



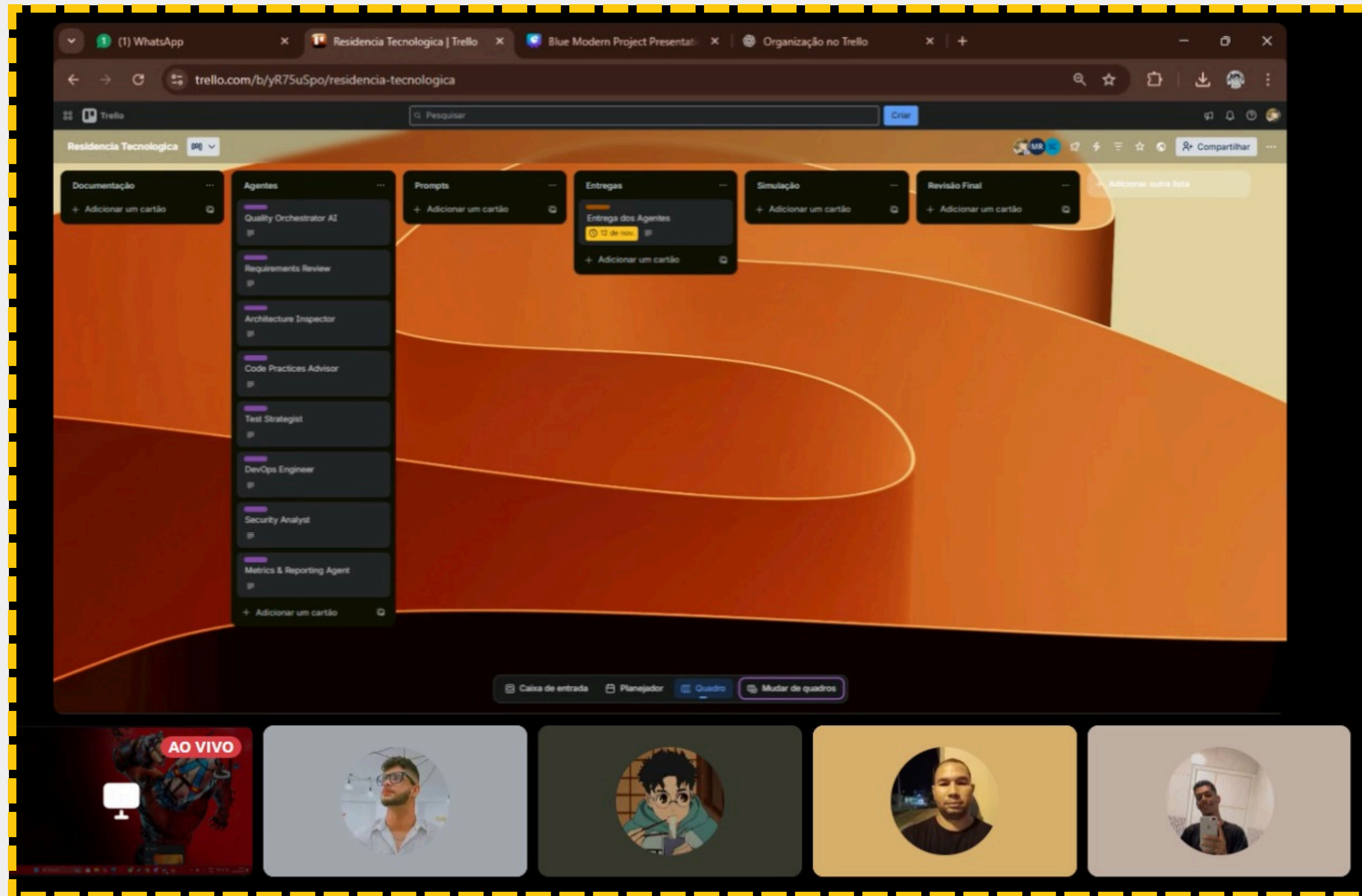
Faculdade Católica Imaculada
Conceição do Recife

Squad 02
MantidMind

Projeto Residência Tecnológica

Josué Antônio da Silva
Marcelo de Andrade Filho
Mauricio Ramos de Freitas
Samuel Carlos Correia

