

OS 10 PROJETOS DE PORTFÓLIO QUE FAZEM O RECRUTADOR TE CHAMAR.

Cada projeto com as funcionalidades obrigatórias, a stack sugerida e o diferencial. O README de cada projeto: o problema que ele resolve, como rodar, a arquitetura em um desenho, as decisões técnicas e o link do projeto no ar.

01 ENCURTADOR DE LINKS COM MÉTRICAS

Em vez de: ~~To-do list~~ · Prova: Banco, cache e API com muito acesso de leitura.

- ☐ Gerar link curto único e redirecionar em poucos milissegundos
- ☐ Cache dos links mais acessados no Redis
- ☐ Contagem de cliques por dia, país e dispositivo
- ☐ Limite de requisições por IP (rate limit)

Stack sugerida: Node.js ou Go, PostgreSQL, Redis, deploy em um serviço gratuito

Diferencial: Teste de carga com k6 mostrando quantas requisições por segundo ele aguenta.

Para começar: Documentação do Redis: cache e estruturas de dados · redis.io/docs/latest/

02 API DE PAGAMENTO COM WEBHOOK

Em vez de: ~~CRUD de produtos~~ · Prova: Integração com sistema externo, dinheiro e erro de rede.

- ☐ Checkout no modo de teste do Stripe
- ☐ Webhook que confirma o pagamento e atualiza o pedido
- ☐ Chave de idempotência: a mesma cobrança nunca é feita duas vezes
- ☐ Estados do pedido: criado, pago, falhou, reembolsado

Stack sugerida: Qualquer back-end, PostgreSQL, Stripe em modo de teste

Diferencial: Reprocessar webhooks que falharam sem duplicar nada.

Para começar: Stripe: requisições idempotentes · docs.stripe.com/api/idempotent_requests

03 PROCESSAMENTO EM FILA

Em vez de: ~~Formulário de contato~~ · **Prova:** Trabalho assíncrono, nova tentativa e tolerância a falha.

- ☐ Usuário pede um relatório e recebe a resposta na hora
- ☐ Um worker gera o relatório em segundo plano
- ☐ Nova tentativa automática quando falha, com limite
- ☐ Tela ou endpoint com o status de cada tarefa

Stack sugerida: Node.js com BullMQ ou Python com Celery, Redis

Diferencial: Painel com tarefas na fila, concluídas e com erro.

Para começar: Documentação do BullMQ · docs.bullmq.io/

04 ASSISTENTE DE IA SOBRE DOCUMENTOS

Em vez de: ~~Chatbot que só repassa a pergunta para a API~~ · **Prova:** IA usada como engenharia, não como mágica.

- ☐ Upload de PDFs e divisão do texto em partes
- ☐ Busca por similaridade com embeddings em um banco vetorial
- ☐ Resposta que cita o trecho e o documento de origem
- ☐ Custo e tempo de cada pergunta registrados

Stack sugerida: Python ou TypeScript, PostgreSQL com pgvector, API de um modelo de IA

Diferencial: 10 perguntas de teste com a resposta esperada para medir a qualidade.

Para começar: pgvector: busca vetorial no PostgreSQL · github.com/pgvector/pgvector

05 SISTEMA DE AGENDAMENTO

Em vez de: ~~Landing page sem back-end~~ · **Prova:** Concorrência, transação e regra de negócio de verdade.

- ☐ Profissionais com horários disponíveis e clientes que reservam
- ☐ Dois clientes nunca reservam o mesmo horário, nem ao mesmo tempo
- ☐ Fuso horário correto para quem está em outra cidade
- ☐ Cancelamento com regra de prazo e e-mail de confirmação

Stack sugerida: Qualquer back-end, PostgreSQL com transação ou restrição no banco

Diferencial: Teste automatizado que tenta reservar o mesmo horário em paralelo.

Para começar: PostgreSQL: restrições de exclusão (impedem horários sobrepostos) · www.postgresql.org/docs/current/ddl-constraints.html

06 CHAT EM TEMPO REAL

Em vez de: ~~Clone do WhatsApp só com a tela~~ · **Prova:** Comunicação em tempo real e estado compartilhado.

- ☐ Salas com mensagens entregues na hora via WebSocket
- ☐ Indicador de quem está online e de quem está digitando
- ☐ Histórico salvo no banco, com paginação
- ☐ Reconexão automática quando a internet cai

Stack sugerida: Node.js com Socket.IO ou WebSocket puro, Redis, PostgreSQL

Diferencial: Duas instâncias do servidor trocando mensagens pelo Redis.

Para começar: MDN: API de WebSockets (em português) · developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API

07 DASHBOARD COM DADOS PÚBLICOS REAIS

Em vez de: ~~Clone da Netflix~~ · **Prova:** Consumir API real, tratar dados e mostrar o que importa.

- ☐ Coleta diária de uma API pública (ex.: câmbio e Selic no Banco Central)
- ☐ Dados limpos e salvos no seu banco
- ☐ Gráficos com filtro de período
- ☐ Atualização automática agendada

Stack sugerida: Python ou TypeScript, PostgreSQL, uma biblioteca de gráficos

Diferencial: Uma conclusão escrita no topo: o que os dados mostram.

Para começar: Portal de Dados Abertos do Banco Central · dadosabertos.bcb.gov.br/

08 MONITOR DE SITES COM ALERTAS

Em vez de: ~~Calculadora~~ · **Prova:** Tarefas agendadas, notificações e observabilidade.

- ☐ Cadastro de URLs para verificar a cada minuto
- ☐ Histórico de tempo de resposta e de quedas
- ☐ Alerta por e-mail ou Telegram quando um site cai
- ☐ Página pública de status

Stack sugerida: Qualquer back-end, um agendador de tarefas, PostgreSQL

Diferencial: Porcentagem de disponibilidade (uptime) dos últimos 30 dias.

Para começar: API de bots do Telegram · core.telegram.org/bots/api

09 FERRAMENTA DE LINHA DE COMANDO PUBLICADA

Em vez de: ~~Mais um projeto que ninguém instala~~ · **Prova:** Empacotar, documentar, testar e versionar como open source.

- ☐ Resolve uma tarefa repetitiva sua (ex.: gerar changelog, renomear arquivos)
- ☐ Publicada no npm ou no PyPI, instalável com um comando
- ☐ README com exemplo de uso e GIF
- ☐ Testes e versão seguindo versionamento semântico

Stack sugerida: Node.js ou Python, GitHub Actions para publicar

Diferencial: Alguém que não é você instalando e abrindo uma issue.

Para começar: npm: como publicar um pacote · docs.npmjs.com/creating-and-publishing-unscooped-public-packages

10 PROJETO COM USUÁRIO DE VERDADE

Em vez de: ~~10 projetos que ninguém nunca usou~~ · **Prova:** Que você entrega, mantém e corrige com feedback real.

- ☐ Resolve um problema seu, da sua família ou do seu time
- ☐ Pelo menos 5 pessoas usando toda semana
- ☐ Bugs reportados e corrigidos, registrados nas issues
- ☐ Métrica simples de uso no README

Stack sugerida: A stack que você quer usar na próxima vaga

Diferencial: A história de um bug real que você encontrou, entendeu e corrigiu.

Para começar: GitHub: como escrever um bom README ·

docs.github.com/pt/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-readmes

Não faça os 10. Escolha 2 ou 3, termine, coloque no ar e escreva o README.
Projeto pela metade não conta.

Segue @lucasadvice