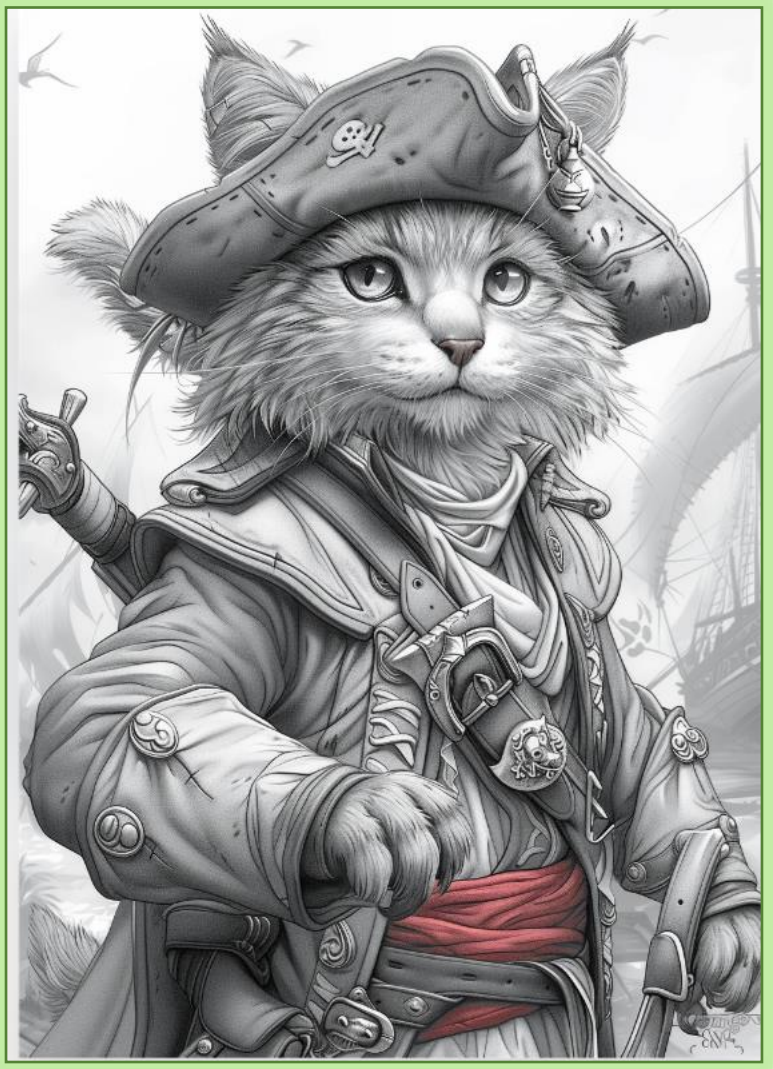
"DOMINE O GITHUB: DO ZERO AO DEPLOY COM ESTRATÉGIAS PRÁTICAS PARA TODOS OS NÍVEIS"



"GUIA COMPLETO PARA CRIAR, COLABORAR E GERENCIAR PROJETOS DE SUCESSO“

Mônica Oliveira

INTRODUÇÃO: DO ZERO AO DEPLOY NO GITHUB

Iniciar um projeto do zero exige uma base sólida e um planejamento meticuloso. Primeiramente, é crucial definir os objetivos do projeto de forma clara e detalhada. Isso inclui entender a necessidade do projeto, o público-alvo e os problemas que ele visa resolver. Essa fase inicial deve envolver a criação de um documento de requisitos, que servirá como guia para toda a equipe durante o desenvolvimento. Ferramentas como Trello ou Jira podem ser utilizadas para mapear as funcionalidades e tarefas, permitindo uma visualização clara do progresso e das prioridades.

Uma vez definidos os objetivos e requisitos, a próxima etapa é escolher a stack de tecnologias que será utilizada. Decidir sobre as linguagens de programação, frameworks e ferramentas de desenvolvimento é essencial para garantir que o projeto seja sustentável e eficiente. Além disso, a estruturação do repositório no GitHub deve ser bem planejada desde o início, com a criação de branches para desenvolvimento, teste e produção. Isso facilita a organização e evita conflitos de código no futuro.

