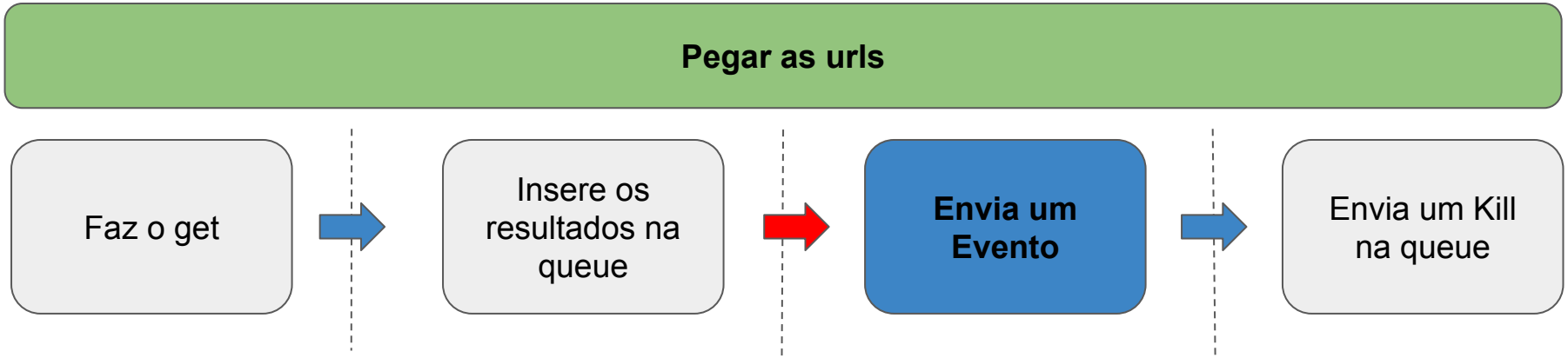
Queue e Evento

```
def get_urls():
    pokemons = get(urljoin(base_url, 'pokemon/?limit=100')).json()['results']
    [fila.put(pokemon) for pokemon in pokemons]
    event.set()
    fila.put('Kill')
```



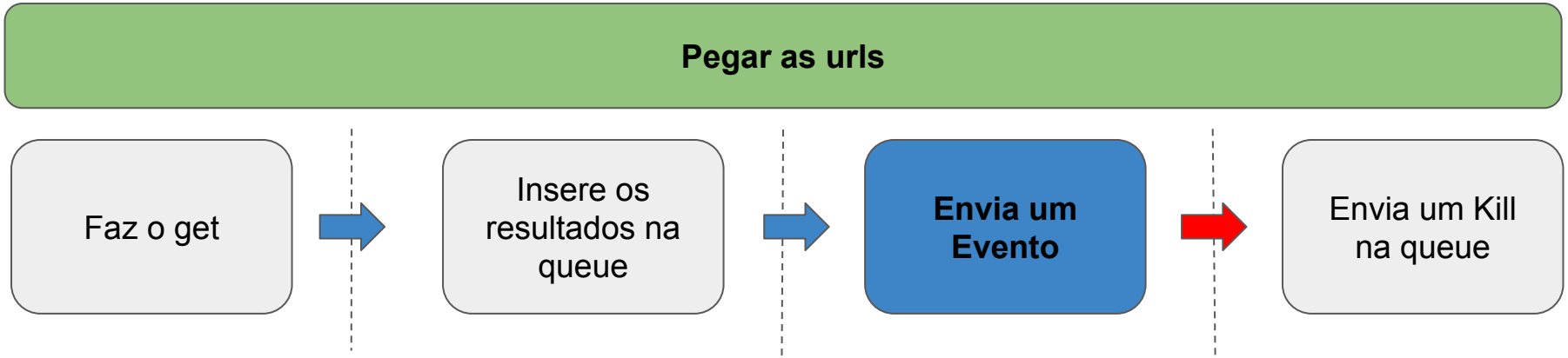
Queue e Evento

```
def get_urls():
    pokemons = get(urljoin(base_url, 'pokemon/?limit=100')).json()['results']
    [fila.put(pokemon) for pokemon in pokemons]
    event.set()
    fila.put('Kill')
```

