IFSP – Instituto Federal de Educação,
Ciência e Tecnologia Câmpus São Paulo

ALYSON CÉSAR FUMAGALLI	SP3121071
BRUNO DE ALMEIDA FISCHER	SP3120139
ELIEL DA SILVA	SP3121054
GIOVANNA CAMILLE SILVA CARVALHO	SP3123162
HENRIQUE SANTIAGO PIRES	SP312262X
HENRRIKY JHONNY DE OLIVEIRA BASTOS	SP3123103

BS Beauty Academy

São Paulo – SP – Brasil
2025

BS Beauty Academy

Este projeto integrado de extensão, desenvolvido como parte do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, tem como objetivo desenvolver uma aplicação web para otimizar a gestão e o agendamento de serviços de estética em um ambiente coworking. A área de concentração do projeto é a inovação tecnológica aplicada ao setor de serviços de beleza.

IFSP – Instituto Federal de Educação,
Ciência e Tecnologia Câmpus São Paulo

Orientador: Marcelo Tavares de Santana

São Paulo – SP – Brasil
2025

IFSP – Instituto Federal de Educação,
Ciência e Tecnologia Câmpus São Paulo

BS Beauty Academy

Este projeto integrado de extensão, desenvolvido como parte do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, tem como objetivo desenvolver uma aplicação web para otimizar a gestão e o agendamento de serviços de estética em um ambiente coworking. A área de concentração do projeto é a inovação tecnológica aplicada ao setor de serviços de beleza.

Trabalho aprovado. São Paulo – SP – Brasil, _____ de _____ de 2025:

Marcelo Tavares de Santana
Orientador 1

Johnata Souza Santicioli
Orientador 2

Daniela dos Santos Santana
Convidado 1

Leonardo Andrade Motta de Lima
Convidado 2

São Paulo – SP – Brasil
2025

RESUMO

Este projeto integrado de extensão apresenta o desenvolvimento da aplicação *web* BS *Beauty*, com o objetivo de otimizar a gestão e o agendamento de serviços de beleza no ambiente *coworking*, sob responsabilidade da gestora, nossa parceira de extensão. Como parte de uma iniciativa competitiva, a solução digital desenvolvida tem como propósito melhorar o atendimento e fidelizar clientes, reduzindo o tempo gasto em tarefas administrativas e garantindo uma interface de usuário intuitiva. Para isso, o sistema centraliza agendas, previne conflitos de horário e fornece notificações automáticas, relatórios financeiros e *dashboards* de desempenho. A fim de atender as demandas da nossa parceira, a comunicação constante foi essencial para o levantamento de requisitos, análise de concorrentes e definição de regras de negócio. Para a gestão das etapas do projeto, foi adotado o framework ágil *Scrum*, formalizando o planejamento e controle de tarefas no *ProjectLibre*, um software de gestão de projetos que permite gerenciar cronogramas e alocar recursos para as tarefas definidas. A arquitetura da aplicação foi idealizada em camadas, sendo detalhada em diagramas de componentes e de implantação. Paralelamente, elaborou-se o plano de testes, padronizou-se a documentação e avaliou-se a viabilidade financeira em cenários realistas, otimistas e pessimistas. Como resultado, a aplicação desenvolvida fortalece conhecimentos teóricos do curso de graduação, aproxima-os das demandas de mercado e promove inovação tecnológica no setor de beleza apoiando o modelo de *coworking*.

Palavras-chave: aplicação *web*. agendamento online. *coworking* de beleza. gestão de serviços. *Scrum*.

ABSTRACT

This integrated extension project presents the development of the BS Beauty web application, aimed at optimizing the management and scheduling of beauty services in a coworking environment under the responsibility of the manager, our extension partner. As part of a competitive initiative, the developed digital solution aims to improve service and foster customer loyalty by reducing the time spent on administrative tasks and ensuring an intuitive user interface. Therefore, the system centralizes schedules, prevents scheduling conflicts and provides automatic notifications, financial reports and performance dashboards. In order to meet our partner's demands, constant communication was essential for requirements gathering, competitor analysis and definition of business rules. For project phase management, the agile framework Scrum was adopted, however the planning and task control was formalized in ProjectLibre, a project management software that allows managing schedules and allocating resources to tasks. The application architecture was designed in layers, detailed in component and deployment diagrams. In parallel, the test plan was developed, documentation was standardized, and financial viability was assessed in realistic, optimistic and pessimistic scenarios. As a result, the developed application strengthens the theoretical knowledge of the undergraduate course, aligns it with market demands and promotes technological innovation in the beauty sector by supporting the coworking model.

Keywords: web application. online scheduling. beauty coworking. service management. Scrum.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

R$ Real (moeda brasileira)

US$ Dólar (moeda estadunidense)

% Porcentagem

W Watt

SUMÁRIO

1 INTRODUÇÃO

A ascensão de novos modelos de negócio vem redefinindo o setor da beleza no Brasil e impulsionando a autonomia de seus profissionais. Entre as inovações mais significativas, destaca-se o *coworking* de beleza, que transforma a dinâmica de trabalho ao oferecer infraestrutura compartilhada e flexível.

