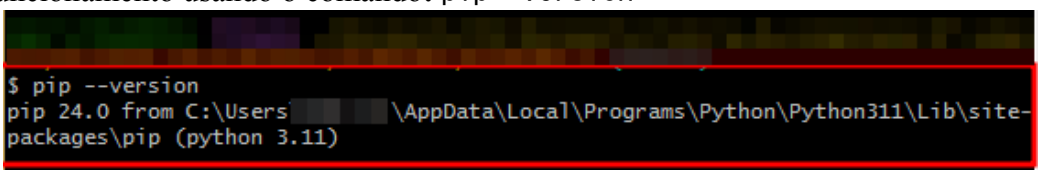
Universidade Federal do Ceará - Campus de Quixadá
Disciplina: Sistemas Distribuídos
Código: QXD0043
Professor: Rafael Braga

1. Instalar o Python: Você pode seguir os seguintes tutoriais a depender do seu sistema operacional:
Linux: https://youtu.be/R9dLGLVqK9Q?si=pXjKGhcY5jtSplvT
Windows: https://youtu.be/72PJBhXFC8I?si=SBcSf8kyECacrnsG

2. Verificar o PIP: Logo após instalar o Python, verifique se o gerenciador de pacotes do Python está em funcionamento usando o comando: `pip -version`



Caso contrário:

- Baixe o script em https://bootstrap.pypa.io/get-pip.py .
- Abra um terminal/prompt de comando, vá até a pasta que contém o arquivo get-pip.py e execute:

```
python get-pip.py
```

3. Criando o projeto: No seu editor de código favorito, crie os seguintes arquivos com o seus respectivos conteúdos:

O arquivo `config.py` terá o seguinte conteúdo:

```
import configparser

# Cria um objeto ConfigParser, equivalente ao Properties em Java
config = configparser.ConfigParser()

# Adiciona as propriedades (chave-valor)
config['DEFAULT'] = {
    'arquivo': 'estoque.xml',
}

# Tenta gravar as propriedades no arquivo "config.ini"
with open('config.ini', 'w') as configfile:
    config.write(configfile)
```

O arquivo app.py terá o seguinte conteúdo:

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import xml.etree.ElementTree as ET
import configparser

# Configuração do arquivo XML
config = configparser.ConfigParser()
config.read('config.ini')
ARQUIVO = config['DEFAULT'].get('arquivo')

app = FastAPI()

# Modelo base para os aparelhos
class Aparelho(BaseModel):
    id: int
    nome: str
    marca: str
    preco: float
    categoria: str
    deposito: str

# Interface Transferível
class Transferivel:
    def transferir(self, novo_deposito):
        self.deposito = novo_deposito

# Funções auxiliares para manipulação do XML
def carregar_aparelhos_xml():
    try:
        tree = ET.parse(ARQUIVO)
        root = tree.getroot()
        aparelhos = []
        for aparelho in root.findall("aparelho"):
            aparelhos.append({
                "id": int(aparelho.find("id").text),
                "nome": aparelho.find("nome").text,
                "marca": aparelho.find("marca").text,
                "preco": float(aparelho.find("preco").text),
                "categoria": aparelho.find("categoria").text,
                "deposito": aparelho.find("deposito").text,
            })
        return aparelhos
    except FileNotFoundError:
        root = ET.Element("estoque")
```

```
        tree = ET.ElementTree(root)
        tree.write(ARQUIVO, encoding="utf-8", xml_declaration=True)
        return []

def salvar_aparelhos_xml(aparelhos):
    root = ET.Element("estoque")

    for aparelho in aparelhos:
        aparelho_elem = ET.SubElement(root, "aparelho")
        ET.SubElement(aparelho_elem, "id").text = str(aparelho["id"])
        ET.SubElement(aparelho_elem, "nome").text = aparelho["nome"]
        ET.SubElement(aparelho_elem, "marca").text = aparelho["marca"]
        ET.SubElement(aparelho_elem, "preco").text = str(aparelho["preco"])
        ET.SubElement(aparelho_elem, "categoria").text = aparelho["categoria"]
        ET.SubElement(aparelho_elem, "deposito").text = aparelho["deposito"]

    tree = ET.ElementTree(root)
    tree.write(ARQUIVO, encoding="utf-8", xml_declaration=True)