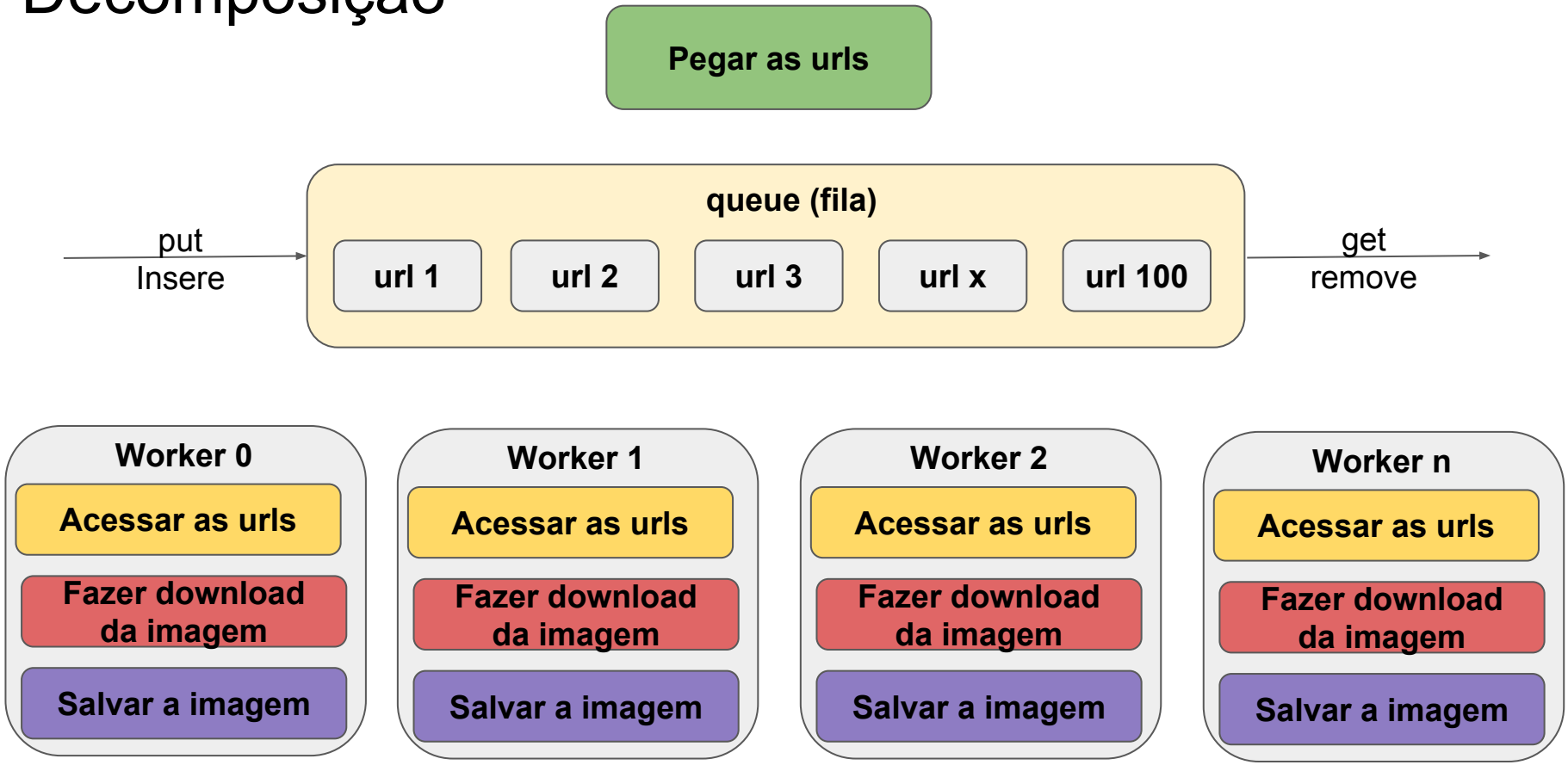


Queue e Evento

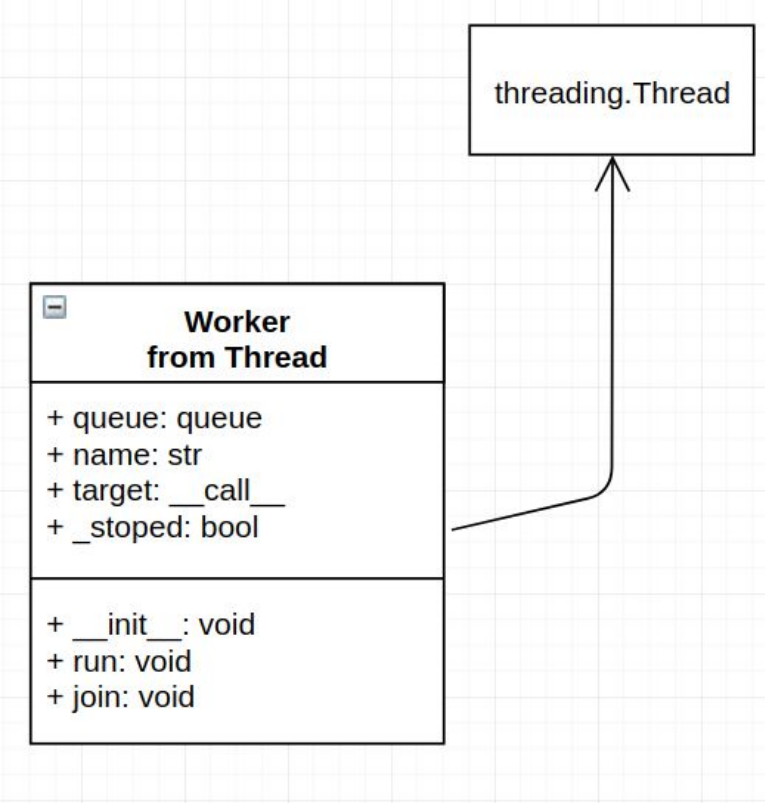
```
def get_urls():
    pokemons = get(urljoin(base_url, 'pokemon/?limit=100')).json()['results']
    [fila.put(pokemon) for pokemon in pokemons]
    event.set()
    fila.put('Kill')
```



