UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

JOANA STHEFANNY GOMES COSTA DOS SANTOS

ENGENHARIA DE REQUISITOS EM APLICAÇÕES WEB: PERSPECTIVAS DE DESENVOLVEDORES E ANALISTAS

SOBRAL

2025

JOANA STHEFANNY GOMES COSTA DOS SANTOS

ENGENHARIA DE REQUISITOS EM APLICAÇÕES WEB: PERSPECTIVAS DE DESENVOLVEDORES E ANALISTAS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia da Computação, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Evilasio Costa Junior

Coorientador: Prof. Dr. Fischer Jônatas Ferreira

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

D762e dos Santos, Joana Sthefanny Gomes Costa.
ENGENHARIA DE REQUISITOS EM APLICAÇÕES WEB: PERSPECTIVAS DE DESENVOLVEDORES E ANALISTAS / Joana Sthefanny Gomes Costa dos Santos. – 2025.
109 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Prof. Dr. Evilasio Costa Junior.
Coorientação: Prof. Dr. Fischer Jônatas Ferreira .

1. Levantamento de Requisitos. 2. Aplicações Web. 3. Experiência do desenvolvedor. 4. Tecnologias Web. 5. Boas Práticas. I. Título.

CDD 621.39

JOANA STHEFANNY GOMES COSTA DOS SANTOS

ENGENHARIA DE REQUISITOS EM APLICAÇÕES WEB: PERSPECTIVAS DE DESENVOLVEDORES E ANALISTAS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia da Computação, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 06/08/2025

BANCA EXAMINADORA

Prof. Dr. Evilasio Costa Junior (Orientador)
Universidade Federal do Ceará (UFC) - Campus de Sobral

Prof. Dr. Fischer Jônatas Ferreira (Coorientador)
Universidade Federal de Itajubá, Campus de Itabira

Prof. Dr. Iális Cavalcante de Paula Júnior
Universidade Federal do Ceará (UFC) - Campus de Sobral

Profa. Dra. Rainara Maia Carvalho
Universidade Federal do Ceará (UFC) - Campus de Quixadá

AGRADECIMENTOS

A jornada até aqui não teria sido possível sem o apoio de pessoas que caminharam ao meu lado, oferecendo amor, incentivo, amizade e conhecimento. A cada uma delas, minha mais sincera gratidão.

À minha mãe, Joelda Costa, por ser meu alicerce e acreditar em mim mesmo nos momentos mais difíceis. Seu apoio incondicional e sua força me inspiraram a seguir em frente.

Aos meus irmãos: Felipe Estevam, exemplo de força e coragem, que me inspira diariamente com sua resiliência e generosidade. Midiã Sthefanny, por sua amizade que vai além dos laços de sangue. Obrigada pelo companheirismo ao longo da vida e por nunca deixarem de acreditar em mim. E, com carinho especial, Matheus Desaulo, pelas risadas e por ter feito parte da minha vida. Matheus, mesmo não estando mais entre nós, permanece vivo em minhas lembranças e em meu coração. Sua presença foi essencial, e sua ausência é sentida todos os dias.

Ao meu pai, Expedito Júnior, que também partiu, mas continua fazendo parte da minha história. Deixo minha gratidão por ter acreditado em mim e por todos os seus ensinamentos que carrego ao longo da vida.

À minha avó, Maria de Lourdes, às minhas tias, Elzilene Costa, Benedita Marta e Elda Maria, e aos meus tios Benedito Zelson e Elson Costa, que me deram o suporte necessário para que eu pudesse continuar e concluir o curso. Obrigada por me acolherem e me proporcionarem a estabilidade de que precisei para seguir com meus estudos.

Ao meu primo Daniel Moisés, por ter me dado alegria ao longo desse percurso. Acompanhar seu crescimento enquanto eu encerrava esse ciclo foi especial pra mim. Espero que, um dia, isso aqui também sirva de inspiração pra você.

Aos meus colegas e amigos — Nájala Kelly, Vinícius Tabosa, Jéssica Marques, Vitória Freitas, Íris Costa, Wesley Fernandes, Yasmin Emily e Raquel Silveira — agradeço pelo companheirismo, pelas conversas, pelas risadas e por me ajudarem a atravessar os desafios da graduação. Ter vocês por perto foi uma das maiores sortes que eu tive nesta jornada.

Ao meu orientador, professor Evilásio Costa Junior, e ao meu coorientador, professor Fischer Jonatas Ferreira, agradeço imensamente pela paciência, pelas orientações valiosas e por compartilharem comigo seus conhecimentos com tanta generosidade. Obrigada por acreditarem no potencial desta pesquisa.

A todos vocês, meu sincero obrigada. Este trabalho é fruto de muitos esforços, mas também de muito amor e apoio.

RESUMO

Este trabalho tem como objetivo analisar como ocorre o levantamento e a aplicação de requisitos em projetos de desenvolvimento de aplicações Web, a partir da perspectiva de desenvolvedores e analistas. A pesquisa inicia-se com uma revisão teórica sobre Engenharia de Requisitos no contexto Web, considerando aspectos como responsividade, integração com APIs, segurança, Search *Engine Optimization (SEO)* e impacto na escolha de tecnologias. Em seguida, foram realizadas entrevistas com 19 profissionais atuantes na área, nos meses de junho e julho de 2025, abordando práticas, desafios, estratégias e aprendizados no processo de elicitação, modelagem, documentação e validação de requisitos. As respostas foram analisadas qualitativamente por meio de blocos temáticos, permitindo a identificação de padrões e boas práticas. Os resultados evidenciam que, embora o levantamento de requisitos seja reconhecido como etapa fundamental, ainda há dificuldades relacionadas à comunicação com stakeholders, definição de escopo e gestão de mudanças. O estudo reforça contribuir para a reflexão sobre os desafios enfrentados por desenvolvedores Web e oferece insights para a melhoria do processo de levantamento de requisitos nesse contexto.

Palavras-chave: Levantamento de Requisitos; Aplicações Web; Experiência do desenvolvedor; Tecnologias Web; Boas Práticas.

ABSTRACT

This study aims to analyze how requirements elicitation and application occur in Web application development projects, from the perspective of developers and analysts. The research begins with a theoretical review of Requirements Engineering in the Web context, considering aspects such as responsiveness, API integration, security, Search Engine Optimization (SEO), and the impact on technology selection. Subsequently, interviews were conducted with 19 professionals working in the field during June and July 2025, addressing practices, challenges, strategies, and lessons learned in the processes of eliciting, modeling, documenting, and validating requirements. The responses were analyzed qualitatively through thematic blocks, enabling the identification of recurring patterns and good practices. The results show that although requirements elicitation is recognized as a fundamental step, there are still challenges related to stakeholder communication, scope definition, and change management. The study contributes to the reflection on the challenges faced by Web developers and offers insights to improve the requirements elicitation process in this context.

Keywords: Requirements Elicitation; Web Applications; Developer Experience; Web Technologies; Good Practices.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BASICO	Benefícios, Abrangência, Satisfação, Investimento, Cliente, Operacionalidade
CD	*Continuous Deployment*
CI	*Continuous Integration*
CSS	*Cascading Style Sheets*
ERP	*Enterprise Resource Planning*
GraphQL	*Graph Query Language*
GUT	Gravidade, Urgência e Tendência
HTTP	*Hypertext Transfer Protocol*
iOS	*Apple iPhone Operating System*
JS	*JavaScript*
LGPD	Lei Geral de Proteção de Dados
OAuth 2.0	*Open Authorization 2.0*
QA	*Quality Assurance*
QP	Questão de Pesquisa
REST	*Representational State Transfer*
RICE	*Reach, Impact, Confidence, Effort*
SEO	*Search Engine Optimization*
SOAP	*Simple Object Access Protocol*
SSR	*Server-Side Rendering*
TS	*TypeScript*
UI	*User Interface*
UML	*Unified Modeling Language*
UX	*User Experience*
XML	*Extensible Markup Language*

LISTA DE SÍMBOLOS

%	Símbolo de porcentagem, utilizado para representar valores percentuais.

SUMÁRIO

1 INTRODUÇÃO

O levantamento de requisitos é o processo de identificar, analisar e documentar as necessidades e expectativas de usuários e stakeholders para o desenvolvimento de um sistema (SOMMERVILLE, 2020). Trata-se de uma etapa inicial da engenharia de requisitos que define as bases para o projeto, determinando o que deve ser construído, quais problemas devem ser resolvidos e quais funcionalidades serão essenciais para alcançar os objetivos propostos (BEDER, 2012).

A importância do levantamento de requisitos está diretamente relacionada à qualidade do produto final (PRESSMAN; MAXIM, 2021). Quando essa etapa é conduzida de forma eficiente, reduz-se o risco de falhas, retrabalhos e insatisfação do cliente. Um bom levantamento assegura que o sistema atenda aos critérios de sucesso definidos, além de permitir que decisões de design e implementação sejam tomadas com maior assertividade (SOMMERVILLE, 2020). Por outro lado, falhas nessa fase comprometem toda a cadeia de desenvolvimento, gerando custos elevados e produtos desalinhados com as expectativas.

O desenvolvimento para a Web apresenta desafios específicos que se diferenciam dos de outras plataformas (ECCHER, 2015; RAMOS, 2019). Aplicações Web lidam com públicos amplos e heterogêneos, exigem alta disponibilidade, integração com múltiplos serviços, adaptação a dispositivos variados e atendimento a normas de segurança e desempenho. Além disso, a evolução tecnológica constante e as mudanças frequentes de requisitos impõem maior complexidade ao processo de desenvolvimento, exigindo estratégias mais flexíveis e colaborativas (KUMAR; SANGWAN, 2011).

Essas particularidades impactam diretamente o levantamento de requisitos em aplicações Web. Diferentemente de sistemas tradicionais, as demandas para aplicações Web envolvem não apenas funcionalidades básicas, mas também aspectos não funcionais, como responsividade, interoperabilidade, acessibilidade, SEO e integração com APIs externas. O levantamento deve, portanto, ir além da coleta de funcionalidades, buscando compreender o contexto do usuário, os objetivos de negócio e as restrições técnicas que influenciam o projeto (CHAWLA; SRIVASTAVA, 2012).

Apesar da relevância dessa etapa, a literatura ainda carece de estudos voltados especificamente para o levantamento de requisitos em sistemas Web. Muitos trabalhos abordam a engenharia de requisitos de forma geral, sem considerar os desafios práticos enfrentados por desenvolvedores no ambiente Web. Essa lacuna inclui, por exemplo, a influência dos requisitos

nas escolhas tecnológicas e a dificuldade de formalizar processos em contextos dinâmicos, onde a comunicação com stakeholders é constantemente desafiada.

Diante desse cenário, este trabalho busca compreender como ocorre o levantamento e a aplicação de requisitos em projetos de desenvolvimento de aplicações Web, a partir da perspectiva de profissionais atuantes na área. Para isso, foram conduzidas entrevistas semi-estruturadas com 19 desenvolvedores entre os meses de junho e julho de 2025, abordando práticas utilizadas, desafios enfrentados, influência dos requisitos nas decisões técnicas e boas práticas adotadas.

Por fim, este capítulo apresenta ainda a estrutura do restante do trabalho. O Capítulo 2 aborda os conceitos fundamentais da engenharia de requisitos e discute as particularidades desse processo no contexto de aplicações Web. O Capítulo 3 apresenta os trabalhos relacionados. O Capítulo 4 descreve os procedimentos metodológicos adotados, detalhando a elaboração do protocolo de entrevistas, a coleta e a análise dos dados. O Capítulo 5 apresenta os resultados obtidos a partir das entrevistas, organizados de acordo com as questões de pesquisa. Por fim, o Capítulo 6 reúne as conclusões, contribuições e sugestões para pesquisas futuras.

1.1 Objetivo

1.1.1 Objetivo Geral

Este trabalho tem como objetivo analisar o processo de levantamento e gestão de requisitos no desenvolvimento de aplicações Web, considerando as práticas adotadas, os tipos de requisitos mais recorrentes, os desafios enfrentados e os fatores específicos desse contexto. Busca-se compreender como os requisitos influenciam decisões técnicas, como a escolha de tecnologias, bem como identificar boas práticas e estratégias utilizadas por profissionais para aprimorar a qualidade e a eficiência dessa etapa, contribuindo para a melhoria desse processo no contexto Web.

1.1.2 Objetivos Específicos

- **Realizar uma revisão da literatura sobre Engenharia de Requisitos com foco no contexto Web:** Espera-se compreender os conceitos fundamentais da Engenharia de Requisitos e identificar lacunas relacionadas à sua aplicação em projetos de aplicações Web, especialmente quanto aos desafios e práticas específicas desse contexto.

- **Elaborar um protocolo de entrevista com base em quatro questões de pesquisa:** Busca-se estruturar um instrumento que permita explorar de forma organizada os métodos de levantamento de requisitos, as dificuldades enfrentadas pelos profissionais, a influência dos requisitos nas escolhas tecnológicas e as boas práticas adotadas.
- **Conduzir entrevistas com profissionais atuantes no desenvolvimento de aplicações Web:** Pretende-se obter dados empíricos que reflitam a experiência prática de desenvolvedores e analistas, permitindo observar como o levantamento de requisitos é realizado no ambiente profissional.
- **Analisar qualitativamente os dados coletados:** Busca-se identificar padrões, tendências e categorias temáticas a partir das respostas, de modo a compreender melhor os fatores que influenciam o processo de levantamento de requisitos.
- **Discutir os achados da pesquisa e apresentar lições aprendidas:** Espera-se que a discussão dos resultados contribua para a reflexão sobre práticas atuais, evidenciando pontos de melhoria e fornecendo subsídios para futuras pesquisas e aprimoramento de processos no desenvolvimento Web.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão discutidos os principais aspectos da engenharia de requisitos no contexto de aplicações Web, abordando o ciclo de vida dos requisitos, as técnicas de elicitação mais utilizadas, a importância da validação e verificação, além da classificação dos requisitos em funcionais e não funcionais. Também será apresentada uma visão geral sobre o desenvolvimento de aplicações Web, incluindo suas camadas principais, tecnologias mais comuns, desafios relacionados à responsividade, segurança, desempenho e integração, bem como a influência dessas características no levantamento e na especificação dos requisitos. Essas discussões permitirão compreender como a especificação e o desenvolvimento podem ser adaptados para atender às particularidades dos sistemas Web, garantindo um processo mais eficiente e alinhado às necessidades do usuário final.

2.1 Desenvolvimento de Aplicações Web

O desenvolvimento de aplicações Web envolve a criação de sistemas que funcionam por meio de navegadores e utilizam tecnologias da Web como HTML, CSS, JavaScript e, em muitos casos, frameworks e bibliotecas específicas, tanto no lado cliente quanto no servidor (BOEHM; RUVALCABA, 2015; PRESSMAN; MAXIM, 2021). Essas aplicações têm como características principais a acessibilidade via internet, a independência de plataforma e a possibilidade de atualizações centralizadas. Uma aplicação Web típica é composta por três camadas principais: a interface do usuário, a lógica de negócios e o armazenamento de dados. O desenvolvimento nessas camadas requer atenção a diversos fatores, como responsividade, segurança, desempenho e integração com serviços externos (PRESSMAN; MAXIM, 2021).

De acordo com Pressman e Maxim (PRESSMAN; MAXIM, 2021), aplicações Web são sistemas dinâmicos que devem lidar com mudanças constantes de requisitos, alta concorrência de usuários e a necessidade de boa experiência de uso. Essas características tornam o processo de desenvolvimento mais ágil e adaptativo, muitas vezes apoiado por metodologias como Scrum e Kanban.

Outro aspecto importante é a diversidade de dispositivos e navegadores utilizados pelos usuários finais, o que impõe desafios adicionais ao projeto da interface e à compatibilidade entre plataformas. Isso exige que os requisitos considerem, desde o início, aspectos como usabilidade, acessibilidade, tempo de carregamento e comportamento responsivo (BOEHM;

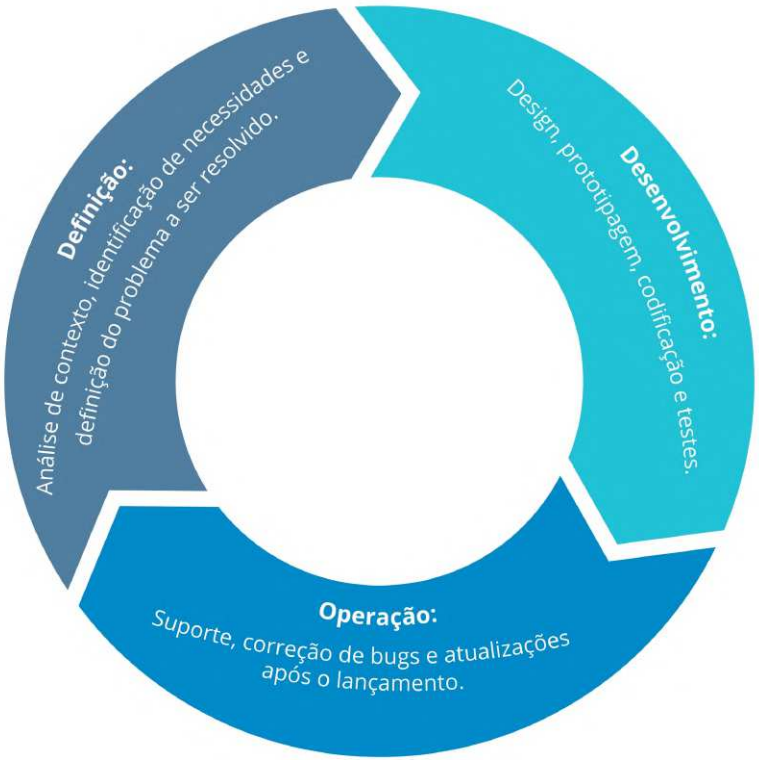
RUVALCABA, 2015).

Além disso, o uso de bibliotecas e frameworks específicos como React, Angular, Vue.js, Django, Laravel, entre outros, influencia diretamente o desenvolvimento. A escolha dessas tecnologias depende de fatores como escalabilidade, comunidade de suporte, curva de aprendizado e alinhamento com os requisitos do sistema (SOMMERVILLE, 2020). Dessa forma, o levantamento de requisitos para aplicações Web deve considerar não apenas o que o sistema deve fazer, mas também como ele deve se comportar em relação ao ambiente Web, como segurança, desempenho, disponibilidade e escalabilidade.

2.2 Requisitos em aplicação Web

O processo de desenvolvimento de aplicações propõe uma série de fases e atividades dentro do seu ciclo de vida, estabelecendo um fluxo estruturado para garantir que os requisitos do sistema sejam corretamente definidos e implementados. Em geral, esse processo possui três fases principais: Definição, Desenvolvimento e Operação (SOMMERVILLE, 2020), conforme a Figura 1.

Figura 1 – Fluxo Simplificado de Desenvolvimento



Fonte: Produzido pela autora com base em (SOMMERVILLE, 2020).

Na fase de definição, é utilizada a Engenharia de Requisitos, sendo responsável

por entender o funcionamento da aplicação, como será a experiência do usuário final e como o sistema irá influenciar nos negócios do cliente. O entendimento correto dos requisitos é uma etapa crucial no desenvolvimento de software, pois tem um impacto direto na qualidade final do produto. Segundo Sommerville (SOMMERVILLE, 2020), a obtenção de requisitos é definida como a descrição das funcionalidades e restrições operacionais de um sistema. Esses requisitos representam as necessidades dos clientes que buscam resolver problemas por meio do sistema, permitindo a definição das restrições e características que o sistema deve apresentar.

Ademais, para se obter esses requisitos, o processo deve coletar informações sobre os requisitos de usuário e de sistema que a aplicação a ser desenvolvida deverá apresentar. Isso envolve identificar e documentar as necessidades, expectativas e restrições dos usuários e partes interessadas envolvidas no projeto. Essas informações são obtidas por meio da interação com as partes interessadas, que podem ocorrer por meio de reuniões, entrevistas e observações. A execução inadequada dessa fase é frequentemente responsável pelo fracasso de projetos, destacando a importância de conduzi-la corretamente. Esse problema pode estar relacionado à dificuldade dos profissionais em realizar essa etapa de forma adequada, indicando uma falta de conhecimento necessário para sua execução.

Diferente de sistemas tradicionais, as aplicações Web demandam que o levantamento de requisitos considere fatores específicos desse ambiente, como a necessidade de responsividade, integração contínua com serviços externos e práticas que favoreçam visibilidade nos mecanismos de busca. Conforme Marcotte (MARCOTTE, 2011) e Eccher (ECCHER, 2015), requisitos relacionados ao design adaptável e à experiência do usuário se tornaram indispensáveis, exigindo que desenvolvedores antecipem múltiplos contextos de uso. Além disso, práticas de *Search Engine Optimization* (SEO), devem ser incorporadas ainda na definição inicial do sistema, pois afetam diretamente o alcance e a performance do produto (DOVER; DAFFORN, 2011). A integração com *Application Programming Interface* (API) de terceiros e a necessidade de garantir segurança em cada troca de informação, reforçam que requisitos técnicos e não funcionais são fundamentais nesse tipo de aplicação (AMUNDSEN, 2023; JACOBSON *et al.*, 2011). Essas particularidades evidenciam que a engenharia de requisitos, quando aplicada a sistemas Web, deve ir além da elicitação tradicional, incorporando práticas capazes de lidar com mudanças rápidas, diversidade tecnológica e demandas de segurança e escalabilidade.

2.3 Engenharia de Requisitos

A engenharia de requisitos desempenha um papel importante na jornada de um desenvolvedor de sucesso, pois ela estabelece uma base sólida para qualquer projeto. Ela proporciona uma compreensão minuciosa das necessidades e expectativas do cliente, o que, por sua vez, é traduzido diretamente na criação de um produto perfeitamente alinhado com as demandas reais. Por outro lado, a ausência de uma abordagem adequada de engenharia de requisitos resulta em uma falta de entendimento dessas necessidades, o que pode levar a um desenvolvimento desorganizado do produto. Isso, por sua vez, gera a correção de muitos erros que poderiam ter sido evitados, atrasos e custos adicionais consideráveis (PRESSMAN; MAXIM, 2021).

Sendo uma fase essencial da engenharia de software e no desenvolvimento de aplicações, a engenharia de requisitos é um estudo que se concentra em auxiliar a comunicação do desenvolvedor com todos os *stakeholders*, com o intuito de que ele trabalhe de forma eficiente com as informações coletadas durante todas as etapas da concepção do produto, entregando qualidade e um custo-benefício alto (RAMOS, 2019). Dessa forma, a engenharia de requisitos trabalha na identificação, análise, documentação e gerenciamento de requisitos, contribuindo para um produto bem-sucedido, pois garante que as metas do projeto sejam atendidas de forma eficaz (PRESSMAN; MAXIM, 2021).

2.3.1 Planejamento e Preparação

A fase de planejamento é o estágio que estabelece as bases para o sucesso de qualquer projeto Web, definindo seus objetivos, escopo e direção geral. Por ser a etapa inicial, assume papel determinante para o bom andamento do desenvolvimento, pois envolve decisões estratégicas sobre como o sistema será organizado, quais serão suas principais funcionalidades e como os diferentes componentes se relacionarão entre si. Nesse contexto, destaca-se a relevância de uma análise aprofundada da concepção do projeto, que tem como propósito principal fornecer fundamentos sólidos para o desenvolvimento de uma aplicação Web.

Durante a preparação, o foco recai na identificação das necessidades e expectativas do cliente, bem como na definição clara dos objetivos que o sistema deve atender. Para alcançar qualidade, é necessário ir além do conhecimento superficial dos usuários finais, compreendendo suas perspectivas e incorporando essa compreensão em todas as fases do desenvolvimento, de

forma a criar soluções eficazes. Esse processo envolve diversas etapas, cada uma contribuindo para a definição inicial do escopo e das características da aplicação web, com impacto que se estende até a conclusão do projeto.

A comunicação eficiente é outro aspecto de grande importância em todas as etapas do levantamento de requisitos. Ela não se limita à interação entre desenvolvedores, mas também exige uma troca clara e objetiva com os *stakeholders* (RAMOS, 2019). Esse alinhamento garante que os requisitos coletados representem fielmente as necessidades do negócio e as expectativas dos usuários. Além disso, promover a colaboração contínua entre todos os membros da equipe é um fator decisivo para o sucesso do projeto.

A revisão periódica dos objetivos e o ajuste das estratégias conforme o projeto evolui permitem que a equipe responda às mudanças de forma ágil e eficaz. O processo de levantamento de requisitos é dinâmico, e manter uma comunicação aberta contribui para que todos os envolvidos estejam sempre alinhados com as prioridades do projeto. Para entender melhor como essas interações se desdobram no desenvolvimento de sistemas, podemos observar o ciclo de vida dos requisitos, que organiza e estrutura cada fase desse processo de forma iterativa e adaptável.

2.3.1.1 Ciclo de vida dos requisitos

O ciclo de vida dos requisitos representa um conjunto de atividades fundamentais envolvidas na gestão e desenvolvimento de requisitos para sistemas ou aplicações. Na Figura 2, é possível visualizar as principais fases desse ciclo. Diferente de um processo estritamente sequencial, as etapas não seguem uma ordem fixa. Elas podem ocorrer de forma paralela ou iterativa, retornando a fases anteriores sempre que forem identificadas novas necessidades, ajustes ou inconsistências. Essa natureza cíclica é essencial para lidar com as incertezas e mudanças típicas de projetos de software, especialmente no contexto Web, onde demandas podem evoluir rapidamente. Cada uma dessas atividades contribui para garantir que o sistema final atenda às expectativas dos *stakeholders*, mantendo-se dentro dos limites de tempo, orçamento e qualidade. Essa flexibilidade permite que o ciclo se adapte à realidade do projeto, sem comprometer a rastreabilidade e o alinhamento com os objetivos iniciais.

Figura 2 – O ciclo de vida dos requisitos

Fonte: Produzido pela autora com base em (HEATH, 2020).

Este ciclo garante a rastreabilidade dos requisitos, permitindo que cada mudança seja monitorada e seu impacto avaliado. Dessa forma, o projeto se mantém alinhado com os objetivos iniciais, com maior controle sobre a entrega final e a satisfação do cliente. A seguir, são descritas as principais etapas do ciclo de vida dos requisitos:

- **Elicitação**: Consiste na coleta de informações sobre o que o sistema deve fazer, com base nas necessidades dos *stakeholders*. Envolve o uso de técnicas como entrevistas, questionários, observações e workshops.
- **Validação**: Busca garantir que os requisitos levantados estejam corretos, completos, consistentes e que representem fielmente as necessidades dos usuários e demais partes interessadas.
- **Modelagem**: Representa os requisitos de forma estruturada, utilizando diagramas ou modelos (como casos de uso ou diagramas de atividades), com o objetivo de facilitar a compreensão e comunicação entre os envolvidos no projeto.
- **Especificação**: Trata da descrição formal dos requisitos, tanto funcionais quanto não funcionais, com clareza, precisão e sem ambiguidades, servindo como base para o desenvolvimento do sistema.

- **Documentação**: Envolve o registro organizado dos requisitos em artefatos formais, como o Documento de Requisitos de Software (DRS), garantindo sua rastreabilidade e reuso.
- **Rastreabilidade**: Refere-se à capacidade de acompanhar cada requisito desde sua origem até sua implementação e testes, permitindo a gestão eficiente de mudanças e validação das entregas.
- **Classificação**: Os requisitos podem ser categorizados por tipo (funcional, não funcional), por prioridade (alta, média, baixa) ou por origem, facilitando a análise e organização durante o projeto.
- **Priorização**: Visa definir quais requisitos devem ser atendidos primeiro, considerando fatores como valor de negócio, urgência, custo e complexidade técnica.
- **Verificação**: Avalia se os requisitos foram corretamente implementados no sistema, por meio de revisões, inspeções e testes. Garante que o que foi desenvolvido está de acordo com o que foi especificado.
- **Gestão de mudanças**: Trata do controle de alterações nos requisitos ao longo do projeto, incluindo a análise de impacto, aprovação e atualização da documentação, assegurando que o sistema se mantenha coerente e alinhado aos objetivos.

2.3.2 Coleta de Informações (Elicitação)

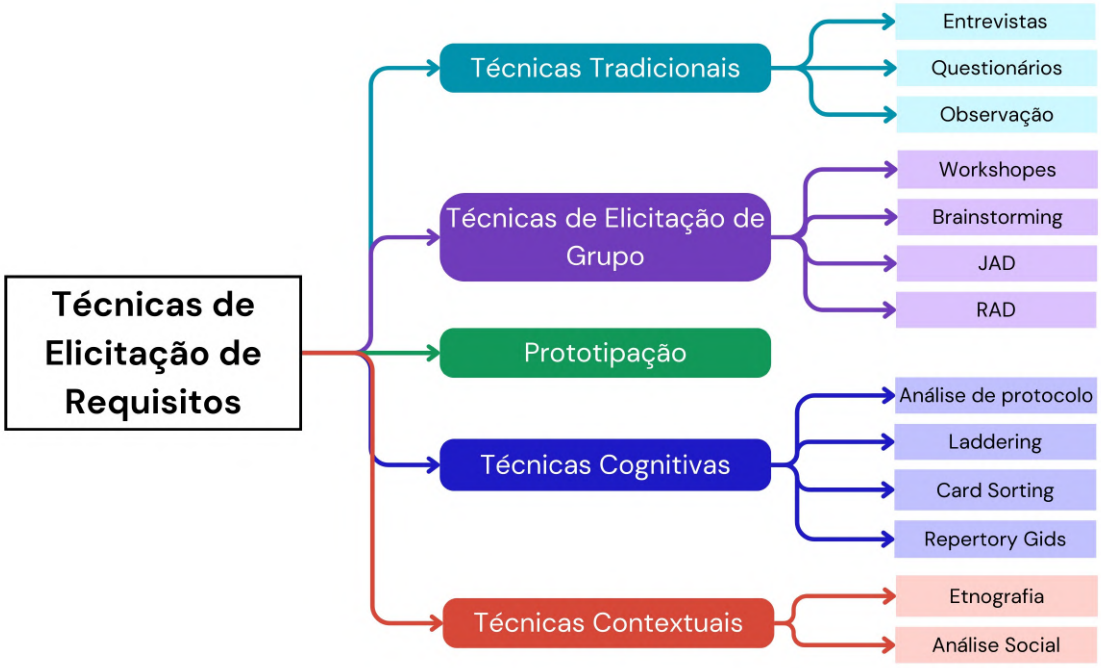
Na fase de coleta de informações, é fundamental realizar uma análise prévia bem estruturada, dado que esse planejamento deve ocorrer considerando que o tempo é um recurso limitado. Essa preparação é particularmente crucial ao planejar estratégias para interagir eficazmente com *stakeholders*, garantindo que todas as expectativas e necessidades sejam compreendidas e adequadamente incorporadas ao desenvolvimento do projeto (RAMOS, 2019).

A identificação das necessidades do usuário, por sua vez, é um aspecto crítico que não pode ser negligenciado, na qual o envolvimento dos *stakeholders* deve ser cuidadosamente planejado. Eles fornecem informações que ajudam a moldar o desenvolvimento Web, tornando-o funcional, eficiente e alinhado com as expectativas de todos os envolvidos. Logo, profissionais como analistas de negócios e engenheiros de requisitos desempenham um papel crucial nesse processo, colaborando e integrando elementos específicos de suas áreas de atuação.

A elicitação de requisitos é um passo que ajuda a entender as necessidades dos *stakeholders*, garantindo que o sistema desenvolvido atenda às expectativas. A Figura 3 mostra um mapa mental com diferentes categorias de Técnicas de Elicitação de Requisitos, que são

ferramentas usadas para identificar e capturar os requisitos necessários para o desenvolvimento de um sistema ou aplicação, seguido por uma breve explicação de cada uma delas:

Figura 3 – Técnicas de Elicitação de Requisitos



Fonte: Produzido pela autora com base em (REGO, 2023).

- **Técnicas Tradicionais:**
  - **Entrevistas:** Consiste em uma conversa estruturada ou semi-estruturada com os *stakeholders* para identificar necessidades e expectativas do sistema.
  - **Questionários:** Um método de coleta de dados em larga escala, onde perguntas padronizadas são feitas aos *stakeholders* para capturar requisitos.
  - **Observação:** Envolve observar como os usuários interagem com o sistema ou realizam suas tarefas, para entender requisitos não verbalizados.
- **Técnicas de Elicitação de Grupo:**
  - **Workshops:** Sessões colaborativas onde *stakeholders* e a equipe de desenvolvimento se reúnem para discutir e identificar os requisitos do sistema.
  - **Brainstorming:** Método criativo de geração de ideias, onde os participantes propõem soluções e identificam requisitos sem restrições iniciais.
  - **JAD (Joint Application Development):** Reuniões focadas em colaboração entre os *stakeholders* e desenvolvedores, facilitadas para capturar requisitos de forma eficiente.

  - **RAD (Rapid Application Development):** Técnica de desenvolvimento rápido que envolve ciclos curtos de feedback e prototipagem para ajustar os requisitos.
- **Prototipação:** A criação de protótipos do sistema permite que os *stakeholders* interajam com uma versão inicial e forneçam feedback, ajudando a ajustar e refinar os requisitos.
- **Técnicas Cognitivas:**
  - **Análise de protocolo:** O usuário descreve em voz alta o que está fazendo durante uma tarefa, permitindo a extração de requisitos baseados nas suas ações.
  - **Laddering:** Um método de entrevistas que explora cadeias de raciocínio do usuário para entender seus valores e como eles influenciam os requisitos.
  - **Card Sorting:** Técnica onde os usuários organizam cartões com termos ou funcionalidades, ajudando a identificar como categorizam informações e requisitos.
  - **Repertory Grids:** Uma técnica para comparar diferentes elementos do sistema e entender as percepções dos usuários sobre eles, resultando em requisitos mais claros.
- **Técnicas Contextuais:**
  - **Etnografia:** Envolve a imersão no ambiente de trabalho dos usuários para observar como eles interagem com suas tarefas diárias, ajudando a identificar requisitos implícitos.
  - **Análise Social:** Examina as interações sociais e o contexto em que o sistema será utilizado, ajudando a capturar requisitos relacionados à dinâmica social e de trabalho.

2.3.3 Verificação e Validação dos Requisitos

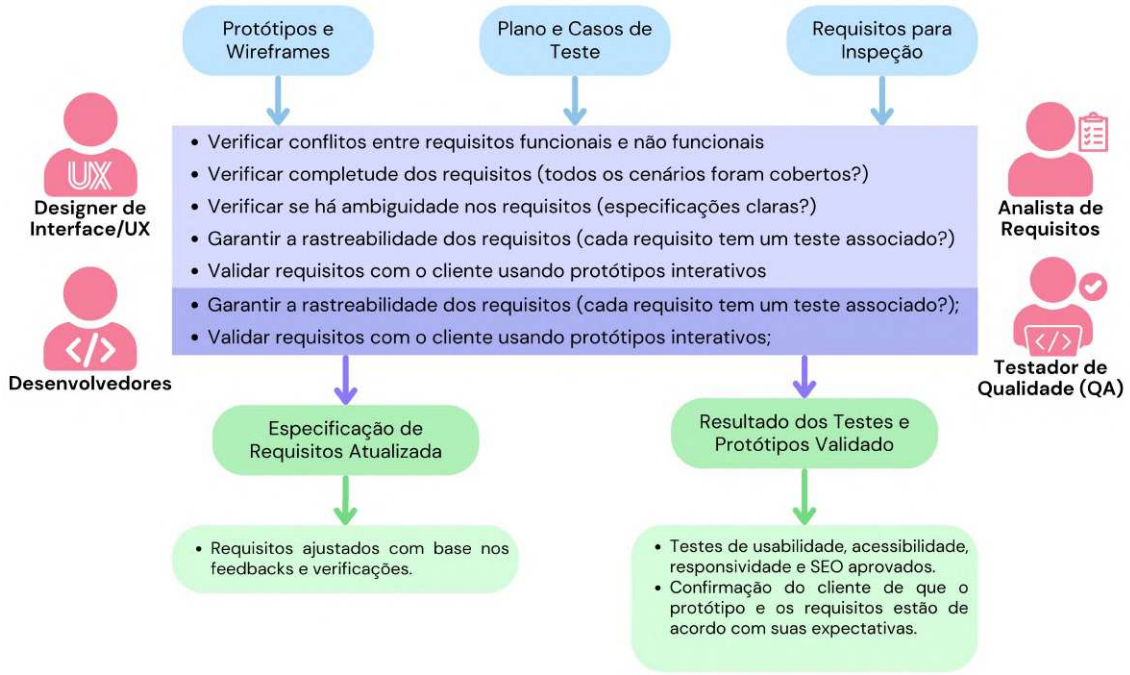
A validação de requisitos garante que as especificações atendam às necessidades dos *stakeholders* e estejam livres de falhas que possam comprometer a entrega do produto final, sendo essencial verificar se a comunicação entre as partes envolvidas está sendo eficaz e se todos os requisitos foram corretamente compreendidos e documentados (SOMMERVILLE, 2020). A validação envolve técnicas e processos que ajudam a identificar e corrigir problemas antes que eles se tornem mais difíceis de resolver, resultando em um alto custo. A seguir, a Figura 4 representa um processo de verificação e validação de requisitos em uma aplicação Web, envolvendo diferentes papéis na área de desenvolvimento — Designer *User Interface* (UI)/*User Experience* (UX), Desenvolvedores, Analista de Requisitos e Testador de Qualidade (*Quality Assurance* (QA)):

- **Azul claro** – Representa as etapas do processo (*Protótipos e Wireframes*, *Plano e Casos de*

Teste e *Requisitos para Inspeção*), indicando o fluxo sequencial da verificação e validação.

- **Roxo** – Destaca as atividades de verificação e validação que são realizadas em conjunto por diferentes papéis, evidenciando pontos de colaboração e interdependência entre equipes.
- **Verde claro** – Representa os produtos gerados ao final do processo (*Especificação de Requisitos Atualizada* e *Resultado dos Testes e Protótipos Validados*), ou seja, entregáveis resultantes das atividades.
- **Rosa** – Identifica os papéis ou funções envolvidas no processo, facilitando a compreensão de responsabilidades e atribuições.

Figura 4 – Exemplo de Verificação e Validação de Requisitos



Fonte: Produzido pela autora.

A Figura 4, destaca o papel de diferentes profissionais e os produtos gerados em cada etapa. O processo se inicia com a criação de *protótipos* e *wireframes*, geralmente conduzida por designers de interface/UX e desenvolvedores. Nessa fase, os protótipos são usados para:

- Identificar conflitos entre requisitos funcionais e não funcionais;
- Avaliar a completude das especificações, verificando se todos os cenários foram contemplados;
- Detectar ambiguidades nas descrições;
- Garantir a rastreabilidade dos requisitos, assegurando que cada um possua um teste associado;
- Validar requisitos junto ao cliente, utilizando protótipos interativos.

Em paralelo, a equipe de desenvolvimento define **planos e casos de teste**. Esses documentos estruturam como cada requisito será verificado, reforçando a rastreabilidade e permitindo a validação contínua durante o projeto. Também ocorre a etapa de **inspeção de requisitos**, conduzida por analistas de requisitos e profissionais de QA. Essa análise foca na consistência, clareza e testabilidade das especificações, verificando se elas permitem a criação de casos de teste adequados e se cumprem requisitos não funcionais, como desempenho e segurança.

A partir dessas atividades, dois principais produtos são gerados:

1. **Especificação de Requisitos Atualizada** – Documento que incorpora ajustes provenientes de feedbacks, testes e verificações, garantindo que o conteúdo esteja alinhado às necessidades do projeto.
2. **Resultado dos Testes e Protótipos Validados** – Confirmação de que os testes de usabilidade, acessibilidade, responsividade e SEO foram aprovados, além da validação por parte do cliente de que o protótipo atende às expectativas.

Esse fluxo é **iterativo**, ou seja, pode ser repetido até que todos os requisitos sejam devidamente verificados e validados. Ele reforça que a validação não é uma etapa isolada, mas um processo contínuo, essencial para prevenir problemas. Estudos apontam que até 60% dos defeitos em sistemas podem ser atribuídos a falhas no levantamento ou na validação de requisitos (SUKUMARAN *et al.*, 2006), o que evidencia a importância de um processo estruturado e colaborativo (DARGHAM; SEMAAN, 2008).

2.3.4 Modelagem

A modelagem de requisitos, tanto funcionais quanto não funcionais, é um processo que envolve revisões contínuas ao longo do ciclo de vida do projeto. Esse processo abrange a definição e estruturação dos serviços e funcionalidades esperados de um sistema. A modelagem deve ser bem estruturada, especialmente em sistemas mais complexos, como os serviços médicos baseados na Web (ANZBÖCK; DUSTDAR, 2005). Outro ponto importante é o uso de padrões já estabelecidos, como exemplo o *Unified Modeling Language* (UML), uma linguagem de modelagem visual amplamente utilizada na engenharia de software para especificar, visualizar, construir e documentar os artefatos de sistemas complexos, e o BPEL *(Business Process Execution Language)*, uma linguagem padrão usada para a orquestração e automação de processos de negócios baseados em serviços web (PATHAK *et al.*, 2006).

A Figura 5 representa o processo de Especificação e Modelagem de Requisitos,

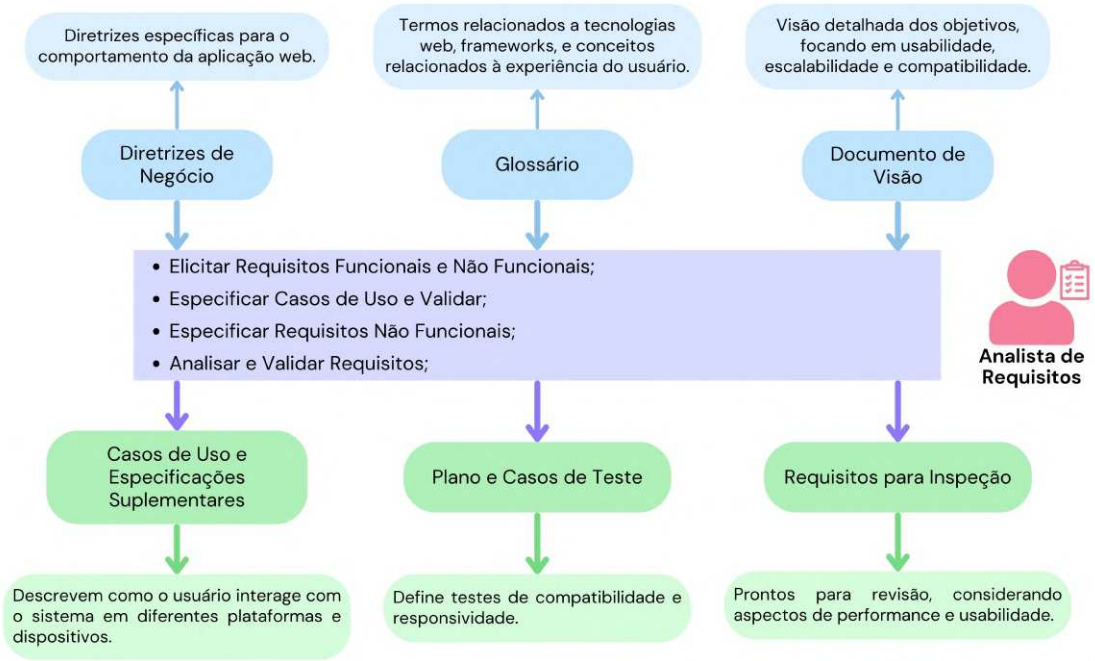
destacando as principais etapas e artefatos envolvidos na definição detalhada de requisitos para um projeto de desenvolvimento web. A seguir, descrevemos os principais elementos do fluxo de trabalho do Analista de Requisitos, destacando suas funções e os artefatos gerados ao longo do processo. Como ilustrado no diagrama, o fluxo se inicia com a definição de Diretrizes de Negócio (em azul claro, à esquerda), que especificam o comportamento esperado da aplicação web; um Documento de Visão (em azul claro, à direita), que detalha os objetivos do projeto, focando em aspectos de usabilidade, escalabilidade e compatibilidade; e o Glossário (em azul claro, ao centro), que define termos técnicos e conceitos-chave relacionados à experiência do usuário e tecnologias envolvidas. O Analista de Requisitos tem o papel (destacado em roxo) de:

- Elicitar Requisitos Funcionais e Não Funcionais;
- Especificar e validar Casos de Uso ou Histórias de Usuário;
- Especificar Requisitos Não Funcionais;
- Analisar e Validar Requisitos.

Essas atividades resultam em três principais artefatos (destacados em verde claro, que são descritos logo abaixo):

- Casos de Uso ou Histórias de Usuário e Especificações Suplementares (à esquerda);
- Plano e Casos de Teste (ao centro);
- Requisitos para Inspeção (à direita);

Figura 5 – Especificação e Modelagem de Requisitos



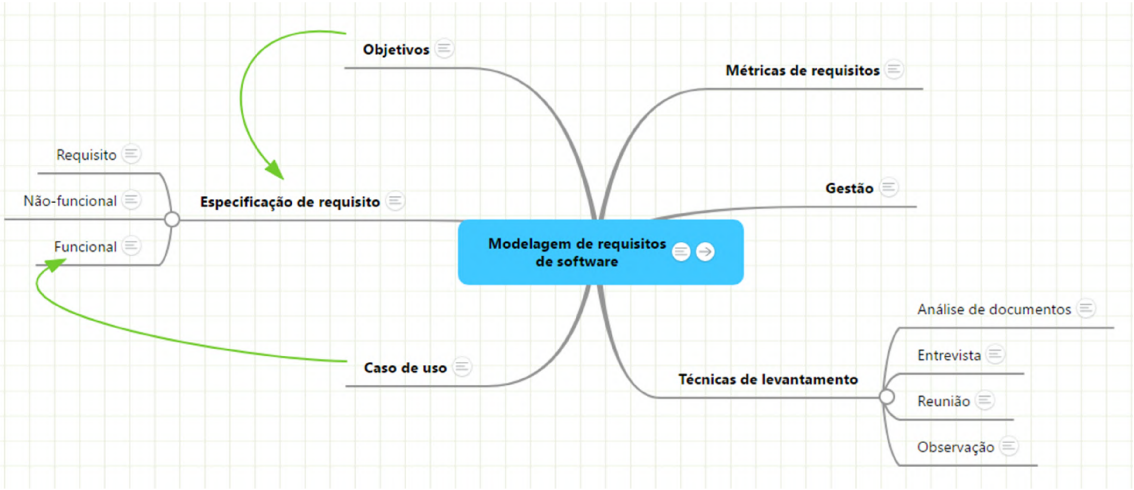
Fonte: Produzido pela autora com base em (GUEDES, 2012).

Esse diagrama mostra como as Diretrizes de Negócio, o Glossário e o Documento de Visão (todos em azul claro) se transformam em um conjunto claro de requisitos que podem ser validados e implementados como funcionalidades no sistema, assegurando que todas as partes envolvidas compartilhem uma visão comum dos objetivos e funcionalidades do software.

A Figura 6 apresenta um mapa mental de Modelagem de Requisitos, descrevendo as etapas que seguem após a fase de levantamento de requisitos. Ela é dividida em seis principais áreas:

- **Especificação de Requisitos**: Divide-se entre requisitos funcionais e não funcionais. Requisitos funcionais descrevem as funcionalidades que o sistema deve realizar, enquanto requisitos não funcionais se referem a restrições e qualidades do sistema, como desempenho e segurança.
- **Caso de Uso**: Representa como os usuários interagem com o sistema. Um caso de uso descreve o comportamento do sistema do ponto de vista de um usuário, ajudando a entender os requisitos de interação. A seta verde que conecta os "Casos de Uso"aos "Requisitos Funcionais", indica que os casos de uso são derivados diretamente dos requisitos funcionais, ou seja, cada funcionalidade especificada deve refletir uma interação entre o usuário e o sistema.
- **Objetivos**: Aqui, são definidos os principais objetivos do sistema, que orientam a modelagem e desenvolvimento, assegurando que as expectativas das partes interessadas sejam atendidas. A seta verde que conecta "Objetivos"a "Especificação de Requisitos"ressalta que os objetivos definidos no início do projeto são essenciais para a especificação de requisitos, garantindo que os requisitos estejam alinhados com as metas do projeto.
- **Métricas de Requisitos**: Determinam como medir o sucesso ou a completude dos requisitos, garantindo que sejam verificáveis e atendam às expectativas e necessidades estabelecidas.
- **Gestão**: Foca na gestão dos requisitos ao longo do ciclo de vida do projeto. A gestão eficiente assegura que mudanças e revisões nos requisitos sejam controladas e documentadas adequadamente.
- **Técnicas de Levantamento**: Estas são as técnicas usadas para coletar os requisitos iniciais. Incluem métodos como análise de documentos, entrevistas, reuniões, observações, dentre outros, que garantem que todas as necessidades dos *stakeholders* sejam capturadas corretamente.

Figura 6 – Mapa mental de Modelagem de Requisitos



Fonte: (GUERRA, 2009).

O objetivo da modelagem de requisitos, conforme descrito na figura, é consolidar e detalhar todos os aspectos do sistema, garantindo que cada requisito seja compreendido e validado antes da implementação. Este processo serve para assegurar que o produto final esteja alinhado com as expectativas dos *stakeholders*. Dessa forma, conclui-se que a modelagem de requisitos não deve se limitar às funcionalidades básicas, mas também deve considerar aspectos críticos, como segurança, eficiência e a evolução contínua dos requisitos ao longo do desenvolvimento da aplicação, que são essenciais para garantir a interoperabilidade e a confiabilidade do sistema (ANZBÖCK; DUSTDAR, 2005).

2.3.5 Documentação dos requisitos

A documentação de requisitos desempenha um papel fundamental no sucesso de um projeto, pois garante que as necessidades identificadas durante a elicitação sejam registradas de forma clara, organizada e acessível para todos os envolvidos no desenvolvimento. Além de formalizar as expectativas do cliente, ela atua como um guia contínuo ao longo do ciclo de vida do sistema, reduzindo ambiguidades e minimizando o risco de retrabalho (SOMMERVILLE, 2020).

Diversas abordagens podem ser utilizadas para representar e organizar esses requisitos, variando desde modelos mais simples, como listas e descrições textuais, até técnicas visuais que facilitam o entendimento e a validação com os *stakeholders*. Entre as práticas mais adotadas no contexto do desenvolvimento Web estão as histórias de usuários, que descrevem funcionalidades sob a perspectiva do usuário final, e os diagramas de casos de uso, que oferecem

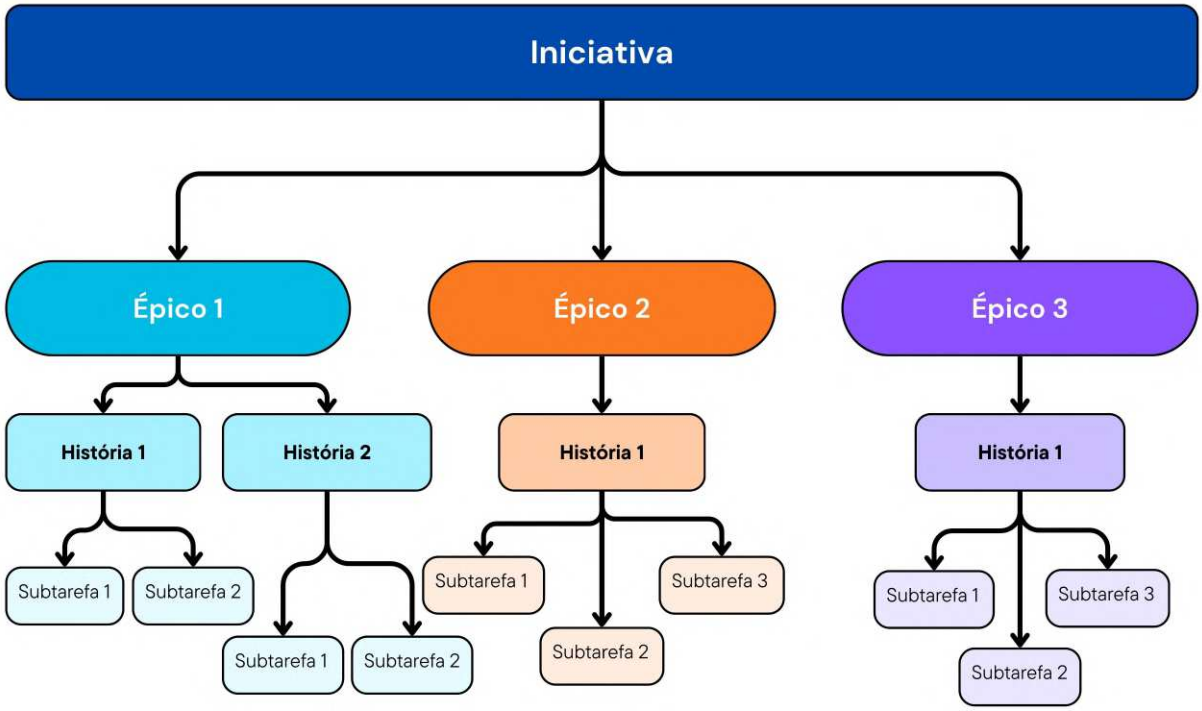
uma visão estruturada das interações entre usuários e o sistema. A seguir, serão apresentadas essas duas técnicas, destacando suas características, objetivos e como elas contribuem para a documentação dos requisitos.

2.3.5.1 História de usuários

Uma das abordagens mais utilizadas para facilitar a documentação do levantamento de requisitos no processo de desenvolvimento é o uso de "histórias de usuário", que representam cenários baseados nas experiências e necessidades do usuário final, com o cliente colaborando diretamente com a equipe de desenvolvimento para criar “cartões de história”, que descrevem de forma resumida um cenário relevante para o usuário. Esses cartões são utilizados pelo time de desenvolvimento para planejar e implementar as funcionalidades em versões futuras do sistema. Durante o planejamento das iterações, as histórias de usuário são detalhadas e divididas em tarefas, com o objetivo de medir o esforço e os recursos necessários para sua implementação. Além disso, as histórias são priorizadas pelo cliente, que decide quais delas trarão mais valor imediato ao negócio (SOMMERVILLE, 2020).

A Figura 7 ilustra o fluxo de organização do desenvolvimento de funcionalidades. Esse processo é dividido em diferentes níveis, começando pela Iniciativa, que representa um objetivo do projeto, que posteriormente vai ser dividida em Épicos, que são grandes blocos de funcionalidades relacionados ao objetivo principal. Cada épico engloba uma série de Histórias de Usuário, que são descrições de funcionalidades específicas que devem ser implementadas para atender às necessidades do usuário, na qual, dentro de cada uma, são identificadas Subtarefas, que são as ações ou atividades específicas necessárias para a conclusão daquela história. Essas subtarefas facilitam o planejamento e a execução de cada funcionalidade de forma mais organizada e eficiente.

Figura 7 – Modelo de histórias de usuários



Fonte: Produzido pela autora.

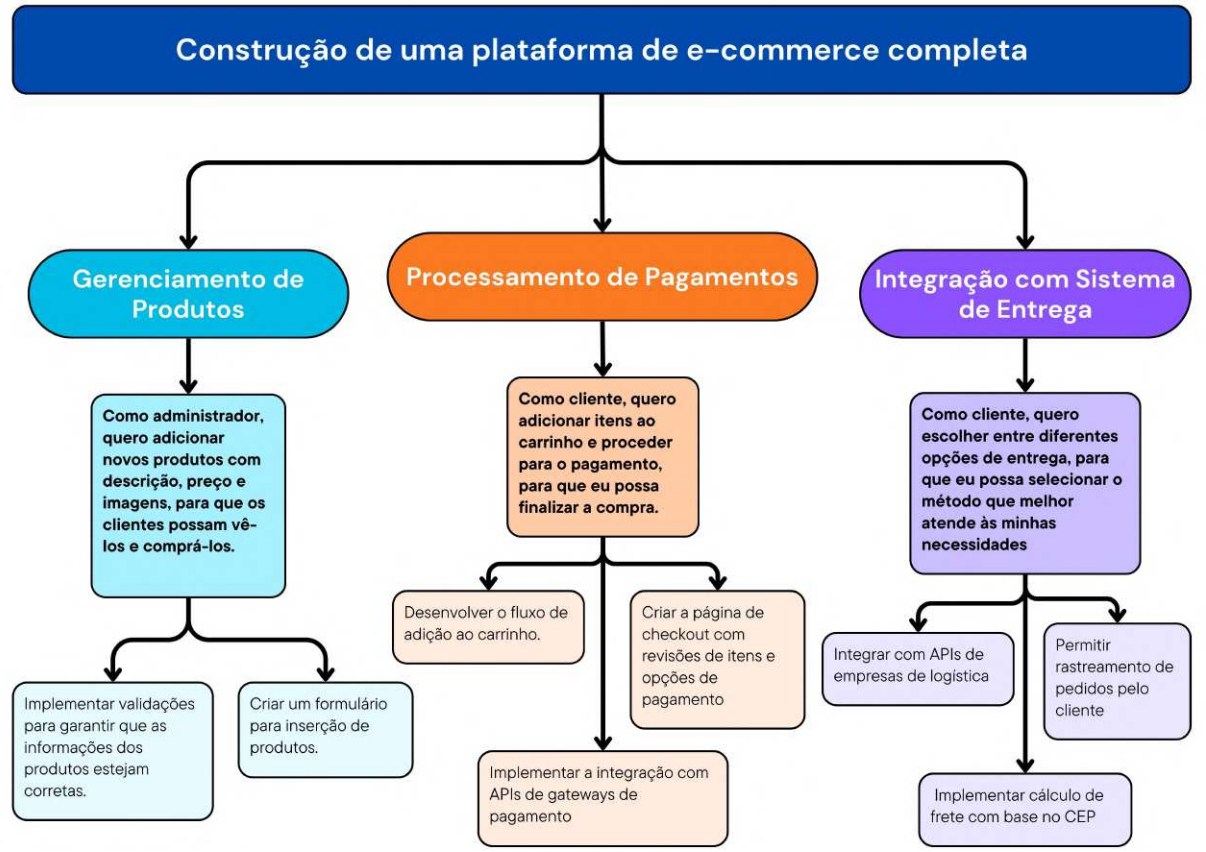
A Figura 8 apresenta a estrutura de histórias de usuário dentro do contexto da construção de uma plataforma de e-commerce. O fluxo segue a organização de Iniciativa > Épico > Histórias de Usuário > Subtarefas, similar ao modelo da figura anterior, porém com um exemplo aplicado, detalhando como uma iniciativa pode ser dividida em partes menores e mais gerenciáveis, facilitando o planejamento e a execução do projeto. A Iniciativa principal é a 'Construção de uma plataforma de e-commerce completa'. A partir dessa iniciativa, surgem três grandes Épicos:

- **Gerenciamento de Produtos**: Como administrador, o objetivo é adicionar novos produtos com descrição, preço e imagens para que os clientes possam visualizá-los e comprá-los.
- **Processamento de Pagamentos**: Como cliente, a meta é adicionar itens ao carrinho e proceder ao pagamento para finalizar a compra.
- **Integração com Sistema de Entrega**: Como cliente, o objetivo é escolher entre diferentes opções de entrega para selecionar a que melhor atende às necessidades.

Cada épico é dividido em histórias de usuário. Por exemplo, no Gerenciamento de Produtos, uma história de usuário pode descrever o desejo do administrador de adicionar produtos corretamente, enquanto no Processamento de Pagamentos, a história do cliente envolve a necessidade de adicionar itens ao carrinho e finalizar a compra utilizando APIs para o pagamento. No nível seguinte, surgem as subtarefas, que detalham as ações necessárias para atender a essas

histórias, como exemplo no Gerenciamento de Produtos, que deve ser incluída a implementação de validações e a criação de um formulário para inserção de produtos.

Figura 8 – Exemplo de histórias de usuário



Fonte: Produzido pela autora.

2.3.5.2 Casos de uso

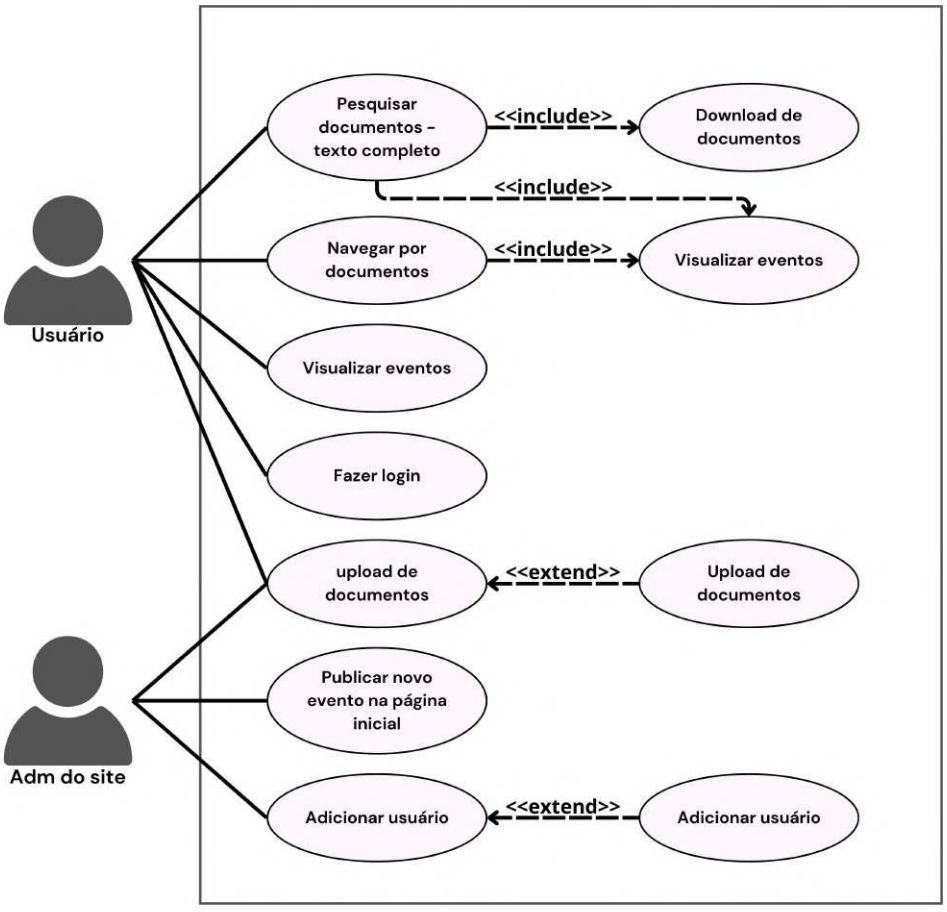
O diagrama de casos de uso desempenha um papel fundamental na modelagem do escopo inicial de um projeto, tendo como objetivo descrever quem realiza determinadas ações dentro do sistema, estabelecendo uma representação visual das interações entre os atores e os eventos que requerem uma resposta do sistema. Cada caso de uso está associado a uma tarefa que beneficia o ator e está sob sua responsabilidade, destacando os resultados entregues pelo sistema (SOMMERVILLE, 2020). Esse tipo de diagrama é útil para definir as interações principais e as entidades envolvidas, mas não se aprofunda em detalhes como a sequência de passos necessários, as regras de negócio aplicáveis ou a sequência de casos de uso que compõem um processo de negócio completo (VAZQUEZ; SIMõES, 2020).

Além de descrever as interações principais entre o sistema e seus atores, os casos de uso também são ferramentas eficazes para identificar macroprocessos que englobam objetivos

colaborativos, mas que ainda não possuem escopo claramente definido. À medida que o desenvolvimento avança e as decisões sobre quais processos serão automatizados são tomadas, os casos de uso devem ser constantemente atualizados para refletir o novo escopo do sistema. Essa evolução garante que o diagrama continue relevante e alinhado com os requisitos do projeto, permitindo uma visão progressiva das funções a serem implementadas. Portanto, a qualidade do diagrama de casos de uso está diretamente relacionada ao momento em que ele é produzido e atualizado, sendo fundamental que ele acompanhe o ciclo de vida do software e as decisões que impactam o escopo do sistema (VAZQUEZ; SIMõES, 2020).

O diagrama da Figura 9 mostra como usuários e administradores de um site interagem com diferentes funcionalidades. Ele usa os relacionamentos `include` para indicar que algumas ações dependem de outras (como baixar um documento após pesquisá-lo) e `extend` para representar que algumas ações são uma extensão de funcionalidades semelhantes feitas por outro ator (por exemplo, o upload de documentos por um usuário ou um administrador do site).

Figura 9 – Diagrama de caso de uso de um site



Fonte: Produzido pela autora.

2.3.6 Rastreabilidade (traceability)

O termo rastreabilidade refere-se à capacidade de vincular requisitos a todos os artefatos gerados ao longo do desenvolvimento de uma aplicação. Isso inclui a relação entre documentos de requisitos, especificações, modelos de análise e design, código-fonte, casos de teste e outros artefatos produzidos durante o ciclo de vida do desenvolvimento (JIRAPANTHONG, 2015). Em aplicações web, os requisitos tendem a evoluir ao longo do tempo, devido a diversos fatores, como, por exemplo, o surgimento de novas necessidades, que torna o processo de atualização de requisitos difícil. Neste contexto, a rastreabilidade é essencial por permitir avaliar o impacto de mudanças e facilitar atividades de manutenção. O uso de rastreabilidade tem sido amplamente reconhecido como uma característica fundamental de uma aplicação bem projetada, contribuindo para a qualidade e a eficiência no desenvolvimento e na evolução contínua de aplicações web (GARCIA; PAIVA, 2016).

Basicamente, rastrear significa estabelecer um vínculo entre essas informações para garantir que os requisitos possam ser monitorados e gerenciados ao longo do projeto. A rastreabilidade é essencial para qualquer tipo de desenvolvimento, pois uma grande quantidade de informações é utilizada e gerada, e todas elas precisam ser mantidas relacionadas para garantir consistência e controle do projeto (PINHEIRO, 2004).

De acordo com a norma IEEE 29148:2011, um requisito é considerado rastreável quando é possível documentar sua origem e acompanhar seu progresso, garantindo que atendam aos objetivos dos *stakeholders* e seu gerenciamento ao longo do ciclo de vida seja facilitado. Dessa forma, a rastreabilidade dos requisitos deve ser estabelecida e mantida para documentar como os requisitos foram projetados para atender aos objetivos dos *stakeholders* (IEEE, 2018).

No contexto do desenvolvimento Web, a rastreabilidade ajuda a garantir que os requisitos de stakeholders coletados na fase de levantamento sejam continuamente monitorados e atualizados. O uso de ferramentas de gerenciamento de requisitos facilita esse processo, garantindo que os requisitos estejam associados aos elementos arquitetônicos do sistema, interfaces e critérios de verificação e validação. À medida que a aplicação avança pelas fases do ciclo de vida, a rastreabilidade também deve ser mantida, assegurando que cada requisito tenha um identificador único para facilitar seu controle e acompanhamento (GARCIA; PAIVA, 2016).

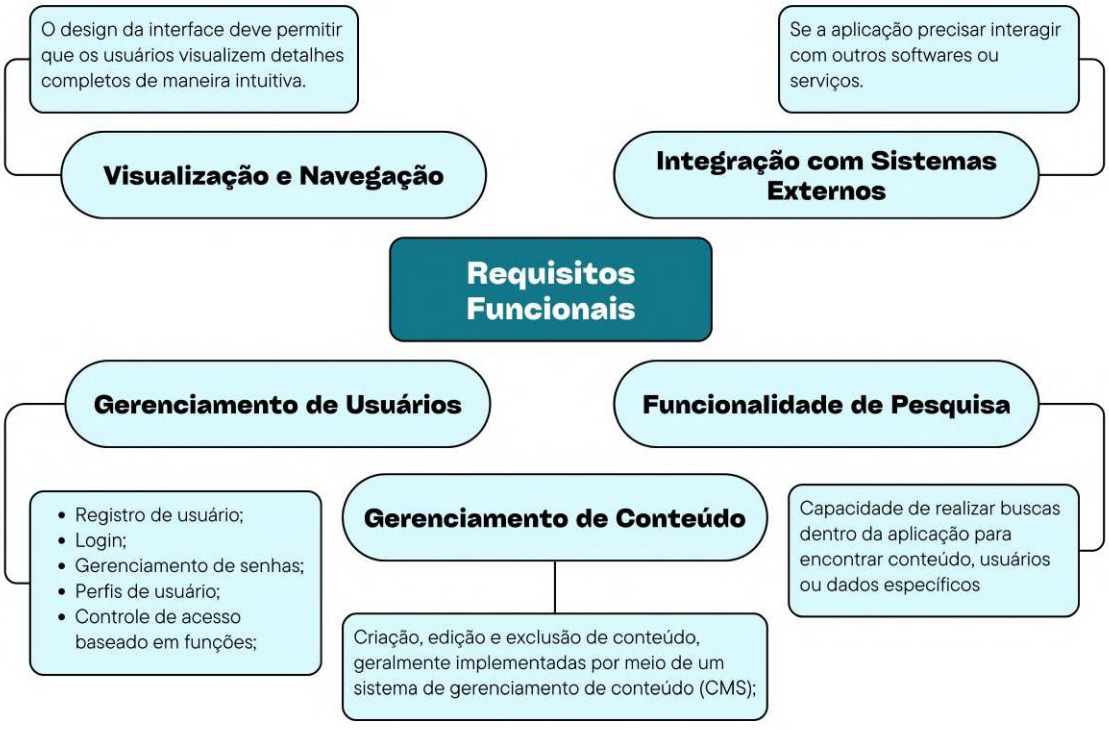
2.3.7 Classificação dos requisitos

Após a identificação das necessidades e expectativas dos *stakeholders*, o próximo passo é classificar essas necessidades em dois tipos principais de requisitos: funcionais e não funcionais.

2.3.7.1 Requisitos Funcionais

Os requisitos funcionais estão diretamente ligados aos serviços que o sistema deve oferecer, ou seja, definem as funcionalidades específicas que o software precisa executar para satisfazer as necessidades dos usuários e atingir os objetivos do projeto. Esses requisitos são fundamentais para assegurar que o sistema atenda às expectativas dos usuários e realize as tarefas necessárias para cumprir seu propósito. Além disso, eles fornecem uma orientação clara para o desenvolvimento e a testagem do sistema, permitindo que as equipes de projeto compreendam com precisão o que deve ser implementado e validado (SOMMERVILLE, 2020). Na Figura 10 é possível observar alguns exemplos de requisitos funcionais voltados para desenvolvimento Web:

Figura 10 – Exemplos de Requisitos Funcionais em uma aplicação Web



Fonte: Produzido pela autora.

2.3.7.2 Requisitos Não Funcionais

Os requisitos não funcionais, também conhecidos como atributos de qualidade, são características fundamentais que definem o comportamento do sistema além de suas funcionalidades básicas. Eles abrangem fatores como segurança, desempenho, confiabilidade e escalabilidade,e influenciam diretamente as decisões de design e arquitetura (SOMMERVILLE, 2020). Ignorar esses requisitos pode gerar custos elevados, já que alterações após a implementação do sistema podem ser complexas. Por exemplo, a decisão de modularizar um sistema para facilitar sua manutenção pode impactar negativamente o desempenho. Portanto, é fundamental priorizar esses requisitos durante as fases iniciais do desenvolvimento para evitar conflitos futuros (ANTONIO, 2008). Na Figura11 é possível observar alguns exemplos de requisitos não funcionais voltados para desenvolvimento Web:

Figura 11 – Exemplos de Requisitos Não Funcionais em uma aplicação Web

Fonte: Produzido pela autora.

2.3.8 Requisitos de sistemas comuns em aplicações Web

Além dos requisitos funcionais e não funcionais já discutidos, existem outros aspectos que são frequentemente encontrados no desenvolvimento de aplicações Web. Esses requisitos estão relacionados a funcionalidades específicas que garantem a usabilidade, performance e segurança das aplicações, além de atenderem às expectativas dos usuários. A seguir, serão apresentados alguns desses requisitos comuns:

2.3.8.1 Responsividade e Adaptação a Diferentes Telas

O design responsivo é um dos principais aspectos a serem considerados no desenvolvimento Web. Conforme descrito por Ethan Marcotte (MARCOTTE, 2011), um layout responsivo permite que uma aplicação se adapte a diferentes tamanhos de tela, proporcionando uma experiência consistente para usuários em desktops, tablets e dispositivos móveis. Essa abordagem é baseada em três pilares fundamentais:

- **Grades Flexíveis**: O uso de unidades relativas, como porcentagens em vez de pixels fixos, garante que os elementos da interface se ajustem dinamicamente ao espaço disponível.
- **Imagens Flexíveis**: A aplicação de propriedades como `max-width: 100%` impede que imagens ultrapassem os limites dos contêineres, mantendo a proporção adequada em telas menores.
- **Media Queries**: Permitem definir regras de estilo específicas para diferentes tamanhos de tela, garantindo que a interface seja ajustada conforme necessário. Um exemplo é o código abaixo, em que mostra uma consulta de mídia para ajustar o tamanho da fonte em telas menores:

Código-fonte 1 – Consulta de mídia para responsividade

```
@media screen and (max-width: 768px) {
    body {
        font-size: 100%;
    }
}
```

O Código-Fonte 1 apresenta um exemplo simples de *media query* em *Cascading Style Sheets* (CSS), utilizado para tornar uma aplicação Web responsiva. Nesse caso, a regra

`@media screen and (max-width: 768px)` define que os estilos dentro desse bloco serão aplicados apenas quando a largura da tela do dispositivo for igual ou inferior a 768 pixels, valor comumente associado a tablets e dispositivos móveis. Dentro do bloco, o seletor `body` ajusta o tamanho da fonte para `100%`, garantindo que o texto seja exibido de forma legível em telas menores. Essa abordagem permite que o layout se adapte dinamicamente a diferentes resoluções, melhorando a experiência do usuário em dispositivos variados.

Clint Eccher (ECCHER, 2015) reforça a importância do uso de frameworks como **Bootstrap e Tailwind CSS**, bem como tecnologias nativas como **CSS Grid e Flexbox**, para estruturar layouts flexíveis e responsivos. Essas ferramentas facilitam a organização dos elementos da página e melhoram a compatibilidade entre diferentes dispositivos. Além disso, a definição de *breakpoints* adequados é essencial para evitar problemas como a sobreposição de elementos e a necessidade de rolagem horizontal (MARCOTTE, 2011). Um design responsivo bem estruturado melhora não apenas a experiência do usuário, mas também o desempenho e a acessibilidade da aplicação. Por fim, garantir a responsividade de uma aplicação exige testes em múltiplos dispositivos e resoluções, de forma a identificar possíveis inconsistências e otimizar a apresentação do conteúdo (ECCHER, 2015; MARCOTTE, 2011) .

2.3.8.2 SEO (Seach Engine Optimization)

A indexação de páginas é essencial para que um site apareça nos resultados de mecanismos de busca. Isso significa que os buscadores precisam encontrar o site, entender seu conteúdo e armazená-lo para exibição quando alguém fizer uma pesquisa relacionada. Para facilitar esse processo, algumas boas práticas são recomendadas (BRASIL, 2020; ENGE *et al.*, 2022):

- **Estruturação de títulos:** O site deve ter títulos bem organizados, com cabeçalhos estruturados, como `H1` e `H2`, para facilitar a leitura do conteúdo pelos mecanismos de busca;
- **Uso do sitemap.xml:** Criar um arquivo chamado `sitemap.xml` e enviá-lo ao Google Search Console, garantindo que os buscadores consigam encontrar e indexar todas as páginas do site;
- **Configuração do robots.txt:** Evitar bloqueios desnecessários no arquivo `robots.txt`, pois ele pode impedir que os mecanismos de busca acessem certas partes do site e prejudiquem a indexação.

O arquivo `sitemap.xml` é essencial para a indexação de sites nos mecanismos

de busca, pois funciona como um mapa que lista as páginas mais relevantes, facilitando o rastreamento pelos buscadores. O Google e outras ferramentas de pesquisa utilizam esse arquivo para localizar e indexar páginas de forma mais eficiente, principalmente em sites grandes ou com conteúdos que não estão diretamente acessíveis pelo menu principal (DOVER; DAFFORN, 2011). Para otimizar o processo, é recomendável manter o `sitemap.xml` sempre atualizado e, no caso de sites grandes, dividi-lo em múltiplos arquivos para melhorar a leitura pelos mecanismos de busca (DOVER; DAFFORN, 2011; BRASIL, 2020). Sua estrutura segue um formato padronizado em *Extensible Markup Language* (XML), conforme exemplificado no Código-Fonte 2.

Código-fonte 2 – Exemplo de sitemap.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc>https://www.exemplo.com/</loc>
      <lastmod>2024-03-19</lastmod>
      <priority>1.0</priority>
   </url>
   <url>
      <loc>https://www.exemplo.com/produtos</loc>
      <lastmod>2024-03-19</lastmod>
      <priority>0.8</priority>
   </url>
</urlset>
```

- **Declaração XML:** A primeira linha (`<?xml version="1.0"encoding="UTF-8"?>`) define a versão do XML e a codificação dos caracteres, garantindo compatibilidade com diferentes idiomas e símbolos especiais.
- **Elemento raiz (`<urlset>`):** Abrange todas as URLs do sitemap e utiliza um namespace padronizado (`xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"`) para indicar que o arquivo segue o protocolo oficial de Sitemaps.
- **Tag `<url>`:** Cada bloco `<url>` representa uma página do site que será indexada pelos mecanismos de busca.
- **Tag `<loc>`:** Define a URL da página a ser indexada. No exemplo, foram listadas duas URLs: a página inicial (`https://www.exemplo.com/`) e a página de produtos

(`https://www.exemplo.com/produtos`).

- **Tag `<lastmod>`:** Indica a última data de modificação da página. Isso ajuda os motores de busca a identificar se o conteúdo foi atualizado recentemente.
- **Tag `<priority>`:** Define a prioridade da página para os mecanismos de busca. Os valores variam de 0.0 (menos importante) a 1.0 (mais importante). No exemplo, a página principal tem prioridade 1.0, enquanto a página de produtos tem 0.8.
- **Encerramento do arquivo:** O `</urlset>` fecha o documento, indicando o fim do sitemap.

O `robots.txt` é um arquivo de configuração que informa aos mecanismos de busca quais páginas do site podem ou não ser rastreadas. Ele deve ser colocado na raiz do site (ex: `https://www.exemplo.com/robots.txt`) e segue um formato padronizado para definir permissões de acesso (DOVER; DAFFORN, 2011). Embora o `robots.txt` seja útil para restringir o rastreamento, ele não impede que páginas sejam indexadas caso outros sites apontem links para elas. Para evitar a indexação completa, recomenda-se o uso da meta tag `noindex` diretamente no HTML da página (DOVER; DAFFORN, 2011). Um exemplo básico de um arquivo `robots.txt` pode ser visto no Código-Fonte 3:

Código-fonte 3 – Exemplo de robots.txt