# Endpoints
@app.post("/aparelhos", status_code=201)
def adicionar_aparelho(aparelho: Aparelho):
    aparelhos = carregar_aparelhos_xml()
    if any(a["id"] == aparelho.id for a in aparelhos):
        raise HTTPException(status_code=400, detail="ID já existe.")

    aparelhos.append(aparelho.dict())
    salvar_aparelhos_xml(aparelhos)
    return {"erro": False, "message": "Aparelho adicionado com sucesso"}

@app.get("/aparelhos")
def listar_aparelhos():
    return carregar_aparelhos_xml()

@app.get("/aparelhos/{id}")
def buscar_aparelho(id: int):
    aparelhos = carregar_aparelhos_xml()
    for aparelho in aparelhos:
        if aparelho["id"] == id:
            return aparelho
    raise HTTPException(status_code=404, detail="Aparelho não encontrado.")

@app.put("/aparelhos/{id}")
def atualizar_aparelho(id: int, aparelho_atualizado: Aparelho):
    aparelhos = carregar_aparelhos_xml()
    for aparelho in aparelhos:
```

```
        if aparelho["id"] == id:
            aparelho.update(aparelho_atualizado.dict())
            salvar_aparelhos_xml(aparelhos)
            return {"erro": False, "message": "Aparelho atualizado com sucesso"}
    raise HTTPException(status_code=404, detail="Aparelho não encontrado.")

@app.delete("/aparelhos/{id}")
def deletar_aparelho(id: int):
    aparelhos = carregar_aparelhos_xml()
    aparelhos_atualizados = [a for a in aparelhos if a["id"] != id]
    if len(aparelhos) == len(aparelhos_atualizados):
        raise HTTPException(status_code=404, detail="Aparelho não encontrado.")

    salvar_aparelhos_xml(aparelhos_atualizados)
    return {"erro": False, "message": "Aparelho removido com sucesso"}

@app.put("/aparelhos/{id}/transferir/{novo_deposito}")
def transferir_aparelho(id: int, novo_deposito: str):
    aparelhos = carregar_aparelhos_xml()
    for aparelho in aparelhos:
        if aparelho["id"] == id:
            aparelho["deposito"] = novo_deposito
            salvar_aparelhos_xml(aparelhos)
            return {"erro": False, "message": "Aparelho transferido com
sucesso"}
    raise HTTPException(status_code=404, detail="Aparelho não encontrado.")
```

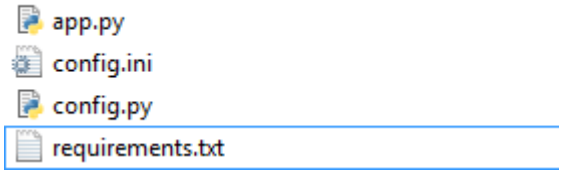
Agora você deve criar um novo arquivo, chamado `requirements.txt` que deve conter o seguinte conteúdo:

```
fastapi==0.115.14
pydantic==2.11.7
uvicorn
```

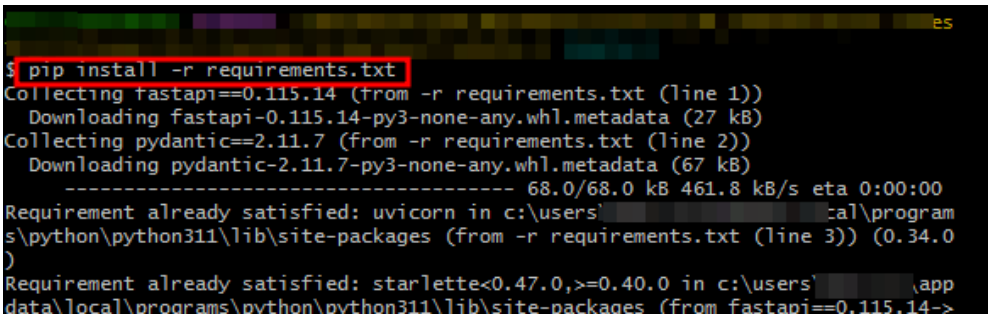
Após isso, execute o seguinte comando:

