v7.n11.2026

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

L.I.L.Y. (LÓGICA INTEGRADA E INTELIGÊNCIA DE ORÇAMENTOS)

L.I.L.Y. (INTEGRATED LOGIC AND QUOTE INTELLIGENCE)

L.I.L.Y. (LÓGICA INTEGRADA E INTELIGENCIA DE PRESUPUESTOS)

Murilo Bueno de Godoy[1], João Henrique Gião Borges[2], André Luiz da Silva[3]

e7119498
https://doi.org/10.47820/recima21.v7i11.9498

PUBLICADO: 11/2026

RESUMO

Este trabalho apresenta o desenvolvimento da L.I.L.Y., uma aplicação *web* voltada para a automação e a gestão de orçamentos em oficinas mecânicas especializadas em radiadores. O problema central reside na falta de padronização e de agilidade nos orçamentos elaborados manualmente, o que compromete a precisão financeira e a rentabilidade do negócio. A metodologia caracteriza-se como pesquisa aplicada, com estudo de caso em uma oficina de radiadores e desenvolvimento iterativo de um protótipo funcional construído com React, Vite e TypeScript, persistência em nuvem por meio do Firebase, serviços de retaguarda em Java com Spring Boot e um motor de inteligência artificial em Python integrado aos modelos de linguagem Gemini e Groq, com interação por texto e por voz. Os resultados mostraram que a parametrização dos custos e a automatização do cálculo de venda, custo e lucro eliminaram a variação entre orçamentos de um mesmo serviço, com concordância de 100% em relação aos valores de referência em 10.000 casos de teste, centralizaram o histórico de contas por usuário e permitiram ao assistente virtual converter comandos em linguagem natural em ações na aplicação, com taxa de acerto de 90% em 40 execuções e nenhuma ação indevida. Conclui-se que a combinação de tecnologias modernas de desenvolvimento *web* com serviços em nuvem e modelos de linguagem permite que microempresas otimizem seus processos internos com baixo custo de infraestrutura.

PALAVRAS-CHAVE: Automação de processos. Desenvolvimento *web*. Firebase. Inteligência artificial. Orçamentos. React.

ABSTRACT

This paper presents the development of L.I.L.Y., a web application aimed at automating and managing quotes in mechanical workshops specializing in radiators. The central problem lies in the lack of standardization and agility in manually prepared quotes, which compromises financial accuracy and business profitability. The methodology is characterized as applied research, with a case study in a radiator workshop and the iterative development of a functional prototype built with React, Vite and TypeScript, cloud persistence through Firebase, back-end services in Java with Spring Boot, and an artificial intelligence engine in Python integrated with the Gemini and Groq language models, supporting text and voice interaction. The results showed that cost parameterization and the automation of selling price, cost and profit calculations eliminated the variation between quotes for the same service, with 100% agreement with reference values in 10,000 test cases, centralized the history of accounts per user, and allowed the virtual assistant to convert natural-language commands into actions in the application, with a 90% success rate in 40 runs and no improper actions.

[1] Graduando em Sistemas de Informação pela Universidade de Araraquara (UNIARA), com atuação em desenvolvimento de *software*.
[2] Doutor em Química Orgânica pela Universidade de São Paulo (USP) e docente do curso de Sistemas de Informação da Universidade de Araraquara (UNIARA).
[3] Doutor em Educação Escolar pela Universidade Estadual Paulista (UNESP) e docente do curso de Sistemas de Informação da Universidade de Araraquara.

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.

It is concluded that combining modern web development technologies with cloud services and language models allows micro-businesses to optimize their internal processes at a low infrastructure cost.

KEYWORDS: *Artificial intelligence. Firebase. Process automation. Quotes. React. Web development.*

RESUMEN

Este trabajo presenta el desarrollo de L.I.L.Y., una aplicación web orientada a la automatización y la gestión de presupuestos en talleres mecánicos especializados en radiadores. El problema central radica en la falta de estandarización y de agilidad en los presupuestos elaborados manualmente, lo que compromete la precisión financiera y la rentabilidad del negocio. La metodología se caracteriza como investigación aplicada, con un estudio de caso en un taller de radiadores y el desarrollo iterativo de un prototipo funcional construido con React, Vite y TypeScript, persistencia en la nube mediante Firebase, servicios de back-end en Java con Spring Boot y un motor de inteligencia artificial en Python integrado con los modelos de lenguaje Gemini y Groq, con interacción por texto y por voz. Los resultados mostraron que la parametrización de los costos y la automatización del cálculo del precio de venta, del costo y de la ganancia eliminaron la variación entre presupuestos de un mismo servicio, con una concordancia del 100% con los valores de referencia en 10.000 casos de prueba, centralizaron el historial de cuentas por usuario y permitieron que el asistente virtual convirtiera comandos en lenguaje natural en acciones en la aplicación, con una tasa de acierto del 90% en 40 ejecuciones y ninguna acción indebida. Se concluye que la combinación de tecnologías modernas de desarrollo web con servicios en la nube y modelos de lenguaje permite que las microempresas optimicen sus procesos internos con un bajo costo de infraestructura.

PALABRAS CLAVE: *Automatización de procesos. Desarrollo web. Firebase. Inteligencia artificial. Presupuestos. React.*

INTRODUÇÃO

O avanço das tecnologias digitais tem impulsionado a modernização dos processos administrativos em empresas de diferentes segmentos, incluindo pequenos negócios que tradicionalmente utilizam métodos manuais de gestão. No Brasil, 76% dos pequenos negócios já utilizam computador na operação e 44% declaram usar alguma forma de inteligência artificial (SEBRAE, 2025); ainda assim, apenas cerca de metade utiliza *softwares* que integram diferentes áreas da empresa e menos de 13% usam a internet para plataformas de gestão (SEBRAE-PR, 2025), o que indica que a digitalização dos processos internos permanece incompleta. Nesse contexto, este trabalho apresenta o desenvolvimento da L.I.L.Y. (Lógica Integrada e Inteligência de Orçamentos), uma aplicação *web* projetada para automatizar a elaboração de orçamentos, gerenciar dados operacionais e incorporar recursos de Inteligência Artificial (IA) para auxiliar processos de identificação e consulta de informações. O nome L.I.L.Y. identifica a assistente virtual que dá identidade à aplicação e com a qual o usuário conversa por texto e por voz; o

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.

subtítulo Lógica Integrada e Inteligência de Orçamentos descreve o seu propósito, que é reunir em um único fluxo a lógica de cálculo dos orçamentos e os recursos de inteligência artificial que auxiliam o usuário.

A problemática que motivou esta pesquisa está relacionada à falta de padronização nos orçamentos de microempresas, frequentemente calculados de forma manual, com base em estimativas empíricas que não consideram adequadamente custos de insumos, despesas logísticas, variações de mercado e horas técnicas. O cadastro manual de produtos e componentes torna o processo lento e suscetível a erros, o que compromete a precisão financeira dos orçamentos, uma vez que a formação do preço de venda exige conhecer os custos envolvidos e aplicar sobre eles margens que cubram despesas e remunerem o negócio (BRUNI; FAMÁ, 2012).