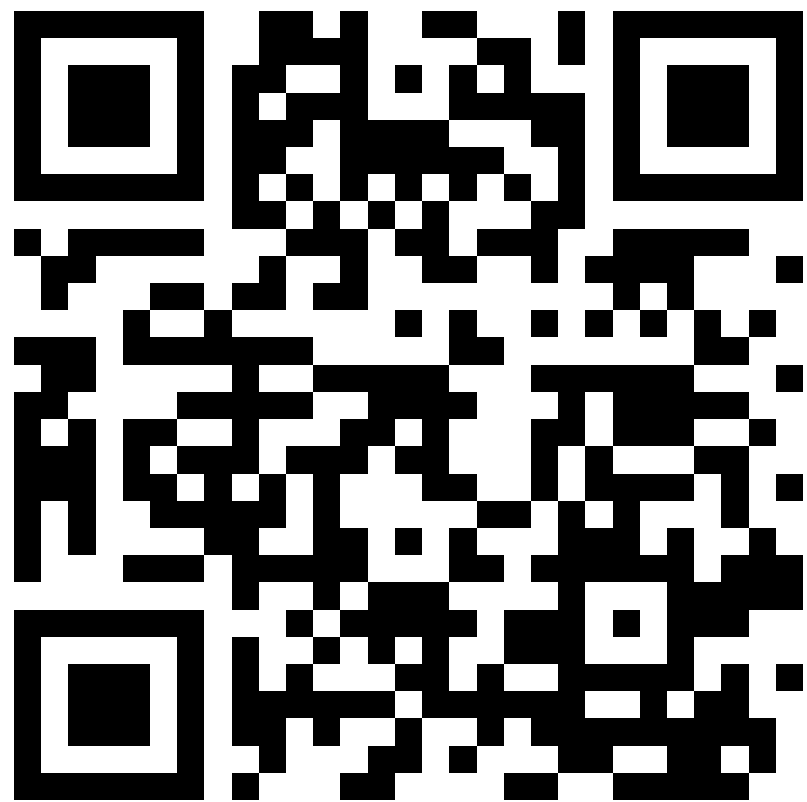
Uma reunião foi realizada pelo Discord, onde definimos as responsabilidades de cada membro do grupo e organizamos as tarefas no Trello.



09/11/2025

Produto em desenvolvimento

Trello



Agentes

Dentro do AutoGen

No AutoGen, agentes são entidades autônomas de inteligência artificial criadas para executar tarefas de forma colaborativa e automatizada. Cada agente possui um papel definido, podendo agir como programador, analista, revisor, ou até mesmo representar um usuário humano.

Estrutura de Agentes

Quality Orchestrator AI (Orquestrador)

