v7.n11.2026

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

PRINT3D MANAGER: DESENVOLVIMENTO DE UM ERP PARA EMPRESAS DE IMPRESSÃO 3D COM APOIO DE AGENTES DE INTELIGÊNCIA ARTIFICIAL

PRINT3D MANAGER: DEVELOPMENT OF AN ERP FOR 3D PRINTING COMPANIES WITH SUPPORT FROM ARTIFICIAL INTELLIGENCE AGENTS

PRINT3D MANAGER: DESARROLLO DE UN ERP PARA EMPRESAS DE IMPRESIÓN 3D CON APOYO DE AGENTES DE INTELIGENCIA ARTIFICIAL

João Antônio Oliveira Barroso[1]; Felipe Diniz Dallilo[2]

e7119565
https://doi.org/10.47820/recima21.v7i11.9565

PUBLICADO: 11/2026

RESUMO

Este trabalho apresenta o Print3D Manager ERP, desenvolvido para empresas de impressão 3D, e analisa o uso de agentes de inteligência artificial na revisão do software. Trata-se de uma pesquisa aplicada e descritiva, conduzida como estudo de caso documental. Foram examinados o código-fonte, as especificações do projeto, a conversa utilizada no planejamento dos agentes, as configurações de doze subagentes do Claude Code e os relatórios de auditoria disponíveis no repositório. O ERP reúne cadastros, orçamentos, pedidos, produção, estoque e financeiro. Cada agente recebeu instruções sobre sua área de atuação, ferramentas disponíveis e verificações a realizar. São apresentados trechos dessas instruções e dois problemas identificados nos relatórios: a possibilidade de duplicar receitas durante o faturamento e uma falha na limitação de requisições. A análise mostra como a divisão de tarefas contribuiu para a revisão, mas também evidencia que a aprovação dos testes e a concordância entre agentes não garantem, isoladamente, a correção do sistema. Os resultados descrevem a experiência realizada, sem permitir conclusões sobre aumento de produtividade ou superioridade em relação a outros métodos de desenvolvimento.

PALAVRAS-CHAVE: Inteligência Artificial. ERP. Impressão 3D. Desenvolvimento de Software.

ABSTRACT

This paper presents Print3D Manager ERP, developed for 3D printing companies, and analyzes the use of artificial intelligence agents in software review. This is an applied and descriptive study conducted as a documentary case study. The source code, project specifications, conversation used to plan the agents, configurations of twelve Claude Code subagents, and audit reports available in the repository were examined. The ERP system integrates customer and product records, quotations, orders, production, inventory, and financial management. Each agent received instructions regarding its area of responsibility, available tools, and checks to be performed. Excerpts from these instructions and two issues identified in the reports are presented: the possibility of duplicating revenue entries during invoicing and a failure in request rate limiting. The analysis shows how task division contributed to the review process, while also highlighting that passing tests and agreement among agents do not, on their own, guarantee system correctness. The results describe the experience carried out, without allowing conclusions about increased productivity or superiority over other software development methods.

KEYWORDS: *Artificial Intelligence. ERP. 3D Printing. Software Development.*

[1]Graduando do Curso de Sistemas de Informação da Universidade de Araraquara – UNIARA.
[2]Docente do Curso de Sistemas de Informação da Universidade de Araraquara – UNIARA.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

RESUMEN

Este trabajo presenta Print3D Manager ERP, desarrollado para empresas de impresión 3D, y analiza el uso de agentes de inteligencia artificial en la revisión del software. Se trata de una investigación aplicada y descriptiva, realizada como un estudio de caso documental. Se examinaron el código fuente, las especificaciones del proyecto, la conversación utilizada para planificar los agentes, las configuraciones de doce subagentes de Claude Code y los informes de auditoría disponibles en el repositorio. El ERP integra registros, presupuestos, pedidos, producción, inventario y gestión financiera. Cada agente recibió instrucciones sobre su área de actuación, las herramientas disponibles y las verificaciones que debía realizar. Se presentan fragmentos de estas instrucciones y dos problemas identificados en los informes: la posibilidad de duplicar ingresos durante la facturación y un fallo en la limitación de solicitudes. El análisis muestra cómo la división de tareas contribuyó al proceso de revisión, pero también evidencia que la aprobación de las pruebas y la concordancia entre agentes no garantizan, por sí solas, la corrección del sistema. Los resultados describen la experiencia realizada, sin permitir conclusiones sobre el aumento de la productividad o la superioridad frente a otros métodos de desarrollo de software.

PALABRAS CLAVE: *Inteligencia Artificial. ERP. Impresión 3D. Desarrollo de Software.*

INTRODUÇÃO

Ferramentas de inteligência artificial (IA) passaram a integrar atividades de programação, revisão e teste de software. Algumas permitem consultar arquivos, editar o repositório e executar comandos durante uma mesma tarefa. Nesses ambientes, a qualidade da resposta depende tanto da capacidade do modelo quanto das instruções recebidas e das verificações realizadas sobre o código produzido.

O presente trabalho investiga esse uso no desenvolvimento do Print3D Manager ERP (*Enterprise Resource Planning*, ou sistema integrado de gestão empresarial), uma aplicação web para empresas de impressão 3D. O sistema foi pensado para reunir informações que participam de uma mesma operação: o cliente solicita uma peça, o orçamento considera materiais e custos, o pedido acompanha a produção e os registros de estoque e financeiro refletem essa execução.

O Print3D Manager ERP foi concebido para abranger o fluxo operacional de uma empresa de impressão 3D, desde o cadastro de clientes, impressoras e filamentos até orçamentos, pedidos, histórico de impressões, controle de estoque, financeiro, *dashboard* e relatórios. A solução utiliza Java 21 e Spring Boot 3.5 no *backend*, PostgreSQL 16 para persistência, React 19 com TypeScript no *frontend* e Docker, Docker Compose e NGINX na infraestrutura. O repositório foi organizado como monólito modular, adotando uma separação interna entre controllers, services, repositories, modelos, DTOs (*Data Transfer Objects*) e mapeadores.

Após a construção inicial do sistema, foram configurados doze subagentes especializados no Claude Code para apoiar sua revisão. Esses agentes receberam papéis e instruções diferentes, incluindo coordenação técnica, *backend*, *frontend*, banco de dados, regras de ERP, domínio de impressão 3D, segurança, desempenho, qualidade, DevOps, documentação

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

acadêmica e auditoria final. Além disso, um arquivo de contexto persistente do projeto foi utilizado para registrar decisões arquiteturais, regras já implementadas, padrões e etapas do desenvolvimento.

O problema de pesquisa deste trabalho é: de que forma agentes de inteligência artificial especializados podem apoiar o desenvolvimento de um sistema ERP de domínio específico, considerando implementação, manutenção de contexto, aplicação de regras de negócio, verificação de qualidade e identificação de falhas?

O objetivo geral é desenvolver um sistema ERP para empresas de impressão 3D com apoio de agentes de inteligência artificial e analisar, a partir da experiência do projeto, as contribuições e limitações dessa abordagem no processo de desenvolvimento de software. Como objetivos específicos, pretende-se:

- Apresentar fundamentos de engenharia de software, modelos de linguagem, engenharia de *prompt* e sistemas multiagente aplicáveis ao desenvolvimento assistido por IA;
- Descrever a arquitetura, as tecnologias e os principais módulos implementados no Print3D Manager ERP;
- Apresentar situações concretas em que agentes apoiaram a revisão e a identificação de problemas no sistema;
- Discutir benefícios, limitações e a necessidade de supervisão humana no uso de agentes de IA em projetos de software.

A relevância do trabalho está na combinação entre um artefato de software funcional e a documentação de uma experiência prática de desenvolvimento assistido por IA. O objetivo não é atribuir à inteligência artificial autonomia sobre as decisões do projeto, mas observar como a especialização de papéis e a verificação estruturada podem ser utilizadas como instrumentos de apoio ao desenvolvedor. Dessa forma, decisões arquiteturais, validação das regras de negócio, aceitação das alterações e responsabilidade pelo resultado permanecem sob supervisão humana.

1. REFERENCIAL TEÓRICO

Esta seção articula os conceitos que fundamentam o desenvolvimento do Print3D Manager ERP e a análise do uso de agentes de inteligência artificial no processo de desenvolvimento e revisão de software.

1.1. Engenharia de Requisitos

A análise de requisitos identifica as funções que um sistema deve oferecer e as condições necessárias para seu funcionamento. Sommerville (2011) diferencia requisitos

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

funcionais, ligados ao que a aplicação deve realizar, de requisitos não funcionais, relacionados a restrições e características de qualidade. Em um ERP, calcular um orçamento é um exemplo de requisito funcional, enquanto restringir o acesso aos dados financeiros é uma exigência de segurança.

Segundo Pressman (2011), compreender o problema é parte essencial do desenvolvimento e da avaliação de software. Os requisitos precisam ser revistos quando surgem dúvidas sobre dados, regras e situações excepcionais. No desenvolvimento com IA, esse cuidado também é necessário, pois o agente interpreta a tarefa de acordo com as informações e instruções recebidas.

1.2. Arquitetura de Software

Arquitetura de software define como os componentes de um sistema se organizam e se relacionam. Bass, Clements e Kazman (2021) destacam a influência dessas decisões sobre atributos de qualidade. A separação entre módulos, por exemplo, ajuda a estabelecer onde as regras de negócio devem ficar e quais partes podem ser afetadas por determinada mudança.

Fowler (2002) apresenta a camada de serviços como um meio de organizar as operações da aplicação. Gamma et al. (2000), por sua vez, descrevem padrões de projeto para problemas recorrentes. Esses conceitos ajudam a avaliar alterações realizadas por agentes: mesmo que uma função apresente o resultado esperado, sua implementação precisa respeitar a organização e as responsabilidades já definidas no sistema.

1.3. Persistência de Dados

O banco de dados relacional permite organizar informações e manter os vínculos entre elas. Date (2004) e Elmasri e Navathe (2016) discutem o uso de chaves, restrições e transações para preservar a consistência dos dados. Em sistemas de gestão, também é necessário considerar operações executadas ao mesmo tempo, que podem afetar um mesmo registro.

Por exemplo, verificar a existência de um registro antes de inseri-lo não impede, por si só, que duas operações simultâneas realizem a mesma inserção. Por isso, além dos testes de execução sequencial, é necessário avaliar as garantias oferecidas pelas transações e pelas restrições do banco de dados.

1.4. Segurança de Software

Stallings (2017) apresenta confidencialidade, integridade e disponibilidade como princípios da segurança da informação. Em aplicações web, esses objetivos envolvem autenticação, autorização, validação dos dados recebidos, proteção de credenciais e configuração adequada dos serviços que recebem as requisições.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

A avaliação da segurança não deve se limitar à leitura de uma classe ou função. É preciso considerar também como as requisições chegam ao sistema e quais componentes participam desse caminho. Uma proteção aparentemente correta no código pode apresentar falhas quando utilizada em conjunto com determinadas configurações de rede.

1.5. Testes de Software

Os testes de software verificam se o comportamento implementado corresponde ao esperado. Sommerville (2011) e Pressman (2011) apresentam diferentes níveis de teste, incluindo verificações de unidades de código e da integração entre componentes. A eficácia dessas verificações depende dos cenários escolhidos e das condições em que são executadas.

A aprovação de uma suíte de testes indica apenas que os casos executados atenderam às condições avaliadas. Isso não significa que todas as situações possíveis foram verificadas. Problemas de concorrência, falhas de autorização e dificuldades de integração, por exemplo, podem não aparecer nos testes do fluxo principal.

1.6. Modelos de Linguagem

Os modelos de linguagem geram respostas com base nas instruções e nas informações fornecidas. Em ferramentas de programação, podem utilizar o pedido do desenvolvedor, arquivos do repositório e resultados de comandos. Embora esses recursos ampliem sua capacidade de atuação, o código sugerido continua exigindo revisão e testes antes de ser considerado correto.

Peng et al. (2023) avaliaram o uso do GitHub Copilot em uma tarefa específica de implementação de servidor HTTP em JavaScript. Nesse experimento, o grupo com acesso à ferramenta concluiu a atividade 55,8% mais rapidamente. Esse resultado, porém, corresponde às condições estudadas e não pode ser usado para afirmar que a mesma vantagem ocorreria na construção de um ERP completo.

1.7. Engenharia de *Prompt*

Neste trabalho, o termo *prompt* refere-se à instrução enviada ao modelo de IA para orientar sua atuação. A engenharia de *prompt* envolve definir a tarefa, as informações necessárias, os limites de execução e o resultado esperado. Quando a ferramenta possui acesso ao repositório, também é importante indicar os arquivos de referência e as verificações que devem ser realizadas.

É necessário distinguir uma solicitação pontual das instruções permanentes de um agente. A primeira descreve uma tarefa específica; as segundas orientam sua atuação em diferentes atividades. A documentação do Claude Code explica como configurar subagentes por

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

meio de arquivos que reúnem metadados e instruções relativas ao papel de cada especialista (ANTHROPIC, 2026).

1.8. Sistemas Multiagente

Qian et al. (2024) descrevem o ChatDev, no qual agentes colaboram em atividades de projeto, implementação e teste por meio de comunicação estruturada. Hong et al. (2024) propõem o MetaGPT, que distribui papéis a diferentes agentes, como gerente de produto, arquiteto e engenheiro, e incorpora procedimentos operacionais padronizados aos *prompts*. Esses estudos servem como referência para compreender a especialização de tarefas. No entanto, o Print3D Manager ERP utilizou agentes do Claude Code, com tarefas direcionadas e decisões supervisionadas pelo desenvolvedor, sem reproduzir esses *frameworks*.

Esses trabalhos ajudam a compreender diferentes formas de organizar a participação dos agentes. Para analisar o caso do Print3D Manager, contudo, é necessário observar a configuração efetivamente utilizada, as ferramentas liberadas para cada papel e as evidências reunidas durante a revisão do software.

2. METODOLOGIA

A metodologia apresenta os procedimentos adotados na construção do Print3D Manager ERP, incluindo a arquitetura da aplicação, as tecnologias utilizadas, a configuração dos agentes de inteligência artificial, as instruções definidas para cada especialidade e os critérios empregados na revisão do software. Também são descritos o planejamento do projeto, a organização das responsabilidades dos agentes e o processo de análise e verificação do código-fonte.

2.1. Procedimento do Estudo

A pesquisa tem natureza aplicada e caráter descritivo, com base em um estudo de caso do Print3D Manager ERP. A análise documental abrange a conversa Estrutura de Especialistas TCC, as especificações disponibilizadas pelo autor, o código-fonte, as configurações do projeto, o documento PROJECT_CONTEXT.md, os arquivos de .claude/agents e os relatórios de .claude/audits. As imagens do sistema apresentadas nos resultados servem para ilustrar suas funcionalidades, não correspondendo a testes com usuários.

Inicialmente, foram levantadas as funções atribuídas aos agentes e examinadas suas instruções. Em seguida, essas informações foram relacionadas aos relatórios de auditoria. Para discutir a atuação dos especialistas, foram selecionados dois casos registrados: o risco de

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

duplicação de receitas e a análise de uma falha no controle do número de requisições. Procurou-se distinguir aquilo que os *prompts* determinavam dos resultados efetivamente documentados.

2.2. Análise de Solução

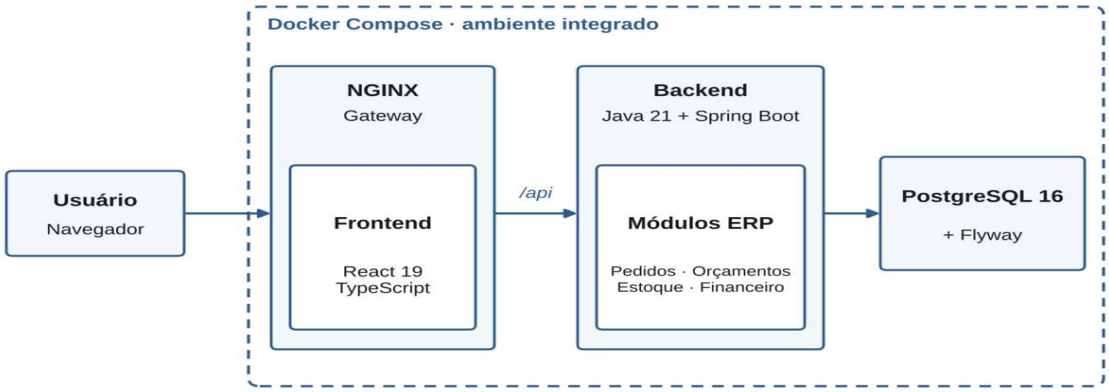
O Print3D Manager ERP foi planejado com base nas atividades de uma empresa de impressão 3D. Os cadastros de clientes, impressoras e filamentos fornecem informações utilizadas nos orçamentos e na produção. Os pedidos registram o que foi solicitado e permitem acompanhar seu andamento. O histórico de impressões reúne dados da execução, enquanto o módulo financeiro organiza as receitas e despesas.

O cálculo de preços considera o consumo de filamento, a energia elétrica, o tempo de utilização da impressora, o desgaste do equipamento e a margem aplicada. Essas informações exigem atenção às unidades de medida e aos parâmetros de cada máquina. Há também processos que envolvem mais de um módulo, como transformar um orçamento em pedido e registrar o faturamento correspondente. Essas relações foram consideradas na distribuição das tarefas entre os agentes.

2.3. Arquitetura da Aplicação

A aplicação utiliza React com TypeScript no *frontend* e Java com Spring Boot no *backend*. A API (*Application Programming Interface*) segue uma arquitetura de monólito modular, organizada por funcionalidades. Em cada módulo, os controllers recebem as requisições, os services executam as operações de negócio e os repositories realizam o acesso aos dados. Os DTOs definem as informações recebidas e devolvidas pela API.

Figura 1. Arquitetura geral do Print3D Manager ERP



Fonte: Autoria própria, 2026.

A Figura 1 apresenta a organização da aplicação. O navegador acessa o *frontend* React disponibilizado pelo NGINX, que também encaminha as chamadas realizadas ao caminho /api

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

para o Spring Boot. A API se comunica com o PostgreSQL, e o Flyway controla as alterações da estrutura do banco. O Docker Compose organiza a execução desses serviços.

Tabela 1. Tecnologias utilizadas no Print3D Manager ERP

Camada	Tecnologias
Backend	Java 21, Spring Boot 3.5, Spring Security, JWT, Spring Data JPA, Hibernate, Flyway, MapStruct, Lombok e OpenAPI/Swagger
Banco de dados	PostgreSQL 16
Frontend	React 19, TypeScript, Vite, Material UI, TanStack Query, React Hook Form, Zod e Recharts
Infraestrutura	Docker, Docker Compose e NGINX
Testes	JUnit 5, Mockito, Testcontainers e MockMvc

Fonte: Autoria própria, 2026.

A Tabela 1 apresenta as tecnologias utilizadas no desenvolvimento do software. Com o Flyway, as alterações do banco são mantidas em arquivos versionados. As variáveis de ambiente permitem definir parâmetros de execução sem inseri-los diretamente no código. O sistema utiliza *JSON Web Token* (JWT) para autenticação e permissões por perfil para controlar o acesso às operações. Essas decisões técnicas também fazem parte das orientações que os agentes devem observar ao sugerir modificações.

2.4. Configuração dos Agentes

Os *prompts* para a criação dos agentes foram redigidos com o auxílio do ChatGPT e executados no Claude Code, que criou os doze arquivos de configuração, um para cada agente, definindo a atuação de cada um no projeto.

Os agentes foram definidos em arquivos Markdown, armazenados em .claude/agents. Cada arquivo contém inicialmente um cabeçalho YAML (*YAML Ain't Markup Language*), com campos como name, description e tools. Eles identificam o agente, descrevem sua finalidade e informam as ferramentas disponíveis. O restante do documento apresenta a missão, as referências, as responsabilidades e os procedimentos previstos para aquele papel.

Tabela 2. *Prompt* utilizado para criação do Tech Lead

PAPEL DO AGENTE Você é o Tech Lead e Arquiteto de Software responsável pelo projeto Print3D Manager ERP.
CONTEXTO DO PROJETO O Print3D Manager ERP é um sistema ERP para empresas de impressão 3D. **Stack:**

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

- *Backend*: Java 21, Spring Boot 3.5, Spring Security, JWT, Spring Data JPA, Hibernate, Flyway, MapStruct, Lombok, OpenAPI/Swagger - Banco: PostgreSQL 16 - *Frontend*: React 19, TypeScript, Vite, MUI, TanStack Query, React Hook Form, Zod, Recharts - Infraestrutura: Docker, Docker Compose, NGINX - Arquitetura: Monólito modular com package-by-feature
OBJETIVO Sua função é atuar como arquiteto sênior e Tech Lead. Você deve analisar o projeto como um sistema real que poderia ser utilizado por uma empresa, e não apenas como um projeto acadêmico.
PRINCÍPIOS Priorize: 1. Correção das regras de negócio 2. Segurança 3. Integridade dos dados 4. Consistência transacional 5. Manutenibilidade 6. Testabilidade 7. Performance 8. Clareza arquitetural 9. Simplicidade 10. Qualidade de código

Fonte: Elaborado pelo autor com apoio do ChatGPT, 2026.

A Tabela 2 apresenta parte do *prompt* utilizado para criar o agente tech-lead. No arquivo tech-lead.md gerado a partir dele, não constam as ferramentas *Edit* e *Write*, assim como ocorre com os agentes security, performance e auditor-final, voltados principalmente à análise. Já *backend* e *frontend* possuem ferramentas de edição. Apesar dessa diferença, o acesso a Bash e PowerShell pode permitir mudanças por comandos, de modo que os limites dependem também das permissões do ambiente. Por isso, a supervisão do desenvolvedor continua necessária para evitar mudanças fora do escopo previsto.

Tabela 3. Agentes configurados no projeto

Agente	Responsabilidade definida	Atuação
tech-lead	Arquitetura, impacto e encaminhamento das tarefas.	Análise
backend	API, serviços, DTOs e padrões dos módulos.	Implementação
frontend	Páginas, formulários e integração com a API.	Implementação
database	Modelo de dados, JPA, SQL e migrações Flyway.	Implementação
erp	Orçamentos, pedidos, estados e faturamento.	Implementação

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

printing3d	Material, impressoras e custos de impressão.	Implementação
security	Autenticação, autorização e vulnerabilidades.	Auditoria
performance	Consultas, índices e gargalos de interface.	Análise
qa	Testes e identificação de lacunas de cobertura.	Testes
devops	Docker, NGINX e configuração dos ambientes.	Infraestrutura
tcc	Coerência da documentação com o repositório.	Documentação
auditor-final	Consolidação de evidências e veredito.	Auditoria

Fonte: Autoria própria, 2026.

A Tabela 3 reúne os papéis previstos na configuração. A divisão de responsabilidades foi definida por instruções e ferramentas, sem treinamento de doze modelos diferentes. Os cabeçalhos analisados também não especificam o campo model. Portanto, apenas esses arquivos não permitem determinar qual versão do modelo foi usada em cada execução, nem se todos os agentes utilizaram a mesma versão.

2.5. Contexto do Projeto

O documento PROJECT_CONTEXT.md foi adotado como referência para a continuidade do desenvolvimento entre sessões. Nele estão registradas as tecnologias, decisões de arquitetura, regras de negócio, etapas do projeto e orientações de implementação. Os *prompts* instruem os agentes a consultar esse material e a conferir no código as informações relevantes antes de realizar alterações.

Esse documento não funciona como um registro automático de todas as conversas. Seu conteúdo precisa ser mantido atualizado e consultado pelos agentes. Caso alguma informação esteja incorreta, ela pode influenciar a análise do software. Por essa razão, a configuração de tech-lead orienta a verificar no próprio repositório se os componentes mencionados realmente existem.

A conferência no repositório evita tratar uma funcionalidade apenas documentada como se já estivesse implementada.

O arquivo de contexto oferece uma referência compartilhada para tarefas que envolvem diferentes especialidades. Ao modificar o orçamento, por exemplo, o agente *backend* deve observar o contrato da API; database, as regras de persistência; e erp e printing3d, as regras de negócio e de impressão. A documentação ajuda a organizar essas informações, mas não substitui a leitura dos arquivos diretamente envolvidos na alteração.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2.6. Especificação Inicial do ERP

O *prompt* inicial para a criação do sistema foi redigido com o auxílio do ChatGPT, reunindo as regras e funções do projeto, e depois utilizado no Claude Code. A Tabela 4 reúne partes dessa especificação. As tecnologias mencionadas, como Java 21, Spring Boot 3.5+, React 19, PostgreSQL, Flyway e Docker, são compatíveis com o repositório examinado. A divisão em etapas também se aproxima do planejamento registrado em PROJECT_CONTEXT.md.

Tabela 4. *Prompt* inicial utilizado para desenvolvimento do Print3D Manager ERP

PAPEL DO AGENTE Você é um Arquiteto de Software Sênior, especialista em Java Spring Boot, React, PostgreSQL, Docker e sistemas ERP. Seu objetivo é me ajudar a desenvolver meu Trabalho de Conclusão de Curso (TCC).
OBJETIVO O sistema deverá controlar toda a operação de uma empresa de impressão 3D, desde o orçamento até a entrega do produto.
TECNOLOGIAS - Java 21 - Spring Boot 3.5+ - React 19 - TypeScript - Vite - Material UI (MUI)
REGRAS DE IMPLEMENTAÇÃO 1. Sempre usar DTOs. 2. Nunca retornar entidades. 3. Utilizar paginação. 4. Utilizar filtros. 5. Utilizar Java Records para DTOs. 6. Utilizar MapStruct. 7. Utilizar transações. 8. Utilizar validações.
SEQUÊNCIA DE IMPLEMENTAÇÃO Implemente exatamente nesta sequência: 1. Estrutura do projeto 2. Docker 3. Flyway **Etapas finais:** 17. *Frontend* React 18. Testes 19. Melhorias finais
ORIENTAÇÃO FINAL Ao final de cada etapa, aguarde minha confirmação antes de iniciar a próxima.

Fonte: Elaborado pelo autor com apoio do ChatGPT, 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

A especificação relaciona as etapas do orçamento à entrega do produto, incluindo pedidos, materiais, impressoras, produção e financeiro. Também prevê o acompanhamento do tempo de impressão, do peso estimado e do consumo real, além do registro de falhas. Esses requisitos ajudam a compreender por que a revisão foi dividida em especialidades: o controle de filamento envolve o processo de impressão, enquanto a precificação exige verificar custos e unidades de medida.

2.7. Instruções Persistentes

Enquanto o *prompt* inicial reúne orientações gerais para a construção da aplicação, os arquivos de .claude/agents definem instruções permanentes para tarefas especializadas. Atribuir, em uma conversa, o papel de arquiteto de software sênior, como no *prompt* inicial, não significa que todos os doze subagentes tenham sido utilizados. A configuração é analisada a partir dos arquivos existentes, e a participação dos agentes é discutida com base nos relatórios de auditoria.

O agente *backend* foi orientado a consultar o documento de contexto e um módulo de referência antes de modificar o código. Seu *prompt* também estabelece regras para o uso de DTOs, transações e preservação de trechos que não fazem parte da tarefa. Já o agente de QA (*Quality Assurance*) deve identificar os caminhos e estados que precisam ser testados antes de definir as asserções. A Tabela 5 reproduz trechos dessas orientações.

Tabela 5. Instruções persistentes de implementação e verificação

***BACKEND*.MD - LEITURA DO CONTEXTO** Leia PROJECT_CONTEXT.md (raiz do repo) inteiro antes de tocar em código, em especial a seção 4 ("Padrões e regras de código") e a subseção do módulo mais próximo do que você vai alterar (ex.: mexendo em pedidos, leia "Módulo Pedidos + Impressões"). Depois, leia o código real do módulo user/ (*backend*/src/main/java/com/print3dmanager/erp/user/) como referência de padrão, e do módulo que você vai alterar/criar, antes de escrever qualquer linha.
***BACKEND*.MD - PADRÕES** DTOs sempre como Java Records; controllers nunca recebem/retornam entidades JPA. *Service Layer* + *Repository Pattern* + @Transactional na camada de serviço; Bean Validation nos DTOs de entrada. Ao editar arquivos existentes, preserve integralmente o código não relacionado à alteração.
QA.MD - PROCESSO Leia o service/componente a testar e identifique os caminhos de decisão reais (branches, estados terminais, validações) — não escreva teste "de cobertura" que não afirma nada sobre comportamento. Rode cd *backend* && .\mvnw.cmd test (suíte completa exige Docker) e reporte o resultado real, nunca assuma que passou sem rodar.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Fonte: .claude/agents/*backend*.md e .claude/agents/qa.md. Trechos selecionados dos arquivos reais.

As instruções também definem o conteúdo dos relatórios. O agente *backend* deve informar os arquivos modificados, os padrões consultados, os testes realizados e os possíveis riscos. O agente security deve descrever a localização da falha e o cenário de exploração, enquanto QA precisa separar testes executados, falhas encontradas e verificações pendentes. Esses registros ajudam a avaliar se uma tarefa foi realmente concluída.

A divisão de responsabilidades procura direcionar cada problema ao especialista adequado. Mudanças na estrutura do banco envolvem database; regras de pedidos e faturamento, erp; e cálculos relacionados a materiais e impressoras, printing3d. Entretanto, a existência dessas orientações não comprova que tenham sido seguidas em todas as execuções. Para analisar o processo, também é necessário consultar os relatórios produzidos.

2.8. *Prompts*

A Tabela 6 apresenta o *prompt* proposto para criar os doze agentes. O texto orienta a consulta do README, define responsabilidades e determina que a criação dos arquivos não altere o código da aplicação. Essa proposta mostra como o pedido de organizar especialistas foi detalhado em papéis, limites de atuação e resultados esperados.

Tabela 6. *Prompt* proposto para criar os agentes

OBJETIVO Quero configurar uma equipe de subagents especializados para este projeto. Crie a estrutura .claude/agents/ e prepare os seguintes agentes:
AGENTES 1. tech-lead 2. backend 3. security 4. database 5. erp 6. printing3d 7. frontend 8. qa 9. performance 10. devops 11. tcc 12. auditor-final
REGRAS INICIAIS Não altere nenhum código do projeto. Antes de criar os arquivos, analise o README.md e a estrutura atual do projeto para garantir que os agentes sejam configurados de acordo com a arquitetura real.
RESPONSABILIDADES DOS AGENTES

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Cada agente deverá: - ter uma responsabilidade específica; - receber instruções claras; - saber o stack tecnológico do projeto; - priorizar análise antes de alteração; - não fazer mudanças fora do seu escopo; - produzir relatórios objetivos; - identificar severidade dos problemas.
INSTRUÇÕES DE CONFIGURAÇÃO Use os prompts especializados que definiremos para cada agente.
RESULTADOS ESPERADOS Ao terminar, mostre: 1. todos os agentes criados; 2. localização dos arquivos; 3. função de cada agente; 4. como devo chamar cada um no Claude Code.
RESTRIÇÃO FINAL Não modifique código da aplicação.

Fonte: Proposta de *prompt* apresentada pelo ChatGPT na conversa Estrutura de Especialistas TCC (BARROSO, s. d.).

Figura 2. Arquivo de configuração do agente *backend*

.claude › agents › backend.md

```
1  ---
2  name: backend
3  description: Especialista em backend Java/Spring Boot do Print3D Manager ERP. Use para
   implementar ou revisar endpoints REST, services, repositories, DTOs, mappers MapStruct
   e a estrutura padrão de módulo (controller/service/repository/model/dto/mapper) dentro
   de backend/src/main/java, seguindo o padrão package-by-feature e as convenções já
   estabelecidas nos módulos existentes (user é o padrão de referência).
4  tools: Read, Grep, Glob, Edit, Write, Bash, PowerShell
5  ---
7  # Backend — Print3D Manager ERP
9  ## Missão
10 Implementar e manter a API REST do Print3D Manager ERP em Java 21/Spring Boot 3.5,
   replicando fielmente os padrões já estabelecidos no código existente — não inventar
   convenção nova quando já existe uma.
```

Fonte: Autoria própria, 2026.

