Serviço de Sincronização de Pastas

Projeto de LPIII - Desenvolvido por: Victor Bitencourt e Isabel Nunes

Objetivo principal

Sincronização Unidirecional: Desenvolver um serviço que sincroniza automaticamente arquivos de uma pasta no servidor para uma pasta no cliente.

Passo a passo

Windows

> Criação e instalação dos pacotes da venv

`python3 -m venv venv` //criação da venv

`venv\Scripts\activate` //ativar a venv

`pip install -r requirements.txt` //instalar requerimentos

python3 main.py // executar programa

Linux

> Criação e instalação dos pacotes da venv

`python3 -m venv venv` //criação da venv

`venv\bin\activate` //ativar a venv

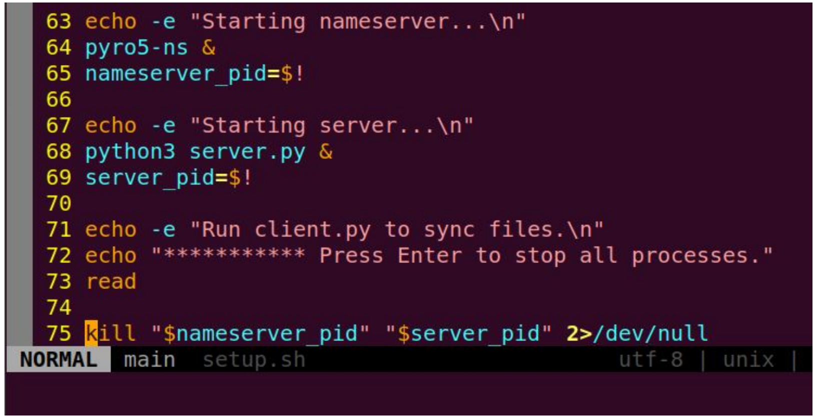
`pip install -r requirements.txt` //instalar requerimentos

python3 main.py // executar programa

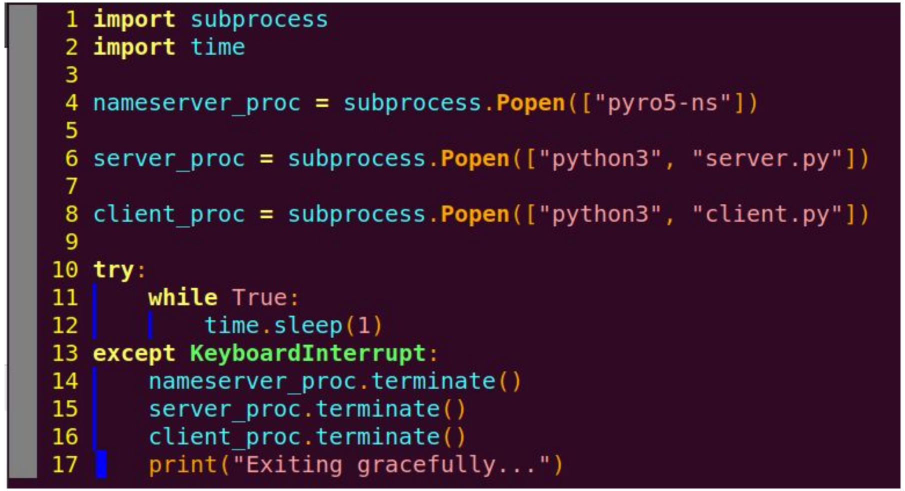
Passo a passo

- Popular pasta ./source-folder.
- Arquivos, pastas e subpastas estarão espelhados na pasta ./mirror-folder.

