

**UNIVERSIDADE DO VALE DO RIO DOS SINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

GUILHERME CLOSS FRAGA

**INTELIGÊNCIA ARTIFICIAL E VIBE CODING NO PROCESSO DE
DESENVOLVIMENTO DE SOFTWARE:**

Um estudo comparativo entre desenvolvedores com e sem o uso de ferramentas de IA

São Leopoldo
2025

GUILHERME CLOSS FRAGA

**INTELIGÊNCIA ARTIFICIAL E VIBE CODING NO PROCESSO DE
DESENVOLVIMENTO DE SOFTWARE:**

Um estudo comparativo entre desenvolvedores com e sem o uso de ferramentas de IA

Artigo apresentado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, pelo Curso de Ciência da Computação da Universidade do Vale do Rio dos Sinos (UNISINOS)

Orientador(a): Prof.^a Dra. Rosemary Francisco

São Leopoldo
2025

INTELIGÊNCIA ARTIFICIAL E VIBE CODING NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: Um estudo comparativo entre desenvolvedores com e sem o uso de ferramentas de IA

Guilherme Closs Fraga¹

Rosemary Francisco²

Resumo: Lorem ipsum.

Palavras-chave: inteligência artificial. generativa. desenvolvimento de software.

Abstract: Lorem ipsum.

Keywords: artificial intelligence. generative. software development.

1 INTRODUÇÃO

A Inteligência Artificial (IA) é considerada uma das tecnologias mais transformadoras da atualidade, impulsionando mudanças em diversos setores e despertando interesse tanto no meio acadêmico quanto no mercado. Trata-se do uso de algoritmos e modelos computacionais capazes de realizar tarefas tradicionalmente dependentes da cognição humana, como análise de dados, raciocínio lógico e tomada de decisão (CÂNDIDO, 2022; AGUIAR, 2023). Segundo relatório da McKinsey (2025), a utilização de ferramentas de IA generativa pode aumentar a produtividade de desenvolvedores em tarefas de codificação, permitindo a execução de certas atividades até duas vezes mais rápido que processos totalmente manuais. Apesar desses avanços, a adoção dessas tecnologias levanta questões sobre dependência excessiva, erosão de habilidades humanas e confiabilidade das soluções geradas.

No contexto do desenvolvimento de *software*, a presença da IA tem se tornado particularmente evidente. Ferramentas que combinam aprendizado de máquina e processamento de linguagem natural auxiliam desenvolvedores em diversas etapas do ciclo de vida do *software*, desde a escrita de código até a detecção de falhas e a otimização de desempenho (COUTINHO et al., 2024). Um exemplo é o *GitHub Copilot*, lançado em 2021 pela *GitHub* em parceria com a *OpenAI*, que utiliza modelos de IA Generativa (também chamadas de IA Gen) para sugerir trechos de código em tempo real (GITHUB, 2021). Da mesma forma, agentes de codificação baseados em IA, alinhados ao conceito de *vibe coding*, permitem que o desenvolvedor interaja de forma mais conversacional com a ferramenta, gerando código de maneira assistida, mas ainda mantendo controle sobre decisões críticas (DELANEY, 2025). Essas práticas têm sido discutidas como um avanço na programação assistida, trazendo benefícios de produtividade e

¹Graduando em Ciência da Computação pela Unisinos. Email: guifraga8@edu.unisinos.br

²Doutora em Administração. Email: rosemaryf@unisinos.br

colaboração entre humano e máquina.

Esses recursos, ao mesmo tempo em que oferecem praticidade e aumentam a produtividade, também levantam debates relevantes. Há questionamentos sobre até que ponto a utilização de IA generativa pode impactar o desenvolvimento de habilidades de raciocínio lógico e resolução de problemas, uma vez que o programador pode se tornar excessivamente dependente da ferramenta (VERASTEGUI; RUIZ; VILLANUEVA, 2025). Além disso, o código gerado pode apresentar erros lógicos, violações de boas práticas e vulnerabilidades de segurança, exigindo revisão crítica por parte do profissional (DEVOPS, 2024). Estudos experimentais indicam resultados variados: enquanto tarefas simples podem ser concluídas significativamente mais rápido com IA, contextos complexos podem exigir maior tempo de revisão e adaptação do código (PENG et al., 2023; METR, 2025).

Por outro lado, pesquisas recentes indicam que a adoção de IA no desenvolvimento de *software* pode representar ganhos significativos em termos de produtividade e redução do tempo necessário para a entrega de soluções. Um estudo piloto com profissionais de *software* demonstrou percepções majoritariamente positivas quanto ao impacto desses recursos na produtividade individual (COUTINHO et al., 2024). Essa realidade torna essencial a investigação de como essas ferramentas impactam o trabalho de desenvolvedores em contextos práticos. Assim, mais do que um recurso de conveniência, a IA se apresenta como um instrumento que pode redefinir processos e metodologias de engenharia de *software*.

Diante desse cenário, torna-se essencial investigar como a IA Generativa influencia o desempenho de desenvolvedores. Este trabalho propõe um estudo comparativo entre dois grupos de participantes: um com acesso a ferramentas de IA Generativa e outro sem acesso a esses recursos, aplicando um desafio técnico de programação. O objetivo é avaliar variáveis como eficiência, tempo de resolução e capacidade para solucionar os problemas propostos, fornecendo uma investigação sobre vantagens e limitações do uso dessas tecnologias.

Assim, o objetivo geral desta pesquisa é analisar o uso de ferramentas de IA Generativa no Desenvolvimento de *Software* por meio de um desafio técnico comparativo entre desenvolvedores com e sem acesso a essas ferramentas. A questão de pesquisa que orienta este estudo é: de que maneira o uso de ferramentas de IA Generativa contribui ou afeta o desempenho de desenvolvedores em atividades práticas de programação? Destacam-se também os objetivos específicos necessários para atingir a conclusão deste trabalho, sendo eles:

- a) desenvolver uma plataforma *web* simples para disponibilização do desafio técnico e coleta de dados de desempenho dos participantes;
- b) aplicar o desafio técnico a dois grupos de desenvolvedores, sendo um autorizado a utilizar ferramentas de IA e outro restrito ao uso dessas tecnologias;
- c) comparar os resultados obtidos pelos grupos em relação ao tempo de conclusão e resolução do desafio proposto;
- d) analisar as vantagens, limitações e possíveis implicações do uso de ferramentas de IA no

processo de desenvolvimento de *software*.

As seções seguintes deste documento estão dispostas da seguinte forma: a segunda seção contempla a fundamentação teórica do trabalho, que discorre sobre o desenvolvimento de *software* e IA Generativa aplicada ao desenvolvimento de *software* propriamente. A terceira seção apresenta os trabalhos relacionados a esta pesquisa e uma breve comparação entre os mesmos. Na quarta seção apresenta-se a metodologia do trabalho, descrevendo como foi conduzida todas as etapas e elaborações da pesquisa. Na quinta seção está a proposta de modelo do trabalho, apresentando brevemente a arquitetura da plataforma, tecnologias utilizadas e seu fluxo de uso do lado dos participantes. Na sexta seção apresenta-se os resultados e discussões após coleta de dados por meio do experimento aplicado como um todo.

Por fim, ao fechamento do trabalho na última seção, encontra-se as conclusões e considerações finais levantadas com o experimento, assim como apontamentos para trabalhos futuros. Espera-se que os resultados forneçam evidências consistentes para compreender o papel da IA como aliada no processo de desenvolvimento de *software*, oferecendo resultados para pesquisadores, profissionais e instituições interessadas em compreender e orientar o uso dessas ferramentas emergentes.

2 FUNDAMENTAÇÃO TEÓRICA

Com base na temática que o presente trabalho visa abordar, se faz necessário discorrer sobre dois tópicos fundamentais para a construção e consolidação do mesmo, sendo eles: Desenvolvimento de *Software* e IA Generativa aplicada ao desenvolvimento de *software*. Nas seções seguintes, os conceitos são explorados e detalhados a fim de estabelecer os fundamentos teóricos necessários para compor este trabalho.

2.1 Desenvolvimento de *Software*

Um *software* refere-se a todos os componentes não físicos de um computador, redes de computadores ou dispositivos móveis. Em linhas gerais, refere-se aos programas e aplicações, como o próprio sistema operacional, que fazem aquele aparelho (ou ferramenta) operar de acordo com o que o usuário necessita. Isto é, *software* se trata de uma coleção de instruções, dados ou programas utilizados para operar computadores e executar determinadas tarefas (COUTINHO; BEZERRA, 2021).

Em outras palavras, pode-se dizer que é um termo genérico que se refere à aplicações, *scripts* e programas executáveis em um determinado dispositivo. Assim, o diferencia de *hardware*, que descreve os aspectos físicos de um aparelho. Sendo assim, pode-se estabelecer o *software* como um componente variável de um computador, enquanto o *hardware* é constante (SAKURAI; ZUCHI, 2018).

De acordo com Maynard (2015), existem basicamente dois tipos de *software*, sendo eles:

software de aplicativo e *software* de sistema. Um aplicativo seria aquele que atende uma necessidade específica ou executa determinadas tarefas. E, segundo Schwab (2019), *software* de sistema seria a parte essencial para que o *hardware* de um computador opere, servindo como a plataforma para a execução de aplicativos. Ademais, pode-se mencionar outros tipos de *software*, como os de programação, que são ferramentas necessárias para desenvolvedores, *middlewares*, responsáveis por atuarem entre o *software* do sistema e aplicativos e até mesmo *drivers*, que são responsáveis por operarem dispositivos e periféricos de computadores.

Por fim, a sincronia entre diferentes tipos de *software* e *hardware*, é peça fundamental para o funcionamento eficiente de sistemas computacionais. Enquanto um aplicativo permite que seus usuários realizem determinadas tarefas, um sistema assegura que o *hardware* opere corretamente, dando suporte para diferentes aplicações. Com a constante crescente de complexidade em sistemas modernos, a presença de um *middleware* eficaz se torna indispensável para assegurar que diferentes *softwares* funcionem em harmonia. Da mesma forma que, *drivers* são indispensáveis para comunicação entre o sistema e periféricos, o que amplia as funcionalidades de um computador, resultando na melhoria da experiência do usuário.

Os *softwares*, diferentemente do hardware, não são tangíveis e representam os componentes lógicos responsáveis pelo funcionamento de um computador ou dispositivo móvel. De acordo com Cândido (2022), o sistema operacional constitui o software fundamental, sendo exemplos mais conhecidos o *Windows*, *macOS* e *Linux*, que variam em suas interfaces de usuário. Além dele, outros programas complementam as funções básicas, como o *Microsoft Office* e o *MediaPlayer* (NETO, 2019), além de *softwares* utilitários, como antivírus, ferramentas de *backup* e manutenção de disco, que garantem desempenho e segurança. O avanço tecnológico também possibilitou o surgimento da virtualização, permitindo a execução de diferentes sistemas operacionais em uma única máquina física e ampliando a flexibilidade no uso dos recursos.

O *software* pode ainda estar integrado diretamente ao *hardware*, como em painéis eletrônicos de automóveis ou leitores de *Blu-Ray*, caracterizando os chamados “embutidos” (COUTINHO; BEZERRA, 2021). Historicamente, *software* e *hardware* eram concebidos como uma única unidade, sendo o termo “*software*” empregado pela primeira vez por John W. Tukey em 1958 (MAYNARD, 2015). Somente vinte anos depois, o governo dos Estados Unidos determinou que a IBM passasse a contabilizar *hardware* e *software* separadamente, o que abriu espaço para empresas especializadas (SAKURAI; ZUCHI, 2018). Essa dissociação entre *hardware* e *software* impulsionou a inovação, permitindo atualizações independentes, acelerando ciclos de desenvolvimento e fomentando novos modelos de negócio baseados em *software*, que hoje ocupam posição central na economia digital.

Schwab (2019) destaca que o trabalho do programador consiste, de forma essencial, em traduzir requisitos e algoritmos para uma linguagem de programação que o computador consiga interpretar e executar. A partir desse processo, é possível desenvolver diferentes tipos de *software*, como aplicativos, jogos e sistemas de controle para robôs. Um exemplo histórico ligado à ideia de programação foi a invenção de um *tear* “programável” pelo francês Joseph-Marie

Jacquard, que representou um marco na Revolução Industrial. O equipamento não possuía processador, sendo controlado por meio de cartões perfurados. Conforme explica Maynard (2015), esses cartões não utilizavam valores binários como 0 e 1, mas funcionavam através da presença ou ausência de perfurações: um furo indicava uma ação de movimento, enquanto a ausência significava outra. A criação de Jacquard é considerada uma das precursoras da automação moderna.

Maynard (2015) aponta que, em 1843, Augusta Ada King-Noel, mais conhecida como Ada Lovelace, desempenhou um papel marcante ao colaborar com Charles Babbage no projeto do *Analytical Engine*, uma calculadora mecânica de uso geral que acabou não sendo concluída. Durante esse processo, Ada elaborou diversas anotações e descreveu um método para calcular os números de Bernoulli por meio da máquina, o que, segundo Coutinho e Bezerra (2021), é considerado o primeiro programa de computador da história. Posteriormente, em 1941, Konrad Ernst Otto Zuse desenvolveu o Z3, reconhecido como o primeiro sistema de computação totalmente automatizado e programável do mundo.

O Z3 foi construído a partir dos conceitos idealizados por Babbage e tinha como diferencial a capacidade de realizar operações, como a adição, em menos de um quarto de segundo. O propósito central de sua criação estava ligado à busca por maior eficiência nos cálculos. Nos primeiros anos, os *softwares* eram desenvolvidos para máquinas específicas e eram comercializados em conjunto com o *hardware*. Já na década de 1980, passaram a ser distribuídos em mídias físicas, como disquetes, e mais tarde em CDs e DVDs. Atualmente, grande parte dos *softwares* é obtida por meio da *Internet*, seja diretamente nos sites dos fabricantes ou através de plataformas de distribuição de aplicativos.

Os modelos de ciclo de vida de *software* descrevem as etapas que um sistema percorre desde sua concepção até sua desativação, organizando a sequência de atividades necessárias para o desenvolvimento. O objetivo central desses modelos é garantir qualidade, reduzir custos e otimizar o tempo de entrega. Entre as fases mais comuns estão:

- Análise de Requisitos: onde são levantadas e documentadas as necessidades do sistema;
- Design: define a arquitetura, fluxos de trabalho e tecnologias a serem utilizadas;
- Implementação: fase em que o código é desenvolvido com base no planejamento estabelecido;
- Testes: que envolvem diferentes tipos de verificação para assegurar a confiabilidade do *software*;
- Manutenção: responsável por corrigir falhas e implementar melhorias após a entrega do sistema.

De acordo com Cândido (2022), observa-se nos últimos anos uma tendência de desprofissionalização no aprendizado de programação, já que muitas pessoas têm adquirido essas habilidades de forma autodidata, recorrendo a tutoriais *online* em vez de cursos formais. Além

disso, o *networking* global, aliado às diversas plataformas digitais, tem favorecido a formação de grandes comunidades que reúnem tanto entusiastas quanto programadores experientes, tornando cada vez menos nítida a distinção entre esses grupos.

Ao mesmo tempo, percebe-se um avanço significativo nos padrões e na qualidade do desenvolvimento de *software*. Os sistemas estão cada vez mais sofisticados e complexos, o que aumenta a demanda por soluções digitais. Esse contexto reflete uma sociedade em constante transformação, onde as necessidades mudam rapidamente, gerando uma procura crescente por Desenvolvedores *Full Stack* (um profissional de tecnologia que possui a capacidade de trabalhar em todas as camadas de um projeto de *software*) em detrimento de profissionais especializados apenas em uma área.

A complexidade crescente dos sistemas requer profissionais capazes de atuar em todas as camadas do desenvolvimento, abrangendo desde o *Frontend* (interface do usuário) até o *Backend* (servidor, banco de dados e lógica de negócios de uma aplicação). Nesse cenário, os Desenvolvedores *Full Stack* ganham destaque por sua flexibilidade e capacidade de adaptação às mudanças do mercado e às demandas dos usuários. Sua habilidade em compreender e integrar diferentes componentes de um sistema favorece um desenvolvimento mais unificado e eficiente, além de melhorar a comunicação e colaboração dentro das equipes. Assim, em um ambiente em que a agilidade e a adaptabilidade são fatores essenciais, esses profissionais se consolidam como recursos estratégicos para empresas que buscam manter a competitividade.

O desenvolvimento de *software* pode ser entendido como um conjunto de atividades voltadas ao projeto, criação, implementação e manutenção de programas que orientam o funcionamento de computadores, sendo independente do *hardware* e responsável por torná-los programáveis (SCHWAB, 2019).

Conforme aponta Cândido (2022), o desenvolvimento de *software* deve ser integrado ao desenvolvimento de produtos mecânicos e elétricos, o que faz com que o papel dos desenvolvedores de *software* seja menos delimitado em comparação ao dos engenheiros. Esses profissionais podem focar em áreas específicas do projeto, como a codificação, enquanto o ciclo de vida do *software* envolve a transformação de requisitos em funcionalidades, o trabalho em equipes multifuncionais, a gestão de processos e equipes, além de atividades de teste e manutenção.

De acordo com Schwab (2019), a atuação dos desenvolvedores de *software* vai além das equipes de programação, envolvendo também cientistas e fabricantes de dispositivos e *hardware*, que contribuem para a criação de códigos, mesmo não sendo os principais responsáveis pelo desenvolvimento. Essa participação não se limita às indústrias tradicionais de tecnologia da informação, abrangendo também empresas que não se dedicam exclusivamente à produção de *software* ou semicondutores.

A ampliação do papel dos desenvolvedores de *software* evidencia a crescente integração da tecnologia da informação com diversas áreas, como ciência, medicina, engenharia e artes. Atualmente, o desenvolvimento de *software* é fundamental para múltiplos setores, desde a indústria automotiva até o entretenimento, impulsionando a inovação e a transformação digital. Na in-

dústria automotiva, esses profissionais são essenciais para o desenvolvimento de sistemas de condução autônoma e tecnologias avançadas de segurança. Já no cinema, os efeitos visuais e a animação digital dependem fortemente do *software* para criar mundos e personagens realistas. Dessa forma, os desenvolvedores de *software* exercem um papel cada vez mais diversificado e influente, moldando o futuro em diferentes áreas do conhecimento e da atividade humana.

2.1.1 Modelos de Desenvolvimento de *Software*

O desenvolvimento de *software* pode ser conduzido por diferentes abordagens, conhecidas como modelos de processo de *software*. Esses modelos têm como objetivo estruturar e organizar as atividades necessárias para transformar requisitos em um produto final de qualidade. Existem diversos modelos conhecidos, entretanto, os mais populares e utilizados acabam sendo o Modelo Cascata e os Modelos Ágeis.

O Modelo Cascata (ou *Waterfall*) foi um dos primeiros modelos formais de processo de *software*. Proposto por Winston Royce em 1970, consiste em uma sequência linear de fases: levantamento de requisitos, análise, projeto, implementação, testes, implantação e manutenção. Cada fase deve ser concluída antes do início da próxima, funcionando como uma “cascata” de atividades (SOMMERVILLE, 2011). Entre suas vantagens podemos citar:

- Estrutura simples e de fácil entendimento;
- Definição clara das fases e entregas;
- Útil em projetos com requisitos bem definidos e estáveis.

Por outro lado, há também desvantagens que valem ser destacadas deste modelo, sendo elas:

- Pouca flexibilidade diante de mudanças nos requisitos;
- Risco de identificação tardia de falhas, geralmente apenas na fase de testes;
- Dificuldade em lidar com projetos complexos ou inovadores.

Os Modelos Ágeis surgiram como uma alternativa aos modelos tradicionais, sendo formalizados em 2001 com o Manifesto Ágil. Esses modelos priorizam colaboração com o cliente, entregas rápidas e contínuas, adaptação às mudanças e interações entre pessoas mais do que processos rígidos (BECK et al., 2001).

Entre os métodos ágeis mais utilizados, principalmente nos dias atuais, estão o *Scrum*, o *Kanban* e o *Extreme Programming* (conhecido também como *XP*). Todos eles seguem o princípio de dividir o projeto em pequenos ciclos (iterações), resultando em entregas frequentes e incrementais do produto. Como as principais vantagens, podemos citar:

- Alta flexibilidade para mudanças de requisitos;
- Entregas rápidas que geram valor contínuo ao cliente;
- Melhora na comunicação e na colaboração entre equipe e *stakeholders*.

E, das suas principais desvantagens:

- Pode gerar falta de documentação formal, dependendo da equipe;
- Requer alto envolvimento do cliente no processo;
- Menos indicado para projetos de missão crítica que demandam forte rastreabilidade.

Sendo assim, evidencia-se que cada modelo de desenvolvimento possui suas características, vantagens e limitações. A escolha do modelo mais adequado depende do contexto do projeto, da complexidade do sistema, do nível de risco, do grau de inovação e da estabilidade dos requisitos. Modelos tradicionais como Cascata são mais estruturados, os modelos Ágeis oferecem maior flexibilidade e adaptabilidade, características cada vez mais valorizadas no mercado atual. Na Tabela 1, temos um comparativo resumido dos principais pontos dos modelos citados.

Tabela 1 – Comparativo entre Modelo Cascata e Modelos Ágeis

Modelo	Resumo	Vantagem	Desvantagem	Indicado para
Cascata	Linear e sequencial	Simples e claro	Pouca flexibilidade	Projetos pequenos e estáveis
Ágeis	Iterativo e adaptável	Entregas rápidas	Pouca documentação	Projetos dinâmicos e mutáveis

Fonte: Elaborado pelo autor

2.2 IA Generativa (aplicada ao desenvolvimento de software)

A Inteligência Artificial (IA) consolidou-se como uma das áreas mais relevantes da Ciência da Computação, devido à sua capacidade de simular processos cognitivos humanos como aprendizagem, raciocínio e tomada de decisão por meio de algoritmos e modelos matemáticos (CÂNDIDO, 2022; AGUIAR, 2023). Nos últimos anos, a evolução tecnológica permitiu que esses sistemas ultrapassassem o campo do reconhecimento de padrões e da automação, alcançando um estágio em que são capazes de gerar novos conteúdos, fenômeno que deu origem ao conceito de Inteligência Artificial Generativa. Segundo Uniphore (2024), a IA Generativa refere-se ao ramo da IA que cria resultados originais sejam eles textos, imagens, sons ou códigos, a partir de grandes volumes de dados já existentes, distinguindo-se de modelos meramente classificatórios ou preditivos. Conforme aponta TechTarget (2024), esse processo ocorre por meio da análise de padrões e estruturas aprendidas, sendo ativado a partir de instruções em linguagem natural, conhecidas como prompts.

No campo do desenvolvimento de software, a IA Generativa tem desempenhado papel crescente ao possibilitar a geração automática de trechos de código, documentação, testes e até refatoração de programas. Idrisov e Schlippe (2024) compararam códigos produzidos por ferramentas de IA e por programadores humanos, identificando que, embora as soluções geradas

automaticamente sejam satisfatórias em tarefas de baixa complexidade, ainda apresentam limitações em contextos que exigem maior profundidade técnica ou conhecimento de domínio. Em estudo semelhante, Yu (2025) observou que assistentes de codificação generativa, como Codeium e ferramentas similares, aumentam a produtividade em atividades repetitivas, mas enfrentam dificuldades quando aplicados a tarefas que demandam compreensão arquitetural ou integração com sistemas legados.