Com a complexidade crescente dos projetos de software, é fundamental adotar práticas eficientes desde a concepção até a implementação final. Este guia completo visa fornecer uma abordagem estruturada para transformar uma ideia inicial em um produto totalmente funcional, utilizando o GitHub como a principal ferramenta de controle de versão e colaboração.

ETAPAS

Criar um projeto no GitHub é um processo que envolve várias etapas, desde a concepção da ideia até a manutenção contínua do projeto. Aqui está um guia minucioso de cada etapa:

Concepção e Planejamento

a. Ideia e Objetivo:

•Defina a ideia central do projeto.
•Estabeleça os objetivos e o propósito do projeto.

b. Requisitos:

•Identifique os requisitos funcionais e não funcionais.
•Faça uma análise de viabilidade técnica.

c. Documentação Inicial:

•Crie um documento de visão geral do projeto.
•Elabore um plano de projeto com metas e prazos.

1. Criação do Repositório

A primeira etapa é criar um novo repositório no GitHub. Siga os seguintes passos:

•Acesse sua conta do GitHub e clique no botão "New" ou "Novo Repositório".

•Escolha um nome significativo e descritivo para o seu repositório. Esse nome deve refletir o propósito do projeto.

•Adicione uma descrição curta para que outros usuários e colaboradores entendam rapidamente o objetivo do projeto.
•Decida se o repositório será público ou privado. Repositórios públicos são visíveis para todos, enquanto repositórios privados são acessíveis apenas para você e os colaboradores que você convidar.

•Selecione a opção de inicializar o repositório com um README. O README é um arquivo crucial que deve conter informações básicas sobre o projeto, incluindo seu propósito, como configurá-lo e usá-lo.

•Escolha uma licença adequada. Licenças, como MIT ou Apache 2.0, definem os termos sob os quais outros podem usar, modificar e distribuir seu código.

2. Estruturação do Projeto

Após criar o repositório, é importante estruturar o projeto de maneira organizada:

•**README.md**: Este arquivo deve ser detalhado e informativo. Além das informações básicas, inclua seções sobre como instalar dependências, executar o projeto e contribuir com ele.

•**.gitignore**: Este arquivo lista os arquivos e diretórios que o Git deve ignorar. Isso é útil para evitar a inclusão de arquivos desnecessários, como configurações locais, logs e dependências instaladas.

•**LICENSE**: Inclua o arquivo de licença escolhido para deixar claro os direitos e obrigações dos usuários do código.

•**src/**: Crie uma pasta para o código-fonte do projeto. Mantenha uma estrutura de diretórios lógica e bem organizada, facilitando a navegação e manutenção do código.

•**tests/**: Inclua uma pasta separada para os testes, garantindo que eles estejam organizados e sejam facilmente executáveis.

3. Primeiros Commits e Documentação

Com a estrutura básica em vigor, é hora de fazer os primeiros commits e começar a documentar o projeto:

Faça o commit inicial com a estrutura do repositório e os arquivos principais. Utilize mensagens de commit claras e descritivas.

Comece a desenvolver a funcionalidade básica do projeto, fazendo commits frequentes para registrar o progresso e facilitar a colaboração.

Documente cada nova funcionalidade adicionada no README e, se necessário, crie arquivos de documentação adicionais na pasta docs/.

Configure o repositório para permitir a colaboração, definindo regras para Pull Requests e revisões de código. Isso pode incluir a criação de um arquivo CONTRIBUTING.md com diretrizes para contribuidores.

Iniciar um projeto no GitHub de forma organizada e bem planejada estabelece uma base sólida para o desenvolvimento contínuo e a colaboração eficaz.

Seguir essas etapas ajuda a manter o projeto gerenciável e acessível para todos os membros da equipe, facilitando o crescimento e a evolução do código.

2. Configuração do Repositório no GitHub

a. Criação do Repositório:

1.Acesse o GitHub e faça login.

2.No canto superior direito, clique no ícone "+" e selecione "New repository".

3.Preencha o nome do repositório, descrição (opcional) e escolha a visibilidade (público ou privado).

4.Marque a opção de adicionar um arquivo README se desejar.