A ideia do *coworking* de beleza surgiu após a popularização das salas de escritório compartilhadas, denominadas como espaço *coworking* (do inglês "trabalhando em conjunto"), durante a pandemia da covid-19 em 2020. Assim como no formato original, o maior benefício do *coworking* de beleza é a possibilidade dos profissionais autônomos de dividir os altos custos de um salão próprio. Além disso, esse modelo, em específico, libera cabeleireiros, maquiadores, esteticistas e outros profissionais da beleza de depender de parcerias em estabelecimentos de terceiros ou de atender a domicílio, práticas muito comuns nessa área (Beauty Fair, 2024; Gazeta do Povo, 2023).

Essa modernização ocorre em um mercado robusto, que movimentou aproximadamente US$ 27 bilhões em 2024 e posicionou o país entre os cinco maiores do mundo no ramo, evidenciando a necessidade de adaptação contínua dos empreendedores às novas tendências (SEBRAE RS, 2024).

Contudo, à medida que esse formato de trabalho se expande, a gestão eficiente de agendas, espaços e custos torna-se um desafio central para maximizar a autonomia e a rentabilidade. A necessidade de evitar conflitos de reserva e falhas de cobrança, de forma ágil e intuitiva, mostra-se cada vez mais evidente.

Este projeto propõe-se, portanto, a desenvolver uma aplicação web para otimizar reservas, uso de espaços e gestão financeira em ambientes de *coworking* de beleza.

1.1 Objetivos

A aplicação *web* BS Beauty foi desenvolvida especialmente para gerenciar um salão de beleza que opera em modelo *coworking*, sob a gestão da nossa parceira de extensão Bruna. Seu objetivo principal é otimizar os processos internos e centralizar o agendamento de serviços, atendendo tanto às demandas administrativas da gestora quanto às necessidades logísticas dos profissionais autônomos, e sugestões dos clientes finais.

1.2 Problema e Solução Proposta

A gestão de um salão por pequenos empreendedores é frequentemente desafiadora. Ademais, demandas surgem e muitas vezes são realizadas manualmente. Portanto, quando alguma etapa falha, evidencia-se a necessidade de uma solução digital capaz de reduzir erros e diminuir o esforço administrativo.

Por isso, o objetivo geral do projeto é suprir as necessidades de um salão de beleza em modelo *coworking* de forma ágil. Como explicado anteriormente, esse modelo de trabalho é recente (popularizado após a pandemia de *Coronavirus Disease 2019* – Doença por Coronavírus 2019 (COVID-19) em 2020) e atende diferentes profissionais autônomos (relacionados à gerente por locação ou comissão), não uma equipe com objetivo comum. Desta forma, o problema central é a gerência da ocupação de cada profissional no espaço de trabalho, além do controle das finanças e da agenda dos clientes.

Nossa parceira Bruna já utilizava um sistema digital para gerenciamento do salão. Contudo, apesar dos benefícios trazidos pela solução, o sistema apresentava pontos insatisfatórios, sendo o principal deles a instabilidade da plataforma, que gerava insatisfação e perda de clientes.

Nossa solução consiste em criar uma aplicação *web* que mantenha todas as funcionalidades que já atendem bem a Bruna como o agendamento *on-line* e pesquisa de satisfação. Além disso, a plataforma incluirá funções ainda ausentes e ajustará requisitos funcionais e não funcionais cuja concepção é adequada, mas apresenta falhas, como o *login* instável, senhas excessivamente complexas e erros recorrentes na troca de senha. De forma específica, nossa solução facilita o agendamento de serviços para as três entidades existentes no *coworking* de beleza:

Para os Clientes Finais: A plataforma possibilita o agendamento de serviços de forma intuitiva e flexível. Os clientes poderão escolher profissionais específicos ou optar pelo melhor horário disponível, visualizando facilmente a lista de prestadores, seus serviços, preços, tempo de execução e agendas atualizadas.

Para os Profissionais Autônomos: O sistema BS Beauty tem como propósito reforçar a autonomia dos profissionais sobre sua agenda e finanças. A aplicação permite bloquear horários, editar preços e a duração dos serviços, além de acompanhar os agendamentos realizados (sejam eles do dia, futuros ou passados) e visualizar relatórios detalhados com a receita gerada pelos serviços prestados.

Para a Gestora: Nossa parceira, Bruna, terá acesso a funcionalidades exclusivas que incluem análise de métrica de desempenho (a partir de *dashboards)*, gerenciamento do aluguel ou comissão de cada profissional, visualização do fluxo de agendamentos em períodos específicos, envio de mensagens de *marketing* e promoções aos clientes, e acesso a relatórios financeiros detalhados. Ademais, a gestora poderá incluir ou remover profissionais

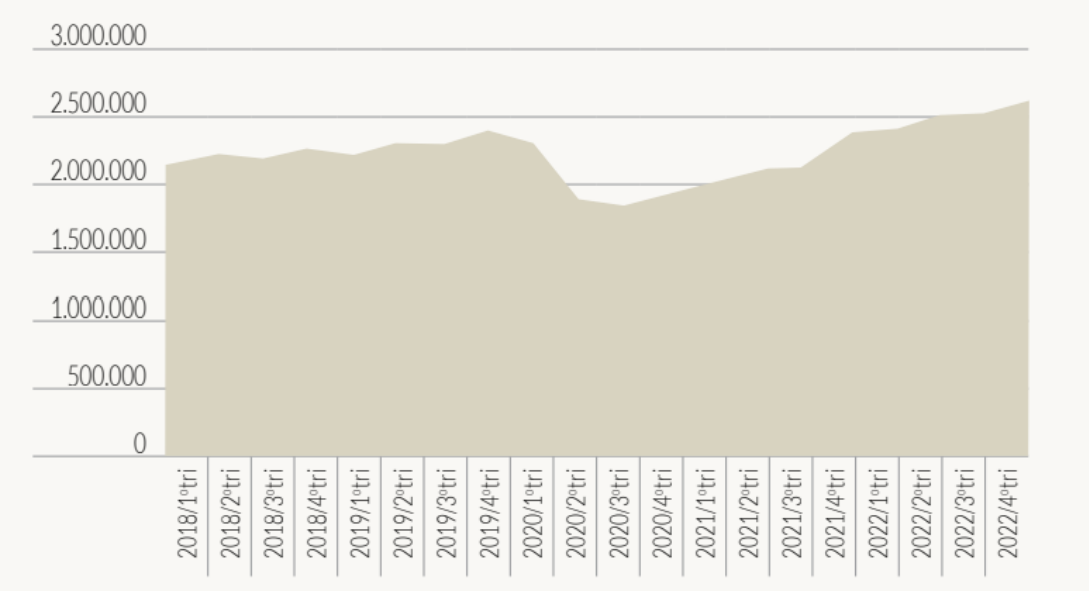
da plataforma conforme a necessidade.

Em síntese, a solução proposta é uma plataforma com *login* simplificado (integrado ao *Single Sign-On* – Autenticação Única (SSO) [1] do Google) e agendamento fácil e transparente para os clientes (incluindo todos os serviços e atributos necessários para uma melhor decisão). Também contará com agenda totalmente controlada pelos profissionais, notificações de agendamento e cancelamento para clientes e profissionais, lista de aniversariantes, desconto por frequência e retenção de dados em conformidade com a Lei Geral de Proteção de Dados – Lei nº 13.709/2018 (LGPD). Além disso, a gerente terá acesso à relatórios financeiros e *dashboards* com métricas de produtividade e frequência de clientes.

1.3 Justificativa

O setor da beleza no Brasil representa um mercado de grande magnitude, com mais de 1,3 milhão de atividades econômicas que geram um faturamento anual de aproximadamente R$ 75 bilhões (SEBRAE, 2023b). O crescimento contínuo no número de profissionais, conforme ilustrado na Figura 1, evidencia a vitalidade e a expansão deste cenário.

Figura 1 – Profissionais da área da beleza no Brasil 2018–2022

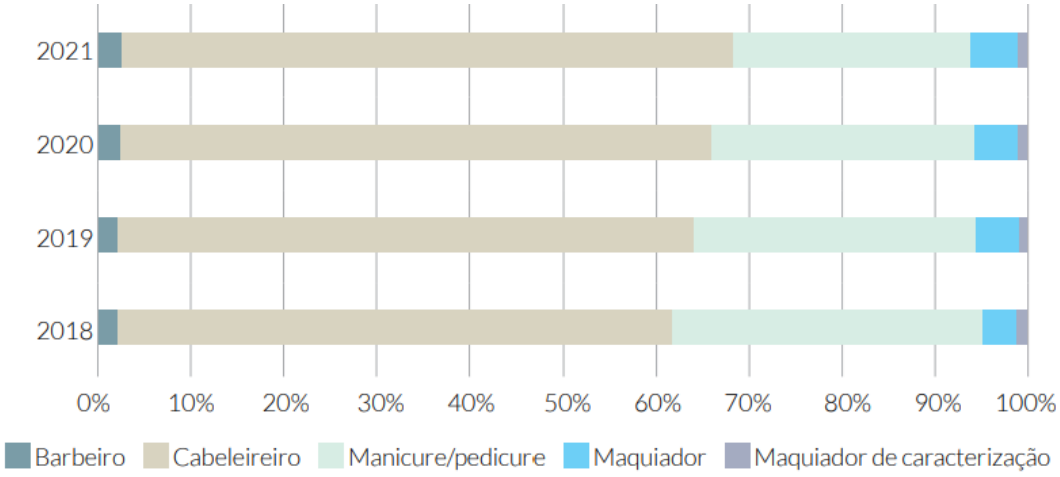


Fonte: (SENAC, 2023)

Dentre o constante crescimento de profissionais neste setor, os cabeleireiros representam a maior parte, conforme indica o gráfico abaixo (2).

[1] Single Sign-On é um sistema que permite usar um único nome de usuário e senha para acessar vários serviços diferentes, sem precisar criar contas ou lembrar várias senhas.

Figura 2 – Distribuição dos profissionais da área da beleza 2018-2021



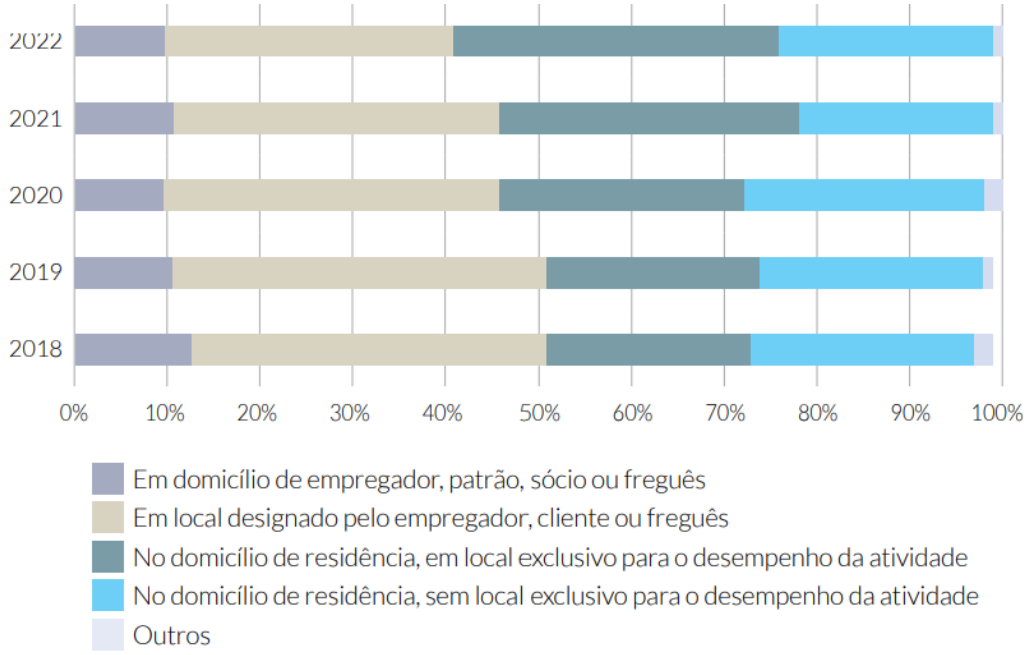
Fonte: (SENAC, 2023)

Apesar da expressividade econômica, o setor enfrenta desafios operacionais que limitam a rentabilidade e a eficiência dos empreendedores. Dados indicam que:

- Até 30% do tempo de um pequeno empreendedor é consumido por tarefas administrativas (SENAC-SP, 2022);
- Taxa média de não comparecimento de clientes atinge 25% (BOOKSY, 2022);
- Perda de 20% da receita por não comparecimento (ABIHPEC, 2021);
- Média de 15 horas semanais são dedicadas ao controle manual de agenda e finanças (FGV, 2020);
- Insatisfação de 40% dos clientes devido a falhas de comunicação e alterações de última hora (MINDMINERS, 2022).

Como resposta à necessidade de reduzir custos fixos e aumentar a flexibilidade, o modelo de *coworking*, originado em ambientes de escritório, expandiu-se para o setor da beleza, permitindo o compartilhamento de espaços e recursos e a redução de custos (SEBRAE, 2023a; SEBRAE SC, 2025). Anteriormente à popularização dos *cowrokings* de beleza, os profissionais se distribuiam em diversos locais para economizar recursos, como mostra a Figura 3:

Figura 3 – Distribuição dos profissionais da área da beleza por local de trabalho 2018-2022



Fonte: (SENAC, 2023)

Embora solucione a questão do investimento em infraestrutura, esse modelo introduz novas complexidades relacionadas à gestão compartilhada de espaços, agendamentos e finanças. Portanto, nesse contexto justifica-se o projeto de extensão *BS Beauty*, uma aplicação *web* customizada para o gerenciamento de salões em modelo *coworking*, sob a coordenação de nossa parceira de extensão Bruna. Ao digitalizar e centralizar processos principais, a BS Beauty empodera pequenos empreendedores por meio da redução de custos operacionais e erros humanos. O sistema oferece controle preciso de comissões e frequências, ao passo que *dashboards* e relatórios financeiros detalhados fornecem *insights* estratégicos para o negócio.

Dessa forma, a plataforma não apenas soluciona os desafios de gestão administrativa, mas também aprimora a experiência do cliente e gera valor para todos os envolvidos. Além disso, como iniciativa de extensão, o projeto simultaneamente permite que os alunos-desenvolvedores apliquem e aprimorem conhecimentos técnicos e de gestão, enfrentando desafios reais de mercado e aproximando a graduação da prática profissional.

1.4 Análise da Concorrência

Foi conduzida uma pesquisa de mercado centrada em plataformas brasileiras que combinam agendamento *on-line* e gestão financeira para espaços de beleza no modelo *coworking*. Deste levantamento emergiram três empresas que servirão de referência nesta análise: uma já amplamente consolidada no mercado nacional — embora atue além do

universo *coworking* — e outras duas que, apesar de conhecidas, ainda estão em expansão, mas com foco mais relacionado ao da nossa proposta, o que as torna concorrentes que merecem maior atenção estratégica.

1.4.1 Trinks

Figura 4 – Logo plataforma Trinks

Fonte: (Trinks Tecnologia da Informação, 2025)

Trinks é uma plataforma já bem consolidada no mercado de gestão de negócios de beleza, com soluções personalizadas para barbearias, salões de beleza e clínicas de estética. Criada em 2012, é hoje a plataforma de gestão para beleza com a maior base instalada do país, englobando aproximadamente 2,8 milhões de usuários e mais de 40 mil estabelecimentos, sediada no Rio de Janeiro. A plataforma começou como um empreendimento de consultoria em software personalizado, mas logo identificou uma oportunidade no mercado da beleza e mudou de nicho. Em 2024, foi adquirida pelo grupo Stone, o que alavancou ainda mais funcionalidades do aplicativo, como o autoatendimento.

Atualmente, a Trinks oferece software de *back-office* (conjunto de módulos internos que controlam o funcionamento do negócio como finanças, estoque, comissões e relatórios), *marketplace* Business to Consumer – *Negócio para Consumidor (B2C)* e meios de pagamento próprios (Trinks *Pay*), funcionando praticamente como um "*Enterprise Resource Planning* – Sistema Integrado de Gestão Empresarial (ERP) + *iFood*" para salões e barbearias. Existe um plano grátis que engloba apenas 150 agendamentos por mês, e os planos pagos variam de R$ 59 a R$ 249/mês (Trinks Tecnologia da Informação, 2025).

Além dos serviços comuns, seus principais diferenciais são:

- **Ponto de Venda (PDV) completo:** integração com Transferência Eletrônica de Fundos (TEF), Pagamento Instantâneo (Pix) e split de comissão, atendendo desde Microempreendedor Individual (MEI)s até redes com exigência de Nota Fiscal de Consumidor Eletrônica (NFC-e) e Sistema Autenticador e Transmissor de Cupons Fiscais (SAT)/Emissor de Cupom Fiscal (ECF);
- **Estrutura em nuvem madura:** Conta com *Service Level Agreement* – Acordo de Nível de Serviço (SLA) de 99,9 % e aplicativos nativos para *iPhone Operating System* – Sistema Operacional do iPhone (iOS)/Android;

- ***Marketplace*:** O domínio *Trinks.com* gera maior fluxo de clientes, expõe o salão ao público final e permite pagamento antecipado.

Apesar dos grandes benefícios, identificamos algumas brechas do ponto de vista do negócio da nossa parceira de extensão, Bruna:

- **Complexidade visual da interface:** Devido ao grande número de funcionalidades voltadas para diversos públicos, a plataforma apresenta uma curva de aprendizado elevada para gestores focados exclusivamente em *coworking*. Para superar essa barreira, o *BS Beauty* propõe uma interface minimalista e focada nas operações essenciais do modelo de negócio da parceira, garantindo maior agilidade e uma experiência de uso intuitiva;
- **Baixo foco no aluguel de estações:** A gestão de locação de espaços não é uma funcionalidade nativa, exigindo configurações manuais de comissão que são pouco eficientes e suscetíveis a erros. Em contrapartida, o *BS Beauty* é projetado com o aluguel de estações como funcionalidade central. O sistema permitirá a configuração de diferentes modelos de locação (por hora ou dia), automatizando o cálculo e o débito dos valores devidos, o que elimina a necessidade de controles manuais;
- **Segregação de funcionalidades em planos superiores:** Recursos importantes para uma gestão completa ficam restritos aos planos mais caros. Já a proposta do *BS Beauty* visa incluir todas as funcionalidades essenciais para a gestão de coworking em um único plano, oferecendo uma solução completa desde o início, sem custos inesperados para acessar ferramentas estratégicas.

1.4.2 Gendo

Figura 5 – Logo plataforma Gendo

Fonte: (Gendo Sistemas, 2025)

Lançado em 2017 e sediado em Curitiba-PR, o Gendo se posiciona como um hub[2] de gestão 100 % em nuvem para negócios além do setor da beleza, como estética, saúde, bem-estar, *pet-shop* e mais recentemente, espaços em formato *coworking*. Atualmente mantém mais de 10 mil assinantes, com maior penetração nas regiões Sul e Sudeste do Brasil. Foi criado no modelo SaaS com o intuito de oferecer prontamente agenda *on-line*,

[2] Hub: plataforma centralizada que integra agenda, PDV, finanças e pagamentos em um único ambiente, funcionando como "nó" que organiza os fluxos de dados do negócio.

automação de lembretes (*e-mail/WhatsApp*), módulo financeiro completo e integrações com *gateways* de pagamento (Stone, Cielo e Mercado Pago). Atualmente, os planos são somente pagos e variam de R$ 32 a R$ 293/mês, após 14 dias de teste gratuito (Gendo Sistemas, 2025).

Seus principais diferenciais são:

- **Caixa do profissional:** Módulo pensado para *coworking*, possibilitando débito automático de aluguel de estação e visualização dos ganhos de cada profissional;
- **Aplicativo Gendo Pro (iOS e Android):** É permitido ao profissional ver a agenda, acompanhar comissões, pedir saques e registrar fotos de antes e depois dos serviços;
- **Relatórios instantâneos:** São exibidos *ticket* médio, previsão de faturamento e dados de cancelamentos, com opção de exportar para *Excel*.

Já os maiores pontos de melhoria identificados são:

- **Dependência de gateways externos:** A integração com gateways de pagamento externos para realizar o *split*[3] adiciona custos transacionais e complexidade à operação. Para endereçar essa questão, o *BS Beauty* irá abstrair a camada de transação financeira. O sistema será responsável apenas pelo cálculo detalhado dos valores devidos e pelo registro do método de pagamento, enquanto a liquidação financeira ocorrerá de forma presencial. Essa abordagem estratégica elimina os custos com taxas de gateway e simplifica o fluxo de caixa, focando o sistema na gestão e conciliação dos recebimentos;
- **Acesso restrito a relatórios fiscais:** Funcionalidades de análise fiscal mais aprofundadas estão disponíveis apenas no plano *Premium*. O *BS Beauty*, por sua vez, disponibilizará um painel de relatórios financeiros e fiscais detalhados como parte de sua oferta principal, permitindo que a gestão tenha acesso a insights estratégicos sem a necessidade de um upgrade.

1.4.3 Avec

[3] *Split* é a divisão automática do pagamento entre salão e profissional que, se feita por um *gateway* externo, gera uma taxa extra.

Figura 6 – Logo plataforma Avec

Fonte: (Avec Company, 2025)

Atualmente, a Avec é a principal concorrente do nosso projeto, pois a entidade parceira que motivou este trabalho utiliza essa plataforma para gerenciar seu salão de beleza em modelo *coworking*. Por esse motivo, ela foi adotada como referência: buscamos manter as funcionalidades que já funcionam bem na Avec e, ao mesmo tempo, acrescentar ou aprimorar recursos que ainda fazem falta para a nossa parceira.

Lançada em 2014 e sediada em São Paulo-SP, a Avec se apresenta como solução"360º" para salões, barbearias, esmaltarias, spas e estúdios de tatuagem. A plataforma integra software de gestão, um sistema próprio de pagamentos (*Avec Pay*) e um *marketplace B2C* que encaminha novos clientes aos estabelecimentos. Segundo a empresa, mais de 40 mil negócios utilizam o serviço no Brasil. Também desenvolvida no modelo SaaS, a ferramenta oferece agenda *on-line* multiprofissional com confirmações via *WhatsApp* ou *Short Message Service* – Serviço de Mensagens Curtas (SMS), PDV completo com TEF, Pix e *split* interno de comissões, além de módulo financeiro integrado. Dispõe ainda de uma carteira digital empregada em pacotes pré-pagos, *gift-cards*, *cashback*, e possui dois aplicativos: o *Avec*, voltado ao cliente final, e o *Avec Pro*, destinado aos profissionais. Há um plano gratuito "*Avec Go*" que inclui funções básicas e cobra apenas a taxa transacional, enquanto os planos pagos variam de R$ 77 a R$249 por mês (Avec Company, 2025).

Com base no *feedback* da nossa entidade parceira, destacam-se três funcionalidades que a plataforma *Avec* executa bem:

- ***Split* instantâneo de comissões:** A estratégia elimina gastos ao dispensar *gateways* externos;
- ***Marketplace* B2C e aplicativo do cliente:** Plataformas que ampliam a visibilidade do salão e aumentam os agendamentos *on-line*;
- **Aplicativo *Avec Pro* (iOS/Android):** O profissional pode acompanhar facilmente agenda, comissões, saques, e registrar fotos de "antes e depois" dos serviços.

