

Mauricio Zaffari

Lead / Staff Software Engineer | Rails & AI Systems

Brasil · UTC-3 · Remoto, com sobreposição diária ao horário comercial do Reino Unido e dos Estados Unidos

linkedin.com/in/mauriciozaffari · github.com/mauriciozaffari · rubygems.org/profiles/mauriciozaffari · mauriciozaffari@gmail.com

Dezoito anos construindo sistemas em Ruby on Rails, e ainda escolho Rails de propósito. Até setembro de 2026 liderei a engenharia de uma plataforma SaaS de analytics usada por Disney, Warner Bros, Sony e NBCUniversal, onde terminei como o maior contribuidor de todos os tempos do código principal. Estou aberto a posições de lead e staff de engenharia.

A maior parte do meu trabalho se parece com uma de três coisas: deixar um sistema lento rápido o bastante para as pessoas pararem de contorná-lo, quebrar um monolito em partes sem parar o negócio, ou ser a pessoa que decide qual vai ser a arquitetura e depois convive com essa decisão. Levei aplicações Rails da versão 2 até a versão 8, o que significa que migrei a maior parte dos erros que o framework desde então corrigiu.

Também construo e opero meu próprio produto, sozinho. O Inscripto é uma plataforma multi-tenant de eventos esportivos que lida com inscrições reais e pagamentos reais para ligas de kart e de tênis, o que me mantém honesto sobre a diferença entre um sistema que vai bem em uma demonstração e um sistema que sobrevive a uma manhã de sábado.

01 RESULTADOS

31%

MAIOR CONTRIBUIDOR DE TODOS OS TEMPOS

de todos os commits do código principal de analytics da Looper Insights em toda a história da empresa, e 81,7% dos commits de 2025

720.000+

DOWNLOADS DA GEM

de mongoid_search, a gem de busca full-text que criei e mantive

R\$ 482.800+

PAGAMENTOS PROCESSADOS

em 109 eventos esportivos e 3.603 cobranças no Inscripto, a plataforma que construí e ainda opero

18

ANOS ENTREGANDO RAILS

em produção, passando por todas as versões maiores, de Rails 2 a Rails 8

99, 1%

COBERTURA DE TESTES

cobertura final do código principal de analytics da Looper Insights quando eu saí em setembro de 2026, aproximadamente 244.000 linhas de Ruby

02
EXPERIÊNCIA

Looper Insights

Lead Software Engineer

julho de 2018 – setembro de 2026 · Londres, Reino Unido (remoto)

A Looper Insights é uma plataforma global SaaS de analytics para estúdios e serviços de streaming. Estive lá em três passagens: desenvolvedor full-stack em 2018, desenvolvedor sênior a partir de 2020 e tech lead de março de 2022 até setembro de 2026. Terminei como o maior contribuidor de todos os tempos do código principal, que atende Disney, Warner Bros, Sony e NBCUniversal e responde por 55% da receita da empresa. Liderei a troca de plataforma do Webstores 2.0 e a migração de autenticação de toda a empresa para o AWS Cognito.

Inscripto

Solo Founder, Arquiteto de Produto e Lead Engineer

2018 – atual · Brasil

O Inscripto é uma plataforma multi-tenant de gestão e inscrição em eventos esportivos que construí e ainda opero como sua única pessoa — produto, arquitetura, engenharia e operações de ponta a ponta —, substituindo fichas de papel e pagamentos manuais em ligas de kart e de tênis. Já processou 109 eventos, 3.603 cobranças e mais de R\$ 482.800 em pagamentos online. A stack é Rails 8, PostgreSQL, Hotwire, Tailwind e um frontend PWA, com Stripe, MercadoPago e PIX para pagamentos, e usa a API do Gemini para transformar resultados de partidas enviados soltos pelo WhatsApp em classificações atualizadas.

Develoz

Fundador e Engenheiro Principal

outubro de 2013 – atual · Brasil

Fundei a Develoz em 2013. Há treze anos ela desenvolve software, integra sistemas e automatiza processos para operações no Brasil e no exterior. Hoje trabalha com IA aplicada — identificar onde a IA vale a pena em um processo, construir o piloto de integração e definir a governança e a revisão humana em volta dele. Entre os clientes anteriores estão ArtistSignal, NY Water Taxi e Explica Fácil.

ATUAÇÕES ANTERIORES

Kinscape

Tech Lead de Desenvolvimento

junho de 2018 – março de 2020 · Remoto

Liderei a modernização técnica de uma rede social privada para famílias, desacoplando um monolito legado em Rails e AngularJS em um frontend Vue.js que consome uma API JSON em Rails.