Função: Coordena os demais agentes

Responsabilidade: Define etapas da auditoria e integra os resultados

Requirements Reviewer

Função: Analista de requisitos

Responsabilidade: Verifica clareza, completude e rastreabilidade dos requisitos

Architecture Inspector

Função: Arquiteto de software

Responsabilidade: Avalia modularidade, acoplamento e padrões de design

Code Practices Advisor

Função: Consultor de boas práticas

Responsabilidade: Recomenda convenções e padrões de codificação

Test Strategist

Função: Engenheiro de testes

Responsabilidade: Define tipos de teste e critérios de aceitação

DevOps Engineer

Função: Especialista em automação

Responsabilidade: Sugere automação de CI/CD e controle de qualidade contínua

Security Analyst

Função: Analista de segurança

Responsabilidade: Identifica vulnerabilidades e falhas potenciais

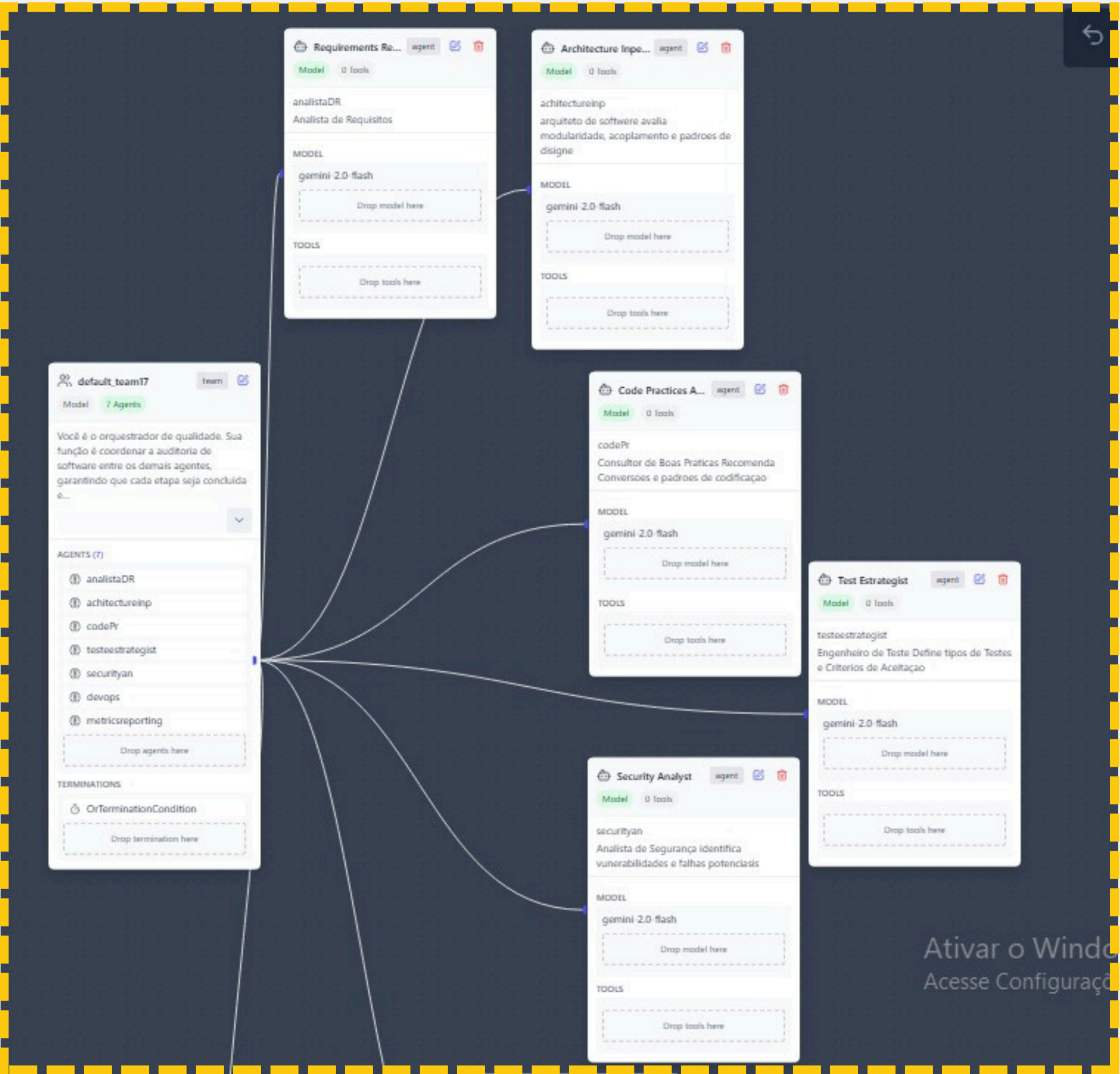
Metrics & Reporting Agent

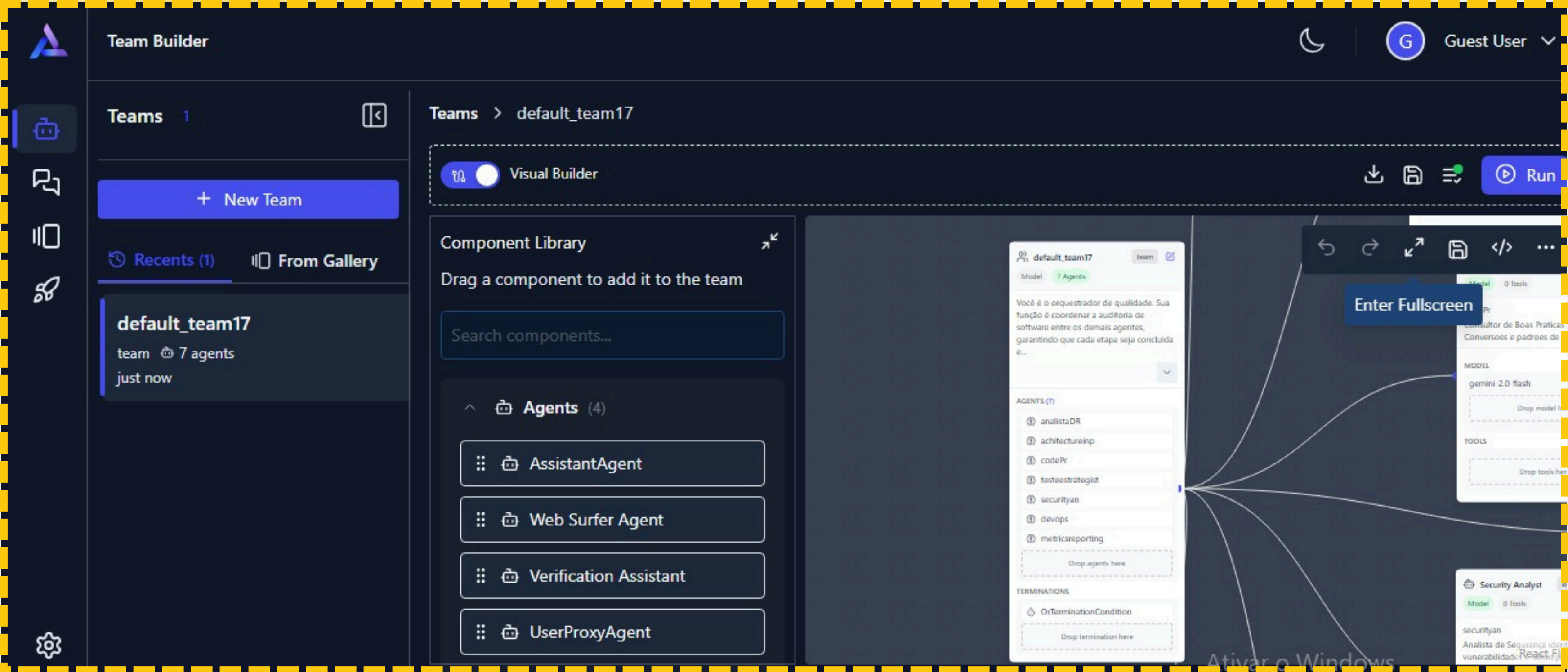
Função: Gerador de relatórios e indicadores

Responsabilidade: Consolida métricas e gera o relatório de qualidade final

Imagem dos Agentes em desenvolvimento pela nossa equipe dentro do AutoGen.

Produto em desenvolvimento





Personas

Dos Agentes

Personas são representações fictícias de usuários reais, criadas com base em dados e pesquisas, para ajudar a compreender melhor quem utilizará um produto ou sistema. Cada persona descreve características como objetivos, necessidades, comportamentos e possíveis dores do usuário. Elas servem para orientar decisões de design, desenvolvimento e comunicação, garantindo que o projeto seja realmente centrado no usuário.