As principais brechas identificadas são:

- **Acesso restrito a módulos fiscais:** Recursos como emissão de Nota Fiscal Eletrônica (NF-e) e integração SAT estão disponíveis apenas nos planos superiores. A

solução proposta pelo *BS Beauty* busca democratizar o acesso a essas ferramentas, integrando relatórios fiscais essenciais na versão padrão do sistema;

- **Dependência de hardware e tarifas próprias:** O uso pleno do sistema está atrelado ao ecossistema *Avec Pay*, limitando a escolha de equipamentos e taxas. O *BS Beauty* será desenvolvido com uma arquitetura mais flexível, não se relacionando com a liquidação de pagamentos e não atrelando o uso do sistema a um hardware específico, o que confere maior liberdade ao gestor;
- **Custos adicionais para comunicação:** Campanhas de *marketing* via SMS ou *WhatsApp* em massa geram cobranças extras. Em contrapartida, o *BS Beauty* inclui um módulo de comunicação que utiliza APIs de baixo custo, integrando o envio de notificações ao plano principal para que o salão possa se comunicar com seus clientes de forma mais acessível;
- **Instabilidade recorrente da plataforma:** Conforme relatado pela parceira, o sistema apresenta indisponibilidade periódica. Um dos pilares do desenvolvimento do *BS Beauty* é a garantia de alta disponibilidade, utilizando uma infraestrutura em nuvem moderna e robusta para assegurar a estabilidade e a confiabilidade da plataforma.

1.4.4 Quadro comparativo

Quadro 1 – Comparação entre as plataformas concorrentes e a aplicação proposta

Recurso	Trinks	Ger
Gestão e Operações Essenciais		
Aplicação *web*	✓	✓
Agendamento 100% *on-line*	✓	✓
Plataformas do Cliente e do Profissional	✓	✓
Controle de Conflitos de Agenda	✓	✓
Diferenciais Estratégicos para Coworking		
Modelo de Negócio Focado em Coworking	—	✓
Gestão Avançada de Aluguel (hora/dia)	—	✓
Interface Intuitiva e Focada	—	–
Alta Disponibilidade e Estabilidade	✓	✓
Financeiro e Administrativo		
Cálculo de *Split* de Comissão	✓	✓
Processamento de Pagamento On-line	✓	✓
Gestão de Pagamentos Presenciais	—	–
Zero Taxas de Gateway	—	–
Independência de Hardware de Pagamento	✓	✓
Relatórios Financeiros em Tempo Real	✓	✓
Relatórios Fiscais Essenciais Inclusos	—	–
Marketing e Modelo de Negócio		
Marketplace B2C	✓	–
Marketing Integrado (Notificações)	✓	✓
SSO Google	✓	✓
Plano Gratuito Disponível	✓	–
Plano Base com Todas as Funcionalidades Operacionais	—	–

Fonte: Produzido pelos autores

2 REVISÃO DA LITERATURA

Devido ao crescimento contínuo dos *coworkings* de beleza, gerenciar informações, manter processos ágeis e oferecer atendimento de qualidade tornaram-se atitudes essenciais para se destacar em um mercado cada vez mais competitivo. Nesse contexto, o uso de sistemas capazes de administrar os dados gerados é crucial para o sucesso dos empreendedores. Considerando o rápido crescimento de tecnologias da informação que registram dados com confiabilidade, tais sistemas permitem otimizar o fluxo de trabalho interno e automatizar processos manuais (**??**).

2.1 Histórico de sistemas gerenciadores de agendamento

O ato de agendar serviços é uma prática antiga, e o gerenciamento desses agendamentos sempre foi um processo trabalhoso e passível de erros. No entanto, à medida que a tecnologia avança, surgem também ferramentas que facilitam esse processo, diminuindo a chance de falhas e perda de informações.

Nos primórdios, o agendamento de qualquer serviço era feito apenas presencialmente, dada a falta de tecnologias de comunicação à distância. Consequentemente, o controle financeiro e de clientes era realizado manualmente, o que gerava a necessidade de contratar outras pessoas para auxiliar neste processo, causando mais gastos. Com o advento do telefone, o acesso dos clientes tornou-se mais fácil (SEBRAE, 2023b). Porém, o trabalho de gestão ainda permanecia manual, exceto nos casos em que se adquiriam soluções de gestão em *Digital Versatile Disc* – Disco Digital Versátil (DVD)s, que eram pouco personalizadas para o negócio específico e não integradas aos agendamentos. Além disso, mantinha-se a necessidade de alguém disponível para atender às chamadas ou para controlar a correlação das agendas com as informações do DVD (**??**).