StudioBridge

Tech Lead Full-Stack

janeiro de 2018 – março de 2018 · Alpharetta, GA, EUA (remoto)

Migrei o motor de cobrança de assinaturas do produto de Stripe para CardConnect ao longo de mais de 100 commits, sem downtime para os estúdios de dança e de música em atividade.

Atlantic Coast Conference

Consultor Sênior de Software

setembro de 2017 – janeiro de 2018 · Charlotte, NC, EUA (remoto)

Construí um motor dinâmico de modelos de avaliação multiesportiva em Node.js, AngularJS e PostgreSQL, usado nas avaliações de árbitros e de instalações da conferência da NCAA.

Habit

Engenheiro Full-Stack Sênior

novembro de 2016 – setembro de 2017 · Região da Baía de São Francisco, CA, EUA (remoto)

Construí software de operação de cozinha e de logística em Rails, React e PostgreSQL para uma startup de nutrição personalizada, transformando resultados de exames de DNA e de sangue em fluxos diários de preparo e entrega de refeições. A empresa foi depois adquirida pela Viome.

Decisiv

Engenheiro de Software Sênior

2016

Construí um motor de recomendação de manutenção em Rails e MySQL para uma plataforma de gestão do relacionamento de serviços de frotas comerciais que processava mais de 4 milhões de eventos por ano em 74.000 frotas.

Alldayeveryday

Engenheiro Full-Stack Sênior

2015 – 2016

Reduzi o tempo de carregamento das páginas de uma revista digital de 5-10 segundos para menos de um segundo, com ajuste de consultas e cache. Também integrei a cobrança via Chargebee e introduzi a primeira suíte de testes emRSpec.

Everwise

Engenheiro Full-Stack Sênior

janeiro de 2013 – novembro de 2016 · São Francisco, CA, EUA (remoto)

Em duas passagens, ajudei a construir a terceira geração do motor de pareamento entre mentores e mentorados em Rails e PostgreSQL, atendendo mais de 60.000 relacionamentos, e desenvolvi uma biblioteca reutilizável de componentes de interface em React. A empresa depois se fundiu com a Torch.

Toptal

Engenheiro de Software Sênior

2012 – 2013

Passei pelo processo seletivo de cinco etapas e depois ajudei a construir a própria plataforma de pareamento e de gestão de projetos da Toptal.

Stack Builders, Bit Zesty e BlockShopper

Desenvolvedor Ruby on Rails

2008 – 2012

Onde minha carreira profissional com Rails começou, em 2008. Liderei o MiMo, uma aplicação de pagamentos no estilo PayPal para a Nigéria; construí o Feesability, um produto de controle de horas em Rails 3, Mongoid e Resque; e construí um CMS de notícias do mercado imobiliário que cobria mais de 32 sites.

03

TRABALHOS SELECIONADOS

Colocando os primeiros recursos de IA em uma plataforma de clientes

Looper Insights · 2026

PROBLEMA

A empresa queria recursos de IA no produto voltado aos clientes, e tinha seis engenheiros em quem confiava para descobrir onde a IA realmente se paga. Dentro da engenharia, assistentes de código com IA já estavam mudando a forma de trabalhar, sem padrões compartilhados: cada engenheiro inventava seu próprio workflow.

ABORDAGEM

Eu era um dos seis no piloto de IA da empresa, e assumi as duas metades do problema — como construímos com IA, e o que entregamos com IA.

Para a primeira metade, escrevi o padrão de workflow AGENTS.md do time e o compartilhei com toda a organização, de modo que o desenvolvimento guiado por assistentes saiu de hábito pessoal para uma forma de trabalho documentada e revisável. Publiquei avaliações de modelos que informaram a escolha de ferramentas da organização.

Para a segunda metade, fui delivery lead da iniciativa de IA no produto principal: páginas de IA voltadas aos clientes, incluindo uma autenticada pelo single sign-on de um estúdio, e a arquitetura de segurança de dados multi-tenant que permite um

recurso de chat com LLM responder a partir dos dados de um cliente sem que os dados de outro cliente estejam ao alcance. O isolamento foi projetado antes do recurso ser aprovado, não retroajustado depois.

RESULTADO

Padrões adotados por toda a organização; páginas de IA no ar na frente de clientes cujos nomes todos reconhecem; uma arquitetura de chat cujo modelo de segurança foi revisado antes de entrar em produção.

