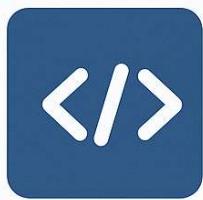


COMO FUNCIONA O GITHUB



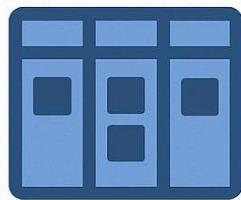
CODE



ISSUES



WIKI



PROJETOS



Luís Simões da Cunha (2025)

Índice

| | |
|--|----|
|  Parte 1 – Introdução ao GitHub | 5 |
| O que é o GitHub e para que serve? | 5 |
| Conceitos fundamentais: Git vs GitHub | 5 |
| Principais casos de uso (programação, documentação, colaboração) | 5 |
|  Programação | 5 |
|  Documentação | 6 |
|  Colaboração | 6 |
|  Em resumo: | 6 |
|  Parte 2 – Criação de Conta e Exploração Inicial | 7 |
| Como criar uma conta no GitHub | 7 |
| Tour pela interface: repositórios, issues, pull requests | 7 |
|  Repositórios (Repos) | 7 |
|  Issues | 8 |
|  Pull Requests (PR) | 8 |
| O teu primeiro "star"  e "fork"  | 8 |
|  Dar "star" a um repositório | 8 |
|  Fazer "fork" de um repositório | 8 |
|  Em resumo: | 9 |
|  Parte 3 – Conceitos Essenciais de Git | 10 |
| O que é versionamento? | 10 |
| Commits, branches e merges | 10 |
|  Commit | 10 |
|  Branch | 10 |
|  Merge | 11 |
| Como Git e GitHub trabalham juntos | 11 |
|  Em resumo: | 11 |
|  Parte 4 – Começar um Projeto no GitHub | 12 |
| Criar um novo repositório | 12 |
| Adicionar e editar ficheiros (README.md, etc.) | 12 |
|  README.md – A tua carta de apresentação | 12 |
|  Como editar ficheiros no GitHub Web Interface | 13 |
| Trabalhar com o GitHub Web Interface (sem terminal ainda) | 13 |

| | |
|--|----|
|  Em resumo:..... | 14 |
|  Parte 5 – Git Local: Instalação e Primeiro Commit  | 15 |
| Instalar Git no teu computador | 15 |
|  Windows | 15 |
|  MacOS | 15 |
|  Linux | 15 |
| Clonar um repositório | 15 |
| Fazer alterações locais e enviar para o GitHub (push)..... | 16 |
| 1 Fazer uma alteração | 16 |
| 2 Verificar o estado | 16 |
| 3 Adicionar alterações (stage) | 16 |
| 4 Fazer o commit..... | 16 |
| 5 Enviar (push) para o GitHub | 16 |
|  Em resumo:..... | 17 |
|  Parte 6 – Branches e Colaboração  | 18 |
| Criar e gerir branches | 18 |
| O que é um branch? | 18 |
| Como criar um branch..... | 18 |
| Como ver todos os branches..... | 18 |
| Como mudar de branch..... | 18 |
| Criar um Pull Request (PR)..... | 19 |
| Como criar um PR no GitHub: | 19 |
| Revisão de código e merge de PRs | 19 |
| Como fazer merge de um Pull Request: | 19 |
|  Em resumo:..... | 20 |
|  Parte 7 – Issues, Wiki e Projetos  | 21 |
| Criar e gerir issues | 21 |
| O que são Issues? | 21 |
| Como criar uma Issue..... | 21 |
| Gerir Issues..... | 21 |
| Usar a Wiki para documentação | 22 |
| O que é a Wiki? | 22 |
| Como criar e editar uma Wiki | 22 |
| Usar o GitHub Projects (Kanban Board) | 22 |

| | |
|---|----|
| O que é o GitHub Projects? | 22 |
| Como criar um Projeto no GitHub | 23 |
| 🎯 Em resumo:..... | 23 |
| 📘 Parte 8 – Boas Práticas e Recursos Avançados 🧠 | 24 |
| Estratégias para commits claros e organizados | 24 |
| Boas práticas: | 24 |
| Licenciamento e ficheiros .gitignore..... | 24 |
| 📜 Licenciamento..... | 24 |
| 📄 .gitignore | 25 |
| Introdução ao GitHub Actions e Pages | 25 |
| ⚙️ GitHub Actions – Automatizações mágicas..... | 25 |
| 🌐 GitHub Pages – Publicar sites facilmente..... | 26 |
| 🎯 Em resumo:..... | 27 |
| 🎓 Fim do Curso: O teu novo superpoder digital! | 28 |
| 📝 Cartão de Revisão – GitHub em 8 Partes..... | 29 |
| Parte 1 – Introdução ao GitHub 🌟 | 29 |
| Parte 2 – Criação de Conta e Exploração Inicial 🔍 | 29 |
| Parte 3 – Conceitos Essenciais de Git 🔥 | 29 |
| Parte 4 – Começar um Projeto no GitHub 🚀 | 29 |
| Parte 5 – Git Local: Instalação e Primeiro Commit 💻 | 29 |
| Parte 6 – Branches e Colaboração 👤 | 30 |
| Parte 7 – Issues, Wiki e Projetos 🛡️ | 30 |
| Parte 8 – Boas Práticas e Recursos Avançados 🧠 | 30 |
| 🏆 Lembrar sempre: | 31 |
| 🏆 Agora sim: GitHub é teu! 🚀 | 31 |

Parte 1 – Introdução ao GitHub

O que é o GitHub e para que serve?

Imagina uma enorme **biblioteca online**  onde programadores, equipas e empresas guardam, organizam e partilham o seu código. Essa biblioteca chama-se **GitHub**.

→ GitHub é uma **plataforma de alojamento de código** que permite:

- Guardar projetos de programação de forma segura na "nuvem" 
- Acompanhar **todas as alterações** feitas aos ficheiros ao longo do tempo (versionamento)
- Trabalhar **em equipa**, de forma organizada e controlada
- Mostrar o teu trabalho a potenciais empregadores ou colaboradores

Resumo rápido:

GitHub = onde o teu código vive + onde colaboras com o mundo! 

Conceitos fundamentais: Git vs GitHub

Antes de avançares, é crucial entenderes esta distinção:

| Git | GitHub |
|--|--|
| Ferramenta que guarda o histórico de alterações do teu projeto | Plataforma online que usa o Git para permitir partilha e colaboração |
| Corre no teu computador (localmente)  | Vive na internet (nuvem)  |
| Podes usar Git sem GitHub | GitHub facilita o uso de Git com ferramentas sociais e web |

🌐 Analogia simples:

- Git é como o **motor**  que regista todas as viagens que fazes com o teu projeto.
 - GitHub é o **parque de estacionamento**  onde guardas o carro para outras pessoas verem, usarem ou até melhorarem contigo!
-

Principais casos de uso (programação, documentação, colaboração)

O GitHub não é só para "coders hardcore"! Vamos ver as aplicações mais comuns:

Programação

- Criar aplicações, sites, jogos, APIs, etc.

- Trabalhar em projetos pessoais ou de empresa
- Open Source: partilhar projetos com o mundo

Documentação

- Criar **Wikis** para organizar conhecimento
- Gerir tutoriais, manuais de utilizador e FAQs
- Usar **Markdown** (linguagem simples) para escrever de forma estruturada

Colaboração

- Trabalhar em equipa: vários programadores num mesmo projeto, sem conflitos
 - Discutir melhorias através de **Issues** e **Pull Requests**
 - Fazer revisões de código, atribuir tarefas e acompanhar progresso (Kanban Boards)
-

Em resumo:

GitHub é a casa dos teus projetos digitais, onde tudo é registado, protegido, melhorado e partilhado. E tudo começa por entenderes bem a diferença entre Git (a ferramenta) e GitHub (a plataforma). 

Parte 2 – Criação de Conta e Exploração Inicial ⚡

Como criar uma conta no GitHub

Vamos começar pelo primeiro passo: **entrar no mundo GitHub!** 🚀 ⚡

1. Acede ao site:

👉 github.com

2. Clica em "Sign Up" (no canto superior direito).

3. Preenche os dados:

- **Username:** o teu nome público (ex: joaoprogramador)
- **Email:** válido e que uses frequentemente
- **Password:** segura e fácil de lembrar

4. Escolhe o tipo de plano:

- Podes escolher o **plano gratuito**, que é mais do que suficiente para começares.

5. Confirma o teu email:

- Vais receber um email de verificação — confirma para poderes usar todas as funcionalidades.

🎉 **Parabéns!** Agora tens uma conta no GitHub e estás pronto para criar e colaborar em projetos incríveis.

Tour pela interface: repositórios, issues, pull requests

Ao entrares na tua conta GitHub, vais encontrar uma interface limpa e funcional. Vamos fazer um mini-tour



📁 Repositórios (Repos)

- **O que são?**

São como **pastas mágicas** 🌟 onde guardas todos os ficheiros e o histórico de um projeto.

- **Ações comuns:**

- Criar um repositório novo
- Ver o código de projetos existentes
- Ler o README.md para entender cada projeto

Issues

- **O que são?**

São como **tarefas** ou **problemas** registados para resolver no projeto.

- **Exemplos:**

- "Corrigir erro na página de login"
- "Adicionar nova funcionalidade de pesquisa"

- **Importância:**

Facilitam a **gestão de trabalho** entre equipas.

Pull Requests (PR)

- **O que são?**

Um **pedido para juntar alterações** que fizeste a um projeto.

- **Fluxo típico:**

- Criar uma cópia do projeto (fork ou branch)
- Fazer alterações
- Submeter um Pull Request para que os donos do projeto analisem e aceitem as tuas melhorias

Resumo visual:

Ficheiros → Repositórios → Tarefas → Issues → Alterações → Pull Requests

O teu primeiro "star" e "fork"

Agora que já tens conta, é hora de interagir com projetos!

Dar "star" a um repositório

- Um **Star** é como um "Gosto"  para projetos que admiras ou queres seguir.
- Clica no botão **Star** no topo de qualquer repositório para o adicionares à tua lista de favoritos.

Fazer "fork" de um repositório

- **Fork** significa criar uma **cópia pessoal** do projeto para poderes trabalhar à vontade.
- Útil para:
 - Experimentares sem medo de estragar o original
 - Começar a contribuir para projetos Open Source

Como fazer:

1. Entra num repositório que gostes.

2. Clica em **Fork** (canto superior direito).
 3. Boom  ! Agora tens uma cópia só tua para explorar!
-

Em resumo:

Criar conta, dar o teu primeiro Star e fazer o primeiro Fork são os primeiros passos para começares a explorar o vasto universo de projetos, ideias e colaboração que o GitHub oferece. 

📘 Parte 3 – Conceitos Essenciais de Git 🌐

O que é versionamento?

Imagina que estás a escrever um livro 📖. Se alterares o final hoje e amanhã quiseres voltar à versão anterior, como fazias?

Sem controlo de versões, seria um caos!

➡ Versionamento é o processo de **guardar diferentes versões** do teu projeto ao longo do tempo, permitindo:

- Voltar a qualquer estado anterior (como um "Undo" gigante 🔍)
- Saber quem fez o quê, quando e porquê
- Trabalhar em paralelo com outras pessoas sem atropelos

No fundo, é como **um diário de bordo 📜** do teu projeto, guardando cada mudança importante.

Commits, branches e merges

Vamos agora às peças básicas do Git ✨:

✓ Commit

- Um **commit** é como tirar uma foto 📸 do teu projeto num determinado momento.
- Cada commit guarda:
 - As alterações feitas
 - Uma descrição breve (mensagem de commit)
 - Quem fez as alterações e quando

Exemplo real de mensagem de commit:

"Corrigido bug no login de utilizador"

➡ Moral da história:

Cada vez que fazes um commit, estás a guardar um "checkpoint" no histórico do projeto.

🛠 Branch

- Um **branch** é como criar uma **linha paralela** de desenvolvimento.
- Podes trabalhar numa nova funcionalidade ou corrigir um erro sem afetar o projeto principal.

Exemplo:

- `main` → linha principal

- feature/adicionar-pesquisa → linha onde crias uma nova funcionalidade

💡 Porque usar branches?

Para **experimentar e inovar** sem medo de "estragar" o projeto principal.

☒ Merge

- Depois de terminares o teu trabalho num branch, queres **juntar** essas alterações de volta à linha principal.
- **Merge** é o ato de **combinar** duas linhas de trabalho num só fluxo.

Exemplo:

Fizeste a nova funcionalidade no feature/adicionar-pesquisa → Agora fazes **merge** para o **main**.

Como Git e GitHub trabalham juntos

Agora que já conheces o básico do Git, vamos ver como ele e o GitHub se completam:

| Git | GitHub |
|---|--|
| Guarda versões no teu computador localmente | Guarda as versões online na nuvem |
| Trabalhas offline | Trabalhas online, colaboras com outros |
| Precisas de comandos (<code>git add</code> , <code>git commit</code>) | Interages pela web ou apps visuais |

✳️ Fluxo típico de trabalho:

1. Clonar o repositório do GitHub → Git cria uma cópia local
2. Trabalhar localmente → Fazer commits
3. Enviar alterações para o GitHub → Push
4. Receber alterações de outros → Pull

⌚ Resumo visual:

Local (Git) ⇌ Nuvem (GitHub)

⌚ Em resumo:

Git é o motor que guarda o histórico do teu projeto.

GitHub é a garagem gigante onde esse motor é partilhado e melhorado com o mundo.  

Parte 4 – Começar um Projeto no GitHub

Criar um novo repositório

Agora que já tens conta e sabes o básico, vamos **criar o teu primeiro projeto** no GitHub!

1. **Inicia sessão** no GitHub.
2. **No canto superior direito**, clica no ícone "+"  **New repository**.
3. **Preenche os detalhes:**
 - **Repository name**: o nome do teu projeto (ex: meu-primeiro-projeto)
 - **Description** (opcional): uma breve descrição do que é o projeto.
 - **Visibility**:
 - **Public**: qualquer pessoa pode ver (ideal para projetos abertos)
 - **Private**: só tu e quem tu autorizares podem ver
4. **Inicializar o repositório com:**
 - **README.md** (altamente recomendado!): é o cartão de visita do teu projeto.
 - (Opcional) **.gitignore**: lista de ficheiros a ignorar (ex.: ficheiros temporários, pastas de build).
 - (Opcional) **License**: tipo de licença de utilização do teu código (ex: MIT, Apache 2.0).
5. **Clica em "Create repository"**.

 Parabéns! Acabaste de criar a "casa" onde o teu projeto vai viver.  

Adicionar e editar ficheiros (README.md, etc.)

README.md – A tua carta de apresentação

O ficheiro README .md é onde explicas:

- O que faz o projeto
- Como instalar e usar
- Quem contribuiu
- Licença (se aplicável)

Dica:

O .md vem de **Markdown**, uma linguagem simples para formatar texto (negrito, títulos, listas, links...).

Exemplo básico de README.md:

Meu Primeiro Projeto 🚀

Este é o meu primeiro projeto no GitHub!

Funcionalidades

- Simples
- Divertido
- Educativo

Como usar

1. Clone o repositório
2. Execute o projeto

Feito com ❤️ por [O Teu Nome].

✍ Como editar ficheiros no GitHub Web Interface

Não precisas de instalar nada para fazer pequenas alterações — basta usar a própria interface web!

1. Entra no teu repositório.
2. Clica no ficheiro que queres editar (por exemplo, README.md).
3. Clica no ícone de lápis 🖍 (canto superior direito do ficheiro).
4. Faz as alterações desejadas.
5. Escreve uma **mensagem de commit** (ex: "Atualizei descrição do projeto").
6. Clica em **Commit changes**.

💡 Nota importante:

Cada alteração que fazes é registada como um **commit**, ajudando-te a manter o histórico do que foi mudado.

Trabalhar com o GitHub Web Interface (sem terminal ainda)

A **interface web do GitHub** é perfeita para:

- Criar e editar ficheiros
- Adicionar novas pastas
- Navegar no histórico de alterações
- Criar branches e pull requests
- Gerir issues e projetos

Vantagem para iniciantes:

Começar pelo browser reduz a sobrecarga de aprendizagem — podes focar-te nos conceitos fundamentais sem te preocupares com comandos complicados. 😊

Em resumo:

Criar o teu primeiro repositório, escrever um README e fazer alterações diretamente no browser são passos fundamentais para começares a dominar o GitHub de forma simples e prática!



Parte 5 – Git Local: Instalação e Primeiro Commit

Agora que já dominamos o GitHub pelo browser, é hora de entrar no mundo real dos programadores: **usar Git no teu computador!**  

Instalar Git no teu computador

Antes de tudo, precisas de instalar o **Git**:

Windows

- Vai a  git-scm.com
- Clica em **Download for Windows**.
- Corre o instalador e segue os passos:
 - Escolhe as opções padrão (não te preocipes, são seguras).
 - Importante: garante que a opção "**Git from the command line**" está ativa.

MacOS

- Usa o **Homebrew** (se tens instalado):
`brew install git`
- Ou descarrega diretamente em  git-scm.com.

Linux

- Usa o gestor de pacotes da tua distribuição:

```
sudo apt install git      # (Ubuntu/Debian)  
sudo dnf install git     # (Fedora)
```

Depois de instalar, verifica se o Git ficou disponível:

```
git --version
```

Deves ver algo como `git version 2.43.0`.

Clonar um repositório

"**Clonar**" significa criar uma cópia local de um projeto que está no GitHub.

1. Vai ao repositório que queres clonar no GitHub.

2. Clica em <> **Code** e copia o link **HTTPS**.
3. No teu computador, abre o terminal (Prompt de Comando / Bash / Terminal).
4. Escreve:

```
git clone https://github.com/username/repositorio.git
```

(Substitui pelo link real.)

5. Vais ver uma nova pasta com o projeto clonado! 🎉
-

Fazer alterações locais e enviar para o GitHub (push)

Agora que tens o repositório no teu computador, podes trabalhar nele localmente:

1 Fazer uma alteração

- Abre um ficheiro no editor de texto (ex: Visual Studio Code).
- Faz uma pequena modificação no README.md, por exemplo.

2 Verificar o estado

```
git status
```

Vais ver que há ficheiros modificados.

3 Adicionar alterações (stage)

Antes de guardar oficialmente no histórico, tens de "preparar" os ficheiros:

```
git add nome-do-ficheiro
```

Ou, para adicionar tudo de uma vez:

```
git add .
```

4 Fazer o commit

Guarda as alterações com uma mensagem clara:

```
git commit -m "Atualizei o README com mais detalhes"
```

Cada commit é um ponto de recuperação!

5 Enviar (push) para o GitHub

Finalmente, envia as tuas alterações para o repositório online:

```
git push
```



Nota:

Na primeira vez, o Git pode pedir o teu utilizador e palavra-passe ou um token de acesso pessoal.

🔗 Em resumo:

Com o Git instalado, podes trabalhar no teu computador, fazer alterações, guardar versões (commits) e enviar tudo para o GitHub com um simples comando de push. Bem-vindo ao verdadeiro mundo dos programadores!

Parte 6 – Branches e Colaboração

Agora que já sabes trabalhar localmente com Git e enviar para o GitHub, vamos explorar o que realmente torna o Git e o GitHub tão poderosos: **colaboração estruturada!**  

Criar e gerir branches

O que é um branch?

Um **branch** (ramo) é uma **cópia paralela** do teu projeto, onde podes:

- Trabalhar numa nova funcionalidade
- Corrigir um erro
- Testar ideias novas

 **Vantagem:** podes experimentar sem medo de mexer no projeto principal (normalmente chamado `main` ou `master`).

Como criar um branch

1. No terminal, dentro do teu projeto:

```
git checkout -b nome-do-branch
```

Exemplo:

```
git checkout -b feature/pagina-de-contacto
```

Este comando:

- Cria um novo branch
 - Muda-te automaticamente para ele
-

Como ver todos os branches

```
git branch
```

O branch atual aparecerá com um *.

Como mudar de branch

```
git checkout nome-do-branch
```

Podes saltar entre diferentes linhas de trabalho facilmente!

Criar um Pull Request (PR)

Quando terminares o teu trabalho num branch, é hora de pedir para o integrar no projeto principal. Isto faz-se através de um **Pull Request (PR)**.

Como criar um PR no GitHub:

1. **Faz push** do teu branch para o GitHub:

```
git push --set-upstream origin nome-do-branch
```

2. No GitHub, vais ver um botão "**Compare & Pull Request**" — clica nele!

3. **Preenche o PR**:

- Título: algo simples e claro (ex: "Adicionar página de contacto")
- Descrição: explica o que mudaste e porquê

4. **Submete o PR**.



Dica:

É boa prática explicar as motivações e as mudanças de forma clara para quem for rever o teu código.

Revisão de código e merge de PRs

Depois de criares um PR:

- Outros colaboradores (ou tu próprio, se for um projeto pessoal) vão **analisar** as alterações:
 - Verificar se o código está correto
 - Sugerir melhorias
 - Confirmar se não quebra outras partes do projeto

Como fazer merge de um Pull Request:

1. Depois da revisão aprovada, clica em "**Merge Pull Request**" no GitHub.
 2. Depois podes **apagar o branch** (opcional mas recomendado para manter o projeto limpo).
-



Resumo visual do fluxo:

```
main → cria novo branch → faz alterações → commit e push → cria Pull Request  
→ revisão → merge no main
```

⌚ Em resumo:

Trabalhar com branches e Pull Requests permite **colaborar de forma organizada, experimentar sem medo e garantir qualidade** antes de integrar novas alterações ao projeto principal. É o segredo para projetos de sucesso em equipa!  

Parte 7 – Issues, Wiki e Projetos

À medida que os projetos crescem, torna-se essencial manter tudo **organizado e documentado**. GitHub oferece ferramentas incríveis para isso — e vamos explorá-las agora! 

Criar e gerir issues

O que são Issues?

- Uma **Issue** é como uma **tarefa, alerta ou pedido** registado dentro do projeto.
 - Pode ser:
 - Reportar um **bug** 
 - Sugerir uma **nova funcionalidade** 
 - Levantar uma **dúvida** 
-

Como criar uma Issue

1. Dentro do teu repositório GitHub, vai à aba **Issues**.
 2. Clica em **New Issue**.
 3. Preenche:
 - **Título** claro e objetivo
 - **Descrição** detalhada (se possível, com imagens ou exemplos)
 4. Opcionalmente, podes:
 - Atribuir a Issue a alguém
 - Associar a um projeto ou milestone
 - Adicionar **labels** (etiquetas) para melhor organização (ex: bug, enhancement, question)
-

Gerir Issues

- **Fechar uma Issue:** Quando o problema for resolvido ou a tarefa concluída, fecha a Issue.
- **Referenciar Issues em Commits e PRs:**
 - Se num commit resolveres uma issue, podes indicar:

```
git commit -m "Corrigir erro no login. Fixes #23"
```

(Substitui #23 pelo número da Issue.)

Usar a Wiki para documentação

O que é a Wiki?

- Cada repositório pode ter a sua **Wiki**: um espaço organizado para escrever documentação detalhada sobre o projeto.

 Ideal para:

- Guias de instalação
 - Manual de utilizador
 - FAQs
 - Explicação da arquitetura do sistema
-

Como criar e editar uma Wiki

1. No teu repositório, clica na aba **Wiki**.
2. Clica em **Create the first page**.
3. Escreve o teu conteúdo usando **Markdown** (como no README).
4. Salva a página.

Podes criar:

- Páginas ligadas
- Índices
- Subdivisões temáticas

 **Dica:**

Documentação boa faz toda a diferença para quem chega ao projeto mais tarde!

Usar o GitHub Projects (Kanban Board)

O que é o GitHub Projects?

- É uma ferramenta de **gestão visual de tarefas** baseada em **Kanban** (quadro de colunas como "A Fazer", "Em Progresso" e "Concluído").

Ideal para:

- Organizar o trabalho de equipas
 - Acompanhar o progresso de funcionalidades
 - Priorizar bugs e melhorias
-

Como criar um Projeto no GitHub

1. No repositório, clica em **Projects**.
 2. Clica em **New Project**.
 3. Escolhe um modelo (ex: **Kanban**).
 4. Cria colunas como:
 - **To Do** (A Fazer)
 - **In Progress** (Em Progresso)
 - **Done** (Feito)
 5. Liga Issues e Pull Requests diretamente ao Projeto:
 - Arrasta Issues entre colunas conforme o progresso.
-

💡 Resumo visual:

Issues = tarefas individuais 🔧

Wiki = documentação organizada 📖

Projects = gestão visual de tarefas 📋

Estas ferramentas, combinadas, fazem do GitHub muito mais do que um sítio para guardar código — transformam-no num **centro de gestão de projetos completo!** 🚀 ✨

⌚ Em resumo:

Dominar Issues, Wikis e Projects permite que tu e a tua equipa **organizem tarefas, documentem conhecimento e acompanhem o progresso** de forma simples, visual e eficiente. 🔥

Parte 8 – Boas Práticas e Recursos Avançados

Chegámos à reta final! Agora vamos dar aquele salto de qualidade: tornar o teu trabalho mais **profissional, limpo e automatizado**.  

Estratégias para commits claros e organizados

Os commits são o **diário de bordo** do teu projeto.

Commits **claros** tornam o histórico fácil de ler e ajudam-te (e à tua equipa) a entender rapidamente o que foi feito.

Boas práticas:

Mensagens breves mas descriptivas:

- Em vez de: Update
- Prefere: Corrigir erro de validação no formulário de contacto

Usar o tempo verbal imperativo:

- "Corrigir erro" em vez de "Corrigido erro"
- "Adicionar nova secção ao README"

Commits pequenos e focados:

- Um commit deve corresponder a uma **única ideia** ou alteração.
 - Evita "mega commits" que fazem tudo ao mesmo tempo!
-

Licenciamento e ficheiros .gitignore

Licenciamento

Se queres que outros usem o teu código, precisas de **especificar uma licença**.

- Durante a criação do repositório, podes escolher uma licença (ex.: **MIT, Apache 2.0, GPL**).
- Ou cria manualmente um ficheiro LICENSE no repositório.

Exemplo breve de licença MIT:

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy...



Nota:

Sem licença explícita, o teu código é **protegido automaticamente por direitos de autor** — outros não podem usá-lo livremente!



O ficheiro `.gitignore` diz ao Git **quais ficheiros ou pastas ignorar** — ou seja, ficheiros que **não devem ser versionados**.

Exemplos comuns:

- Ficheiros temporários (`*.tmp`)
- Pastas de build (`/build/`)
- Configurações locais (`*.env`, `.vscode/`)

Exemplo básico de `.gitignore`:

```
# Ignorar ficheiros de sistema
.DS_Store
Thumbs.db

# Ignorar pastas de build
build/
dist/
```



Dica:

GitHub oferece templates prontos para `.gitignore`, adaptados a diferentes linguagens (Python, Java, Node.js, etc.).

Introdução ao GitHub Actions e Pages



GitHub Actions – Automatizações mágicas

GitHub Actions permite **automatizar tarefas**:

- Fazer testes automáticos sempre que alguém faz um Push
- Compilar código automaticamente
- Atualizar a documentação

Funciona através de **workflows** que escreves em ficheiros `.yml` dentro da pasta `.github/workflows/`.

Exemplo simples:

```
name: CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Instalar Dependências
        run: npm install
      - name: Executar Testes
        run: npm test
```

💡 Moral:

O GitHub pode fazer muitas tarefas chatas por ti, de forma automática! 🚀

🌐 GitHub Pages – Publicar sites facilmente

GitHub Pages permite **publicar um site** diretamente a partir do teu repositório!

Ideal para:

- Portfólios
- Documentação de projetos
- Landing Pages de apps

Como funciona:

1. Ativa GitHub Pages nas definições do teu repositório.
2. Escolhe uma branch para servir os ficheiros (`main`, por exemplo).
3. Cria um ficheiro `index.html`.
4. O GitHub gera automaticamente o teu site — gratuito e sem complicações!

🔗 Link típico:

<https://o-teu-utilizador.github.io/nome-do-projeto/>

Em resumo:

Boas práticas de commits, licenciamento adequado, exclusão inteligente de ficheiros, automatizações com Actions e publicação de sites com Pages — são estes os ingredientes que te transformam de utilizador básico em **verdadeiro mestre do GitHub!** 

Fim do Curso: O teu novo superpoder digital!

Parabéns por chegares até aqui! Agora tens:

- As bases sólidas para criar e gerir projetos no GitHub
- As ferramentas para colaborar com outros de forma profissional
- Conhecimentos para expandir com recursos avançados

Este é só o começo — o GitHub é uma porta aberta para oportunidades incríveis no mundo da programação, ciência de dados, documentação e muito mais!  

Cartão de Revisão – GitHub em 8 Partes

Parte 1 – Introdução ao GitHub

- **Git** = motor de controlo de versões no computador.
 - **GitHub** = plataforma online para alojar, colaborar e partilhar projetos.
 - Casos de uso: **programação, documentação, colaboração**.
-

Parte 2 – Criação de Conta e Exploração Inicial

- Criar conta em github.com.
 - Interface:
 - **Repos** = projetos
 - **Issues** = tarefas/problemas
 - **Pull Requests** = pedidos de integração
 - Primeiras interações: dar **Star**  e fazer **Fork** .
-

Parte 3 – Conceitos Essenciais de Git

- **Versionamento** = guardar versões do projeto.
 - **Commit** = registo de alterações.
 - **Branch** = linha paralela de desenvolvimento.
 - **Merge** = juntar branches.
 - **Git local ⇌ GitHub online**.
-

Parte 4 – Começar um Projeto no GitHub

- Criar repositório: nome, descrição, visibilidade.
 - Adicionar e editar ficheiros (`README.md` com Markdown).
 - Trabalhar inicialmente pelo browser (interface web).
-

Parte 5 – Git Local: Instalação e Primeiro Commit

- Instalar Git (git-scm.com).
- **Clonar** repositórios: `git clone`.
- Trabalhar localmente:

- `git add .`
 - `git commit -m "mensagem"`
 - `git push` (enviar para GitHub).
-

Parte 6 – Branches e Colaboração

- Criar branches: `git checkout -b novo-branch`.
 - Trabalhar em paralelo sem mexer no `main`.
 - Criar **Pull Requests** para propor alterações.
 - Revisar código e fazer **Merge** depois de aprovado.
-

Parte 7 – Issues, Wiki e Projetos

- **Issues**: tarefas e bugs registados.
 - **Wiki**: documentação organizada por páginas.
 - **GitHub Projects**: quadros Kanban para gestão visual do trabalho.
-

Parte 8 – Boas Práticas e Recursos Avançados

- **Commits claros**: breves, descritivos, verbos no imperativo.
- **Licenciamento**: escolher licença (ex.: MIT) e criar ficheiro `LICENSE`.
- **.gitignore**: ignorar ficheiros desnecessários no versionamento.
- **GitHub Actions**: automatizar testes, builds, etc.
- **GitHub Pages**: criar sites gratuitos a partir de repositórios.

Lembrar sempre:

- Commits pequenos e claros
- Branches para novas funcionalidades
- Pull Requests para integrar alterações
- Issues e Projects para organização
- Documentar bem com WIKIs e READMEs
- Aproveitar automações e Pages para ir mais longe

Resumo visual da dinâmica GitHub:

Código Local \leftrightarrow Git \leftrightarrow GitHub (Nuvem) \leftrightarrow Colaboração

 Agora sim: GitHub é teu! 