Setup: antes vs. depois



```
echo -e "Starting nameserver...\n"
pyro5-ns &
nameserver_pid=$!

echo -e "Starting server...\n"
python3 server.py &
server_pid=$!

echo -e "Run client.py to sync files.\n"
echo "*********** Press Enter to stop all processes."
read

kill "$nameserver_pid" "$server_pid" 2>/dev/null
NORMAL  main  setup.sh                    utf-8 | unix |
```



```
import subprocess
import time

nameserver_proc = subprocess.Popen(["pyro5-ns"])

server_proc = subprocess.Popen(["python3", "server.py"])

client_proc = subprocess.Popen(["python3", "client.py"])

try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    nameserver_proc.terminate()
    server_proc.terminate()
    client_proc.terminate()
    print("Exiting gracefully...")
```

Tecnologias utilizadas

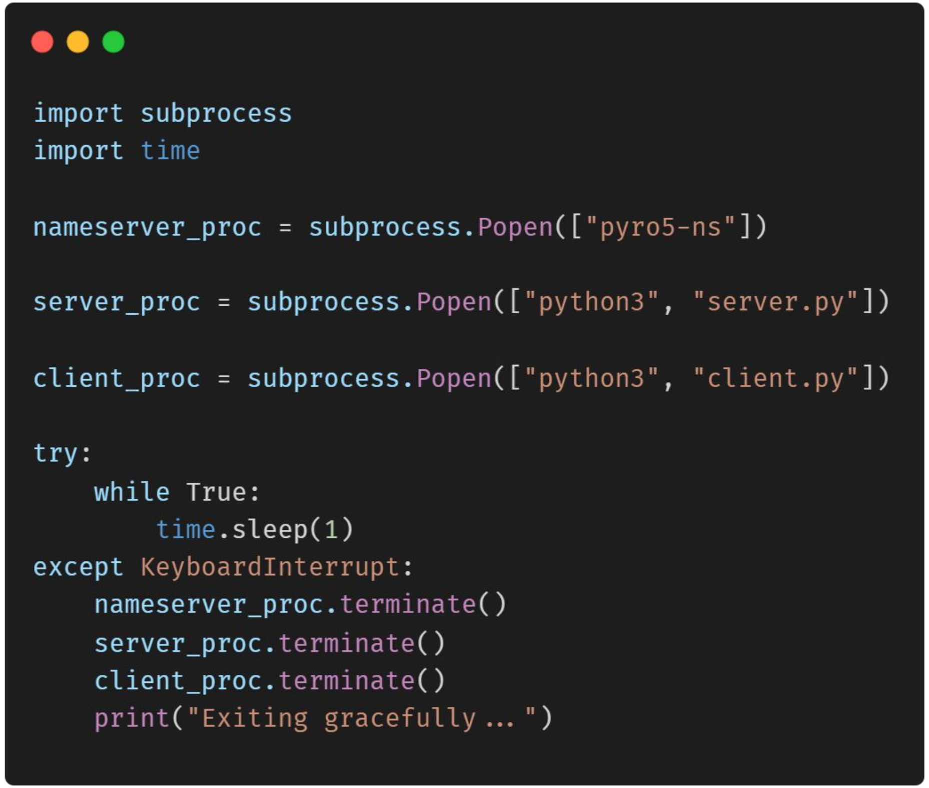
Pyro5 v5.15

Serpent v1.41

Módulos

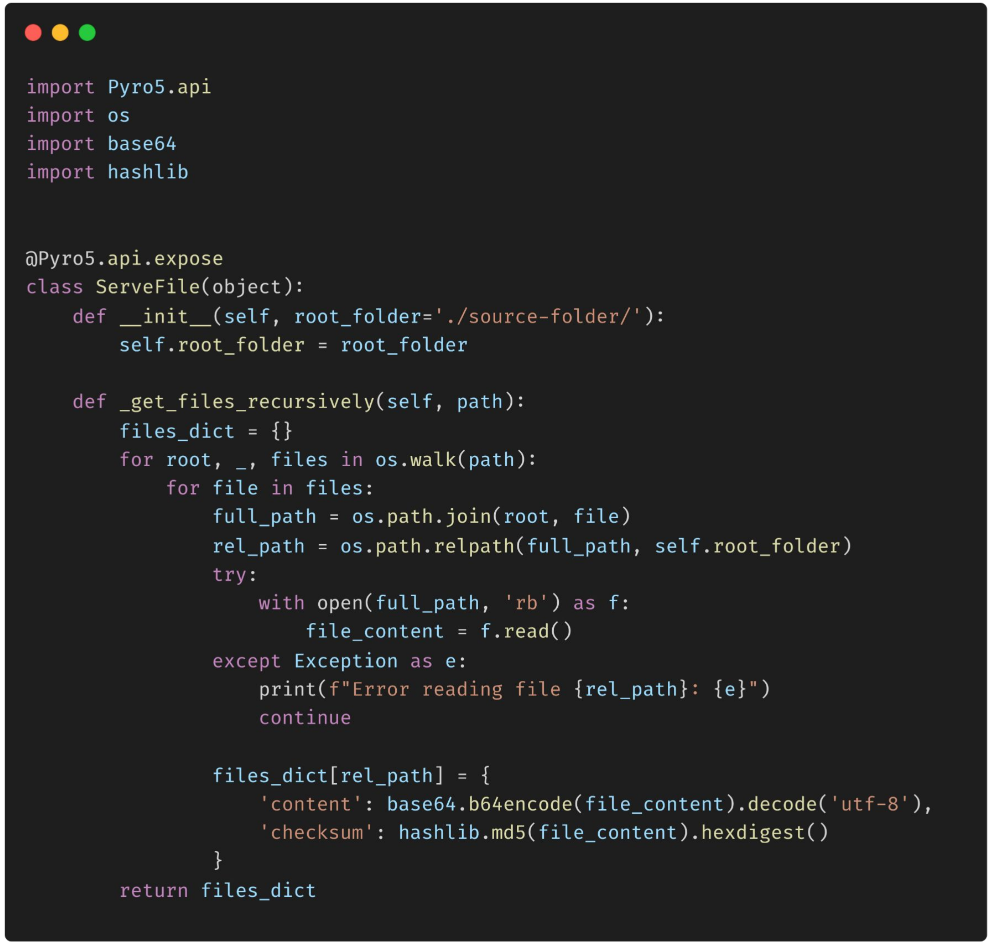
- OS
- Hashlib
- Base64
- Subprocess

main.py



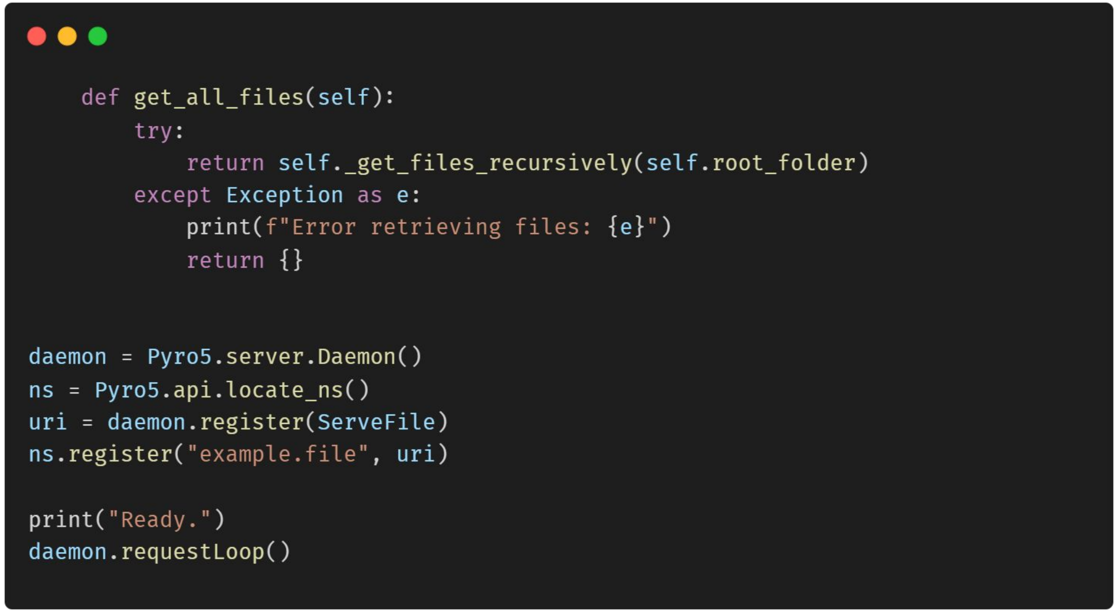
```python
import subprocess
import time

nameserver_proc = subprocess.Popen(["pyro5-ns"])

server_proc = subprocess.Popen(["python3", "server.py"])

client_proc = subprocess.Popen(["python3", "client.py"])

try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    nameserver_proc.terminate()
    server_proc.terminate()
    client_proc.terminate()
    print("Exiting gracefully...")
```