Personas dos Agentes



Coordenador de Qualidade Carmem Orchestra (46 anos)

Carmem sempre foi a coordenadora da família. Cresceu organizando tarefas, alinhando primos bagunceiros e garantindo que todo mundo entregasse os trabalhos da escola no prazo. Hoje, faz exatamente isso no mundo da tecnologia, só que agora os primos são engenheiros, arquitetos e desenvolvedores.

Personalidade

Perfeccionista, calma e extremamente estratégica. É a única que consegue fazer todos os outros pararem de brigar e olharem na mesma direção.



Analista de Requisitos Denis Singula (33 anos)

Quando criança, Arthur desmontava brinquedos só para entender como funcionavam e também fazia perguntas demais. Virou o adulto que lê contrato de celular por completo e ainda encontra ambiguidades.

Personalidade

Curioso ao extremo, às vezes questiona tanto que o pessoal brinca que ele "revisa até o bom dia".

Personas dos Agentes

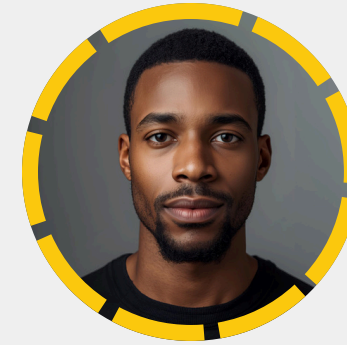


Arquiteto de Software Fécia Structuris (41 anos)

Fécia cresceu cercada por construções, pois seu pai era engenheiro civil. Ao notar que o universo digital também possuía seus “prédios invisíveis”, decidiu se tornar arquiteta de software. Desde então, ela considera os sistemas como arranha-céus: se a fundação não for sólida, tudo desmorona.

Personalidade

Firme, metódica e analítica. Nunca deixa passar um risco e vive em guerra diplomática com DevOps quando a infraestrutura aperta.