Decomposição

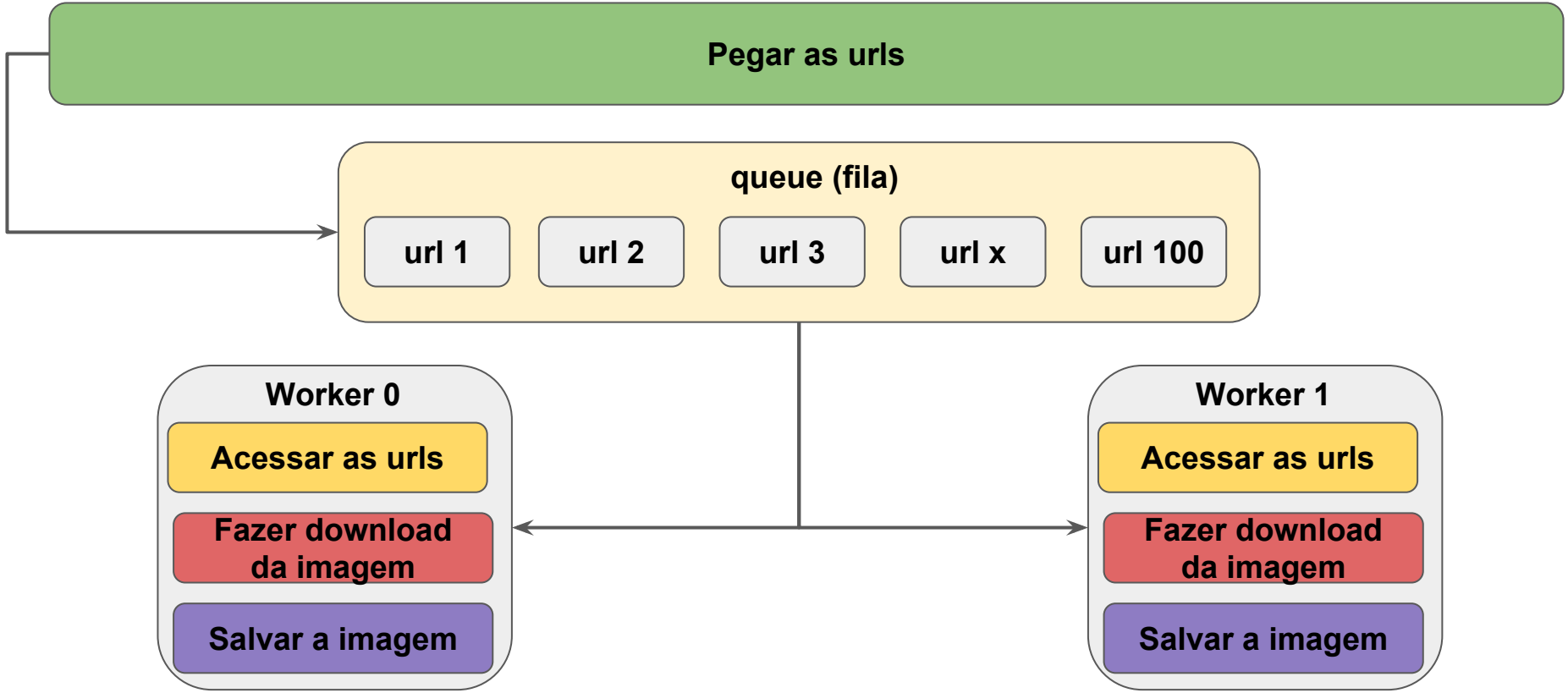


Construindo um worker

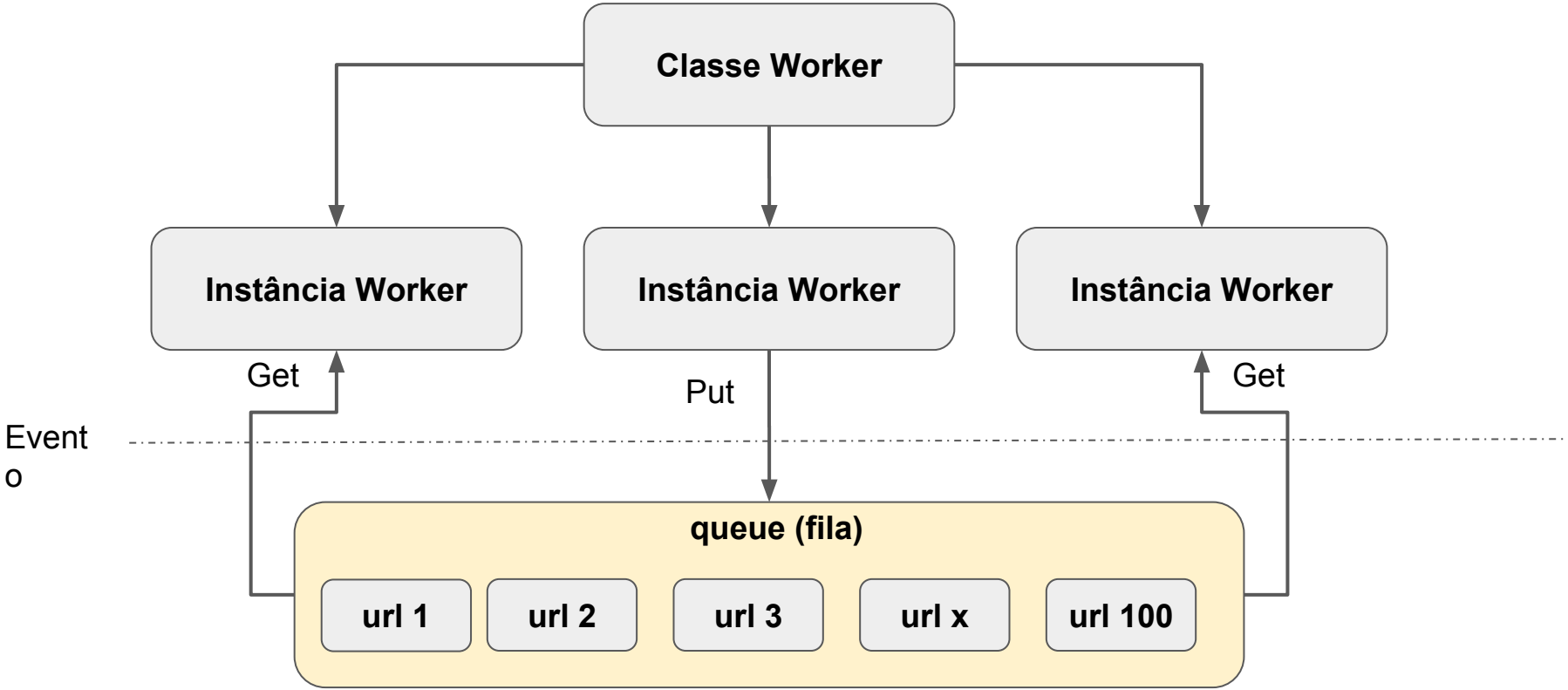



```python
class Worker(Thread):
    def __init__(self, target, queue, *, name):
        super().__init__()
        self.name = name
        self.queue = queue
        self._target = target
        self._stoped = False

    def run(self):
        event.wait()

    def join(self):
        while not self._stoped:
            sleep(0.1)
```