Outro aspecto recorrente na literatura é a avaliação da qualidade do código gerado. Yetiştiren et al. (2023) realizaram um comparativo entre GitHub Copilot, Amazon CodeWhisperer e ChatGPT, utilizando o benchmark HumanEval. Os resultados apontaram variações significativas em critérios como correção, confiabilidade e manutenibilidade, indicando que tais ferramentas funcionam melhor como apoio ao programador do que como substitutos plenos. A segurança do software também se apresenta como desafio relevante: de acordo com Alwageed e Khan (2025), embora a IA Generativa possa contribuir para práticas de codificação segura, existe o risco de introdução de vulnerabilidades e de não conformidade com padrões regulatórios.

Por fim, é importante destacar as discussões éticas e sociais associadas ao uso da IA Generativa. Afreen, Mohaghegh e Doborjeh (2025) ressaltam que problemas como a contextualização insuficiente do código, a geração de erros sutis e as preocupações relacionadas a direitos autorais e privacidade ainda carecem de soluções adequadas. Nesse sentido, observa-se que a IA Generativa aplicada ao desenvolvimento de software apresenta grande potencial de aumento de produtividade e inovação, mas exige acompanhamento crítico, validações constantes e reflexões acerca de suas limitações e implicações sociais.

3 TRABALHOS RELACIONADOS

Para fins de melhor localizar os trabalhos com temáticas relacionadas para o presente estudo, foi utilizado a plataforma de busca especializada em literatura acadêmica e científica Google Scholar. Os procedimentos de pesquisa e seleção consistiram em utilizar palavras chaves em português como “inteligência artificial generativa”, “desenvolvimento de software”, “produtividade”, “suporte”, “otimização” e “performance”, variando também os termos em inglês. Por fim, para selecionar os trabalhos a seguir, foi considerado o contexto do estudo de cada autor, conceitos abordados, tecnologias mencionadas e/ou utilizadas e problemáticas ressaltadas, que melhor correlacionassem com o presente estudo.

3.1 Trabalhos Acadêmicos

Dado o contexto do aprimoramento e evolução de Inteligências Artificiais Generativas nos anos recentes, tal como suas aplicabilidades para otimização de tarefas voltadas para a área de desenvolvimento de software, diversos estudos têm sido realizados com o intuito de explorar

ainda mais os conceitos técnicos e práticos. A seguir, são apresentados estudos e pesquisas relacionados ao tema, estabelecendo um paralelo com a temática abordada para este trabalho.

Gomes (2023) apresenta uma análise comparativa de diferentes Inteligências Artificiais Generativas (IAs Generativas) aplicadas no desenvolvimento de software, mas particularmente focando em ferramentas auxiliares como *GitHub Copilot*, *Codeium*, *Tabnine* e *CodeGeeX*. Para fins práticos, seu trabalho focou em experimentos utilizando *GitHub Copilot* e *Codeium* para mensurar a eficiência das ferramentas.

Como resultado, o Gomes (2023) aponta que ambas as tecnologias se mostram extremamente úteis no processo de otimização do desenvolvimento, produtividade e qualidade de códigos, com a escolha entre elas dependendo das necessidades particulares do projeto e das preferências do profissional. Entretanto, o destaque ficou para a ferramenta *GitHub Copilot*, que apresentou uma leve vantagem, com uma capacidade superior de compreensão para construção de uma lógica de desenvolvimento mais coesa.

Em Silva et al. (2025), é realizado o teste prático de viabilidade de desenvolvimento de uma aplicação *web* para gerenciamento de sócios torcedores de um clube de futebol, utilizando IA Generativa como apoio na construção da aplicação. Ainda que, neste estudo os autores desenvolveram uma solução simples, utilizando tecnologias como *JavaScript*³, *PHP*, *HTML5* e entre outras, o trabalho demonstrou que utilizar de Inteligência Artificial otimiza o processo de desenvolvimento, desde a geração de código, identificação de falhas e sugestões de melhores práticas para a construção de soluções.

De acordo com os autores e os resultados obtidos (SILVA et al., 2025), a adoção de IA Generativa no processo de desenvolvimento contribui tanto para acelerar a produção, quanto para a qualidade e otimização da aplicação. Para o estudo em questão, os autores contaram com o suporte de modelos de IA Generativa como *GPT-3.5* e *Claude 1*.

Em contrapartida, no trabalho de Ferreira (2023), o autor aborda sobre a influência das IAs Generativas na área de desenvolvimento de software, com base em um estudo qualitativo, combinando cenários exploratórios e descritivos. Aqui, o autor menciona o uso de ferramentas como o *ChatGPT*, *Bard*, *GitHub Copilot* para tratar sobre a temática de modelos baseados no conceito de Processamento de Linguagem Natural (PLN).

Ao longo de seu estudo, Ferreira (2023) aborda questões como o impacto da Inteligência Artificial na programação, trazendo dados e referências, para dissertar sobre tópicos como os desafios e benefícios do uso de ferramentas de IA. Sobre os desafios, o autor menciona a questão da dependência do uso de IA, limitando assim o desenvolvimento de habilidades próprias por parte do programador e a substituição do desenvolvedor por IA, assunto fortemente debatido desde a evolução das IAs Generativas nos anos recentes.

Santos (2024) investigou a influência das ferramentas de IA Generativas, citando *ChatGPT* e *GitHub Copilot*, com foco no processo de ensino e aprendizagem das instituições de ensino superior na área de desenvolvimento de *software*. O objetivo deste estudo consistiu em identificar

³ *JavaScript*: <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>>

oportunidades e desafios no uso destas tecnologias. Para isso, o autor realizou um comparativo com grupos de alunos iniciantes, estagiários no setor de tecnologia, alocando-os em um projeto simulado.

Para tanto o autor (SANTOS, 2024) seguiu um delineamento *crossover 2x2 AB/BA*, isto é, alternando os grupos entre o uso e não uso de IA. Após a coleta de dados, foi mensurado o conhecimento técnico dos grupos, tal como qualidade dos artefatos gerados, avaliação do produto desenvolvido e provas simuladas para testar o conhecimento adquirido. Aqui, o autor ressalta que os grupos que fizeram o uso de IA obtiveram melhores avaliações nos artefatos desenvolvidos, desde a documentação ao desenvolvimento de *software* propriamente dito. Entretanto, a nível de conhecimento, não refletiu em avaliações melhores nas provas simuladas (notas de avaliações, propriamente).

Costa (2024) explorou como o uso de IA Generativa pode otimizar e aprimorar o processo de desenvolvimento de *software* no contexto empresarial. O autor conduziu uma extensa revisão bibliográfica com o intuito de identificar as ferramentas e metodologias mais atuais e eficazes no mercado. A partir de então, selecionou e implementou *frameworks* como *Spring Boot* para a construção de uma aplicação escalável em *Java*, utilizando o *ChatGPT* como ferramenta de suporte durante o processo.

Ademais, o autor (COSTA, 2024) fez o uso de ferramentas de teste e documentação de *APIs*, tais como *Postman* e até mesmo *JUnit*, como *framework* para validação de testes unitários. Em seu trabalho, o autor utilizou a IA Generativa através de *prompts* para criação de todos os requisitos, códigos e validações do seu projeto. Como resultado, ele evidenciou uma melhoria na eficiência tanto do desenvolvimento quanto da qualidade da aplicação entregue. Entretanto, ressaltou a importância de que, o conhecimento prévio para uso de IAs Generativas é fundamental, para que se torne possível o uso da mesma como ferramenta de apoio no processo de desenvolvimento de *software*.

Por fim, com base nos estudos analisados e considerando o crescimento acelerado das ferramentas de Inteligência Artificial Generativa na atualidade, evidencia-se que o profissional, independentemente da área de atuação, pode, e deve, tirar proveito dessas tecnologias para se manter em constante aprendizado e adaptado às novas realidades do mercado. Ou seja, não é necessário dominar todos os princípios técnicos por trás dessas ferramentas, mas sim saber utilizá-las estratégicamente, visando otimização, desempenho e melhores resultados.

Dessa forma, a proposta deste estudo é analisar, com base em dados e indicadores, o quanto o uso da Inteligência Artificial Generativa pode contribuir, junto aos profissionais da área de desenvolvimento de *software*, para a entrega de soluções mais ágeis. Para delimitar o foco principal do trabalho, o segmento analisado será o desenvolvimento de software, considerando o papel do desenvolvedor no cenário atual do mercado de trabalho.

Com base no levantamento de todas as informações dos trabalhos relacionados ao tema, é possível agrupar as principais características de cada um deles para fins de comparação. Dada as informações levantadas, separa-se as características em quatro parâmetros, sendo eles:

- **Tema principal:** define qual foi o principal tema abordado ao longo do estudo;
- **Metodologia aplicada:** define o modelo utilizado para o estudo em questão;
- **Tecnologias/Inteligências Artificiais utilizadas:** se aplicável, define as ferramentas e/ou tecnologias utilizadas para fins de estudo, tal como o modelo de Inteligência Artificial utilizado ou analisado;
- **Objetivo:** define o objetivo do modelo, ou seja, qual problemática o modelo buscou esclarecer ao seu decorrer.

A partir do resultado do levantamento de trabalhos relacionados, foi possível elaborar a Tabela 2, onde mostra o conteúdo dos critérios apontados em cada estudo, por ordem de apresentação.

Tabela 2 – Comparativo entre os trabalhos relacionados

Autores	Tema	Metodologia	Tecnologias/IA	Objetivo
Gomes (2023)	Comparação de IAs no desenvolvimento de <i>software</i>	Testes práticos comparativos	<i>GitHub Copilot, Codeium, Tabnine, CodeGeeX</i>	Avaliar eficiência de ferramentas de IA no desenvolvimento
Silva et al. (2025)	Aplicação de IA na criação de sistemas web	Estudo de caso com experimento	<i>JavaScript, PHP, HTML5, GPT-3.5, Claude 1</i>	Verificar ganho de produtividade com uso de IA
Ferreira (2023)	Impacto da IA na programação	Estudo qualitativo exploratório	<i>ChatGPT, Bard, GitHub Copilot</i>	Analizar benefícios e riscos do uso da IA
Santos (2024)	IA no ensino de tecnologia	Experimento com grupos (<i>crossover</i>)	<i>ChatGPT, GitHub Copilot</i>	Medir impacto da IA na aprendizagem e desempenho técnico
Costa (2024)	Uso de IA Generativa no desenvolvimento de <i>software</i> empresarial	Revisão bibliográfica com aplicação prática	<i>Spring Boot, Java, ChatGPT, Postman, JUnit</i>	Otimizar e melhorar eficiência e qualidade no desenvolvimento de aplicações

Fonte: Elaborado pelo autor

4 MATERIAIS E MÉTODOS

Este trabalho tem como principal objetivo investigar a utilização de IA Generativa no processo de desenvolvimento de *software*. A partir do conhecimento consolidado com base na fundamentação teórica, o presente estudo consiste em um breve experimento, dividido em dois grupos participantes do mesmo.

Sendo assim, para atingir os objetivos, a pesquisa percorreu cinco fases. A primeira consistiu em analisar trabalhos relacionados, almejando identificar tópicos existentes sobre o assunto

ou que explorassem conceitos e práticas no que dizem respeito ao desenvolvimento de *software* com auxílio de IA. Com base nesta análise, foi possível traçar um paralelo entre os estudos relacionados e o modelo proposto.

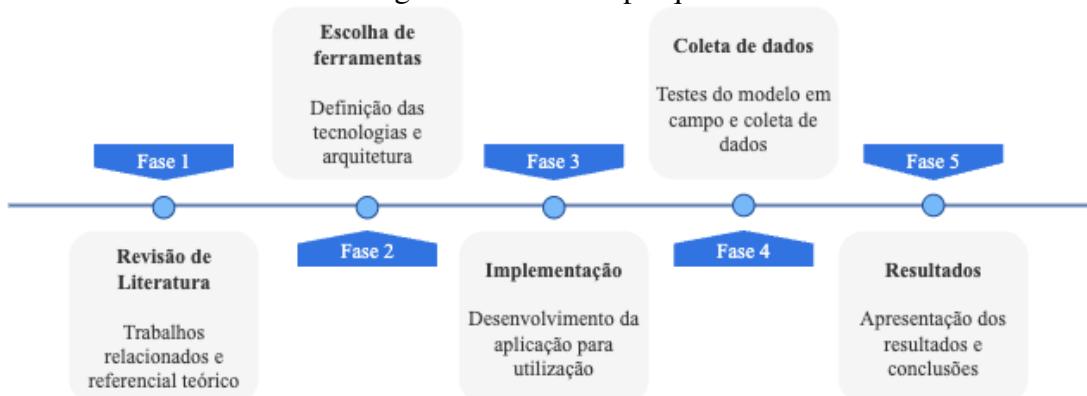
Na segunda fase, definiu-se as tecnologias, bem como a arquitetura da aplicação *web* proposta para o desenvolvimento do trabalho. Com isso, estabelece-se o alicerce para o desenvolvimento da solução e para o cumprimento dos objetivos do trabalho.

A terceira fase consistiu no desenvolvimento da aplicação *web* que foi utilizada como ambiente de execução do desafio técnico, assim como a elaboração do desafio disponibilizado dentro da plataforma. O objetivo central da ferramenta foi de oferecer, de forma prática e organizada, o enunciado do desafio, permitindo que os participantes iniciassem a atividade e, ao final, submetessem suas soluções. Além disso, a aplicação foi responsável por registrar automaticamente o tempo total despendido por cada participante, desde o momento em que iniciou a tarefa até a finalização do envio, viabilizando a coleta de dados confiáveis para a análise posterior.

A quarta fase compreendeu a aplicação prática do modelo e a coleta de dados. Para tanto, foram selecionados participantes por conveniência, ou seja, profissionais de desenvolvimento de *software* previamente conhecidos pelo pesquisador e que se dispuseram a colaborar com o estudo. Após a seleção, os desenvolvedores foram distribuídos de forma aleatória em dois grupos: um com autorização para utilizar ferramentas de IA e outro restrito ao desenvolvimento sem esses recursos. Essa etapa tem como propósito garantir condições controladas para comparar os desempenhos e mensurar a utilização da IA nos resultados obtidos.

Após a coleta dos resultados, a quinta e última fase consistiu na apresentação e validação dos dados, bem como na elaboração das considerações finais do estudo, de forma a contribuir para a área de pesquisa em questão. A Figura 1 ilustra as fases da pesquisa, contendo uma breve descrição de cada uma.

Figura 1 – Fases da pesquisa



Fonte: Elaborado pelo autor

4.1 Proposta de Solução

Conforme mencionado anteriormente, observa-se que o impacto da Inteligência Artificial no desenvolvimento de *software* ainda é um tema em debate, sendo comum que as análises existentes se restrinjam a relatos de uso ou percepções subjetivas por parte dos desenvolvedores. Em muitos casos, a avaliação da influência da IA sobre fatores como produtividade e tempo de execução de tarefas ocorre de forma fragmentada, sem um método padronizado que permita mensurar de maneira objetiva os efeitos reais de tais ferramentas.

Dentro desse cenário, a proposta do estudo surge com a elaboração do um desafio técnico, disponibilizado através da plataforma *web* desenvolvida especificamente para este fim. A ferramenta funcionou como um ambiente de avaliação controlado, no qual os participantes foram previamente organizados em dois grupos: um grupo com autorização para utilizar ferramentas de IA durante a resolução do desafio e outro grupo para realizar a atividade sem recorrer a tais recursos.

A proposta buscou, assim, criar condições equivalentes para ambos os grupos, garantindo que todos recebessem o mesmo enunciado do desafio, apresentado diretamente na plataforma. Para cada participante foi disponibilizado um *link* único, correspondente ao grupo ao qual pertencia. A partir desse acesso, o desenvolvedor pôde visualizar as instruções, iniciar a tarefa e, ao concluir, submeter os arquivos com sua solução.

A plataforma também atuou como mecanismo de registro, armazenando automaticamente o tempo total gasto por cada participante, desde o momento em que deu início ao desafio até a finalização da entrega. Dessa forma, além da coleta da solução proposta, foi possível analisar métricas objetivas como tempo de conclusão, resolução dos problemas propostos (*bugs*) e até mesmo, eventuais diferenças de abordagem entre os grupos.

5 MODELO PROPOSTO

Esta seção tem como objetivo descrever brevemente a ferramenta desenvolvida e utilizada para a execução deste trabalho, incluindo sua arquitetura, funcionalidades e tecnologias utilizadas. O modelo proposto consiste em uma aplicação *web* que centraliza as informações do desafio técnico, disponibilizando-o de forma prática e organizada para os desenvolvedores participantes do estudo.

A aplicação coleta o nome e o cargo atual do participante no mercado de trabalho. O nome é utilizado exclusivamente para exibir uma mensagem personalizada de boas-vindas. Funciona como um identificador simples durante a sessão, mantendo o usuário ativo no desafio, sem necessidade de cadastro, *tokens* ou *logins* complexos. Dessa forma, os nomes dos participantes não são utilizados na apresentação dos resultados do estudo, garantindo anonimato e simplicidade na utilização da ferramenta.

5.1 Visão Geral

A plataforma desenvolvida tem como objetivo oferecer um ambiente controlado, acessível e intuitivo para a realização de um desafio técnico. Sua concepção visa permitir a comparação entre dois grupos de participantes, um autorizado a utilizar ferramentas de IA Generativa e outro sem esse recurso, garantindo condições iguais de execução e coleta precisa de dados para análise posterior.

A aplicação dispõe de duas telas de cadastro semelhantes, diferenciadas apenas pelo tipo de participante. Uma é destinada aos desenvolvedores do grupo sem IA, e a outra aos do grupo com IA. Essa distinção é registrada internamente no banco de dados, identificando a qual grupo o participante pertence. A partir desse ponto, todo o fluxo de utilização da plataforma é idêntico para ambos os grupos, assegurando que todos recebam o mesmo desafio técnico, com os mesmos detalhes e informações.

Um aspecto essencial considerado durante o desenvolvimento foi o registro automático do tempo total de execução de cada participante. Esse recurso elimina a necessidade de solicitar uma estimativa manual de tempo após a conclusão, o que poderia gerar dados imprecisos e subjetivos. Assim, o sistema contabiliza precisamente o período compreendido entre o momento em que o participante inicia o desafio e o instante em que o finaliza.

Além disso, a plataforma foi projetada para proporcionar uma experiência simples e direta tanto para o acesso ao código-fonte do desafio quanto para o envio da solução desenvolvida. Ao término da atividade, o sistema disponibiliza um botão que direciona o participante ao formulário de *feedback*, ajustado conforme o grupo ao qual pertence (sem ou com IA). Essa diferenciação se faz necessária, pois embora algumas perguntas sejam comuns entre os grupos, outras foram elaboradas especificamente para compreender a experiência de poder, ou não, utilizar ferramentas de IA generativa durante a execução do desafio.

5.2 Arquitetura e Tecnologias

A plataforma é dividida em três partes: a aplicação *web*, uma *API REST* e um banco de dados. A aplicação *web* é responsável pelo formulário inicial (cadastro e identificação) dos desenvolvedores, exibição dos detalhes do desafio técnico e por todo o fluxo de etapas ao longo do experimento, como início do desafio, envio do arquivo com a solução por parte do desenvolvedor e finalização do desafio.

A *API REST* é responsável por controlar toda a comunicação com o banco de dados do sistema, através das rotas configuradas que realizam operações como: registrar no banco de dados o tempo de ínicio do desenvolvedor (*timestamp*) ao iniciar o desafio, registrar o tempo de término, armazenar o tempo total em milissegundos (ms) para a conclusão do desafio, armazenar o tipo do grupo do grupo do desenvolvedor (com ou sem IA), cargo do mesmo, realizar o download da pasta *.zip* do código-fonte na máquina do participante, assim como realizar o

upload da solução do mesmo para o *back-end*.

Sobre as tecnologias utilizadas, a aplicação *web* foi desenvolvida utilizando *React.js*⁴. Esta biblioteca foi escolhida por ser bastante similar e utilizar *JavaScript* como base, simplificando assim a construção da plataforma, até mesmo pela sua praticidade para a organização de páginas e componentes reutilizáveis entre elas, o que facilita muito a organização durante o desenvolvimento. Para otimizar a construção do *front-end*, foi utilizado também a biblioteca *Material UI*⁵, uma biblioteca de componentes *React*, *open source*, que implementa o *Material Design* do *Google*.

Através das escolhas anteriores, acabou sendo determinante para que a *API REST* fosse construída utilizando *Express.js*⁶, sendo este um *framework* minimalista e flexível para *Node.js*⁷, que simplifica consideravelmente a criação de *APIs*, fornecendo ferramentas para tarefas como roteamento e manipulação de requisições *HTTP*. Além de prático com sua leveza e liberdade oferecida durante o processo de desenvolvimento, resultam em conseguir realizar a entrega de uma solução simples com mais precisão.

Por fim, a transferência de dados entre a aplicação *web* com a *API* foi feita utilizando arquivos *JSON*, recebendo-os e manipulando-os. Para o banco de dados, foi utilizado *PostgreSQL*⁸, por ser um banco de dados relacional de objeto e *open source*, conhecido por sua robustez, confiabilidade, extensibilidade e até mesmo, fácil manuseio e implementação. Ele suporta tanto dados relacionais (*SQL*) quanto não relacionais (*JSON*), compatibilidade com padrões *SQL*, e funcionalidades avançadas como chaves estrangeiras, gatilhos e recuperação pontual. Por ser *open source*, o *PostgreSQL* é igualmente gratuito. Para a estrutura das tabelas e seus dados, foi seguido o modelo de dados relacionais (*SQL*), afinal, foram necessárias somente três tabelas para o banco de dados, sendo elas:

- *Developer*, responsável por armazenar as informações do administrador e dos participantes;
- *Challenge*, responsável por armazenar os dados de início, término e tempo total da conclusão do desafio de cada participante;
- *DeveloperChallenge*, responsável por armazenar os dados referentes ao arquivo de solução do desenvolvedor.

O diagrama ER do banco de dados está disponível no Apêndice A. Sobre o *deploy* de todas as camadas da plataforma, a aplicação *web* foi hospedada na plataforma *Vercel*⁹, que possui planos gratuitos e também pela praticidade de sua integração ao repositório no *GitHub*¹⁰, que

⁴*React.js*: <<https://pt-br.reactjs.org>>

⁵*Material UI*: <<https://mui.com/material-ui/>>

⁶*Express.js*: <<https://expressjs.com/>>

⁷*Node.js*: <<https://nodejs.org/pt>>

⁸*PostgreSQL*: <<https://www.postgresql.org/>>

⁹*Vercel*: <<https://vercel.com/>>

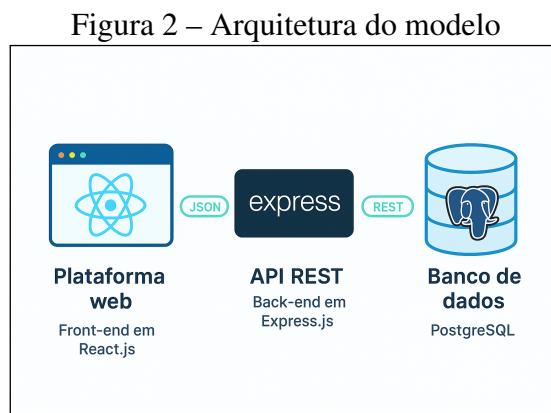
¹⁰*GitHub*: <<https://github.com/>>

facilita no processo de *deploys* automáticos conforme novos *commits* vão sendo feitos ao repositório remoto.