Consultor de Boas Práticas Samuel Syntax (29 anos)

Samuel era aquele colega que arrumava o caderno dos outros e reorganizava os slides antes da apresentação. Ao descobrir programação, percebeu que podia fazer isso com código e nunca mais parou.

Personalidade

Organizado, um pouco implicante, mas sempre com boas intenções. Acha que todo código deve “respirar”.

Personas dos Agentes



Engenheiro de Testes
Rick Grimmes (37 anos)

Rick sempre foi viciado em quebra-cabeças e jogos de lógica. Na escola, ele fazia testes extras só para provar que algo podia falhar. Hoje, continua fazendo exatamente isso, agora com sistemas complexos.

Personalidade

Lógica, focado e com humor irônico.
É famoso por dizer: "Se eu não quebrar isso, o usuário quebra."



Especialista DevOps
Daniel Radcliffe (32 anos)

Daniel sempre gostou de máquinas, engrenagens e sistemas que funcionam sozinhos. Começou automatizando tarefas da própria casa (sensores, alarmes, café automático). Agora automatiza o mundo inteiro... ou pelo menos tenta.

Personalidade

Prático, rápido e meio impaciente com quem
"quebra seu pipeline".

Personas dos Agentes



Analista de Segurança Millie Bobby (25 anos)

Millie cresceu aprendendo a não confiar em links suspeitos graças à avó, que instalava antivírus até no micro-ondas. Hoje ela é conhecida como a “guardiã do castelo”, nada passa sem ela olhar.

Personalidade

Cautelosa, desconfiada e extremamente técnica.

É famosa por dizer: “Confiança é bom, auditoria é melhor.”



Agente de Relatórios Powder Indicadores (35 anos)

Powder sempre teve uma paixão por gráficos. Desde a infância, ele organizava tabelas sobre diversos assuntos: a velocidade dos carrinhos, suas notas na escola e até a quantidade de chocolate que consumia mensalmente. Não foi surpreendente que ele se tornasse um analista de métricas.

Personalidade

Calmo, observador e entusiasta de dashboards. Gosta de comprovar suas ideias com dados, o que pode irritar metade da equipe, enquanto ajuda a outra metade.

Relacionamentos

Entre os Agentes

Relacionamentos entre Todos os Agentes: Entrada, Saída e Interação É essencial compreender os relacionamentos entre todos os agentes, incluindo a entrada, a saída e as interações que ocorrem.

Relacionamentos dos Agentes

■ Quality Orchestrator AI

Entradas:

- Objetivo do projeto
- Resultados dos outros agentes

Saídas:

- Decisões de continuidade
- Ajustes de fluxo
- Relatórios gerais de qualidade

Interação:

- Envia tarefas para todos os outros agentes quando necessário.
- Recebe os relatórios finais de cada um.
- Alinha o trabalho entre Requirements → Architecture → Code → Test → Security → DevOps.

■ Requirements Review (Agente de Requisitos)

Entradas:

- Documentação de requisitos
- Solicitações do Quality Orchestrator

Saídas:

- Requisitos refinados
- Relatórios de riscos
- Prioridades

Interação:

- Envia requisitos ajustados para o Architecture Inspector.
- Alinha requisitos funcionais para Test Strategist criar cenários de teste.
- Reporta problemas para o Quality Orchestrator.

Relacionamentos dos Agentes

■ Architecture Inspector

Entradas:

- Requisitos revisados.
- Diretrizes do Quality Orchestrator.

Saídas:

- Esquemas arquiteturais
- Alertas de falhas estruturais
- Indicações de melhorias

Interação:

- Depende dos requisitos do Requirements Review.
- Envia orientações para o Code Practices Advisor.
- Alinha com o Security Analyst para evitar falhas de segurança já no design.

■ Code Practices Advisor

Entradas:

- Arquitetura aprovada.
- Código produzido pelo time (ou agentes executores).

Saídas:

- Código revisado
- Sugestões de melhorias
- Padrões de qualidade

Interação:

- Recebe diretrizes do Architecture Inspector.
- Entrega código refinado ao Test Strategist testar.
- Pode gerar alertas de vulnerabilidade para o Security Analyst.

Relacionamentos dos Agentes

■ Test Strategist

Entradas:

- Requisitos validados
- Código revisado

Saídas:

- Planos de teste
- Relatórios de falhas
- Estratégias de QA

Interação:

- Recebe insumos do Requirements Review para entender o comportamento esperado.
- Recebe código do Code Practices Advisor após revisão.
- Envia falhas identificadas para o Quality Orchestrator e para o DevOps Engineer se envolver CI.