server.py



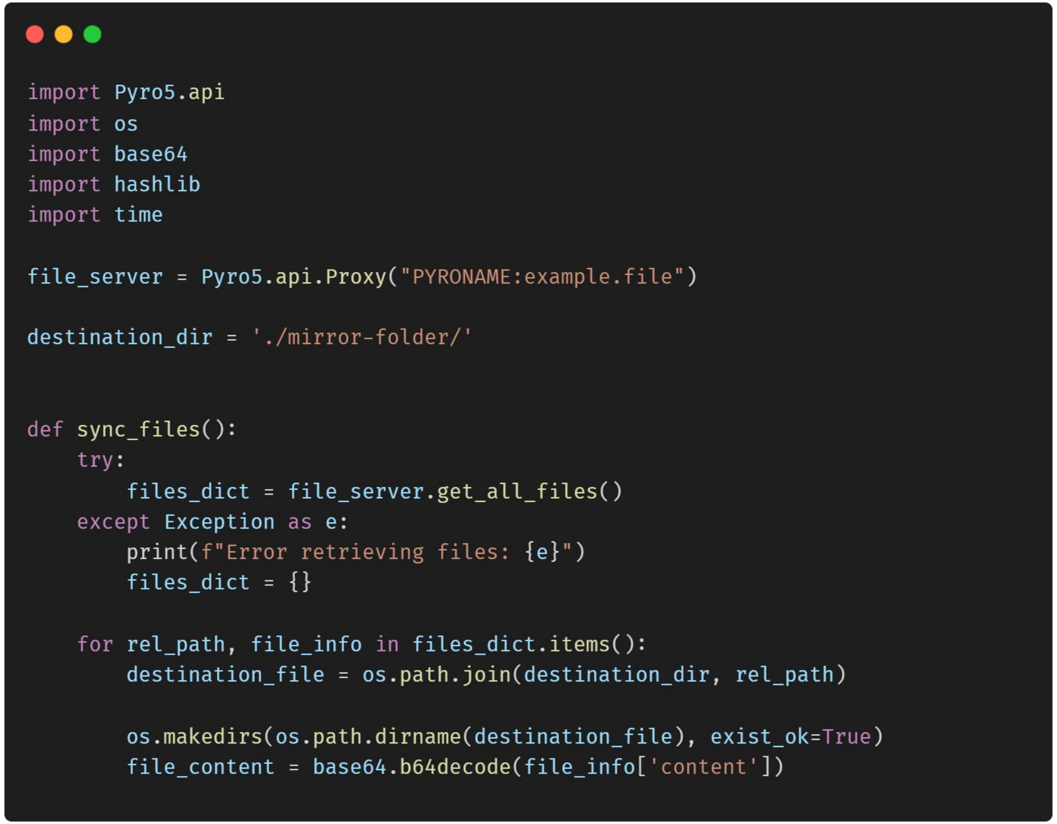
```python
import Pyro5.api
import os
import base64
import hashlib


@Pyro5.api.expose
class ServeFile(object):
    def __init__(self, root_folder='./source-folder/'):
        self.root_folder = root_folder

    def _get_files_recursively(self, path):
        files_dict = {}
        for root, _, files in os.walk(path):
            for file in files:
                full_path = os.path.join(root, file)
                rel_path = os.path.relpath(full_path, self.root_folder)
                try:
                    with open(full_path, 'rb') as f:
                        file_content = f.read()
                except Exception as e:
                    print(f"Error reading file {rel_path}: {e}")
                    continue

                files_dict[rel_path] = {
                    'content': base64.b64encode(file_content).decode('utf-8'),
                    'checksum': hashlib.md5(file_content).hexdigest()
                }
        return files_dict
```

