XVIII CONINC
CONGRESSO DE INICIAÇÃO CIENTÍFICA EDUVALE
ISSN 26755734
20 a 24 de outubro de 2025 – Avaré/SP

APLICAÇÃO WEB PARA GERENCIAMENTO DE PRESTAÇÕES DE SERVIÇO PARA EMPRESAS DE CONTABILIDADE

Bruno David Martins – UNEDUVALE – dbrunobruno369@gmail.com
Marcos Alfredo Mendes do Rego – UNEDUVALE – marcos.rego@ead.eduvaleavare.com.br

ÁREA: Exatas e Tecnologia

RESUMO

Este trabalho descreve o desenvolvimento de uma aplicação web para gestão integrada de prestações de serviço em empresas de contabilidade, gestão financeira eficaz, automatizando processos de cadastro e controle de funcionários, gerentes, clientes PF/PJ, serviços, solicitações, propostas comerciais e ordens de serviço. A arquitetura em camadas foi implementada com Python e Flask no *backend*, React e TypeScript no frontend, SQLite para persistência de dados e Tailwind CSS para estilização. O *Middleware* realiza autenticação, validação de dados e registro de execuções, assegurando segurança e consistência. As regras de negócio incluem exclusão reversível em todas as exclusões, limites de desconto de até 20% para funcionários, vínculos obrigatórios entre entidades (funcionário-cargo-departamento, cliente-endereço, empresa-responsável legal) e controle diferenciado de status conforme o tipo de entidade. As interfaces foram prototipadas no Figma e documentadas no Notion, garantindo alinhamento entre design e implementação. A geração de propostas comerciais utiliza templates Jinja2 renderizados em PDF pelo *backend*, padronizando a apresentação dos documentos e registrando versões com status (rascunho, enviado, aprovado). Como extensões futuras, propõe-se incorporar Inteligência Artificial para implementar *chatbot* de suporte, detecção de discrepâncias financeiras em lançamentos de valores e análise preditiva de padrões em ordens de serviço por cliente. Espera-se que a aplicação proporcione maior eficiência operacional, visibilidade dos processos contábeis, melhoria da comunicação interna e externa, redução de erros e suporte à tomada de decisões estratégicas. Esta ferramenta moderna e flexível visa atender às demandas específicas de escritórios de contabilidade, promovendo organização, automação e governança eficaz de todos os fluxos de serviço.

PALAVRAS-CHAVE: sistema web; contabilidade; Flask; React; comercial;

INTRODUÇÃO

Nos últimos anos, a tecnologia tem revolucionado a forma como os escritórios de contabilidade operam, trazendo mais eficiência à gestão de processos e à análise de informações financeiras. O uso de softwares especializados deixou de ser um diferencial para

se tornar essencial, pois automatiza tarefas repetitivas, reduz falhas humanas e permite que os profissionais concentrem-se em atividades mais estratégicas. Nesse contexto, o desenvolvimento de sistemas web voltados para a contabilidade surge como solução inovadora e necessária para otimizar o cotidiano desses escritórios.

O mercado de software de contabilidade, avaliado em 16,2 bilhões de dólares em 2020, deverá alcançar 26,4 bilhões de dólares em 2026. As soluções mais recentes permitiram maior compreensão da saúde financeira das organizações, automatizando a coleta de dados e simplificando as tarefas de gestão, de modo que os contadores possam dedicar-se ao que sabem fazer de melhor: analisar informações e identificar oportunidades de crescimento (Conselho Federal de Contabilidade, 2022).

Estudos realizados na região metropolitana de Belo Horizonte/MG analisaram o nível de gerenciamento dos escritórios de contabilidade por meio da Taxonomia de Kaplan e Cooper. Essa taxonomia permite classificar as empresas em diferentes estágios de maturidade gerencial, identificando práticas contábeis e de controle interno mais eficazes em cada nível de desenvolvimento. O estudo demonstrou que empresas que adotam sistemas gerenciais bem estruturados conseguem otimizar a alocação de recursos, reduzir falhas operacionais e aprimorar a tomada de decisões estratégicas, servindo de base para projetar soluções adaptáveis à dinâmica do setor contábil.

Diante desse cenário, a aplicação proposta neste trabalho tem como objetivo solucionar desafios relacionados à organização interna, ao controle financeiro e à geração de propostas comerciais em escritórios de contabilidade. O sistema buscará otimizar a emissão e o gerenciamento de ordens de serviço (O.S.) por meio de interface intuitiva, bem como automatizar a criação de propostas a partir de templates dinâmicos processados no Flask (Jinja2) e exportados em PDF, garantindo consistência visual e rastreabilidade de versões. Ademais, visa minimizar falhas operacionais decorrentes da falta de familiaridade dos funcionários com a ferramenta e aprimorar a comunicação entre empresa e clientes, garantindo maior eficiência e transparência nos processos contábeis.

MATERIAL E MÉTODOS

O desenvolvimento do sistema *web* proposto seguiu uma abordagem estruturada, fundamentada em tecnologias atuais e práticas consolidadas de engenharia de software.

Esta seção descreve as ferramentas, tecnologias e arquitetura utilizadas no projeto.

Durante o processo de desenvolvimento, foram empregues diversas tecnologias selecionadas conforme suas capacidades de atender às necessidades específicas do projeto, no *backend* utilizou-se a linguagem Python 3.12.6, escolhida pela sintaxe clara e objetiva que favorece o desenvolvimento ágil, além de sua ampla adoção em aplicações web e análise de dados. Para a estruturação, empregando-se o *microframework* Flask, selecionado por sua leveza, flexibilidade e modularidade, responsável pelo gerenciamento de rotas, processamento de requisições HTTP e integração com o banco de dados.

Para o armazenamento e gerenciamento dos dados, adotou-se o SQLite, sistema de gerenciamento de banco de dados relacional leve, autônomo e portável. Esta escolha fundamentou-se na simplicidade de configuração e adequação às fases de prototipação e testes. Já no ambiente de desenvolvimento foi configurado no Visual Studio Code, com utilização de ambientes virtuais (venv) para garantir o isolamento das dependências do projeto.

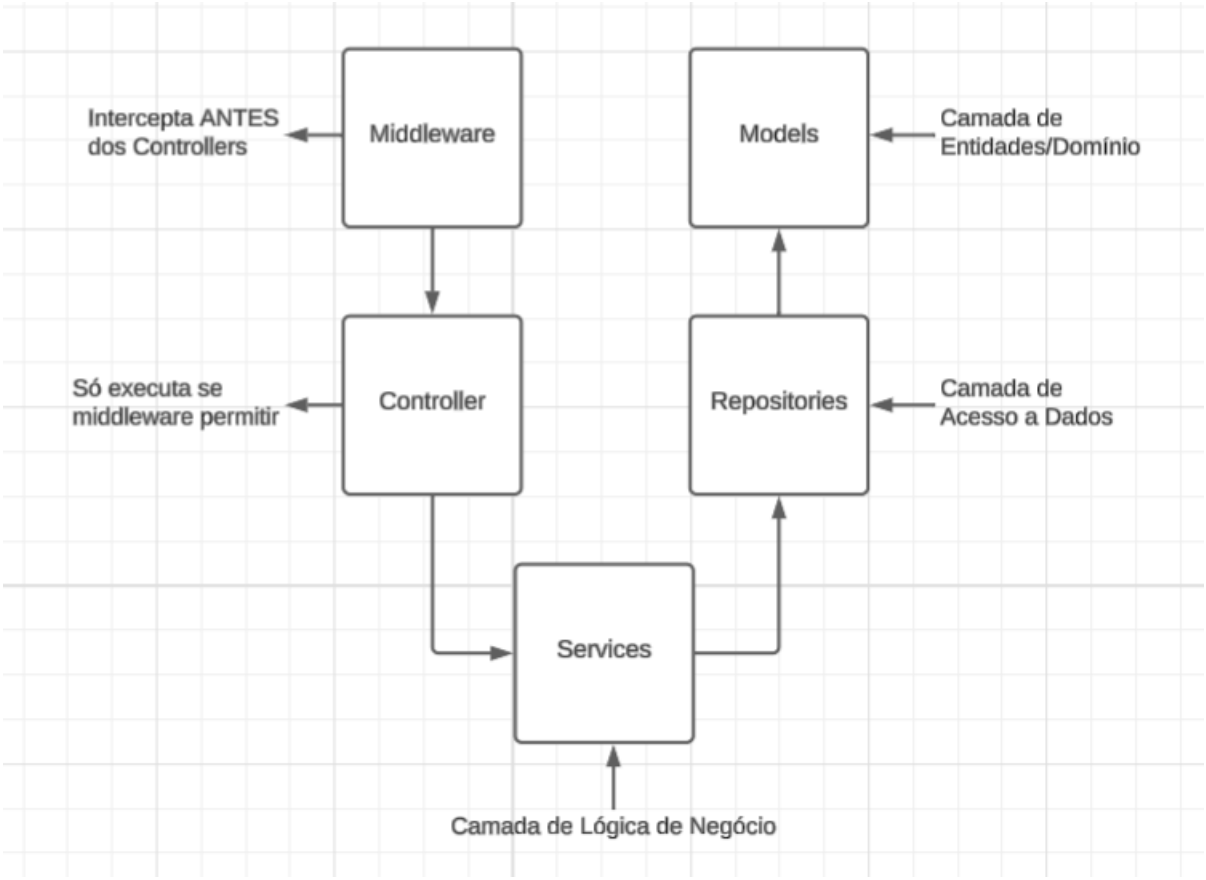
O *frontend* foi desenvolvido empregando-se o *framework* React em conjunto com TypeScript (TS/TSX), proporcionando tipagem estática que aumenta a robustez do código e facilita a manutenção. Para a estilização das interfaces, utilizou-se o Tailwind CSS, *framework* utilitário que permitiu a criação de layouts responsivos e consistentes de forma eficiente.

Para a concepção e validação das interfaces, empregou-se a plataforma Figma, ferramenta que possibilitou a criação de protótipos interativos e layouts de alta-fidelidade. Esta etapa permitiu validações de experiência do usuário (UX) e interface (UI) antes da implementação, além de facilitar a exportação de estilos CSS. A documentação do projeto e o gerenciamento de requisitos foram centralizados na plataforma Notion, que permitiu o registo estruturado de requisitos funcionais e não funcionais, cronograma de atividades e histórico de decisões técnicas.

Implementou-se uma arquitetura em camadas no *backend*, fundamentada em padrões de engenharia de software que promovem a separação de responsabilidades, facilitando a manutenção, testabilidade e escalabilidade do código. A arquitetura adotada seguiu o fluxo de

processamento apresentado abaixo (Figura 1):

Figura 1: Arquitetura de API



Fonte: Elaborado pelo autor.

A arquitetura implementada no sistema segue o padrão em camadas, que é fundamental para garantir a separação de responsabilidades e facilitar a manutenção do código. O *Middleware* foi posicionado como uma camada transversal que intercepta todas as requisições antes que cheguem aos demais componentes, sendo responsável por processar a autenticação de usuários (diferenciando gerentes de funcionários), validar os dados de entrada conforme as regras específicas do negócio (como validação de CPF, CNPJ e limites de desconto) e fazer o registro de todas as operações para fins de auditoria para manter a estabilidade do sistema.

As camadas *Controllers* e *Services* trabalham em conjunto para processar a lógica de negócio do sistema financeiro. Os *Controllers* atuam como ponte entre o *frontend* React e o *backend* Flask, expondo as rotas da API REST organizadas por módulos (funcionários/gerentes, clientes, serviços, solicitações, ordens de serviço e propostas comerciais), recebendo as requisições HTTP e direcionando-as para os serviços apropriados. Já os Services concentram toda a lógica de negócio crítica, implementando regras como o controle automático de descontos até 20% para funcionários e processo de aprovação para valores superiores, gerenciamento dos vínculos obrigatórios entre entidades, aplicação consistente de exclusão reversível em todas as rotas de delete e controle diferenciado de status conforme o tipo de modificação.

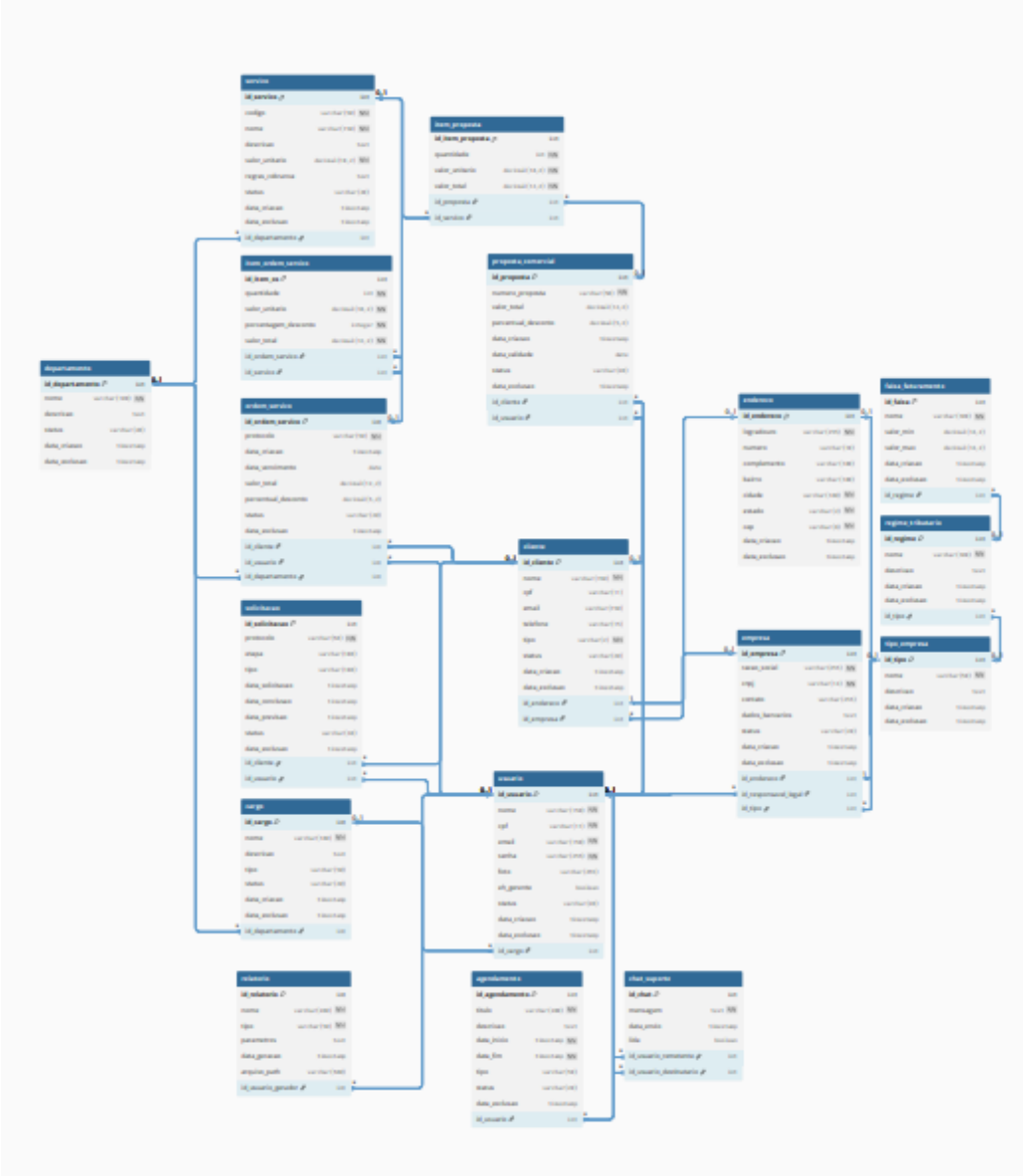
Na base da arquitetura, as camadas *Repositories* e *Models* são responsáveis pela persistência e modelagem dos dados. Os *Repositories* encapsulam completamente o acesso ao banco SQLite, fornecendo métodos de Create, Read, Update e Delete (CRUD) especializados para cada entidade do domínio contábil e implementando consultas específicas para relatórios financeiros, controle de inadimplência e análise de receitas e despesas. Os Models definem as estruturas de dados do sistema através de arquivos Python dedicados, garantindo a modelagem correta dos relacionamentos obrigatórios entre as entidades e mantendo a consistência dos dados financeiros por meio de validações de domínio e controles de status específicos para cada tipo de registro.

A metodologia utilizada no desenvolvimento do sistema foi iterativa e incremental, apoiada em práticas ágeis de engenharia de software para viabilizar entregas frequentes de valor e permitir a incorporação contínua de feedback por parte dos usuários. Inicialmente, procedeu-se à análise e levantamento de requisitos, em que as necessidades funcionais e não funcionais foram capturadas e documentadas no Notion, incluindo a definição de casos de uso e cenários de teste para nortear as etapas seguintes.

Em seguida, realizou-se a prototipação de interface no Figma, criando-se *wireframes* e protótipos de alta fidelidade que foram submetidos a validações de experiência do usuário (UX) e design de interface (UI). Essa fase garantiu que a usabilidade e a estética estivessem

alinhadas às expectativas antes da implementação propriamente dita. Paralelamente, desenvolveu-se a modelagem do banco de dados SQLite por meio de diagramas UML, estabelecendo as entidades, atributos, relacionamentos e restrições de integridade necessários para suportar a lógica de negócio (Figura 2).

Figura 2: Diagrama UML



Fonte: Elaborado pelo autor.

ISSN 26755734

RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados esperados e as discussões relativas às funcionalidades e às interfaces do sistema financeiro integrado para empresas de contabilidade, de modo a evidenciar como as etapas de desenvolvimento descritas em material e métodos se traduziram em entregas concretas e alinhadas aos objetivos do projeto.

O sistema deverá disponibilizar, por meio da API desenvolvida em Flask e em React + TypeScript, todos os módulos previstos: gestão de pessoas (funcionários e gerentes), clientes PF/PJ, serviços, solicitações, propostas comerciais e ordens de serviço. Cada módulo oferecerá operações completas de criação, leitura, atualização e exclusão, respeitando as regras de negócio como limites de desconto, vínculos obrigatórios e controles de status diferenciados. Espera-se que o *Middleware* impeça acessos não autorizados e valide todos os dados submetidos, garantindo a consistência da base SQLite. Os serviços REST retornarão respostas JSON padronizadas, permitindo a fácil integração com o frontend e futura expansão para aplicações móveis.

As telas prototipadas no Figma serão implementadas com Tailwind CSS, resultando em interfaces responsivas e uniformes. Na tela de login, o usuário acessará o sistema mediante autenticação segura; na *dashboard*, serão exibidos atalhos para cada módulo, indicando status de pendências financeiras, solicitações em processamento e relatórios disponíveis. Nas páginas de cadastro e listagem de clientes, serviços e ordens de serviço, esperam-se controles de pesquisa e filtros dinâmicos, permitindo a navegação ágil em grandes volumes de dados. A seção de relatórios deverá oferecer gráficos e tabelas resumindo receitas, despesas, inadimplência e padrões de O.S. por cliente, com possibilidade de exportação em PDF ou CSV.

Como avanços futuros, propõe-se a incorporação de IA em três frentes: *chatbot* de suporte, detecção de discrepâncias financeiras e análise de padrões de O.S. Por meio de APIs de *machine learning*, o sistema poderá responder automaticamente a dúvidas frequentes no chat de suporte, identificar lançamentos atípicos em tempo real e gerar insights sobre comportamento de clientes com base em ordens de serviço recorrentes. Esses componentes deverão interagir com o banco de dados SQLite e os serviços do Flask, enriquecendo o sistema

sem comprometer a performance.

A combinação de arquitetura em camadas, práticas ágeis e ferramentas modernas tende a proporcionar um produto com alta manutenibilidade e boa experiência do usuário. A adoção de exclusão reversível assegura rastreabilidade completa dos registros, enquanto o uso de TypeScript e validações de *Middleware* minimiza erros em produção. Tornando-o não apenas um gerenciador de dados, mas também uma ferramenta analítica capaz de apoiar decisões estratégicas. A avaliação dos resultados se dará por meio de testes de usabilidade, métricas de performance da API e *feedback* de usuários-teste em ambiente controlado.

CONCLUSÃO

O desenvolvimento do sistema financeiro integrado para empresas de contabilidade demonstrou ser viável e alinhado às necessidades identificadas na fase de levantamento de requisitos. A arquitetura em camadas adotada com as ferramentas escolhidas permitiu a construção de um produto modular, de fácil manutenção e escalável. O uso de *Middleware* assegurou a segurança, a validação de dados e o registro de execuções. A metodologia iterativa e incremental, suportada por ferramentas de documentação no Notion, possibilitou entregas regulares e ajustes baseados em feedback, resultando em maior qualidade no produto.

A proposta de incorporação de Inteligência Artificial, abrangendo *chatbot* de suporte, detecção de discrepâncias financeiras e análise de padrões de ordens de serviço, caracteriza-se como um diferencial competitivo, capaz de elevar o sistema a um patamar analítico e preditivo. Futuras etapas incluem a implementação desses módulos de IA, a execução de testes de usabilidade e performance em ambiente real e a avaliação contínua com usuários para garantir a aderência às demandas do setor contábil.

REFERÊNCIAS BIBLIOGRÁFICAS

AWS. **O que é middleware? – Explicação sobre software de middleware – AWS**. Disponível em: <https://aws.amazon.com/what-is/middleware/>. Acesso em: 3 set. 2025.

CONSELHO FEDERAL DE CONTABILIDADE. **Saiba quais serão as 6 tendências para as empresas contábeis em 2022**. Disponível em: https://cfc.org.br/noticias/confira-as-6-tendencias-para-as-empresas-contabeis-em-2022/. Acesso em: 16 mar. 2025

FIGMA. *Centro de Ajuda – Figma.* Disponível em: https://help.figma.com/hc/en-us. Acesso em: 5 jun. 2025.

FLASK. *Documentação oficial do Flask.* Disponível em: https://flask.palletsprojects.com/. Acesso em: 5 jun. 2025.

NOTION. *Documentação – Notion.* Disponível em: https://www.notion.com/pt/help/guides/category/documentation. Acesso em: 5 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *Python Release Python 3.12.6.* Disponível em: https://www.python.org/downloads/release/python-3126/. Acesso em: 5 jun. 2025.

VISTA do Sistema de gerenciamento e controle interno: uma análise dos escritórios de contabilidade de Belo Horizonte/MG e Região Metropolitana a partir da taxonomia de Kaplan e Cooper. Disponível em: https://anaiscbc.emnuvens.com.br/anais/article/view/731/731 Acesso em: 20 mar. 2025.

VISUAL STUDIO CODE. *Documentação do Visual Studio Code.* Disponível em: https://code.visualstudio.com/docs. Acesso em: 5 jun. 2025.