TECNOLOGIAS

Ruby on Rails · Integrações de LLM · ClickHouse

Reduzindo uma importação de 46 minutos para 16 minutos

Looper Insights · 2026

PROBLEMA

As importações de merchandising da Amazon US levavam cerca de 46 minutos. As entregas para clientes ficavam na fila atrás delas, então uma única importação lenta atrasava os relatórios que as pessoas realmente estavam esperando. O volume era a parte difícil: aproximadamente 273.000 posições de título por scan, contra 12,5 a 13 milhões de títulos processados por semana.

ABORDAGEM

O gargalo não era a busca dos dados, era o caminho de escrita — cada linha pagava por durabilidade e manutenção de índice de que não precisava enquanto ainda estava em trânsito. Movi a ingestão para tabelas de staging UNLOGGED no PostgreSQL, de modo que as linhas em andamento não passam pelo write-ahead log, e depois promovi essas linhas para as tabelas reais quando a importação terminava. Junto com isso, paralelizei a etapa de busca e passei a transmitir os resultados por Redis em vez de acumulá-los na memória do processo.

As tabelas de staging são seguras aqui justamente porque o dado é reconstruível: uma tabela UNLOGGED não sobrevive a uma queda do banco, e essa é uma troca aceitável quando a ação de recuperação é rodar a importação de novo.

RESULTADO

De 46 minutos para cerca de 16 — uma redução de 65%, confirmada de forma independente por um colega que mediu o mesmo pipeline. As entregas para clientes deixaram de ficar na fila atrás da janela de importação.

TECNOLOGIAS

Ruby on Rails · PostgreSQL · Redis

Uma página que levava quatro minutos, e os cinco motivos

Looper Insights · 2026

PROBLEMA

A principal página de scan levava mais de quatro minutos para carregar. Para um

cliente era pior: de dez a quinze minutos, tempo suficiente para as pessoas deixarem de usar o recurso e pedirem a outra pessoa que mande uma planilha no lugar.

ABORDAGEM

Não havia uma causa única, e essa é a parte interessante. Cinco coisas distintas se acumulavam: um `COUNT DISTINCT` se espalhando sobre um join, um scope que varria a tabela inteira em vez de fatia daquela requisição, um sequential scan sobre 16,7 milhões de linhas, um join contra uma tabela de aproximadamente 1,5 bilhão de linhas e a falta de cache em um agregado agrupado caro.

Corrigi uma por vez e medi depois de cada correção, porque causas que se acumulam mascaram umas às outras — corrija a segunda pior primeiro e a melhora parece desprezível, o que tenta você a concluir que ali não estava o problema. Duas das cinco pediam desnormalização em vez de ajuste de consulta: um booleano levado para a linha que precisava dele, o que fez o join de um bilhão de linhas desaparecer, e um agregado `JSONB` em cache, atualizado por um worker dedicado em vez de calculado a cada requisição.

Para o cliente mais afetado, a correção foi mais estreita e saiu no mesmo dia: uma verificação de existência que fazia trabalho linear passou a ser uma busca $O(1)$.

RESULTADO

De quatro minutos para menos de um segundo. O caminho específico daquele cliente saiu de dez a quinze minutos para praticamente instantâneo, entregue como hotfix no mesmo dia. Um filtro relacionado na mesma página saiu de 322 segundos a frio para 25,5 segundos, ancorando uma subconsulta sem escopo para que ela pudesse usar um índice que já existia.

Depois fiz essa classe inteira de regressão falhar de forma ruidosa, habilitando `strict_loading` em toda a suíte de testes, para que um N+1 acidental quebre o CI em vez de chegar silenciosamente à produção.

TECNOLOGIAS

Ruby on Rails · PostgreSQL · ActiveRecord

Trocando a plataforma do produto principal sem pará-lo

Looper Insights · 2024 - 2025

PROBLEMA

O produto principal tinha acumulado a década habitual de decisões: um frontend React embutido dentro do monolito Rails, de modo que os dois não podiam ser implantados nem testados de forma independente, e uma parcela grande do código anterior aos padrões que o time depois adotou. Reescrever tudo de uma vez não era opção — ele atende Disney, Warner Bros, Sony e NBCUniversal e responde por 55% da receita da empresa.

ABORDAGEM

Conduzi a troca de plataforma como uma sequência de passos entregáveis, e não como um branch que vivesse por seis meses. Em 261 commits sobre 844 arquivos, reescrevi mais de 20% de uma base de código de aproximadamente 244.000 linhas, elevando a cobertura de testes conforme avançava, porque eram os testes que tornavam o passo seguinte seguro.

Em paralelo, extraí o frontend — cerca de 138.759 linhas de React — do monólito para um repositório próprio, o que permitiu que frontend e backend passassem a ser construídos, testados e implantados em cadências próprias.