Por fim, a *API* e o banco de dados *PostgreSQL* foram hospedados na plataforma *Railway*¹¹, por possuir planos com créditos limitados para consumo/uso por alguns dias ou até o término dos créditos. Sua versatilidade para igualmente integrar ao repositório no *GitHub*, também trouxe praticidade para que o *deploy* da *API* fosse realizado sem muitas complicações a cada novo *commit*. Para o banco de dados, foi exportado somente o arquivo `.dump` do ambiente local de desenvolvimento e importado para o *Railway*.

As ferramentas utilizadas no processo de desenvolvimento foram o *Visual Studio Code*¹² (*VS Code*), editor de código-fonte popularmente conhecido, especialmente para o desenvolvimento de aplicações *web* e praticamente para o uso do *terminal* dentro do *VS Code*. Outra ferramenta utilizada para os testes das rotas da *API* a medida em que iam sendo criadas, antes de serem integradas ao *front-end*, foi a ferramenta *Postman*¹³, por facilitar a criação, testes e documentação de *APIs*, permitindo o envio de requisições *HTTP/HTTPS*. Com sua interface intuitiva, se tornou prático ir fazendo o acompanhamento das requisições que eram feitas e a armazenagem dos dados no banco.

Um breve diagrama da arquitetura do modelo é ilustrado na Figura 2.



Fonte: Elaborado pelo autor

5.3 Funcionalidades

A seguir serão apresentadas as principais funcionalidades da plataforma proposta para o desenvolvimento deste modelo, detalhando cada uma delas. Nesta seção estão as telas dos fluxos de uso e etapas dentro da plataforma. As demais telas estão disponíveis no Apêndice B.

¹¹ Railway: <<https://railway.com/>>

¹² Visual Studio Code: <<https://code.visualstudio.com/>>

¹³ Postman: <<https://www.postman.com/>>

5.4 Formulário Inicial

Ao acessar a plataforma *web* através do *link* fornecido juntamente com as instruções iniciais via *WhatsApp* para cada participante, o desenvolvedor necessita preencher um breve formulário informando seu "Nome e Sobrenome"(que é somente de uso interno na plataforma), seu "Cargo atual"e então pressionar o botão "Registrar". Os campos "Nome e Sobrenome"e "Cargo atual"são obrigatórios para o participante poder avançar. Na Figura 3 está a tela de registro do participante do grupo sem IA e na Figura 4 está a tela de registro do participante do grupo com IA, respectivamente.

Figura 3 – Cadastro do participante do grupo sem IA

Cadastro - Grupo sem IA

Preencha seu Nome e Sobrenome *

Preencha seu cargo atual *

REGISTRAR

Fonte: Elaborado pelo autor

Figura 4 – Cadastro do participante do grupo com IA

Cadastro - Grupo com IA

Preencha seu Nome e Sobrenome *

Preencha seu cargo atual *

REGISTRAR

Fonte: Elaborado pelo autor

5.5 Tela Inicial e Informações do Desafio

Após o preenchimento do formulário inicial, o participante é direcionado para a primeira tela da plataforma, que neste caso é a tela do desafio técnico propriamente. Porém, nesta tela, antes de exibir os detalhes do desafio, é exibido mais algumas informações que reforçam a necessidade de atenção e dedicação do participante antes de dar início ao teste.

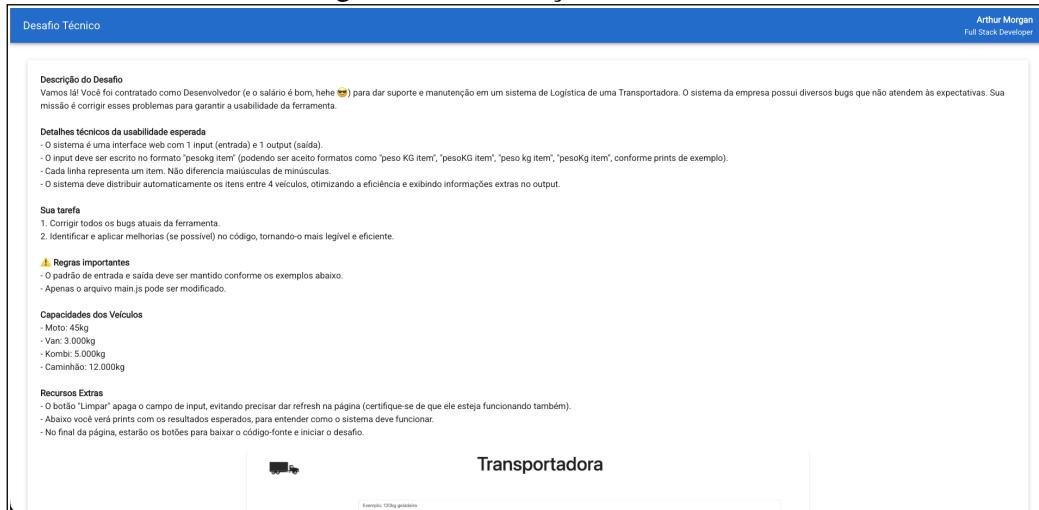
Estando de acordo, o participante clica no botão "De acordo" e na mesma tela é renderizada todos os detalhes e imagens de cenários de exemplo, descrevendo o desafio técnico que o desenvolvedor irá solucionar. A tela inicial e as informações do desafio encontram-se na Figura 5 e Figura 6, respectivamente.

Figura 5 – Tela inicial



Fonte: Elaborado pelo autor

Figura 6 – Informações do desafio



Fonte: Elaborado pelo autor

Ao descer um pouco a tela, o desenvolvedor encontra as imagens de cenários de exemplo que ilustram o objetivo final das funcionalidades que a solução precisa atender para concluir o desafio de modo satisfatório, como exemplo da Figura 7.

Figura 7 – Cenário de exemplo

The figure consists of two vertically stacked screenshots of a web application. Both screenshots have a header 'Guilherme Fraga' and a title 'Transportadora'. The top screenshot is labeled 'Exemplo 1: Operação básica do sistema'. It shows a form with a text area containing 'Digite os dados:' followed by three lines of text: '10kg Item 1', '1000kg Item 2', and '100kg Item 3'. Below the text area are two buttons: a green one labeled 'Enviar' and an orange one labeled 'Lançar'. The result section below shows 'Máx 1: 800kg Item 1', '10kg Item 2', '100kg Item 3', '1000kg Item 2', and 'Capacidade total: 8049kg'. The bottom screenshot is labeled 'Exemplo 2: Operação com multilinhas e diferentes formas escritas no input'. It has a similar layout but with different input data: '10kg Item 1', '1000kg Item 2', '100kg Item 3', '1000kg Item 4', and '1000kg Item 5'. The result section shows '1000kg Item 1', 'Carregando 2...', '1000kg Item 4', '1000kg Item 5', 'Peso total: 2000kg', 'Especie de saída: 2000kg', and 'Percentual carregado: 80.0%'. Both screenshots include a watermark 'Guilherme Fraga' in the bottom right corner.

Fonte: Elaborado pelo autor

Ao final da tela, conforme a Figura 8, está o botão para realizar o *download* do código-fonte do desafio técnico e logo abaixo, o botão para iniciar o desafio, que só é habilitado após o participante acionar o anterior, ou seja, realizar o *download*. Nesta parte também foi colocado mais um aviso reforçando a importância do participante preparar seu ambiente de desenvolvimento antes de clicar em "iniciar desafio", pois ao acionar este botão, uma rota da *API* é chamada que armazena no banco de dados o horário de início (*timestamp*) do desenvolvedor, registrando assim o momento exato em que o participante iniciou a atividade.

Figura 8 – Baixar e iniciar desafio

This screenshot shows the same application interface as Figure 7. At the bottom of the page, there is a large instruction block: 'Clique no botão abaixo para fazer o Download do código-fonte do desafio. Prepare o seu ambiente, se organize com calma e apenas clique no próximo botão quando estiver seguro para iniciar o desafio!'. Below this text are two buttons: a blue rectangular button labeled 'BAIXAR DESAFIO' and a smaller grey rectangular button labeled 'INICIAR DESAFIO'.

Fonte: Elaborado pelo autor

5.6 Desafio Técnico

Para projetar o desafio técnico proposto dentro da plataforma, imaginou-se um cenário fictício, onde para isso foi reaproveitado um projeto particular utilizado anteriormente pelo pesquisador apenas para fins de estudo pessoal. Este projeto consiste em um sistema simples de logística de uma transportadora, onde, conforme mencionado anteriormente, trata-se de uma interface *web* básica, que permite a entrada de um *input* e retorna um *output*.

Sendo assim, os dados de *input* que são utilizados, nada mais são do que itens, no formato "pesokg item"(podendo ser aceito formatos como "peso KG item", "peso kg item" e etc.), que representam itens em que o sistema da transportadora deverá alocar automaticamente em cada tipo de veículo de acordo com a capacidade do mesmo e de acordo com o peso do item, otimizando a eficiência e exibindo os detalhes necessários de acordo com o *output* esperado pelo sistema.

No campo de *input* do sistema, cada linha representa um item, não diferenciando letras maiúsculas de minúsculas. Neste sistema, foram alocados quatro tipos de veículos com suas respectivas capacidades. Numa simulação, mais de um mesmo tipo de veículo poderia ser utilizado se, de acordo com os pesos dos itens, encaixassem neste mesmo tipo de veículo. Como por exemplo: 2 itens de 45kg, o veículo do tipo "moto" possui capacidade de 45kg, ou seja, seria alocado "duas motos" para carregarem estas cargas.

Para o desafio técnico, foi fornecido todo o código-fonte do sistema, incluindo arquivos como `index.html`, `style.css`, que compõem a interface *web* do desafio. Entretanto, como regra do desafio, foi exigido para os participantes que só realizassem as alterações no arquivo `main.js`, visto que este contém toda a lógica de negócio para as funcionalidades exigidas no sistema de logística. Assim, seria mais prático para os desenvolvedores utilizarem da interface *web* já pronta, sem a necessidade de ajustes, para a realização de testes manuais à medida que iriam realizando as correções necessárias para atingir os objetivos.

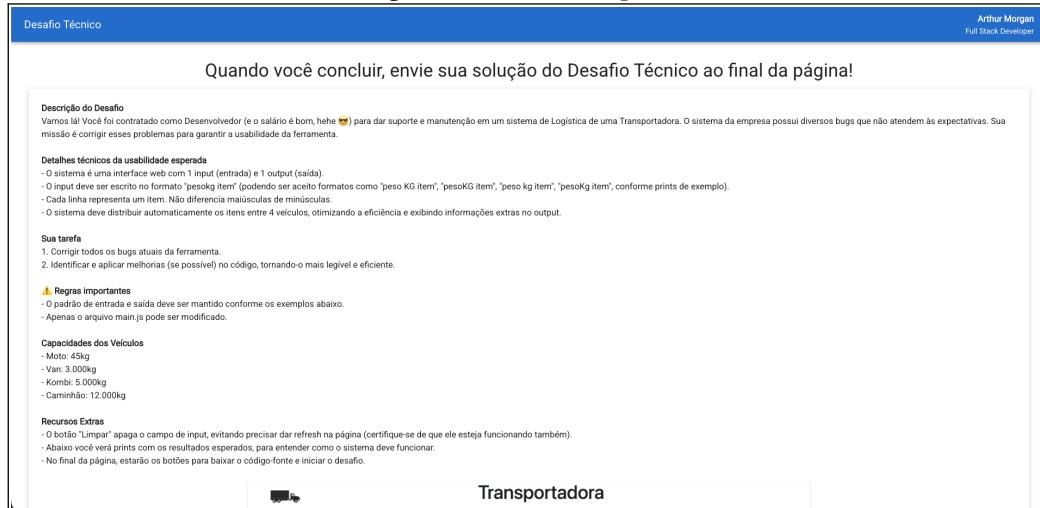
Isto dito, tendo como referência o código-fonte base (disponível completo no Apêndice G), foram sendo adicionados diversos *bugs* propositais no arquivo `main.js`, ou seja, sendo quebrado o código em diversas partes, para que os participantes do experimento solucionassem os mesmos. Estes *bugs* foram organizados em uma lista para controle e análises posteriores, após a conclusão e envio das soluções dos desenvolvedores, que encontra-se no Apêndice H. O arquivo `main.js` fornecido para os desenvolvedores com os *bugs* encontra-se disponível no Apêndice G.

5.7 Tela de *Upload*

Após o participante clicar no botão "Iniciar Desafio", o mesmo é direcionado para a tela de *upload*. Nesta tela, as informações referentes ao desafio permanecem sem alterações, para que fosse possível ao participante consultar a medida que fosse solucionando os *bugs*. O diferencial

desta tela se dá pelo título no topo, conforme a Figura 9, indicando ao participante que ao término da sua solução, o mesmo deve descer até o final da página para realizar o envio do seu arquivo através dos botões indicados.

Figura 9 – Tela de Upload



Fonte: Elaborado pelo autor

No final da página, encontram-se os botões para o participante selecionar seu arquivo e finalizar o desafio. O segundo botão só é habilitado após um arquivo ser selecionado. A plataforma ainda permite após escolher um arquivo, editar para selecionar outro, caso o primeiro tenha sido escolhido erroneamente ou caso o participante tenha feito alguma alteração antes de, de fato, finalizar o desafio.

Para fins de tornar mais prático a experiência do participante, foi limitado para que a plataforma aceitasse apenas arquivos *.zip*, pensando no cenário de que, após a conclusão por parte do desenvolvedor, o mesmo só precisaria compactar a pasta completa da sua solução e realizar o envio. Uma vez que o arquivo é selecionado, a aplicação o armazena em uma pasta temporária (*tmp*) no *back-end* e, após clicar no botão "Finalizar Desafio", a rota da *API* responsável por encerrar o desafio é chamada, onde então é armazenado no banco de dados o *timestamp* de conclusão do desenvolvedor e feito o cálculo da diferença entre o *timestamp* de conclusão e de início, para igualmente armazenar no banco de dados o tempo total, em milissegundos (ms) da duração para o participante concluir o desafio. Após isso, o arquivo é movido da pasta *tmp* para a pasta *uploads* no *back-end*. Na Figura 10 encontra-se os botões mencionados acima.

Figura 10 – Selecionar arquivo e finalizar desafio

The top part of the screenshot shows a web page with a title 'Transportadora'. It has a form with input fields and two buttons ('Enviar' and 'Cancelar'). Below the form is a 'Resultado:' section displaying some text. The bottom part shows a modal dialog with a blue header 'SELECIONAR ARQUIVO (.ZIP)'. It contains instructions about allowed file types and a button labeled 'FINALIZAR DESAFIO'.

Fonte: Elaborado pelo autor

Após o desafio finalizado, o participante é direcionado para uma tela final de agradecimento, com um botão para direcioná-lo ao preenchimento do formulário de *feedback* do experimento através da ferramenta *Google Forms*¹⁴, conforme a Figura 11 abaixo.

Figura 11 – Tela final e formulário

The page starts with a message 'Obrigado pela sua participação, Arthur Morgan!'. Below it is a note: 'Agora que você finalizou o desafio, clique no botão abaixo para realizar o preenchimento do formulário de feedback!'. At the bottom is a blue button labeled 'PREENCHER FORMULÁRIO'.

Fonte: Elaborado pelo autor

5.8 Painel de Administrador

O acompanhamento das soluções dos desenvolvedores foram realizadas através de um painel de uso exclusivo do administrador do experimento. Este painel foi projetado com o objetivo de centralizar as informações principais dos participantes, agrupando-os em dados gerais e em

¹⁴Google Forms: <<https://workspace.google.com/intl/pt-BR/products/forms/>>

seus grupos alocados (sem IA e com IA). O acesso deste painel é feito através de um *link*¹⁵, que funciona como um "login" simples, apenas informando o nome do administrador, que aciona uma rota da API para validar se este nome informado está registrado no banco de dados como administrador. Esta tela encontra-se na Figura 12.

Figura 12 – Acesso ao Painel de Administrador

Painel de Administrador

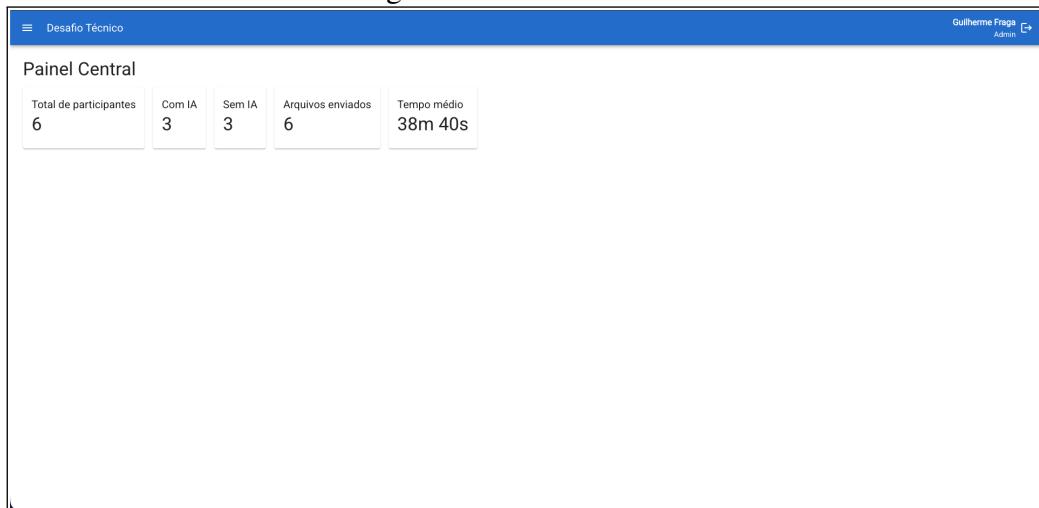
Nome*

ENTRAR

Fonte: Elaborado pelo autor

Acessando então como administrador, direciona-se para a tela de painel central, onde exibe informações gerais como o total de participantes do experimento, quantos pertencem ao grupo com IA, quantos pertencem ao grupo sem IA, a quantidade de arquivos enviados (soluções dos desenvolvedores) e o tempo médio geral de todos os participantes. A Figura 13 demonstra esta tela.

Figura 13 – Painel Central

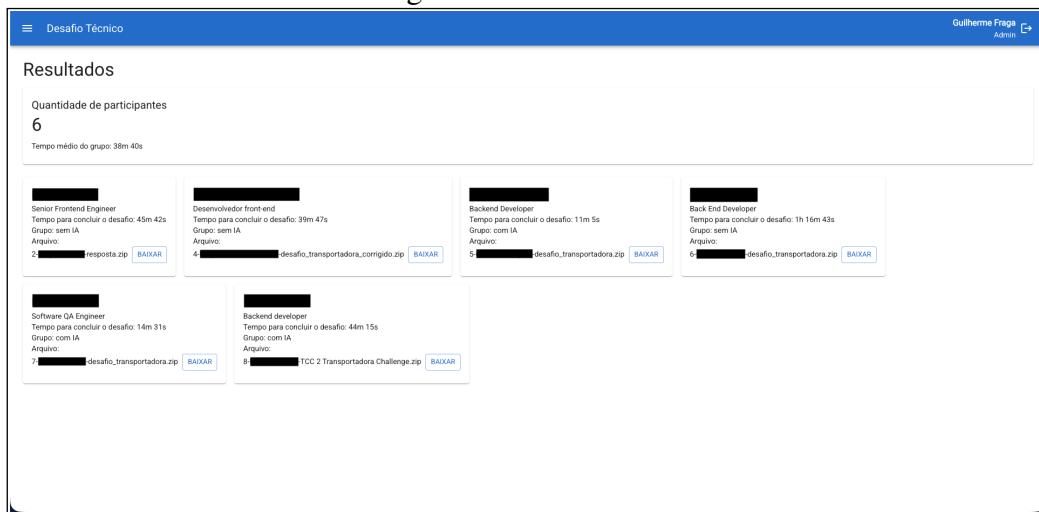


Fonte: Elaborado pelo autor

¹⁵Link para o Painel de Administrador: <<https://ai-use-productivity.vercel.app/admin>>

Através da barra de menu lateral à esquerda, tem-se a opção de "Resultados", que igualmente, exibe todos os resultados, individuais, de cada um dos participantes, em formatos de "cards". Para cada participante é exibido o seu cargo, o tempo total que o mesmo levou para concluir o desafio, o grupo ao qual pertencia e o nome do seu arquivo enviado. Para facilitar no momento de identificação dos arquivos, foi realizada uma lógica para que, no momento em que o participante fizesse o *upload* da sua solução, o arquivo fosse salvo no *backend* seguindo uma lógica de “*id-Nome_Sobrenome-nome_do_arquivo_do_participante.zip*”. Ao lado do nome de cada arquivo de cada participante, há o botão “Baixar”, para realizar o *download* da solução do desenvolvedor para as análises posteriores. O exemplo desta tela encontra-se na Figura 14. Com o intuito de manter o anonimato dos participantes do experimento, as imagens dos resultados do painel de administrador foram editadas para ocultar os nomes dos desenvolvedores.