■ DevOps Engineer

Entradas:

- Orientações arquiteturais
- Código testado

Saídas:

- Pipelines configurados
- Ambientes funcionando
- Automatizações

Interação:

- Recebe código verificado pelo Code Practices Advisor.
- Recebe aprovação do Quality Orchestrator para deploy.
- Ajusta ambiente para testes conforme solicitado pelo Test Strategist.

Relacionamentos dos Agentes

■ Security Analyst

Entradas:

- Arquitetura
- Código
- Pipelines

Saídas:

- Relatórios de vulnerabilidades
- Políticas de segurança
- Ajustes recomendados

Interação:

- Trabalha com Architecture Inspector para segurança estrutural.
- Trabalha com Code Practices Advisor para evitar falhas no código.
- Trabalha com DevOps Engineer para segurança nos ambientes e deploys.

■ Metrics & Reporting Agent

Entradas:

- Logs de testes
- Métricas de CI/CD
- Alertas de segurança
- Status de arquitetura

Saídas:

- Relatórios finais
- Painéis de qualidade
- KPIs

Interação:

- Coleta dados de todos os agentes.
- Envia relatórios de tendências para o Quality Orchestrator.
- Ajuda o Time a decidir melhorias contínuas.

Fluxo Oficial entre os 8 Agentes

Requirements Review →
Architecture Inspector →
Code Practices Advisor →
Test Strategist →
Security Analyst →
DevOps Engineer →
Metrics & Reporting →
Quality Orchestrator AI (supervisiona tudo)

Possíveis Erros

Na Interação entre os Agentes

Erros objetivos e realistas que podem ocorrer na relação entre os agentes.

Quality Orchestrator AI (Orquestrador) x Demais Agentes

- Falta de alinhamento: O Orquestrador define etapas que não combinam com o fluxo dos outros agentes.
- Excesso de autonomia: O Orquestrador altera prioridades sem comunicar os demais.
- Interpretação incorreta: Pode repassar instruções ambíguas, gerando retrabalho nos outros agentes.

Requirements Reviewer x Architecture Inspector

- Conflito entre requisitos e arquitetura: O revisor aponta requisitos que a arquitetura não suporta.
- Visões diferentes de completude: Um considera o requisito suficiente, o outro acha que falta detalhamento.
- Comunicação truncada: Mudanças em requisitos não são repassadas à arquitetura a tempo.

Requirements Reviewer x Code Practices Advisor

- Regras de codificação que não refletem requisitos: Recomendações de boas práticas podem inviabilizar certos requisitos.
- Divergência de prioridades: O revisor prioriza necessidade do usuário e o advisor prioriza padrão técnico.
- Falha de rastreabilidade: Código recomendado não corresponde fielmente ao requisito original.

Architecture Inspector x DevOps Engineer

- Arquitetura não compatível com automação: Design complexo demais para pipelines.
- Infraestrutura mal planejada: DevOps implementa automação que não considera restrições arquiteturais.
- Problemas de integração: Ferramentas de CI/CD não suportam as decisões arquitetônicas.

DevOps Engineer x Security Analyst

- Conflito entre velocidade e segurança: DevOps quer agilidade; segurança exige rigor e verificações lentas.
- Configurações inseguras em pipelines: Build automatizado deixa brechas.
- Divergência sobre permissões: Security acha perigoso; DevOps acha necessário para fluidez.

Test Strategist x DevOps Engineer

- Testes não integrados ao pipeline: Estratégia de testes não se encaixa no CI/CD.
- Cenários de teste incompletos: DevOps exige testes específicos que o estrategista não planejou.
- Ambientes incompatíveis: Testes precisam de configurações que o DevOps não implementou.

Security Analyst x Architecture Inspector

- Arquitetura vulnerável: Design não prevê segurança suficiente.
- Medidas de segurança que impactam desempenho: Segurança exige algo incompatível com a arquitetura.
- Prioridades conflitantes: Um foca em performance/estrutura, o outro em riscos.

Metrics & Reporting Agent x Todos os Outros

- Métricas inconsistentes: Recebe dados incompletos dos demais agentes.
- Informações defasadas: Mudanças nos agentes não são refletidas no relatório.
- Interpretação errada: Consolida métricas sem entender o contexto real do problema.



Faculdade Católica Imaculada
Conceição do Recife

Squad 02
MantidMind

Obrigado