Diante desse cenário, a questão de pesquisa que orienta este trabalho é: de que forma uma aplicação *web*, apoiada em serviços de nuvem e em recursos de inteligência artificial, pode padronizar e agilizar a elaboração de orçamentos em uma microempresa de reparação de radiadores? Parte-se da hipótese de que a implementação de uma ferramenta digital capaz de parametrizar custos, automatizar cálculos e utilizar recursos de inteligência artificial para assistência ao usuário e reconhecimento de componentes pode aumentar a precisão dos orçamentos, reduzir falhas humanas e contribuir para uma gestão financeira mais eficiente.

A relevância da pesquisa está associada à crescente necessidade de digitalização dos processos de pequenos negócios. Sob a perspectiva social e econômica, a proposta contribui para democratizar o acesso à tecnologia em oficinas e prestadores de serviços; no âmbito acadêmico, demonstra a aplicação de tecnologias modernas de desenvolvimento *web* associadas a modelos de linguagem acessados por meio das Interfaces de Programação de Aplicações (APIs) Gemini e Groq, para assistência ao usuário e para síntese e reconhecimento de voz, além da preparação de um módulo de análise de imagens. A solução pode, ainda, servir de modelo para outros setores que enfrentam desafios de gestão de orçamentos, controle operacional e automação de processos.

O objetivo geral deste trabalho é desenvolver uma aplicação *web* capaz de automatizar o ciclo de elaboração de orçamentos e auxiliar na gestão de dados operacionais. Como objetivos específicos, pretende-se: (i) sistematizar o cálculo de preços por meio de parâmetros dinâmicos; (ii) implementar funcionalidades para registro e consulta de históricos financeiros; (iii) centralizar informações de clientes e serviços; (iv) disponibilizar recursos de assistência automatizada por meio de um assistente conversacional integrado, com interação por texto e por voz; e (v)

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.

desenvolver mecanismos de reconhecimento de peças por imagem para reduzir o preenchimento manual de dados.

Trata-se de uma pesquisa aplicada, conduzida por meio de estudo de caso em uma oficina de reparação de radiadores e do desenvolvimento iterativo de um protótipo funcional. O artigo apresenta o referencial teórico na seção 1; a metodologia, a arquitetura e os procedimentos de validação na seção 2; os resultados na seção 3; e as considerações finais na seção 4.

1. REFERENCIAL TEÓRICO

1.1. Sistemas de informação e a transformação digital em pequenos negócios

A sobrevivência de pequenas empresas no mercado atual depende da transição de modelos de gestão manuais para sistemas automatizados. De acordo com Rezende e Abreu (2013), os Sistemas de Informação (SI) não são apenas ferramentas tecnológicas, mas processos integrados que geram valor estratégico ao negócio. Laudon e Laudon (2014) definem um sistema de informação como um conjunto de componentes inter-relacionados que coleta, processa, armazena e distribui informações para apoiar a tomada de decisão e o controle da organização; sob essa ótica, um orçamento elaborado à mão e não registrado é informação que a empresa produz, mas não retém nem consegue analisar.

Os levantamentos do Sebrae mostram que a tecnologia nos pequenos negócios brasileiros é alta na infraestrutura, mas rasa na gestão: o uso concentra-se em redes sociais e serviços bancários, e apenas uma parcela adota sistemas que integram as áreas da empresa (SEBRAE, 2025; SEBRAE-PR, 2025). É nessa lacuna, entre ter o computador e usá-lo para controlar custos, clientes e serviços, que se insere a L.I.L.Y.

No ramo de manutenção automotiva, a precisão na coleta de dados, como custos de insumos, frete, mão de obra e tempo de execução, define a rentabilidade de cada serviço. Uma ferramenta centralizada, que registre esses dados no momento do orçamento, reduz erros humanos comuns em orçamentos empíricos e transforma cada serviço fechado em um registro consultável para decisões futuras, como o reajuste de preços.

1.2. Arquitetura de *Software Frontend*: React e Vite

A escolha de tecnologias para a interface do usuário impacta diretamente a performance e a manutenção do sistema. A biblioteca React introduziu o conceito de componentes reutilizáveis e de interface declarativa, na qual a tela é descrita como função do estado da aplicação (META, 2026). Segundo Flanagan (2013), o isolamento de responsabilidades em

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.

unidades independentes de código otimiza o fluxo de desenvolvimento em JavaScript e reduz o acoplamento entre as partes da interface. Complementarmente, a utilização do Vite como ferramenta de *build* resolve problemas históricos de lentidão no ambiente de desenvolvimento, utilizando módulos nativos do navegador para uma inicialização quase instantânea e gerando pacotes otimizados para produção (VITE, 2026).

1.3. Persistência de dados e computação em nuvem (Firebase)

O armazenamento de informações da L.I.L.Y. fundamenta-se na tecnologia de computação em nuvem. O Firebase, plataforma utilizada neste projeto, oferece uma infraestrutura do tipo *Backend as a Service* (BaaS), reunindo autenticação, banco de dados e armazenamento de arquivos com regras de segurança declarativas (GOOGLE, 2026a). Conforme aponta Taurion (2009), o modelo de computação em nuvem permite que desenvolvedores independentes tenham acesso a recursos de alta disponibilidade e segurança sem a necessidade de gerenciar servidores físicos.

Os bancos de dados NoSQL, como o Cloud Firestore (integrante do ecossistema Firebase), oferecem uma flexibilidade de esquema que os bancos relacionais tradicionais não proporcionam prontamente para a prototipagem rápida. Heuser (2009) destaca que o projeto de banco de dados deve partir das necessidades de consulta da aplicação, e Sadalage e Fowler (2013) caracterizam os bancos orientados a documentos pela ausência de esquema fixo e pela modelagem por agregados, em que cada documento reúne os dados que costumam ser lidos e gravados juntos. Na L.I.L.Y., cada conta é um agregado desse tipo, pois contém o veículo, a peça, o cliente e todos os valores do orçamento em um único documento, o que permite listar o histórico com uma só consulta, sem junções entre tabelas.

1.4. Usabilidade e acessibilidade (W3C e UX)

Em um ambiente dinâmico como uma oficina, a usabilidade e a experiência do usuário (*User Experience* – UX) são críticas: segundo Nielsen (2000), um *software* deve ser fácil de aprender e eficiente de usar, minimizando a carga cognitiva do usuário. As Diretrizes de Acessibilidade para Conteúdo *Web* (*Web Content Accessibility Guidelines* – WCAG), mantidas pelo World Wide Web Consortium (W3C), estabelecem que o conteúdo deve ser perceptível, operável e compreensível, o que se traduz em rótulos e instruções associados a cada campo, identificação clara dos erros de preenchimento, sugestões de correção e navegação previsível (W3C, 2023). Embora concebidas para pessoas com deficiência, essas recomendações

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.

reduzem erros de entrada de dados para todos os usuários, o que é especialmente relevante em um sistema de orçamentos.

