Script-projetoWe

☑ Revisado ☐

Relatório: Configuração do Script para Criar Projetos e Repositórios no GitHub

Este relatório explica como configurar e utilizar o script Python para criar projetos localmente e automaticamente configurar repositórios no GitHub, com as opções de tornar o repositório **público** ou **privado**.

Pré-requisitos

Antes de executar o script, certifique-se de que o seguinte está configurado no computador de destino:

1. **Instalar Python**:
   - Baixe e instale o Python.
   - Durante a instalação, **marque a opção** "Add Python to PATH" para garantir que o Python esteja acessível globalmente no terminal.
2. **Instalar Visual Studio Code**:
   - Baixe e instale o Visual Studio Code.

- Certifique-se de que o comando `code` está acessível no terminal (isso geralmente ocorre durante a instalação do VS Code).

3. **Instalar a biblioteca `requests`**:
   - O script utiliza a biblioteca `requests` para interagir com a API do GitHub.
   - Para instalar essa biblioteca, execute o comando:

     ```
     pip install requests
     ```

4. **Criar um Token de Autenticação no GitHub**:
   - Acesse GitHub Tokens e crie um novo **Personal Access Token**.
   - Selecione as permissões necessárias, como **repo** (acesso completo aos repositórios).
   - **Copie** o token gerado, pois ele será usado no script.
5. **Configurar a variável de ambiente `GITHUB_TOKEN`**:
   - No **Windows**:
     1. Abra o **Painel de Controle**.
     2. Vá para **Sistema e Segurança > Sistema > Configurações Avançadas do Sistema**.
     3. Clique em **Variáveis de Ambiente**.
     4. Na seção **Variáveis de usuário**, clique em **Novo**.
     5. Defina:
        - Nome da variável: `GITHUB_TOKEN`
        - Valor da variável: o token gerado no GitHub.
   - No **Linux ou macOS**, adicione o token ao arquivo `.bashrc` ou `.bash_profile`:

     ```
     export GITHUB_TOKEN="seu_token_aqui"
     ```

6. **Configuração do Caminho para o VS Code**:
   - O script foi configurado para usar o caminho do VS Code no **Windows**. Certifique-se de que o caminho esteja correto, como:

```
C:\Users\seu_usuario\AppData\Local\Programs\Microsoft VS Code\bin\code.cmd
```

- Caso o caminho seja diferente no seu sistema, altere no script para o caminho correto.

Como Usar o Script

Após garantir que os pré-requisitos estão configurados, siga estas etapas para usar o script:

1. **Baixe ou Crie o Script**:
   - Crie um arquivo Python (`projetoWeb.py`) com o conteúdo do script fornecido (ver exemplo abaixo).
2. **Executar o Script**:
   - Abra o terminal ou o **Prompt de Comando**.
   - Navegue até o diretório onde o script está localizado.
   - Execute o script com o comando:

     ```
     python projetoWeb.py
     ```
3. **Passos Durante a Execução**:
   - O script solicitará o **nome do projeto**.
   - Se o projeto já existir, ele perguntará se deseja **substituir** o projeto.
   - Em seguida, o script cria o diretório localmente e abre o projeto no **VS Code**.
   - Após a criação do projeto, será perguntado ao usuário se o repositório GitHub será **público** ou **privado**.
   - O script cria o repositório automaticamente no **GitHub** com base na escolha do usuário.
4. **Detalhes do Funcionamento**:
   - O script utiliza a **API do GitHub** para criar o repositório no GitHub com o nome fornecido.

- O **token de autenticação** (`GITHUB_TOKEN`) é usado para autorizar a criação do repositório na conta do usuário no GitHub.
- A visibilidade do repositório é definida como pública ou privada, dependendo da escolha do usuário.