server.py



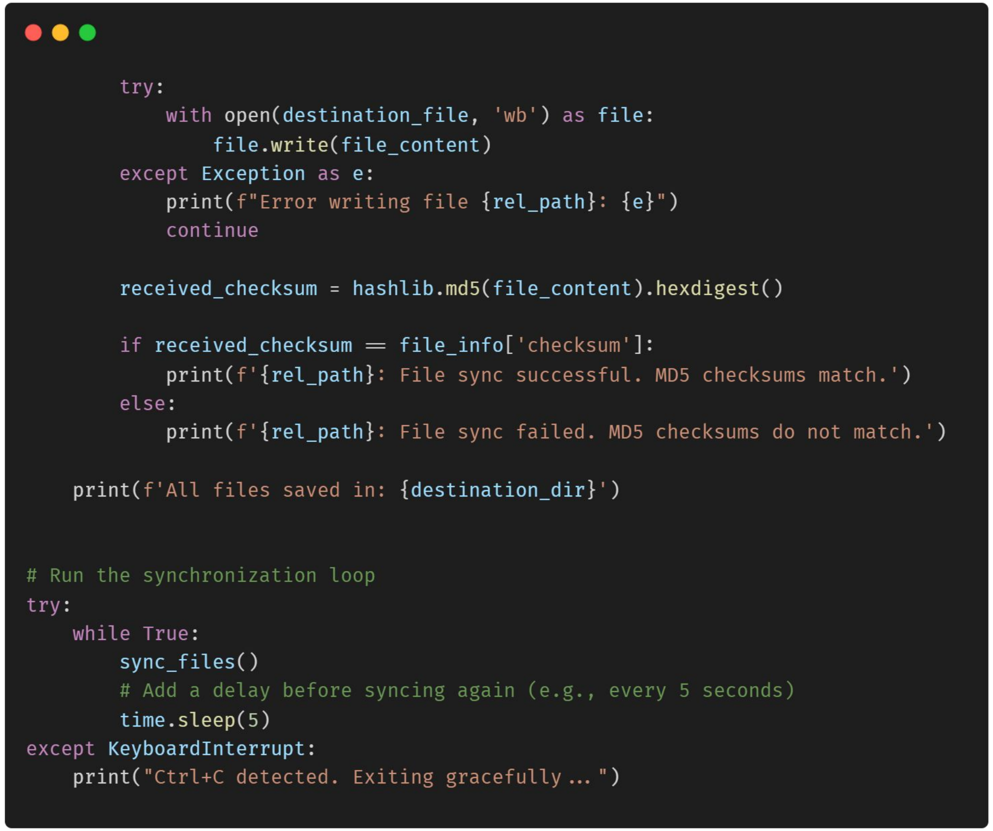
```
    def get_all_files(self):
        try:
            return self._get_files_recursively(self.root_folder)
        except Exception as e:
            print(f"Error retrieving files: {e}")
            return {}


daemon = Pyro5.server.Daemon()
ns = Pyro5.api.locate_ns()
uri = daemon.register(ServeFile)
ns.register("example.file", uri)

print("Ready.")
daemon.requestLoop()
```

client.py



```python
import Pyro5.api
import os
import base64
import hashlib
import time

file_server = Pyro5.api.Proxy("PYRONAME:example.file")

destination_dir = './mirror-folder/'


def sync_files():
    try:
        files_dict = file_server.get_all_files()
    except Exception as e:
        print(f"Error retrieving files: {e}")
        files_dict = {}

    for rel_path, file_info in files_dict.items():
        destination_file = os.path.join(destination_dir, rel_path)

        os.makedirs(os.path.dirname(destination_file), exist_ok=True)
        file_content = base64.b64decode(file_info['content'])
```

client.py



```python
        try:
            with open(destination_file, 'wb') as file:
                file.write(file_content)
        except Exception as e:
            print(f"Error writing file {rel_path}: {e}")
            continue

        received_checksum = hashlib.md5(file_content).hexdigest()

        if received_checksum == file_info['checksum']:
            print(f'{rel_path}: File sync successful. MD5 checksums match.')
        else:
            print(f'{rel_path}: File sync failed. MD5 checksums do not match.')

    print(f'All files saved in: {destination_dir}')


# Run the synchronization loop
try:
    while True:
        sync_files()
        # Add a delay before syncing again (e.g., every 5 seconds)
        time.sleep(5)
except KeyboardInterrupt:
    print("Ctrl+C detected. Exiting gracefully...")
```