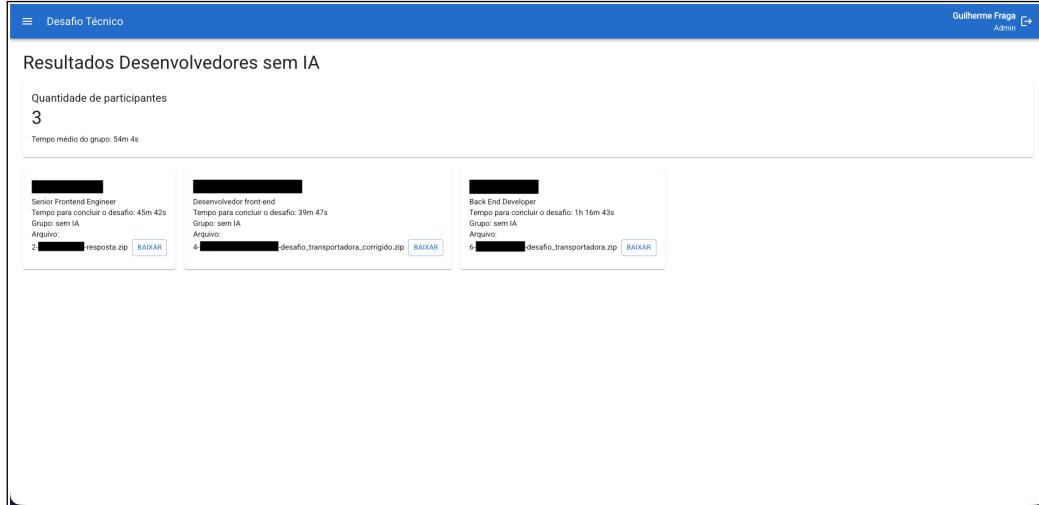
Figura 14 – Resultados



Fonte: Elaborado pelo autor

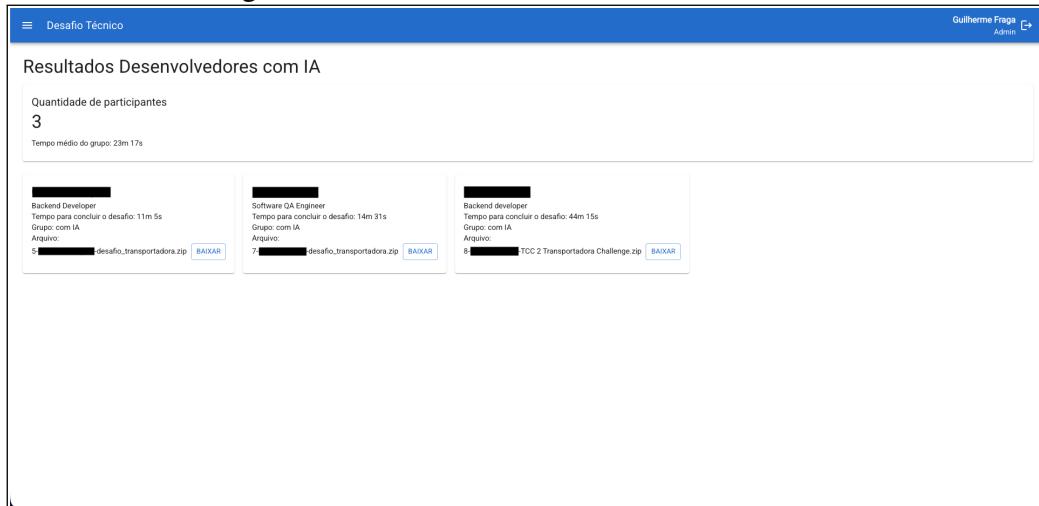
As outras duas opções dos menus na barra lateral são para separar os resultados nos dois grupos, através da tela "Resultados Desenvolvedores sem IA" e "Resultados Desenvolvedores com IA". Os exemplos destas telas com os resultados do grupo sem IA e com IA, podem ser conferidos através das Figuras 15 e 16, respectivamente.

Figura 15 – Resultados Desenvolvedores sem IA



Fonte: Elaborado pelo autor

Figura 16 – Resultados Desenvolvedores com IA



Fonte: Elaborado pelo autor

Todo o código-fonte da plataforma encontra-se disponível através do repositório¹⁶ na plataforma *GitHub*. Ao longo do desenvolvimento, foi utilizado o sistema de controle de versão *Git*¹⁷ para gerenciar todo o código-fonte.

6 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Nesta seção são apresentadas a coleta e análise dos dados levantados a partir das soluções fornecidas pelos participantes, juntamente com o registro do tempo total de duração para a conclusão do desafio de cada desenvolvedor. Foi coletado também dados de *feedback* referente

¹⁶Repositório: <<https://github.com/guifraga8/AI-Use-Productivity>>

¹⁷*Git*: <<https://git-scm.com/>>

ao experimento através de um questionário aplicado no término do desafio, com o intuito de fornecer mais resultados para uma análise relevante do projeto como um todo.

6.1 Coleta de Dados

Após a conclusão do desenvolvimento da plataforma *web*, foram selecionados os participantes do experimento por conveniência, isto é, profissionais conhecidos pelo pesquisador que atuam no mercado de trabalho na área de desenvolvimento de *software* e que se dispuseram a participar do desafio. Assim, o público-alvo foi definido de forma que todos os participantes fossem da área, porém ocupando diferentes cargos, como desenvolvedor *front-end*, desenvolvedor *back-end*, entre outros.

Na sequência, foi realizado um sorteio com os participantes para garantir uma separação igualitária e justa entre os membros de cada grupo, denominando-os como integrantes do grupo sem IA ou do grupo com IA. Em seguida, foi encaminhada, via *WhatsApp*, uma mensagem individual a cada participante com as primeiras instruções, reforçando detalhes importantes para a melhor participação e validação do experimento, de acordo com o grupo definido no sorteio. Ao final da mensagem, constava o *link* de acesso à plataforma (*link* do grupo sem IA¹⁸ e *link* do grupo com IA¹⁹), direcionando o participante à tela do formulário inicial. As mensagens enviadas encontram-se no Apêndice C.

Após a conclusão do desafio, isto é, após o envio da solução e finalização do mesmo (momento em que o banco de dados registrou o tempo total para a conclusão de cada desenvolvedor), cada participante foi direcionado a responder um questionário de *feedback* (sendo um formulário²⁰ para o grupo sem IA e um formulário²¹ para o grupo com IA). Da primeira à sétima questão, o conteúdo foi igual para ambos os grupos, com o objetivo de avaliar aspectos como o nível de conhecimento da linguagem de programação (*JavaScript*) proposta no desafio, a percepção individual quanto ao nível de dificuldade da atividade, o *feedback* sobre a clareza do material disponibilizado para o desenvolvimento da solução, a percepção sobre o tempo demandado para resolução, além de verificar se o participante utiliza ferramentas de IA Generativa em suas atividades profissionais voltadas ao desenvolvimento de *software* e, em caso afirmativo, quais.

Da oitava à décima questão, o foco recaiu sobre a abordagem adotada pelos participantes para solucionar o desafio, sendo estas perguntas diferentes para o grupo sem IA e para o grupo com IA. No grupo sem IA, questionou-se quais ferramentas foram utilizadas para auxiliar na resolução, apresentando múltiplas opções que não envolviam ferramentas de IA. Também foi solicitado um breve relato sobre como o desafio foi abordado sem o uso de IA e, por fim, pediu-se que descrevessem, em poucas palavras, sua percepção individual acerca dos obstáculos

¹⁸Link do grupo sem IA: <https://ai-use-productivity.vercel.app/register/without_ai>

¹⁹Link do grupo com IA: <https://ai-use-productivity.vercel.app/register/with_ai>

²⁰Formulário do grupo sem IA: <<https://forms.gle/zdH8UPeheATCtQg67>>

²¹Formulário do grupo com IA: <<https://forms.gle/cgqkPAs93RTPuJRH9>>

enfrentados (se houveram), considerando a restrição ao uso de ferramentas de IA Generativa.

Para o grupo com IA, as questões foram similares, porém adaptadas ao uso de ferramentas de IA Generativa na resolução do desafio. Assim, questionou-se quais ferramentas de IA foram utilizadas, foi solicitado igualmente um breve relato sobre como o desafio foi abordado com o auxílio de IA e, por fim, pediu-se que descrevessem, em poucas palavras, sua percepção individual quanto aos obstáculos enfrentados (se houveram) durante o desafio com o uso de IA Generativa.

A décima primeira questão foi comum a ambos os grupos, tendo como finalidade coletar *feedbacks* adicionais dos participantes a respeito do experimento como um todo. Essa pergunta foi incluída apenas para o encerramento do questionário e para fins de obtenção de sugestões voltadas à melhoria da plataforma ou do próprio experimento, visando contribuir com trabalhos futuros na mesma linha de pesquisa. Todas as questões encontram-se no Apêndice E, e suas respectivas respostas estão apresentadas no Apêndice F.

6.2 Avaliação dos Resultados

A partir dos arquivos de soluções dos participantes, cada um foi analisado tomando como base os *inputs* fornecidos nas imagens de exemplo dentro da plataforma e validando se o *output* de cada cenário, na solução de cada desenvolvedor, correspondia ao resultado esperado. Ou seja, para cada solução, primeiramente foram realizados testes utilizando a interface *HTML* do desafio, inserindo cada *input* e verificando o resultado. Dessa forma, tornou-se possível identificar de maneira prática se os *bugs* propostais do desafio haviam sido resolvidos. Os detalhes dos cenários de testes encontram-se disponíveis do Apêndice D.

Para cada *bug* pré-estabelecido, e ao executar os cenários de teste, foram atribuídas as classificações “Concluído”, “Concluído parcialmente” ou “Não concluído”, de modo a indicar se, na solução final do desenvolvedor, todos os ajustes necessários haviam sido realizados para atender à conclusão do desafio. Esse processo foi repetido minuciosamente, percorrendo todos os *bugs* pré-estabelecidos, desenvolvedor por desenvolvedor.

Nos cenários em que alguma operação não resultou no *output* esperado, o trecho específico do código era analisado para determinar se o respectivo *bug* havia sido concluído parcialmente ou não concluído de fato. Após essa análise, o código-fonte completo do arquivo *main.js* do participante foi comparado, lado a lado, com o código-fonte completo do arquivo *main.js* base do desafio (solucionado e funcional), a fim de realizar uma última análise minuciosa e verificar as alterações propostas por cada desenvolvedor, as decisões tomadas em aspectos específicos da lógica e reforçar a validação das classificações atribuídas a cada *bug* anteriormente. Todos os resultados desta etapa de análise encontram-se no Apêndice H.

Com base nos procedimentos executados, foi possível constatar que, analisando o escopo total de participantes, dos 6 desenvolvedores:

- 2 (33,33%) concluíram 100% dos *bugs*;

- 3 (50%) concluíram 90% dos *bugs*;
- 2 (66,67%) concluíram 10% dos *bugs* parcialmente;
- 1 (33,33%) não concluiu 10% dos *bugs*;
- 1 (16,67%) concluiu 70% dos *bugs*;
- 10% dos *bugs* foram concluídos parcialmente;
- 20% não foram concluídos.

Fazendo uma média com as porcentagens de "Concluídos, Concluídos parcialmente e Não concluídos" de todos os participantes, nota-se que os desenvolvedores concluíram 90% dos *bugs*, 5% concluídos parcialmente e 5% não foram concluídos. E por fim, o tempo médio fornecido através da plataforma, para a conclusão do desafio considerando os 6 participantes, foi de 38 minutos e 40 segundos.

Analizando somente os resultados do grupo sem IA, ou seja, no total de 3 participantes, constatou-se que:

- 1 (33,33%) concluiu 100% dos *bugs*;
- 2 (66,67%) concluíram 90% dos *bugs*;
- 1 (50%) concluiu 10% dos *bugs* parcialmente;
- 1 (50%) não concluiu 10% dos *bugs*.

A partir da média com as porcentagens somente dos participantes do grupo sem IA, nota-se que os desenvolvedores concluíram 93,33% dos *bugs*, 3,33% concluídos parcialmente e 3,33% não foram concluídos. O tempo médio para a conclusão do desafio do grupo sem IA foi de 54 minutos e 4 segundos.

Analizando somente os resultados do grupo com IA, igualmente no total de 3 participantes, constatou-se que:

- 1 (33,33%) concluiu 100% dos *bugs*;
- 1 (33,33%) concluiu 90% dos *bugs*;
- 10% dos *bugs* foram concluídos parcialmente;
- 1 (33,33%) concluiu 70% dos *bugs*;
- 10% dos *bugs* foram concluídos parcialmente;
- 20% não foram concluídos.

A partir da média somente dos participantes do grupo com IA, nota-se que os desenvolvedores concluíram 86,67% dos *bugs*, 6,67% concluídos parcialmente e 6,67% não foram concluídos. O tempo médio para a conclusão do desafio do grupo com IA foi de 23 minutos e 17 segundos.

A Tabela 3 foi elaborada para trazer os resultados individuais de cada participante, de forma resumida, para que seja possível exemplificar os pontos relevantes para as conclusões deste experimento.

Tabela 3 – Resumo dos resultados dos participantes

Participante	Cargo	Grupo	Resolução dos bugs	Tempo total para conclusão
Desenvolvedor 1	<i>Senior Frontend Engineer</i>	sem IA	90% concluído, 10% concluído parcialmente	45m 42s
Desenvolvedor 2	Desenvolvedor <i>front-end</i>	sem IA	100% concluído	39m 47s
Desenvolvedor 3	<i>Back End Developer</i>	sem IA	90% concluído, 10% não concluído	1h 16m 43s
Desenvolvedor 4	<i>Backend Developer</i>	com IA	90% concluído, 10% concluído parcialmente	11m 5s
Desenvolvedor 5	<i>Software QA Engineer</i>	com IA	70% concluído, 10% concluído parcialmente, 20% não concluído	14m 31s
Desenvolvedor 6	<i>Backend developer</i>	com IA	100% concluído	44m 15s

Fonte: Elaborado pelo autor

Com relação aos resultados dos questionários de *feedback* solicitados aos participantes, pontua-se alguns dados interessantes, onde dos 6 desenvolvedores:

- 6 (100%) afirmaram já possuirem conhecimento ou experiência na linguagem *JavaScript*;
 - 1 (16,67%) com nível de conhecimento muito básico;
 - 1 (16,67%) com nível de conhecimento básico;
 - 3 (50%) com nível de conhecimento avançado;
 - 1 (16,67%) com nível de muito avançado;
- 5 (83,33%) consideraram o desafio como fácil;
 - 1 (16,67%) considerou como neutro;
- 3 (50%) consideraram a clareza das informações do material entregue para a solução como razoável;
 - 2 (33,33%) consideraram como claro;
 - 1 (16,67%) considerou como muito claro;
- 3 (50%) consideraram o tamanho do desafio, com base no tempo levado para resolver, como adequado/na medida;

- 1 (16,67%) considerou como muito rápido;
- 1 (16,67%) considerou como rápido;
- 1 (16,67%) considerou como demorado;
- 6 (100%) afirmaram utilizarem ferramentas de IA Generativa no dia a dia profissional relacionadas ao desenvolvimento de *software*;
 - 3 (50%) utilizam o *ChatGPT*;
 - 2 (33,33%) utilizam o *GitHub Copilot*;
 - 1 (16,67%) utiliza o *Gemini*;
 - 3 (50%) utilizam o *Microsoft Copilot*;
 - 1 (16,67%) utiliza o *Amazon Q*;
 - 1 (16,67%) utiliza o *Cline*;
 - 1 (16,67%) utiliza *software* de autoria da empresa em que trabalha;
 - 1 (16,67%) utiliza o *Windsurf*;
 - 1 (16,67%) utiliza o *Claude*;

Outras informações relevantes obtidas através dos questionários foi de que, dos membros do grupo sem IA, os 3 desenvolvedores (100%) utilizaram do conhecimento prévio para a solução do desafio e destes apenas 1 (16,67%) utilizou também da documentação oficial da linguagem para ajudá-lo. Dos membros do grupo com IA, os 3 desenvolvedores fizeram uso das ferramentas de IA que estão acostumados a utilizarem, sendo elas o *ChatGPT* (66,67%), *Microsoft Copilot* (33,33%) e *Claude* (33,33%).

Por fim, os resultados completos dos questionários de *feedback* encontram-se disponíveis no Apêndice F.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

É inegável que, diante da rápida e constante evolução das ferramentas de Inteligência Artificial Generativa, o profissional da área de desenvolvimento de *software* precisa adaptar-se ao uso dessas tecnologias, buscando maior produtividade e desempenho em suas atividades. Considerando o tempo médio de resolução do desafio técnico proposto, observou-se que o grupo que utilizou ferramentas de IA levou, em média, menos da metade do tempo em comparação ao grupo sem o uso dessas ferramentas.

Esse resultado evidencia o ganho significativo de otimização de tempo e produtividade na resolução de problemas. Entretanto, um ponto relevante observado foi que um dos desenvolvedores do grupo sem IA conseguiu concluir todas as etapas do desafio em tempo inferior ao de um participante do grupo com IA que também completou integralmente a tarefa. Isso demonstra que, apesar de o uso da IA ter contribuído para uma execução mais rápida, nem todos os participantes com acesso à ferramenta conseguiram atingir 100% do resultado esperado.

Da mesma forma, verificou-se que o desenvolvedor do grupo com IA que solucionou completamente o desafio fez uso consciente e criterioso da ferramenta, analisando sua própria solução de forma robusta e não dependendo exclusivamente das respostas fornecidas pela IA. Essa observação foi reforçada pelas respostas do formulário de avaliação, nas quais o participante relatou que utilizou a IA como apoio para o raciocínio e não como substituto do pensamento crítico.

Por outro lado, alguns participantes do grupo sem IA aparentemente não realizaram uma verificação detalhada de suas soluções. Assim, destaca-se a importância de que, mesmo com o auxílio de ferramentas de IA, o desenvolvedor mantenha uma postura analítica e revisora, garantindo que o resultado final atenda integralmente aos requisitos propostos, com clareza e qualidade.

Ressalta-se, portanto, que as ferramentas de IA Generativas são grandes aliadas do desenvolvedor moderno, desde que utilizadas de forma equilibrada. O uso responsável dessas tecnologias deve reforçar e, não substituir, o conhecimento técnico e o senso crítico do profissional. Afinal, trata-se de ferramentas ainda em processo de evolução, mesmo que esse avanço ocorra em ritmo acelerado.

Essa relação de complementaridade entre desenvolvedor e IA Generativa também ficou evidente nas respostas dos participantes do grupo sem IA, que apontaram como principais desafios a ausência de recursos como o *autocomplete* e a dificuldade em realizar pesquisas e revisões de código sem o suporte de IA. Um dos desenvolvedores, inclusive, destacou que, antes do experimento, acreditava ser plenamente capaz de desenvolver sem o uso de IA, mas, após a experiência, percebeu o quanto o processo se torna mais demorado e trabalhoso sem essas ferramentas.

Esses relatos reforçam a percepção de que as ferramentas de IA Generativas se tornaram parte essencial do cotidiano dos desenvolvedores, contribuindo diretamente para a performance, a otimização de tempo e a praticidade em tarefas rotineiras. Contudo, é fundamental que seu uso seja feito de forma consciente, evitando dependências que possam limitar a autonomia e o raciocínio do profissional.

Em relação ao experimento realizado, os *feedbacks* dos participantes indicaram a necessidade de maior clareza na descrição do desafio técnico. Sugere-se que, em futuras aplicações, os exemplos de *inputs* e *outputs* sejam fornecidos em formato textual, e não apenas em imagens, permitindo que os desenvolvedores possam copiar e testar diretamente as informações, evitando erros e retrabalhos.

Por fim, destaca-se que seria relevante repetir o experimento com uma amostra maior de participantes, possibilitando uma análise mais ampla e representativa sobre o impacto real do uso e da ausência de ferramentas de IA Generativa no desempenho de desenvolvedores de *software*. Outra sugestão de melhoria para aplicações futuras seria a de, ao realizar um experimento desse tipo, mensurar não apenas o tempo e a capacidade de resolução do desafio técnico, mas também aspectos técnicos relacionados à qualidade das soluções entregues. Isso incluiria critérios como

a clareza e organização do código, nível de otimização, boas práticas aplicadas, modularização, bem como a análise de complexidade (melhor caso, pior caso e caso médio). Dessa forma, seria possível realizar uma comparação mais abrangente entre os grupos que utilizaram ferramentas de IA e aqueles que não utilizaram, permitindo avaliar não apenas a produtividade, mas também a qualidade técnica das soluções geradas.

Como proposta para trabalhos futuros, além das sugestões acima, seria interessante aplicar novos desafios técnicos envolvendo grupos com e sem IA, em que os participantes com IA não necessariamente sejam desenvolvedores ativos, mas profissionais da área de tecnologia em diferentes funções. Dessa forma, seria possível avaliar o impacto do uso de IA também em profissionais com conhecimento básico em lógica e programação, mas com perfis distintos, ampliando o entendimento sobre o potencial e os limites da IA Generativa em diferentes contextos de atuação.

Referências

- BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <<https://agilemanifesto.org/>>. Acesso em: 29 ago. 2025.
- COSTA, V. J. L. **ChatGPT: uma análise da ferramenta aplicada no processo de desenvolvimento de software**. Goiânia, GO: [s.n.], 2024. Pontifícia Universidade Católica de Goiás. Disponível em: <<https://repositorio.pucgoias.edu.br/jspui/handle/123456789/7929>>. Acesso em: 23 ago. 2025.
- COUTINHO, E.; BEZERRA, C. Simulação de alocação de recursos em projetos de desenvolvimento de software utilizando teoria das filas. In: **Anais do III Workshop em Modelagem e Simulação de Sistemas Intensivos em Software**. Porto Alegre, RS: SBC, 2021. p. 30–39. ISSN 0000-0000. Disponível em: <<https://sol.sbc.org.br/index.php/msss/article/view/17257>>. Acesso em: 24 ago. 2025.
- COUTINHO, M. et al. **The Role of Generative AI in Software Development Productivity: A Pilot Case Study**. 2024. Disponível em: <<https://arxiv.org/abs/2406.00560>>. Acesso em: 12 set. 2025.
- CÂNDIDO, A. K. R. **Uma revisão sistemática de estudos secundários sobre práticas ágeis de desenvolvimento de software**. Goiânia, GO: [s.n.], 2022. Pontifícia Universidade Católica de Goiás. Disponível em: <<https://repositorio.pucgoias.edu.br/jspui/handle/123456789/4436>>. Acesso em: 24 ago. 2025.
- DELANEY, J. **Catching the Vibe of Vibe Coding**. 2025. CACM/ACM. Disponível em: <<https://cacm.acm.org/news/catching-the-vibe-of-vibe-coding/>>. Acesso em: 30 set. 2025.
- FERREIRA, I. S. **Benefícios e desafios do uso de IAs na programação**. Salgueiro, PE: [s.n.], 2023. TCC (Sistemas para Internet) – Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano. Disponível em: <<https://releia.ifsertao-pe.edu.br/jspui/handle/123456789/1184>>. Acesso em: 21 jul. 2025.
- GITHUB. **GitHub Copilot documentation**. 2021. Disponível em: <<https://docs.github.com/en/copilot>>. Acesso em: 12 set. 2025.

GOMES, P. R. P. **Uma análise preliminar das IA generativa no suporte ao desenvolvimento de software.** Serra, ES: [s.n.], 2023. 61 p. Monografia (Bacharelado em Sistemas de Informação) – Instituto Federal do Espírito Santo. Disponível em: <<https://repositorio.ifes.edu.br/handle/123456789/5914>>. Acesso em: 21 jul. 2025.

MAYNARD, A. D. Navigating the fourth industrial revolution. **Nature Nanotechnology**, Tempe, Arizona, EUA, v. 10, p. 1005–1006, 2015. Disponível em: <<https://www.nature.com/articles/nano.2015.286>>. Acesso em: 24 ago. 2025.

NETO, J. G. **Metodologias ágeis em uma microempresa de desenvolvimento de softwares: um estudo de caso com o uso do Scrum.** Curitiba, PR: [s.n.], 2019. Universidade Tecnológica Federal do Paraná. Disponível em: <<http://repositorio.utfpr.edu.br/jspui/handle/1/19439>>. Acesso em: 24 ago. 2025.

SAKURAI, R.; ZUCHI, J. D. AS REVOLUÇÕES INDUSTRIAIS ATÉ A INDUSTRIA 4.0. **Revista Interface Tecnológica**, Taquaritinga, SP, v. 15, n. 2, p. 480–491, 2018. Disponível em: <<https://revista.fatectq.edu.br/interfacetecnologica/article/view/386>>. Acesso em: 24 ago. 2025.

SANTOS, G. P. dos. **Inteligência artificial generativa: o processo de ensino-aprendizagem no ensino superior de tecnologia.** Tese (Dissertação – Mestrado em Administração de Organizações) — Faculdade de Economia, Administração e Contabilidade de Ribeirão Preto, University of São Paulo, Ribeirão Preto, SP, 2024. Disponível em: <<https://www.teses.usp.br/teses/disponiveis/96/96132/tde-23012025-124616/en.php>>. Acesso em: 22 jul. 2025.

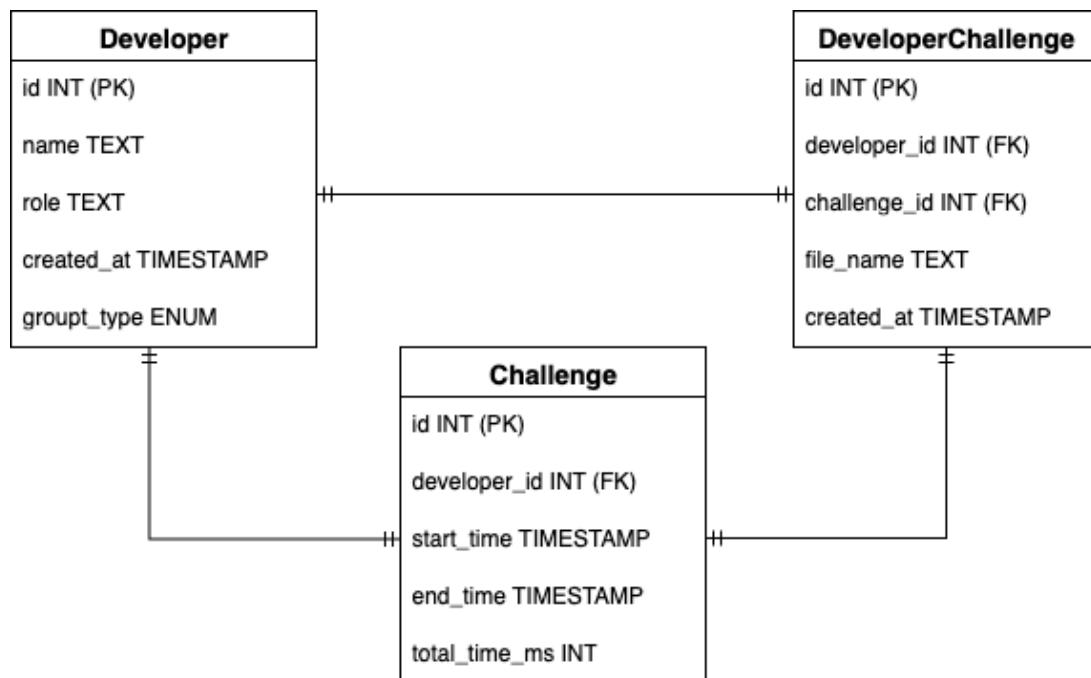
SCHWAB, K. **A Quarta Revolução Industrial.** [S.l.]: Edipro, 2019.

SILVA, I. H. da et al. DESENVOLVIMENTO DE APLICAÇÃO WEB COM INTELIGÊNCIA ARTIFICIAL GENERATIVA COMO AGENTE TUTOR SUPERVISOR NA CODIFICAÇÃO. **ARACÊ**, v. 7, n. 5, p. 22855–22877, 2025. Disponível em: <<https://periodicos.newsciencepubl.com/arace/article/view/4942>>. Acesso em: 21 jul. 2025.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Addison Wesley, 2011.

APÊNDICE A – DIAGRAMA ER DO BANCO DE DADOS

Figura A.1 – Modelo Entidade Relacional da aplicação



Fonte: Elaborado pelo autor

APÊNDICE B – TELAS DA APLICAÇÃO

Figura B.1 – Exemplo 1 do desafio

Tela inicial do sistema

Transportadora

Digite os dados:

Resultado:

```

Vazão:
10kg pedra
Capacidade total: 3000kg
Peso: 10kg
Excesso de peso: 2990kg
Percentual corrigido: 4,83%
  
```

Exemplo 1: Operação básica do sistema

Transportadora

Digite os dados:

Resultado:

```

Vazão:
2kg item 1
5000kg item 2
5000kg item 3
Capacidade total: 804kg
  
```

Fonte: Elaborado pelo autor

Figura B.2 – Exemplo 2 do desafio

Exemplo 1: Operação básica do sistema

Transportadora

Digite os dados:

Resultado:

```

Vazão:
2kg item 1
5000kg item 2
5000kg item 3
Capacidade total: 804kg
  
```

Exemplo 2: Operação com multilinhas e diferentes formas escritas no input

Transportadora

Digite os dados:

Resultado:

```

Vazão:
2kg item 1
5000kg item 2
5000kg item 3
Capacidade total: 804kg
  
```

Fonte: Elaborado pelo autor

Figura B.3 – Exemplo 2.1 do desafio

The figure consists of two vertically stacked screenshots of a web application. Both screenshots have a header 'Guilherme Fraga' at the top right and a logo of a truck at the top left.

Screenshot 1 (Top):

- Header: 'Exemplo 2: Operação com multilinhas e diferentes formas escritas no input.'
- Form: 'Digite os dados:' containing three lines of input: '1000kg item 1', '1000kg item 2', and '1000kg item 3'.
- Buttons: 'Enviar' (green) and 'Cancelar' (orange).
- Result area: 'Resultado:' showing the total weight: 'Capacidade total: 3000kg', 'Peso total: 3000kg', 'Porcentual cheio: 100%', and 'Percentual carregado: 75.00%'. It also includes the text 'Exemplo: 100kg grudeira'.
- Header: 'Guilherme Fraga'

Screenshot 2 (Bottom):

- Header: 'Exemplo 2.1: Operação com multilinhas e diferentes formas escritas no input.'
- Form: 'Digite os dados:' containing the text 'Exemplo: 100kg grudeira'.
- Buttons: 'Enviar' (green) and 'Cancelar' (orange).
- Result area: 'Resultado:' showing the message 'Nenhum item válido para processar.'
- Header: 'Guilherme Fraga'

Fonte: Elaborado pelo autor

Figura B.4 – Exemplo 3 do desafio

The figure consists of two vertically stacked screenshots of a web application. Both screenshots have a header 'Guilherme Fraga' at the top right and a logo of a truck at the top left.

Screenshot 1 (Top):

- Header: 'Exemplo 2.1: Operação com multilinhas e diferentes formas escritas no input.'
- Form: 'Digite os dados:' containing the text 'Exemplo: 100kg grudeira'.
- Buttons: 'Enviar' (green) and 'Cancelar' (orange).
- Result area: 'Resultado:' showing the message 'Nenhum item válido para processar.'
- Header: 'Guilherme Fraga'

Screenshot 2 (Bottom):

- Header: 'Exemplo 3: Operação clicando em "Enviar" com input vazio ou formato inválido'
- Form: 'Digite os dados:' containing the text 'Vazio aí'.
- Buttons: 'Enviar' (green) and 'Cancelar' (orange).
- Result area: 'Resultado:' showing the message 'Nenhum item válido para processar.'
- Header: 'Guilherme Fraga'

Fonte: Elaborado pelo autor

Figura B.5 – Exemplo 3.1 do desafio

The figure consists of two vertically stacked screenshots of a web application. Both screenshots have a header 'Guilherme Fraga' at the top right and a footer 'Exemplo 3.1: Operação clicando em "Enviar" com input vazio ou formato inválido' at the bottom.

Screenshot 1: The input field contains the placeholder 'Digite os dados:' and has a red border. Below the input field, a message box displays 'Nenhum item válido para processar.' The result area is empty.

Screenshot 2: The input field contains the placeholder 'Digite os dados:' and has a red border. Below the input field, a message box displays '500 kg Item 1
2000kg Item 2
10000kg Item 3
10000kg Item 4'. The result area is empty.

Fonte: Elaborado pelo autor

Figura B.6 – Exemplo 4 do desafio

The figure consists of two vertically stacked screenshots of a web application. Both screenshots have a header 'Guilherme Fraga' at the top right and a footer 'Exemplo 4: Outra operação com multilinhas e diferentes formas escritas no input' at the bottom.

Screenshot 1: The input field contains the placeholder 'Digite os dados:' and has a red border. Below the input field, a message box displays 'Item 1:
500kg Item 1
2000kg Item 2
Comprido 1
Comprido 2
10000kg Item 4'. The result area is empty.

Screenshot 2: The input field contains the placeholder 'Digite os dados:' and has a red border. Below the input field, a message box displays '500 kg Item 1
2000kg Item 2
10000kg Item 3
10000kg Item 4'. The result area is empty.

Fonte: Elaborado pelo autor

Figura B.7 – Exemplo 4.1 do desafio

The screenshot shows a challenge interface titled "Transportadora". At the top, there is a header with the text "Exemplo 4: Outra operação com multilinhas e diferentes formas escritas no input". Below this is a section labeled "Digite os dados:" containing the following text:

```
500 kg item 1
2000 kg item 2
1000 kg item 3
10000kg item 4
```

Below this are two buttons: "Enviar" (green) and "Lerar" (yellow).

Underneath the input section is a "Resultado:" section containing the following text:

```
10000kg item 3
Corridinha 2
10000kg item 4
Corridinha total 27500kg
Peso total 24500kg
Espaço de sobre 2500kg
Percentual completo: 90.3%
```

At the bottom right of the main window is the name "Guilherme Fraga".

At the bottom of the page, there is a message: "Clique no botão abaixo para fazer o Download do código-fonte do desafio. Prepare o seu ambiente, se organize com calma e apenas clique no próximo botão quando estiver seguro para iniciar o desafio!" followed by two buttons: "BAIXAR DESAFIO" (blue) and "INICIAR DESAFIO" (green).

Fonte: Elaborado pelo autor

Figura B.8 – Editar arquivo e finalizar desafio

The screenshot shows a challenge interface titled "Transportadora". At the top, there is a header with the text "Exemplo 4: Outra operação com multilinhas e diferentes formas escritas no input". Below this is a section labeled "Digite os dados:" containing the following text:

```
500 kg item 1
2000 kg item 2
1000 kg item 3
10000kg item 4
```

Below this are two buttons: "Enviar" (green) and "Lerar" (yellow).

Underneath the input section is a "Resultado:" section containing the following text:

```
10000kg item 3
Corridinha 2
10000kg item 4
Corridinha total 27500kg
Peso total 24500kg
Espaço de sobre 2500kg
Percentual completo: 90.3%
```

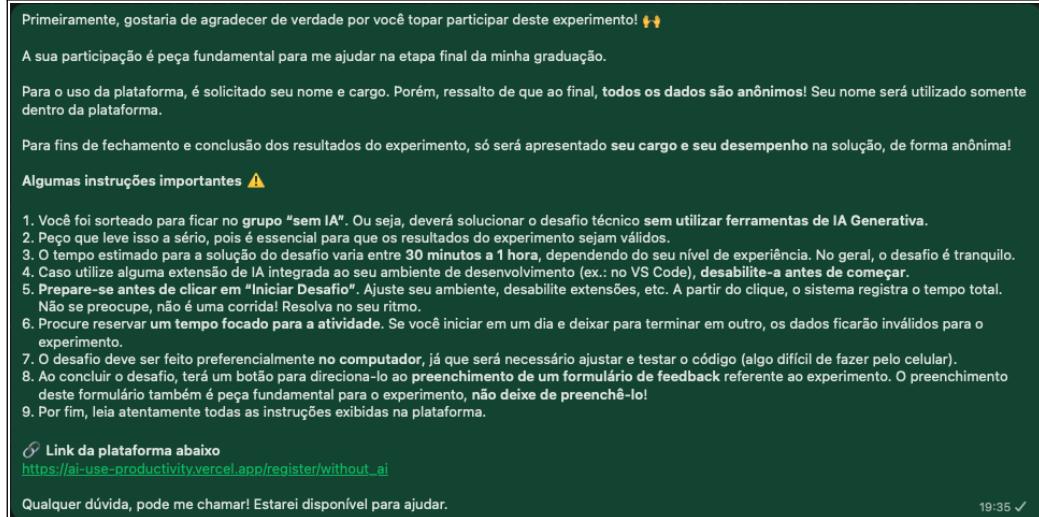
At the bottom right of the main window is the name "Guilherme Fraga".

At the bottom of the page, there is a message: "Arquivo selecionado: desafio_transportadora.zip Somente arquivos .zip são permitidos. Se já estiver seguro que encerrou o desafio e selecionou o arquivo correto, clique no botão abaixo para finalizar o desafio!" followed by a "FINALIZAR DESAFIO" button.

Fonte: Elaborado pelo autor

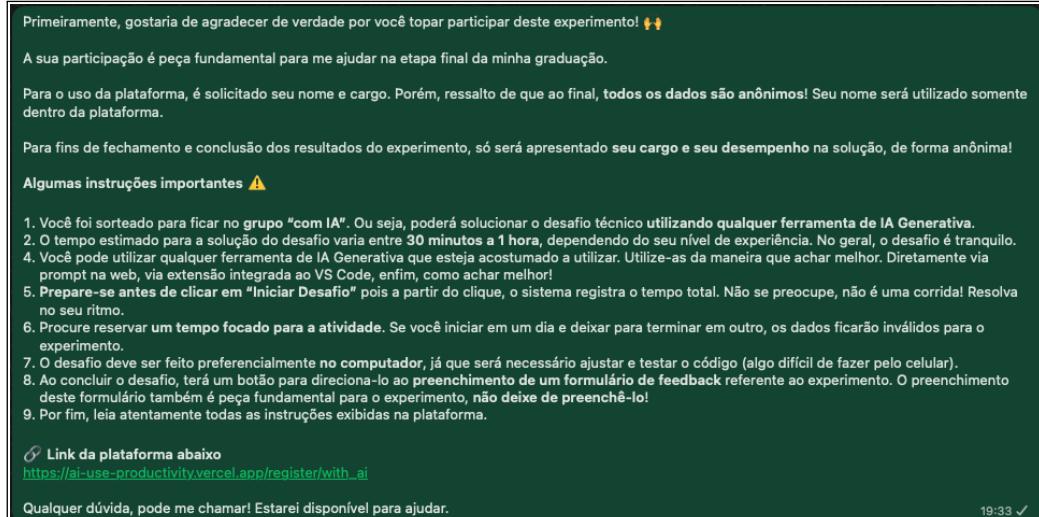
APÊNDICE C – INSTRUÇÕES ENVIADAS AOS PARTICIPANTES VIA WHATSAPP

Figura C.1 – Instruções enviadas via WhatsApp para o grupo sem IA



Fonte: Elaborado pelo autor

Figura C.2 – Instruções enviadas via WhatsApp para o grupo com IA



Fonte: Elaborado pelo autor

APÊNDICE D – CENÁRIOS DE TESTES

O desenvolvedor deveria replicar os *inputs* exibidos nas imagens de exemplo, que representavam os resultados esperados no *output*. Estes foram os *inputs* aplicados ao testes na interface *HTML* e comparado com o código-fonte base funcional para verificar se os resultados estavam coerentes.

Cenário 1

- *input*

120kg geladeira

- *output*

Van 1:

120kg geladeira

Capacidade total: 3000kg

Peso total: 120kg

Espaço de sobra: 2880kg

Percentual carregado: 4.00%

Cenário 2

- *input*

35kg Item 1

1000KG Item 2

5000Kg Item 3

- *output*

Moto 1:

35kg Item 1

Van 1:

1000kg Item 2

Kombi 1:

5000kg Item 3

Capacidade total: 8045kg

Peso total: 6035kg

Espaço de sobra: 2010kg

Percentual carregado: 75.02%

Cenário 3

- *input*

vazio

- *output*

Nenhum item válido para processar.

Cenário 4

- *input*

120abc abc

- *output*

Nenhum item válido para processar.

Cenário 5

- *input*

500 kg Item 1

2000 KG Item 2

10000kG Item 3

12000kg Item 4

- *output*

Van 1:

500kg Item 1

2000kg Item 2

Caminhão 1:

10000kg Item 3

Caminhão 2:

12000kg Item 4

Capacidade total: 27000kg

Peso total: 24500kg

Espaço de sobra: 2500kg

Percentual carregado: 90.74%

Cenário 6 – Melhoria/opcional

- *input*

15000 kg Item 1

- *output*

Nenhum item válido para processar (ou algo similar, mais descriptivo, por exemplo, mostrando que esta carga não comporta em nenhum dos tipos de veículos).

APÊNDICE E – QUESTIONÁRIOS DE AVALIAÇÃO DO MODELO

Figura E.1 – Questão 1 e 2 do formulário final para todos os participantes

The figure shows two questions from a survey. The first question asks if the participant had knowledge or experience with JavaScript, with options 'Sim' (Yes) and 'Não' (No). The second question asks how they assess their level of knowledge in JavaScript, with a scale from 1 (Muito básico - Very basic) to 5 (Muito avançado - Very advanced). The scale is marked with numbers 1, 2, 3, 4, 5 at the top and 'Muito básico' and 'Muito avançado' at the bottom, with five empty circles for rating.

Você já possuía conhecimento ou experiência em JavaScript? *

Sim
 Não

Se sim, como você avalia o seu nível de conhecimento em JavaScript?

Considere:

1. Muito básico
2. Básico
3. Moderado
4. Avançado
5. Muito avançado

1 2 3 4 5

Muito básico Muito avançado

Fonte: Elaborado pelo autor

Figura E.2 – Questão 3 e 4 do formulário final para todos os participantes

<p>Considerando o nível de dificuldade do desafio proposto, como você classificaria? *</p> <p>Considere:</p> <ul style="list-style-type: none"> 1. Muito fácil 2. Fácil 3. Neutro 4. Difícil 5. Muito difícil <p style="text-align: center;">1 2 3 4 5</p> <p>Muito fácil <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Muito difícil</p>				
<p>Como você classificaria a clareza das informações do material entregue (descrição, imagens de exemplo, código-fonte)? *</p> <p>Considere:</p> <ul style="list-style-type: none"> 1. Confuso 2. Pouco claro 3. Razoável 4. Claro 5. Muito claro <p style="text-align: center;">1 2 3 4 5</p> <p>Confuso <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Muito claro</p>				

Fonte: Elaborado pelo autor

Figura E.3 – Questão 5 e 6 do formulário final para todos os participantes

<p>O desafio demorou muito para você resolver, considerando o tamanho dele? *</p> <p>Considere:</p> <p>1. Muito rápido 2. Rápido 3. Adequado/na medida 4. Demorado 5. Muito demorado</p> <p style="text-align: center;">1 2 3 4 5</p> <p>Muito rápido <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Muito demorado</p>					
<p>Você utiliza ferramentas de IA Generativa no seu dia a dia profissional relacionadas ao Desenvolvimento de Software? *</p> <p><input type="radio"/> Sim <input type="radio"/> Não</p>					

Fonte: Elaborado pelo autor

Figura E.4 – Questão 7 do formulário final para todos os participantes

Se a resposta for sim, quais ferramentas você utiliza? (Selecione todas as opções aplicáveis e/ou indique em "Outro" se houver mais).

ChatGPT
 DeepSeek
 GitHub Copilot
 Gemini
 Grok
 Microsoft Copilot
 Outro: _____

Fonte: Elaborado pelo autor

Figura E.5 – Questão 8, 9 e 10 do formulário final para todos os membros do grupo sem IA

Quais ferramentas foram utilizadas na solução do desafio? (Selecione todas as aplicáveis e/ou indique em "Outro" se houver mais). *

Conhecimento prévio

Documentação oficial da linguagem

Comunidades especializadas (exemplo: Stack Overflow)

Fóruns (exemplo: Reddit)

Vídeos tutoriais (exemplo: YouTube)

Projetos semelhantes em repositórios (exemplo: GitHub)

Outro:

Em poucas palavras, como você abordou a solução do desafio sem usar ferramentas de IA Generativa? *

Texto de resposta longa

Como integrante do grupo que resolveu o desafio sem IA Generativa, quais obstáculos (se houveram) você enfrentou? Descreva em poucas palavras.

Texto de resposta curta

Fonte: Elaborado pelo autor

Figura E.6 – Questão 8, 9 e 10 do formulário final para todos os membros do grupo com IA

Quais ferramentas de IA Generativa foram utilizadas na solução do desafio? *
(Selecione todas as aplicáveis e/ou indique em "Outro" se houver mais).

ChatGPT
 DeepSeek
 GitHub Copilot
 Gemini
 Grok
 Microsoft Copilot
 Outro: _____

Em poucas palavras, como você abordou a solução do desafio utilizando ferramentas de IA Generativa? *

Sua resposta _____

Como integrante do grupo que resolveu o desafio com IA Generativa, quais obstáculos (se houveram) você enfrentou? Descreva em poucas palavras. *

Sua resposta _____

Fonte: Elaborado pelo autor

Figura E.7 – Questão 11 do formulário final para todos os participantes

The image shows a screenshot of a survey question. The question text is "Algum feedback adicional que gostaria de compartilhar referente ao experimento?". Below the question is a text input field labeled "Sua resposta".

Algum feedback adicional que gostaria de compartilhar referente ao experimento?

Sua resposta

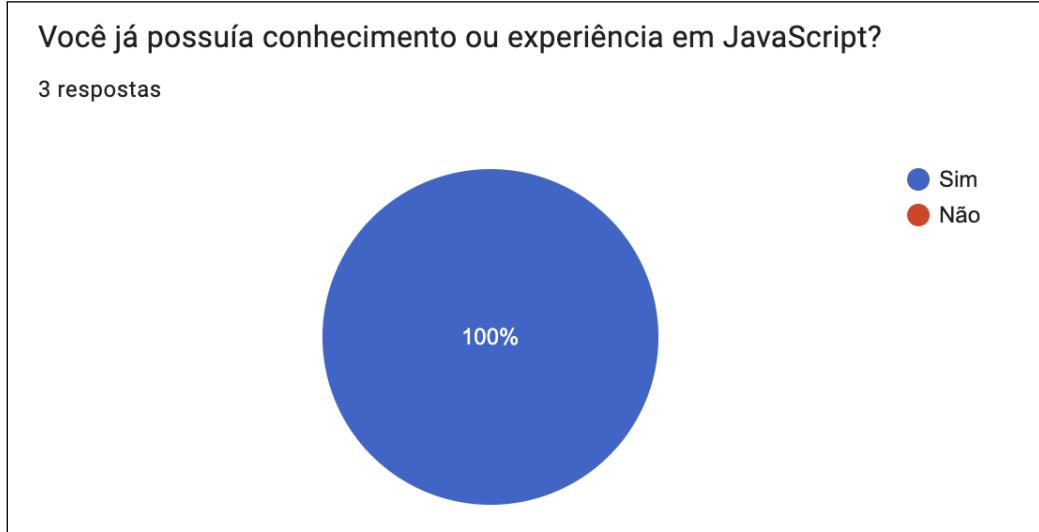
Fonte: Elaborado pelo autor

APÊNDICE F – RESPOSTAS DO FORMULÁRIO

Os gráficos e quadros abaixo representam as respostas completas do questionário, dos grupos sem IA e com IA, respectivamente. Os Gráficos F.1 e F.2 referem-se à questão 1. Os Gráficos F.3 e F.4 são as respostas da questão 2. Os Gráficos F.5 e F.6 referem-se à questão 3. Os Gráficos F.7 e F.8 são da questão 4. Os Gráficos F.9 e F.10 são referentes à questão 5. Os Gráficos F.11 e F.12 tratam-se da questão 6. Os Gráficos F.13 e F.14 referem-se à questão 7. Os Gráficos F.15 e F.16 são as respostas da questão 8.