No projeto L.I.L.Y., a aplicação dessas diretrizes reflete-se no uso de máscaras de entrada para CPF/CNPJ e telefone, rótulos flutuantes que permanecem legíveis após o preenchimento, avisos visuais que confirmam o sucesso das operações e interface bilíngue (português e inglês).

1.5. Formação do preço de venda e *markup*

O objeto central da L.I.L.Y. é o orçamento, e um orçamento é, em essência, a formação de um preço de venda. Martins (2010) distingue custos, que são gastos relativos à produção do bem ou serviço, de despesas, que são gastos para obter receitas; ambos precisam ser cobertos pelo preço. Bruni e Famá (2012) descrevem o método do *markup*, no qual o preço é obtido pela aplicação de um índice multiplicador sobre o custo, dimensionado para cobrir despesas variáveis, despesas fixas, tributos e a margem de lucro desejada. O *markup* pode ser expresso como um único multiplicador ou decomposto em parcelas sucessivas, cada uma associada a um componente que o preço precisa remunerar.

Em uma oficina de radiadores, o custo direto do serviço compõe-se da peça ou insumo, do frete e da mão de obra, somando-se, nos serviços com montagem, material, horas técnicas, encargos e a própria montagem. A prática usual de aplicar percentuais de memória sobre esses custos dificulta manter a coerência entre orçamentos e verificar se a margem cobre as despesas (BRUNI; FAMÁ, 2012); a parametrização dessas margens em um motor de cálculo, como o descrito na seção 2.4, torna o *markup* explícito, repetível e auditável.

1.6. Inteligência Artificial Generativa e modelos de linguagem

Os modelos de linguagem de grande porte (*Large Language Models* – LLM) baseiam-se na arquitetura Transformer, proposta por Vaswani et al. (2017), cujo mecanismo de atenção permite processar sequências de texto de forma paralela e capturar dependências de longo alcance. Tais modelos derivam dos fundamentos do aprendizado profundo descritos por Goodfellow, Bengio e Courville (2016), que também sustentam as redes neurais convolucionais empregadas em tarefas de visão computacional, como a classificação de imagens.

Na prática, esses modelos são consumidos por meio de APIs que recebem uma instrução de sistema, definindo o papel e as regras do assistente, e a mensagem do usuário, devolvendo texto gerado (GOOGLE, 2026b). Essa abordagem permite que aplicações de negócio incorporem assistentes conversacionais sem treinar modelos próprios, desde que o

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.

contexto relevante do domínio seja fornecido no *prompt*. Para a análise de imagens diretamente no navegador, bibliotecas como o TensorFlow.js possibilitam executar modelos pré-treinados no cliente, sem envio das fotografias a um servidor (TENSORFLOW, 2026).

1.7. Processo de Desenvolvimento de *Software*

Pressman e Maxim (2016) descrevem os modelos de processo evolucionários, como a prototipação e o desenvolvimento incremental, como adequados a projetos cujos requisitos são refinados a partir do contato do usuário com versões funcionais do produto. Esse foi o caso da L.I.L.Y.: cada incremento (autenticação, calculadora, histórico de contas, assistente por voz) foi disponibilizado à oficina e ajustado a partir do retorno obtido, o que orientou, por exemplo, a reorganização da navegação e a criação de dois modos de cálculo.

2. METODOLOGIA

Esta pesquisa caracteriza-se como aplicada, de natureza descritiva, com abordagem qualiquantitativa. Os procedimentos técnicos compreendem pesquisa bibliográfica, análise documental e estudo de caso realizado em uma oficina automotiva especializada em radiadores. O desenvolvimento seguiu um processo iterativo e incremental, no qual protótipos funcionais foram entregues e refinados a partir do uso na oficina. A validação foi realizada por meio da comparação entre o método tradicional de elaboração de orçamentos e os resultados gerados pela aplicação, considerando a precisão e a repetibilidade dos cálculos, a organização das informações e a eficiência operacional, e complementada por testes quantitativos de correção, repetibilidade e desempenho, cujo protocolo é descrito na seção 2.9.

As subseções a seguir descrevem a arquitetura da L.I.L.Y., os módulos implementados e os procedimentos de validação. O código-fonte é mantido em repositório público no GitHub e está organizado em três projetos: o aplicativo (lily-app), a API de serviços (lily-api) e o motor de inteligência artificial (engine).

2.1. Arquitetura geral da aplicação

A Figura 1 apresenta a arquitetura geral da solução, organizada em quatro camadas. A camada de acesso compreende o navegador, no qual a aplicação é executada como *Progressive Web App* (PWA) hospedada na plataforma Netlify, e o aplicativo *desktop*, gerado com o *framework* Tauri, que empacota a mesma interface em um executável nativo e é responsável por

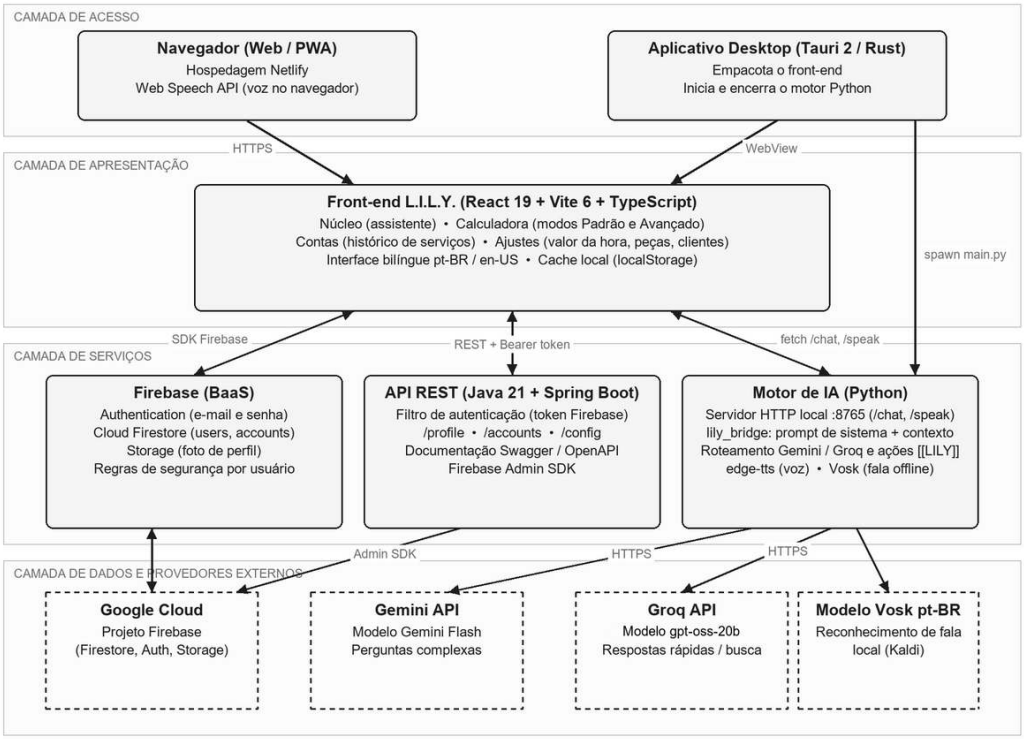
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

iniciar o motor de voz local (TAURI, 2026). Ambas as versões compartilham o mesmo projeto Firebase e, portanto, os mesmos dados.

A camada de apresentação é o *front-end* escrito em React 19, TypeScript e Vite 6, com navegação por um trilho lateral com quatro destinos: Núcleo (tela inicial e assistente), Calculadora, Contas e Ajustes. A camada de serviços reúne o Firebase (autenticação, banco de dados Cloud Firestore e armazenamento de arquivos), a API em Java 21 com Spring Boot, no padrão *Representational State Transfer* (REST), e o motor de IA em Python, que atende o *chat* e a voz por meio de um servidor HTTP local. A camada de dados e provedores externos contém o projeto Firebase no Google Cloud, as APIs Gemini e Groq e o modelo de reconhecimento de fala Vosk em português.

A separação em camadas permite que o *front-end* funcione mesmo quando algum serviço está indisponível: sem as credenciais do Firebase, os dados são mantidos no armazenamento local do navegador; sem o motor Python, o assistente responde por um modo local simplificado e a voz utiliza a Web Speech API do próprio navegador.

Figura 1. Arquitetura geral da aplicação



Fonte: elaborado pelos autores (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.

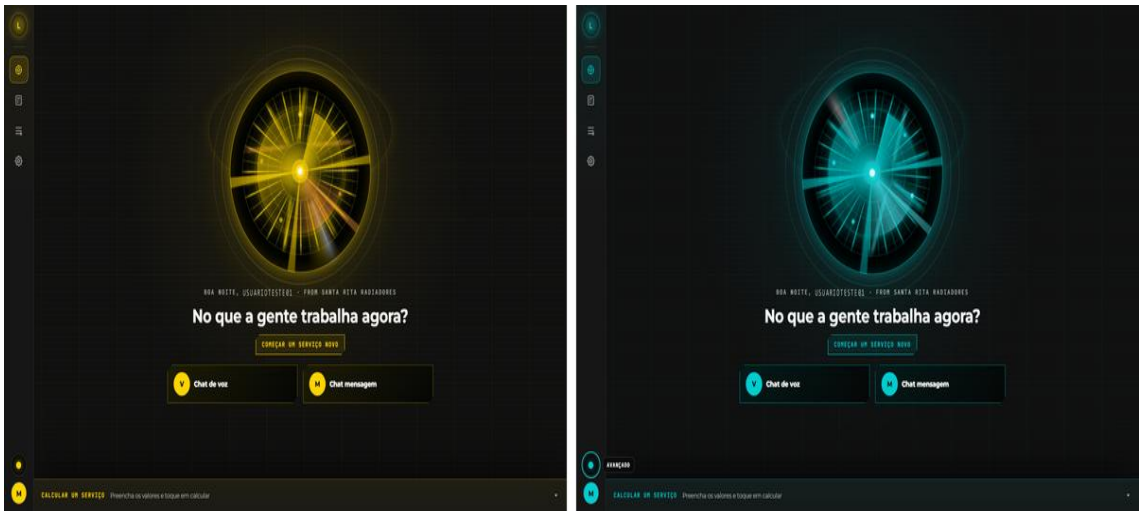
2.2. Autenticação e cadastro de usuários

O controle de acesso utiliza o Firebase Authentication, com autenticação por *e-mail* e senha, sessões persistentes e redefinição de senha. O cadastro distingue pessoa física e pessoa jurídica, aplica máscaras de CPF/CNPJ e telefone, avalia a força da senha e exige o aceite dos termos de uso; os dados ficam no Cloud Firestore, na coleção *users*, identificados pelo código único do usuário autenticado, o que permite sua sincronização entre dispositivos.

2.3. Página inicial e modos de cálculo

A página inicial (Núcleo) concentra o acesso às funcionalidades e abriga o assistente virtual. A calculadora possui dois modos, identificados por cores (Figura 2): o padrão (amarelo), com três campos de entrada, e o avançado (azul), que acrescenta material, horas de serviço e INSS, totalizando seis campos, e calcula também o valor da montagem. O modo é trocado por um seletor no trilho lateral e também filtra a tela de contas, o que deixa claro o contexto de operação e reduz o risco de registrar um serviço com o conjunto errado de custos.

Figura 2. Página inicial nos modos padrão (amarelo, à esquerda) e avançado (azul, à direita)



Fonte: elaborado pelos autores (2026).

2.4. Motor de cálculo de orçamentos

O núcleo funcional da L.I.L.Y. é o motor de cálculo, implementado como uma função pura no *front-end*, que recebe os campos preenchidos, o modo ativo e o valor da hora configurado e devolve o preço de venda, o custo e o lucro. No modo padrão, a base do orçamento é a soma do valor inicial da peça, do frete e do custo de funcionário, conforme a Equação 1.

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.

$$B = Vi + F + Fu \quad (1)$$

Onde: *B* é a base do orçamento; *Vi*, o valor inicial da peça ou insumo; *F*, o frete; e *Fu*, o custo de funcionário, todos em R$.

Sobre a base são aplicadas três margens sucessivas, de 35%, 20% e 10%, que resultam no preço sugerido de venda (Equação 2). O custo total considera a base acrescida das parcelas correspondentes às margens intermediárias (Equação 3), e o lucro final é a diferença entre venda e custo (Equação 4). Em termos do método do *markup* (BRUNI; FAMÁ, 2012), as três margens sucessivas equivalem a um multiplicador único de 1,782 sobre a base; como o custo total corresponde a 1,432 vezes a base, o lucro final resulta sempre em 35% da base do orçamento, independentemente dos valores informados.

$$V = B \times 1{,}35 \times 1{,}20 \times 1{,}10 \quad (2)$$

$$C = B + 0{,}20 \times (B \times 1{,}35) + 0{,}10 \times (B \times 1{,}35 \times 1{,}20) \quad (3)$$

$$L = V - C \quad (4)$$

Onde: *V* é o preço sugerido de venda; *C*, o custo total do serviço; e *L*, o lucro final, todos em R$.

No modo avançado, acrescenta-se a base de montagem (Equação 5), composta pelo material, pelas horas técnicas multiplicadas pelo valor da hora configurado em Ajustes, pelo INSS e pelo funcionário, sobre a qual se aplica a margem de 30% (Equação 6); o custo da montagem soma o custo do modo padrão à base de montagem, e o valor de montagem mais venda soma a montagem ao preço de venda (Equações 7 e 8).

$$Bm = M + H \times Vh + I + Fu \quad (5)$$

$$Mo = Bm \times 1{,}30 \quad (6)$$

$$Cm = C + Bm \quad (7)$$

$$Mv = V + Mo \quad (8)$$

Onde: *Bm* é a base de montagem; *M*, o material; *H*, as horas de serviço (h); *Vh*, o valor da hora técnica configurado (R$/h); *I*, o INSS; *Mo*, o valor da montagem; *Cm*, o custo da montagem; e *Mv*, a montagem mais venda, com os valores monetários em R$.

As Equações 1 a 8 são implementadas no *front-end* por uma função pura, que devolve sempre o mesmo resultado para as mesmas entradas, o que facilita sua verificação por testes automatizados (seção 2.9). As margens são fixas no código, refletindo a política de preços praticada pela oficina do estudo de caso; o único parâmetro ajustável pela interface que participa do cálculo é o valor da hora técnica, e essa decisão, que simplificou a adoção pela equipe, é discutida como limitação na seção 3.

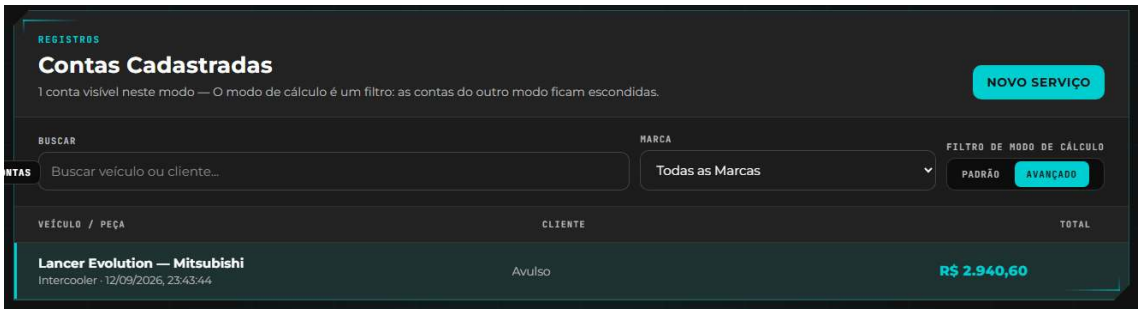
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2.5. Configurações e gerenciamento de contas

O módulo de configurações (Ajustes) permite definir o valor da hora técnica utilizado no modo avançado, manter a lista de tipos de peça (radiador, caixa, *intercooler*, condensador e outros) e cadastrar clientes, pessoa física ou jurídica, posteriormente vinculados aos serviços; as configurações são sincronizadas com o Firestore e mantidas em *cache* local para uso imediato.

No módulo de contas, cada conta representa um serviço fechado, com marca, veículo, tipo de peça, cliente, data, modo de cálculo e todos os valores de entrada e de saída do orçamento, e a tela (Figura 3) permite buscar por veículo ou cliente, filtrar por marca, reabrir uma conta para edição e excluí-la. Os registros ficam na coleção *accounts* do Cloud Firestore, identificados pelo campo *user_id*, e as regras de segurança do banco só autorizam leitura, criação, alteração ou exclusão quando o usuário está autenticado e o *user_id* do documento coincide com o seu identificador, o que garante o isolamento dos dados entre usuários. A listagem por data decrescente exige um índice composto no Firestore.

Figura 3. Tela de gerenciamento de contas



Fonte: elaborado pelos autores (2026).

2.6. API de serviços em Java

Foi construída também uma API REST em Java 21 com Spring Boot (lily-api), que centraliza as regras de acesso aos dados e prepara a futura migração para um banco relacional. A API expõe os recursos de perfil (/api/profile), contas (/api/accounts) e configurações (/api/config), além de uma rota pública de disponibilidade (/api/health), com documentação gerada por Swagger/OpenAPI. Um filtro valida, em cada requisição, o *token* do Firebase Authentication enviado no cabeçalho *Authorization*: sem *token* válido, a API responde com o código 401 (não autorizado); com *token* válido, os serviços operam apenas sobre os documentos do usuário autenticado.

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.

2.7. Inteligência Artificial conversacional

Uma das principais funcionalidades implementadas na L.I.L.Y. é o assistente virtual, que interage com os usuários por meio de mensagens textuais e comandos de voz (Figura 4). O assistente é atendido pelo motor de IA em Python, exposto ao *front-end* por um servidor HTTP local com as rotas /chat e /speak. A cada mensagem, o *front-end* envia também um resumo do estado da tela – modo ativo, valor da hora, campos preenchidos, resultado calculado e serviço em andamento – que é incorporado à instrução de sistema, de modo que o assistente responda sobre os números que o usuário está vendo, sem recalculá-los nem inventar valores.

O motor roteia as mensagens entre dois provedores: perguntas longas, com mais de 120 caracteres, ou que pedem análise, comparação ou explicação são enviadas ao modelo Gemini, cuja resposta é resumida por um modelo da Groq para manter o tom curto exigido pela leitura em voz alta; pedidos de pesquisa e mensagens simples vão diretamente à Groq, que dispõe de busca na *web* integrada. Cada provedor só é acionado se a chave de API correspondente estiver configurada; falhas de rede, esgotamento de cota ou ausência de chaves geram respostas de contingência, e, se o motor não estiver em execução, o *front-end* responde com um conjunto de respostas locais.

Figura 4. *Chat* textual integrado à Inteligência Artificial



Fonte: elaborado pelos autores (2026).

O assistente também age sobre a aplicação: a instrução de sistema orienta o modelo a encerrar a resposta com um bloco estruturado no formato [[LILY:{...}]], com o tipo de ação (calcular, trocar de modo, salvar a conta, navegar entre telas ou preencher o formulário de conta) e seus parâmetros. O motor localiza o bloco por expressão regular, remove-o do texto antes da

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.

exibição e da síntese de voz, interpreta-o como JSON e aceita apenas tipos e campos permitidos e valores numéricos finitos, descartando o restante; no *front-end*, a ação validada preenche os campos, executa o cálculo, troca o modo, muda de tela ou abre o formulário de registro.

No navegador, o reconhecimento e a síntese de fala utilizam a Web Speech API; com o motor ativo, a resposta é sintetizada com a voz neural pt-BR-FranciscaNeural por meio da biblioteca edge-tts, de conversão de texto em fala (*Text-to-Speech* – TTS). No aplicativo *desktop*, o Tauri inicia o processo Python que realiza o reconhecimento de fala *offline* com o modelo Vosk em português (ALPHA CEPHEI, 2026), no modo *push-to-talk*, encaminha o texto ao motor de IA e reproduz a resposta falada.

2.8. Reconhecimento de Imagens

A plataforma possui uma funcionalidade voltada para o reconhecimento de imagens no cadastro de contas, cujo objetivo é permitir que o usuário envie ou capture uma fotografia da peça e tenha os campos de marca, veículo e tipo de peça preenchidos automaticamente, reduzindo a inserção manual de dados.

Na versão atual, o fluxo de interface está completo (seleção ou captura da imagem, validação do formato JPG, PNG ou WEBP, pré-visualização e preenchimento do formulário), mas a identificação é realizada, provisoriamente, por uma heurística que compara o nome do arquivo com as marcas e os tipos de peça cadastrados. A substituição dessa heurística por um modelo de classificação executado no navegador com TensorFlow.js, treinado com fotografias de radiadores, caixas, *intercoolers* e condensadores, encontra-se em desenvolvimento e é discutida nas recomendações de trabalhos futuros.