Script Python Completo

```
import os
import subprocess
import shutil
import requests

def create_and_open_project():
    # Diretório base para os projetos
    base_directory = r"C:/dev/projects"

    # Certifique-se de que o diretório base existe
    os.makedirs(base_directory, exist_ok=True)

    # Solicitar o nome do projeto
    project_name = input("Nome do projeto: ").strip()

    # Caminho completo do novo projeto
    project_path = os.path.join(base_directory, project_nam
e)

    # Verificar se o projeto já existe
    if os.path.exists(project_path):
        print(f"O projeto '{project_name}' já existe em: {p
roject_path}")
        # Perguntar ao usuário se deseja substituir o proje
to
        resposta = input("Deseja substituir o projeto? (s o
u n): ").strip().lower()

        if resposta in ['sim', 's']:
            # Excluir o diretório existente
            shutil.rmtree(project_path)
```

```
            print(f"O projeto '{project_name}' foi excluíd
o.")
        else:
            print("Projeto não foi substituído. Saindo.")
            return

    # Perguntar se o repositório será público ou privado
    visibility = input("O repositório será público ou priva
do? (público/privado): ").strip().lower()
    if visibility in ["público", "publico"]:
        is_private = False
    elif visibility in ["privado", "privado"]:
        is_private = True
    else:
        print("Opção inválida. Usando repositório público p
or padrão.")
        is_private = False

    # Criar o repositório no GitHub
    print("Criando repositório no GitHub...")
    repo_ssh_url = create_github_repo(project_name, is_priv
ate)

    if not repo_ssh_url:
        print("Erro ao criar repositório no GitHub. Saind
o...")
        return

    # Clonar o repositório para o diretório do projeto
    try:
        subprocess.run(["git", "clone", repo_ssh_url, proje
ct_path], check=True)
        print(f"Repositório '{project_name}' clonado com su
cesso em {project_path}")
    except subprocess.CalledProcessError as e:
        print(f"Erro ao clonar o repositório: {e}")
        return
```

```
    # Abrir o diretório no VS Code usando o arquivo .cmd
    try:
        subprocess.run([r"C:\Users\AlmavivA\AppData\Local\P
rograms\Microsoft VS Code\bin\code.cmd", project_path], che
ck=True)
        print(f"Abrindo o projeto '{project_name}' no VS Co
de...")
    except FileNotFoundError:
        print("Erro: O comando 'code' não foi encontrado. C
ertifique-se de que o VS Code está instalado e configurado
corretamente.")
    except subprocess.CalledProcessError as e:
        print(f"Erro ao abrir o projeto no VS Code: {e}")

def create_github_repo(repo_name, is_private):
    # Verificando se o token de autenticação foi encontrado
    token = os.getenv("GITHUB_TOKEN")
    if not token:
        print("Erro: Token de autenticação não encontrado.
Defina a variável de ambiente 'GITHUB_TOKEN'.")
        return None

    # Dados para a criação do repositório no GitHub
    url = "https://api.github.com/user/repos"
    headers = {"Authorization": f"token {token}"}
    data = {
        "name": repo_name,
        "description": f"Repositório do projeto {repo_nam
e}",
        "private": is_private  # Define a privacidade do re
positório
    }

    # Fazendo a requisição para a API do GitHub
    response = requests.post(url, headers=headers, json=dat
a)

    # Verificar se o repositório foi criado com sucesso
```

```
    if response.status_code == 201:
        print(f"Repositório '{repo_name}' criado no GitHu
b!")
        # Solicitar a chave SSH do repositório
        repo_ssh_url = input(f"Agora, insira a chave SSH do
repositório GitHub (geralmente no formato git@github.com:us
uario/{repo_name}.git): ").strip()
        return repo_ssh_url
    else:
        print(f"Erro ao criar repositório: {response.json
().get('message', 'Desconhecido')}")
        return None

if __name__ == "__main__":
    create_and_open_project()
```

Conclusão

Com este script, você pode criar projetos localmente, abrir no VS Code e automaticamente configurar repositórios no GitHub, com a opção de torná-los **públicos** ou **privados**.

Para usá-lo em outro computador, basta garantir que os pré-requisitos (Python, VS Code, token de autenticação) estão configurados corretamente, e seguir os passos descritos.