As Tabelas F.1 e F.2 referem-se à questão 9. As Tabelas F.3 e F.4 são as respostas da questão 10. Por fim, as tabelas F.5 e F.6 tratam-se da questão 11.

Figura F.1 – Respostas da questão 1 de todos do grupo sem IA

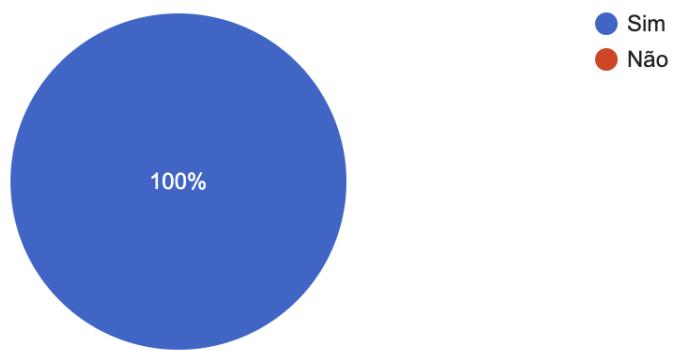


Fonte: Elaborado pelo autor

Figura F.2 – Respostas da questão 1 de todos do grupo com IA

Você já possuía conhecimento ou experiência em JavaScript?

3 respostas

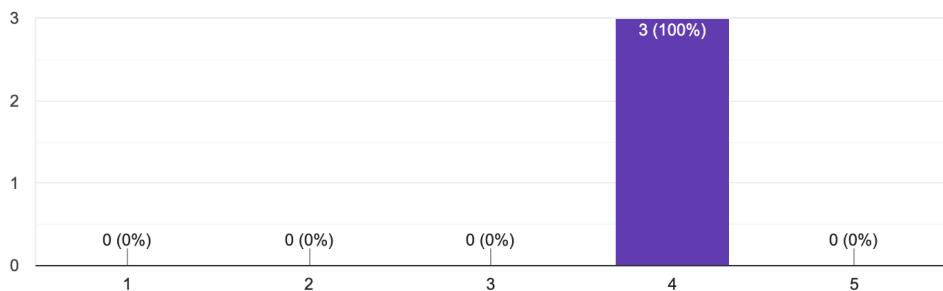


Fonte: Elaborado pelo autor

Figura F.3 – Respostas da questão 2 de todos do grupo sem IA

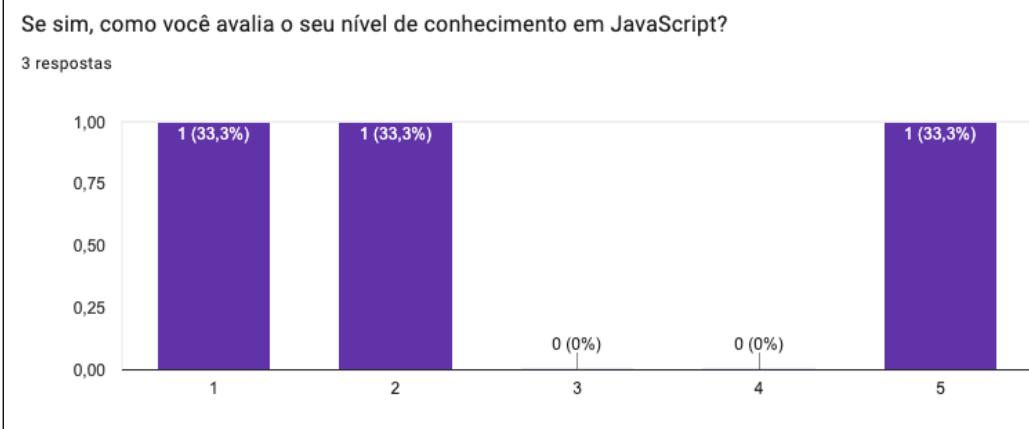
Se sim, como você avalia o seu nível de conhecimento em JavaScript?

3 respostas



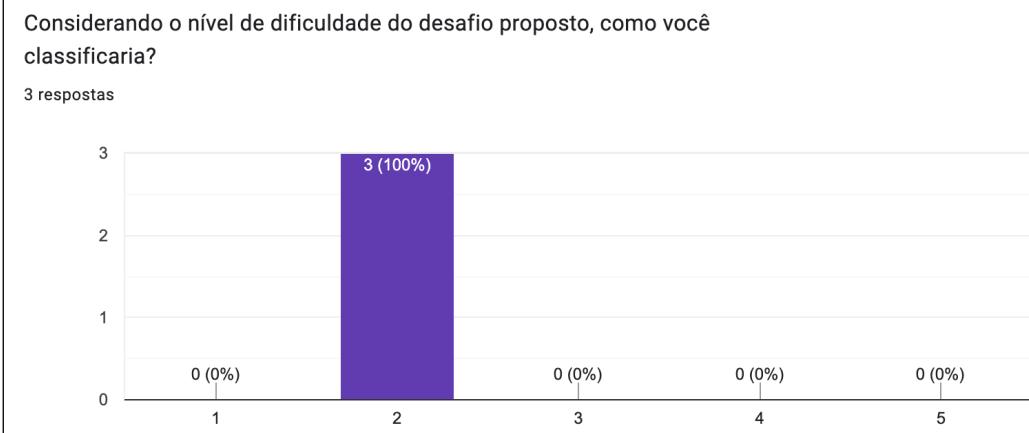
Fonte: Elaborado pelo autor

Figura F.4 – Respostas da questão 2 de todos do grupo com IA



Fonte: Elaborado pelo autor

Figura F.5 – Respostas da questão 3 de todos do grupo sem IA



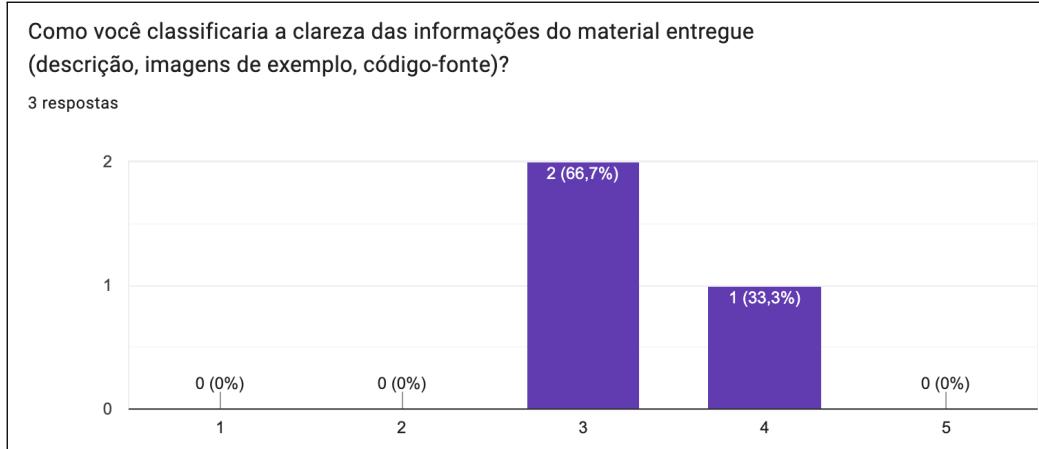
Fonte: Elaborado pelo autor

Figura F.6 – Respostas da questão 3 de todos do grupo com IA



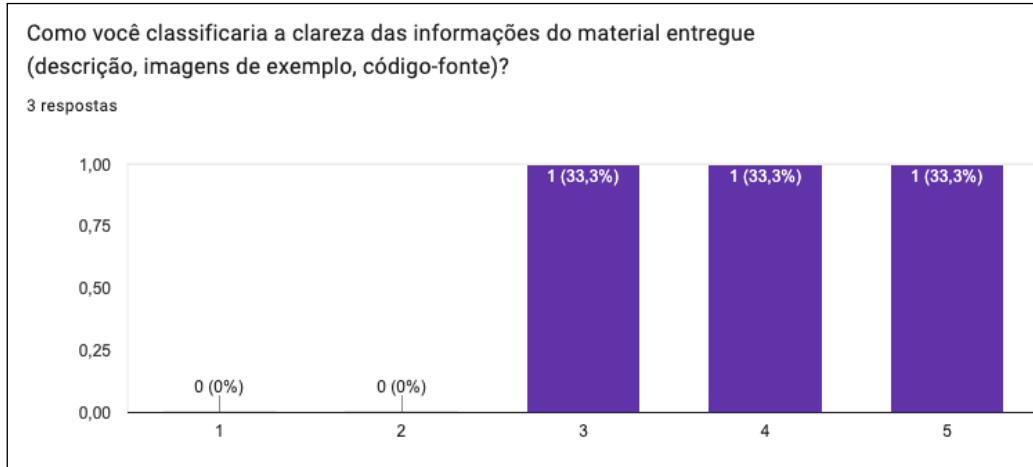
Fonte: Elaborado pelo autor

Figura F.7 – Respostas da questão 4 de todos do grupo sem IA



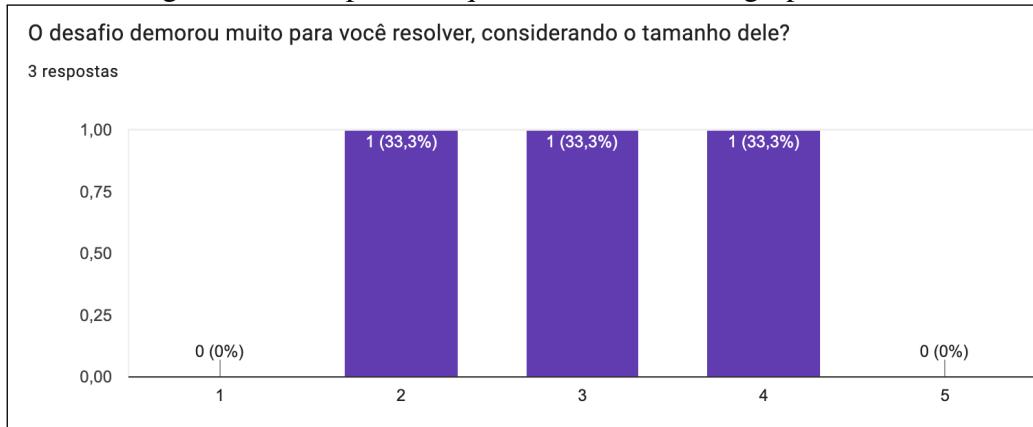
Fonte: Elaborado pelo autor

Figura F.8 – Respostas da questão 4 de todos do grupo com IA



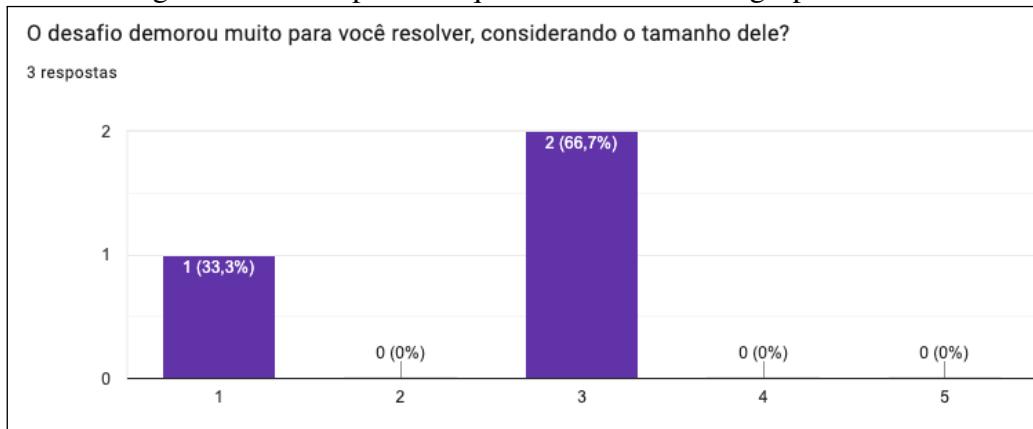
Fonte: Elaborado pelo autor

Figura F.9 – Respostas da questão 5 de todos do grupo sem IA



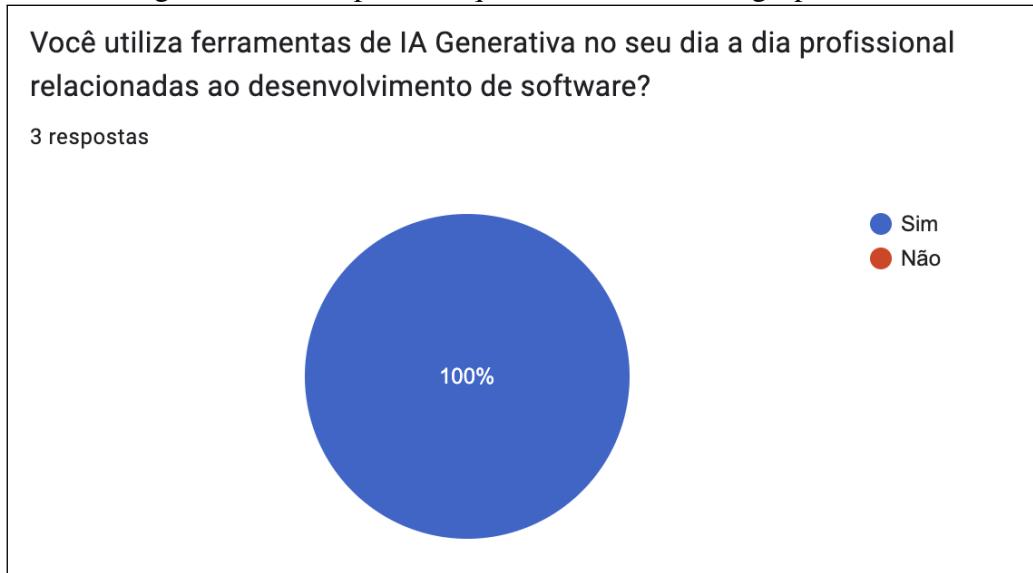
Fonte: Elaborado pelo autor

Figura F.10 – Respostas da questão 5 de todos do grupo com IA



Fonte: Elaborado pelo autor

Figura F.11 – Respostas da questão 6 de todos do grupo sem IA

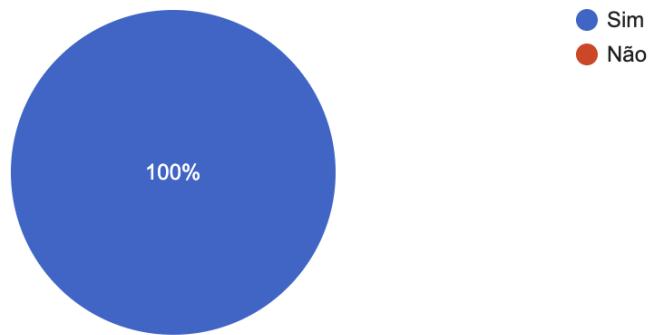


Fonte: Elaborado pelo autor

Figura F.12 – Respostas da questão 6 de todos do grupo com IA

Você utiliza ferramentas de IA Generativa no seu dia a dia profissional relacionadas ao desenvolvimento de software?

3 respostas

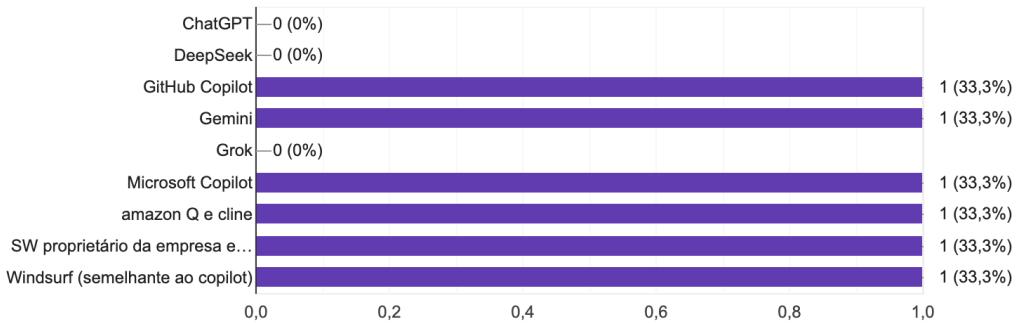


Fonte: Elaborado pelo autor

Figura F.13 – Respostas da questão 7 de todos do grupo sem IA

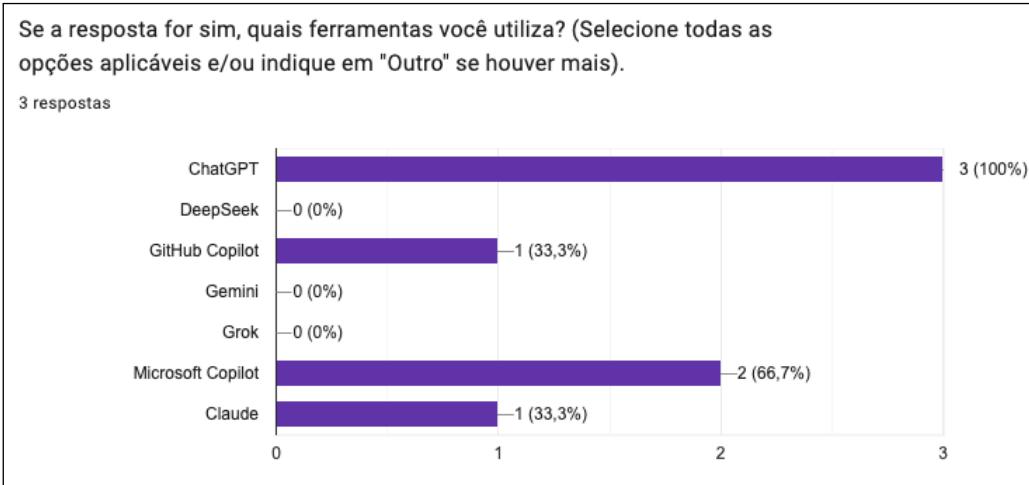
Se a resposta for sim, quais ferramentas você utiliza? (Selecione todas as opções aplicáveis e/ou indique em "Outro" se houver mais).

3 respostas



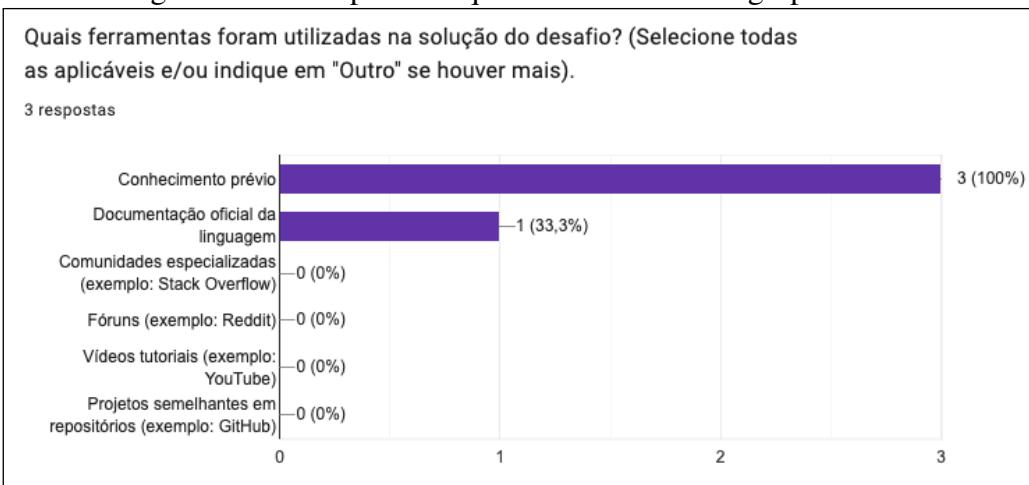
Fonte: Elaborado pelo autor

Figura F.14 – Respostas da questão 7 de todos do grupo com IA



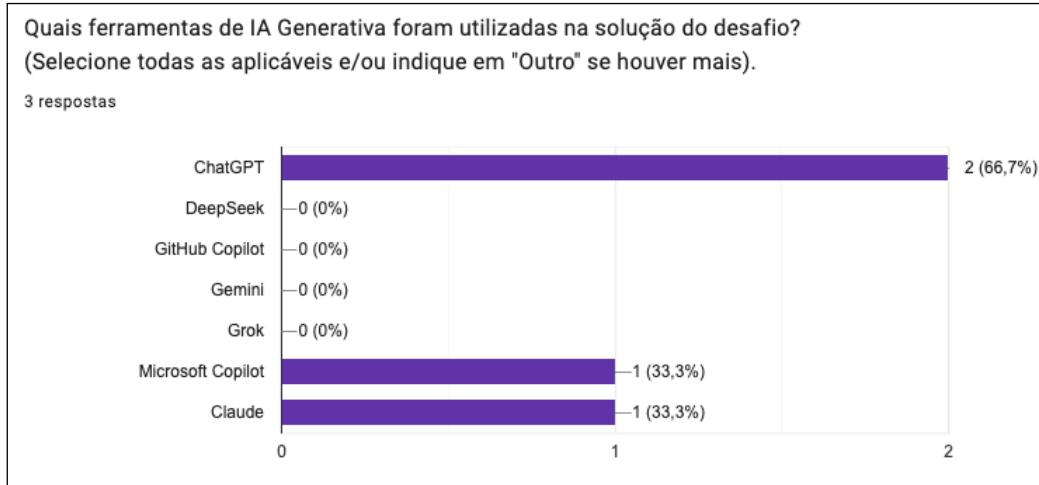
Fonte: Elaborado pelo autor

Figura F.15 – Respostas da questão 8 de todos do grupo sem IA



Fonte: Elaborado pelo autor

Figura F.16 – Respostas da questão 8 de todos do grupo com IA



Fonte: Elaborado pelo autor

Tabela F.1 – Respostas da questão 9 de todos do grupo sem IA

Participante	Em poucas palavras, como você abordou a solução do desafio sem usar ferramentas de IA Generativa?
Desenvolvedor 1	Primeiramente, tentei achar os bugs. Para isso, explorei a aplicação já pronta. Tendo isso em vista, fui encontrando a raiz dos problemas e resolvendo. Usei o debugger do navegador para entender quais os problemas com os dados.
Desenvolvedor 2	Realizei alguns testes e verifiquei o resultado final. A partir daí analisei o código e ajustei conforme fui debugando.
Desenvolvedor 3	Fui analisando os resultados e revisando o código no VSCode conforme eu encontrava os erros neles

Fonte: Elaborado pelo autor

Tabela F.2 – Respostas da questão 9 de todos do grupo com IA

Participante	Em poucas palavras, como você abordou a solução do desafio sem usar ferramentas de IA Generativa?
Desenvolvedor 4	Basicamente copiou os prints de output esperados e a descrição do desafio.
Desenvolvedor 5	Me fiz de leigo. Mandei a IA ler todo o projeto, onde ela ja encontrou e corrigiu bugs, e depois mandei o desafio e pedi pra ele ajustar o projeto de acordo. No final, pedi pra ele testar e me mandar evidencias que está correto. No final, fiz testes exploratorios manuais e entreguei o projeto.
Desenvolvedor 6	De inicio eu tentei fazer tudo sozinho, baseado na leitura do código e outputs do programa. Mas, depois de acreditar ter resolvido tudo e ainda assim ver alguns bugs, submeti o código inteiro ao chatGPT, que prontamente já me sugeriu alguns fixes antes de eu pedir qualquer coisa. Após isso, somente um bug sobrou, onde agora submeti o input e falei "o resultado atual está errado", e ele prontamente me deu os fixes novamente.

