GPS - Gestão de Projetos de Software

Aula 13: Ferramentas de Gestão

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Novembro de 2026

Sumário

Objetivos da Aula

- Analisar o **Custo da Fragmentação** (*Toolchain Chaos*) na rotina de desenvolvimento.
- Compreender os danos da **Carga Cognitiva** no foco do engenheiro.
- Estudar o princípio arquitetural da Única Fonte de Verdade (**SSOT**).
- Mapear o **Workflow Definitivo** dentro de uma plataforma integrada.
- Comparar o ecossistema do **GitLab** contra o peso gerencial do **Jira**.
- Dominar o uso da Comunicação **Assíncrona vs. Síncrona**.
- Aplicar a governança de Código via *Merge Requests* no GitLab (Check).

O Mentor Adverte

A Bancada do Engenheiro

"Um gênio de armadura não é nada sem a bancada certa, garoto. Ferramentas dispersas sugam a sua energia cognitiva e fragmentam a comunicação da equipe até o colapso estrutural."

Neste décimo terceiro capítulo, vamos combater a "fadiga de aplicativos" e centralizar o controle operacional. Você não será eficiente se o seu código estiver no GitHub, suas tarefas no Trello e suas decisões arquiteturais perdidas no WhatsApp.

A Sobrecarga Operacional

Como é a manhã de um desenvolvedor corporativo hoje?

- Abre o *Slack* para checar incêndios.
- Abre o *Jira* para olhar o Backlog.
- Abre o *Notion/Confluence* para ver documentação.
- Abre o *GitHub* para puxar o código.
- Abre o *Jenkins* para ver se o build explodiu.

Esse é o **Toolchain Chaos**.

Carga Cognitiva (Context Switching)

O cérebro humano queima energia pesada a cada troca de contexto.

Pular por seis interfaces de usuário diferentes destrói o estado de foco (*Flow*) da engenharia de software. Quando você interrompe um programador para ele mudar a aba e atualizar um status no sistema de terceiros, ele demora de 15 a 20 minutos para recuperar a concentração no algoritmo.

Silos de Informação Estilhaçados

O segundo efeito colateral do *Toolchain Chaos* é a descontinuidade da informação.

O Bug Arquitetural

Se o *Tech Lead* fechar a tarefa de ajuste de um *Prompt* da IA no Jira, mas esquecer de atualizar o Notion, o próximo desenvolvedor fará o código com base na regra antiga. A raiz de 90% dos bugs de arquitetura corporativa está na dessincronização entre as ferramentas da equipe!

A Ascensão do "Tudo-em-Um"

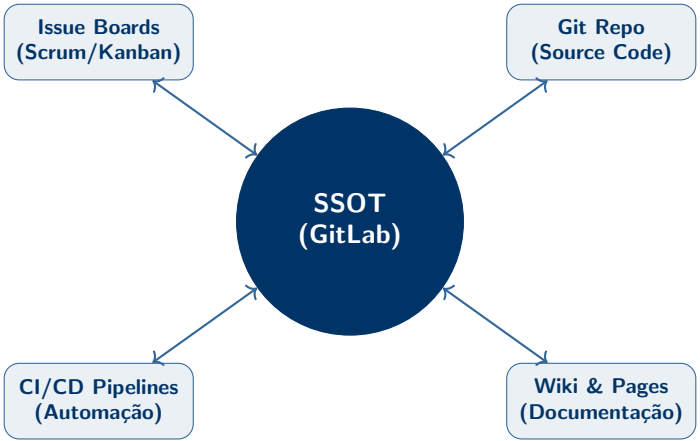
Para obliterar essa fragmentação de energia, a engenharia evoluiu para as **Plataformas Unificadas** (ex: GitLab, Azure DevOps, GitHub Enterprise).

O princípio arquitetural que rege essas ferramentas é o **SSOT**:

Single Source of Truth
(Única Fonte da Verdade)

O Ecossistema SSOT no GitLab

Essas plataformas engolem o ciclo de vida inteiro (DevOps) em uma só aba:



A Rastreabilidade Absoluta

A vantagem de concentrar Cartão (Board), Código (Git), Documentação (Wiki) e Deploy (CI/CD) no mesmo ecossistema é o **Poder Investigativo Forense**.

Se um erro entra em produção, a equipe rastreia:

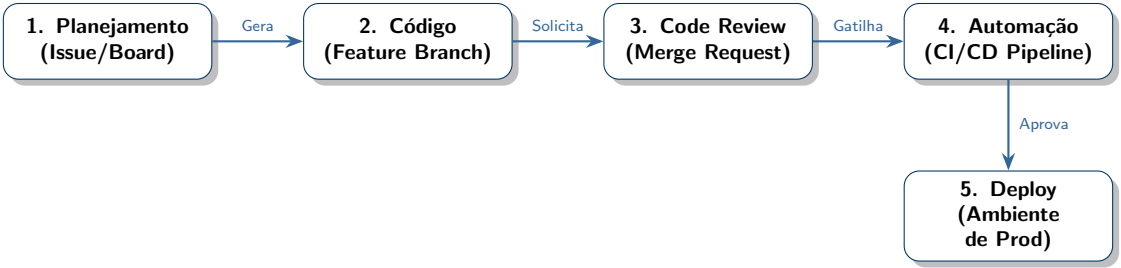
- O binário do erro até o *Pipeline*.
- O *Pipeline* até o *Commit*.
- O *Commit* de volta à *Issue* Original.
- Tudo em cliques no mesmo sistema. Ninguém pode dizer "Eu não sabia".

O Rastreio do Valor

A unificação de ferramentas acaba com a necessidade do "gerente chato" perguntando o *status* o tempo todo. O gestor visualiza o fluxo pelas engrenagens da plataforma de forma passiva.

O trabalho é transformado em etapas conectadas matematicamente.

Governança de Código (O Caminho Feliz)



Este rastro é auditável. Toda nova funcionalidade (ou ajuste de prompt na IA) se converte numa ramificação técnica (*branch*) atrelada à Issue que detalhou o negócio.

Guerra de Titãs: A Arquitetura do Jira

O **Jira** (da Atlassian) é o leviatã da gestão corporativa.

- É construído com foco quase exclusivo em Gestão de Portfólio.
- Ideal para implementar SAFe, pois estrutura perfeitamente hierarquias de *Initiatives* → *Epics* → *Stories* → *Sub-tasks*.
- Permite criar campos customizados avançados (ex: "Centro de Custo", "Risco Financeiro").

A Leveza Focada do GitLab

Se o Jira foca no Gerente, o **GitLab** foca no Engenheiro.

O Problema da Separação Física

O maior problema do Jira é ser um "corpo estranho" isolado. O desenvolvedor olha a tarefa no Jira, mas coda no GitHub e faz deploy no Jenkins.
Já o GitLab engloba tudo na mesma tela. Seus relatórios gerenciais podem não impressionar um Diretor Financeiro, mas a equipe técnica trabalha na velocidade da luz porque o *Board Kanban* e os *Merge Requests* compartilham o exato mesmo ecossistema.

Escolhendo o Canal de Comunicação

Humanos precisam debater para codar. Mas o erro fatal das equipes inexperientes é usar a ferramenta de comunicação errada, gerando estresse e perda de dados.

Você tem duas opções no cinturão:

- Comunicação **Síncrona** (Imediata).
- Comunicação **Assíncrona** (Permanente).

Comunicação Síncrona (A Faca Tática)

Ferramentas: WhatsApp, Slack, Teams, Discord (Voice), Zoom.

Para que serve: Apagar incêndios, coordenar *Swarming* (mutirão) e resolver dependências bloqueantes no dia. *"O servidor de homologação caiu?" "Manda o link do Zoom para debugar o Node.js juntos."*

Alerta Crítico

O chat é volátil. Se uma decisão de *Design Pattern* ou de requisito for tomada no WhatsApp, ela evapora em 48h sob centenas de figurinhas.

Comunicação Assíncrona (A Fundação)

Ferramentas: GitLab Issues, Documentação Wiki, MR Comments.

Para que serve: Histórico blindado. Decisões arquiteturais exigem registro permanente.
Se o cliente alterar o método de autenticação da aplicação, isso não fica no chat. Isso vira um comentário documentado na Issue. Respeita o *Flow* do engenheiro (ele lê quando puder) e a empresa não perde a "memória".

ADR: Architecture Decision Records

Como formalizar as grandes mudanças no *SSOT*?

Muitas equipes maduras utilizam os **ADRs**. São pequenos documentos de texto, versionados dentro do próprio Git, que explicam *por que* uma tecnologia foi escolhida.

- Exemplo: "Decidimos mudar de MongoDB para PostgreSQL porque..."
- Seis meses depois, quando um novo desenvolvedor perguntar "Por que não usamos Mongo?", ele não precisa perguntar no Slack; ele lê o ADR e obtém o contexto exato e a justificativa histórica.

A Lei Marcial do GitLab

No nosso Projeto (Assistente IA), imponho a Lei Marcial do SSOT:

"Se não está no GitLab, nunca existiu."

O grupo do WhatsApp da equipe serve para marcar pizza ou cobrar o colega atrasado. Qualquer decisão sobre como a IA deve processar documentos ou qual biblioteca Python usar DEVE ser oficializada nos comentários da *Issue* no GitLab.

Governança de Código na Prática

É proibido o modo "Deus" nos repositórios profissionais.

Nada na Main!

Nenhum desenvolvedor faz `commit` e envia o código direto para a *branch* principal (`main`). Tudo deve passar por:

1. Criação de uma **Feature Branch**.
2. Envio para o repositório.
3. Abertura de um **Merge Request (MR)**.
4. Um ou dois colegas aprovam o seu código no GitLab.

A Anatomia de uma Feature Branch

Como o desenvolvedor inicia o trabalho sem quebrar a `main`?

- Você abre o *Issue Board* e clica no botão azul mágico do GitLab: **"Create Merge Request"**.
- Esse clique cria automaticamente uma *Feature Branch* vinculada ao número do ticket (ex: `12-login-api`).
- Você agora tem uma cópia paralela e isolada do sistema inteiro para destruir, testar e experimentar sem medo. A `main` continua intocada e os clientes felizes.

A Batalha do Code Review (Merge Request)

O Merge Request (MR) não é só para fundir o código; é a principal ferramenta de **Qualidade** e **Aprendizado** da equipe.

O Que Acontece no MR?

1. Os seus colegas leem as linhas que você alterou (Diff).
2. Eles procuram falhas de lógica, ausência de testes ou *Hardcoded Passwords*.
3. Eles fazem comentários em linhas específicas. Você ajusta o código e faz novos *commits*.
4. Só após a aprovação deles, o botão **"Merge"** fica verde.

Check das Ferramentas e Infraestrutura

Auditoria de Maturidade

O seu projeto está equipado e blindado no ecossistema?

- **Repo + Board:** O Issue Board (Kanban) reflete fielmente o que cada um está desenvolvendo hoje no código?
- **SSOT de Comunicação:** O Scrum Master está garantindo que decisões do WhatsApp sejam transferidas para o GitLab?
- **Branches Protegidas:** O código está fluindo unicamente através de *Merge Requests*?

Perguntas Reflexivas

Antes de abrir a décima quinta aba no navegador, justifique:

1. Como o fenômeno da Carga Cognitiva (*Context Switching*) destrói a produtividade de um desenvolvedor júnior ao fragmentar o ecossistema de ferramentas?
2. Em operações críticas e plantões, qual é o limite de uso do Discord/WhatsApp, e por que ele é uma bomba-relógio para a perda do conhecimento?
3. Defender o princípio da Única Fonte da Verdade (SSOT) no GitLab blinda a equipe contra quais desvios ou manobras do cliente?

Obrigado!

Codificar é só metade da batalha.

Email: alessio@cefetmg.br
Horário de aula: Segunda, 7h às 8:40h
Horário de atendimento: Quarta 16h e whatsapp

Configurem os Merge Requests!

Próxima aula: Entraremos nos sistemas de telemetria e KPIs com Análise e Revisão do projeto!