



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PEDRO II
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOSÉ ROBERTO ALVES VIANA

ARQUITETURA DE MICROFRONTENDS NO DESENVOLVIMENTO
COLABORATIVO: análise da percepção de desenvolvedores em um estudo de caso
experimental

PEDRO II
2026

JOSÉ ROBERTO ALVES VIANA

ARQUITETURA DE MICROFRONTENDS NO DESENVOLVIMENTO
COLABORATIVO: análise da percepção de desenvolvedores em um estudo de caso
experimental

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Pedro II.

Orientador: Prof^o. José Miqueias de Araújo
Pereira

ARQUITETURA DE MICROFRONTENDS NO DESENVOLVIMENTO COLABORATIVO: análise da percepção de desenvolvedores em um estudo de caso experimental

José Roberto Alves Viana¹
José Miqueias de Araújo Pereira²

RESUMO

A crescente complexidade das aplicações web modernas impulsionou a adoção generalizada de arquiteturas de desenvolvimento modulares e colaborativas. Nesse contexto, surgiram os micro front-ends, que permitem a divisão do *front-end* em módulos independentes, aumentando a autonomia das equipes e a agilidade do desenvolvimento. O objetivo deste estudo é investigar a percepção de desenvolvedores sobre o uso da arquitetura de micro front-ends no contexto do desenvolvimento colaborativo. Para tanto, desenvolvemos um sistema experimental de agendamento de horários em salões de cabeleireiro utilizando uma arquitetura de micro front-ends e integrado ao *framework single-spa*. Este estudo se caracteriza por sua aplicabilidade, natureza descritiva, abordagem quantitativa e delineamento experimental. Os dados foram coletados por meio de questionários online respondidos por 12 participantes com diferentes formações e experiências em desenvolvimento de *software*. Os resultados mostram que os participantes expressaram atitudes positivas em relação à arquitetura, particularmente no que diz respeito à escalabilidade, facilidade de navegação, integração de módulos e distribuição de tarefas dentro das equipes. Embora os participantes tenham reconhecido um leve aumento na complexidade do desenvolvimento, os benefícios percebidos superaram os desafios. Em conclusão, os micro front-ends representam uma solução viável para aplicações que requerem desenvolvimento colaborativo, modularidade e evolução contínua do sistema.

Palavras-chave: micro front-ends; desenvolvimento colaborativo; arquitetura de software; aplicações web; modularidade.

ABSTRACT

The increasing complexity of modern web applications has driven the widespread adoption of modular and collaborative development architectures. In this context, micro front-ends have emerged, enabling the front-end to be split into independent modules, thereby enhancing team autonomy and development agility. This study aims to investigate developer perceptions regarding the use of micro front-end architecture within a collaborative development context. To this end, we developed an experimental hair salon appointment scheduling system using a micro front-end architecture integrated with the *single-spa framework*. The study is characterized by its practical applicability, descriptive nature, quantitative approach, and experimental design. Data were collected via online questionnaires completed by 12

¹ Graduando em Análise e Desenvolvimento de Sistemas. Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Pedro II. E-mail: caped.20201p2ads0161@aluno.ifpi.edu.br.

² Graduação em Sistemas de Informação. Orientador. Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Pedro II. E-mail: Jose.miqueias@ifpi.edu.br.

participants with diverse backgrounds and experience in software development. The results indicate that participants held positive attitudes toward the architecture, particularly regarding scalability, ease of navigation, module integration, and task distribution within teams. Although participants acknowledged a slight increase in development complexity, the perceived benefits outweighed the challenges. In conclusion, micro front-ends represent a viable solution for applications requiring collaborative development, modularity, and continuous system evolution.

Keywords: micro front-ends; collaborative development; software architecture; web applications; modularity.

Data de aprovação: 15 de Julho de 2026

1 INTRODUÇÃO

Segundo Nascimento e Sotto (2020), os microfrontends surgem como uma extensão dos princípios da arquitetura de microsserviços aplicados à camada de *front-end*, permitindo a decomposição de aplicações em módulos independentes e de baixo acoplamento. Essa abordagem favorece a autonomia no desenvolvimento e manutenção dos sistemas, possibilitando que diferentes equipes atuem de forma paralela sobre partes distintas da aplicação (Batista; Pinto, 2023).

Além disso, a utilização de módulos independentes contribui para aspectos como escalabilidade, flexibilidade tecnológica e evolução contínua da aplicação, permitindo inclusive a integração de diferentes *frameworks*, como Angular, React e Vue.js (Cheripurathu; KulKarni, 2024). Nesse contexto, Cheripurathu e Kulkarni (2024) destacam que a integração entre microsserviços e microfrontends contribui para a redução do acoplamento e para o aumento da adaptabilidade das aplicações às demandas do mercado.

Apesar dos benefícios, a adoção de microfrontends impõe desafios técnicos, especialmente no que diz respeito à integração entre módulos heterogêneos e à comunicação entre diferentes partes da aplicação. Yudin *et al.* (2024) destacam que, embora a modularização contribua para uma melhor organização do código e distribuição de responsabilidades, problemas de desempenho e complexidade podem surgir quando a integração entre os componentes não é adequadamente gerenciada. Nesse contexto, torna-se necessária a utilização de estratégias arquiteturais robustas para orquestração e comunicação entre os microfrontends.

Em um contexto organizacional, a arquitetura de microfrontend também influencia o desenvolvimento colaborativo e promove uma distribuição clara de responsabilidades entre as equipes. Segundo Batista e Pinto (2023), esse método proporciona maior autonomia para o

desenvolvimento, reduz as dependências entre as equipes e facilita o desenvolvimento contínuo do sistema. Nascimento e Sotto (2020) acrescentam que a segmentação do frontend em módulos independentes promove a especialização da equipe, levando, assim, a um ambiente de desenvolvimento mais dinâmico e eficiente.

Entretanto, em aplicações *front-end* de maior porte, a concentração de funcionalidades em uma única aplicação pode dificultar a manutenção e aumentar o acoplamento entre partes do sistema, comprometendo a escalabilidade e a evolução contínua da aplicação (Yudin *et al.*, 2024). Segundo Nascimento e Sotto (2020), arquiteturas monolíticas tendem a centralizar responsabilidades em um único código-base, o que pode tornar o desenvolvimento colaborativo mais complexo em equipes distribuídas. Nesse contexto, diferentes equipes passam a compartilhar dependências e atuar sobre os mesmos componentes da aplicação, aumentando o risco de conflitos durante o processo de desenvolvimento e integração.

Além disso, a adoção de múltiplas tecnologias em um mesmo projeto pode representar um desafio quando não há uma arquitetura capaz de organizar adequadamente a comunicação e integração entre os módulos (Cheripurathu; Kulkarni, 2024). Diante desse cenário, surge a seguinte problemática: de que forma a arquitetura de microfrontends pode contribuir para o desenvolvimento colaborativo, considerando a autonomia dos desenvolvedores, a divisão de responsabilidades e os desafios de integração entre diferentes *frameworks*?

Apesar desses avanços, observa-se que, embora existam evidências teóricas e estudos aplicados sobre os benefícios dos microfrontends, ainda há necessidade de investigações que analisem, de forma prática, o impacto dessa arquitetura sob a perspectiva dos desenvolvedores, especialmente em contextos que envolvem múltiplas tecnologias integradas em uma mesma aplicação.

Nesse contexto, este trabalho propõe a implementação de uma aplicação experimental no domínio de agendamento de serviços de barbearia, estruturada com base na arquitetura de microfrontends e orquestrada por meio do *framework* single-spa, integrando diferentes *frameworks* de *front-end* (Canopy Taxi, 2025a). A aplicação foi concebida com o objetivo de simular um ambiente real de desenvolvimento colaborativo, no qual diferentes módulos são desenvolvidos de forma independente.

Dessa forma, o presente estudo tem como objetivo geral investigar a percepção de desenvolvedores sobre o uso da arquitetura de microfrontends no contexto do desenvolvimento colaborativo. Como objetivos específicos, destacam-se: (a) desenvolver uma aplicação experimental baseada em microfrontends; (b) analisar a percepção dos desenvolvedores sobre aspectos relacionados à modularidade, integração e autonomia entre módulos; e (c) Identificar

características que diferenciam a arquitetura de microfrontends do modelo monolítico tradicional.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 MICROFRONTENDS: CONCEITO E AVANÇOS TECNOLÓGICOS

A arquitetura de microfrontends surge como uma abordagem para o desenvolvimento de interfaces web baseada na decomposição do *front-end* em módulos independentes. Inspirada nos princípios de microsserviços, essa arquitetura permite que diferentes partes de uma aplicação sejam desenvolvidas, testadas e implantadas de forma autônoma, favorecendo a escalabilidade e a flexibilidade dos sistemas.

De acordo com Schäffer *et al.* (2019), arquiteturas baseadas em microsserviços possibilitam a construção de sistemas modulares e colaborativos, nos quais diferentes componentes podem ser desenvolvidos de forma isolada, mas integrados de maneira coesa. Essa característica é transferida para o contexto dos microfrontends, permitindo maior organização do desenvolvimento e redução de conflitos entre equipes.

No contexto das aplicações web modernas, o uso de microfrontends também contribui para otimizar a eficiência e a manutenibilidade. Para Perera (2023), a adoção de modelos arquiteturais baseados em microsserviços pode aumentar significativamente a eficiência de uma aplicação. Todavia, desde que sejam implementados planos de integração adequados entre os módulos. Assim, a segmentação do frontend permite atualizações mais rápidas e direcionadas sem comprometer a funcionalidade geral do sistema.

Além disso, avanços tecnológicos recentes têm impulsionado o desenvolvimento de ferramentas voltadas à implementação de microfrontends. Celik (2020), por exemplo, propôs uma solução para criação de microfrontends com controle de versões e tipagem, contribuindo para maior segurança e consistência entre os módulos. De forma complementar, Svoboda. (2022) destacam que a integração entre microfrontends e microsserviços amplia o potencial dessa arquitetura, especialmente em ambientes colaborativos e distribuídos.

Dessa forma, os microfrontends consolidam-se como uma evolução na construção de interfaces web, permitindo a adoção de diferentes tecnologias dentro de uma mesma aplicação, ao mesmo tempo em que promovem maior flexibilidade, escalabilidade e eficiência no desenvolvimento.

2.2 MICROFRONTENDS E DESENVOLVIMENTO COLABORATIVO

A implementação de microfrontends visa aprimorar os processos de desenvolvimento colaborativo em empresas de desenvolvimento de software. Ao dividir uma aplicação em módulos independentes, torna-se possível que diferentes equipes trabalhem simultaneamente em diferentes partes do sistema. Reduzindo dependências e aumentando a produtividade.