Fonte: Elaborado pelo autor

Tabela F.3 – Respostas da questão 10 de todos do grupo sem IA

Participante	Como integrante do grupo que resolveu o desafio sem IA Generativa, quais obstáculos (se houveram) você enfrentou? Descreva em poucas palavras.
Desenvolvedor 1	Encontrar os bugs em si.
Desenvolvedor 2	Senti falta do autocomplete fornecido pelo windsurf, já que utilizei bastante para tarefas mais manuais.
Desenvolvedor 3	A pesquisa por informações e a revisão de código tornam-se muito mais demoradas e complicadas sem o uso de IA.

Fonte: Elaborado pelo autor

Tabela F.4 – Respostas da questão 10 de todos do grupo com IA

Participante	Como integrante do grupo que resolveu o desafio com IA Generativa, quais obstáculos (se houveram) você enfrentou? Descreva em poucas palavras.
Desenvolvedor 4	chatgpt free, tem um limite muito pequeno de entradas, tive que ir para o Copilot
Desenvolvedor 5	Nenhum.
Desenvolvedor 6	Com IA, nenhum.

Fonte: Elaborado pelo autor

Tabela F.5 – Respostas da questão 11 de todos do grupo sem IA

Participante	Algum feedback adicional que gostaria de compartilhar referente ao experimento?
Desenvolvedor 2	Faltou ser um pouco mais claro na descrição do problema
Desenvolvedor 3	Interessante notar como a IA se tornou uma ferramenta tão importante no meu processo de desenvolvimento. Antes do experimento eu poderia dizer que facilmente conseguiria desenvolver sem o uso de IA mas após ele consigo notar que o processo se torna muito mais demorado sem o uso da mesma.

Fonte: Elaborado pelo autor

Tabela F.6 – Respostas da questão 11 de todos do grupo com IA

Participante	Algum feedback adicional que gostaria de compartilhar referente ao experimento?
Desenvolvedor 6	seria muito mais simples se nas instruções não houvessem prints da tela, mas sim textos de input e output, pra que a gente copiasse e colasse, invés de ter que ler a imagem e copiar.

Fonte: Elaborado pelo autor

APÊNDICE G – CÓDIGO-FONTE DO DESAFIO E CÓDIGO-FONTE BASE

```
1 /**
2 * @author Guilherme Fraga
3 * @since 13/09/2025
4 * @version 1.0
5 */
6
7 const vehicles = [
8     { description: "Moto", maxWeight: 45, vehicles: [] },
9     { description: "Van", maxWeight: 3000, vehicles: [] },
10    { description: "Kombi", maxWeight: 5000, vehicles: [] },
11    { description: "Caminh o", maxWeight: 12000, vehicles: [] },
12];
13
14 let items = [];
15
16 function controller() {
17     getData();
18     if (items.length === 0) {
19         document.getElementById("result-text").value = "Nenhum item
20             v lido para processar.";
21         return;
22     }
23     items.forEach(distributeItemsBetweenVehicles);
24     showResult();
25     resetArrays();
26 }
27
28 function showResult() {
29     let finalResult = "";
30     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
31         vehicles.length > 0);
32
33     let totalCapacity = 0;
34     let totalWeight = 0;
35
36     const vehicleResults = vehiclesWithItems.map(vehicle => {
37         const weightInVehicle = calculateWeightInVehicle(vehicle);
38         totalCapacity += vehicle.maxWeight * vehicle.vehicles.length;
39         totalWeight += weightInVehicle;
40         return toStringVehicleType(vehicle);
41     });
42 }
```

```

39 });
40
41 finalResult += vehicleResults.join('');
42
43 const totalRemainingSpace = totalCapacity - totalWeight;
44 const totalPercentageLoaded = (totalWeight / totalCapacity * 100)
45     .toFixed(2);
46
47 finalResult += '\nCapacidade total: ${totalCapacity}kg\n';
48 finalResult += 'Peso total: ${totalWeight}kg\n';
49 finalResult += 'Espaço de sobra: ${totalRemainingSpace}kg\n';
50 finalResult += 'Percentual carregado: ${totalPercentageLoaded}%';
51
52 document.getElementById("result-text").value = finalResult;
53 }
54
55 function resetArrays() {
56     vehicles.forEach(vehicle => vehicle.vehicles = []);
57     items = [];
58 }
59
60 function toStringVehicleType(vehicle) {
61     return vehicle.vehicles.map((unit, index) => {
62         return `${vehicle.description} ${index + 1}: ` +
63             unit.map(toString).join('');
64     }).join('');
65 }
66
67 function toString(item) {
68     return `${item.weight}kg ${item.description}\n`;
69 }
70
71 function getData() {
72     const data = document.getElementById("data").value.trim();
73     items = data.split('\n').map(line => {
74         line = line.trim();
75
76         const match = line.match(/^(\d+)\s*kg\s*(.*)$/i);
77         if (!match) return null;
78
79         const weight = parseInt(match[1], 10);
80
81         vehicles = vehicles || [];
82         vehicles.push({
83             description: match[2],
84             vehicles: []
85         });
86
87         items.push({
88             weight: weight,
89             description: match[2]
90         });
91     });
92 }

```

```

78     const description = match[2].trim();
79     return { weight, description };
80   }).filter(item => item !== null);
81 }
82
83 function clearTextArea() {
84   document.getElementById("data").value = "";
85   document.getElementById("result-text").value = "";
86 }
87
88 function distributeItemsBetweenVehicles(item) {
89   for (let vehicle of vehicles) {
90     if (item.weight <= vehicle.maxWeight) {
91       distributeItemInVehicle(vehicle, item);
92       break;
93     }
94   }
95 }
96
97 function distributeItemInVehicle(vehicle, item) {
98   if (vehicle.vehicles.length === 0) {
99     vehicle.vehicles.push([item]);
100    return;
101  }
102
103  const lastVehicle = vehicle.vehicles[vehicle.vehicles.length -
104    1];
105  const currentWeight = lastVehicle.reduce((sum, it) => sum + it.
106    weight, 0);
107
108  if (currentWeight + item.weight <= vehicle.maxWeight) {
109    lastVehicle.push(item);
110  } else {
111    vehicle.vehicles.push([item]);
112  }
113}
114
115 function calculateWeightInVehicle(vehicle) {
116   return vehicle.vehicles.reduce((total, unit) => {
117     return total + unit.reduce((sum, item) => sum + item.weight,
118       0);
119   });
120 }