```
1 User-agent: *
2 Disallow: /admin/
3 Disallow: /private/
4 Allow: /public/
5 Sitemap: https://www.exemplo.com/sitemap.xml
```

- `User-agent: *`: Significa que as regras valem para todos os mecanismos de busca.
- `Disallow: /admin/`: Impede que os buscadores acessem a pasta "admin".
- `Allow: /public/`: Permite acesso à pasta "public".
- `Sitemap: https://www.exemplo.com/sitemap.xml`: Indica a localização do sitemap do site.

Além disso, é essencial que as páginas do site tenham **URLs amigáveis**, ou seja, endereços fáceis de entender. Em vez de uma URL confusa, cheia de números e caracteres estranhos, o ideal é que ela descreva de forma clara o conteúdo da página. Por exemplo, uma URL como `www.exemplo.com/produto/tenis-esportivo` é muito melhor do que

`www.exemplo.com/p=123?ref=xyz`. Para isso, é importante usar palavras-chave relevantes, separar palavras com hífens e evitar caracteres especiais e acentos (DOVER; DAFFORN, 2011; BRASIL, 2020).

Outro ponto fundamental para otimizar um site para os buscadores é o uso de **meta tags**, que são informações adicionadas no código da página para ajudar os mecanismos de busca a entenderem melhor o seu conteúdo. Uma dessas tags é a `meta description`, que aparece nos resultados de pesquisa e deve ser curta e descrever bem o que a página oferece. Usar palavras-chave nessa descrição pode aumentar as chances de mais pessoas clicarem no link. Além disso, é recomendado configurar `meta titles` (títulos da página) e `alt text` nas imagens, ajudando tanto na indexação quanto na acessibilidade para usuários com deficiência visual (ENGE *et al.*, 2022; BRASIL, 2020).

Por fim, um detalhe técnico que pode melhorar a visibilidade do site nos mecanismos de busca é a implementação de *Server-Side Rendering* (SSR). Em muitos sites modernos, o conteúdo é carregado com *JavaScript* (JS) no navegador do usuário, o que pode dificultar a leitura pelos buscadores. O SSR resolve esse problema ao gerar as páginas no servidor antes de enviá-las ao usuário, garantindo que todo o conteúdo já esteja disponível para ser indexado corretamente. Isso melhora não apenas a indexação, mas também a velocidade do site, o que é um fator importante para o ranqueamento no Google (ENGE *et al.*, 2022).

2.3.8.3 Integração com APIs de Terceiros

Muitas aplicações Web precisam se conectar a serviços externos, como mapas interativos, sistemas de pagamento ou redes sociais. Isso é possível graças à **integração com APIs de terceiros**, que permite que um site ou aplicativo troque informações com outros sistemas de forma segura e eficiente. Para fazer isso corretamente, é importante entender alguns conceitos essenciais, como os tipos de APIs disponíveis, como proteger essas conexões e como evitar problemas ao usá-las.

A comunicação entre sistemas pode ocorrer de diferentes formas, sendo as mais comuns as APIs *Representational State Transfer (REST)* e *Graph Query Language (GraphQL)*. O modelo *REST* utiliza um conjunto padronizado de métodos *Hypertext Transfer Protocol* (HTTP), como *GET*, *POST*, *PUT* e *DELETE*, permitindo uma troca estruturada de informações entre serviços. Já o *GraphQL* permite que o cliente especifique exatamente quais dados deseja receber, reduzindo a transferência de informações desnecessárias e tornando as consultas mais

eficientes (AMUNDSEN, 2023).

Como essas APIs podem lidar com informações sensíveis, como dados de pagamento ou informações pessoais dos usuários, é essencial garantir que apenas pessoas e sistemas autorizados consigam acessá-las. Para isso, existe o *Open Authorization 2.0 (OAuth 2.0)*, um sistema que permite que aplicativos se conectem a serviços externos sem precisar armazenar senhas. Um exemplo prático de sua aplicação ocorre em plataformas de e-commerce, onde os usuários podem realizar login utilizando suas contas do Google ou Facebook. Em vez de solicitar diretamente o e-mail e senha do usuário, o site redireciona para a página do provedor de autenticação, onde o usuário concede permissão para o acesso às suas informações. O provedor, então, envia um *token* de acesso temporário, permitindo que o e-commerce autentique o usuário sem armazenar suas credenciais (JACOBSON *et al.*, 2011).

Além da autenticação, o *OAuth 2.0* também é amplamente utilizado para pagamentos online. Um e-commerce pode, por exemplo, permitir que os clientes utilizem o PayPal ou Stripe para realizar compras. Dessa forma, em vez de inserir os dados do cartão diretamente no site, o usuário é redirecionado para a plataforma de pagamento, onde a transação ocorre de forma segura. Esse modelo reduz os riscos de vazamento de informações financeiras e melhora a confiabilidade do sistema (JACOBSON *et al.*, 2011).

Outro desafio ao integrar APIs de terceiros é garantir que elas não sejam sobrecarregadas. Se um grande número de usuários fizer muitas requisições ao mesmo tempo, a API pode ficar lenta ou até parar de funcionar. Para evitar isso, muitas APIs utilizam uma técnica chamada *Rate Limiting*, que limita a quantidade de requisições que um usuário pode fazer dentro de um determinado período. Um exemplo disso é o Twitter, que restringe o número de verificações de timeline permitidas por hora. Outras estratégias incluem o *Spike Arresting*, que impede aumentos súbitos de acessos que possam prejudicar o sistema, e o *Throttling*, que reduz a velocidade das requisições ao invés de bloqueá-las completamente (AMUNDSEN, 2023). Por fim, erros podem acontecer ao se comunicar com APIs de terceiros, e é fundamental que o sistema saiba lidar com essas situações de forma adequada. Algumas estratégias importantes incluem:

- ***Mensagens de erro padronizadas*:** Permitem que o sistema informe claramente o que aconteceu e como o problema pode ser resolvido.
- ***Retry Strategy*:** Faz com que o sistema tente novamente caso a requisição falhe temporariamente, evitando falhas causadas por instabilidades momentâneas.
- ***Fallback Mechanism*:** Caso a API principal esteja fora do ar, o sistema pode recorrer a

um serviço alternativo para continuar funcionando (AMUNDSEN, 2023).

A adoção dessas boas práticas permite que as APIs externas sejam integradas de forma segura e eficiente, garantindo que os usuários tenham uma experiência estável e sem complicações. Plataformas que seguem essas diretrizes conseguem evitar falhas inesperadas e melhorar a confiabilidade dos serviços oferecidos (AMUNDSEN, 2023; JACOBSON *et al.*, 2011).

2.3.9 Análise e priorização

A priorização de requisitos é essencial para concentrar os esforços do projeto nos itens mais críticos, utilizando critérios como risco, valor para o negócio e custo. Essa abordagem contribui para a mitigação de riscos e para a definição da ordem em que os requisitos serão analisados e implementados (VAZQUEZ; SIMõES, 2020). No desenvolvimento de aplicações web, a priorização é ainda mais essencial devido à natureza dinâmica e incerta desse ambiente. Mudanças rápidas em estratégias e regras de negócio exigem adaptações constantes, e a gerência, muitas vezes, requer respostas imediatas, mesmo diante de demandas complexas ou atípicas. Nesse cenário, a entrega rápida de uma aplicação web tende a ser mais valorizada do que o esforço necessário para desenvolvê-la. Portanto, as equipes de engenharia web devem adotar uma abordagem ágil para responder a essas demandas com eficiência (PRESSMAN; MAXIM, 2009).

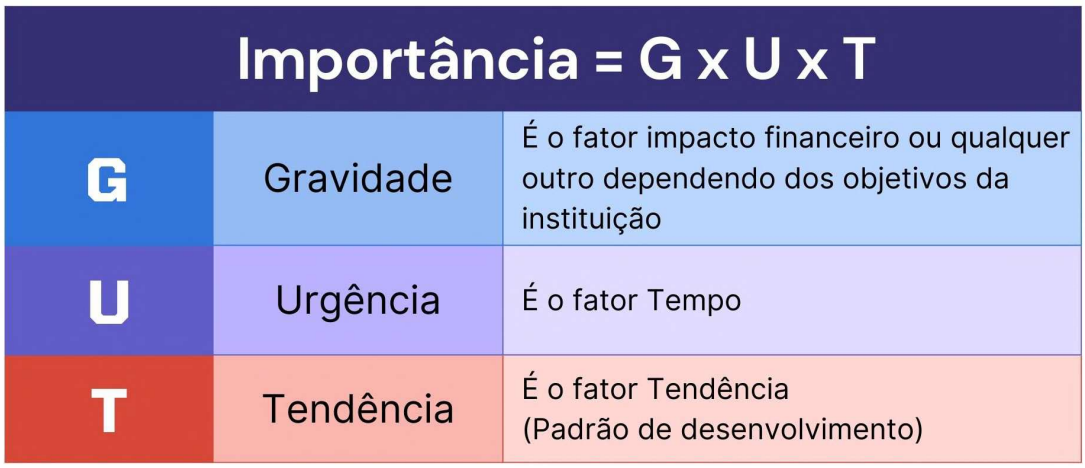
Dessa forma, a participação dos *stakeholders* é essencial nesse processo, pois são eles que determinam a urgência de cada requisito. No entanto, em muitos projetos reais, a priorização é frequentemente negligenciada ou atribuída unicamente à equipe de desenvolvimento, gerando listas de tarefas desorganizadas, nas quais tudo é tratado como urgente, o que pode levar a uma distribuição incorreta de recursos (VAZQUEZ; SIMõES, 2020). Dessa forma, para evitar esse problema, métodos de priorização como GUT, BASICO e RICE podem ser aplicados, permitindo uma comparação objetiva dos requisitos. Esses modelos ajudam a garantir que as decisões sejam fundamentadas e alinhadas com os objetivos do projeto, promovendo uma priorização mais eficaz. A seguir, será falado a respeito de cada um desses métodos.

2.3.9.1 Matriz GUT

O método Gravidade, Urgência e Tendência (GUT) usa uma escala de 1 a 5 para priorizar projetos com base em três critérios: Gravidade (impacto nas atividades), Urgência

(tempo restante para resolução) e Tendência (velocidade com que um problema pode piorar).Para aplicar o método, deve-se montar uma tabela, como ilustrado na Figura 12, na qual são atribuídas notas a cada critério para os projetos. A soma dessas notas determina a prioridade, em que projetos que obtêm as maiores pontuações são priorizados (KEPNER; TREGOE, 1972).

Figura 12 – Matriz GUT



Fonte: Produzido pela autora.

2.3.9.2 RICE

A matriz *Reach, Impact, Confidence, Effort* (RICE) é uma técnica utilizada para classificar e priorizar projetos com base em quatro critérios principais: *Reach* (alcance), *Impact* (impacto), *Confidence* (confiança) e *Effort* (esforço). O critério *Reach* representa o número estimado de pessoas ou processos que serão impactados pelo projeto em um determinado período. O *Impact* avalia a magnitude dessa mudança ou benefício gerado. Já o *Confidence* mede o nível de confiança nas estimativas anteriores, reduzindo o peso de dados incertos. Por fim, o *Effort* refere-se à quantidade de trabalho necessária para executar o projeto, geralmente medida em horas ou dias de trabalho (DAVIS *et al.*, 2001). Projetos que obtêm maior pontuação são considerados de maior prioridade, auxiliando na tomada de decisão e no uso eficiente de recursos.

Cada critério recebe uma pontuação, definida conforme a escala mais adequada ao contexto de aplicação. Essa flexibilidade permite adaptar a matriz a diferentes tipos de projetos e organizações. O cálculo da prioridade é obtido pela seguinte fórmula (equação 2.1), onde R é o

alcance, I é o impacto, C é a confiança e E é o esforço:

$$\text{Pontuação} = \frac{R \times I \times C}{E} \tag{2.1}$$

2.3.9.3 Matriz BASICO

A Matriz Benefícios, Abrangência, Satisfação, Investimento, Cliente, Operacionalidade (BASICO) é uma ferramenta para auxiliar na tomada de decisões importantes, classificando tarefas e ações por prioridade. Utilizando uma escala de 1 a 5, onde 1 representa o pior cenário e 5 o melhor, você avalia cada tarefa com base em seis critérios: Benefícios para a organização, Abrangência da solução, Satisfação dos usuários, Investimento necessário, Cliente externo satisfeito - Impacto sobre os clientes - e capacidade de Operalização. Após atribuir notas a cada critério, somam-se os valores e comparam-se a pontuação total com outras tarefas para determinar a prioridade (KEPNER; TREGOE, 1972).

O resultado dessa fórmula facilita a comparação e priorização de projetos, destacando aqueles com maiores pontuações como os de maior prioridade. A Figura 13 ilustra o funcionamento da matriz BASICO:

Figura 13 – Matriz BASICO

Fonte: Produzido pela autora.

2.3.10 Gerenciamento de Mudanças

Uma das maiores dificuldades no processo de levantamento de requisitos é a sua constante mudança ao longo do ciclo de vida do projeto. Essa instabilidade é frequentemente relatada por analistas de requisitos, sendo que muitas alterações poderiam ser evitadas com um trabalho mais sólido nas etapas iniciais. Em muitos casos, as mudanças decorrem de falhas na elicitação, como a postura passiva do analista ou uma comunicação ineficaz com os *stakeholders*, o que resulta em uma compreensão incompleta do escopo do projeto (VAZQUEZ; SIMõES, 2020).

Sommerville destaca que, especialmente em sistemas de grande porte, os requisitos estão em constante evolução, já que os problemas que o sistema busca resolver raramente podem ser completamente definidos desde o início (SOMMERVILLE, 2020). Conforme o projeto avança, os *stakeholders* podem modificar suas necessidades ou percepções, exigindo que os requisitos acompanhem essas mudanças. Além disso, fatores externos, como novas demandas de mercado, alterações legislativas ou reestruturações organizacionais, influenciam diretamente a necessidade de adaptar os requisitos. Mesmo em metodologias ágeis, onde a flexibilidade é um princípio central, é indispensável manter um processo estruturado para registrar, analisar e incorporar mudanças de forma controlada.

No contexto do desenvolvimento de aplicações web modernas, essa necessidade de adaptação constante também se reflete na forma como o software é entregue e atualizado. Espera-se que os usuários tenham acesso imediato às versões mais recentes do sistema, sem a necessidade de instalar manualmente atualizações. Essa expectativa é viabilizada por práticas de entrega contínua, que permitem evoluir o software com baixo impacto ao usuário final, garantindo que novas funcionalidades, correções de segurança e melhorias sejam disponibilizadas com agilidade (HUMBLE; FARLEY, 2010).

Para isso, tornam-se essenciais as práticas de *Continuous Integration* (CI) e *Continuous Deployment* (CD), que automatizam a integração de código, execução de testes e publicação de novas versões. Ferramentas como GitHub Actions, GitLab CI, Docker, Jenkins e CircleCI são amplamente utilizadas para configurar *pipelines* que reagem a eventos no repositório, como *commits* ou *pull requests*, disparando tarefas como *build*, *test* e *deploy*. Segundo Humble e Farley, a automação desses processos reduz erros manuais, acelera entregas e aumenta a confiabilidade da liberação de software (HUMBLE; FARLEY, 2010).

Um *pipeline* eficaz de entrega contínua deve promover o mesmo artefato entre ambi-

entes de desenvolvimento, homologação e produção, sem recompilações, garantindo consistência e facilitando o rastreamento de falhas. Essa estratégia exige a separação entre código e configuração de ambiente, possibilitando o ajuste seguro de variáveis sensíveis, como URLs ou credenciais (HUMBLE *et al.*, 2006).

Além disso, para mitigar riscos durante atualizações, técnicas como *rollback*, *blue-green deployments*, *canary releases* e *feature toggles* são empregadas. Essas abordagens possibilitam implantações progressivas, monitoramento em tempo real e reversão automática em caso de falhas, oferecendo maior controle sobre o comportamento do sistema em produção (HUMBLE *et al.*, 2006).

A integração entre gerenciamento de mudanças e entrega contínua evidencia a importância de práticas que viabilizem a adaptação constante do sistema, sem comprometer sua estabilidade. Ao automatizar processos e estruturar o controle de mudanças, as equipes de desenvolvimento ganham agilidade para responder a novas demandas e maior previsibilidade na operação. Com isso, o ciclo de vida do software torna-se mais sustentável e aderente a contextos de entrega contínua de valor, conforme proposto pelo movimento DevOps (HUMBLE; FARLEY, 2010).

2.4 Ferramentas de gerenciamento de requisitos

Para garantir que os requisitos sejam documentados e rastreados de maneira eficaz, diversas ferramentas podem ser empregadas. Nesta seção, serão apresentadas três ferramentas amplamente utilizadas para esse fim: JIRA[1], Helix RM[2] e Visure[3], que servem para aprimorar a coleta e a gestão das informações.

2.4.1 JIRA

É uma ferramenta de gerenciamento de projetos e rastreamento de problemas desenvolvida pela Atlassian. Muito utilizada para gerenciamento de requisitos e desenvolvimento ágil. Também é bastante popular em equipes ágeis e integração com outras ferramentas de desenvolvimento (PARKER, 2024).

- Prós

[1] Disponível em:<https://www.atlassian.com/software/jira>
[2] Disponível em:<https://www.perforce.com/products/helix-requirements-management>
[3] Disponível em:<https://www.visuresolutions.com/>

  - Suporte de Conformidade: Facilita o alinhamento com padrões e regulamentações, incluindo assinaturas eletrônicas e trilhas de auditoria;
  - Segurança: Oferece robustos controles de acesso e permissões para proteger dados confidenciais;
  - Boa integração com outras ferramentas;
- Contras:
  - Curva de aprendizado relativamente acentuada para novos usuários;
  - Custo elevado, especialmente para grandes equipes;

2.4.2 Helix RM

É uma ferramenta especializada para o gerenciamento de requisitos desenvolvida pela Perforce Software, voltada para a captura, organização e rastreamento durante o ciclo de vida do projeto. É mais adequada para organizações com projetos mais complexos (Visure Solutions, 2024).

- Prós
  - Suporte robusto para conformidade;
  - Boa segurança e personalização;
  - Recursos avançados de relatórios e colaboração;
- Contras:
  - Curva de aprendizado relativamente acentuada para novos usuários;
  - Custo elevado;

2.4.3 Visure

É uma ferramenta de gerenciamento de requisitos altamente configurável e flexível, adequada para quem busca personalização em seus processos. Além de gerenciar requisitos, o Visure também oferece funcionalidades para gerenciamento de mudanças, testes, riscos e rastreamento de problemas e defeitos (PARKER, 2024).

- Prós
  - Integra-se com softwares importantes como IBM DOORS, JIRA e MATLAB;
  - Suporta várias metodologias de projeto, incluindo Agile, modelo V e cascata;
  - Oferece modelos específicos para diferentes setores;
- Contras:

- Criação de modelos de relatórios pode ser tediosa e complicada;
- A instalação e configuração são desafiadoras.

3 TRABALHOS RELACIONADOS

A literatura acadêmica apresenta propostas relevantes voltadas para o levantamento de requisitos, com algumas iniciativas específicas para o contexto de sistemas Web. No entanto, ainda são escassos os estudos que abordam diretamente a prática de levantamento de requisitos a partir da perspectiva de desenvolvedores atuando em projetos reais. Os trabalhos analisados nesta seção exploram diferentes abordagens para apoiar esse processo: o primeiro propõe um método formal para elicitação e modelagem de requisitos Web; o segundo apresenta uma ferramenta baseada em campos semânticos para reduzir ambiguidades na documentação; o terceiro desenvolve uma plataforma colaborativa voltada à coleta contínua de requisitos; e o presente estudo traz uma análise empírica baseada em experiências práticas de profissionais do desenvolvimento Web.

3.1 Web-semp: Um método para engenharia de requisitos em aplicações web

(ZANIRO; FABBRI, 2015) apresentam o método *Web-SEMP*, voltado para o levantamento e modelagem de requisitos em aplicações Web. O método se baseia na definição de objetivos e casos de uso para guiar o processo de elicitação, buscando reduzir ambiguidades e retrabalho durante o desenvolvimento. A proposta enfatiza a necessidade de considerar as peculiaridades do ambiente Web, como múltiplos stakeholders, prazos reduzidos e a natureza dinâmica das aplicações.

O estudo demonstra que práticas formalizadas podem auxiliar na transformação de necessidades informais em especificações mais claras e estruturadas. No entanto, o método exige um nível significativo de disciplina e adaptação da equipe, o que pode ser um desafio em projetos com recursos limitados. Ainda assim, o Web-SEMP contribui para o avanço das práticas de requisitos, destacando-se como um modelo aplicável em cenários que demandam maior controle e clareza no início do desenvolvimento.

3.2 Uso de campos semânticos para redução de ambiguidades em requisitos de software

(SILVA *et al.*, 2020) propõem uma ferramenta baseada em campos semânticos para apoiar o levantamento de requisitos, com o objetivo de reduzir ambiguidades na comunicação entre desenvolvedores e clientes. A abordagem utiliza ontologias e mapeamentos semânticos para organizar conceitos, permitindo uma documentação mais consistente e de fácil interpretação.

Embora o trabalho não seja restrito ao contexto Web, ele aborda uma das maiores dificuldades relatadas pelos profissionais: a correta interpretação das necessidades dos stakeholders.

A ferramenta facilita a organização de requisitos e minimiza divergências de entendimento ao longo do projeto, mostrando-se especialmente útil em ambientes onde há diversidade de perfis entre clientes e equipe técnica. Apesar de seu potencial, o estudo não discute amplamente sua aplicação em projetos ágeis ou em cenários com mudanças frequentes, o que pode ser uma limitação prática.

3.3 Plataforma colaborativa para engenharia de requisitos

(ARAÚJO, 2024) desenvolveu uma plataforma colaborativa com foco em apoiar o levantamento e o refinamento de requisitos. A solução permite que stakeholders e membros da equipe técnica interajam diretamente, reduzindo ruídos de comunicação e promovendo maior alinhamento entre as expectativas dos envolvidos.

A plataforma foi avaliada em um ambiente controlado, demonstrando ganhos em clareza e organização das informações. Assim como o trabalho de Silva et al., essa proposta não se limita ao desenvolvimento Web, mas aborda aspectos recorrentes nesse contexto, como a necessidade de comunicação efetiva e documentação acessível.

Tabela 1 – Comparação entre os trabalhos relacionados e a pesquisa desenvolvida

Aspecto	**(ZANIRO; FABBRI, 2015)**	**(SILVA *et al.*, 2020)**	**(ARAÚJO, 2024)**	**Presente Estudo (2025)**
Objetivo	Apresentar o método *Web-SEMP*, com o intuito de formalizar o levantamento e a modelagem de requisitos em projetos Web, buscando reduzir ambiguidades e retrabalho.	Criar uma ferramenta baseada em campos semânticos que organize conceitos e reduza falhas de comunicação entre desenvolvedores e clientes, garantindo maior consistência na documentação.	Desenvolver uma plataforma colaborativa para facilitar a comunicação entre stakeholders e apoiar o levantamento incremental e contínuo de requisitos.	Investigar, a partir de experiências reais, como profissionais realizam o levantamento de requisitos em aplicações Web, identificando práticas, desafios e influências sobre decisões técnicas.
Metodologia	Utilização de um método estruturado, guiado por definição de objetivos e casos de uso, aplicado em um estudo de caso para validação de sua eficácia.	Desenvolvimento e avaliação de uma ferramenta apoiada em ontologias e mapeamentos semânticos, aplicada a cenários de elicitação para organizar os requisitos de maneira clara.	Prototipação e implementação de um sistema colaborativo, validado por meio de testes com usuários e cenários simulados.	Pesquisa qualitativa baseada em entrevistas semiestruturadas com 19 profissionais atuantes, seguida de análise temática das respostas e representação gráfica com nuvens de palavras.
Foco	Ênfase na padronização e formalização de processos de elicitação, garantindo maior controle sobre a documentação e redução de retrabalho.	Centralização na clareza semântica e na melhoria da comunicação entre stakeholders, buscando minimizar interpretações equivocadas.	Facilitar a colaboração entre diferentes perfis de usuários e desenvolvedores, promovendo rastreabilidade e flexibilidade em ambientes ágeis.	Exploração prática das técnicas utilizadas, desafios enfrentados e formas como os requisitos impactam as escolhas tecnológicas no desenvolvimento Web atual.
Limitações	Requer alto nível de disciplina e treinamento da equipe, tornando-se menos adaptável a projetos ágeis ou com mudanças frequentes de escopo.	Não aborda de forma detalhada a aplicação em ambientes dinâmicos ou iterativos, como metodologias ágeis, limitando-se a cenários mais estáticos.	A proposta ainda carece de avaliação em larga escala ou aplicação em projetos reais de grande porte.	A análise se restringe a um grupo reduzido de participantes e não apresenta uma proposta metodológica ou ferramenta concreta, concentrando-se na observação empírica.
Contribuição	Oferece um método formal aplicável para elicitação de requisitos, fortalecendo a documentação e minimizando ambiguidades no início do projeto.	Propõe uma solução tecnológica que melhora a clareza e a organização dos requisitos, auxiliando na comunicação entre perfis distintos de usuários e desenvolvedores.	Apresenta uma ferramenta prática que apoia o levantamento contínuo de requisitos com foco em colaboração e adaptação a mudanças.	Fornece uma visão prática e atualizada das práticas de levantamento de requisitos em projetos Web, destacando boas práticas, dificuldades recorrentes e aprendizados do mercado.

Fonte: Elaborado pela autora.

3.4 Análise e comparação dos trabalhos

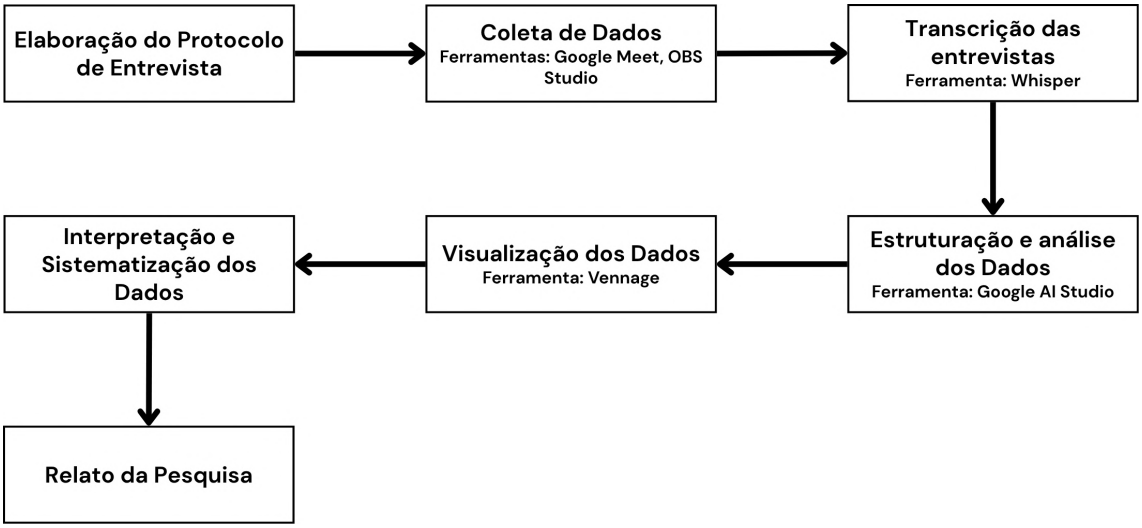
Os trabalhos relacionados analisados neste estudo oferecem perspectivas complementares sobre o levantamento de requisitos. O método *Web-SEMP* (ZANIRO; FABBRI, 2015) propõe uma abordagem formalizada com foco na estruturação e modelagem dos requisitos desde o início do projeto. A ferramenta baseada em campos semânticos (SILVA *et al.*, 2020) busca reduzir ambiguidades na comunicação entre stakeholders por meio de organização conceitual. A plataforma colaborativa desenvolvida por Araújo (ARAÚJO, 2024) enfatiza a construção contínua e coletiva de requisitos, destacando o envolvimento dos usuários ao longo do processo. Diferentemente dessas abordagens, o presente trabalho adota um enfoque empírico, centrado na análise de experiências reais de desenvolvedores e analistas, oferecendo uma visão prática e atualizada sobre os desafios enfrentados, as boas práticas adotadas e os fatores que influenciam decisões técnicas no desenvolvimento de aplicações Web.

A Tabela 1 resume as principais características dos trabalhos relacionados discutidos nesta seção, comparando-os com a presente pesquisa. Cada linha representa um aspecto analisado — objetivo, metodologia, foco, limitações e contribuição — enquanto as colunas apresentam como cada estudo abordou esses elementos. Essa comparação permite evidenciar as diferenças entre métodos formais, ferramentas baseadas em conceitos semânticos e a abordagem empírica adotada neste trabalho, destacando como cada proposta contribui, de forma distinta, para o avanço do levantamento de requisitos em aplicações Web.

4 METODOLOGIA

A presente pesquisa adotou uma abordagem qualitativa, baseada na realização de entrevistas semiestruturadas com profissionais atuantes no desenvolvimento de aplicações Web. O objetivo foi compreender como ocorre o levantamento de requisitos nesse contexto, explorando os desafios enfrentados, as práticas utilizadas e a influência dos requisitos nas decisões técnicas ao longo do projeto. A Figura 14 apresenta a representação gráfica da metodologia adotada neste trabalho. O diagrama ilustra o fluxo sequencial das etapas percorridas, desde a elaboração do protocolo de entrevista até a sistematização e escrita final do trabalho. Cada atividade reflete uma fase da pesquisa, evidenciando como os dados foram coletados, processados, analisados e transformados em resultados.

Figura 14 – Fluxo metodológico da pesquisa



Fonte: Produzido pela autora.

4.1 Elaboração do Protocolo de Entrevista

Foi elaborado um roteiro de entrevista[1] estruturado com base em quatro questões de pesquisa principais, cada uma abordando um aspecto específico do levantamento de requisitos em aplicações Web: métodos utilizados, desafios enfrentados, influência dos requisitos nas escolhas tecnológicas e boas práticas adotadas. Antes da realização das entrevistas definitivas, foi conduzida uma entrevista piloto com o objetivo de testar a clareza e a aplicabilidade das

[1] O protocolo e o roteiro completos da entrevista estão disponíveis em: <https://docs.google.com/document/d/e/2PACX-1vQ0ghAUXqA3RAjZZWbhghUF-yFy1dzk_8xM7_HJo39B1iXQ5OCsLw8UkH2To3BQjRq08YBOk8r1miYP/pub>

perguntas. Essa etapa permitiu identificar e ajustar questões que poderiam gerar ambiguidades, garantindo que o roteiro final estivesse mais objetivo e alinhado aos objetivos da pesquisa. As Questão de Pesquisa (QP) que guiaram esse estudo são:

- **QP1: Como é realizado o levantamento de requisitos em projetos de aplicações Web, e quais são os tipos de requisitos mais comuns nesse contexto?**
- **QP2: Quais são os principais desafios enfrentados por desenvolvedores e analistas durante o levantamento e a documentação de requisitos em aplicações Web?**
- **QP3: De que forma os requisitos influenciam na escolha de tecnologias como bibliotecas, frameworks e ferramentas no desenvolvimento de aplicações Web?**
- **QP4: Quais boas práticas têm sido adotadas pelos profissionais para melhorar o levantamento de requisitos em aplicações Web?**

Cada pergunta foi acompanhada, no documento orientador da entrevista, por um objetivo específico e uma explicação com exemplos, garantindo clareza tanto para a entrevistadora quanto para o entrevistado. Esse formato permitiu maior naturalidade na condução das entrevistas, ao mesmo tempo em que assegurou alinhamento com os objetivos da pesquisa.

4.2 Coleta de Dados

A coleta de dados foi realizada entre os meses de junho e julho de 2025, com a participação de 19 profissionais com experiência em desenvolvimento de aplicações Web. A amostragem dos participantes foi do tipo *por conveniência*, ou seja, selecionados a partir da facilidade de acesso e disponibilidade para participação, considerando sua atuação na área, sem delimitação prévia de cargo ou especialidade. As entrevistas foram conduzidas de forma remota, via Google Meet[2], com duração média entre 15 e 30 minutos. As conversas foram gravadas com o auxílio do software OBS Studio[3], com o consentimento dos participantes, para posterior transcrição e análise. Após a entrevista, os participantes responderam a um formulário complementar, via Google Forms, que coletou informações de perfil como nível de escolaridade, cargo atual, tempo de experiência com desenvolvimento Web e os contextos em que atuaram.

[2] Google Meet é uma plataforma de videoconferência desenvolvida pelo Google, que permite reuniões online, disponível em: <https://meet.google.com>

[3] OBS Studio é um software livre e de código aberto para gravação de vídeo e transmissão ao vivo, disponível em: <https://obsproject.com>

4.3 Transcrição das Entrevistas

As transcrições das entrevistas foram realizadas com o apoio da ferramenta `Whisper`[4], executada localmente em um ambiente Python. Para isso, foi utilizado o Código Fonte 4:

Código-fonte 4 – Script para transcrição automática com whisper em Python