1. Clone o repositório para sua máquina local:

bash Copiar código

```
git clone https://github.com/usuario/nome-do-repositorio.git
```

2. Navegue até o diretório do repositório:

bash Copiar código

```
cd nome-do-repositorio
```

3. Estruturação do Projeto

a. Organização de Diretórios:

- Crie uma estrutura de diretórios lógica para o seu projeto. Exemplo:

kotlin — Copiar código

```
/nome-do-repositorio
  ├── src
  ├── docs
  ├── tests
  ├── data
  └── scripts
```

b. Arquivos Essenciais:

•README.md: Descrição do projeto, como instalar, usar e contribuir.

•.gitignore: Arquivos e diretórios a serem ignorados pelo Git.

•LICENSE: Escolha e adicione uma licença apropriada para seu projeto.

4. Desenvolvimento e Versionamento

a. Commits e Branches:

- Faça commits frequentes e descritivos.
- Use branches para desenvolver novos recursos ou corrigir bugs.

bash Copiar código

```
git checkout -b nome-da-branch
```

- Depois de implementar e testar, mescle a branch com a principal.

bash Copiar código

```
git checkout main
git merge nome-da-branch
```

b. Pull Requests:

- Se o projeto for colaborativo, utilize pull requests para revisão de código.
- No GitHub, crie um pull request para discutir e revisar as mudanças antes de mesclar.

5. Documentação

a. Documentação Técnica:

•Documente o código com comentários e docstrings.
•Crie documentação detalhada no diretório docs.

b. Wiki:

•Utilize a wiki do GitHub para documentar informações adicionais sobre o projeto.

6. Testes e Integração Contínua

a. Testes:

•Desenvolva testes automatizados para seu código (unitários, de integração, etc.).
•Use frameworks de teste apropriados (ex.: pytest para Python).

b. Integração Contínua:

•Configure uma pipeline de CI/CD (ex.: GitHub Actions, Travis CI).
•Automatize a execução dos testes e o deployment.

7. Lançamento

a. Versões e Releases:

•Utilize as tags do Git para marcar versões estáveis do projeto.

bash Copiar código

```
git tag -a v1.0.0 -m "Primeira versão estável"
git push origin v1.0.0
```

- No GitHub, utilize a seção de "Releases" para criar releases formais.

8. Manutenção e Suporte

a. Gestão de Issues:

•Utilize o sistema de Issues do GitHub para rastrear bugs e solicitações de recursos.

•Priorize e atribua issues à equipe de desenvolvimento.

b. Feedback e Comunidade:

•Mantenha canais de comunicação abertos com a comunidade (issues, discussions, etc.).

•Aceite e revise pull requests da comunidade.

c. Atualizações:

•Mantenha o projeto atualizado com novas funcionalidades, correções de bugs e melhorias.

Exemplos de Comandos e Configurações

Clonagem e Configuração Inicial

bash Copiar código

```
git clone https://github.com/usuario/nome-do-repositorio.git
cd nome-do-repositorio
echo "# Nome do Projeto" >> README.md
git add README.md
git commit -m "adiciona README inicial"
git push origin main
```

Criando e Mesclando uma Branch

bash Copiar código

```
git checkout -b nova-funcionalidade
# Implementa a nova funcionalidade
git add .
git commit -m "implementa nova funcionalidade"
git push origin nova-funcionalidade
# No GitHub, cria um Pull Request e mescla após revisão
```

Configurando Integração Contínua com GitHub Actions

1.No GitHub, vá para a aba "Actions" do seu repositório.

2.Escolha um template de workflow ou crie o seu próprio.

3.Adicione um arquivo .yml em .github/workflows com a configuração desejada.

Exemplo de um workflow simples:

yaml

Copiar código

```yaml
name: CI

on: [push, pull_request]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - name: Checkout repository
      uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: '3.x'

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: Run tests
      run: |
        pytest
```

Seguir esses passos com cuidado garantirá um processo organizado e eficiente para criar e manter um projeto no GitHub.

CONCLUSÃO

Ebook digital criado como resultado da aprendizagem do módulo CRIANDO UM EBOOK COM CHATGPT & MIDJOURNEY aula ministrada pelo Professor Felipe Aguiar da Dio.

O conteúdo foi redigido com as facilidades das ferramentas de IA.

LINK: C:\Users\monic\OneDrive\Área de Trabalho\EBOOK DIO