Segundo Batista e Pinto (2023), práticas colaborativas aliadas a boas estratégias de engenharia de software contribuem para processos mais eficientes e com menor incidência de erros. Nesse contexto, os microfrontends favorecem a organização do trabalho em equipes multidisciplinares, permitindo que cada grupo se concentre em um domínio específico da aplicação.

Além disso, a utilização de múltiplas tecnologias dentro de uma mesma aplicação amplia as possibilidades de desenvolvimento, possibilitando que equipes utilizem *frameworks* mais adequados às suas necessidades. Conforme evidenciado por Cheripurathu e Kulkarni (2024), essa abordagem contribui para a construção de sistemas mais dinâmicos e adaptáveis, especialmente em projetos que demandam integração entre diferentes módulos.

A relevância do desenvolvimento colaborativo também é destacada no contexto da transformação digital. Neste sentido, Barros (2024) enfatiza que a modernização de sistemas requer soluções tecnológicas flexíveis e integradas. Enquanto Lima (2025) demonstra que arquiteturas modulares podem otimizar processos e melhorar a qualidade dos serviços. Nesse cenário, os microfrontends representam uma estratégia eficaz para promover a colaboração, a inovação e a escalabilidade em aplicações modernas.

2.3 DESAFIOS DA IMPLEMENTAÇÃO DE MICROFRONTENDS

Apesar das suas vantagens, a implementação de microfrontends enfrenta muitos obstáculos, especialmente relacionados com a integração de módulos e a complexidade da arquitetura. A harmonização das diversas tecnologias e a comunicação eficiente entre os componentes exigem um planejamento cuidadoso e a definição de normas.

De acordo com Cheripurathu e Kulkarni (2024), a interoperabilidade entre módulos

desenvolvidos com diferentes *frameworks* pode gerar dificuldades técnicas, principalmente quando não há uma estratégia bem definida para orquestração e comunicação. Nesse contexto, a ausência de padronização pode impactar negativamente tanto o desempenho quanto a experiência do usuário.

Além disso, aspectos relacionados à manutenção e governança também representam desafios. Cheripurathu e Kulkarni (2024) destacam que, embora a arquitetura favoreça a distribuição de responsabilidades, a falta de alinhamento entre equipes pode gerar inconsistências no sistema. Batista e Pinto (2023) reforçam que a comunicação entre equipes e a adoção de práticas uniformes são essenciais para garantir a sustentabilidade da arquitetura.

Portanto, embora os microfrontends ofereçam vantagens, sua implementação exige maturidade técnica e organizacional, sendo fundamental equilibrar autonomia das equipes com a necessidade de padronização e integração.

2.4 FERRAMENTAS PARA MICROFRONTENDS

A implementação de arquiteturas baseadas em microfrontends se tornou possível graças ao surgimento de diversas operações e *frameworks* que auxiliam na orquestração, integração e gerenciamento de múltiplos módulos em uma única aplicação. Essas ferramentas visam garantir a interoperabilidade entre diferentes tecnologias, preservando a independência e a autonomia do microfrontend.

Dentre as abordagens existentes, destacamos as soluções baseadas em *Web Components*, a integração através de iframes e *frameworks* de orquestração. Cada uma dessas estratégias tem vantagens e limitações, variando de acordo com o nível de isolamento desejado, a complexidade da aplicação e os requisitos de desempenho.

Web Components, por exemplo, permitem a criação de elementos reutilizáveis e encapsulados, facilitando a integração entre diferentes partes de uma aplicação padronizada. Essa abordagem é baseada em especificações oficiais da web, conforme documentado pela *Mozilla Developer Network* (Mozilla Contributors, 2026). No entanto, pode exigir mais esforço na implementação e adaptação, especialmente em aplicativos que usam estruturas modernas.

Outra abordagem consiste na utilização de iframes, que garantem alto nível de isolamento entre os módulos. Essa estratégia é suportada nativamente pelos navegadores web, conforme especificado pelo *World Wide Web Consortium* (Whatwg, 2026). Apesar disso, pode impactar negativamente a experiência do usuário e dificultar a comunicação entre os microfrontends, sendo menos utilizada em aplicações modernas.

Nesse contexto, *frameworks* de orquestração têm se destacado como uma solução mais flexível e eficiente para a implementação de microfrontends. Entre esses *frameworks*, destaca-se o single-spa, amplamente utilizado para integrar aplicações desenvolvidas com diferentes tecnologias em um único ambiente, conforme descrito em sua documentação oficial (Canopy Taxi, 2025a).

Um único SPA atua como um orquestrador, permitindo o registro, carregamento e gerenciamento dinâmicos de múltiplos microfrontends. Essa abordagem permite que cada módulo seja desenvolvido e implementado separadamente, mantendo a coerência da aplicação como um todo. Além disso, esse *framework* suporta a integração de diversos *frameworks*, como Angular, React e Vue.js, tornando a solução adequada para ambientes heterogêneos.

Para viabilizar essa integração, o single-spa frequentemente é utilizado em conjunto com ferramentas como o *SystemJS*, responsável pelo carregamento dinâmico de módulos, e os *import maps*, que permitem a resolução de dependências em tempo de execução. (Systemjs Contributors, 2024; Whatwg, 2026). Essa combinação possibilita uma arquitetura desacoplada, na qual os microfrontends podem ser carregados sob demanda, contribuindo para a flexibilidade e escalabilidade da aplicação.

Assim, a escolha de ferramentas para implementar microfrontends deve considerar fatores como interoperabilidade, eficiência, facilidade de manutenção e suporte a múltiplas tecnologias. O uso de estruturas de orquestração, como o single-spa, tem se mostrado um método eficaz para lidar com a complexidade inerente a esse tipo de arquitetura, especialmente em situações que envolvem desenvolvimento colaborativo e integração entre equipes.

2.5 DEFINIÇÃO DOS *FRAMEWORKS* UTILIZADOS

No contexto do desenvolvimento de aplicações web modernas, diversos *frameworks* têm sido amplamente utilizados para a construção de interfaces de usuário. Entre os principais, destacam-se o React, o Angular e o Vue.js, cada um com características específicas que influenciam sua adoção em projetos de software.

React é uma biblioteca JavaScript que se concentra na construção de interfaces baseadas em componentes reutilizáveis, com forte ênfase em desempenho e atualizações eficientes da interface por meio do conceito de DOM Virtual (Meta, 2022). Angular, por outro lado, é um *framework* completo que oferece uma estrutura robusta para o desenvolvimento de aplicações complexas, incluindo gerenciamento de estado, roteamento e injeção de dependência (Google, 2021). Vue.js, por sua vez, oferece uma abordagem progressiva, permitindo uma adoção gradual

e oferecendo simplicidade combinada com flexibilidade (You, 2021). Esses três *frameworks* foram selecionados para compor a aplicação experimental deste estudo, permitindo avaliar, na prática, a integração de diferentes tecnologias por meio da arquitetura de microfrontends.

A utilização desses diferentes *frameworks* em uma mesma aplicação é viabilizada pela arquitetura de microfrontends, permitindo que equipes adotem tecnologias distintas conforme suas necessidades, sem comprometer a integração do sistema como um todo.

2.6 ARQUITETURA MONOLÍTICA VS MICROFRONTENDS

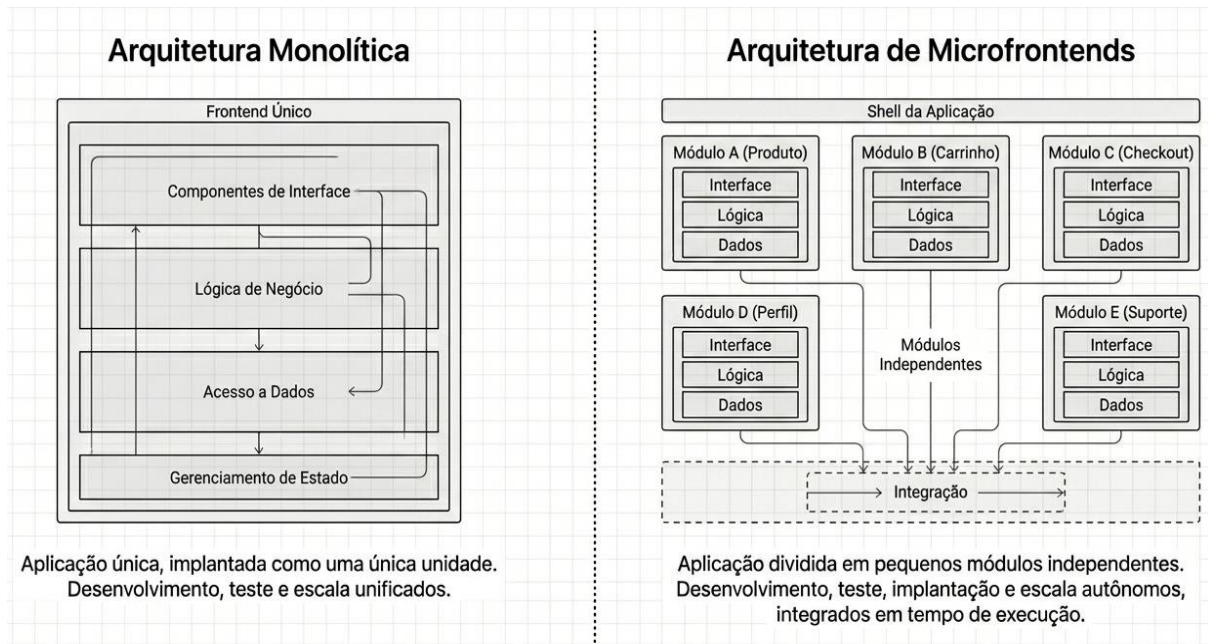
Tradicionalmente, aplicações web foram desenvolvidas utilizando arquiteturas monolíticas, nas quais todo o *front-end* é construído como uma única aplicação integrada. Nesse modelo, todas as funcionalidades compartilham o mesmo código-base, o que pode facilitar o desenvolvimento inicial, mas tende a gerar dificuldades à medida que o sistema cresce.

Em arquiteturas monolíticas, as equipes geralmente trabalham nos mesmos pacotes e dependências, o que pode aumentar o acoplamento entre os componentes e dificultar a manutenção e o desenvolvimento do sistema. Além disso, a necessidade de tecnologias de propósito único para todo o projeto pode limitar a flexibilidade e a adaptabilidade a novos requisitos.

Por outro lado, a arquitetura microfrontend propõe dividir os recursos em módulos independentes, permitindo que diferentes equipes desenvolvam diferentes partes da aplicação de forma autônoma. Essa abordagem reduz o acoplamento, melhora a organização do código e facilita o desenvolvimento paralelo. A Figura 1 apresenta uma comparação entre arquiteturas monolíticas e baseadas em microfrontends, evidenciando diferenças estruturais relacionadas ao acoplamento, modularidade e organização do sistema.

Conforme ilustrado na Figura 1, observa-se que a arquitetura monolítica concentra todas as funcionalidades em um único sistema, o que pode facilitar o desenvolvimento inicial, porém dificulta a escalabilidade e a manutenção à medida que o sistema evolui. Em contrapartida, a arquitetura de microfrontends promove a separação da aplicação em módulos independentes, permitindo maior flexibilidade, autonomia das equipes e suporte ao desenvolvimento colaborativo.

Figura 1 – Comparativo: Monolito vs Microfrontends.



Fonte: Elaborado pelo autor (2026).

No entanto, apesar das vantagens, os microfrontends também introduzem desafios, como a necessidade de gerenciar a integração entre módulos e garantir a consistência da aplicação. Dessa forma, a escolha entre arquitetura monolítica e microfrontends deve considerar fatores como complexidade do sistema, tamanho da equipe e necessidade de escalabilidade.

2.7 APLICAÇÕES REAIS DE MICROFRONTENDS

No contexto do desenvolvimento de aplicações web modernas, diversos *frameworks* têm sido amplamente utilizados para a construção de interfaces de usuário. Entre os principais, destacam-se React, Angular e Vue.js, cada um com características específicas que influenciam sua adoção em projetos de software.

React é uma biblioteca JavaScript voltada à construção de interfaces baseadas em componentes reutilizáveis, oferecendo mecanismos eficientes para atualização da interface (Meta, 2022). Angular, por outro lado, é um *framework* voltado ao desenvolvimento de aplicações web, oferecendo recursos integrados para organização e desenvolvimento da aplicação (Google, 2021). Vue.js caracteriza-se como um *framework* JavaScript progressivo, permitindo adoção gradual e desenvolvimento baseado em componentes (You, 2021). Esses três *frameworks* foram selecionados para compor a aplicação experimental deste estudo, permitindo avaliar, na prática, a integração de diferentes tecnologias por meio da arquitetura de microfrontends.

A utilização desses diferentes *frameworks* em uma mesma aplicação é viabilizada pela

arquitetura de microfrontends, permitindo que equipes adotem tecnologias distintas conforme suas necessidades, sem comprometer a integração do sistema como um todo.

3 METODOLOGIA

3.1 CARACTERIZAÇÃO DA PESQUISA

A presente investigação caracteriza-se como pesquisa aplicada, focada na solução de problemas práticos na engenharia de software ao analisar microfrontends em um cenário real (Gil, 2019). Adota-se uma abordagem quantitativa para analisar variáveis através de mensuração numérica e análise estatística, validando as percepções dos desenvolvedores de forma objetiva (Sampieri; Collado; Lucio, 2013).

Quanto aos objetivos, o estudo classifica-se como pesquisa descritiva, buscando registrar o impacto do uso de microfrontends no desenvolvimento colaborativo, focando em aspectos como divisão de responsabilidades, integração, navegação e escalabilidade (Gil, 2019). O percurso estruturado visa garantir a validade científica, delimitando o fenômeno sem interferir nas variáveis técnicas (Marconi; Lakatos, 2017).

O delineamento adota natureza experimental, com a implementação de uma aplicação para avaliação prática pelos participantes (Gil, 2019; Yin, 2016). O experimento foi conduzido em ambiente controlado, no qual todos os participantes avaliaram a mesma aplicação e responderam ao mesmo instrumento de coleta de dados. O procedimento envolveu o desenvolvimento da aplicação, a elaboração de materiais explicativos e a aplicação de um questionário estruturado, cujos dados foram tratados por estatística descritiva para ordenação e interpretação (Marconi; Lakatos, 2017).

3.2 PLANEJAMENTO DO ESTUDO EXPERIMENTAL

O estudo experimental foi planejado com o objetivo de possibilitar a avaliação prática de uma aplicação desenvolvida com arquitetura de microfrontends. Para isso, foi utilizado um sistema de agendamento de serviços de barbearia como cenário experimental, por apresentar funcionalidades que podem ser segmentadas em módulos independentes. A aplicação contempla dois perfis de usuários cliente e barbeiro, permitindo simular um ambiente de desenvolvimento semelhante ao encontrado em aplicações reais. A descrição detalhada da aplicação, incluindo seus requisitos e funcionalidades, é apresentada na Seção 4.1.

Participaram da pesquisa desenvolvedores de software com diferentes perfis de atuação, incluindo profissionais das áreas de desenvolvimento *front-end*, *back-end* e *full stack*. A participação ocorreu de forma voluntária e anônima.

Antes da avaliação, os participantes receberam um vídeo explicativo contendo a contextualização do estudo, os conceitos relacionados à arquitetura de microfrontends e uma apresentação da aplicação experimental. Em seguida, tiveram acesso ao sistema hospedado na plataforma Netlify (Netlify, 2026) e, opcionalmente, ao repositório público do projeto no GitHub (Github, 2026; Software Freedom Conservancy, 2026) para consulta à implementação.

3.3 INSTRUMENTOS DE COLETA DE DADOS

A coleta de dados foi realizada por meio de um questionário eletrônico estruturado, disponibilizado após a apresentação do vídeo explicativo e a interação dos participantes com a aplicação experimental.

O instrumento foi composto por dez questões fechadas. As três primeiras tiveram como finalidade caracterizar o perfil dos respondentes, considerando área de atuação, tempo de experiência em desenvolvimento de software e contato prévio com a arquitetura de microfrontends. As demais questões foram elaboradas utilizando escala Likert de cinco pontos. O questionário completo aplicado aos participantes encontra-se no Apêndice A deste trabalho.

Os itens avaliados contemplaram aspectos relacionados à divisão de responsabilidades entre equipes, integração entre módulos, fluidez da navegação, contribuição para a escalabilidade, impacto na complexidade do desenvolvimento, potencial para colaboração e avaliação geral da arquitetura proposta.

Dessa forma, o questionário permitiu identificar tendências na percepção dos participantes acerca dos benefícios e desafios da adoção de microfrontends em contextos de desenvolvimento colaborativo.

3.4 PROCEDIMENTOS DE ANÁLISE DOS DADOS

Os dados coletados foram analisados por meio de estatística descritiva. Inicialmente, as respostas relacionadas ao perfil dos participantes foram organizadas por frequência absoluta e percentual, possibilitando a caracterização da amostra quanto à área de atuação, experiência profissional e familiaridade com microfrontends.

Em relação às questões estruturadas em escala *Likert*, foram calculadas as frequências,

os percentuais e as médias das respostas, com o objetivo de identificar tendências na percepção dos desenvolvedores sobre a arquitetura avaliada. Os resultados foram apresentados por meio de tabelas e gráficos, facilitando a interpretação dos dados.

Por fim, os dados obtidos foram discutidos com base na fundamentação teórica, estabelecendo relações entre as percepções dos participantes e os benefícios e desafios da arquitetura de microfrontends reportados na literatura especializada.

4 DESENVOLVIMENTO DA APLICAÇÃO EXPERIMENTAL

4.1 VISÃO GERAL DO SISTEMA

A aplicação experimental desenvolvida neste trabalho consiste em um sistema web de agendamento de serviços de barbearia. O sistema foi concebido com o objetivo de simular um cenário de desenvolvimento colaborativo baseado na arquitetura de microfrontends, no qual diferentes funcionalidades são organizadas em módulos independentes e integrados.

O domínio escolhido permite representar funcionalidades distintas dentro de uma mesma aplicação, envolvendo perfis de usuários com responsabilidades específicas. Nesse contexto, o sistema contempla dois perfis principais: cliente e barbeiro. O cliente é responsável por consultar serviços, visualizar horários disponíveis e realizar agendamentos. Já o barbeiro possui acesso às funcionalidades de gerenciamento de horários, serviços e atendimentos.

A aplicação foi desenvolvida como um ambiente experimental, não tendo como finalidade atender integralmente aos requisitos de um sistema comercial, mas servir como base para a análise prática da arquitetura proposta. Dessa forma, o foco esteve na modularização da interface, na independência dos módulos e na integração entre diferentes tecnologias de desenvolvimento *front-end*.

Os principais requisitos funcionais implementados na aplicação são apresentados na Tabela 1.

Tabela 1 – Principais requisitos funcionais da aplicação.

Código	Requisito Funcional
RF01	Permitir a autenticação de usuários no sistema
RF02	Disponibilizar área específica para o perfil cliente
RF03	Disponibilizar área específica para o perfil barbeiro
RF04	Permitir a visualização de barbeiros cadastrados

RF05	Permitir a visualização dos serviços disponíveis
RF06	Permitir a consulta de horários disponíveis
RF07	Permitir a realização de agendamentos
RF08	Permitir o gerenciamento de horários pelo barbeiro
RF09	Permitir o gerenciamento de serviços pelo barbeiro
RF10	Permitir a visualização dos agendamentos registrados

Fonte: Elaborado pelo autor (2026).

Além dos requisitos funcionais, foram definidos requisitos não funcionais diretamente relacionados aos objetivos da pesquisa e às características da arquitetura de microfrontends. Esses requisitos são apresentados na Tabela 2.

Tabela 2 – Principais requisitos não funcionais da aplicação.

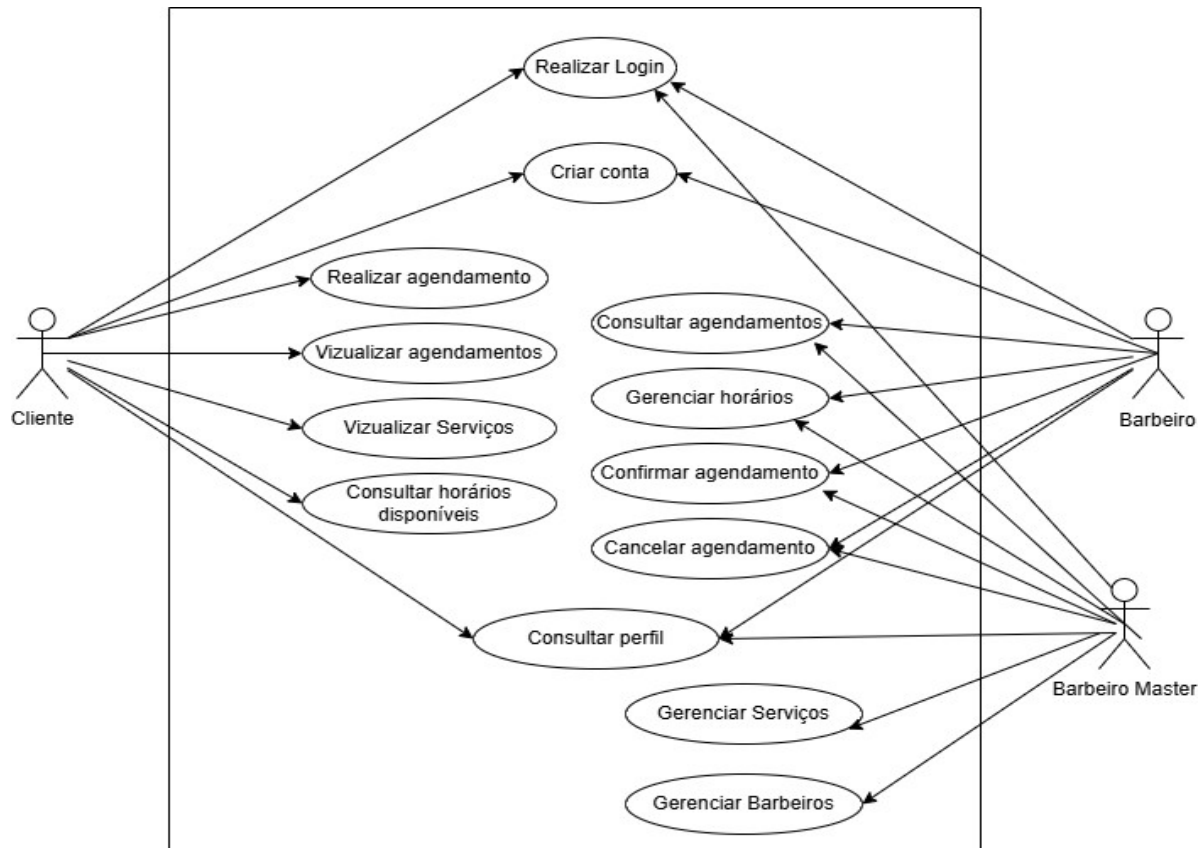
Código	Requisito Não Funcional
RNF01	Garantir a modularidade da aplicação por meio de aplicação web microfrontends independentes
RNF02	Promover a separação de responsabilidades entre os módulos
RNF03	Permitir a utilização de diferentes <i>frameworks</i> em uma mesma aplicação
RNF04	Manter identidade visual consistente entre os módulos
RNF05	Possibilitar o carregamento dinâmico dos microfrontends
RNF06	Facilitar a manutenção e evolução individual dos módulos
RNF07	Assegurar a integração dos módulos por meio do <i>framework</i> single-spa

Fonte: Elaborado pelo autor (2026).

Dessa forma, o sistema de agendamento de serviços de barbearia foi utilizado como ambiente experimental para demonstrar como a decomposição do *front-end* em módulos independentes pode contribuir para a organização da aplicação e para a simulação de cenários de desenvolvimento colaborativo.

A Figura 2 apresenta o diagrama de caso de uso da aplicação experimental. De acordo com a Unified Modeling Language (UML), os diagramas de caso de uso representam as interações entre os atores externos e as funcionalidades disponibilizadas por um sistema, auxiliando na identificação e na compreensão de seus requisitos funcionais (Object Management Group, 2017). O diagrama identifica os principais atores do sistema: cliente, barbeiro e barbeiro master, bem como as funcionalidades disponibilizadas para cada perfil de usuário. Foram representadas as associações diretas entre os atores e os casos de uso, priorizando a visualização dos requisitos funcionais essenciais da aplicação.

Figura 2 – Diagrama de caso de uso da aplicação experimental.



Fonte: Elaborado pelo autor (2026).

Conforme apresentado na Figura 2, o cliente utiliza o sistema principalmente para realizar agendamentos e consultar horários disponíveis. O barbeiro utiliza funcionalidades voltadas ao gerenciamento de horários, serviços e atendimentos. Já o barbeiro master possui permissões ampliadas, podendo gerenciar horários e também cadastrar, editar ou remover informações relacionadas aos barbeiros do sistema. Essa separação funcional reforça a decomposição por domínio adotada na arquitetura de microfrontends, pois as responsabilidades de cada perfil estão concentradas em módulos específicos da aplicação.

4.2 ARQUITETURA PROPOSTA

A arquitetura proposta foi estruturada com base no conceito de microfrontends, no qual a interface da aplicação é dividida em módulos independentes, cada um responsável por um domínio funcional específico. Diferentemente de uma aplicação *front-end* monolítica, em que todas as funcionalidades são concentradas em um único código-base, a abordagem adotada permite que partes distintas do sistema sejam desenvolvidas, mantidas e evoluídas de forma separada.

A aplicação foi organizada utilizando uma decomposição vertical, na qual cada

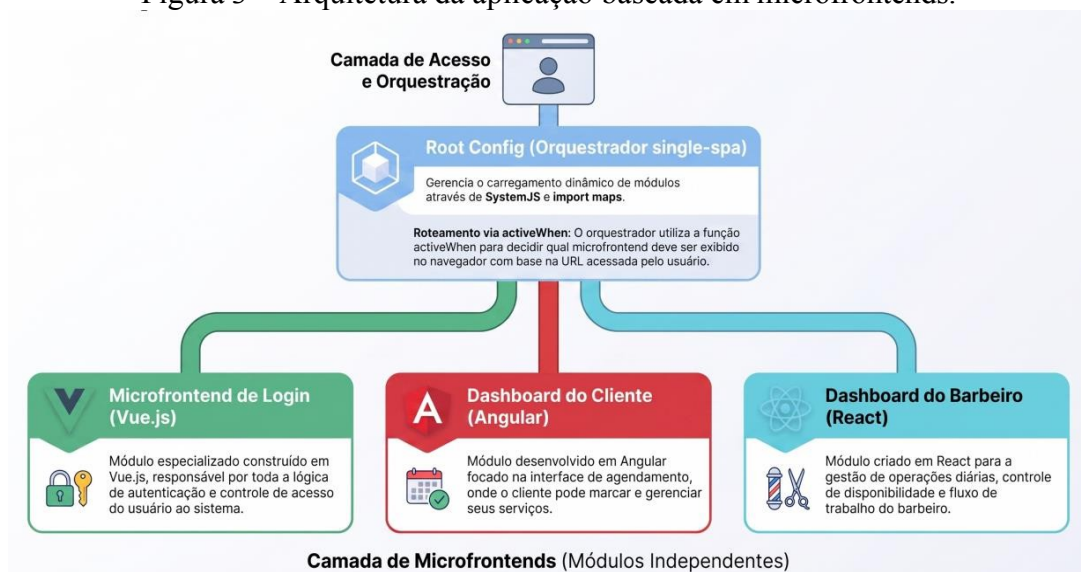
microfrontend representa um contexto funcional completo. Assim, cada módulo não corresponde apenas a um componente visual isolado, como cabeçalho, menu ou rodapé, mas a uma parte funcional da aplicação, contendo sua própria interface, lógica de negócio e fluxo de navegação. Essa estratégia favorece a autonomia dos módulos e aproxima o experimento de um cenário real de desenvolvimento colaborativo.

A arquitetura geral da aplicação é composta por um módulo central, denominado root-config, responsável pela orquestração dos microfrontends, e por três microfrontends principais: o módulo de autenticação, o módulo do cliente e o módulo do barbeiro. O root-config atua como ponto de entrada da aplicação, controlando o carregamento dinâmico dos módulos conforme as rotas acessadas pelo usuário.

A orquestração dos microfrontends foi realizada por meio do *framework* single-spa, que permite integrar aplicações desenvolvidas com diferentes *frameworks* em um único ambiente. No contexto deste projeto, o carregamento dos módulos ocorre em tempo de execução, de acordo com a navegação do usuário. Assim, ao acessar determinada rota, o microfrontend correspondente é carregado e exibido na aplicação.

Conforme apresentado na Figura 3, o usuário acessa a aplicação por meio do navegador web, que se comunica com o módulo central de orquestração. Esse módulo é responsável por registrar, carregar e ativar os microfrontends conforme as rotas definidas. Na camada de microfrontends, encontram-se os módulos independentes da aplicação: autenticação, cliente e barbeiro. Cada um deles foi desenvolvido com uma tecnologia específica, evidenciando a possibilidade de coexistência entre diferentes *frameworks* dentro de uma mesma aplicação.

Figura 3 – Arquitetura da aplicação baseada em microfrontends.



Fonte: Elaborado pelo autor (2026).

A escolha por utilizar diferentes *frameworks* teve finalidade experimental. Embora a arquitetura de microfrontends não exija obrigatoriamente o uso de múltiplas tecnologias, essa decisão foi adotada para evidenciar a independência entre os módulos e demonstrar como equipes distintas poderiam utilizar tecnologias diferentes sem comprometer a integração geral do sistema.

Para garantir consistência visual entre os microfrontends, foi utilizada a biblioteca CSS Tailwind CSS nos módulos desenvolvidos com Vue.js, Angular e React (Google, 2021; You, 2021; Meta, 2022). A adoção de classes utilitárias comuns permitiu padronizar elementos da interface, como cores, espaçamentos, tipografia, botões, campos de formulário e organização dos layouts (Tailwind Labs, 2026). Dessa forma, apesar de cada módulo utilizar um *framework* distinto, a aplicação mantém uma identidade visual uniforme e é percebida pelo usuário como um único sistema integrado.

4.3 ESTRUTURA DOS MÓDULOS

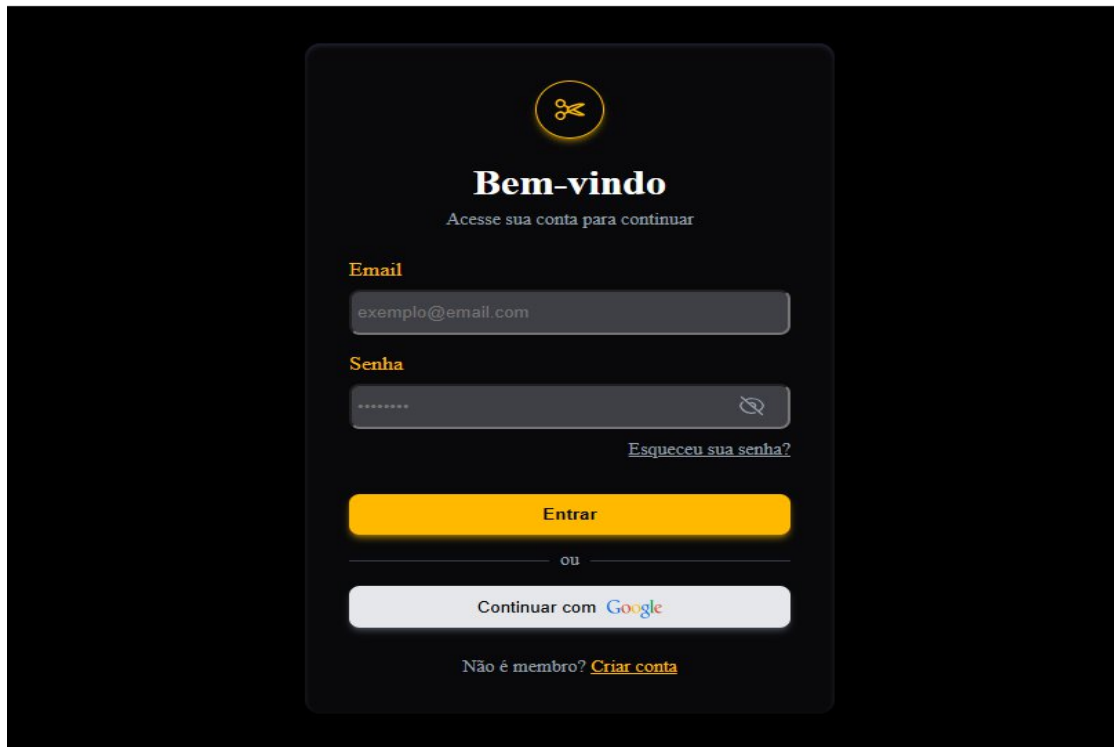
A aplicação experimental foi dividida em três microfrontends principais, além do módulo central de orquestração. Essa divisão teve como objetivo representar uma organização modular, na qual diferentes equipes poderiam atuar paralelamente sobre partes distintas do sistema.

4.3.1 Módulo de autenticação

O módulo de autenticação é responsável pelo fluxo inicial de acesso à aplicação (You, 2021). Esse microfrontend foi desenvolvido utilizando Vue.js 2.6.14 e tem como principal função permitir que o usuário acesse o sistema e seja direcionado conforme o perfil correspondente.

Esse módulo opera de forma independente em relação aos demais microfrontends, sendo carregado quando a rota de autenticação é acessada. No contexto do experimento, foi disponibilizado um perfil de demonstração para facilitar a avaliação pelos participantes, permitindo o acesso ao sistema com dados previamente cadastrados.

Figura 4 – Tela do módulo de autenticação da aplicação experimental.



Fonte: Elaborado pelo autor (2026).

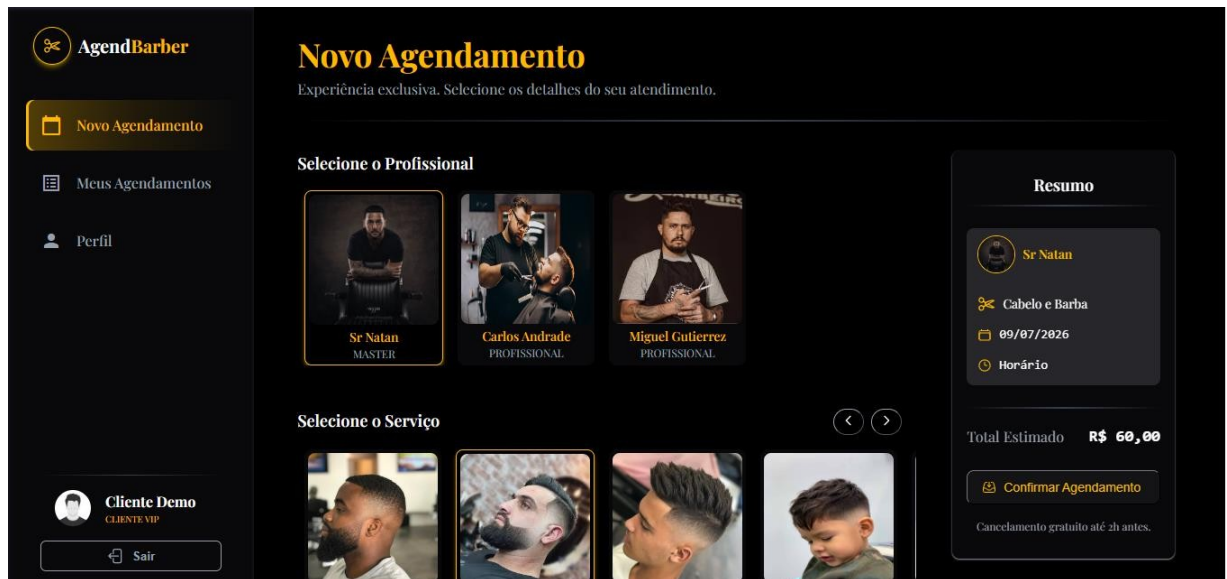
A Figura 4 apresenta a tela de autenticação da aplicação. A partir dela, o usuário pode iniciar o fluxo de acesso ao sistema. Embora a tela represente apenas o ponto inicial da aplicação para o usuário final, do ponto de vista arquitetural ela constitui um microfrontend independente, com sua própria estrutura interna e integração ao orquestrador central.

4.3.2 Módulo do cliente

O módulo do cliente é responsável pelas funcionalidades relacionadas ao usuário que realiza agendamentos. Esse microfrontend foi desenvolvido utilizando Angular 13 e concentra as principais operações do cliente dentro da aplicação, como navegação pelo painel, consulta de informações, escolha de serviços e realização de agendamentos.

Esse módulo representa um domínio funcional completo, pois encapsula sua própria interface, lógica e fluxo de navegação. Dessa forma, as funcionalidades do cliente permanecem organizadas em um módulo independente, reduzindo o acoplamento com os demais módulos da aplicação.

Figura 5 – Tela de agendamento de serviço no módulo do cliente.



Fonte: Elaborado pelo autor (2026).

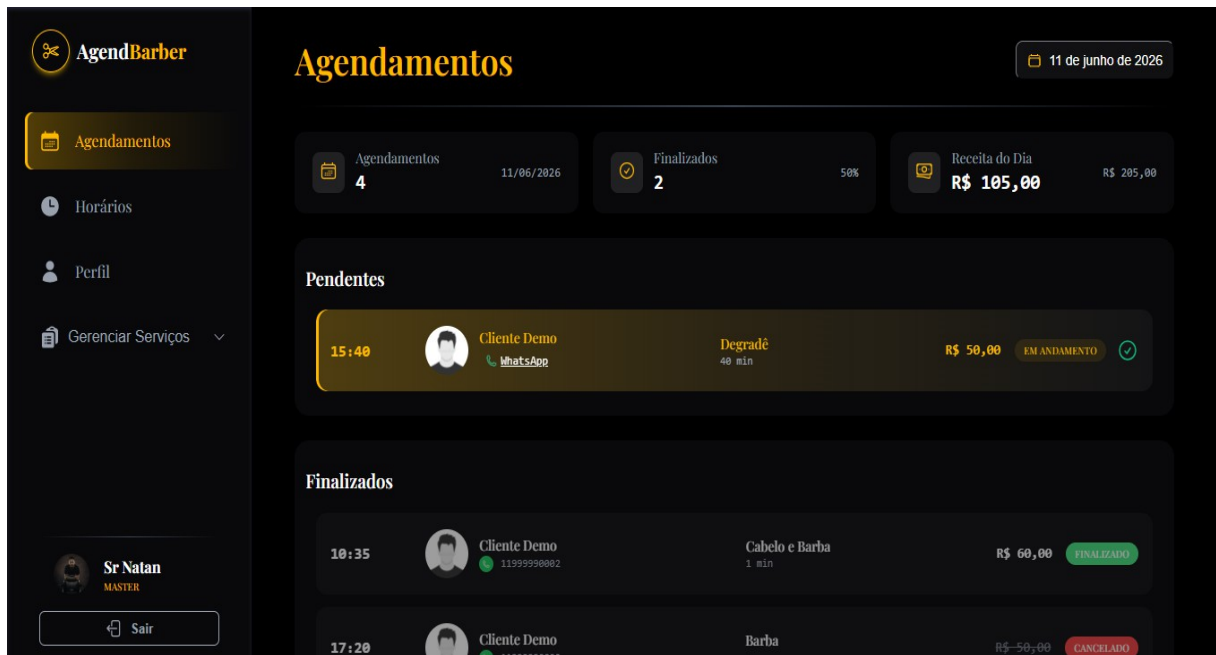
A Figura 5 apresenta a funcionalidade de agendamento de serviço. Nessa tela, o cliente pode selecionar informações necessárias para realizar o agendamento, como barbeiro, serviço e horário disponível. Essa funcionalidade demonstra que o microfrontend do cliente não atua apenas como um conjunto de componentes visuais, mas como um módulo funcional completo dentro da aplicação.

4.3.3 Módulo do barbeiro

O módulo do barbeiro é responsável pelas funcionalidades administrativas relacionadas ao gerenciamento da barbearia. Esse microfrontend foi desenvolvido utilizando React 18 e contempla operações voltadas ao controle de atendimentos, horários e serviços.

Assim como os demais módulos, o microfrontend do barbeiro possui estrutura própria e funcionamento independente. Essa separação permite que as funcionalidades associadas ao barbeiro sejam desenvolvidas e mantidas sem interferir diretamente no módulo do cliente ou no módulo de autenticação.

Figura 6 – Tela de gerenciamento no módulo do barbeiro.



Fonte: Elaborado pelo autor (2026).

A Figura 6 apresenta uma funcionalidade de gerenciamento disponível no módulo do barbeiro. Essa área permite demonstrar a diferença entre os fluxos do cliente e do barbeiro, evidenciando que cada microfrontend possui responsabilidades próprias dentro do sistema.

Apesar de os módulos terem sido desenvolvidos com *frameworks* diferentes, a aplicação mantém uma identidade visual padronizada. Essa padronização foi obtida por meio do uso consistente de estilos, cores, botões e componentes visuais semelhantes, contribuindo para que o usuário final perceba o sistema como uma aplicação única, e não como um conjunto fragmentado de aplicações independentes.

4.4 PROCESSO DE INTEGRAÇÃO

A integração entre os microfrontends foi realizada por meio do *framework* single-spa, utilizado como mecanismo central de orquestração. Esse *framework* permite registrar diferentes aplicações *front-end* e controlar seu carregamento, montagem e desmontagem conforme as rotas acessadas pelo usuário (Canopy Taxi, 2025b).

O módulo central da aplicação, denominado root-config, foi desenvolvido com Vue.js 2.6.14 e configurado como ponto principal de entrada do sistema. Nele foram definidos os mecanismos responsáveis pelo registro dos microfrontends e pela associação de cada módulo às suas respectivas rotas. Dessa forma, quando o usuário acessa uma determinada área da aplicação, o root-config identifica qual microfrontend deve ser ativado.

Cada microfrontend foi integrado ao *single-spa* por meio de adaptadores específicos. O módulo de autenticação, desenvolvido com Vue.js 2.6.14, utilizou o adaptador *single-spa-vue*. O módulo do cliente, desenvolvido com Angular (Google, 2021), utilizou o *single-spa-Angular*. Já o módulo do barbeiro, desenvolvido com *React*, foi integrado por meio do *single-spa-react* (Meta, 2022).

Para viabilizar o carregamento dinâmico dos módulos, foram utilizados mecanismos como *SystemJS* e *import maps*. Esses recursos permitem mapear e carregar os microfrontends em tempo de execução, favorecendo a independência entre os módulos e possibilitando que cada um deles seja desenvolvido e executado separadamente.

A estrutura do projeto foi organizada em diretórios independentes. O *root-config* concentra a configuração principal da aplicação; o *login-app* representa o microfrontend de autenticação; o *client-dashboard* corresponde ao módulo do cliente; e o *barber-dashboard* corresponde ao módulo do barbeiro. Essa organização contribui para a separação de responsabilidades e facilita a manutenção individual de cada parte do sistema.

O desenvolvimento foi realizado utilizando Node.js na versão 18.20.4 (Openjs Foundation, 2024) e o gerenciador de pacotes npm (Github, 2026b). As principais tecnologias utilizadas foram Vue.js 2.6.14, Angular 13, React 18 e *single-spa*. Além disso, foram empregadas ferramentas auxiliares para roteamento, construção dos projetos e integração entre os microfrontends.

A aplicação experimental foi hospedada na plataforma *Netlify*, permitindo que os participantes acessassem o sistema durante a etapa de avaliação. O código-fonte também foi disponibilizado publicamente em um repositório no *GitHub*, possibilitando a consulta à estrutura do projeto e contribuindo para a reprodutibilidade do estudo. As instruções para obtenção e execução local do código-fonte encontram-se no Apêndice B deste trabalho.

4.5 DESAFIOS TÉCNICOS DA IMPLEMENTAÇÃO

Durante o desenvolvimento da aplicação experimental, foram identificados desafios relacionados principalmente à compatibilidade entre *frameworks*, configuração dos microfrontends e integração dos módulos no ambiente orquestrado pelo *single-spa*.

A utilização conjunta de Vue.js, Angular e React exigiu a definição de versões compatíveis e a realização de ajustes específicos em cada projeto. Embora o *single-spa* ofereça suporte à integração de diferentes tecnologias, a configuração simultânea de múltiplos

frameworks demanda atenção quanto às dependências, portas de execução, adaptadores e formas de construção dos projetos.

Outro desafio observado esteve relacionado ao carregamento dinâmico dos módulos. Como cada microfrontend possui sua própria estrutura e processo de execução, foi necessário configurar corretamente o módulo central para que as aplicações fossem ativadas conforme as rotas acessadas pelo usuário. Esse processo exigiu testes e ajustes até que os módulos pudessem coexistir de forma integrada dentro da mesma aplicação.

Também foi observada a necessidade de manter uma experiência visual consistente entre os módulos. Como cada microfrontend foi desenvolvido com um *framework* diferente, foi necessário adotar padrões de interface semelhantes para evitar que o usuário percebesse rupturas visuais durante a navegação. Essa padronização foi importante para reforçar a ideia de que, embora a aplicação seja composta por módulos independentes, ela deve se comportar como um sistema único para o usuário final.

Esses desafios evidenciam que a arquitetura de microfrontends oferece vantagens relacionadas à modularidade e à independência entre equipes, mas também exige planejamento técnico, padronização e definição clara de estratégias de integração. Assim, a aplicação desenvolvida permitiu observar, na prática, tanto os benefícios quanto as complexidades envolvidas na adoção dessa abordagem arquitetural.

5 RESULTADOS

Esta seção apresenta os resultados obtidos a partir do questionário aplicado aos desenvolvedores que participaram da avaliação da aplicação experimental. As respostas foram coletadas entre os dias 4 e 15 de junho de 2026. Foram registradas doze respostas válidas, não sendo identificadas respostas incompletas ou duplicadas.

Conforme definido na metodologia, todos os participantes assistiram ao vídeo explicativo e acessaram a aplicação experimental antes de responder ao formulário. Os resultados foram organizados por meio de frequências, percentuais e médias das respostas em escala *Likert*, permitindo analisar a percepção dos participantes sobre a arquitetura de microfrontends no contexto do desenvolvimento colaborativo.

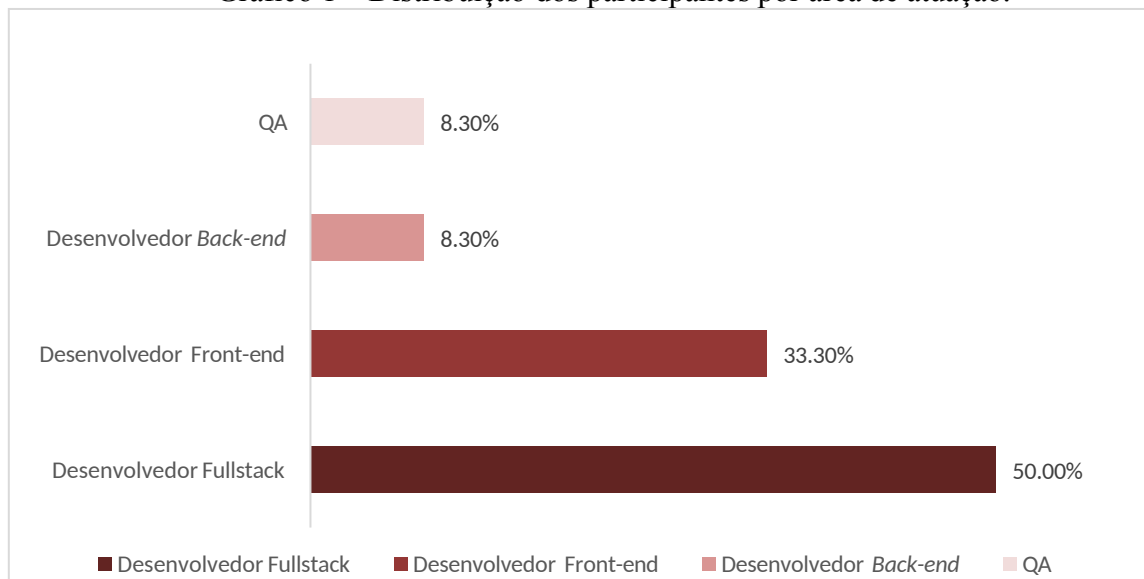
5.1 PERFIL DOS PARTICIPANTES

A caracterização dos participantes é importante para contextualizar os resultados da

avaliação, considerando a área de atuação, a experiência profissional e o contato prévio com a arquitetura de microfrontends.

O gráfico 1 apresenta a distribuição dos participantes conforme a área de atuação profissional. Observa-se que a maior parte dos participantes atuava como desenvolvedor fullstack, correspondendo a 50,0% da amostra. Os desenvolvedores *front-end* representaram 33,3% dos respondentes, enquanto as áreas de *back-end* e QA apresentaram participação de 8,3% cada.

Gráfico 1 – Distribuição dos participantes por área de atuação.



Fonte: Dados da pesquisa (2026).

Em relação ao tempo de experiência com desenvolvimento de software, a Tabela 3 apresenta a distribuição das respostas. Os resultados indicam que a maioria dos participantes possuía entre três e cinco anos de experiência em desenvolvimento de software, representando 41,7% da amostra. Além disso, 25,0% possuíam mais de cinco anos de experiência, enquanto outros 25,0% apresentavam entre um e dois anos de atuação. Apenas um participante possuía menos de um ano de experiência. Esses dados demonstram que a avaliação foi realizada por profissionais com diferentes níveis de conhecimento, predominando participantes com experiência intermediária ou avançada na área de desenvolvimento de software.

Tabela 3 – Tempo de experiência dos participantes em desenvolvimento de software.

Tempo de experiência	Frequência	Percentual
Menos de 1 ano	1	8,3%
1 a 2 anos	3	25,0%
3 a 5 anos	5	41,7%

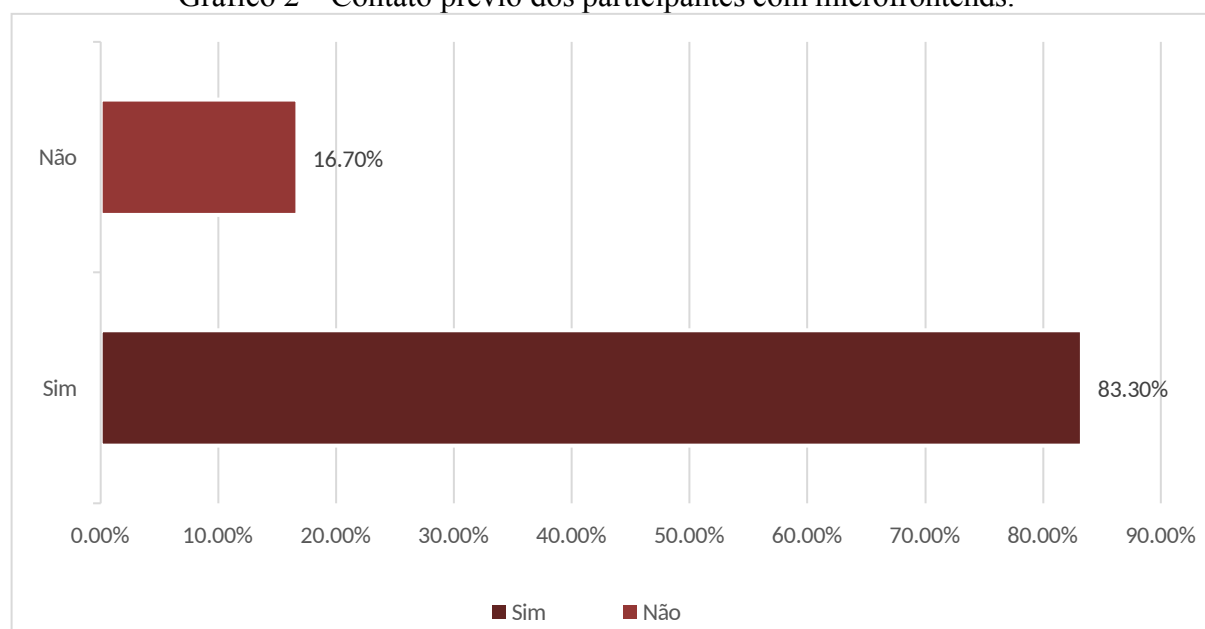
Mais de 5 anos	3	25,0%
-----------------------	---	-------

Fonte: Dados da pesquisa (2026).

Quanto ao contato prévio com microfrontends, sete participantes informaram já conhecer ou ter tido contato com essa arquitetura, enquanto apenas um participante informou não possuir experiência anterior (Gráfico 2). Em relação ao contato prévio com a arquitetura de microfrontends, 83,3% dos participantes afirmaram já possuir algum nível de familiaridade com o tema, enquanto 16,7% relataram não ter experiência anterior.

Esse resultado demonstra que a maior parte da amostra possuía conhecimento prévio sobre a arquitetura avaliada, contribuindo para análises mais fundamentadas acerca de seus benefícios e desafios.

Gráfico 2 – Contato prévio dos participantes com microfrontends.



Fonte: Dados da pesquisa (2026).

5.2 ANÁLISE DA PERCEPÇÃO DOS DESENVOLVEDORES

A avaliação da arquitetura foi realizada por meio de sete questões em escala Likert de cinco pontos. Para interpretação dos dados, os valores mais próximos de cinco representam maior concordância ou avaliação mais positiva, enquanto os valores mais próximos de um representam menor concordância ou avaliação mais negativa.

Os resultados apresentados na Tabela 4 demonstram uma percepção amplamente positiva dos participantes em relação aos benefícios proporcionados pela arquitetura de microfrontends. Destacam-se especialmente os critérios relacionados à fluidez da navegação entre módulos e à contribuição para a escalabilidade da aplicação, ambos com média de 4,75, evidenciando elevado nível de concordância entre os respondentes. Os resultados apresentados

a seguir correspondem às médias obtidas para cada critério avaliado na escala Likert de cinco pontos.

Tabela 4 – Percepção dos participantes sobre os benefícios da arquitetura de microfrontends.

Critério avaliado	Média
Facilidade para divisão do trabalho entre equipes	4,67
Consistência da integração entre microfrontends	4,67
Fluidez da navegação entre módulos	4,75
Contribuição para a escalabilidade da aplicação	4,75

Fonte: Dados da pesquisa (2026).

Conforme apresentado na Tabela 5, os participantes também avaliaram positivamente a contribuição dos microfrontends para o desenvolvimento colaborativo, embora tenham reconhecido um aumento moderado da complexidade do desenvolvimento. Esse resultado sugere que os benefícios associados à modularização e à autonomia das equipes tendem a superar os desafios técnicos decorrentes da adoção dessa arquitetura.

Tabela 5 – Percepção dos participantes sobre complexidade e contribuição para o desenvolvimento colaborativo.

Critério avaliado	Média
Aumento da complexidade do desenvolvimento	3,58
Facilidade para o desenvolvimento colaborativo	4,50
Avaliação geral da contribuição dos microfrontends	4,67

Fonte: Dados da pesquisa (2026).

5.3 SÍNTESE DOS RESULTADOS

De modo geral, os resultados evidenciam uma percepção positiva dos participantes em relação à arquitetura de microfrontends. Os maiores índices foram observados nos critérios relacionados à fluidez da navegação entre módulos e à contribuição para a escalabilidade da aplicação, ambos com média de 4,75. Além disso, a divisão do trabalho entre equipes, a consistência da integração entre os microfrontends e a avaliação geral da arquitetura apresentaram média de 4,67, demonstrando elevado nível de concordância quanto aos benefícios da modularização e da autonomia proporcionadas por essa abordagem.

Por outro lado, o critério referente ao aumento da complexidade do desenvolvimento apresentou a menor média (3,58), indicando que os participantes reconhecem desafios técnicos

associados à implementação e manutenção dos microfrontends. Ainda assim, a facilidade para o desenvolvimento colaborativo obteve média de 4,50, sugerindo que os benefícios percebidos superam as dificuldades identificadas. Dessa forma, os resultados indicam que a arquitetura de microfrontends pode contribuir significativamente para ambientes de desenvolvimento colaborativo, desde que sejam adotadas estratégias adequadas de integração e gerenciamento dos módulos.

5.4 DISCUSSÃO DOS RESULTADOS

Os resultados obtidos indicam que a arquitetura de microfrontends foi percebida como uma abordagem capaz de favorecer a organização do desenvolvimento colaborativo. A avaliação positiva sobre a divisão do trabalho entre equipes está relacionada à proposta de decomposição do *front-end* em módulos independentes, permitindo que cada parte da aplicação seja associada a responsabilidades específicas.

A percepção positiva sobre a integração e a fluidez da navegação também demonstra que, no contexto da aplicação experimental, a adoção de um mecanismo de orquestração permitiu a coexistência dos módulos sem comprometer de forma evidente a experiência do usuário. Esse resultado é relevante, pois uma das principais preocupações associadas aos microfrontends está relacionada à necessidade de manter integração e consistência entre aplicações independentes.

A avaliação favorável sobre escalabilidade reforça a percepção de que a organização modular pode facilitar a evolução da aplicação. Ao dividir o sistema por domínios funcionais, torna-se possível alterar ou ampliar módulos específicos sem necessariamente modificar toda a estrutura do *front-end*. Esse resultado converge com os achados de Cheripurathu e Kulkarni (2024), que apontam a modularização como um dos principais fatores para a evolução contínua de aplicações baseadas em microfrontends.

Por outro lado, o resultado relacionado à complexidade mostra que a adoção de microfrontends envolve desafios técnicos. A integração entre *frameworks*, o gerenciamento de dependências, a configuração do carregamento dinâmico e a padronização visual entre módulos são fatores que podem ampliar o esforço inicial de desenvolvimento. Esse achado está alinhado com os desafios identificados durante a implementação da aplicação experimental e com a literatura apresentada na fundamentação teórica.

Considerando o objetivo geral do estudo, os resultados sugerem que a arquitetura de microfrontends pode contribuir para o desenvolvimento colaborativo ao favorecer a divisão de

responsabilidades, a autonomia dos módulos e a evolução independente de partes da aplicação. Entretanto, sua adoção deve ser acompanhada de estratégias de padronização, governança e integração, a fim de evitar que a autonomia entre módulos resulte em inconsistências ou aumento excessivo da complexidade.

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente estudo teve como objetivo investigar a percepção de desenvolvedores sobre o uso da arquitetura de microfrontends no contexto do desenvolvimento colaborativo. Para alcançar esse objetivo, foi desenvolvida uma aplicação experimental de agendamento de serviços de barbearia, estruturada com base na arquitetura de microfrontends e integrada por meio do *framework single-spa*. Essa aplicação possibilitou a avaliação prática da abordagem por desenvolvedores com diferentes perfis de atuação e níveis de experiência em desenvolvimento de *software*.

Os resultados obtidos demonstraram uma percepção positiva em relação à arquitetura proposta. Os participantes destacaram principalmente aspectos relacionados à escalabilidade da aplicação, à fluidez da navegação entre módulos, à divisão de responsabilidades e à facilitação do trabalho colaborativo entre equipes. Esses resultados indicam que a modularização do *front-end* em componentes independentes pode favorecer a organização do desenvolvimento, a autonomia das equipes e a evolução contínua do sistema.

Em relação aos objetivos específicos, foi possível desenvolver uma aplicação experimental baseada em microfrontends, analisar a percepção dos participantes acerca da arquitetura e identificar características que diferenciam essa abordagem do modelo monolítico tradicional. Os resultados evidenciaram que os microfrontends oferecem vantagens relacionadas à modularidade, integração e colaboração, embora também introduzam desafios associados à configuração, à comunicação entre módulos e à manutenção da consistência da aplicação.

Considerando a hipótese de que a arquitetura de microfrontends contribui para o desenvolvimento colaborativo ao promover maior autonomia entre equipes e melhor divisão de responsabilidades, os resultados obtidos oferecem evidências que sustentam essa hipótese no contexto do estudo realizado. As avaliações atribuídas pelos participantes indicam que os benefícios percebidos superam os desafios identificados, reforçando o potencial dessa abordagem em projetos de software que envolvem múltiplas equipes e sistemas de maior complexidade.

Como limitação da pesquisa, destaca-se o número reduzido de participantes e a utilização de uma única aplicação experimental, fatores que restringem a generalização dos resultados. Além disso, a avaliação foi baseada na percepção dos participantes após a interação com o sistema, não contemplando métricas objetivas de desempenho, produtividade ou manutenção.

Portanto, pode-se concluir que a arquitetura de microfrontends representa uma alternativa para o desenvolvimento de aplicações web modernas, especialmente em situações que exigem colaboração entre equipes, independência tecnológica e escalabilidade. No entanto, sua adoção requer projetos arquitetônicos adequados, a definição de padrões de integração e mecanismos de governança capazes de equilibrar autonomia e consistência entre os módulos.

6.1 TRABALHOS FUTUROS

Como sugestões para pesquisas futuras, recomenda-se que os estudos sejam conduzidos com modelos maiores e mais diversificados, envolvendo especialistas de diferentes empresas e contextos de desenvolvimento. Pesquisas comparativas entre arquiteturas monolíticas e baseadas em microfrontends também podem ser realizadas utilizando métricas objetivas relacionadas à produtividade, desempenho, sustentabilidade e qualidade do software.

Além disso, futuras pesquisas podem explorar mecanismos de comunicação entre microfrontends, estratégias de compartilhamento de componentes, governança arquitetural e impactos da utilização de diferentes *frameworks* em aplicações de grande porte. Por fim, recomenda-se a aplicação da arquitetura em domínios distintos e em ambientes corporativos reais, ampliando a compreensão sobre seus benefícios, limitações e desafios de implementação.

REFERÊNCIAS

- BARROS, C. P. **Banca tradicional: declínio ou reinvenção: os desafios da banca tradicional face ao fintech bancário**. 2024. Dissertação (Mestrado em Economia Monetária, Bancária e Financeira) – Universidade do Minho, Braga, 2024. Disponível em: <https://repositorium.uminho.pt/entities/publication/4dfa56b5-38e6-4a11-9d20-6cafd400d065>. Acesso em: 19 ago. 2026.
- BATISTA, M. V.; PINTO, G. S. Arquitetura de microfrontend. **Revista Interface Tecnológica**, Taquaritinga, v. 20, n. 2, p. 208-218, 2023. DOI: <https://doi.org/10.31510/inf.v20i2.1803>. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/1803>. Acesso em: 14 maio 2026..
- CANOPY TAXI, INC. **single-spa: JavaScript microfrontends framework**. single-spa, 2025a. Disponível em: <https://single-spa.js.org/>. Acesso em: 20 jun. 2026.
- CANOPY TAXI, INC. **The single-spa ecosystem**. single-spa, 2025b. Disponível em: <https://single-spa.js.org/docs/ecosystem/>. Acesso em: 19 ago. 2026.
- CELIK, Julius Recep Colliander. **Plutt: a tool for creating type-safe and version-safe microfrontends**. 2020 79f. Dissertação (Mestrado em Ciência da Computação e Engenharia) – School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Estocolmo, 2020. Disponível em: <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1463748>. Acesso em: 24 jun. 2026.
- CHERIPURATHU, K. G.; KULKARNI, S. Integrating microservices and microfrontends: a comprehensive literature review on architecture, design patterns, and implementation challenges. **Journal of Scientific Research and Technology**, [S. l.], v. 2, n. 7, p. 1-12, jul. 2024. DOI: <https://doi.org/10.61808/jsrt115>. Disponível em: <https://www.jsrtjournal.com/index.php/JSRT/article/view/115>. Acesso em: 15 maio 2026.
- GIL, A. C. **Como elaborar projetos de pesquisa**. 6. ed. São Paulo: Atlas, 2019.
- GITHUB. **GitHub Docs**. GitHub, Inc. 2026. Disponível em: <https://docs.github.com/>. Acesso em: 19 ago. 2026.
- GITHUB. **NPM Docs**. npm, 2026. Disponível em: <https://docs.npmjs.com/>. Acesso em: 19 ago. 2026.
- GOOGLE. **Angular**. Versão 13. 2021. Framework web. Disponível em: <https://blog.Angular.dev/Angular-v13-is-now-available-cce66f7bc296>. Acesso em: 19 ago. 2026.
- LIMA, M. G. T. **Transformação digital na construção civil: lições e desafios a partir da jornada de construtoras do nordeste brasileiro**. 2025. Dissertação (Mestrado em Engenharia Civil – Construção Civil) – Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2025. Disponível em: <https://repositorio.ufc.br/handle/riufc/79850>. Acesso em: 19 ago. 2026.
- MARCONI, M. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. 8. ed. São Paulo: Atlas, 2017.

META. **React**. Versão 18.0. Biblioteca JavaScript. 2022. Disponível em: <https://react.dev/blog/2022/03/29/react-v18>. Acesso em: 19 ago. 2026.

MOZILLA CONTRIBUTORS. **Web Components**. MDN Web Docs, 22 fev. 2026. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/Web_components. Acesso em: 4 jun. 2026.

NASCIMENTO, C. P.; SOTTO, E. C. S. Microfrontend: um estudo sobre o conceito e aplicação no frontend. **Revista Interface Tecnológica**, Taquaritinga, v. 17, n. 1, p. 153-165, 2020. DOI: <https://doi.org/10.31510/inf.v17i1.798>. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/798>. Acesso em: 4 jun. 2026.

NETLIFY. **Netlify Documentation**. Netlify. 2026. Disponível em: <https://docs.netlify.com/>. Acesso em: 19 ago. 2026.

OBJECT MANAGEMENT GROUP. **OMG Unified Modeling Language: version 2.5.1**. Needham: Object Management Group, 2017. Disponível em: <https://www.omg.org/spec/UML/2.5.1/PDF>. Acesso em: 3 jul. 2026.

OPENJS FOUNDATION. **Node.js**. Versão 18.20.4. OpenJS Foundation. 2024. Ambiente de execução JavaScript. Disponível em: <https://nodejs.org/en/blog/release/v18.20.4>. Acesso em: 19 ago. 2026.

PERERA, Y. **Enhancing the front end web applications performance using design patterns and microservices based architecture**. Orientadora: D. B. Jayasuriya. 2023. Trabalho de Conclusão de Curso (Bachelor of Science Honors in Computer Studies) – Faculty of Science, University of Kelaniya, Kelaniya, 2023. DOI: <https://doi.org/10.13140/RG.2.2.36067.53286>. Disponível em: https://www.researchgate.net/publication/373294687_ENHANCING_THE_FRONT_END_WEB_APPLICATIONS_PERFORMANCE_USING_DESIGN_PATTERNS_AND_MICROSERVICES_BASED_ARCHITECTURE. Acesso em: 16 jun. 2026.

SAMPIERI, R. H.; COLLADO, C. F.; LUCIO, M. del P. B. **Metodologia de pesquisa**. 5. ed. Porto Alegre: Penso, 2013.

SCHÄFFER, E.; MAYR, A.; FUCHS, J.; SJAROV, M.; VORNDRAN, J. Microservice-based architecture for engineering tools enabling a collaborative multi-user configuration of robot-based automation solutions. **Procedia CIRP**, v. 86, p. 86-91, 2019. DOI: <https://doi.org/10.1016/j.procir.2020.01.017>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2212827120300172>. Acesso em: 14 jun. 2026.

SOFTWARE FREEDOM CONSERVANCY. **Git Documentation**. Versão 2.55.0. 2026. Disponível em: <https://git-scm.com/docs/git>. Acesso em: 19 ago. 2026.

SVOBODA, M. **Erweiterung von Microservices mit Microfrontends: ein Vergleich von Microfrontend Implementierungen**. 2022. Dissertação (Mestrado em Software Design and Engineering) – FH Campus Wien, Wien, 2022. Disponível em: <https://pub.hcw.ac.at/obvfcwhsacc/content/titleinfo/7942635>. Acesso em: 19 ago. 2026.

SYSTEMJS CONTRIBUTORS. **SystemJS: dynamic ES module loader**. Versão 6.15.1. [S. l.], 2024. Software. Disponível em: <https://github.com/systemjs/systemjs>. Acesso em: 19 ago.

2026.

TAILWIND LABS. **Tailwind CSS**. Tailwind Labs Inc., 2026. Framework CSS. Disponível em: <https://tailwindcss.com/>. Acesso em: 22 jun. 2026.

WHATWG. **HTML Living Standard**. 2026. Disponível em: <https://html.spec.whatwg.org/>. Acesso em: 19 ago. 2026.

YIN, R. K. **Pesquisa qualitativa do início ao fim**. Porto Alegre: Penso, 2016.

YOU, E. **Vue.js**. Versão 2.6.14. Vue.js, 2021. Framework JavaScript. Disponível em: <https://v2.vuejs.org/>. Acesso em: 19 ago. 2026.

YUDIN, A. V.; MAKIEVSKIY, S. E.; ADYSHKIN, S. S.; GROSHEVA, P. Y. Analysis of performance and features of the functioning of microfrontends. **Computational Nanotechnology**, v. 11, n. 1, p. 25-35, 2024. DOI: <https://doi.org/10.33693/2313-223X-2024-11-1-25-35>. Disponível em: <https://journals.eco-vector.com/2313-223X/article/view/631116>. Acesso em: 24 jun. 2026.

APÊNDICE A – QUESTIONÁRIO APLICADO

Avaliação da Arquitetura de Microfrontends no Desenvolvimento Colaborativo

1. Qual sua área de atuação?
2. Quantos anos de experiência você possui com desenvolvimento de software?
3. Você já teve contato com microfrontends antes?
4. A arquitetura de microfrontends facilita a divisão do trabalho entre equipes.
5. A integração entre os microfrontends apresentou boa consistência durante o uso.
6. A navegação entre os módulos ocorreu de forma fluida.
7. A arquitetura de microfrontends contribui para a escalabilidade da aplicação.
8. A arquitetura de microfrontends aumenta a complexidade do desenvolvimento.
9. Essa arquitetura facilita o desenvolvimento colaborativo.
10. De forma geral, como você avalia a contribuição dos microfrontends para o desenvolvimento colaborativo entre equipes?

As questões 4 a 10 foram respondidas por 12 participantes, compostos por profissionais com atuação em desenvolvimento de software e áreas correlatas, em escala Likert de cinco pontos. Nas questões 4 a 9, o valor 1 correspondeu a “Discordo totalmente” e o valor 5 a “Concordo totalmente”. Na questão 10, o valor 1 correspondeu a “Muito negativo” e o valor 5 a “Muito positivo”

APÊNDICE B – MANUAL DE EXECUÇÃO DO PROJETO

B.1 REPOSITÓRIO DO PROJETO

O código-fonte da aplicação experimental está disponível publicamente no GitHub: <<https://github.com/roberto-vianna/microfrontends-singleSpa>>. O repositório contém os módulos independentes que compõem a aplicação, bem como arquivos de configuração e documentação técnica.

B.2 PRÉ-REQUISITOS

Para executar o projeto localmente, é necessário possuir os seguintes recursos instalados:

- Node.js, versão 18.20.4;
- npm, gerenciador de pacotes do Node.js;
- navegador web atualizado, preferencialmente Google Chrome, Mozilla Firefox ou Microsoft Edge;
- Git, caso seja necessário clonar o repositório.

B.3 ESTRUTURA DO PROJETO

A Tabela B.1 apresenta a organização dos módulos da aplicação.

Tabela B.1 – Estrutura dos módulos da aplicação experimental

Módulo	Tecnologia principal	Responsabilidade
<i>root-config</i>	Vue.js 2 e single-spa	Orquestração, registro e carregamento dos microfrontends.
<i>login-app</i>	Vue.js 2	Autenticação e fluxo de acesso ao sistema.
<i>client-dashboard</i>	Angular 13	Área do cliente, agendamentos e consulta de informações.
<i>barber-dashboard</i>	React 18	Área do barbeiro, gestão de horários, serviços e atendimentos.

Fonte: Elaborado pelo autor (2026).

O módulo root-config atua como ponto central da aplicação, sendo responsável por identificar as rotas acessadas e carregar o microfrontend correspondente.

B.4 TECNOLOGIAS UTILIZADAS

A aplicação foi desenvolvida utilizando as seguintes tecnologias:

- Vue.js 2.6.14;
- Angular 13;
- React 18;
- single-spa;
- single-spa-vue;
- single-spa-Angular;
- single-spa-react;
- Node.js 18.20.4;
- npm;
- Tailwind CSS;
- Netlify, para hospedagem da aplicação experimental;
- GitHub, para armazenamento e disponibilização do código-fonte.

B.5 OBTENÇÃO DO CÓDIGO-FONTE

Para obter o código-fonte, deve-se executar o comando a seguir em um terminal: `git clone https://github.com/roberto-vianna/microfrontends-singleSpa.git`

Após o download, deve-se acessar o diretório do projeto:

```
cd microfrontends-singleSpa
```

B.6 INSTALAÇÃO DAS DEPENDÊNCIAS

Cada microfrontend possui dependências próprias. Por esse motivo, as dependências devem ser instaladas individualmente em cada módulo.

No módulo principal de orquestração:

```
cd microfrontend-root/root-config npm install
```

No módulo de autenticação: `cd ../../login-app`

```
npm install
```

No módulo do barbeiro:

```
cd ../barber-dashboard  
npm install
```

No módulo do cliente:

```
cd ../client-  
dashboard npm  
install
```

B.7 EXECUÇÃO LOCAL DA APLICAÇÃO

Após a instalação das dependências, cada microfrontend deve ser executado separadamente.

Para iniciar o módulo central de orquestração:

```
cd microfrontend-root/root-  
config npm run serve
```

Para iniciar o módulo de autenticação:

```
cd login-app  
npm run  
serve
```

Para iniciar o módulo do barbeiro:

```
cd barber-  
dashboard npm  
start
```

Para iniciar o módulo do cliente:

```
cd client-  
dashboard npm  
start
```

O módulo central da aplicação é executado localmente na porta 8081. Os demais microfrontends utilizam portas específicas para possibilitar sua execução independente e posterior integração pelo root-config.

B.8 ACESO À VERSÃO HOSPEDADA

Além da execução local, a aplicação foi disponibilizada em ambiente remoto por meio da plataforma Netlify. A versão hospedada pode ser acessada no endereço: <<https://root-config-mfe.netlify.app/login>>.

A disponibilização do código-fonte e das instruções de execução contribui para a transparência e a reprodutibilidade do estudo, pois permite que outros desenvolvedores consultem a estrutura da aplicação e reproduzam o ambiente experimental.