```
import os
import subprocess
import sys

pasta = r"C:\Users\Felipe\Desktop\Entrevistas"

python_venv = sys.executable

for nome_arquivo in os.listdir(pasta):
    if nome_arquivo.lower().endswith(".mp4") or nome_arquivo.lower().
        endswith(".wav"):
        nome_base = os.path.splitext(nome_arquivo)[0]
        caminho_arquivo = os.path.join(pasta, nome_arquivo)
        print(f"\n Transcrevendo: {nome_arquivo}...")
        comando = [
            python_venv,
            "-m", "whisper",
            caminho_arquivo,
            "--model", "small",
            "--language", "Portuguese",
            "--output_format", "txt",
            "--output_dir", pasta
        ]
        resultado = subprocess.run(comando)
        if resultado.returncode == 0:
            print(f"Transcricao salva: {nome_base}.txt")
        else:
            print(f"Erro ao transcrever: {nome_arquivo}")
print("\nTodas as transcricoes foram processadas!")
```

[4] Whisper é um modelo de reconhecimento automático de fala (ASR) de código aberto desenvolvido pela OpenAI, capaz de transcrever e traduzir áudios, disponível em: <https://github.com/openai/whisper>

Esse script percorre todos os arquivos com extensão `.mp4` ou `.wav` contidos na pasta indicada e, para cada um deles, executa o modelo `whisper` na linguagem portuguesa utilizando o modelo leve (`small`). A saída gerada é um arquivo de texto contendo a transcrição da entrevista. O uso do ambiente virtual do Python (`sys.executable`) garante que o script funcione dentro do ambiente previamente configurado com as dependências necessárias.

4.4 Estruturação e Análise dos Dados

As entrevistas transcritas foram analisadas qualitativamente por meio de categorização temática. As respostas foram agrupadas com base em temas recorrentes e expressões equivalentes, buscando identificar padrões, boas práticas e dificuldades relatadas. A organização dos dados contou com o apoio da ferramenta *Google AI Studio*[5], utilizada para auxiliar na padronização semântica, identificação de sinônimos e agrupamento de termos similares.

4.5 Visualização dos Dados

Para facilitar a visualização e interpretação dos padrões nas respostas, foram criadas nuvens de palavras para cada pergunta do roteiro, representando a frequência relativa dos termos mencionados pelos entrevistados. As imagens foram elaboradas por meio da plataforma *Venngage*[6], com inserção manual dos dados extraídos da análise. Os gráficos referentes ao perfil dos participantes também foram gerados com base no formulário do *Google Forms*[7].

4.6 Interpretação e Sistematização dos Resultados

A partir da análise temática, foram identificadas categorias interpretativas e insights relevantes sobre o processo de levantamento de requisitos em aplicações Web. Os resultados foram discutidos à luz dos objetivos da pesquisa, estruturando-se a apresentação dos achados por pergunta e bloco temático, com exemplos reais extraídos das entrevistas.

---

[5] Google AI Studio é uma plataforma online que permite a desenvolvedores e profissionais de dados prototiparem e interagirem com modelos de IA generativa, como o Gemini, através de uma interface visual. Disponível em: <https://aistudio.google.com>

[6] Venngage é uma ferramenta online para criação de infográficos, gráficos e outros materiais visuais, voltada especialmente para apresentação de dados e comunicação visual. Disponível em: <https://infograph.venngage.com>.

[7] Google Forms é uma ferramenta gratuita do Google que permite criar formulários para coleta de dados, enquetes, avaliações e pesquisas online. Disponível em: <https://forms.google.com>.

4.7 Relato de Pesquisa

Por fim, os dados analisados e sistematizados foram incorporados ao desenvolvimento deste trabalho, compondo os capítulos de resultados, discussão e conclusão. A estrutura do TCC visa oferecer uma visão abrangente sobre as práticas atuais de levantamento de requisitos no contexto do desenvolvimento Web, com base em experiências reais de profissionais da área.

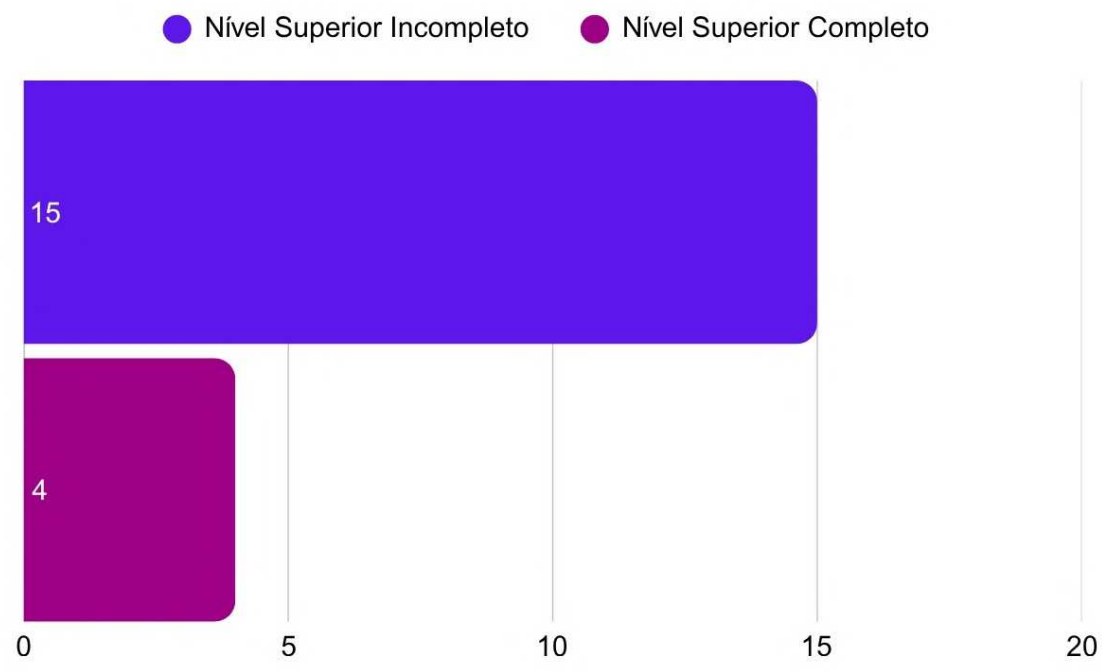
5 RESULTADOS

5.1 Perfil dos Entrevistados

A pesquisa foi composta por entrevistas com 19 profissionais atuantes no desenvolvimento de aplicações Web. Após a etapa de entrevistas, os participantes responderam a um formulário complementar, elaborado por meio da ferramenta *Google Forms*[1], com o objetivo de caracterizar o perfil dos respondentes. Os dados coletados estão organizados em gráficos e discutidos a seguir.

Em relação ao nível de escolaridade, na Figura 15, observa-se que a maioria dos entrevistados está cursando o ensino superior (15 participantes, cerca de 79%), enquanto apenas 4 profissionais (21%) já concluíram a graduação. Esse resultado revela um perfil majoritariamente jovem, em formação ou início de carreira.

Figura 15 – Distribuição dos entrevistados por nível de escolaridade



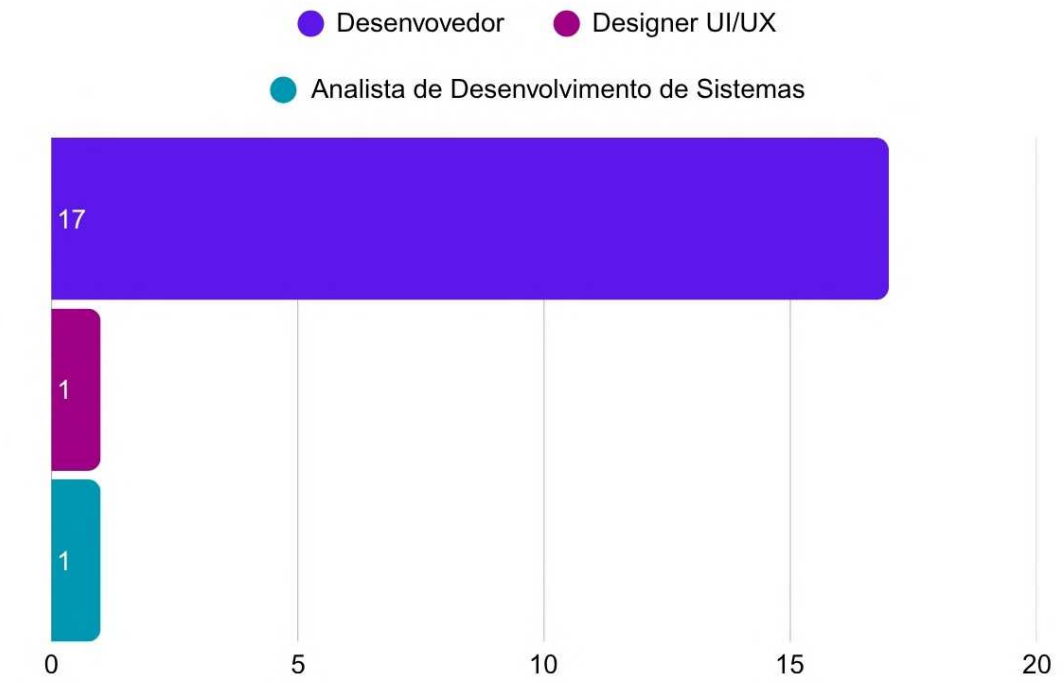
Fonte: Produzido pela autora.

Quanto ao cargo ocupado, a ampla maioria dos participantes se identifica como desenvolvedor(a), representando 17 dos 19 entrevistados (cerca de 89%), como demonstrado na Figura 16. Os demais atuam como analista de desenvolvimento de sistemas (1) e designer

[1] Google Forms é uma ferramenta gratuita do Google que permite criar formulários para coleta de dados, enquetes, avaliações e pesquisas online. Disponível em: <https://forms.google.com>.

UI/UX (1), o que indica uma predominância do foco técnico na amostra, ainda que haja alguma diversidade de funções.

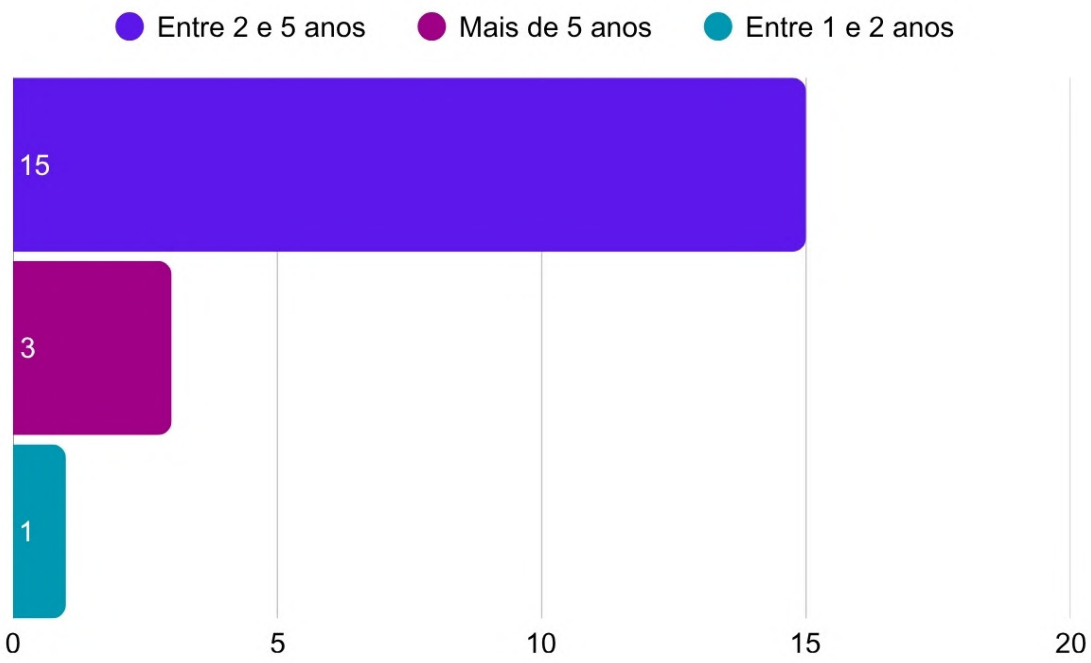
Figura 16 – Distribuição dos entrevistados por cargo ocupado



Fonte: Produzido pela autora.

A Figura 17 se refere ao tempo de atuação com desenvolvimento de aplicações Web. A maioria dos participantes possui entre 2 e 5 anos de experiência (15 profissionais, aproximadamente 79%). Três entrevistados (16%) atuam há mais de 5 anos, enquanto apenas um (5%) declarou ter entre 1 e 2 anos de experiência.

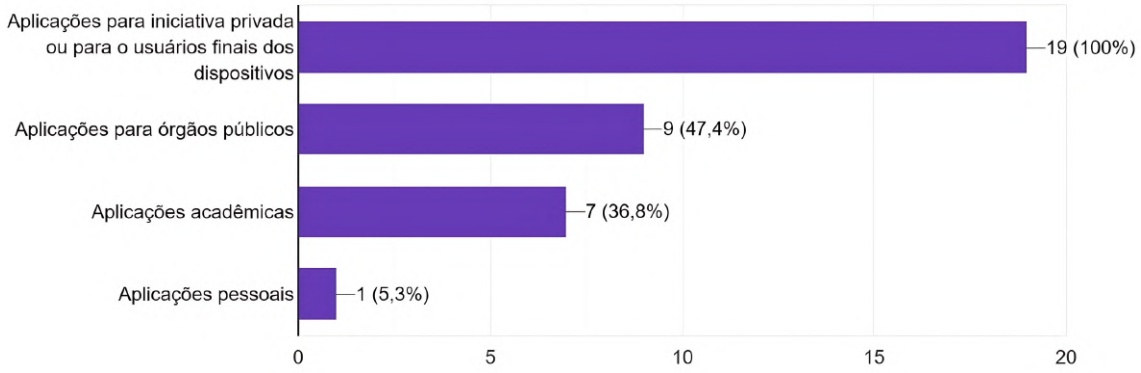
Figura 17 – Tempo de atuação em atividades relacionadas ao desenvolvimento Web



Fonte: Produzido pela autora.

Na Figura 18 é demonstrado os contextos nos quais os entrevistados já atuaram com desenvolvimento Web, em que puderam selecionar múltiplas opções. Todos os participantes afirmaram ter trabalhado com aplicações voltadas à iniciativa privada ou diretamente para usuários finais. Nove entrevistados relataram experiência com aplicações para órgãos públicos, sete atuaram em projetos acadêmicos e um desenvolveu aplicações pessoais.

Figura 18 – Contextos de atuação no desenvolvimento de aplicações Web (respostas múltiplas)



Fonte: Produzido pela autora.

As entrevistas realizadas com os participantes foram estruturadas em quatro blocos temáticos, com o objetivo de abordar diferentes dimensões do levantamento e aplicação de

requisitos em sistemas Web. Cada bloco foi composto por três perguntas abertas, permitindo que os entrevistados compartilhassem suas experiências de forma livre e contextualizada.

- **Bloco 1 – Entendimento Geral:** investigou a visão inicial dos desenvolvedores sobre os pedidos mais comuns dos clientes e usuários, além das demandas recorrentes no início de projetos Web.
- **Bloco 2 – Requisitos Específicos de Aplicações Web:** explorou como os participantes identificam e tratam requisitos específicos desse tipo de aplicação, como responsividade, acessibilidade, segurança e desempenho.
- **Bloco 3 – Influência dos Requisitos na Escolha de Tecnologias:** buscou compreender como os requisitos levantados impactam na definição das tecnologias utilizadas nos projetos.
- **Bloco 4 – Experiência Profissional com Requisitos:** abordou aspectos mais subjetivos e reflexivos, como mudanças na forma de lidar com requisitos ao longo da carreira, dificuldades enfrentadas e aprendizados adquiridos.

Para a análise qualitativa das respostas abertas, adotou-se uma abordagem de categorização temática e agrupamento semântico dos termos mais recorrentes. O objetivo foi identificar padrões de resposta, facilitando a compreensão dos aspectos mais relevantes apontados pelos participantes.

O agrupamento das palavras-chave foi realizado com o apoio da ferramenta *Google AI Studio*, utilizada como suporte na organização textual, padronização de termos e identificação de sinônimos. Essa etapa permitiu consolidar expressões equivalentes e estruturar os dados de forma mais uniforme, respeitando a integridade das falas dos entrevistados.

A visualização dos dados foi realizada por meio de nuvens de palavras, que representam graficamente a frequência relativa dos termos identificados. As nuvens foram elaboradas utilizando a plataforma *Venngage*, que permite a inserção manual dos dados, garantindo maior controle visual sobre a proporcionalidade dos elementos apresentados.

5.2 Bloco 1 – Entendimento Geral

5.2.1 Q1: Quais são as coisas mais comuns que os clientes ou usuários costumam pedir quando você está começando um sistema web?

A Figura 19 apresenta a nuvem de palavras gerada a partir da análise qualitativa das respostas dos participantes. A partir dela, é possível identificar padrões temáticos que indicam as principais preocupações dos clientes no início do desenvolvimento de um sistema Web.

Figura 19 – Nuvem de palavras da análise da questão de entrevista 01

Fonte: Produzido pela autora.

A seguir, serão abordados os tópicos mais relevantes discutidos a partir dessa pergunta e que se refletem na nuvem de palavras da Figura 19.

5.2.1.1 Aspectos Visuais e Interface (UI/UX)

A preocupação mais evidente entre os entrevistados refere-se à apresentação visual do sistema, possuindo um total de 28 menções. Muitos relataram que os clientes têm forte necessidade de visualizar o andamento do projeto, solicitando desde o início protótipos, esboços ou modelos visuais, como telas no *Figma*[2] ou *wireframes*. Esse desejo está fortemente ligado à

[2] Figma é uma ferramenta colaborativa baseada em nuvem para design de interfaces e prototipagem. Disponível em: <https://www.figma.com>

percepção de progresso. Um entrevistado afirma: *"Percebo que a maioria sempre precisa de um retorno visual. Por exemplo, se eu começar pelo backend do projeto, parece que não estou fazendo nada"*. Outro complementa: *"Eles insistem que seja visualmente agradável para que os usuários tenham prazer em acessar a plataforma"*.

5.2.1.2 Prazo, Tempo e Velocidade de Entrega

O tempo de entrega é um tema recorrente nas falas, possuindo um total de 22 menções. Muitos clientes demonstram ansiedade para obter resultados rápidos, mesmo sem compreender totalmente a complexidade técnica envolvida. Essa expectativa por agilidade pode pressionar o desenvolvedor desde o início. Como destacou um participante: *"Quanto tempo vai demorar pra fazer isso?" é quase sempre a primeira pergunta"*. Outro relata: *"O principal pedido é esse: rapidez"*.

5.2.1.3 Definição de Requisitos e Escopo

Muitos entrevistados apontaram que os clientes chegam com ideias vagas ou mal estruturadas, exigindo esforço adicional por parte do desenvolvedor para compreender e transformar essas ideias em requisitos claros e executáveis, possuindo um total de 18 menções. Um entrevistado relatou: *"O cliente geralmente não tem essa capacidade de detalhar o que quer. Às vezes ele diz 'quero um sistema que funcione', mas não sabe dizer o que isso inclui"*.

5.2.1.4 Funcionalidades e Características Técnicas

Alguns entrevistados destacaram solicitações específicas por funcionalidades técnicas, possuindo um total de 16 menções. Dentre as mais citadas estão: login/autenticação de usuários, dashboards com visualização de dados, responsividade para dispositivos móveis e geração de relatórios. Como explicou um dos participantes: *"Hoje em dia, os clientes querem que tudo funcione tanto no celular quanto no computador. Responsividade é básico"*. Outro acrescenta: *"Quase sempre pedem login, dashboards com gráficos e relatórios exportáveis"*.

5.2.1.5 Custo e Orçamento

Embora não tenha sido o tema mais citado, tendo um total de 4 menções, o orçamento é um ponto que aparece principalmente no início da conversa com o cliente, geralmente vinculado

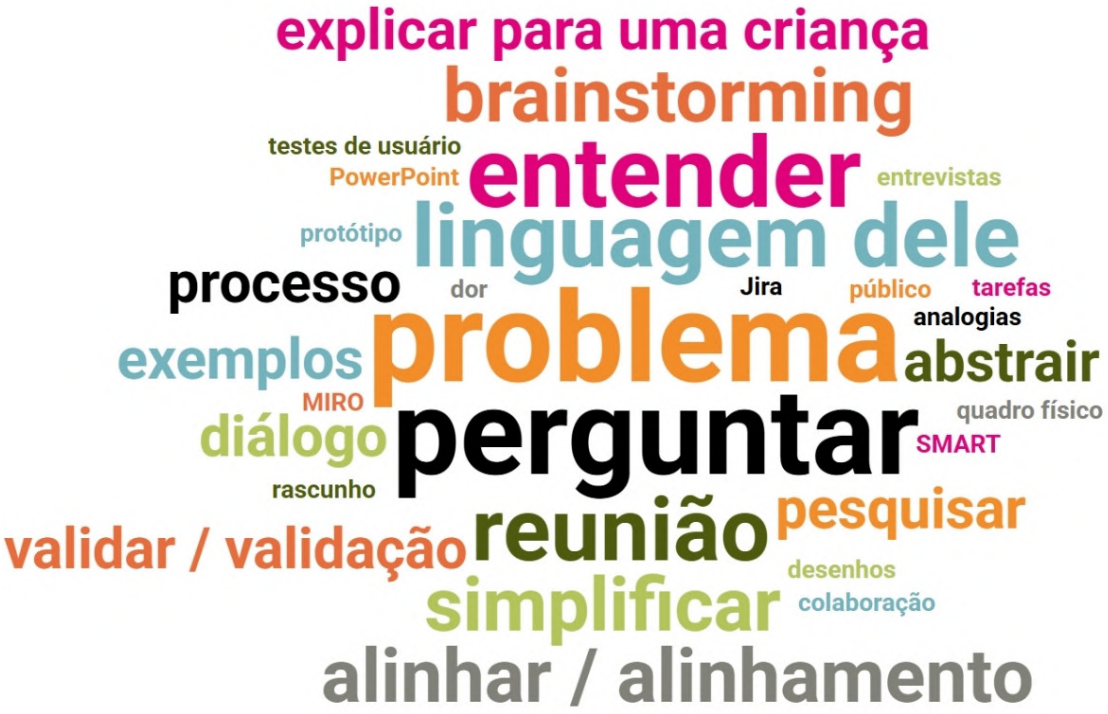
à questão do tempo. Um desenvolvedor relatou: *"Eles perguntam primeiro o orçamento e quanto tempo vai levar"*, indicando que essas variáveis são muitas vezes tratadas como interdependentes.

Resposta da Q1: A análise das entrevistas indica que, no início de um projeto Web, os clientes tendem a priorizar aspectos visuais e de interface como forma de acompanhar o progresso, exigindo protótipos e modelos visuais desde as primeiras etapas. A pressão por prazos curtos e entregas rápidas é evidente, ainda que muitas vezes descolada da realidade técnica do desenvolvimento. Soma-se a isso a dificuldade de explicitação de requisitos, o que exige do desenvolvedor um papel ativo na tradução de ideias em escopos formais. Por fim, aspectos técnicos como responsividade e funcionalidades específicas aparecem com frequência, enquanto questões como custo e orçamento, apesar de relevantes, são tratadas de forma mais pontual.

5.2.2 Q2: Quando você vai conversar com um cliente sobre o que o sistema precisa fazer, você usa alguma técnica para ajudar a entender melhor o que ele quer?

A Figura 20 apresenta a nuvem de palavras gerada a partir das respostas dos entrevistados ao responderem a pergunta 2, permitindo visualizar os termos mais frequentes relacionados às estratégias utilizadas para compreender as demandas dos clientes. A imagem reflete a diversidade de abordagens empregadas, desde métodos formais até práticas comunicativas informais, mas eficazes.

Figura 20 – Nuvem de palavras da análise da questão de entrevista 02



Fonte: Produzido pela autora.

A maioria dos desenvolvedores entrevistados afirma empregar alguma técnica ou abordagem específica, mesmo que de forma intuitiva, para extrair e interpretar os requisitos dos clientes. Dos 19 participantes:

- **Sim (usa técnicas ou processos estruturados):** 16 entrevistados
- **Não (não usa ou não tem contato com o cliente):** 3 entrevistados

Curiosamente, mesmo entre os que inicialmente negam o uso de uma técnica, muitos acabam descrevendo práticas que se enquadram como métodos conhecidos de comunicação e levantamento de requisitos, como o uso de analogias, perguntas estratégicas e ferramentas de organização visual. A análise das respostas levou à identificação de quatro grandes categorias temáticas:

5.2.2.1 Investigação e Foco no Problema Central

Esta foi a categoria mais recorrente, tendo um total de 28 menções. Os desenvolvedores relatam que preferem investigar a origem do problema, entendendo o contexto e os objetivos reais do cliente antes de pensar em soluções. A estratégia está centrada no "porquê" da demanda. Um participante relata: *"Geralmente, eu começo sempre perguntando qual é o problema que o cliente quer resolver. Porque, como eu te falei, ele sabe o que quer, mas não necessariamente a*

maneira que ele está me explicando vai resolver o problema dele".

5.2.2.2 Comunicação Simplificada e Empática

Muitos entrevistados destacaram a necessidade de adaptar a linguagem ao nível de compreensão do cliente, evitando termos técnicos e utilizando exemplos do cotidiano. Esse aspecto foi mencionado 21 vezes, evidenciando sua relevância na comunicação entre as partes. A simplificação do discurso técnico facilita a comunicação e fortalece a confiança do cliente. Um desenvolvedor resume essa abordagem ao dizer: *"Eu tive um professor na faculdade que dizia que a gente tinha que explicar as coisas como se fosse para uma criança de cinco anos. Isso me marcou muito, então tento sempre fazer isso com os clientes"*.

5.2.2.3 Técnicas e Ferramentas Formais de Planejamento

Um grupo significativo dos participantes mencionou, em 16 ocorrências ao longo das entrevistas, o uso de metodologias específicas e ferramentas visuais para organizar as ideias, como *brainstorming*, diagramas UML *(Unified Modeling Language)*, plataformas de colaboração ou mesmo apresentações em *PowerPoint*. Essas práticas ajudam a materializar os requisitos e alinhar expectativas. Um dos entrevistados afirma: *"Gosto de usar essa técnica [SMART] para deixar claro qual é exatamente o objetivo ou o requisito. E quando vou apresentar ideias, gosto muito de usar um diagrama simples de caso de uso"*.

5.2.2.4 Processos Colaborativos e Validação

Alguns desenvolvedores enfatizaram a importância de envolver o cliente e a equipe ao longo do processo, tema que apareceu 14 vezes nas falas analisadas. A realização de reuniões, validações intermediárias e testes com usuários são estratégias recorrentes para manter o alinhamento e refinar os requisitos conforme o projeto evolui. Um entrevistado descreve: *"A gente reúne o time — eu, o PM (Gerente de Projeto), o designer e os stakeholders — e juntos refinamos esse problema, criamos um fluxo ou protocolo para resolvê-lo. Depois disso, apresentamos essa proposta ao cliente antes de começar o desenvolvimento"*.

Resposta da Q2: Embora nem todos os entrevistados tenham nomeado formalmente uma técnica, a grande maioria adota práticas consistentes e intencionais para compreender as expectativas dos clientes. As estratégias vão desde perguntas investigativas sobre o problema até a utilização de ferramentas colaborativas e métodos de validação contínua. A simplificação da linguagem e o esforço para gerar empatia também aparecem como elementos centrais na construção de um entendimento mútuo entre desenvolvedor e cliente.

5.2.3 *Q3: Nos sistemas Web que você trabalhou, já teve algum problema porque algo combinado com o cliente não estava registrado?*

A Figura 21 apresenta a nuvem de palavras gerada a partir das respostas dos entrevistados, destacando as principais expressões relacionadas aos problemas decorrentes da falta de formalização de acordos com os clientes. A imagem evidencia a relevância de termos como *registrado*, *contrato*, *documentado*, *escopo* e *combinado*, sugerindo que o tema da documentação está diretamente associado a falhas de comunicação, retrabalho e conflitos recorrentes em projetos de sistemas Web.

Figura 21 – Nuvem de palavras da análise da questão de entrevista 03

Fonte: Produzido pela autora.

A análise das entrevistas revelou que a ausência de registros formais é uma causa

amplamente reconhecida de problemas. Em números:

- **Sim (já enfrentaram esse problema):** 16 entrevistados
- **Não (não enfrentaram ou relataram problemas relacionados):** 3 entrevistados

Mesmo entre os que afirmaram não ter passado por situações críticas, é comum encontrar menções a falhas de comunicação ou mudanças de escopo não documentadas. Assim, a falta de alinhamento entre expectativa e entrega aparece como um desafio transversal. A partir da análise qualitativa, quatro categorias foram identificadas:

5.2.3.1 Causa Principal: Falta de Registro Formal

A categoria mais citada, mencionada 33 vezes pelos participantes, diz respeito à ausência de documentação clara. Acordos verbais ou registros informais são apontados como fontes recorrentes de conflito. Um participante relata: *"O problema foi que isso ficou somente no verbal e não registrado em contrato, o que gerou um problema grande para a empresa"*. Outro menciona: *"O cliente fala algo em uma reunião ou por mensagem, mas isso não fica registrado por escrito"*.

5.2.3.2 Ocorrência Recorrente do Problema

O problema não é pontual. Muitos desenvolvedores relatam que situações desse tipo são comuns, frequentes e, muitas vezes, esperadas no cotidiano do desenvolvimento Web. Um dos entrevistados desabafa: *"Acho que é um dos motivos de maior dor de cabeça hoje: eles cobrarem coisas que não estão combinadas"*.

5.2.3.3 A Dinâmica do Problema: Escopo Oculto e Mudanças

Em muitos relatos, o problema, mencionado 19 vezes, se manifesta quando o cliente apresenta uma ideia nova ou cobra algo que não foi previamente acordado. Isso gera retrabalho, conflitos sobre valores e atrasos nas entregas. Um exemplo típico foi descrito por um entrevistado: *"O cliente sempre vinha com uma ideia nova [...] que mudava completamente o escopo da sprint e ele não queria pagar a diferença"*.

5.2.3.4 Soluções Mencionadas: Prevenção e Formalização

Para evitar esse tipo de problema, alguns desenvolvedores relataram boas práticas de prevenção, como o uso de ferramentas de gestão (Trello[3], Jira), comunicação formal por e-mail e validação contínua do que foi acordado. Essas estratégias, mencionadas 10 vezes, são vistas como fundamentais para manter o alinhamento com o cliente. Um participante explica: *"O certo é registrar todos os requisitos e validar. Chegar para eles e dizer assim: 'O combinado é esse, esse e esse'"*. Outro complementa: *"Deixar tudo descrito por card, ou se alguma coisa foi combinada por fora, passar em e-mail [...] para deixar registrado"*.

Resposta da Q3: A falta de registro formal de decisões é um dos principais fatores de conflito entre desenvolvedores e clientes no contexto do desenvolvimento Web. Situações de escopo indefinido, mudanças informais e cobranças indevidas por funcionalidades não previstas geram retrabalho e desgaste. A documentação clara e a validação contínua aparecem como estratégias fundamentais para mitigar esses riscos e manter o alinhamento durante o projeto.

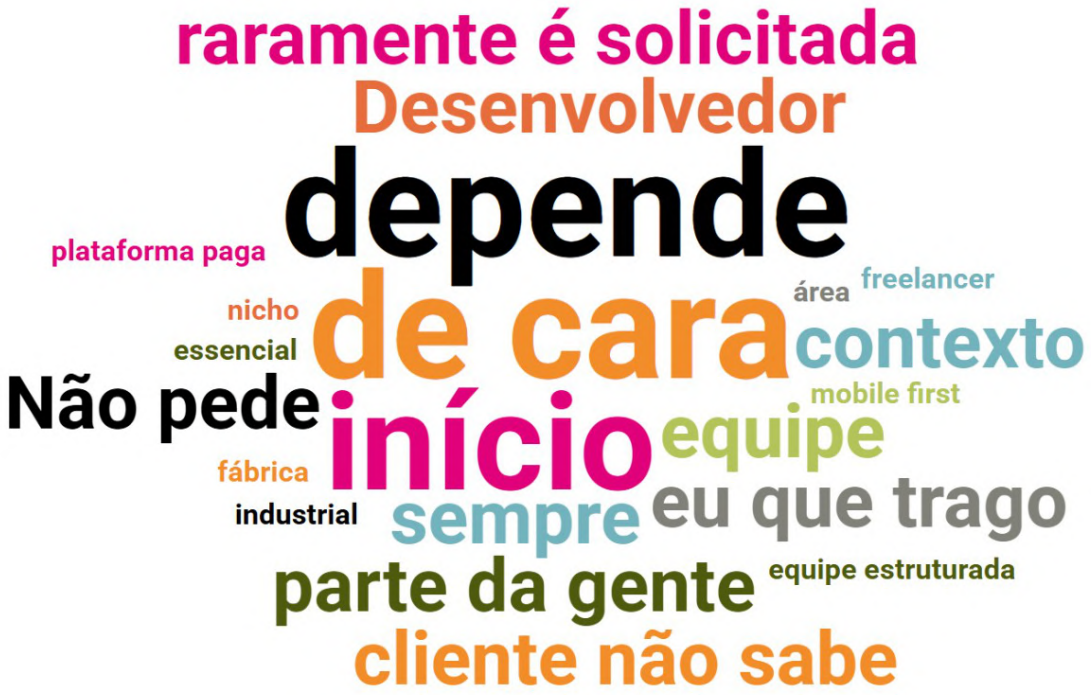
5.3 Bloco 2 – Requisitos Específicos da Web

5.3.1 Q4: Responsividade costuma ser solicitada de cara ou apenas posteriormente?