```
python config.py
```

Uma estrutura semelhante a essa deve ser criada no seu projeto:

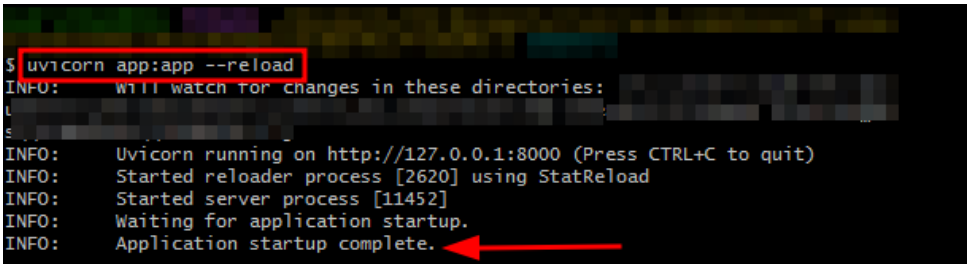


4. Instale as dependências: `pip install -r requirements.txt`



5. Executando o servidor:

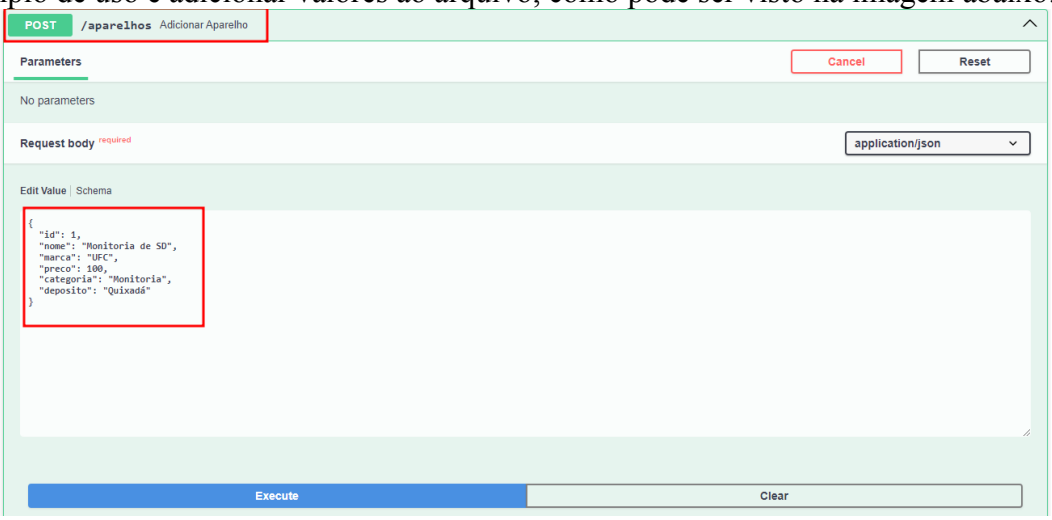
No diretório digite `uvicorn app:app --reload`



6. Executando o código:

Ao acessar a URL `http://127.0.0.1:8000/docs` você irá visualizar uma tela igual a essa que permite você fazer inúmeras operações em nosso XML usando os métodos HTTP.

Um exemplo de uso é adicionar valores ao arquivo, como pode ser visto na imagem abaixo:



Se voltarmos ao nosso XML, é possível ver nossos dados armazenados: