

MBA IA • BUG EVALUATION

Do bug relatado ao pacote pronto para o time

O que a interface entrega de específico, comparada a perguntar direto ao Claude Code ou ao Codex da OpenAI



Assistente de código ou processo para o time?

Claude Code · Codex

Um assistente de programação

Trabalha dentro do código, conduzido por um desenvolvedor. O resultado depende de como cada pessoa conversa com ele e fica na conversa de quem perguntou.

MBA IA · Bug Evaluation

Um processo padronizado

Transforma o relato do bug em um pacote sempre no mesmo formato, guardado no histórico e usável por todo o time, do PO ao QA.

Um pedido, seis entregas no mesmo formato

User story

No padrão "Como um..., eu quero..., para que...", pronta para o backlog.

Critérios de aceitação

Em Dado / Quando / Então, para saber quando a correção está pronta.

Análise técnica

Causa provável do problema e pontos a verificar no sistema.

Plano de solução

Passo a passo da correção para o desenvolvedor seguir.

Casos de teste

No mínimo quatro cenários, incluindo os caminhos de erro, para o QA.

Projeto .zip

main.py, README e requirements, mais um arquivo .md para cada seção.

O mesmo padrão em todo bug, não importa quem pergunta

No chat direto

A qualidade depende de quem escreve o pedido. Dois analistas descrevendo o mesmo bug recebem respostas em formatos diferentes.

No MBA IA, as regras já vêm prontas

- Instruções fixas para a IA, escritas uma vez
- Resposta em formato estruturado, com seções obrigatórias
- Seção faltante é preenchida em vez de sumir
- Palavras-chave do bug são preservadas
- Regra explícita: "não invente funcionalidades"
- Baixa variação entre respostas (temperatura 0.2)

O projeto passa por checagens antes de chegar ao time

- ✓ Arquivos obrigatórios presentes e preenchidos
- ✓ O main.py compila sem erro de sintaxe
- ✓ A aplicação FastAPI está declarada
- ✓ Dependências coerentes com o que o código usa
- ✓ README escrito a partir das tecnologias realmente usadas



Autocorreção

Se o código gerado não compilar, o sistema devolve o erro para a IA e pede a correção, até duas vezes, sem ninguém intervir.

Cada projeto mostra o status: válido ou inválido.

Cada bug fica registrado e fácil de encontrar



Busca e filtros

Por texto do bug, status de validação e origem, com contagem e resumo.



Histórico por bug

Data, descrição e resultado de cada geração, do mais novo ao mais antigo.



Download como preferir

Projeto inteiro em .zip, arquivo por arquivo, ou texto limpo para colar no Word.



Cada um vê o seu

Projetos ligados à conta de quem gerou, guardados em armazenamento durável.

Útil para quem não programa, não só para o desenvolvedor



PO e analista

User story e critérios de aceitação prontos para o backlog.



QA

Casos de teste e critérios para validar a entrega.



Desenvolvedor

Análise, plano de solução e um projeto inicial de referência.



Gestor

Histórico do time, limites de uso e custo por geração.

Tudo pelo navegador: sem instalar ferramenta, sem acesso ao repositório e sem saber escrever um bom pedido para a IA.

Controle de uso e custo para quem paga a conta

5 • 100 • 1000

gerações por mês nos planos Free, Pro e Team, com o saldo sempre visível

- Painel do administrador para usuários, planos e acessos
- Fila de gerações: até duas por usuário, retomadas após reinício
- Assinatura mensal com Pix, boleto ou cartão, liberada automaticamente

US\$ / geração

tokens e custo de IA registrados em cada geração, por plano e por período

Lado a lado

Critério	Claude Code · Codex	MBA IA · Bug Evaluation
Quem usa	Desenvolvedor, no editor ou terminal	Qualquer pessoa do time, no navegador
Entrada	O código do projeto e a conversa	O texto do bug, como chegou no chamado
Saída	Alterações no código real	Pacote de seis entregas documentadas
Formato	Varia conforme o pedido	Sempre o mesmo, com seções obrigatórias
Registro	Na conversa de quem perguntou	Histórico por bug, com busca e filtros
Controle	Por conta individual	Limites por plano e custo por geração
Corrige o código real	Sim	Não: gera um projeto de referência

O que ainda não faz, e o que vem a seguir

Limites atuais

- Não lê o repositório do sistema com o bug
- O código gerado é sempre Python com FastAPI
- O projeto é um ponto de partida, não uma correção aplicada
- Os testes gerados não são executados automaticamente

Próximos passos possíveis

- Ler o código no GitHub ou GitLab, inclusive interno
- Escolher a linguagem da solução, como PHP
- Usar o Azure OpenAI da própria empresa
- Executar os testes gerados antes da entrega

Melhor usar os dois juntos



O MBA IA organiza e documenta o bug para o time; o assistente de código aplica a correção; os critérios e testes gerados dizem quando o trabalho está pronto.