A Figura 22 apresenta a nuvem de palavras construída a partir das respostas dos entrevistados sobre a percepção da responsividade como um requisito inicial. A imagem mostra termos como *de cara*, *cliente*, *equipe*, *depende* e *mobile*, evidenciando a diversidade de experiências e a ausência de um padrão único entre os projetos relatados.

[3] Trello é uma ferramenta de gerenciamento de projetos baseada em quadros e cartões. Disponível em <https://trello.com>

Figura 22 – Nuvem de palavras da análise da questão de entrevista 04



Fonte: Produzido pela autora.

A análise revela que a responsividade, apesar de amplamente adotada como padrão técnico em sistemas Web, nem sempre é percebida pelo cliente como algo a ser solicitado logo no início. Dos 19 entrevistados:

- **Sim, é solicitada de cara:** 7 entrevistados
- **Não, é proposta pelo desenvolvedor ou surge depois:** 6 entrevistados
- **Depende do contexto:** 6 entrevistados

Essa divisão revela uma realidade prática fragmentada, em que o reconhecimento da responsividade como requisito depende mais do tipo de cliente, sistema e experiência do desenvolvedor do que de uma prática consolidada no mercado. A seguir, são apresentadas as categorias extraídas das respostas, com base nas recorrências de palavras-chave e nas falas mais representativas.

5.3.1.1 Sim, é uma Exigência Inicial

Para esse grupo, a responsividade é vista como premissa básica de qualquer sistema Web, conforme indicaram 7 entrevistados. Os clientes, mesmo que sem conhecimento técnico profundo, já chegam com essa expectativa. Um entrevistado resume bem: *"Pronto, é de cara. Normalmente, é um dos requisitos principais. Acho que nunca peguei um projeto, desde o início,*

em que não pedissem responsividade".

5.3.1.2 Não, Parte da Iniciativa do Desenvolvedor

Neste caso, conforme indicado por 6 entrevistados, os desenvolvedores relatam que o cliente raramente menciona responsividade. Muitas vezes, cabe à equipe introduzir o conceito e explicar sua importância. O cliente nem sempre entende o impacto da ausência de responsividade ou assume que será feita por padrão. Como relatado por um participante: *"Normalmente, não é o cliente que pede. Sou eu quem trago esse ponto para eles. [...] sou eu mesmo quem fala sobre isso"*. Outro reforça: *"Muitos acham que, se você fizer uma vez, já vai funcionar em tudo. Então, a gente precisa especificar"*.

5.3.1.3 Depende do Contexto do Projeto

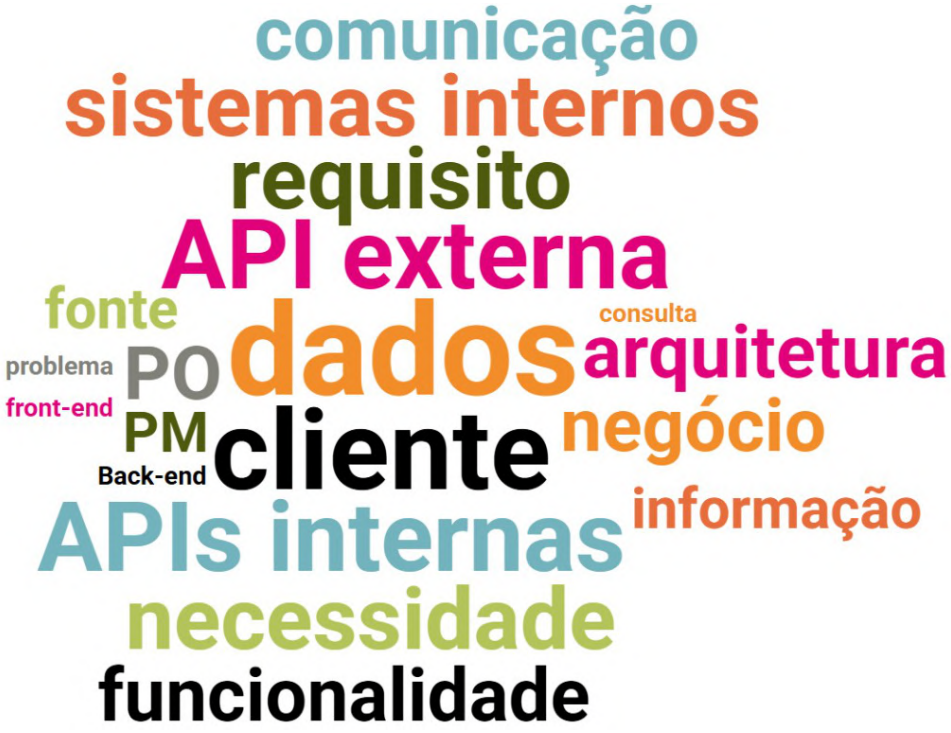
Para esse grupo, conforme relataram 6 entrevistados, a responsividade é solicitada de forma condicional. Fatores como o tipo de sistema (interno ou externo), o perfil do cliente e o nicho de atuação determinam se a demanda surgirá logo no início ou mais adiante. Um entrevistado explica: *"Em fábricas[...], os sistemas não são bonitos, eles são funcionais. [...] ninguém liga para responsividade"*. No entanto, o mesmo participante acrescenta: *"Mas se for uma plataforma privada, paga, o cliente vai querer que funcione bem"*.

Resposta da Q4: Embora a responsividade seja considerada um padrão técnico consolidado no desenvolvimento Web, sua solicitação explícita ainda depende fortemente do perfil do cliente e do tipo de sistema. Em projetos voltados ao grande público ou plataformas comerciais, ela tende a ser requisitada desde o início. Por outro lado, em sistemas internos, especialmente em contextos industriais, a preocupação com a responsividade é frequentemente negligenciada. Quando não é mencionada, cabe ao desenvolvedor assumir um papel proativo, explicando sua relevância e garantindo sua implementação mesmo sem uma demanda direta.

5.3.2 *Q5: Já participou de um projeto Web onde o sistema precisava se integrar com outros sistemas, como APIs de terceiros ou sistemas internos? Como foi identificado isso nos requisitos?*

A Figura 23 demosntra a nuvem de palavras resultante da análise das entrevistas da pergunta 05. Os termos *cliente*, *API*, *dados*, *sistema*, *integração* e *necessidade* aparecem em destaque, refletindo o caráter central das integrações no desenvolvimento Web e a variedade de gatilhos que levam à sua identificação nos requisitos.

Figura 23 – Nuvem de palavras da análise da questão de entrevista 05



Fonte: Produzido pela autora.

A análise das respostas demonstra que a necessidade de integração com outros sistemas é quase universal no desenvolvimento de sistemas Web. Dos 19 entrevistados:

- **Sim (já participaram de projetos com integração):** 18 entrevistados
- **Não (não participou da fase de implementação):** 1 entrevistado

Mesmo o único participante que respondeu negativamente declarou estar envolvido em um projeto que exigirá esse tipo de integração, embora ainda não tenha atuado diretamente na implementação. Isso indica que esse tipo de requisito faz parte da realidade prática da maioria dos profissionais da área. A seguir, apresentam-se as categorias de identificação da necessidade de integração, com base nas recorrências de palavras-chave e exemplos extraídos das entrevistas:

5.3.2.1 Identificação pela Necessidade do Negócio ou do Cliente

A forma mais comum de identificação ocorre por meio de solicitações explícitas do cliente ou do *Product Owner*, geralmente ligadas a uma funcionalidade esperada ou à continuidade de processos já existentes, totalizando 25 menções entre os entrevistados. A necessidade de integração surge diretamente da lógica de negócio. Um exemplo citado foi: *"Já teve um cliente que pediu para mapear a rota que os entregadores dele faziam"*, o que levou à integração com o *Google Maps*[4] . Outro relatou: *"Geralmente isso já vem dizendo quando eles já têm um sistema que vai fornecer aqueles dados"*.

5.3.2.2 Identificação por uma Lacuna de Dados ou Funcionalidade

Aqui, conforme indicado por 20 menções, a necessidade de integração surge a partir da percepção da equipe técnica de que certas informações não estão disponíveis no sistema atual. A solução, então, é buscar essas informações em fontes externas. Um entrevistado comentou: *"Geralmente, eu recorro a APIs quando identifico um problema que depende de dados externos, que eu sei que a empresa não vai conseguir me fornecer, e aí procuro uma API na Web"*. Outro exemplo envolveu a consulta a feriados por meio de uma API pública: *"Tinha que utilizar API. A gente só chegou para eles e falou que a implementação seria difícil [...] mas que tinha outra solução"*.

5.3.2.3 Identificação por Definição Técnica ou por Arquitetura

Conforme indicado por 17 menções, em sistemas com arquitetura mais robusta ou baseados em microsserviços, a integração é uma exigência desde o início, não dependendo de uma demanda explícita. O próprio desenho do sistema impõe a comunicação entre componentes distintos. Um desenvolvedor descreveu: *"O levantamento de requisitos basicamente tem que ser feito no início, antes de montar essas APIs, saber como elas vão se comunicar entre si"*. Outro detalhou a integração com o *Enterprise Resource Planning* (ERP) da empresa: *"A gente fazia a comunicação, tipo de comunicação REST, Simple Object Access Protocol (SOAP) também, os XMLs, e também via banco de dados"*.

[4] Google Maps é um serviço de mapeamento e navegação online que oferece visualização de mapas, rotas e informações geográficas em tempo real. Disponível em: <https://maps.google.com>

5.3.2.4 Tipos de Integrações Mais Citadas:

- **Sistemas de pagamento:** Mercado Pago[5], Moip[6], Apple In-App Purchase[7].
- **Comunicação:** APIs do WhatsApp (oficiais[8] e não-oficiais como Z-API[9] e Evolution[10]).
- **Mapas e dados geográficos:** Google Maps, APIs de CEP.
- **Sistemas internos e ERPs:** TOTVS[11], legados corporativos, integração via banco.
- **Dados públicos e informações gerais:** APIs de feriados, clima, dados governamentais.

Resposta da Q5: A integração com outros sistemas é uma prática essencial e recorrente no desenvolvimento Web, sendo considerada parte integrante do processo de levantamento de requisitos. As entrevistas revelam que essa necessidade pode emergir de diferentes formas, seja por uma solicitação direta do cliente, por uma lacuna funcional percebida pela equipe técnica ou como uma exigência de arquitetura do projeto. A variedade de integrações mencionadas reflete a complexidade dos ecossistemas digitais contemporâneos e destaca a importância de mapear corretamente essas necessidades logo nas etapas iniciais do projeto.

5.3.3 Q6: Em algum sistema que você participou, foi necessário se preocupar com segurança Web (como login seguro, proteção de dados ou ataques)? Como isso foi discutido nos requisitos? Precisou seguir normas específicas como as do LGPD?

A Figura 24 apresenta a nuvem de palavras resultante da análise das entrevistas da pergunta 06, na qual se destacam termos como *Lei Geral de Proteção de Dados (LGPD)*, *dados*, *login*, *token*, *segurança* e *autenticação*. A predominância desses termos evidencia a centralidade das preocupações com segurança web e proteção de dados nos projetos analisados.

---

[5] Mercado Pago é uma plataforma de pagamentos online do Mercado Livre que permite processamento de transações financeiras. Disponível em: <https://www.mercadopago.com.br>

[6] Moip é um serviço brasileiro de pagamentos digitais que oferece soluções para e-commerce e marketplaces. Disponível em: <https://lojaintegrada.com.br/loja-aplicativos/moip>

[7] Apple In-App Purchase é o sistema da Apple que permite compras dentro de aplicativos para *Apple iPhone Operating System* (iOS). Disponível em: <https://developer.apple.com/in-app-purchase>

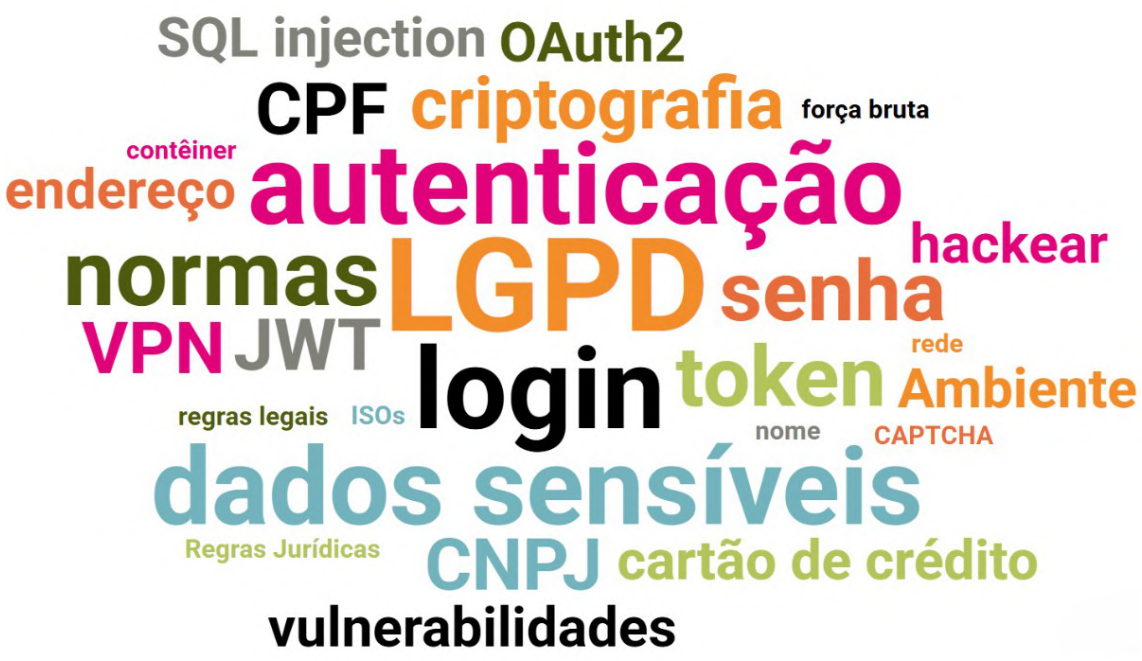
[8] WhatsApp Business API é a interface oficial para integração de mensagens com o WhatsApp, permitindo comunicação automatizada com clientes. Disponível em: <https://business.whatsapp.com/developers/developer-hub>

[9] Z-API é uma API não oficial que possibilita a automação de mensagens e integração com o WhatsApp. Disponível em: <https://www.z-api.io>

[10] Evolution API é outra solução não oficial para automação e envio de mensagens no WhatsApp. Disponível em: <https://evo-ai.co>

[11] TOTVS é uma empresa brasileira que fornece sistemas ERP para gestão corporativa, com diversas APIs para integração. Disponível em: <https://www.totvs.com>

Figura 24 – Nuvem de palavras da análise da questão de entrevista 06



Fonte: Produzido pela autora.

Com base nas respostas, nota-se que a segurança é uma preocupação amplamente reconhecida no desenvolvimento de sistemas Web:

- **Sim (se preocuparam com segurança):** 18 entrevistados
- **Não (a preocupação foi deixada para depois ou não se aplicou):** 1 entrevistado

A única resposta negativa ocorreu em um contexto de desenvolvimento extremamente acelerado, onde a implementação de medidas de segurança foi postergada, o que, segundo o próprio relato, acabou se tornando um problema. A seguir, estão os principais eixos temáticos observados, organizados por categorias de conteúdo:

5.3.3.1 LGPD como Pilar Central

A Lei Geral de Proteção de Dados (LGPD) foi a norma mais citada pelos entrevistados, aparecendo em 20 menções. Em muitos casos, ela orienta a estruturação dos requisitos desde o início do projeto, especialmente em empresas que desenvolvem sistemas voltados à adequação legal. Um entrevistado relata: *"A empresa que eu trabalho hoje é uma empresa focada em LGPD. A gente desenvolve ferramentas para que outras empresas possam se adequar à LGPD. Então, acaba que é pré-requisito para os nossos sistemas estarem adequados."*

5.3.3.2 Autenticação e Controle de Acesso

A primeira camada de segurança citada pelos entrevistados, mencionada 30 vezes, está relacionada a mecanismos de autenticação e autorização, considerados obrigatórios na maioria dos sistemas. Como um dos participantes explicou: *"Sim, normalmente a gente sempre tem que se preocupar com a segurança. [...] Também existem alguns requisitos que precisamos tratar diretamente com o cliente no desenvolvimento inicial, como os controles de acesso."*

5.3.3.3 Proteção de Dados Sensíveis

A manipulação de dados pessoais — como CPF, CNPJ, endereço, cartão de crédito e nome completo — foi citada em 20 menções e exige cuidados técnicos como criptografia, anonimização e armazenamento seguro. Um dos entrevistados descreveu: *"Trabalhamos com CPF, cartão, essas coisas sensíveis. Então, desde o início o sistema precisa seguir os padrões de proteção exigidos por lei."*

5.3.3.4 Infraestrutura e Prevenção de Ataques

Foram mencionadas 14 vezes, citando práticas e mecanismos para garantir a segurança da infraestrutura, evitar invasões e mitigar vulnerabilidades. Um dos relatos destacou: *"Houve um tempo em que a segurança não era o foco principal [...] atualmente a empresa está passando por uma mudança relacionada à segurança."*

5.3.3.5 Como a Segurança é Integrada ao Processo

- **Requisito Explícito desde o Início:** Em projetos mais robustos ou sensíveis (financeiros, jurídicos, hospitalares), a segurança é tratada como requisito essencial logo nas fases iniciais. A preocupação envolve desde a definição de escopo até a implementação de testes de penetração.
- **Responsabilidade de Especialistas ou Desenvolvedores Sêniores:** Em ambientes mais estruturados, a segurança é atribuída a times especializados ou aos profissionais mais experientes. Isso garante o uso de boas práticas, mas também pode afastar o tema do processo de levantamento de requisitos. Um participante comentou: *"Geralmente fica a cargo do sênior, do tech lead, e eles que lidam sozinhos com esses trabalhos."*
- **Preocupação Tardia ou Reativa:** Em contextos de alta pressão ou baixa maturidade, a

segurança é discutida apenas após incidentes ou auditorias, o que pode comprometer a confiabilidade do sistema. Em um dos relatos: *"Só depois que contratamos pessoas com foco em testes e segurança que começamos a revisar os processos."*

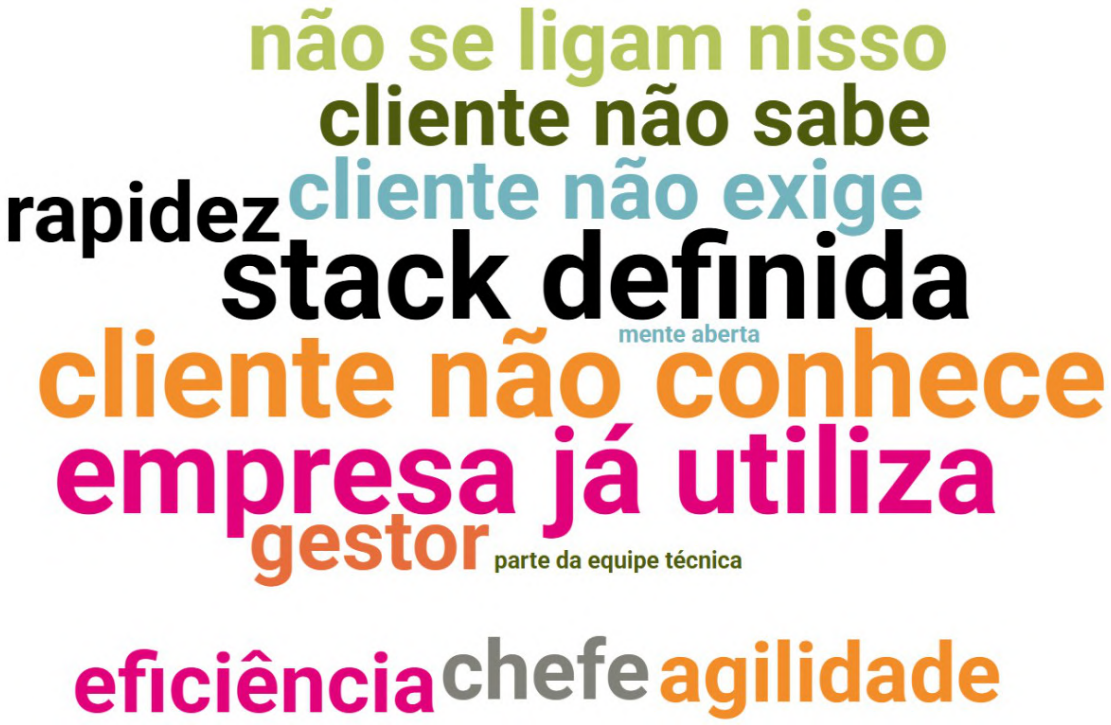
Resposta da Q6: A segurança é um elemento central nos sistemas Web, seja pela necessidade legal (como a LGPD), seja pela proteção de informações sensíveis e pela prevenção de ataques. A maioria dos entrevistados relatou experiências práticas com autenticação, criptografia e controle de acesso, ainda que a responsabilidade por essas questões nem sempre recaia sobre os analistas de requisitos. A maturidade da equipe e o contexto do projeto influenciam fortemente em quando e como a segurança é incorporada ao ciclo de desenvolvimento.

5.4 Bloco 3 – Influência dos Requisitos na Tecnologia

5.4.1 Q7: Você já escolheu uma tecnologia (como uma biblioteca ou framework) por causa de alguma exigência que o cliente fez no início?

A Figura 25 apresenta a nuvem de palavras gerada a partir das respostas dos entrevistados referente à pergunta 07. Os termos mais frequentes — como *cliente*, *tecnologia*, *escolha*, *empresa*, *stack*, *problema*, *resolver* — indicam que, embora o cliente esteja presente na discussão, a decisão sobre a tecnologia costuma estar centrada na equipe técnica ou ser determinada por políticas organizacionais.

Figura 25 – Nuvem de palavras da análise da questão de entrevista 07



Fonte: Produzido pela autora.

5.4.1.1 O Cliente Não Técnico

A maioria dos entrevistados afirmou, em 14 menções, que os clientes geralmente não possuem conhecimento técnico para sugerir tecnologias específicas. Nesses casos, o cliente foca nos resultados esperados e no custo, deixando a decisão técnica a cargo da equipe. Um dos entrevistados comenta: *"Eles só querem que funcione e entendem mais a questão dos custos."* Outro complementa: *"Normalmente, quando é para o público em geral, ninguém exige tecnologia específica."*

5.4.1.2 A Stack definida pela Empresa como Decisão Organizacional

Muitos participantes explicaram que a tecnologia usada já está padronizada pela empresa ou é definida por gestores, possuindo um total de 12 menções. Assim, mesmo que o cliente fizesse uma sugestão, ela dificilmente seria implementada se não estivesse de acordo com o stack institucional. Exemplos disso foram: *"A gente sempre resolve o problema com a tecnologia que a equipe já trabalha."* e *"Não foi uma coisa que a gente decidiu, foi lá de cima."*

5.4.1.3 A Escolha Técnica pelo Time de Desenvolvimento

Quando há liberdade de escolha, a equipe de desenvolvimento seleciona as tecnologias com base em critérios como agilidade, performance, qualidade do código e familiaridade, aspecto citado em 12 menções. Em alguns casos, a equipe propõe a tecnologia ao cliente como solução estratégica. Um dos relatos exemplifica bem essa prática: *"Eu chego para o cliente e falo: 'Olha, com o OutSystems*[12] *a gente reduz um ano de trabalho para cinco meses.'"*.

5.4.1.4 Exceções: Quando o Cliente Influencia a Escolha

Casos em que o cliente influencia diretamente a escolha da tecnologia foram citados em 8 menções e são minoria, ocorrendo geralmente quando o cliente possui perfil técnico ou exige algo muito específico. Isso pode envolver desempenho, orçamento ou requisitos técnicos inusitados. Um dos participantes relatou: *"O projeto veio de uma pessoa mais técnica, então já foi requisitado um stack específico."* Outro comentou: *"Tivemos que migrar para React Native*[13] *por causa de reclamações sobre performance."* E ainda: *"Escolhemos Neo4j*[14] *porque os dados do cliente eram muito correlacionados e estavam travando no banco relacional."*

Resposta da Q7: A análise revela que a escolha de tecnologias em projetos Web geralmente não parte de uma exigência direta do cliente. Na maioria dos casos, o cliente não possui familiaridade técnica ou não interfere na decisão, confiando na equipe. Em contextos empresariais, a tecnologia costuma ser previamente definida por diretrizes organizacionais. Quando há autonomia, a decisão técnica é feita com base em critérios como eficiência, familiaridade e viabilidade. Casos em que o cliente influencia a escolha são exceções e estão ligados a perfis mais técnicos ou a exigências muito específicas ligadas a performance, orçamento ou estrutura dos dados.

[12] OutSystems é uma plataforma low-code para desenvolvimento rápido de aplicações empresariais, permitindo integração com diversos sistemas e automação de processos. Disponível em: <https://www.outsystems.com>

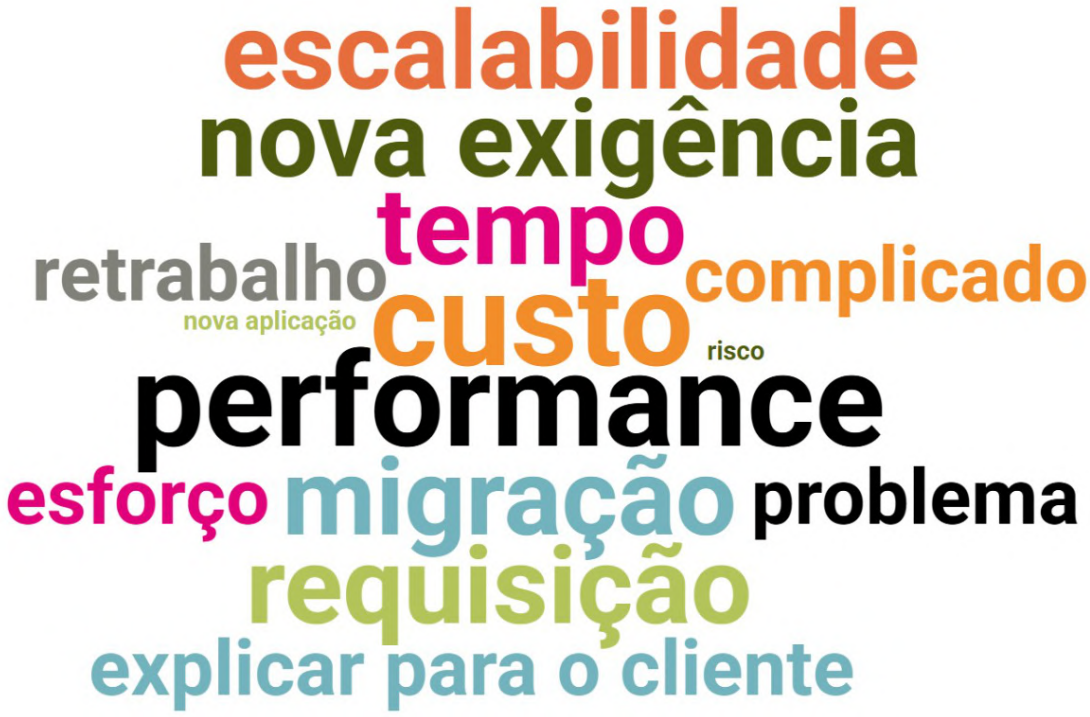
[13] React Native é um framework de código aberto criado pelo Facebook que permite o desenvolvimento de aplicativos móveis nativos utilizando JavaScript e React. Disponível em: <https://reactnative.dev>

[14] Neo4j é um banco de dados orientado a grafos, projetado para armazenar e consultar relacionamentos complexos entre dados. Disponível em: <https://neo4j.com>

5.4.2 *Q8: Já precisou mudar uma tecnologia no meio do projeto por causa de uma nova exigência do cliente?*

A Figura 26 apresenta a nuvem de palavras gerada a partir das respostas dos entrevistados da pergunta 08. Os termos mais recorrentes, como *custo*, *tempo*, *migração*, *problema*, *escopo*, *performance* e *cliente*, refletem a natureza crítica e delicada da mudança de tecnologia durante o ciclo de vida de um projeto Web.

Figura 26 – Nuvem de palavras da análise da questão de entrevista 08



Fonte: Produzido pela autora.

Com base nas respostas, observa-se que a troca de tecnologia no decorrer de um projeto é uma medida extrema, geralmente evitada, mas que pode tornar-se necessária em situações de grande impacto nos requisitos técnicos ou de negócio.

5.4.2.1 Por que evitar a Troca de Tecnologia

A maioria dos entrevistados expressa forte resistência à substituição de tecnologia durante o projeto, justificando essa decisão com argumentos ligados ao custo, tempo, risco e complexidade técnica, possuindo um total de 23 menções. A mudança é vista como algo oneroso e, muitas vezes, contraproducente. Um dos entrevistados afirma que *"migrar é muito custoso em tempo, às vezes mais em dinheiro. O tempo que você levaria para migrar, você*

poderia desenvolver na tecnologia que está usando agora". Outro complementa que *"é muito complicado. Você acaba perdendo tempo, criando problemas e, no final, pode nem entregar o que o cliente quer"*. Há também relatos de desenvolvedores que preferem contornar limitações dentro da tecnologia existente, buscando outras soluções para evitar a troca.

5.4.2.2 Mudança de Escopo e Requisitos Inviáveis na Stack Atual

Apresentando 7 menções, foram relatados casos em que mudanças drásticas no escopo tornaram a tecnologia inicial inviável, exigindo a adoção de uma nova stack para atender aos novos objetivos ou volume do sistema. Um dos participantes relatou que *"o projeto inicialmente era para ser um sistema simples, mas virou algo totalmente diferente. Tivemos que refazer com Angular*[15]*, Python e Spring Boot*[16]*"*. Outro desenvolvedor acrescentou que a mudança foi motivada porque *"a mudança de escopo foi tão grande que a tecnologia original não dava mais conta. Era mais seguro refazer tudo"*.

5.4.2.3 Problemas de Performance e Escalabilidade

Alguns desenvolvedores relataram que, diante de exigências relacionadas à performance e escalabilidade, a troca foi inevitável, possuindo um total de 6 menções. Um participante relatou: *"A consulta estava muito lenta e o gestor exigiu uma solução. A gente migrou para outra tecnologia"*. Outro apontou que *"fizemos a troca por conta de performance. A aplicação começou a crescer e o que estávamos usando não acompanhava"*.

5.4.2.4 Inadequação da Arquitetura de Dados

A estrutura de dados também apareceu como um fator crítico, apesar de possuir apenas 4 menções. Um dos entrevistados explicou que *"precisamos migrar de banco não relacional para relacional por causa de uma nova funcionalidade que exigia estrutura mais rígida"*. Outro desenvolvedor relatou um caso semelhante: *"O cliente pediu uma modelagem*