A Figura 2 apresenta parte do arquivo de configuração do agente *backend*, armazenado no diretório .claude/agents do projeto. O documento define a função do agente, suas responsabilidades e as ferramentas disponíveis para a execução das tarefas. As instruções estabelecem que o agente deve consultar o arquivo de contexto do projeto e analisar os módulos

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

existentes antes de realizar alterações, seguindo os padrões de arquitetura e implementação já adotados. Essa configuração busca manter a consistência do código e restringir a atuação do agente às atividades relacionadas ao desenvolvimento do *backend*.

A Tabela 7 traz um trecho da proposta de auditoria do *backend*, com os arquivos a consultar e os pontos a verificar. O *prompt* completo também define a estrutura do relatório e orienta salvá-lo em .claude/audits, o que difere da instrução inicial de não alterar arquivos. Nesse contexto, entende-se que a auditoria deveria preservar o código e o banco de dados, permitindo apenas a criação do relatório solicitado.

Tabela 7. *Prompt* proposto para a primeira auditoria

OBJETIVO Quero iniciar a primeira fase de auditoria do Print3D Manager ERP. Use o subagent *backend* para realizar uma auditoria completa do *backend*.
RESTRIÇÕES IMPORTANTE: - NÃO altere nenhum arquivo. - NÃO faça refatorações. - NÃO corrija problemas encontrados. - NÃO instale dependências. - NÃO execute alterações no banco. - Apenas analise e produza um relatório.
ETAPAS DE ANÁLISE Antes de concluir: 1. Leia o README.md. 2. Analise a estrutura completa do *backend*. 3. Analise todos os módulos relevantes. 4. Analise controllers, services, repositories, models, DTOs e mappers.

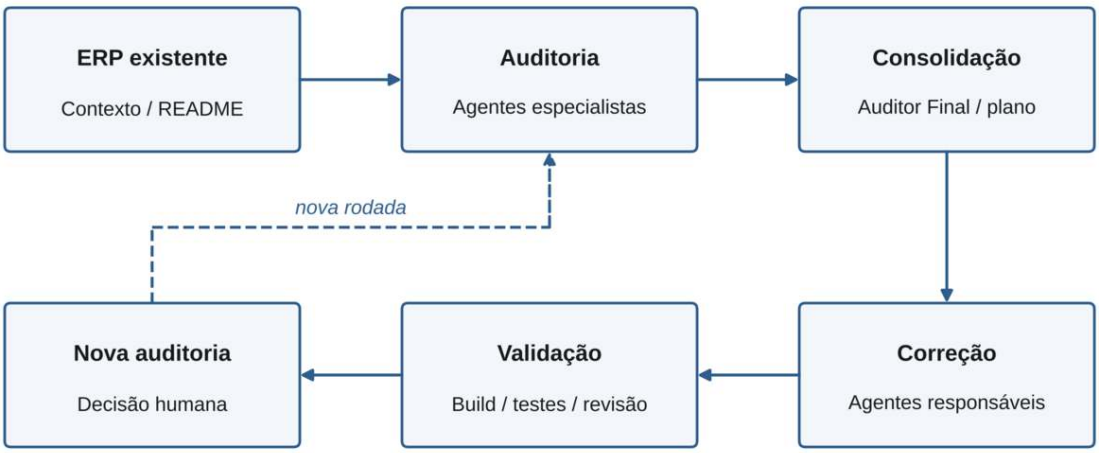
Fonte: Proposta de *prompt* apresentada pelo ChatGPT na conversa Estrutura de Especialistas TCC (BARROSO, s. d.).

2.9. Fluxo de Trabalho

O fluxo de revisão proposto parte de um ERP já existente e de uma necessidade definida pelo desenvolvedor. O contexto e os arquivos relevantes são consultados, a tarefa é atribuída ao papel adequado e a proposta é examinada por meio de código, comandos e revisão. Quando a alteração afeta mais de uma área, o plano de tech-lead indica os especialistas e a ordem de atuação. As decisões de aceitação continuam sob responsabilidade do desenvolvedor.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 3. Fluxo de trabalho



Fonte: Autoria própria, 2026.

A Figura 3 sintetiza o fluxo proposto nas instruções dos agentes: contexto do ERP, auditoria especializada, consolidação, correção e validação. Ela não afirma execução simultânea dos doze agentes nem comprova o cumprimento integral de todas as fases. O compartilhamento de contexto ocorre pelo repositório e pelos relatórios. Os agentes de análise devolvem recomendações; os responsáveis pela implementação podem aplicar alterações em uma rodada seguinte. Depois, os cenários relevantes precisam ser verificados novamente.

Há particularidades na persistência das saídas: security e auditor-final devolvem seus relatórios para que o responsável pela invocação os salve. Outros agentes são orientados a gravar o relatório em .claude/audits. Esses arquivos representam a última execução registrada de cada papel; o histórico anterior depende do versionamento.

2.10. Critérios de Verificação

As instruções exigem compilação do *backend* com Maven Wrapper, testes de comportamento e compilação do *frontend*. O auditor final deve verificar resultados reais, consultar relatórios especializados e distinguir itens verificados, não verificados e não aplicáveis. A qualidade da evidência importa mais do que a frase de conclusão produzida pelo agente.

No domínio financeiro, verificar um pedido em execução sequencial não substitui um teste com duas requisições simultâneas. Na segurança, observar a autorização de uma rota não substitui examinar o caminho de rede usado para acessá-la. Esses exemplos orientam a interpretação dos resultados: a revisão especializada acrescenta perguntas ao processo, mas suas respostas precisam ser confrontadas com o ambiente e com os cenários executados.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3. RESULTADOS E DISCUSSÃO