```

```

116     }, 0);
117 }

```

Listing G.1 – Código-fonte base do Desafio Técnico

```

1 /**
2 * @author Guilherme Fraga
3 * @since 13/09/2025
4 * @version 1.0
5 */
6
7 const vehicles = [
8     { description: "Moto", maxWeight: 45, vehicles: [] },
9     { description: "Van", maxWeight: 3000, vehicles: [] },
10    { description: "Kombi", maxWeight: 5000, vehicles: [] },
11    { description: "Caminhão", maxWeight: 12000, vehicles: [] },
12];
13
14 let items = [];
15
16 function controller() {
17     getData();
18     if (items.length === 0) {
19         document.getElementById("result-text").value = "Nenhum item
20             válido para processar.";
21         return;
22     }
23     items.forEach(distributeItemsBetweenVehicles);
24     showResult();
25     resetArrays();
26 }
27
28 function showResult() {
29     let finalResult = "";
30     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
31         vehicles.length > 0);
32
33     let totalCapacity = 0;
34     let totalWeight = 0;
35
36     const vehicleResults = vehiclesWithItems.map(vehicle => {
37         const weightInVehicle = calculateWeightInVehicle(vehicle);
38
39         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
40         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
41
42         vehicle.vehicles.forEach(availableVehicle => {
43             const availableCapacity = availableVehicle.capacity - weight
44             InVehicle;
45             if (availableCapacity > 0) {
46                 const itemIndex = items.findIndex(item => item.vehicle
47                     .id === availableVehicle.id);
48
49                 if (itemIndex !== -1) {
50                     const item = items[itemIndex];
51
52                     item.vehicle.vehicles.push(availableVehicle);
53                     item.vehicle.weight += item.weight;
54
55                     items.splice(itemIndex, 1);
56
57                     availableVehicle.weight += item.weight;
58                     availableVehicle.capacity -= item.weight;
59
60                     finalResult += `Item ${item.description} adicionado ao ve
61                    ículo ${availableVehicle.description} com peso ${item.w
62                     eight} kg.\n`;
63
64                 }
65             }
66         });
67     });
68
69     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
70     - Total de peso distribuído: ${totalWeight} kg`;
71
72     document.getElementById("result-text").value = finalResult;
73 }
74
75 function calculateWeightInVehicle(vehicle) {
76     let weight = 0;
77
78     vehicle.items.forEach(item => {
79         weight += item.weight;
80     });
81
82     return weight;
83 }
84
85 function resetArrays() {
86     vehicles = [
87         { description: "Moto", maxWeight: 45, vehicles: [] },
88         { description: "Van", maxWeight: 3000, vehicles: [] },
89         { description: "Kombi", maxWeight: 5000, vehicles: [] },
90         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
91     ];
92
93     items = [];
94 }
95
96 function getData() {
97     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
98         .then(response => response.json())
99         .then(data => {
100            vehicles = data.vehicles;
101            items = data.items;
102        })
103        .catch(error => console.error("Erro ao carregar os dados: ", error));
104 }
105
106 function distributeItemsBetweenVehicles(item) {
107     const availableVehicles = vehicles.filter(vehicle => vehicle.
108         vehicles.length < vehicle.maxWeight);
109
110     if (availableVehicles.length === 0) {
111         return;
112     }
113
114     const itemIndex = items.findIndex(item => item.id === item.id);
115
116     if (itemIndex === -1) {
117         return;
118     }
119
120     const item = items[itemIndex];
121
122     const availableVehicle = availableVehicles[Math.floor(Math.random()
123         * availableVehicles.length)];
124
125     item.vehicle.vehicles.push(availableVehicle);
126     item.vehicle.weight += item.weight;
127
128     items.splice(itemIndex, 1);
129
130     availableVehicle.weight += item.weight;
131     availableVehicle.capacity -= item.weight;
132
133     const finalResult = `Item ${item.description} adicionado ao ve
134        ículo ${availableVehicle.description} com peso ${item.w
135         eight} kg.`;
136
137     document.getElementById("result-text").value += finalResult;
138 }
139
140 function showResult() {
141     let finalResult = "";
142
143     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
144         vehicles.length > 0);
145
146     let totalCapacity = 0;
147     let totalWeight = 0;
148
149     const vehicleResults = vehiclesWithItems.map(vehicle => {
150         const weightInVehicle = calculateWeightInVehicle(vehicle);
151
152         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
153         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
154
155         vehicle.vehicles.forEach(availableVehicle => {
156             const availableCapacity = availableVehicle.capacity - weight
157             InVehicle;
158             if (availableCapacity > 0) {
159                 const itemIndex = items.findIndex(item => item.vehicle
160                     .id === availableVehicle.id);
161
162                 if (itemIndex !== -1) {
163                     const item = items[itemIndex];
164
165                     item.vehicle.vehicles.push(availableVehicle);
166                     item.vehicle.weight += item.weight;
167
168                     items.splice(itemIndex, 1);
169
170                     availableVehicle.weight += item.weight;
171                     availableVehicle.capacity -= item.weight;
172
173                     finalResult += `Item ${item.description} adicionado ao ve
174                    ículo ${availableVehicle.description} com peso ${item.w
175                     eight} kg.\n`;
176
177                 }
178             }
179         });
180     });
181
182     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
183     - Total de peso distribuído: ${totalWeight} kg`;
184
185     document.getElementById("result-text").value = finalResult;
186 }
187
188 function calculateWeightInVehicle(vehicle) {
189     let weight = 0;
190
191     vehicle.items.forEach(item => {
192         weight += item.weight;
193     });
194
195     return weight;
196 }
197
198 function resetArrays() {
199     vehicles = [
200         { description: "Moto", maxWeight: 45, vehicles: [] },
201         { description: "Van", maxWeight: 3000, vehicles: [] },
202         { description: "Kombi", maxWeight: 5000, vehicles: [] },
203         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
204     ];
205
206     items = [];
207 }
208
209 function getData() {
210     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
211         .then(response => response.json())
212         .then(data => {
213            vehicles = data.vehicles;
214            items = data.items;
215        })
216         .catch(error => console.error("Erro ao carregar os dados: ", error));
217 }
218
219 function distributeItemsBetweenVehicles(item) {
220     const availableVehicles = vehicles.filter(vehicle => vehicle.
221         vehicles.length < vehicle.maxWeight);
222
223     if (availableVehicles.length === 0) {
224         return;
225     }
226
227     const itemIndex = items.findIndex(item => item.id === item.id);
228
229     if (itemIndex === -1) {
230         return;
231     }
232
233     const item = items[itemIndex];
234
235     const availableVehicle = availableVehicles[Math.floor(Math.random()
236         * availableVehicles.length)];
237
238     item.vehicle.vehicles.push(availableVehicle);
239     item.vehicle.weight += item.weight;
240
241     items.splice(itemIndex, 1);
242
243     availableVehicle.weight += item.weight;
244     availableVehicle.capacity -= item.weight;
245
246     const finalResult = `Item ${item.description} adicionado ao ve
247        ículo ${availableVehicle.description} com peso ${item.w
248         eight} kg.`;
249
250     document.getElementById("result-text").value += finalResult;
251 }
252
253 function showResult() {
254     let finalResult = "";
255
256     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
257         vehicles.length > 0);
258
259     let totalCapacity = 0;
260     let totalWeight = 0;
261
262     const vehicleResults = vehiclesWithItems.map(vehicle => {
263         const weightInVehicle = calculateWeightInVehicle(vehicle);
264
265         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
266         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
267
268         vehicle.vehicles.forEach(availableVehicle => {
269             const availableCapacity = availableVehicle.capacity - weight
270             InVehicle;
271             if (availableCapacity > 0) {
272                 const itemIndex = items.findIndex(item => item.vehicle
273                     .id === availableVehicle.id);
274
275                 if (itemIndex !== -1) {
276                     const item = items[itemIndex];
277
278                     item.vehicle.vehicles.push(availableVehicle);
279                     item.vehicle.weight += item.weight;
280
281                     items.splice(itemIndex, 1);
282
283                     availableVehicle.weight += item.weight;
284                     availableVehicle.capacity -= item.weight;
285
286                     finalResult += `Item ${item.description} adicionado ao ve
287                    ículo ${availableVehicle.description} com peso ${item.w
288                     eight} kg.\n`;
289
290                 }
291             }
292         });
293     });
294
295     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
296     - Total de peso distribuído: ${totalWeight} kg`;
297
298     document.getElementById("result-text").value = finalResult;
299 }
300
301 function calculateWeightInVehicle(vehicle) {
302     let weight = 0;
303
304     vehicle.items.forEach(item => {
305         weight += item.weight;
306     });
307
308     return weight;
309 }
310
311 function resetArrays() {
312     vehicles = [
313         { description: "Moto", maxWeight: 45, vehicles: [] },
314         { description: "Van", maxWeight: 3000, vehicles: [] },
315         { description: "Kombi", maxWeight: 5000, vehicles: [] },
316         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
317     ];
318
319     items = [];
320 }
321
322 function getData() {
323     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
324         .then(response => response.json())
325         .then(data => {
326            vehicles = data.vehicles;
327            items = data.items;
328        })
329         .catch(error => console.error("Erro ao carregar os dados: ", error));
330 }
331
332 function distributeItemsBetweenVehicles(item) {
333     const availableVehicles = vehicles.filter(vehicle => vehicle.
334         vehicles.length < vehicle.maxWeight);
335
336     if (availableVehicles.length === 0) {
337         return;
338     }
339
340     const itemIndex = items.findIndex(item => item.id === item.id);
341
342     if (itemIndex === -1) {
343         return;
344     }
345
346     const item = items[itemIndex];
347
348     const availableVehicle = availableVehicles[Math.floor(Math.random()
349         * availableVehicles.length)];
350
351     item.vehicle.vehicles.push(availableVehicle);
352     item.vehicle.weight += item.weight;
353
354     items.splice(itemIndex, 1);
355
356     availableVehicle.weight += item.weight;
357     availableVehicle.capacity -= item.weight;
358
359     const finalResult = `Item ${item.description} adicionado ao ve
360        ículo ${availableVehicle.description} com peso ${item.w
361         eight} kg.`;
362
363     document.getElementById("result-text").value += finalResult;
364 }
365
366 function showResult() {
367     let finalResult = "";
368
369     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
370         vehicles.length > 0);
371
372     let totalCapacity = 0;
373     let totalWeight = 0;
374
375     const vehicleResults = vehiclesWithItems.map(vehicle => {
376         const weightInVehicle = calculateWeightInVehicle(vehicle);
377
378         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
379         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
380
381         vehicle.vehicles.forEach(availableVehicle => {
382             const availableCapacity = availableVehicle.capacity - weight
383             InVehicle;
384             if (availableCapacity > 0) {
385                 const itemIndex = items.findIndex(item => item.vehicle
386                     .id === availableVehicle.id);
387
388                 if (itemIndex !== -1) {
389                     const item = items[itemIndex];
390
391                     item.vehicle.vehicles.push(availableVehicle);
392                     item.vehicle.weight += item.weight;
393
394                     items.splice(itemIndex, 1);
395
396                     availableVehicle.weight += item.weight;
397                     availableVehicle.capacity -= item.weight;
398
399                     finalResult += `Item ${item.description} adicionado ao ve
400                    ículo ${availableVehicle.description} com peso ${item.w
401                     eight} kg.\n`;
402
403                 }
404             }
405         });
406     });
407
408     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
409     - Total de peso distribuído: ${totalWeight} kg`;
410
411     document.getElementById("result-text").value = finalResult;
412 }
413
414 function calculateWeightInVehicle(vehicle) {
415     let weight = 0;
416
417     vehicle.items.forEach(item => {
418         weight += item.weight;
419     });
420
421     return weight;
422 }
423
424 function resetArrays() {
425     vehicles = [
426         { description: "Moto", maxWeight: 45, vehicles: [] },
427         { description: "Van", maxWeight: 3000, vehicles: [] },
428         { description: "Kombi", maxWeight: 5000, vehicles: [] },
429         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
430     ];
431
432     items = [];
433 }
434
435 function getData() {
436     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
437         .then(response => response.json())
438         .then(data => {
439            vehicles = data.vehicles;
440            items = data.items;
441        })
442         .catch(error => console.error("Erro ao carregar os dados: ", error));
443 }
444
445 function distributeItemsBetweenVehicles(item) {
446     const availableVehicles = vehicles.filter(vehicle => vehicle.
447         vehicles.length < vehicle.maxWeight);
448
449     if (availableVehicles.length === 0) {
450         return;
451     }
452
453     const itemIndex = items.findIndex(item => item.id === item.id);
454
455     if (itemIndex === -1) {
456         return;
457     }
458
459     const item = items[itemIndex];
460
461     const availableVehicle = availableVehicles[Math.floor(Math.random()
462         * availableVehicles.length)];
463
464     item.vehicle.vehicles.push(availableVehicle);
465     item.vehicle.weight += item.weight;
466
467     items.splice(itemIndex, 1);
468
469     availableVehicle.weight += item.weight;
470     availableVehicle.capacity -= item.weight;
471
472     const finalResult = `Item ${item.description} adicionado ao ve
473        ículo ${availableVehicle.description} com peso ${item.w
474         eight} kg.`;
475
476     document.getElementById("result-text").value += finalResult;
477 }
478
479 function showResult() {
480     let finalResult = "";
481
482     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
483         vehicles.length > 0);
484
485     let totalCapacity = 0;
486     let totalWeight = 0;
487
488     const vehicleResults = vehiclesWithItems.map(vehicle => {
489         const weightInVehicle = calculateWeightInVehicle(vehicle);
490
491         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
492         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
493
494         vehicle.vehicles.forEach(availableVehicle => {
495             const availableCapacity = availableVehicle.capacity - weight
496             InVehicle;
497             if (availableCapacity > 0) {
498                 const itemIndex = items.findIndex(item => item.vehicle
499                     .id === availableVehicle.id);
500
501                 if (itemIndex !== -1) {
502                     const item = items[itemIndex];
503
504                     item.vehicle.vehicles.push(availableVehicle);
505                     item.vehicle.weight += item.weight;
506
507                     items.splice(itemIndex, 1);
508
509                     availableVehicle.weight += item.weight;
510                     availableVehicle.capacity -= item.weight;
511
512                     finalResult += `Item ${item.description} adicionado ao ve
513                    ículo ${availableVehicle.description} com peso ${item.w
514                     eight} kg.\n`;
515
516                 }
517             }
518         });
519     });
520
521     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
522     - Total de peso distribuído: ${totalWeight} kg`;
523
524     document.getElementById("result-text").value = finalResult;
525 }
526
527 function calculateWeightInVehicle(vehicle) {
528     let weight = 0;
529
530     vehicle.items.forEach(item => {
531         weight += item.weight;
532     });
533
534     return weight;
535 }
536
537 function resetArrays() {
538     vehicles = [
539         { description: "Moto", maxWeight: 45, vehicles: [] },
540         { description: "Van", maxWeight: 3000, vehicles: [] },
541         { description: "Kombi", maxWeight: 5000, vehicles: [] },
542         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
543     ];
544
545     items = [];
546 }
547
548 function getData() {
549     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
550         .then(response => response.json())
551         .then(data => {
552            vehicles = data.vehicles;
553            items = data.items;
554        })
555         .catch(error => console.error("Erro ao carregar os dados: ", error));
556 }
557
558 function distributeItemsBetweenVehicles(item) {
559     const availableVehicles = vehicles.filter(vehicle => vehicle.
560         vehicles.length < vehicle.maxWeight);
561
562     if (availableVehicles.length === 0) {
563         return;
564     }
565
566     const itemIndex = items.findIndex(item => item.id === item.id);
567
568     if (itemIndex === -1) {
569         return;
570     }
571
572     const item = items[itemIndex];
573
574     const availableVehicle = availableVehicles[Math.floor(Math.random()
575         * availableVehicles.length)];
576
577     item.vehicle.vehicles.push(availableVehicle);
578     item.vehicle.weight += item.weight;
579
580     items.splice(itemIndex, 1);
581
582     availableVehicle.weight += item.weight;
583     availableVehicle.capacity -= item.weight;
584
585     const finalResult = `Item ${item.description} adicionado ao ve
586        ículo ${availableVehicle.description} com peso ${item.w
587         eight} kg.`;
588
589     document.getElementById("result-text").value += finalResult;
590 }
591
592 function showResult() {
593     let finalResult = "";
594
595     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
596         vehicles.length > 0);
597
598     let totalCapacity = 0;
599     let totalWeight = 0;
600
601     const vehicleResults = vehiclesWithItems.map(vehicle => {
602         const weightInVehicle = calculateWeightInVehicle(vehicle);
603
604         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
605         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
606
607         vehicle.vehicles.forEach(availableVehicle => {
608             const availableCapacity = availableVehicle.capacity - weight
609             InVehicle;
610             if (availableCapacity > 0) {
611                 const itemIndex = items.findIndex(item => item.vehicle
612                     .id === availableVehicle.id);
613
614                 if (itemIndex !== -1) {
615                     const item = items[itemIndex];
616
617                     item.vehicle.vehicles.push(availableVehicle);
618                     item.vehicle.weight += item.weight;
619
620                     items.splice(itemIndex, 1);
621
622                     availableVehicle.weight += item.weight;
623                     availableVehicle.capacity -= item.weight;
624
625                     finalResult += `Item ${item.description} adicionado ao ve
626                    ículo ${availableVehicle.description} com peso ${item.w
627                     eight} kg.\n`;
628
629                 }
630             }
631         });
632     });
633
634     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
635     - Total de peso distribuído: ${totalWeight} kg`;
636
637     document.getElementById("result-text").value = finalResult;
638 }
639
640 function calculateWeightInVehicle(vehicle) {
641     let weight = 0;
642
643     vehicle.items.forEach(item => {
644         weight += item.weight;
645     });
646
647     return weight;
648 }
649
650 function resetArrays() {
651     vehicles = [
652         { description: "Moto", maxWeight: 45, vehicles: [] },
653         { description: "Van", maxWeight: 3000, vehicles: [] },
654         { description: "Kombi", maxWeight: 5000, vehicles: [] },
655         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
656     ];
657
658     items = [];
659 }
660
661 function getData() {
662     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
663         .then(response => response.json())
664         .then(data => {
665            vehicles = data.vehicles;
666            items = data.items;
667        })
668         .catch(error => console.error("Erro ao carregar os dados: ", error));
669 }
670
671 function distributeItemsBetweenVehicles(item) {
672     const availableVehicles = vehicles.filter(vehicle => vehicle.
673         vehicles.length < vehicle.maxWeight);
674
675     if (availableVehicles.length === 0) {
676         return;
677     }
678
679     const itemIndex = items.findIndex(item => item.id === item.id);
680
681     if (itemIndex === -1) {
682         return;
683     }
684
685     const item = items[itemIndex];
686
687     const availableVehicle = availableVehicles[Math.floor(Math.random()
688         * availableVehicles.length)];
689
690     item.vehicle.vehicles.push(availableVehicle);
691     item.vehicle.weight += item.weight;
692
693     items.splice(itemIndex, 1);
694
695     availableVehicle.weight += item.weight;
696     availableVehicle.capacity -= item.weight;
697
698     const finalResult = `Item ${item.description} adicionado ao ve
699        ículo ${availableVehicle.description} com peso ${item.w
700         eight} kg.`;
701
702     document.getElementById("result-text").value += finalResult;
703 }
704
705 function showResult() {
706     let finalResult = "";
707
708     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
709         vehicles.length > 0);
710
711     let totalCapacity = 0;
712     let totalWeight = 0;
713
714     const vehicleResults = vehiclesWithItems.map(vehicle => {
715         const weightInVehicle = calculateWeightInVehicle(vehicle);
716
717         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
718         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
719
720         vehicle.vehicles.forEach(availableVehicle => {
721             const availableCapacity = availableVehicle.capacity - weight
722             InVehicle;
723             if (availableCapacity > 0) {
724                 const itemIndex = items.findIndex(item => item.vehicle
725                     .id === availableVehicle.id);
726
727                 if (itemIndex !== -1) {
728                     const item = items[itemIndex];
729
730                     item.vehicle.vehicles.push(availableVehicle);
731                     item.vehicle.weight += item.weight;
732
733                     items.splice(itemIndex, 1);
734
735                     availableVehicle.weight += item.weight;
736                     availableVehicle.capacity -= item.weight;
737
738                     finalResult += `Item ${item.description} adicionado ao ve
739                    ículo ${availableVehicle.description} com peso ${item.w
740                     eight} kg.\n`;
741
742                 }
743             }
744         });
745     });
746
747     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
748     - Total de peso distribuído: ${totalWeight} kg`;
749
750     document.getElementById("result-text").value = finalResult;
751 }
752
753 function calculateWeightInVehicle(vehicle) {
754     let weight = 0;
755
756     vehicle.items.forEach(item => {
757         weight += item.weight;
758     });
759
760     return weight;
761 }
762
763 function resetArrays() {
764     vehicles = [
765         { description: "Moto", maxWeight: 45, vehicles: [] },
766         { description: "Van", maxWeight: 3000, vehicles: [] },
767         { description: "Kombi", maxWeight: 5000, vehicles: [] },
768         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
769     ];
770
771     items = [];
772 }
773
774 function getData() {
775     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
776         .then(response => response.json())
777         .then(data => {
778            vehicles = data.vehicles;
779            items = data.items;
780        })
781         .catch(error => console.error("Erro ao carregar os dados: ", error));
782 }
783
784 function distributeItemsBetweenVehicles(item) {
785     const availableVehicles = vehicles.filter(vehicle => vehicle.
786         vehicles.length < vehicle.maxWeight);
787
788     if (availableVehicles.length === 0) {
789         return;
790     }
791
792     const itemIndex = items.findIndex(item => item.id === item.id);
793
794     if (itemIndex === -1) {
795         return;
796     }
797
798     const item = items[itemIndex];
799
800     const availableVehicle = availableVehicles[Math.floor(Math.random()
801         * availableVehicles.length)];
802
803     item.vehicle.vehicles.push(availableVehicle);
804     item.vehicle.weight += item.weight;
805
806     items.splice(itemIndex, 1);
807
808     availableVehicle.weight += item.weight;
809     availableVehicle.capacity -= item.weight;
810
811     const finalResult = `Item ${item.description} adicionado ao ve
812        ículo ${availableVehicle.description} com peso ${item.w
813         eight} kg.`;
814
815     document.getElementById("result-text").value += finalResult;
816 }
817
818 function showResult() {
819     let finalResult = "";
820
821     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
822         vehicles.length > 0);
823
824     let totalCapacity = 0;
825     let totalWeight = 0;
826
827     const vehicleResults = vehiclesWithItems.map(vehicle => {
828         const weightInVehicle = calculateWeightInVehicle(vehicle);
829
830         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
831         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
832
833         vehicle.vehicles.forEach(availableVehicle => {
834             const availableCapacity = availableVehicle.capacity - weight
835             InVehicle;
836             if (availableCapacity > 0) {
837                 const itemIndex = items.findIndex(item => item.vehicle
838                     .id === availableVehicle.id);
839
840                 if (itemIndex !== -1) {
841                     const item = items[itemIndex];
842
843                     item.vehicle.vehicles.push(availableVehicle);
844                     item.vehicle.weight += item.weight;
845
846                     items.splice(itemIndex, 1);
847
848                     availableVehicle.weight += item.weight;
849                     availableVehicle.capacity -= item.weight;
850
851                     finalResult += `Item ${item.description} adicionado ao ve
852                    ículo ${availableVehicle.description} com peso ${item.w
853                     eight} kg.\n`;
854
855                 }
856             }
857         });
858     });
859
860     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
861     - Total de peso distribuído: ${totalWeight} kg`;
862
863     document.getElementById("result-text").value = finalResult;
864 }
865
866 function calculateWeightInVehicle(vehicle) {
867     let weight = 0;
868
869     vehicle.items.forEach(item => {
870         weight += item.weight;
871     });
872
873     return weight;
874 }
875
876 function resetArrays() {
877     vehicles = [
878         { description: "Moto", maxWeight: 45, vehicles: [] },
879         { description: "Van", maxWeight: 3000, vehicles: [] },
880         { description: "Kombi", maxWeight: 5000, vehicles: [] },
881         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
882     ];
883
884     items = [];
885 }
886
887 function getData() {
888     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
889         .then(response => response.json())
890         .then(data => {
891            vehicles = data.vehicles;
892            items = data.items;
893        })
894         .catch(error => console.error("Erro ao carregar os dados: ", error));
895 }
896
897 function distributeItemsBetweenVehicles(item) {
898     const availableVehicles = vehicles.filter(vehicle => vehicle.
899         vehicles.length < vehicle.maxWeight);
900
901     if (availableVehicles.length === 0) {
902         return;
903     }
904
905     const itemIndex = items.findIndex(item => item.id === item.id);
906
907     if (itemIndex === -1) {
908         return;
909     }
910
911     const item = items[itemIndex];
912
913     const availableVehicle = availableVehicles[Math.floor(Math.random()
914         * availableVehicles.length)];
915
916     item.vehicle.vehicles.push(availableVehicle);
917     item.vehicle.weight += item.weight;
918
919     items.splice(itemIndex, 1);
920
921     availableVehicle.weight += item.weight;
922     availableVehicle.capacity -= item.weight;
923
924     const finalResult = `Item ${item.description} adicionado ao ve
925        ículo ${availableVehicle.description} com peso ${item.w
926         eight} kg.`;
927
928     document.getElementById("result-text").value += finalResult;
929 }
930
931 function showResult() {
932     let finalResult = "";
933
934     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
935         vehicles.length > 0);
936
937     let totalCapacity = 0;
938     let totalWeight = 0;
939
940     const vehicleResults = vehiclesWithItems.map(vehicle => {
941         const weightInVehicle = calculateWeightInVehicle(vehicle);
942
943         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
944         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
945
946         vehicle.vehicles.forEach(availableVehicle => {
947             const availableCapacity = availableVehicle.capacity - weight
948             InVehicle;
949             if (availableCapacity > 0) {
950                 const itemIndex = items.findIndex(item => item.vehicle
951                     .id === availableVehicle.id);
952
953                 if (itemIndex !== -1) {
954                     const item = items[itemIndex];
955
956                     item.vehicle.vehicles.push(availableVehicle);
957                     item.vehicle.weight += item.weight;
958
959                     items.splice(itemIndex, 1);
960
961                     availableVehicle.weight += item.weight;
962                     availableVehicle.capacity -= item.weight;
963
964                     finalResult += `Item ${item.description} adicionado ao ve
965                    ículo ${availableVehicle.description} com peso ${item.w
966                     eight} kg.\n`;
967
968                 }
969             }
970         });
971     });
972
973     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
974     - Total de peso distribuído: ${totalWeight} kg`;
975
976     document.getElementById("result-text").value = finalResult;
977 }
978
979 function calculateWeightInVehicle(vehicle) {
980     let weight = 0;
981
982     vehicle.items.forEach(item => {
983         weight += item.weight;
984     });
985
986     return weight;
987 }
988
989 function resetArrays() {
990     vehicles = [
991         { description: "Moto", maxWeight: 45, vehicles: [] },
992         { description: "Van", maxWeight: 3000, vehicles: [] },
993         { description: "Kombi", maxWeight: 5000, vehicles: [] },
994         { description: "Caminhão", maxWeight: 12000, vehicles: [] }
995     ];
996
997     items = [];
998 }
999
1000 function getData() {
1001     fetch("https://api.jsonbin.org/b/645f3a2a5a1a1a001f5a5a5a")
1002         .then(response => response.json())
1003         .then(data => {
1004            vehicles = data.vehicles;
1005            items = data.items;
1006        })
1007         .catch(error => console.error("Erro ao carregar os dados: ", error));
1008 }
1009
1010 function distributeItemsBetweenVehicles(item) {
1011     const availableVehicles = vehicles.filter(vehicle => vehicle.
1012         vehicles.length < vehicle.maxWeight);
1013
1014     if (availableVehicles.length === 0) {
1015         return;
1016     }
1017
1018     const itemIndex = items.findIndex(item => item.id === item.id);
1019
1020     if (itemIndex === -1) {
1021         return;
1022     }
1023
1024     const item = items[itemIndex];
1025
1026     const availableVehicle = availableVehicles[Math.floor(Math.random()
1027         * availableVehicles.length)];
1028
1029     item.vehicle.vehicles.push(availableVehicle);
1030     item.vehicle.weight += item.weight;
1031
1032     items.splice(itemIndex, 1);
1033
1034     availableVehicle.weight += item.weight;
1035     availableVehicle.capacity -= item.weight;
1036
1037     const finalResult = `Item ${item.description} adicionado ao ve
1038        ículo ${availableVehicle.description} com peso ${item.w
1039         eight} kg.`;
1040
1041     document.getElementById("result-text").value += finalResult;
1042 }
1043
1044 function showResult() {
1045     let finalResult = "";
1046
1047     const vehiclesWithItems = vehicles.filter(vehicle => vehicle.
1048         vehicles.length > 0);
1049
1050     let totalCapacity = 0;
1051     let totalWeight = 0;
1052
1053     const vehicleResults = vehiclesWithItems.map(vehicle => {
1054         const weightInVehicle = calculateWeightInVehicle(vehicle);
1055
1056         finalResult += `Veículo: ${vehicle.description} - Capacidade: ${ve
1057         hicle.capacity} - Peso Total: ${weightInVehicle} kg\n`;
1058
1059         vehicle.vehicles.forEach(availableVehicle => {
1060             const availableCapacity = availableVehicle.capacity - weight
1061             InVehicle;
1062             if (availableCapacity > 0) {
1063                 const itemIndex = items.findIndex(item => item.vehicle
1064                     .id === availableVehicle.id);
1065
1066                 if (itemIndex !== -1) {
1067                     const item = items[itemIndex];
1068
1069                     item.vehicle.vehicles.push(availableVehicle);
1070                     item.vehicle.weight += item.weight;
1071
1072                     items.splice(itemIndex, 1);
1073
1074                     availableVehicle.weight += item.weight;
1075                     availableVehicle.capacity -= item.weight;
1076
1077                     finalResult += `Item ${item.description} adicionado ao ve
1078                    ículo ${availableVehicle.description} com peso ${item.w
1079                     eight} kg.\n`;
1080
1081                 }
1082             }
1083         });
1084     });
1085
1086     finalResult += `Total de capacidade disponível: ${totalCapacity} kg
1087     - Total de peso distribuído: ${totalWeight} kg`;
1088
1089     document.getElementById("result-text").value = finalResult;
1090 }
1091
1092 function calculateWeightInVehicle(vehicle) {
1093     let weight = 0;
1094
1095     vehicle.items.forEach(item => {
1096         weight += item.weight;
1097     });
1098
1099     return weight;
1100 }
1101
1102 function resetArrays() {
1103     vehicles = [
1104         { description: "Moto", maxWeight: 45, vehicles: [] },
1105         { description: "Van", maxWeight: 3000, vehicles:
```

```

36     totalCapacity += vehicle.maxWeight;
37     totalWeight += weightInVehicle;
38     return toStringVehicleType(vehicle);
39   );
40
41 finalResult += vehicleResults.join(' ');
42
43 const totalRemainingSpace = totalWeight - totalCapacity;
44 const totalPercentageLoaded = (totalWeight / (totalCapacity || 1)
45   * 100).toFixed(2);
46
47 finalResult += '\nCapacidade total: ${totalCapacity}kg\n';
48 finalResult += 'Peso total: ${totalWeight}kg\n';
49 finalResult += 'Espaço de sobra: ${totalRemainingSpace}kg\n';
50 finalResult += 'Percentual carregado: ${totalPercentageLoaded}%';
51
52 document.getElementById("result-text").value = finalResult;
53
54 function resetArrays() {
55   vehicles.forEach(vehicle => {
56     // vehicle.vehicles = [];
57   });
58   items = [];
59 }
60
61 function toStringVehicleType(vehicle) {
62   return vehicle.vehicles.map((unit, index) => {
63     if (!Array.isArray(unit)) unit = [unit];
64     return `${vehicle.description} ${index}: \n${unit.map(
65       toString).join('')} `;
66   }).join('');
67 }
68
69 function toString(item) {
70   return `${item.weight}kg ${item.description}`;
71 }
72
73 function getData() {
74   const data = document.getElementById("data").value.trim();
75   items = data.split('\n').map(line => {

```

```

75   line = line.trim();
76   if (!line) return null;
77
78   const match = line.match(/^(\d+)\s*kg(.*?)$/i);
79   if (!match) return null;
80
81   const weight = parseInt(match[1], 10);
82   const description = match[2].trim();
83   return { weight, description };
84 }).filter(item => item !== null);
85 }
86
87 function clearTextArea() {
88   document.getElementById("data").value = "";
89   document.getElementById("result-text").value = "";
90 }
91
92 function distributeItemsBetweenVehicles(item) {
93   for (let vehicle of vehicles) {
94     if (item.weight < vehicle.maxWeight) {
95       distributeItemInVehicle(vehicle, item);
96       break;
97     }
98   }
99 }
100
101 function distributeItemInVehicle(vehicle, item) {
102   if (vehicle.vehicles.length === 0) {
103     vehicle.vehicles.push(item);
104     return;
105   }
106
107   const lastVehicle = vehicle.vehicles[vehicle.vehicles.length -
108     1];
109   const currentWeight = lastVehicle.reduce((sum, it) => sum + it.
110     weight, 0);
111
112   if (currentWeight + item.weight <= vehicle.maxWeight) {
113     lastVehicle.push(item);
114   } else {
115     vehicle.vehicles.push([item]);
116   }
117 }

```

```
114     }
115 }
116
117 function calculateWeightInVehicle(vehicle) {
118     return vehicle.vehicles.reduce((_, unit) => {
119         if (!Array.isArray(unit)) unit = [unit];
120         return unit.reduce((sum, item) => sum + item.weight, 0);
121     }, 0);
122 }
```

Listing G.2 – Código-fonte do Desafio Técnico

APÊNDICE H – LISTA DE BUGS E ANÁLISES DOS DESENVOLVEDORES

Figura H.1 – Lista de bugs implementados no arquivo main.js

Bug	Local	Nível	Impacto
Index começa em 0, \${index} ao invés de \${index+1}	toStringVehicleType	Fácil	Numeração das unidades confusa: "Moto 0:"
toString sem \n	toString	Fácil	Itens aparecem grudados na mesma linha
Number sem trim/regex não usado	getData()	Fácil	Entradas com formatos diferentes podem não ser lidas
Substituído total + (...) por _ + (...)	calculateWeightInVehicle	Fácil	Ignora acumulador total, peso total exibido incorreto
totalCapacity não multiplica unidades, totalCapacity += vehicle.maxWeight	showResult()	Médio	Capacidade total menor que o real
Espaço restante invertido, totalRemainingSpace = totalWeight - totalCapacity	showResult()	Médio	Espaço de sobra negativo
< ao invés de <	distributedItemsBetweenVehicles	Médio	Itens com peso exato não entram na distribuição
reset comentado	resetArrays()	Difícil	Itens antigos permanecem entre execuções
unit não é array	toStringVehicleType	Difícil	Quebra map/reduce, trava execução
push direto, vehicle.vehicles.push(item)	distributeItemInVehicle	Difícil	Estrutura da unidade quebrada, efeitos visíveis confusos

Fonte: Elaborado pelo autor

Figura H.2 – Análises dos desenvolvedores do grupo sem IA

Bug	Local	Nível	Impacto	Grupo sem IA		
				Desenvolvedor 1 - Senior Frontend Engineer (45m 42s)	Desenvolvedor 2 - Desenvolvedor front-end (39m 47s)	Desenvolvedor 3 - Back End Developer (1h 16m 43s)
Index começa em 0, \${index} ao invés de \${index+1}	toStringVehicleType	Fácil	Numeração das unidades confusa: "Moto 0:"	Concluído	Concluído	Concluído
toString sem \n	toString	Fácil	Itens aparecem grudados na mesma linha	Concluído parcialmente	Concluído	Concluído
Number sem trim/regex não usado	getData()	Fácil	Entradas com formatos diferentes podem não ser lidas	Concluído	90,00%	Concluído
Substituído total + (...) por _ + (...)	calculateWeightInVehicle	Fácil	Ignora acumulador total, peso total exibido incorreto	Concluído	Concluído	Concluído
totalCapacity não multiplica unidades, totalCapacity += vehicle.maxWeight	showResult()	Médio	Capacidade total menor que o real	Concluído	Concluído parcialmente	Concluído
Espaço restante invertido, totalRemainingSpace = totalWeight - totalCapacity	showResult()	Médio	Espaço de sobra negativo	Concluído	10,00%	Concluído
< ao invés de <	distributedItemsBetweenVehicles	Médio	Itens com peso exato não entram na distribuição	Concluído	Não concluído	Concluído
reset comentado	resetArrays()	Difícil	Itens antigos permanecem entre execuções	Concluído	Não concluído	Não concluído
unit não é array	toStringVehicleType	Difícil	Quebra map/reduce, trava execução	Concluído	0,00%	Concluído
push direto, vehicle.vehicles.push(item)	distributeItemInVehicle	Difícil	Estrutura da unidade quebrada, efeitos visíveis confusos	Concluído	0,00%	Concluído

Fonte: Elaborado pelo autor

Figura H.3 – Análises dos desenvolvedores do grupo com IA

Bug	Local	Nível	Impacto	Grupo com IA		
				Desenvolvedor 4 - Backend Developer (11m 5s)	Desenvolvedor 5 - Software QA Engineer (14m 31s)	Desenvolvedor 6 - Backend developer (44m 15s)
Index começa em 0, \${index} ao invés de \${index+1}	toStringVehicleType	Fácil	Numeração das unidades confusa: "Moto 0:"	Concluído	Concluído	Concluído
toString sem \n	toString	Fácil	Itens aparecem grudados na mesma linha	Concluído parcialmente	Concluído	Concluído
Number sem trim/regex não usado	getData()	Fácil	Entradas com formatos diferentes podem não ser lidas	Concluído	90,00%	Concluído
Substituído total + (...) por _ + (...)	calculateWeightInVehicle	Fácil	Ignora acumulador total, peso total exibido incorreto	Concluído	Concluído	Concluído
totalCapacity não multiplica unidades, totalCapacity += vehicle.maxWeight	showResult()	Médio	Capacidade total menor que o real	Concluído	Concluído parcialmente	Concluído
Espaço restante invertido, totalRemainingSpace = totalWeight - totalCapacity	showResult()	Médio	Espaço de sobra negativo	Concluído	10,00%	Não concluído
< ao invés de <	distributedItemsBetweenVehicles	Médio	Itens com peso exato não entram na distribuição	Concluído	Não concluído	Concluído
reset comentado	resetArrays()	Difícil	Itens antigos permanecem entre execuções	Concluído	Não concluído	Concluído
unit não é array	toStringVehicleType	Difícil	Quebra map/reduce, trava execução	Concluído	0,00%	Concluído
push direto, vehicle.vehicles.push(item)	distributeItemInVehicle	Difícil	Estrutura da unidade quebrada, efeitos visíveis confusos	Concluído	20,00%	Não concluído

Fonte: Elaborado pelo autor