[15] Angular é um framework front-end para desenvolvimento de aplicações Web, mantido pelo Google, que utiliza *TypeScript* (TS) e promove a construção de aplicações baseadas em componentes. Disponível em: <https://angular.io>

[16] Spring Boot é um framework Java que simplifica o desenvolvimento de aplicações Web e microsserviços, facilitando a configuração e execução rápida de projetos. Disponível em: <https://spring.io/projects/spring-boot>

que o MongoDB[17] *não conseguia atender direito. Migramos para PostgreSQL*[18] *".*

Resposta da Q8: A análise evidencia que a substituição de tecnologia em projetos Web é vista como uma ação extrema, tomada somente quando não há alternativas viáveis. A maioria dos profissionais busca preservar a stack definida inicialmente, seja por questões de custo e tempo, seja para evitar retrabalho e riscos. Quando a troca ocorre, está geralmente associada a mudanças drásticas de escopo, problemas de performance que afetam a experiência do usuário ou incompatibilidades técnicas ligadas à modelagem de dados. A decisão é cuidadosamente ponderada e, na maior parte dos relatos, só é tomada quando todos os outros caminhos já se mostraram inviáveis.

5.4.3 Q9: Quando o cliente muda de ideia no meio do projeto e pede coisas novas, como você lida com isso? Como você atualiza os requisitos? Há alguma peculiaridade em comum que você percebeu quando a mudança solicitada é em um sistema Web?

A Figura 27 apresenta a nuvem de palavras gerada a partir das respostas da pergunta 09. Os termos mais recorrentes — *comunicação*, *visual*, *custo*, *prazo*, *requisitos* e *atualizar documento* — evidenciam que esse tipo de situação é um dos maiores desafios no desenvolvimento de sistemas Web, exigindo do profissional não apenas conhecimento técnico, mas também habilidades de negociação, planejamento e clareza na comunicação.

A grande maioria dos entrevistados relatou que mudanças no meio do projeto são comuns e fazem parte da rotina, mas também destacaram a necessidade de estabelecer processos claros para lidar com elas. As respostas foram agrupadas em três estratégias principais que, quando combinadas, formam uma abordagem estruturada para o controle de mudanças:

5.4.3.1 Análise e Comunicação com o Cliente

Sendo mencionado 33 vezes, a primeira atitude diante de uma nova demanda é entender a solicitação do cliente e analisar os impactos que ela pode gerar no escopo, no cronograma e nos custos. Muitos desenvolvedores relataram que essa fase é essencial para evitar retrabalho e frustrações futuras. Um dos participantes explicou que *"não é sair fazendo"*, pois primeiro é preciso *"analisar o impacto no escopo e no cronograma"*. Outro destacou

[17] MongoDB é um banco de dados NoSQL orientado a documentos. Disponível em: <https://www.mongodb.com>

[18] PostgreSQL é um sistema de gerenciamento de banco de dados relacional objeto aberto. Disponível em: <https://www.postgresql.org>

Figura 27 – Nuvem de palavras da análise da questão de entrevista 09

Fonte: Produzido pela autora.

a dificuldade de explicar tecnicamente certas limitações de forma compreensível ao cliente, ressaltando a importância da habilidade de comunicação.

5.4.3.2 Negociação de Custos e Prazos

Após a análise, muitos entrevistados afirmam que a mudança precisa ser negociada com o cliente em termos de prazos e custos, tendo um total de 22 menções. Em alguns casos, mudanças ignoradas ou não formalizadas causaram sérios atrasos e prejuízos. Um dos participantes contou que o projeto tinha previsão de cinco meses, mas durou quase um ano e meio devido à falta de controle sobre novas demandas, o que fez com que o pagamento se tornasse apenas simbólico. Outro comentou que, ao identificar que se trata de uma nova funcionalidade, costuma *"cobrar por fora"*, indicando que a mudança de escopo precisa ser tratada com seriedade e profissionalismo.

5.4.3.3 Formalização e Atualização dos Requisitos

A última etapa de uma mudança bem conduzida é a sua formalização, possuindo um total de 12 menções. Isso pode ocorrer por meio de documentos de requisitos atualizados ou ferramentas de gestão ágil como *sprints* e *backlogs*. Um dos entrevistados explicou que, quando

a alteração cabe dentro da Sprint atual, ele substitui o *card* pela nova tarefa; caso contrário, a nova demanda vai para o *backlog* e é discutida em ciclos seguintes. Esse tipo de controle é importante para manter o alinhamento entre a equipe e o cliente ao longo do projeto.

5.4.3.4 Peculiaridade nas Mudanças de Sistemas Web: Foco Visual

Um padrão observado nas entrevistas foi a concentração dos pedidos de mudança em aspectos visuais do sistema. Isso evidencia o quanto o cliente tende a perceber o progresso com base na interface e na usabilidade, e não no funcionamento interno da aplicação. Vários entrevistados mencionaram que as mudanças mais comuns dizem respeito a layout, cores, botões e outros elementos do front-end. Um deles observou que *"nunca lidou com um cliente que pedisse algo que não envolvesse o que ele está vendo"*, enquanto outro reforçou que *"o que eles pedem é isso, mudança de visual"*. Isso sugere que sistemas Web, por sua própria natureza voltada à interação com o usuário, estão mais sujeitos a modificações ligadas à aparência e experiência visual.

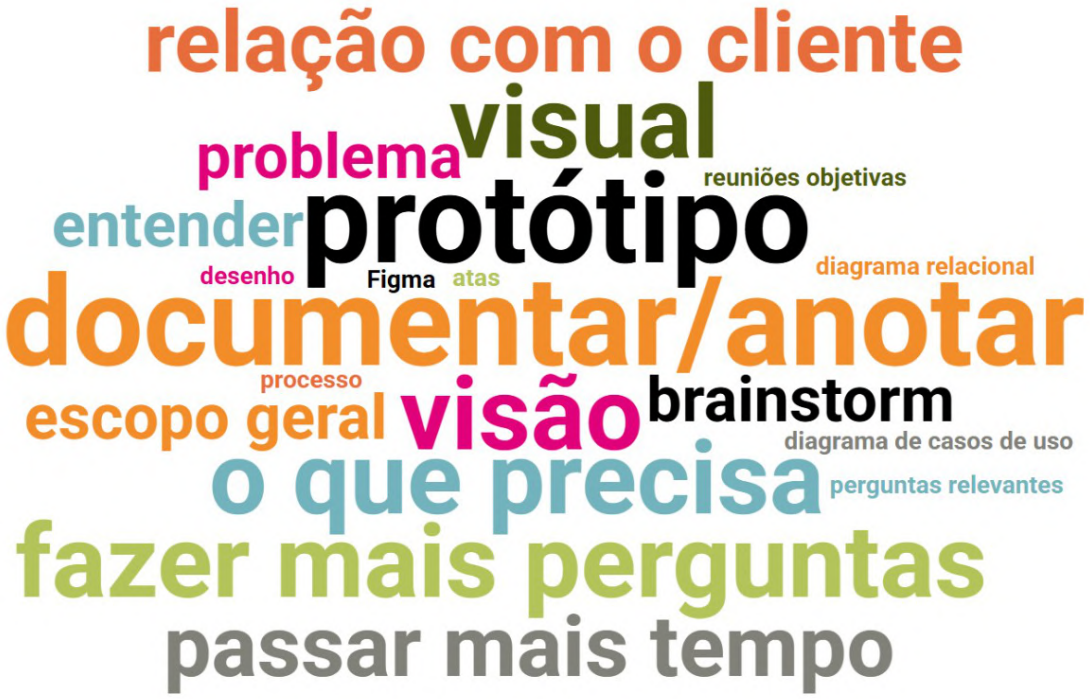
Resposta da Q9: A análise evidencia que mudanças de requisitos durante o desenvolvimento são quase inevitáveis em projetos Web. A maioria dos desenvolvedores adota uma abordagem estruturada baseada em análise, negociação e formalização. Essa postura permite manter o controle do escopo e evitar prejuízos financeiros ou atrasos significativos. Além disso, foi identificado um padrão forte relacionado ao tipo de mudança: nos projetos Web, elas tendem a se concentrar no que é visível ao cliente, como interfaces e elementos visuais, o que reforça a importância de validações frequentes e protótipos bem definidos ao longo do processo.

5.5 Bloco 4 – Experiência do Desenvolvedor

5.5.1 Q10: Com o tempo, teve algo que você passou a fazer diferente no levantamento de requisitos que hoje funciona melhor para sistemas web?

A Figura 28 apresenta a nuvem de palavras construída a partir das respostas da pergunta 10. Termos como *cliente*, *visual*, *entender*, *documentar*, *perguntas*, *problema*, *comunicação* e *protótipo* evidenciam um padrão claro de amadurecimento profissional. Com a experiência, os desenvolvedores deixam de apenas reagir às demandas e adotam posturas mais estratégicas, comunicativas e preventivas.

Figura 28 – Nuvem de palavras da análise da questão de entrevista 10



Fonte: Produzido pela autora.

As mudanças relatadas pelos desenvolvedores ao longo de sua trajetória revelam quatro grandes áreas de aprimoramento no processo de levantamento de requisitos: postura estratégica, organização documental, foco visual e comunicação ativa.

5.5.1.1 Mudança de Mindset: De "O Que Fazer" para "Por Que Fazer"

Possuindo 21 menções, esta transformação de mentalidade é apontada como o principal marco de evolução profissional. No início, os desenvolvedores relatam que seguiam as solicitações dos clientes literalmente, sem questionamentos. Com o tempo, passaram a buscar o entendimento da visão geral do sistema e do problema de negócio. Um entrevistado relata que deixou de perguntar apenas *"o que o cliente quer"* para começar a explorar *"qual a visão que ele tem do sistema"*, promovendo sessões de *brainstorm* antes de converter as ideias em requisitos concretos. Outro destacou que *"a relação com o cliente é mais difícil do que aprender uma tecnologia"*, indicando o quanto o aspecto humano se tornou central no processo.

5.5.1.2 Adoção de Processos e Documentação Estruturada

Os entrevistados apontam que confiar apenas na memória ou em anotações informais já provocou muitos problemas no passado, tendo um total de 19 menções. Atualmente, priorizam

reuniões mais objetivas e documentação clara, mesmo que simples. Um participante afirmou que *"passa mais tempo no levantamento do que programando"* e que, quanto mais cedo o código começa, maiores são as chances de retrabalho. Outros mencionaram o uso de *cards* com descrições detalhadas e atas de reuniões. O amadurecimento profissional os levou a ver o planejamento como parte essencial da entrega de qualidade.

5.5.1.3 Foco Intenso no Visual: Prototipagem e Diagramação

Uma das práticas mais eficazes adotadas com o tempo é o uso de elementos visuais para comunicar e validar requisitos com o cliente, possuindo um total de 15 menções. Participantes relataram que o uso de protótipos (em ferramentas como Figma) e diagramas de fluxo ou de casos de uso passou a fazer parte do processo. Um desenvolvedor explica que, quando começou a trabalhar com protótipos antes de codar, passou a ter menos retrabalho. Outro destaca que sempre desenha fluxos do usuário logo no início, especialmente em projetos que começam do zero, pois isso facilita tanto a validação quanto o planejamento técnico.

5.5.1.4 Aprimoramento da Comunicação e Investigação

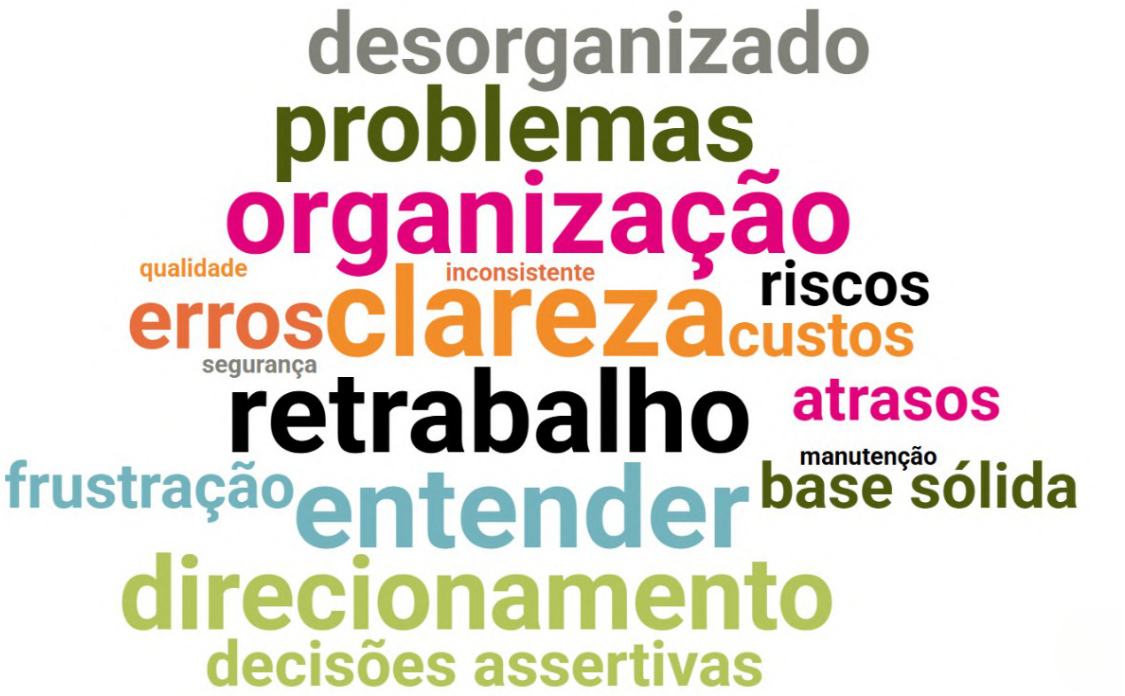
Os desenvolvedores também passaram a investir mais na comunicação com o cliente, tendo um total de 13 menções. Isso inclui fazer perguntas mais relevantes, antecipar possíveis problemas e sugerir melhorias. Um entrevistado compartilhou que, com a experiência, passou a prever reclamações com base em dados e comunicar aos responsáveis antes que se tornassem falhas reais. Outro relatou que aprendeu a fazer perguntas como *"Qual o público do sistema?"* ou *"Em qual região ele vai rodar?"*, evitando decisões erradas que surgem da falta de contexto.

Resposta da Q10: A experiência acumulada no desenvolvimento de sistemas Web leva a uma mudança significativa na forma de lidar com os requisitos. Os profissionais amadurecem de executores reativos para agentes estratégicos, que entendem o contexto do cliente, documentam de forma precisa, utilizam recursos visuais para validar decisões e investem na comunicação como ferramenta de prevenção. Essa evolução resulta em projetos mais alinhados, com menos retrabalho e maior valor entregue ao cliente.

5.5.2 *Q11: Você acha que levantar os requisitos de forma organizada no início ajuda a ter um sistema Web melhor no final? Por quê? É diferente para outros tipos de sistemas?*

A Figura 29 apresenta a nuvem de palavras gerada a partir das respostas dos entrevistados. Os termos mais recorrentes — como *problemas*, *clareza*, *organização*, *direcionamento*, *retrabalho* e *decisões* — evidenciam o reconhecimento da importância do levantamento de requisitos no início do desenvolvimento. A imagem reforça o consenso unânime entre os participantes sobre os benefícios dessa prática para sistemas Web.

Figura 29 – Nuvem de palavras da análise da questão de entrevista 11



Fonte: Produzido pela autora.

Com base nas respostas, a conclusão é categórica: todos os participantes que responderam à pergunta consideram que levantar requisitos de forma organizada no início impacta positivamente a qualidade final de um sistema Web.

- **Sim (acreditam que ajuda):** 18 entrevistados
- **Não respondeu diretamente (comentou apenas sobre design):** 1 entrevistado
- **Não (discorda da importância do levantamento):** 0 entrevistados

5.5.2.1 Prevenção de Problemas e Redução de Riscos

A justificativa mais citada, possuindo um total de 30 menções, refere-se à prevenção de problemas comuns, como retrabalho, erros de escopo e atrasos no cronograma. Os desenvolvedores relatam que um planejamento bem feito economiza tempo e dinheiro, reduz falhas e garante mais segurança no processo. Um entrevistado resume bem esse pensamento: *"Quando a gente não levanta os requisitos da forma correta no início, acaba modificando muita coisa depois que não foi bem estabelecida. Dependendo do caso, isso pode até mudar a arquitetura e impactar grandemente o projeto."* Outro comenta: *"Começar a programar rápido, às vezes até impressiona o cliente [...] Só que eu acho que isso é muito errado, porque é uma parte que [...] precisou ser completamente refeita depois."*

5.5.2.2 Direcionamento, Qualidade e Tomada de Decisão

Além de evitar falhas, o levantamento de requisitos é visto como um guia essencial para as decisões técnicas, a escolha das tecnologias mais adequadas e a garantia de um produto mais consistente e escalável, conforme indicado por 25 menções. Como destaca um participante: *"Se você programa com um objetivo, tudo que você faz fica muito melhor."* Outro reforça o papel estratégico dessa etapa: *"A falta de experiência [...] fazia com que eu não soubesse qual era a melhor tecnologia para certos cenários. [...] passei a fazer muito mais perguntas depois de errar nesses projetos."*

5.5.2.3 Comparação com Outros Tipos de Sistemas

Visão 1: É um Princípio Universal:A maioria dos entrevistados afirmou que o levantamento de requisitos é essencial para qualquer tipo de sistema, independentemente da plataforma. Essa prática é vista como um princípio básico do bom desenvolvimento. Um dos participantes explica: *"Acho que o levantamento de requisitos é mais abrangente, né? Vai depender de muitas coisas, é algo obrigatório para qualquer tipo de aplicação."*

Visão 2: O Sistema Web Tem Peculiaridades que Tornam o Processo Ainda Mais Crítico: Outros entrevistados destacam que, embora a prática seja importante em qualquer cenário, os sistemas Web apresentam desafios específicos que tornam o levantamento ainda mais crucial. Entre os fatores citados estão: a diversidade de frameworks e bibliotecas, a exigência por responsividade e a dinâmica das atualizações constantes. Um entrevistado comenta: *"Você tem*

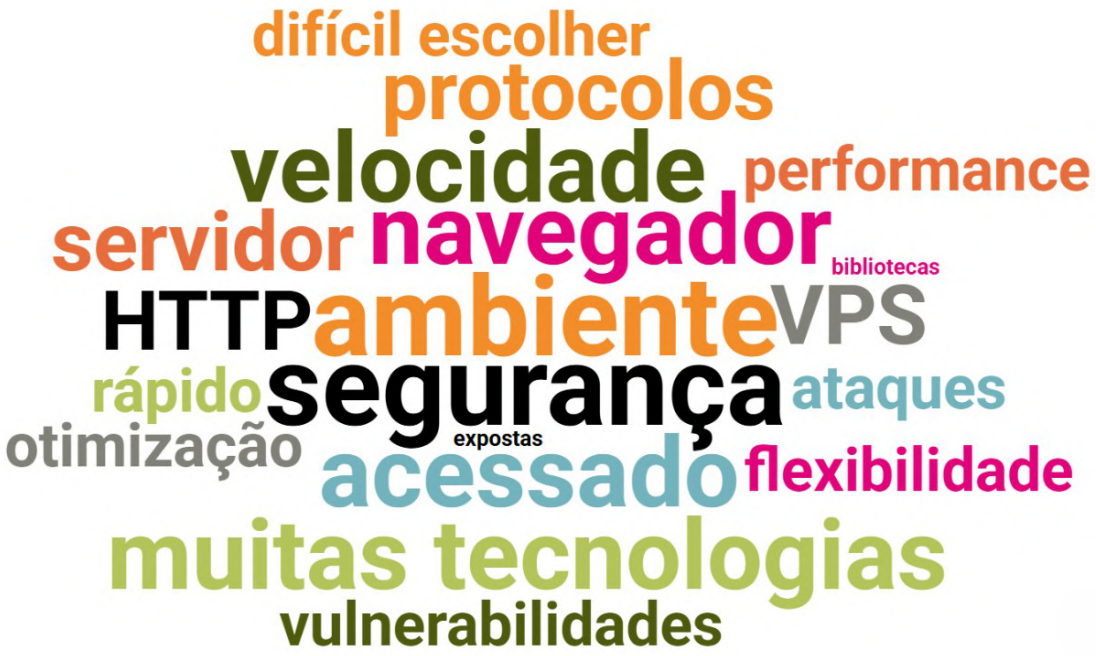
que lidar ali com o tráfego de dados. [...] Tem ainda a questão do servidor, da VPS (Servidor Privado Virtual), que é onde você hospeda." Outro observa: *"Acontece muita atualização, então tem que ser bem otimizado."* E ainda: *"A interface é o produto. Se você não valida o visual, acaba tendo que refazer tudo."*

Resposta da Q11: A análise mostra um consenso absoluto entre os desenvolvedores: levantar os requisitos de forma organizada no início de um projeto Web é essencial para evitar retrabalho, direcionar decisões e garantir a qualidade final do sistema. Além disso, embora esse cuidado seja importante em qualquer tipo de aplicação, os sistemas web apresentam particularidades — como foco visual, dinamismo e diversidade tecnológica — que tornam esse processo ainda mais estratégico.

5.5.3 *Q12: Finalizando, na sua percepção, levantar requisitos para sistemas web é diferente de levantar para outros tipos de sistemas, como sistemas desktop ou mobile? Por quê?*

A Figura 30 apresenta a nuvem de palavras gerada a partir das respostas dos entrevistados. Termos como *ambiente*, *servidor*, *segurança*, *protocolos*, *navegador*, *performance* e *tecnologias* aparecem com destaque, refletindo os principais aspectos que diferenciam o levantamento de requisitos para sistemas Web em relação a outras plataformas.

Figura 30 – Nuvem de palavras da análise da questão de entrevista 12



Fonte: Produzido pela autora.

Com base nas respostas, a percepção predominante é de que o levantamento de requisitos para sistemas Web apresenta particularidades significativas em relação a outras plataformas, como desktop ou mobile. Embora alguns princípios sejam universais, as especificidades do ambiente Web impõem desafios e exigências próprias que afetam diretamente a forma como os requisitos são levantados e discutidos.

- **Sim (acreditam que é diferente):** 14 entrevistados
- **Não (acreditam que o processo é o mesmo):** 3 entrevistados
- **Não souberam responder (falta de experiência):** 2 entrevistados

5.5.3.1 Ambiente, Infraestrutura e Acessibilidade

O aspecto mais destacado pelos participantes, com um total de 24 menções, refere-se às particularidades do ambiente Web, que é acessado por meio de diferentes dispositivos, navegadores e redes. Ao contrário de um sistema desktop, que roda em um ambiente controlado, o sistema Web depende de servidores, protocolos de comunicação e infraestrutura distribuída, exigindo atenção especial desde o início do levantamento. Um dos entrevistados observa: *"O Web não roda na minha máquina, ele vai rodar na VPS. Agora o executável, eu crio ali e vai estar usando seu computador."* Outro complementa: *"Um sistema Web vai depender da funcionalidade de outros, como navegadores, por exemplo. Já um sistema local, desktop ou mobile, vai depender muito do hardware."*

5.5.3.2 Segurança como Ponto Crítico

A visibilidade e exposição natural dos sistemas Web fazem com que a segurança seja tratada como um requisito central desde o levantamento, conforme apontado em 10 menções. A preocupação com ataques, rotas expostas e vulnerabilidades é muito mais presente do que em sistemas locais. Como relatado por um dos desenvolvedores: *"No Web, você tem que pensar além da aplicação, tem que pensar no ambiente, na disponibilidade, em ataques cibernéticos."*

5.5.3.3 Performance e Otimização

A performance em sistemas Web está fortemente ligada à experiência do usuário e à disponibilidade da aplicação em redes diversas, como indicado em 8 menções. Isso implica uma série de exigências específicas, como otimização de scripts, imagens e fluxos de dados. Um

entrevistado explica: *"Você tem que fazer com que rode rápido para que não fique carregando muito, funcione em diferentes telas e com um fluxo constante de dados. Você tem que se preocupar com a otimização de memória, otimização de espaço mesmo."*

5.5.3.4 Vasta Gama de Tecnologias

O Desenvolvimento Web é vasto, dinâmico e constantemente atualizado, o que torna a escolha de tecnologias um processo estratégico que começa ainda na fase de requisitos, conforme indicado em 7 menções. Essa diversidade impacta a clareza do escopo técnico e a viabilidade do projeto. Como afirmou um participante: *"Eu acho que é difícil escolher tecnologia para Web em relação a desktop e sistemas embarcados justamente por isso: é muita coisa, muita coisa para escolher."*.

5.5.3.5 A Visão da Minoria

Apesar da maioria perceber diferenças claras, alguns desenvolvedores defenderam que os fundamentos do levantamento de requisitos permanecem os mesmos, independentemente da plataforma. Segundo essa visão, o mais importante é entender o problema do cliente, sendo a escolha da tecnologia uma consequência posterior. Um dos entrevistados sintetiza esse ponto de vista: *"A plataforma é decidida com base nos requisitos, e não o contrário."*

Resposta da Q12: A maioria dos desenvolvedores identifica que levantar requisitos para sistemas Web envolve especificidades que exigem atenção redobrada desde o início do projeto. Aspectos como segurança, performance, infraestrutura e diversidade tecnológica tornam o processo mais complexo e influenciam diretamente as decisões arquiteturais. Apesar disso, alguns princípios, como compreender a necessidade do cliente, ainda são considerados universais. Para a maioria, no entanto, as demandas da web tornam o levantamento de requisitos não apenas diferente, mas essencialmente mais desafiador.

5.5.4 Comentários Finais dos Entrevistados

Ao final do roteiro de entrevistas, foi disponibilizado um espaço para que os participantes deixassem comentários livres, caso desejassem compartilhar observações adicionais. Embora a maioria tenha optado por não se pronunciar ou apenas tenha registrado agradecimentos,

cinco entrevistados utilizaram esse espaço para fazer reflexões mais amplas, críticas construtivas e desabafos sobre a realidade do desenvolvimento Web.

A análise desses comentários revelou quatro temas principais, que contribuem de forma relevante para compreender o contexto mais amplo da prática profissional e os desafios enfrentados no dia a dia da área.

5.5.4.1 A Necessidade de Melhor Orientação e Educação

Uma das críticas mais recorrentes foi sobre a ausência de materiais práticos e acessíveis que orientem desenvolvedores — especialmente os iniciantes — na escolha de tecnologias e boas práticas no desenvolvimento Web. Os participantes demonstraram carência de conteúdos baseados em experiência real, que auxiliem na tomada de decisões técnicas de forma clara e direta. Um dos entrevistados resumiu: *"Sinto falta de materiais que tragam essa experiência — anos de experiência de alguém que já passou por tudo, que sabe o que fazer e o que não fazer, e que fala claramente: 'não faça isso, faça aquilo'."* A presença desse tipo de conteúdo poderia, segundo os relatos, reduzir o estresse e melhorar a qualidade das entregas.

5.5.4.2 Crítica à Cultura da Área e à Saúde Mental

Outro tema marcante foi a crítica à cultura da área de desenvolvimento, considerada por alguns como hostil, competitiva e desumanizada. Comentários mencionaram o tratamento inadequado para com profissionais iniciantes, como estagiários e juniores, além da sobrecarga psicológica recorrente. Um participante afirmou: *"Qualquer programador que trabalhe em qualquer empresa [...] vai sempre estar lidando com algum problema mental."* Outro completou: *"É raro alguém ficar 30, 40 anos na área de desenvolvimento [...] tem alguma coisa errada aí, não é normal."* As falas indicam uma preocupação crescente com a saúde mental no setor, sugerindo a necessidade de mudanças na cultura organizacional.

5.5.4.3 A Importância do Profissionalismo e da Organização

Também surgiram comentários sobre a necessidade de maior organização e profissionalismo na prática do desenvolvimento. Alguns entrevistados criticaram a utilização de ferramentas avançadas, como o *ChatGPT*, sem o devido cuidado com a clareza e manutenção do código gerado. Um dos relatos destacou: *"Um projeto desorganizado é um custo extra*

para o cliente, porque o desenvolvedor terá que gastar mais tempo refatorando o que está mal feito, levando mais tempo do que o planejado." O foco, aqui, está na responsabilidade do desenvolvedor em manter práticas sustentáveis e bem planejadas.

5.5.4.4 Valorização da Engenharia de Requisitos

Por fim, um entrevistado utilizou o espaço para destacar a importância da Engenharia de Requisitos, reconhecendo que essa etapa é frequentemente negligenciada no mercado. Ele ressaltou a necessidade de maior aproximação entre academia e prática profissional, especialmente em projetos Web, cuja complexidade exige mais atenção à fase de planejamento. O participante comentou: *"Acho excelente esse tipo de pesquisa porque a parte de requisitos é negligenciada. É uma etapa essencial e que precisa de mais atenção no desenvolvimento web, que é bastante complexo."*

Resumo dos Comentários Finais: Apesar de terem sido deixados por uma minoria, os comentários livres revelaram pontos sensíveis e profundos da realidade dos desenvolvedores. As falas reforçam a importância de práticas bem estruturadas, formação crítica, acolhimento profissional e valorização da fase de requisitos. Esses testemunhos espontâneos acrescentam uma dimensão humana e reflexiva à pesquisa, sugerindo caminhos para melhoria não apenas técnica, mas também cultural e institucional na área de desenvolvimento Web.

5.6 Sumarização das Repostas às Questões de Pesquisa

A análise das 19 entrevistas realizadas com desenvolvedores Web revelou um panorama amplo e detalhado sobre os desafios e práticas envolvidos no levantamento de requisitos nesse contexto. As informações foram organizadas com base nas quatro questões de pesquisa que nortearam o estudo, cada uma relacionada a um eixo temático do roteiro de entrevistas: a visão geral do processo de levantamento, as particularidades do desenvolvimento Web, a influência dos requisitos na escolha de tecnologias e a trajetória profissional dos participantes. Para organizar esses achados, a Tabela 2 reúne as palavras-chave mais citadas e os temas associados a cada QP, evidenciando os pontos de maior relevância nas entrevistas e servindo de base para a análise apresentada a seguir.

Tabela 2 – Associação entre Questões de Pesquisa (QPs) e palavras-chave identificadas a partir das entrevistas

Questão de Pesquisa	**Palavras-Chave / Temas Associados**
QP1: Como é realizado o levantamento de requisitos em projetos de aplicações Web, e quais são os tipos de requisitos mais comuns nesse contexto?	Aspectos visuais e interface (UI/UX), prototipação, definição de escopo, requisitos funcionais básicos (login, dashboards, relatórios), responsividade, prazos, velocidade de entrega, custo, orçamento.
QP2: Quais são os principais desafios enfrentados por desenvolvedores e analistas durante o levantamento e a documentação de requisitos em aplicações Web?	Comunicação simplificada, validação contínua, registro formal, mudanças frequentes de requisitos, retrabalho, desempenho, infraestrutura, segurança, dificuldades de alinhamento com stakeholders.
QP3: De que forma os requisitos influenciam na escolha de tecnologias como bibliotecas, frameworks e ferramentas no desenvolvimento de aplicações Web?	Escolha tecnológica orientada ao projeto, bibliotecas, frameworks, integração com APIs, performance, escalabilidade, custo de migração, qualidade do código.
QP4: Quais boas práticas têm sido adotadas pelos profissionais para melhorar o levantamento de requisitos em aplicações Web?	Documentação formal, prototipação, ferramentas de apoio (Trello, Jira, backlog), comunicação clara, planejamento, organização, aprendizado com a experiência.

Fonte: Elaborado pela autora.

5.6.1 QP1: Como é realizado o levantamento de requisitos em projetos de aplicações Web, e quais são os tipos de requisitos mais comuns nesse contexto?

O levantamento de requisitos é reconhecido como uma etapa crítica, mas ainda marcada por informalidade em muitos contextos. A maioria dos desenvolvedores relatou que os pedidos mais comuns de clientes envolvem aspectos visuais, funcionalidades básicas e fluxos simples, como login, cadastro e dashboards. Isso reforça a importância de uma abordagem que saiba traduzir pedidos genéricos em especificações claras. Embora todos reconheçam os benefícios de um levantamento bem estruturado, poucos utilizam processos formalizados desde o início. A comunicação direta com o cliente e o uso de ferramentas visuais (como protótipos e diagramas) se destacam como formas eficazes de evitar mal-entendidos.

5.6.2 QP2: Quais são os principais desafios enfrentados por desenvolvedores e analistas durante o levantamento e a documentação de requisitos em aplicações Web?

As entrevistas revelaram que sistemas Web impõem desafios próprios, que afetam diretamente a forma de levantar e interpretar requisitos. Dentre esses desafios, destacam-se a necessidade de responsividade, preocupação com desempenho, complexidade da infraestrutura e foco acentuado na experiência do usuário. A frequência de mudanças durante o desenvolvimento também foi apontada como uma característica marcante desse tipo de sistema. Os entrevistados relataram que alterações de escopo são quase sempre visuais e, por isso, exigem estratégias de negociação, registro e versionamento que acompanhem o ritmo das expectativas do cliente.

5.6.3 QP3: De que forma os requisitos influenciam na escolha de tecnologias como bibliotecas, frameworks e ferramentas no desenvolvimento de aplicações Web?

Outro ponto de destaque foi o papel dos requisitos como direcionadores na seleção de frameworks, bibliotecas e stacks. Os desenvolvedores demonstraram maturidade ao apontar que a escolha da tecnologia deve ser subordinada às demandas do projeto, e não o contrário. Casos em que foi necessário mudar de tecnologia no meio do projeto foram raros e tratados como exceções estratégicas, geralmente motivadas por mudanças drásticas de escopo, problemas de performance ou limitações técnicas severas. A troca de tecnologia, quando inevitável, é sempre vista como onerosa e arriscada.

5.6.4 QP4: Quais boas práticas têm sido adotadas pelos profissionais para melhorar o levantamento de requisitos em aplicações Web?

Uma mudança de postura foi percebida com o amadurecimento dos profissionais. Os relatos apontam uma transição de uma atitude reativa, focada na implementação direta, para uma conduta mais investigativa, comunicativa e estratégica. Com o tempo, os desenvolvedores passaram a valorizar o entendimento profundo do problema, a prototipação, a documentação e o alinhamento com os objetivos do cliente. Além disso, houve consenso absoluto sobre a importância do levantamento de requisitos organizado como fator decisivo para a qualidade final do sistema. Embora alguns profissionais defendam que esse princípio se aplica a qualquer tipo de sistema, outros ressaltaram que o ambiente Web amplifica essa necessidade, dada a sua natureza dinâmica, distribuída e visual.

5.6.5 Reflexões Adicionais dos Entrevistados

Os comentários finais deixados por alguns participantes revelaram críticas relevantes à cultura da área de desenvolvimento, especialmente quanto à falta de apoio para iniciantes, desvalorização da fase de requisitos e impacto negativo na saúde mental. Também se destacou o desejo por materiais mais práticos e experiências compartilhadas que orientem escolhas técnicas com base em vivência real.

5.6.6 Considerações Finais

A análise das entrevistas permitiu identificar padrões relevantes e pontos de reflexão sobre o levantamento de requisitos em projetos Web. A discussão a seguir é organizada de acordo com os blocos temáticos utilizados na pesquisa, sintetizando as mensagens centrais extraídas das falas dos entrevistados e destacando as lições aprendidas.

5.6.6.1 Bloco 1 – Entendimento Geral do Levantamento de Requisitos

Os resultados mostraram que o levantamento de requisitos em aplicações Web é frequentemente conduzido de maneira informal, com grande dependência da comunicação direta entre cliente e desenvolvedor. Apesar de reconhecerem a importância dessa fase, muitos profissionais relataram que os clientes costumam apresentar demandas vagas, o que exige do desenvolvedor um papel ativo na tradução de ideias em especificações claras.

- **Lição aprendida 1:** a prototipação e a validação no início do projeto são ferramentas indispensáveis para alinhar expectativas e reduzir retrabalho.
- **Lição aprendida 2:** alterações visuais são predominantes e reforçam a necessidade de validação constante durante o projeto.

5.6.6.2 Bloco 2 – Requisitos Específicos da Web

Os desafios mais citados relacionam-se à comunicação, documentação e mudanças de escopo. A ausência de registros formais gera conflitos e retrabalho, enquanto a frequência de alterações, especialmente em aspectos visuais, exige que os desenvolvedores adotem estratégias de prevenção, como o uso de ferramentas colaborativas e validações contínuas.

- **Lição aprendida 3:** a documentação clara, mesmo em projetos ágeis, é um diferencial para reduzir conflitos e tornar o processo mais previsível.
- **Lição aprendida 4:** a natureza dinâmica do ambiente Web, com demandas de segurança, responsividade e integração, exige planejamento adaptativo e comunicação constante.

5.6.6.3 Bloco 3 – Influência dos Requisitos na Escolha de Tecnologias

Os entrevistados destacaram que a escolha de bibliotecas, frameworks e ferramentas deve ser guiada pelas necessidades do projeto, e não pelo gosto pessoal do desenvolvedor. Embora casos de troca de tecnologia sejam raros, quando ocorrem, geram impacto significativo

no cronograma e no orçamento.

- **Lição aprendida 5:** decisões tecnológicas mais diretas surgem quando há clareza nos requisitos desde o início, evitando mudanças custosas no decorrer do desenvolvimento.
- **Lição aprendida 6:** ainda há carência de materiais práticos e diretrizes que auxiliem na escolha de tecnologias adequadas ao contexto do projeto.

5.6.6.4 Bloco 4 – Experiência Profissional com Requisitos

Com a experiência, os desenvolvedores evoluem de uma postura reativa para uma abordagem mais analítica e consultiva, atuando não apenas como executores, mas também como facilitadores do entendimento entre o cliente e a equipe técnica. Essa mudança de postura foi apontada como fundamental para o sucesso dos projetos.

- **Lição aprendida 7:** o amadurecimento profissional leva a práticas mais eficazes de levantamento de requisitos, com maior ênfase em análise estratégica, comunicação empática e prevenção de falhas.
- **Lição aprendida 8:** há uma demanda evidente por mentorias, compartilhamento de experiências e materiais que apoiem os profissionais, especialmente iniciantes, na condução dessa fase crítica.

Os resultados reforçam que o levantamento de requisitos é base do desenvolvimento Web e que sua condução eficiente depende de práticas comunicativas, documentação adequada, escolhas tecnológicas conscientes e do aprendizado contínuo dos profissionais envolvidos.

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho investigou como os requisitos impactam o desenvolvimento de sistemas Web, com ênfase nas práticas, desafios e percepções de desenvolvedores a partir de entrevistas qualitativas. A análise dos depoimentos de 19 profissionais da área permitiu mapear padrões e estratégias recorrentes, bem como revelar aspectos subjetivos do processo que não são comumente registrados em manuais técnicos ou metodologias tradicionais.

Os dados apontam que o sucesso de um sistema Web vai muito além do domínio técnico. Os maiores desafios relatados envolvem comunicação com clientes, organização de requisitos, mudanças durante o projeto e a necessidade de traduzir ideias vagas em soluções concretas. A experiência revelou que a maturidade do processo está ligada não apenas ao uso de ferramentas ou frameworks, mas à capacidade do desenvolvedor de atuar como um facilitador estratégico, capaz de compreender o contexto de negócio, negociar escopo e antecipar problemas. Entre os principais achados, destacam-se:

- A predominância de alterações visuais como foco das mudanças solicitadas durante o projeto, o que reforça a importância da prototipação e validação precoce com o cliente.
- A diferença marcante entre o levantamento de requisitos para sistemas Web e para outras plataformas, sobretudo devido a aspectos como segurança, responsividade, infraestrutura e diversidade tecnológica.
- A evolução do desenvolvedor, que com o tempo passa de executor técnico para um profissional mais analítico, consultivo e estratégico.
- A influência decisiva de um bom levantamento de requisitos na redução de retrabalho, clareza de objetivos e entrega de sistemas com maior qualidade.
- A carência de materiais práticos, mentorias e diretrizes claras que auxiliem na escolha de tecnologias, formalização de mudanças e comunicação com o cliente.

6.1 Contribuições do Trabalho

As contribuições deste estudo são múltiplas. Primeiramente, oferece um retrato realista e aprofundado das práticas de levantamento de requisitos sob a ótica de desenvolvedores que atuam na construção de sistemas Web. Em segundo lugar, fornece insumos para o desenvolvimento de novas ferramentas, modelos e frameworks que podem apoiar tanto profissionais quanto estudantes na fase inicial do projeto. Por fim, o trabalho também lança luz sobre temas

menos explorados, como a saúde mental na área de desenvolvimento e o impacto da cultura organizacional na sustentabilidade da carreira.

6.2 Limitações

Por ser um estudo qualitativo com foco em entrevistas, o número de participantes foi limitado a 19 profissionais, o que restringe a generalização estatística dos achados. Além disso, embora os entrevistados tenham perfis variados, há predominância de experiências com sistemas Web voltados ao setor privado e desenvolvimento sob demanda, o que pode não refletir todos os contextos de atuação, como desenvolvimento interno corporativo ou sistemas governamentais.

6.3 Trabalhos Futuros

A partir das evidências coletadas nas entrevistas, foram apontadas diversas direções promissoras para pesquisas futuras e aplicações práticas:

- Desenvolvimento de frameworks de apoio à decisão para escolha de tecnologias com base nos requisitos iniciais.
- Criação de modelos padronizados para formalização de escopo, análise de impacto de mudanças e comunicação com clientes.
- Estudos empíricos comparando diferentes técnicas de levantamento de requisitos em projetos Web, contabilizando métricas como retrabalho, qualidade da entrega e satisfação do cliente.
- Pesquisas longitudinais sobre a sustentabilidade da carreira na área de desenvolvimento, investigando fatores de burnout e abandono precoce da área técnica.
- Iniciativas educacionais voltadas à formação de competências consultivas no levantamento de requisitos, com foco em comunicação, análise crítica e empatia.

Conclui-se que o levantamento de requisitos não é apenas uma etapa do desenvolvimento Web, mas sim a base sobre a qual todo o projeto se sustenta. Ao compreender seu papel estratégico e investir continuamente em sua melhoria, torna-se possível entregar sistemas não apenas tecnicamente corretos, mas também alinhados às reais necessidades dos usuários e sustentáveis ao longo do tempo.

REFERÊNCIAS

AMUNDSEN, M. **RESTful Web API Patterns and Practices Cookbook: Connecting and Orchestrating Microservices and Distributed Data**. 1. ed. [S.l.]: O'Reilly Media, 2023.

ANTONIO. Requisitos não funcionais e arquitetura de software. **Revista Engenharia de Software**, n. 3, 2008. Acesso em 14 set. 2024. Disponível em: <https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525>.

ANZBÖCK, R.; DUSTDAR, S. Modeling and implementing medical web services. **Data & Knowledge Engineering**, Elsevier, v. 55, n. 2, p. 203–236, 2005.

ARAÚJO, J. V. R. **Uma Plataforma Colaborativa para Apoio ao Levantamento de Requisitos de Software**. Dissertação (Dissertação de Mestrado) — Universidade Federal do Ceará (UFC), Quixadá, 2024.

BEDER, D. M. **Engenharia Web: uma abordagem sistemática para o desenvolvimento de aplicações web**. São Carlos: EdUFSCar, 2012. (Coleção UAB-UFSCar). ISBN 978-85-7600-290-1.

BOEHM, A.; RUVALCABA, Z. **Murach's HTML5 and CSS3**. [S.l.]: Mike Murach & Associates, 2015.

BRASIL, S. E. de Comunicação Social Governo Federal do. **Manual de SEO: Otimização para Mecanismos de Busca**. [S.l.], 2020. Disponível em: <https://www.gov.br/governodigital/pt-br/acesso-a-informacao/legislacao-e-normativos/guia-seo.pdf>.

CHAWLA, S.; SRIVASTAVA, S. Improving web requirements engineering with goals, aspects and scenarios. In: **Proceedings of the 2012 Students Conference on Engineering and Systems**. Allahabad, India: IEEE, 2012.

DARGHAM, J.; SEMAAN, R. A navigational web requirements validation through animation. In: **Proceedings of The Third International Conference on Internet and Web Applications and Services (ICIW)**. [S.l.]: IEEE, 2008. p. 56–61. ISBN 978-0-7695-3163-2.

DAVIS, M. M.; CHASE, R. B.; AQUILANO, N. J. **Fundamentos da administração da produção**. [S.l.]: Bookman, 2001.

DOVER, D.; DAFFORN, E. **Search Engine Optimization (SEO) Secrets**. [S.l.]: Wiley, 2011. ISBN 9780470554180.

ECCHER, C. **Professional Web Design: Techniques and Templates**. [S.l.]: Cengage Learning, 2015.

ENGE, E.; SPENCER, S.; STRICCHIOLA, J. **The Art of SEO: Mastering Search Engine Optimization**. 4. ed. [S.l.]: O'Reilly Media, 2022. ISBN 9781098102616.

GARCIA, J. E.; PAIVA, A. C. A requirements-to-implementation mapping tool for requirements traceability. **J. Softw.**, v. 11, n. 2, p. 193–200, 2016.

GUEDES, G. T. A. Um metamodelo uml para a modelagem de requisitos em projetos de sistemas multiagentes. 2012.

GUERRA, P. **Modelagem de Requisitos de Software**. 2009. Acessado em: 13 set. 2024. Disponível em: <https://www.mindmeister.com/pt/28238820/modelagem-de-requisitos-de-software>.

HEATH, F. **Managing Software Requirements the Agile Way: Bridge the gap between software requirements and executable specifications to deliver successful projects**. [S.l.]: Packt Publishing Ltd, 2020.

HUMBLE, J.; FARLEY, D. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. [S.l.]: Addison-Wesley, 2010. ISBN 9780321601919.

HUMBLE, J.; READ, C.; NORTH, D. The deployment production line. In: **Proceedings of AGILE 2006 Conference (AGILE'06)**. [S.l.]: IEEE, 2006. p. 6–14.

IEEE. **ISO/IEC/IEEE International Standard 29148-2018 – Systems and software engineering – Life cycle processes – Requirements engineering**. New York, NY, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8559686>.

JACOBSON, D.; BRAIL, G.; WOODS, D. **APIs: A Strategy Guide**. 1. ed. [S.l.]: O'Reilly Media, 2011.

JIRAPANTHONG, W. Requirements traceability on web applications. In: IEEE. **2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)**. [S.l.], 2015. p. 18–23.

KEPNER, C.; TREGOE, B. O administrador racional: Uma abordagem sistemática para solução de problemas e tomada de decisão. 2 ª edição. **São Paulo**, 1972.

KUMAR, S.; SANGWAN, S. Adapting the software engineering process to web engineering process. **International Journal of Computing and Business Research**, v. 2, n. 1, p. 42–63, 2011.

MARCOTTE, E. **Responsive Web Design**. [S.l.]: A Book Apart, 2011.

PARKER, H. **Ferramentas de gerenciamento de requisitos**. 2024. Acesso em: 13 de set. 2024. Disponível em: <https://clickup.com/pt-BR/blog/41671/ferramentas-de-gerenciamento-de-requisitos>.

PATHAK, J.; BASU, S.; HONAVAR, V. Modeling web services by iterative reformulation of functional and non-functional requirements. In: SPRINGER. **Service-Oriented Computing–ICSOC 2006: 4th International Conference, Chicago, IL, USA, December 4-7, 2006. Proceedings 4**. [S.l.], 2006. p. 314–326.

PINHEIRO, F. A. Requirements traceability. **Perspectives on software requirements**, Springer, p. 91–113, 2004.

PRESSMAN, R. S.; MAXIM. **Web Engineering: A Practitioner's Approach**. [S.l.]: McGraw-Hill, 2009.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software-9**. [S.l.]: McGraw Hill Brasil, 2021.

RAMOS, E. d. S. **Lidando com Stakeholders: boas práticas em Análise de Negócios e Engenharia de Requisitos**. Amazon.com: Kindle Direct Publishing (KDP) by Amazon.com, Inc., 2019. Edição do Kindle.

REGO, A. da S. **Técnicas de Elicitação de Requisitos**. 2023. Accessed on September 13, 2024. Disponível em: <https://www.estrategiaconcursos.com.br/blog/tecnicas-elicitacao-requisitos-ebserh/>.

SILVA, J.; ALMEIDA, M.; PEREIRA, L. Uso de campos semânticos para redução de ambiguidades em requisitos de software. **Revista de Sistemas e Computação**, v. 10, n. 2, p. 45–58, 2020.

SOMMERVILLE, I. **Engineering software products**. [S.l.]: Pearson London, UK, 2020. v. 355.

SUKUMARAN, S.; SREENVIAS, A.; VENKATESH, R. Uma abordagem rigorosa para validação de requisitos. In: **Anais da 4ª Conferência Internacional de Engenharia de Software (SEFM 06)**. Washington, DC, USA: IEEE, 2006.

VAZQUEZ, C. E.; SIMõES, G. S. **Engenharia de Requisitos: software orientado ao negócio**. [S.l.]: BRASPORT, 2020. 92 p. Edição do Kindle.

Visure Solutions. **O Guia Mais Completo para Gerenciamento de Requisitos e Rastreabilidade**. 2024. Acesso em: 13 de set. 2024. Disponível em: <https://visuresolutions.com/pt/requirements-management-traceability-guide/requirements-gathering/>.

ZANIRO, R.; FABBRI, S. Web-semp: Um método para engenharia de requisitos em aplicações web. In: **Anais do Simpósio Brasileiro de Engenharia de Software (SBES)**. [S.l.: s.n.], 2015. p. 100–111.