2.9. Procedimentos de validação

A validação combinou três procedimentos quantitativos, executados sobre a versão do código-fonte de 29 de setembro de 2026 em um computador com processador AMD Ryzen 7 5800X, 24 GB de memória e Windows 11, com Node.js 24 e Python 3.13. No primeiro, a função de cálculo foi extraída do código-fonte da aplicação e executada sobre 10.000 casos gerados aleatoriamente com semente fixa, 5.000 em cada modo, com valor inicial entre R$ 10,00 e R$ 8.000,00, frete de até R$ 300,00, funcionário e material de até R$ 1.000,00 cada, até 12 horas de serviço, valor da hora entre R$ 20,00 e R$ 150,00 e INSS de até R$ 200,00. Os resultados foram comparados com uma implementação de referência das Equações 1 a 8 em aritmética inteira exata, considerando-se correto o caso em que todas as grandezas diferiram da referência em menos de meio centavo (R$ 0,005); mediram-se ainda a repetibilidade, em 100.000

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.

execuções de uma mesma entrada, e o tempo médio de processamento, em 1.000.000 de execuções.

No segundo procedimento, executaram-se os 45 testes unitários do motor de IA, que verificam a extração e a validação dos blocos de ação diante de blocos válidos, vazios, truncados ou malformados, de tipos e campos não permitidos, de valores não numéricos ou infinitos e de textos com caracteres de controle.

No terceiro procedimento, 20 mensagens representativas do uso na oficina, sendo 15 comandos operacionais (calcular, trocar de modo, salvar a conta e navegar entre telas) e cinco mensagens que não devem gerar ação (cumprimentos, dúvidas sobre o aplicativo e pedidos não suportados, como apagar todas as contas ou alterar a margem de lucro), foram enviadas ao motor em duas rodadas, com o modelo openai/gpt-oss-20b na Groq, totalizando 40 execuções: na primeira, com intervalo de 2 s entre as mensagens, simulando uso intenso; na segunda, com intervalo de 25 s, mais próximo do ritmo de um atendimento. Registraram-se o tempo de resposta e a correspondência entre a ação extraída e a esperada (taxa de acerto). Mediu-se também o tempo de resposta de cinco perguntas de explicação, encaminhadas ao modelo Gemini (gemini-3.6-flash) e resumidas em seguida pela Groq.

3. RESULTADOS E DISCUSSÃO

3.1. Atendimento aos objetivos específicos

O Quadro 1 relaciona os objetivos específicos definidos na introdução com os módulos implementados e a situação de cada um ao término do trabalho.

Quadro 1. Situação dos objetivos específicos ao término do trabalho

Objetivo específico	Módulo(s) da L.I.L.Y.	Situação
(i) Sistematizar o cálculo de preços por parâmetros dinâmicos	Calculadora com modos padrão e avançado; valor da hora configurável	Implementado
(ii) Registrar e consultar históricos financeiros	Módulo de contas com busca, filtro por marca e por modo, edição e exclusão; Firestore com regras por usuário	Implementado
(iii) Centralizar informações de clientes e serviços	Ajustes (clientes e tipos de peça); vínculo cliente–conta; API REST	Implementado
(iv) Assistência automatizada por assistente conversacional	Motor de IA (Gemini/Groq), *chat* textual, voz no navegador e no *desktop*, ações estruturadas	Implementado

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.

Objetivo específico	Módulo(s) da L.I.L.Y.	Situação
(v) Reconhecimento de peças por imagem	Fluxo de captura/envio e preenchimento automático com heurística sobre metadados	Parcialmente implementado – classificador por rede neural em desenvolvimento

Fonte: elaborado pelos autores (2026).

Quatro dos cinco objetivos específicos foram integralmente atingidos; o quinto foi entregue em versão inicial, com a interface e a integração ao formulário prontas para receber o classificador de imagens. Em síntese, estão implementados e em uso a calculadora nos dois modos, o histórico de contas, os cadastros de clientes e de tipos de peça, a API REST e o assistente por texto e por voz; permanecem em desenvolvimento o classificador de imagens por rede neural, a parametrização das margens pela interface e a migração para um banco de dados relacional.

3.2. Validação do motor de cálculo

Para verificar a correção e a repetibilidade do motor de cálculo, foram executados casos de teste com valores de entrada representativos de um serviço de radiador na oficina do estudo de caso. A Tabela 1 apresenta um desses casos, com valor inicial de R$ 1.000,00, frete de R$ 100,00 e funcionário de R$ 200,00 no modo padrão e, no modo avançado, material de R$ 150,00, duas horas de serviço com valor da hora de R$ 40,00 e INSS de R$ 50,00.

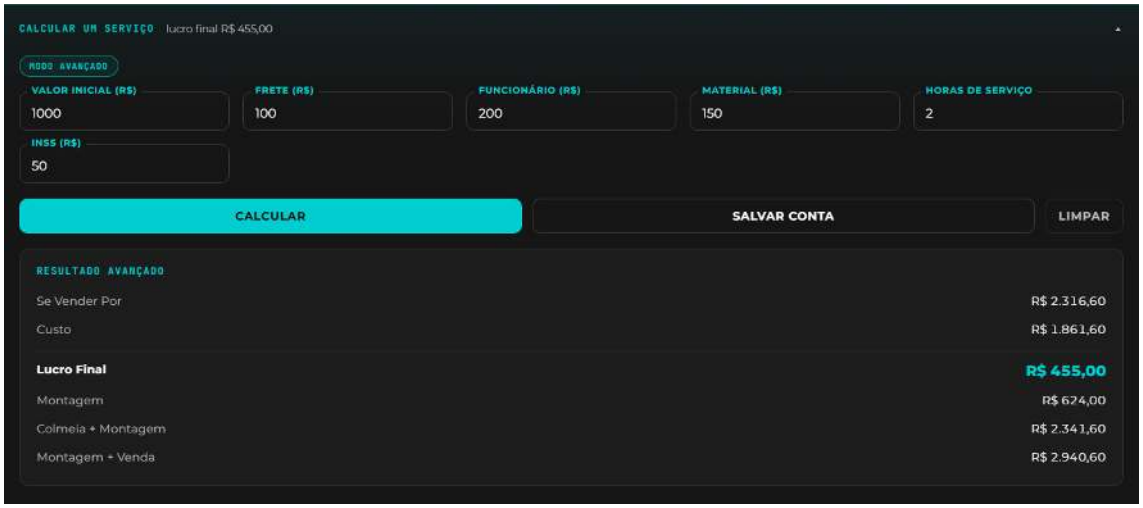
Tabela 1. Resultado do motor de cálculo para um caso de teste

Grandeza	Modo padrão (R$)	Modo avançado (R$)
Base do orçamento (B)	1.300,00	1.300,00
Preço sugerido de venda (V)	2.316,60	2.316,60
Custo total (C)	1.861,60	1.861,60
Lucro final (L)	455,00	455,00
Base de montagem (Bm)	–	480,00
Montagem (Mo)	–	624,00
Custo da montagem (Cm)	–	2.341,60
Montagem mais venda (Mv)	–	2.940,60

Fonte: elaborado pelos autores (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.

Figura 5. Calculadora com o resultado do caso de teste no modo avançado



Fonte: elaborado pelos autores (2026).

A Figura 5 mostra a calculadora com esse caso de teste executado na aplicação; os valores exibidos coincidiram com os calculados manualmente a partir das Equações 1 a 8 em todos os casos testados, e a repetição do mesmo conjunto de entradas produziu sempre o mesmo resultado, tanto na versão *web* quanto na versão *desktop*. O caso também confirma a propriedade derivada na seção 2.4: o lucro final de R$ 455,00 corresponde exatamente a 35% da base de R$ 1.300,00. No processo manual observado na oficina, o mesmo serviço podia receber valores diferentes conforme quem elaborava o orçamento e conforme a lembrança dos percentuais aplicados; com a L.I.L.Y., essa variação foi eliminada, uma vez que as margens passaram a ser aplicadas de forma sistemática.

A Tabela 2 consolida os indicadores obtidos no primeiro procedimento de validação descrito na seção 2.9.

Tabela 2. Indicadores de correção, repetibilidade e desempenho do motor de cálculo

Indicador	Resultado
Casos de teste avaliados (modo padrão + modo avançado)	10.000 (5.000 + 5.000)
Casos concordantes com a referência (diferença inferior a R$ 0,005)	10.000 (100%)
Erro absoluto máximo em relação à referência	$7{,}3 \times 10^{-12}$ R$
Casos em que o lucro final corresponde a 35% da base	10.000 (100%)
Execuções repetidas com resultado idêntico	100.000 de 100.000 (100%)
Tempo médio de processamento por cálculo	0,91 µs
Operações aritméticas manuais substituídas por orçamento (padrão / avançado)	10 / 17

Fonte: elaborado pelos autores (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.

O erro máximo, da ordem de 10^{-12} real, decorre da representação em ponto flutuante adotada pelo JavaScript e não afeta os valores exibidos com duas casas decimais. Com processamento inferior a um microssegundo, o esforço de elaboração do orçamento passa a se restringir ao preenchimento de três campos no modo padrão ou de seis no avançado, em lugar das 10 operações aritméticas manuais do modo padrão (duas somas e três multiplicações para o preço de venda, duas multiplicações e duas somas para o custo e uma subtração para o lucro) e das 17 do modo avançado.

3.3. Comparativo entre o processo manual e a L.I.L.Y.

O Quadro 2 sintetiza a comparação entre o método tradicional de orçamentação praticado na oficina, baseado em anotações e cálculo mental ou em calculadora, e o processo apoiado pela L.I.L.Y., considerando os aspectos definidos na metodologia.

Quadro 2. Comparação entre o processo manual e o processo com a L.I.L.Y.

Aspecto	Processo manual	Processo com a L.I.L.Y.
Padronização das margens	Dependente da memória do responsável; sujeita a variação	Margens aplicadas de forma sistemática pelo motor de cálculo
Custos considerados	Frequentemente apenas peça e mão de obra	Peça, frete, funcionário e, no modo avançado, material, horas técnicas, INSS e montagem
Esforço de cálculo	10 (modo padrão) a 17 (modo avançado) operações aritméticas por orçamento, refeitas a cada alteração	Preenchimento de 3 a 6 campos ou comando por voz; cálculo em menos de 1 µs
Registro e histórico	Anotações em papel ou mensagens; difícil recuperação	Histórico por usuário na nuvem, com busca, filtros e edição
Vínculo com o cliente	Informal	Cadastro de clientes (PF/PJ) vinculado a cada serviço
Acesso	Restrito ao local onde está a anotação	*Web*, PWA em celulares e aplicativo *desktop*, com os mesmos dados
Apoio ao usuário	Inexistente	Assistente que explica o app, cita os valores da tela e executa cálculos por comando
Custo de infraestrutura	Nenhum	Plano gratuito do Firebase e da Netlify; APIs de IA em faixa gratuita

Fonte: elaborado pelos autores (2026).

A aplicação avançou em todos os aspectos analisados, com destaque para a rastreabilidade dos orçamentos, inexistente no processo anterior, e para a inclusão de custos antes ignorados, o que tende a aumentar a precisão financeira dos registros. O processo manual

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.

mantém vantagem apenas na ausência de dependências, pois a L.I.L.Y. requer conexão à internet para sincronizar dados e consultar os modelos de linguagem.

3.4. Comportamento do assistente inteligente

O assistente foi avaliado quanto à sua capacidade de responder com base no contexto da tela e de converter comandos em linguagem natural em ações válidas na aplicação. A Tabela 3 reúne os indicadores obtidos no segundo e no terceiro procedimentos de validação descritos na seção 2.9.

Tabela 3. Indicadores de desempenho do assistente inteligente

Indicador	Resultado
Testes unitários da validação de ações aprovados	45 de 45 (100%)
Taxa de acerto geral das ações	36 de 40 (90%)
Taxa de acerto nos comandos operacionais	26 de 30 (86,7%)
Mensagens sem ação respondidas corretamente apenas com texto	10 de 10 (100%)
Ações indevidas executadas	0
Tempo de resposta com intervalo de 25 s: mediana (mín.–máx.)	1,23 s (0,81–6,48 s)
Tempo de resposta com intervalo de 2 s: mediana (mín.–máx.)	16,58 s (0,88–23,27 s)
Tempo de resposta das perguntas de explicação (Gemini): mediana (mín.–máx.)	6,52 s (4,27–7,46 s)

Fonte: elaborado pelos autores (2026).

A taxa de acerto de 90% foi obtida sem nenhuma ação indevida: nas quatro execuções classificadas como erro, o assistente não executou ação e solicitou valores que considerou ausentes, como o frete quando apenas o valor da peça foi informado, comportamento conservador que preserva a integridade dos dados. Os pedidos não suportados foram recusados em todas as execuções. Com intervalo de 25 s, a mediana de 1,23 s mostrou-se compatível com a interação por voz; em sequência rápida, a mediana subiu para 16,58 s, o que se atribui aos limites de requisições e de *tokens* por minuto do plano gratuito da Groq, que fazem o cliente aguardar a liberação da cota. As perguntas de explicação, que passam pelo Gemini e pelo resumo na Groq, levaram 6,52 s em mediana, tempo aceitável para respostas mais elaboradas.

Os casos de robustez também se comportaram como esperado: blocos de ação com tipo desconhecido, valores não numéricos ou vazios foram descartados, restando apenas a fala; com o motor Python desligado, a resposta veio do modo local em até 8 s; e, com a cota da API esgotada (erro 429), o usuário foi informado do limite, enquanto a calculadora continuou

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.

operando. Sem a validação do bloco de ação no motor, o texto de comando seria lido em voz alta e valores inválidos poderiam ser inseridos nos campos.