RESULTADO

Entregue de forma incremental, com a cobertura de testes saindo de 72% para 80% ao longo do trabalho e continuando a subir depois. O product manager descreveu o resultado como um release transformador. A extração do frontend é o que tornou o trabalho de frontend seguinte independente dos ciclos de release do backend.

TECNOLOGIAS

Ruby on Rails · React · PostgreSQL · RSpec

Sessenta releases e um botão de release

Looper Insights · 2022 - 2026

PROBLEMA

Os releases eram manuais e tribais: quem conhecia os passos os executava, e os passos viviam na cabeça de alguém. O produto atende Disney, Warner Bros, Sony e NBCUniversal, então um release ruim não é um inconveniente interno — é um incidente visível para o cliente. Dois repositórios viraram três quando o frontend foi extraído, o que triplicou a cerimônia.

ABORDAGEM

Eu conduzia o trem de releases de ponta a ponta nos três repositórios: um code freeze quinzenal que eu instituí, gating de QA, deploys em produção, hotfixes e um plano de rollback para cada release. Depois transformei as partes que uma máquina consegue fazer em uma máquina fazendo: um workflow automatizado de release que builda, roda os gates de qualidade, faz deploy e verifica a saúde. O gate são nove linters mais a suíte end-to-end de Cypress ligada aos deploys e aos merges de pull requests, então as verificações que dependiam de alguém lembrar agora são o pipeline se recusando a continuar. Os templates de plano de teste e checklist de rollback que escrevi se tornaram os que PM e QA reproduziram para cada release seguinte.

Por um período também fui release manager de um código frontend que eu não desenvolvia, porque conduzir o trem é uma habilidade separada de escrever o código que ele entrega.

RESULTADO

Mais de sessenta releases e hotfixes entregues ao longo de quatro eras de nomenclatura do produto, sem o processo de release ser o gargalo. Os releases pausavam quando eu estava de férias — a evidência mais forte que posso dar de que o processo funcionava e de que eu ainda não tinha terminado de torná-lo independente de mim.

TECNOLOGIAS

Ruby on Rails · RSpec · Cypress · GitHub Actions

Levando todos os usuários para uma autenticação gerenciada

Looper Insights · 2021 - 2022

PROBLEMA

A autenticação era feita em casa, e os clientes corporativos começavam a pedir coisas em que autenticação feita em casa é ruim: identidade federada, o single sign-on deles e um tratamento de tokens que pudéssemos mostrar durante uma revisão de segurança.

ABORDAGEM

Liderei a migração de toda a empresa para o AWS Cognito ao longo de cerca de dois meses: integração OIDC, verificação de tokens bearer, tratamento de refresh e sessões guardadas em Redis com persistência em banco, para que um restart não desconectasse todo mundo.

A parte que exige cuidado não é o sistema novo, é a transição. Todo usuário existente tinha de continuar funcionando o tempo todo, o que significou rodar os dois caminhos contra tráfego real antes de o código legado poder ser removido — e então removê-lo de verdade, em vez de deixar um ramo morto para confundir quem fosse ler depois.

RESULTADO

Um único provedor de identidade gerenciado para toda a plataforma, com o SSO de cliente passando a ser um exercício de configuração e não um projeto de engenharia. Depois construí sobre essa base as páginas voltadas ao cliente, autenticadas pelo próprio single sign-on de um estúdio.

TECNOLOGIAS

Ruby on Rails · AWS Cognito · OIDC · Redis

04

COMO EU TRABALHO

Liderei um time pequeno de engenharia e nunca parei de escrever código.

Permaneci o maior contribuidor da base de código que liderei, e essa combinação foi deliberada, não uma falha em delegar: revisava código que eu mesmo poderia ter escrito, e não impunha uma arquitetura que eu não estivesse disposto a construir. A alavancagem funcionou nas duas direções — pelas decisões que as outras pessoas herdam e pelo trabalho que eu entrego.

Encontro a causa antes de escrever a correção. Uma página lenta normalmente tem mais de um motivo para ser lenta, e o primeiro que você encontra raramente é o mais caro. Eu faço profiling, leio o plano de execução da consulta e corrijo a coisa de verdade. Uma página que herdei levava mais de quatro minutos para carregar por cinco motivos que se acumulavam; corrigindo um por vez, com uma medição depois de cada um, ela ficou abaixo de um segundo. Adivinhar teria corrigido um dos cinco.

Um teste de regressão que passa antes da correção não vale nada. Já rejeitei testes meus por isso. Um deles inseria um valor em branco por um caminho de escrita do ActiveRecord que silenciosamente o convertia em `nil`, então o teste nunca exercitava o bug que dizia cobrir. Isso é falsa confiança, e é pior do que não ter teste, porque impede que alguém volte a olhar.

