# Diretrizes Estratégicas de Versionamento

O **Guia Estratégico de Versionamento** define os padrões e práticas para uso eficiente do Git no ciclo de desenvolvimento. Seu objetivo é garantir organização, rastreabilidade e qualidade na evolução do código, promovendo colaboração estruturada e entregas previsíveis.

O guia estabelece:

- Estratégia de branches (feature, release, hotfix)
- Padrão de commits claros e rastreáveis
- Regras para Pull/Merge Requests
- Critérios de aprovação e integração
- Versionamento e controle de releases

Mais do que regras, trata-se de um alinhamento técnico que fortalece a governança do código, reduz riscos e sustenta a escalabilidade do produto.

- Guia de Desenvolvimento: GitHub Desktop & Azure
- Fluxo de Trabalho de Code Review e Status de Comentários

# Guia de Desenvolvimento: GitHub Desktop & Azure

1. Configuração Inicial (Clone)

Ao baixar o software, ir em File > Options na aba Git, preencher com:

Name -> Nome do desenvolvedor
Email -> O Lider do desenvolvimento deverá passar

**Lembrando que esse e-mail não é próprio, o mesmo é usado em comum entre os desenvolvedores.**

image.png
Image not found or type unknown

---

Ao baixar o projeto pela primeira vez, siga estes passos para manter a estrutura de pastas padronizada:

1. No **GitHub Desktop**, vá em `File` > `Clone Repository`.
2. Selecione a aba **URL**.
3. **URL do Repositório:** > `https://SuperSoftDelphi@dev.azure.com/SuperSoftDelphi/SistemaERPFISCO/_git/SistemaERPFISCO`
4. **Local Path (Atenção):** O GitHub Desktop tentará sugerir o nome `SistemaERPFISCO`.
   - **Obrigatório:** Altere o final do caminho para que a pasta se chame **`Codigo Fonte`**.
   - *Exemplo:* `C:\Codigo Fonte`

---

2. Fluxo de Branches

---

Trabalhamos com uma branch principal (`main`) e **branches de versão** para cada Sprint.

# Iniciando uma Tarefa (Caso)

1. Certifique-se de que a **Current Branch** selecionada é a branch da versão atual (ex: `Branch 166`).
2. Clique em **Fetch Origin** para baixar as últimas atualizações.
3. Vá em **New Branch**.

4. Nomeie sua branch com o número do Desenvolvedor/Numero do Caso (ex: `vinicius.pereira_0053315`).
5. Garanta que a base da nova branch seja a branch da versão (ex: `166-Caso-0053542`).

---

3. Commits e Squash

---

Para manter o histórico do Azure DevOps limpo e fácil de ler:

- **Frequência:** Faça commits conforme avança no código.
- **Squash:** Antes de finalizar a tarefa, você deve consolidar seus commits.
  - No GitHub Desktop, vá na aba **History**.
  - Selecione seus commits da tarefa, clique com o botão direito e escolha **Squash commits**.
  - Isso transformará vários commits pequenos em um único commit robusto com a descrição da tarefa.

**Exemplo de Commit**

image.png
Image not found or type unknown

---

4. Atualização de Código (Rebase)

---

**NUNCA utilize "Merge"** para atualizar sua branch com a versão. Use sempre o **REBASE**.

O Rebase mantém o histórico linear, colocando seus commits no topo das últimas alterações da equipe.
**Como fazer:**

1. Com sua branch de tarefa selecionada, vá no menu superior: `Branch` > **Rebase Current Branch...**
2. Selecione a branch da versão (ex: `166-Caso-0053542`).
3. O GitHub Desktop processará a reescrita do histórico. Se houver conflitos, resolva-os no VS Code/Delphi e finalize o rebase no GitHub Desktop.

---

**OBS:**

Caso apareça uma mensagem de erro ao clonar, dar fetch.

Só clicar em Clone > Generate Git Credencials e copiar o Password.

Clipboard - 23 de fevereiro de 2026 às 15_05.png
Image not found or type unknown

---

# Regras de Ouro (Resumo)

| O que fazer | O que NÃO fazer |
| --- | --- |
| Usar a pasta **Codigo Fonte** | Deixar o nome padrão `SistemaERPFISCO` |
| Fazer **Rebase** da versão | Dar "Merge" da versão na sua branch |
| Fazer **Squash** dos commits | Enviar dezenas de commits de "ajuste" no PR |
| Criar branch a partir da **Versão** | Criar branch a partir da `main` sem orientação |

# Fluxo de Trabalho de Code Review e Status de Comentários

Este documento estabelece o padrão para o ciclo de vida de Pull Requests (PRs) e o fluxo de status dos comentários dentro do Azure DevOps. O objetivo é garantir que nenhuma correção seja perdida e manter a clareza sobre quem deve atuar em cada etapa.

# Definições de Papéis

- **Autor:** Desenvolvedor responsável pela implementação e abertura do PR.
- **Revisor:** Desenvolvedor responsável pela análise técnica.

# O Fluxo do Pull Request

## 1. Abertura do PR

O **Autor** deve abrir o Pull Request apenas após o cumprimento dos seguintes requisitos:

- Implementação do caso finalizada.
- Funcionalidade validada e aprovada pela equipe de QA/Testes.

## 2. Revisão Inicial (Status: Active)

O **Revisor** analisa o código. Ao encontrar pontos de atenção, bugs ou sugestões de melhoria, insere comentários nas linhas correspondentes.

- **Ação do Sistema:** O Azure DevOps define automaticamente o status do comentário como `Active`.
- **Significado:** O ponto requer ação ou resposta do Autor.

# 3. Tratativa e Correção (Status: Pending)

O **Autor** atua sobre os comentários do Revisor.

- Após realizar o *commit* com a correção solicitada ou responder a uma dúvida, o Autor deve alterar manualmente o status do comentário para **`Pending`**.
- **Significado:** A correção foi realizada e aguarda revalidação do Revisor.
- *Nota:* O Autor **não** deve marcar o comentário como `Resolved`.

# 4. Validação e Encerramento (Status: Resolved)

O **Revisor** é notificado das atualizações e verifica os itens marcados como *Pending*.

- Se a correção estiver satisfatória, o Revisor altera o status para **`Resolved`**.
- **Significado:** O tópico foi concluído e o comentário será colapsado/ocultado da visualização principal.
- Se a correção não for satisfatória, o Revisor pode reverter o status para **`Active`** e adicionar novas observações.

image.png
Image not found or type unknown

# Boas Práticas

1. **Não resolva seus próprios tópicos:** Apenas quem criou o comentário (ou outro Revisor) deve marcá-lo como `Resolved`. Isso garante o duplo cheque de qualidade.
2. **Use o status "WontFix" com cautela:** Caso uma sugestão não possa ser aplicada, discuta no chat do PR antes de marcar como "Won't Fix".
3. **Contexto nos comentários:** Ao passar para `Pending`, se a correção não for óbvia, adicione uma resposta explicando o que foi feito (ex: "Corrigido no commit x").