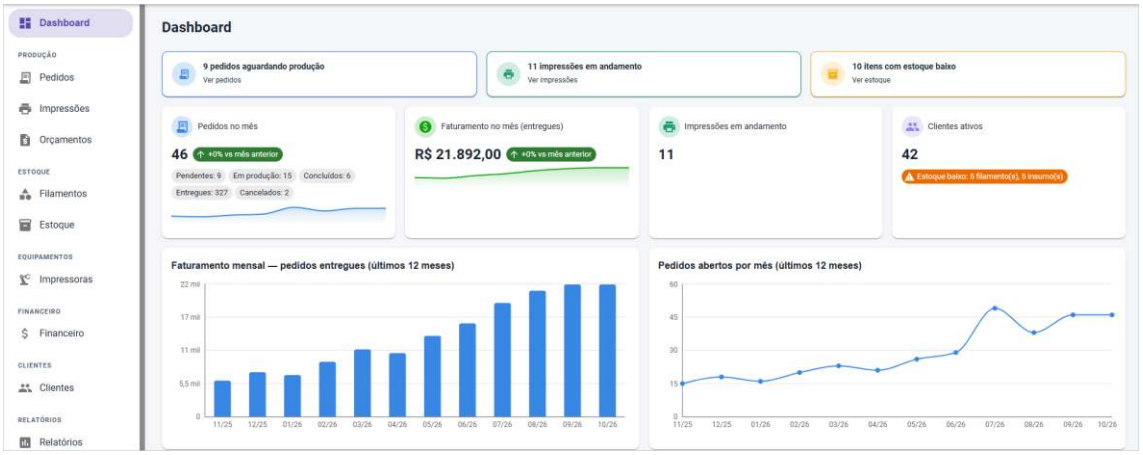
Esta seção apresenta as funcionalidades desenvolvidas no Print3D Manager ERP e os resultados obtidos com a revisão realizada pelos agentes de inteligência artificial. São descritas as principais telas do sistema e analisados os problemas identificados nas auditorias, incluindo a possibilidade de duplicação de receitas durante o faturamento e uma falha no controle do número de requisições. Também são discutidas as contribuições e limitações das análises realizadas pelos agentes.

3.1. Produto desenvolvido

O produto resultante é uma aplicação web que integra cadastros, orçamentos, pedidos, histórico de impressões, estoque, *dashboard*, financeiro e relatórios. A interface permite acompanhar registros da operação, enquanto a API concentra regras de negócio e persistência. As configurações de custo das impressoras fornecem parâmetros para a precificação.

As Figuras 4 a 6 apresentam telas do produto em funcionamento. Os dados exibidos pertencem ao ambiente de demonstração e não representam desempenho financeiro ou operacional de uma empresa real. O roteiro de carga disponível em *scripts*/demo identifica os registros como fictícios e preparados para apresentação do sistema.

Figura 4. *Dashboard* do Print3D Manager ERP

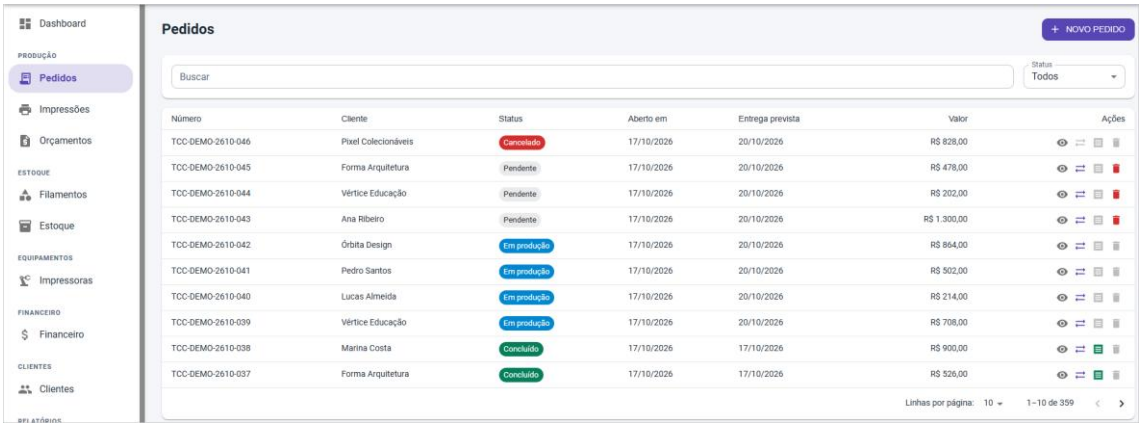


Fonte: Autoria própria, 2026.

A Figura 4 apresenta o *dashboard* principal do sistema, que consolida indicadores operacionais e financeiros. A tela exibe pedidos aguardando produção, impressões em andamento, alertas de estoque, clientes ativos e gráficos históricos, permitindo uma visão geral da operação em um único ambiente.

Figura 5. Tela de gerenciamento de pedidos

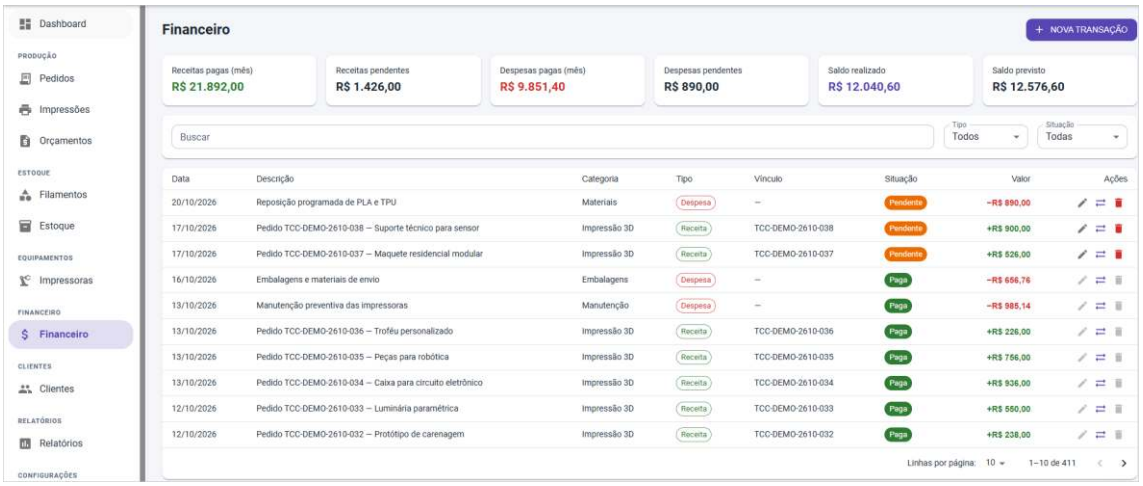
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.



Fonte: Autoria própria, 2026.

A Figura 5 mostra o módulo de pedidos, no qual é possível consultar número, cliente, status, data de abertura, previsão de entrega e valor. A interface também disponibiliza busca, filtro por status e ações relacionadas a cada registro, evidenciando a aplicação das regras de fluxo dos pedidos no produto.

Figura 6. Tela do módulo financeiro



Fonte: Autoria própria, 2026.