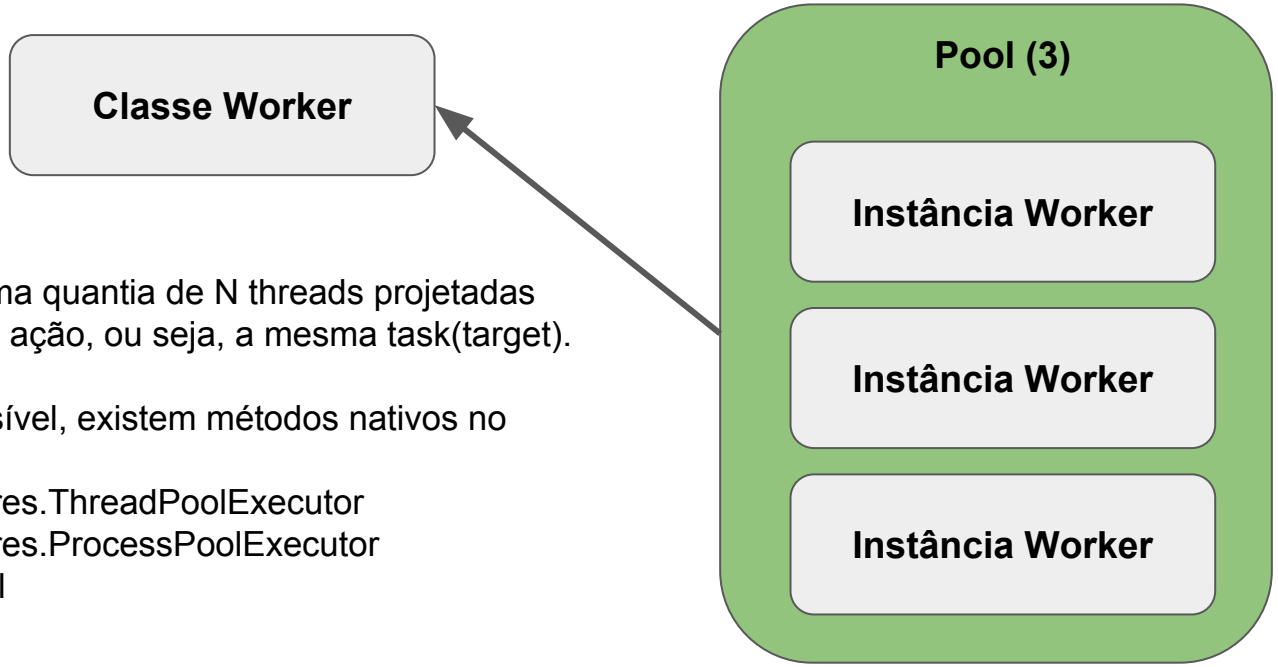
Entendendo o esquema



Entendendo o esquema



Pool de Threads

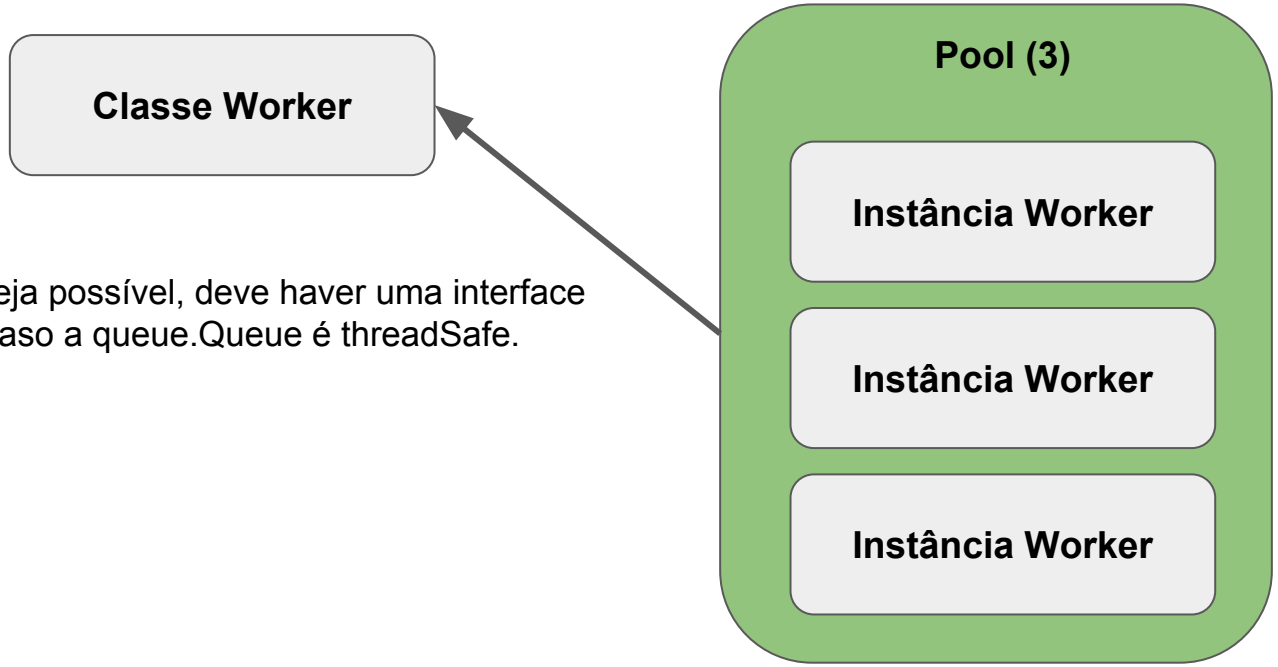


Pools de Thread são uma quantia de N threads projetadas para executar a mesma ação, ou seja, a mesma task(target).

Para que isso seja possível, existem métodos nativos no Python, como:

- concurrence.futures.ThreadPoolExecutor
- concurrence.futures.ProcessPoolExecutor
- multiprocess.Pool

Pool de Threads



Porém, para que isso seja possível, deve haver uma interface threadSafe. No nosso caso a queue.Queue é threadSafe.

XOXO

Dúvidas?

Nome:
Eduardo Mendes

Instituição:
Unicamp / Diebold Nixdorf

Contatos:
{facebook, github, gist
instagram, linkedin,
telegram, twitter}/dunossauro