3.5. Limitações identificadas

Além da versão inicial do reconhecimento de imagens, foram identificadas as seguintes limitações: as margens de lucro são fixas no código, o que restringe a adaptação a outros negócios; o assistente depende de provedores externos e está sujeito a limites de cota das APIs gratuitas; a voz *offline* com Vosk está disponível apenas na versão *desktop*, que exige o interpretador Python instalado; e a listagem de contas depende de um índice composto no Firestore. Por fim, o tempo de elaboração dos orçamentos pelos usuários da oficina não foi cronometrado, de modo que os ganhos de eficiência foram estimados pelo esforço de cálculo eliminado e pelo tempo de processamento medido.

4. CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento da L.I.L.Y., uma aplicação *web* para automação e gestão de orçamentos em oficinas especializadas em radiadores, que integra serviços de nuvem, uma API REST e um motor de inteligência artificial baseado em modelos de linguagem.

Retomando a questão de pesquisa, os resultados demonstraram que uma aplicação *web* apoiada em serviços de nuvem e em recursos de inteligência artificial é capaz de padronizar e agilizar a elaboração de orçamentos em uma microempresa: o motor de cálculo aplicou as margens de forma sistemática e reproduzível, com 100% de concordância com os valores de referência em 10.000 casos de teste, eliminando a variação entre orçamentos de um mesmo serviço; o histórico de contas passou a ser centralizado e acessível de qualquer dispositivo; e o assistente converteu comandos em linguagem natural, por texto ou por voz, em ações na aplicação, com taxa de acerto de 90% e nenhuma ação indevida em 40 execuções. Confirmou-se, assim, no contexto do estudo de caso, a hipótese de que a parametrização de custos e a automatização dos cálculos aumentam a precisão dos orçamentos e reduzem falhas humanas. O objetivo geral foi atingido, e quatro dos cinco objetivos específicos foram integralmente concluídos, tendo o reconhecimento de peças por imagem sido entregue em versão inicial.

Em relação às soluções existentes, a L.I.L.Y. tem como vantagens o custo de infraestrutura praticamente nulo, viabilizado pelo modelo de computação em nuvem discutido por Taurion (2009), a interface simples orientada às diretrizes de usabilidade de Nielsen (2000) e a

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.

assistência inteligente contextualizada, recurso incomum em sistemas de gestão voltados a microempresas; como desvantagens, destacam-se a dependência de conectividade e de provedores externos de IA, as margens fixas no código e o menor conjunto de funcionalidades administrativas em relação a sistemas comerciais de gestão de oficinas, que, em contrapartida, têm custo de licenciamento e curva de aprendizado maiores.

Como aplicações práticas, a arquitetura modular permite adaptar a L.I.L.Y. a outros prestadores de serviços que precifiquem a partir de insumos, frete, mão de obra e horas técnicas, ajustando apenas os tipos de peça e as margens. Sob a perspectiva acadêmica, o trabalho evidenciou a viabilidade de integrar modelos de linguagem a aplicações de negócio de pequeno porte por meio de instruções de sistema, contexto da tela e ações estruturadas validadas, sem treinar modelos próprios.

Recomenda-se, para trabalhos futuros: (a) a implementação do classificador de imagens com TensorFlow.js, treinado com fotografias reais das peças, substituindo a heurística atual; (b) a parametrização das margens de lucro pela interface, com histórico de alterações; (c) a migração da persistência para um banco de dados relacional, conforme previsto na arquitetura, com relatórios financeiros consolidados; (d) a geração de orçamentos em PDF e o envio ao cliente por mensagem; (e) a incorporação do teste do motor de cálculo ao repositório, ao lado dos testes unitários já existentes para as ações do assistente, em um processo de integração contínua; e (f) a avaliação formal de usabilidade com um número maior de usuários e a medição quantitativa do tempo de elaboração de orçamentos antes e depois da adoção da ferramenta.

REFERÊNCIAS

ALPHA CEPHEI. **Vosk speech recognition toolkit**. 2026. Disponível em: https://alphacephei.com/vosk/. Acesso em: 12 set. 2026.

BRUNI, A. L.; FAMÁ, R. **Gestão de custos e formação de preços**: com aplicações na calculadora HP 12C e Excel. 6. ed. São Paulo: Atlas, 2012.

FLANAGAN, D. **JavaScript: o guia definitivo**. 6. ed. Porto Alegre: Bookman, 2013.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge: MIT Press, 2016.

GOOGLE. **Firebase documentation**. 2026a. Disponível em: https://firebase.google.com/docs. Acesso em: 12 set. 2026.

GOOGLE. **Gemini API documentation**. 2026b. Disponível em: https://ai.google.dev/gemini-api/docs. Acesso em: 12 set. 2026.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

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.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de informação gerenciais**. 11. ed. São Paulo: Pearson, 2014.

MARTINS, E. **Contabilidade de custos**. 10. ed. São Paulo: Atlas, 2010.

META. **React: the library for web and native user interfaces**. 2026. Disponível em: https://react.dev/. Acesso em: 12 set. 2026.

NIELSEN, J. **Projetando websites**. Rio de Janeiro: Elsevier, 2000.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

REZENDE, D. A.; ABREU, A. F. de. **Tecnologia da informação aplicada a sistemas de informação empresariais**: o papel estratégico da informação e dos sistemas de informação nas empresas. 9. ed. São Paulo: Atlas, 2013.

SADALAGE, P. J.; FOWLER, M. **NoSQL essencial**: um guia conciso para o mundo emergente da persistência poliglota. São Paulo: Novatec, 2013.

SEBRAE. **Transformação digital dos pequenos negócios 2025**. Brasília: Sebrae, 2025. Disponível em: https://datasebrae.com.br/transformacao-digital/. Acesso em: 13 set. 2026.

SEBRAE-PR. **Pesquisa TIC 2025**: transformação digital nos pequenos negócios. Curitiba: Sebrae/PR, 2025. Disponível em: https://sebraepr.com.br/impulsiona/pesquisa-tic-2025-transformacao-digital-nos-pequenos-negocios/. Acesso em: 13 set. 2026.

TAURI. **Tauri 2.0 documentation**. 2026. Disponível em: https://tauri.app/. Acesso em: 12 set. 2026.

TAURION, C. **Cloud computing**. Rio de Janeiro: Brasport, 2009.

TENSORFLOW. **TensorFlow.js: machine learning for JavaScript developers**. 2026. Disponível em: https://www.tensorflow.org/js. Acesso em: 12 set. 2026.

VASWANI, A. et al. Attention is all you need. In: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 31., 2017, Long Beach. Advances in Neural Information Processing Systems 30. Red Hook: Curran Associates, 2017. p. 5998-6008.

VITE. **Vite: next generation frontend tooling**. 2026. Disponível em: https://vite.dev/. Acesso em: 12 set. 2026.

W3C. **Web Content Accessibility Guidelines (WCAG) 2.1**. 2023. Disponível em: https://www.w3.org/TR/WCAG21/. Acesso em: 12 abr. 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.