A integridade fica no banco de dados quando pode ficar. Verificações de unicidade na aplicação perdem corridas. Se uma duplicata nunca pode existir, prefiro deduplicar uma vez, criar o índice único e deixar o sistema falhar de forma fechada a partir dali.

Code review é onde os padrões de fato se propagam. Comentário sobre rigor de teste e formato da arquitetura, não sobre formatação — formatação é trabalho do linter. Um princípio ao qual sempre volto: um controller sem um model correspondente é um mau sinal, porque uma mudança de estado em um registro é quase sempre o update daquele registro, e não um controller novo.

Releases precisam de um plano de rollback que alguém realmente pensou. Eu cuidava dos checklists de release: reversibilidade das migrações, drenagem das filas, uma referência registrada do último estado bom conhecido. Quando nossa ferramenta de release encontrou um padrão de merge que tornava o rebase inseguro, corriji a ferramenta e escrevi o motivo, para que a próxima pessoa encontre um modo de falha documentado em vez de um mistério.

Eu registro as coisas por escrito. Decisões de arquitetura, notas de domínio sobre os subsistemas que surpreendem as pessoas e os padrões de agentes de IA que as organizações que liderei seguem. Uma decisão que vive apenas em uma thread do Slack precisa ser rediscutida cada vez que alguém novo pergunta.

Eu ensino o sistema que lidero. Construí um currículo interno de treinamento e conduzi as sessões “Webstores Unlocked” que percorrem com o time como cada parte da plataforma funciona. Essas sessões gravadas se tornaram o material de onboarding para os novos engenheiros — os que eu contratei, integrei e avaliei.

05
CÓDIGO ABERTO

mongoid_search

Criador e mantenedor · 720.962 downloads

Busca full-text para Mongoid, e a gem que a maioria das aplicações Rails sobre MongoDB usava antes de o Atlas Search existir. Já passou de 720.000 downloads.

Acabei transferindo a gem para a organização oficial do Mongoid, que é o desfecho que eu queria: ela sobreviveu à minha própria necessidade e agora está onde quem procura por ela realmente vai encontrar.

mongoid_activity

Criador · 5.216 downloads

Registro de atividade para documentos do Mongoid, extraído de uma aplicação que precisava de uma trilha de auditoria e de um feed de mudanças sem um segundo datastore.

Ecossistema Ruby e Rails

Contribuidor

Correções e manutenção em gems das quais eu dependia em produção, incluindo o próprio Rails, `active_admin`, `active_model_serializers`, `cancan`, `carrierwave`,

administrate, blueprinter, cardconnect e cognito_token_verifier.

Quase nada disso foi contribuição planejada. É o que acontece quando você encontra um bug no trabalho, descobre que a causa está em uma dependência e decide corrigir upstream em vez de contornar com um monkey patch.

06
COMPETÊNCIAS

SISTEMAS DE IA E LLM

Integrações de LLM em produção · OpenAI · Anthropic Claude · Google Gemini · Pipelines assíncronos de LLM · Padrões de fluxo de trabalho para agentes · Isolamento de dados multi-tenant em recursos de LLM

BACKEND E ARQUITETURA

Ruby on Rails (v2 a v8) · Ruby · APIs REST · GraphQL · Microsserviços · Sidekiq · ActiveJob · Máquinas de estado · Modelagem de domínio

BANCOS DE DADOS E PERFORMANCE

Ajuste de consultas no PostgreSQL · Tabelas de staging UNLOGGED · Migrações de dados sem downtime · MySQL · Redis · OpenSearch · Elasticsearch

ENTREGA E TESTES

Docker · AWS (Cognito, ECS, S3) · GitHub Actions · RSpec · Capybara · Cypress · Quality gates com múltiplos linters · Administração de Linux

LIDERANÇA DE ENGENHARIA

Liderança de times técnicos · Governança de arquitetura · Code review · Contratação e entrevistas · Gestão de releases · Mentoria · Agile e Shape Up

ENGENHARIA DE FRONTEND

Hotwire (Turbo e Stimulus) · React · Vue.js · TypeScript · JavaScript · Tailwind CSS · Propshaft · esbuild · Progressive web apps

FORMAÇÃO

Tecnólogo em Sistemas para Internet	2011
Faculdade Dom Bosco	

Técnico em Eletrônica	2005
Instituto Anglicano Barão do Rio Branco	

Conteúdo revisado em 5 de outubro de 2026

A versão atual deste documento está em mauricio.zaffari.casa/pt-BR