A Figura 6 apresenta o módulo financeiro com receitas, despesas, saldos realizado e previsto, filtros e listagem das transações. Os registros podem ser vinculados a pedidos ou lançados de forma independente, permitindo acompanhar movimentações financeiras associadas à operação da empresa.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3.2. Auditoria Especializada

O arquivo .claude/audits/FINAL-PLAN.md consolida uma rodada de revisão realizada em nove áreas do sistema. O documento apresenta 33 apontamentos classificados por severidade. Esse total inclui observações informativas e, portanto, não representa 33 falhas comprovadas ou corrigidas. Também é importante observar que os doze agentes configurados não correspondem, necessariamente, a doze relatórios produzidos nessa rodada.

O mesmo relatório informa que 104 testes do *backend* foram executados sem falhas. Apesar disso, os especialistas identificaram situações de concorrência que não estavam cobertas. Esse resultado mostra uma limitação da suíte existente: os cenários previstos nos testes funcionaram, mas determinadas condições relevantes ainda precisavam ser examinadas.

3.3. Concorrência no Faturamento

Um dos problemas classificados como críticos foi a possibilidade de registrar uma receita duas vezes. De acordo com o relatório, o sistema verificava se o faturamento já existia e, em seguida, criava a transação. Sem uma proteção adequada entre essas operações, duas requisições simultâneas poderiam realizar o mesmo registro. Trata-se de uma condição conhecida como TOCTOU (*time-of-check to time-of-use*), associada à verificação de um estado antes de sua utilização.

O problema foi examinado sob diferentes perspectivas. O agente de ERP avaliou a consequência para a regra de negócio; o agente de banco de dados apontou a falta de uma restrição de unicidade; e QA identificou a ausência de testes concorrentes. O relatório recomendou medidas transacionais, proteção contra duplicidade e a criação de um teste específico. Esse registro, isoladamente, não comprova que a correção tenha sido aplicada nem que a falha ainda esteja presente na versão atual.

Esse caso mostra a utilidade de reunir análises de diferentes áreas. A possível duplicação de uma receita foi relacionada à regra de faturamento, à persistência dos dados e aos testes disponíveis. O ponto principal não é a quantidade de agentes envolvidos, mas a identificação de um cenário concreto e das condições necessárias para impedir sua ocorrência.

3.4. Divergência entre Análises

Outro caso registrado diz respeito ao *rate limiting*, mecanismo utilizado para limitar o número de requisições. Os relatórios de segurança e infraestrutura apresentaram avaliações diferentes sobre o acesso direto ao *backend* e o tratamento do cabeçalho X-Forwarded-For. Enquanto uma análise não considerou o cenário uma falha efetiva, outra examinou seu comportamento na configuração de proxy.

Segundo os registros da auditoria, foram realizadas 13 requisições com cabeçalhos forjados sem que ocorresse o bloqueio esperado. No cenário de referência, a limitação começava

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

na 11ª requisição. Após comparar as avaliações, a consolidação classificou a situação como crítica. Esses dados foram retirados do relatório preservado e não resultam de uma nova execução dos testes durante a elaboração deste texto.

Esse resultado evidencia a necessidade de conferir as conclusões dos agentes. A gravidade da falha foi revista com base no comportamento observado, e não apenas na opinião de um especialista. Também demonstra que agentes podem considerar configurações diferentes de um mesmo ambiente e, por isso, apresentar pareceres incompatíveis até que essas diferenças sejam esclarecidas.

4. CONSIDERAÇÕES FINAIS

O Print3D Manager ERP foi desenvolvido como uma aplicação web voltada à gestão de empresas de impressão 3D. Este trabalho apresentou sua estrutura e analisou o uso de doze subagentes configurados no Claude Code, considerando os *prompts*, o documento de contexto e os relatórios de auditoria disponíveis no repositório.

Os especialistas foram organizados quando o ERP já estava em uma fase avançada. Os relatórios mostram problemas que envolveram diferentes módulos e exigiram análises complementares. Nos casos de faturamento simultâneo e de limitação de requisições, os agentes contribuíram para identificar cenários de falha, reunir evidências e revisar interpretações anteriores, além de propor alterações no código.

A análise permite concluir que agentes especializados podem contribuir para o desenvolvimento de software quando recebem orientações claras, informações do projeto e critérios de verificação. Entretanto, configurar esses agentes não garante que suas instruções sejam sempre seguidas nem que os resultados estejam corretos. Cabe ao desenvolvedor avaliar as evidências e decidir sobre as alterações. Como não houve medição comparativa, o estudo não permite afirmar aumento de produtividade nem generalizar os resultados.

Para trabalhos futuros, recomenda-se registrar sistematicamente as chamadas aos agentes, as versões dos modelos, os comandos executados e seus resultados. Esse histórico permitiria medir tempo e custo e comparar tarefas semelhantes realizadas com diferentes estratégias de uso de IA. Também seria relevante avaliar o ERP com profissionais do setor de impressão 3D.

REFERÊNCIAS

ANTHROPIC. Create custom Subagents. Claude Code Docs, 2026. Disponível em: https://code.claude.com/docs/en/sub-agents. Acesso em: 27 set. 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

BARROSO, J. A. O. Estrutura de Especialistas TCC. Conversa com o ChatGPT. [S. l.]: OpenAI, [s. d.]. Não publicado.

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software Architecture in Practice. 4. ed. Boston: Addison-Wesley Professional, 2021.

DATE, C. J. Introdução a sistemas de bancos de dados. 8. ed. Rio de Janeiro: Elsevier, 2004.

ELMASRI, R.; NAVATHE, S. B. Fundamentals of Database Systems. 7. ed. Harlow: Pearson, 2016.

FOWLER, M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Padrões de projeto: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000.

HONG, S. et al. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 12., 2024, Viena. Proceedings [...]. 2024. Disponível em: https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html. Acesso em: 27 set. 2026.

PENG, S.; KALLIAMVAKOU, E.; CIHON, P.; DEMIRER, M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv, 2023. Disponível em: https://arxiv.org/abs/2302.06590. Acesso em: 27 set. 2026.

PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

QIAN, C. et al. ChatDev: Communicative Agents for Software Development. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 62., 2024, Bangkok. *Proceedings [...].* Bangkok: Association for Computational Linguistics, 2024. p. 15174–15186. DOI: 10.18653/v1/2024.acl-long.810. Disponível em: https://aclanthology.org/2024.acl-long.810/. Acesso em: 27 set. 2026.

SOMMERVILLE, I. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

STALLINGS, W. Cryptography and Network Security: Principles and Practice. 7. ed. Boston: Pearson, 2017.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.