Posteriormente, com o surgimento da internet e das redes sociais, a maioria dos empresários prestadores de serviços aproveitou a oportunidade para concentrar seus agendamentos em mensagens de texto. Essa abordagem eliminava a necessidade de alguém estar sempre disponível para responder e confirmar, além de permitir a comunicação paralela com clientes (**??**). Ademais, a gestão já podia ser mais integrada a calendários virtuais (como o *Google Calendar*), aos do próprio *smartphone*, ou mesmo a planilhas digitais. Contudo, o processo de agendamento ainda dependia de uma ferramenta de comunicação que exigia intervenção humana: uma pessoa precisava estar envolvida na conversa para anotar o serviço agendado em outra ferramenta e controlar as finanças do negócio – tarefas ainda manuais ou realizadas com um sistema à parte, o que poderia

levar à perda de informações. Esse excesso de ferramentas e a comunicação fragmentada consumiam tempo e podiam resultar na desistência de clientes (**??**).

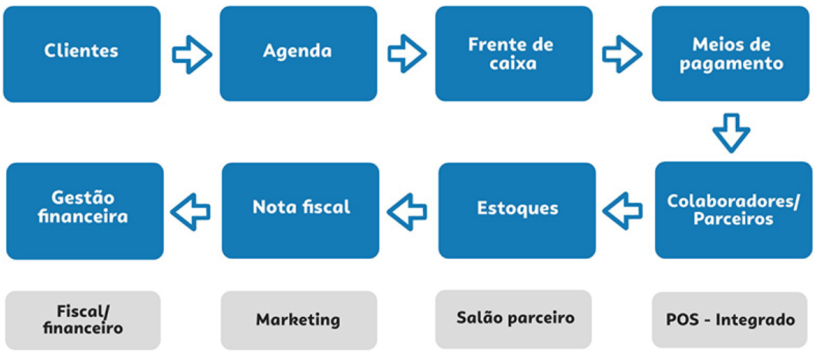
Diante desse cenário, surgiu a necessidade de um sistema de agendamento automático, que dispensasse a comunicação direta e já integrasse o processo de gestão de clientes e finanças na mesma plataforma. Portanto, surgiram os sistemas de gestão de agendamento *on-line*, com a promessa de reduzir gastos, minimizar processos manuais e aumentar a produtividade. Além disso, a possibilidade de fazer reservas a qualquer hora ajuda a atrair mais clientes e a mantê-los (Reservio, 2024).

Por fim, os sistemas de gestão e agendamento de serviços evoluíram rapidamente, tornando-se cada vez mais personalizados para diferentes setores ou até mesmo para empresas distintas dentro do mesmo setor. Atualmente, é possível contratar facilmente uma instância de aplicação específica para salões de beleza, adaptada ao próprio negócio, dada a vasta quantidade de soluções já existentes no mercado.

2.2 Atualidade de sistemas gerenciadores de agendamento

Atualmente, soluções digitais para gerenciamento e agendamento de serviços já são personalizadas para o setor da beleza, e implementadas por meio de assinaturas. Algumas focam até mesmo na gestão de *coworking*. Consequentemente, consolidaram-se funcionalidades padrão que são fundamentais para manter a competitividade no mercado. A Figura 7 ilustra os principais módulos embutidos em softwares e aplicativos disponíveis para espaços de beleza:

Figura 7 – módulos básicos em sistemas para salão de beleza



Fonte: (SEBRAE, 2023c)

Pensando especificamente em *coworkings* de beleza, surgem necessidades particulares:

- Evitar sobreposições de horários em cadeiras, cabines ou equipamentos;

- Gerenciar a divisão de recursos (produtos ou aparelhos);
- Calcular automaticamente alugueis de estação ou comissões, com *split* de pagamento entre o espaço e o profissional;
- Controlar acessos e permissões de cada perfil (administrador, profissional, cliente).

Embora algumas plataformas já ofereçam módulos de *coworking*, muitas ainda exigem configurações e ajustes manuais para o cálculo de comissões e a alocação de recursos, o que gera retrabalho e eleva o risco de erros. Para enfrentar esses desafios, destacam-se, hoje, as seguintes tendências tecnológicas:

- Aplicativos dedicados ao relacionamento independente entre profissionais e clientes para gestão de agenda;
- Pagamentos *on-line* em tempo real;
- *Dashboards* analíticos com indicadores de ocupação, performance individual e projeção de demanda.

No entanto, escolher uma solução digital para um *coworking* de beleza vai além de automatizar processos e melhorar a gestão de serviços. Dada a rapidez do avanço tecnológico e a influência das informações na internet, o *marketing* integrado e uma interface de usuário intuitiva passam a ser diferenciais decisivos. Um design previsível melhora a experiência, fideliza clientes e reduz o tempo de treinamento dos profissionais.

3 GESTÃO DO PROJETO

Neste capítulo de Gestão do Projeto são abordados tópicos como a organização da equipe (definindo as responsabilidades, papéis e atividades de cada integrante do grupo), a metodologia adotada para a gestão e desenvolvimento do projeto, bem como o repositório da aplicação.

3.1 Organização da Equipe

A equipe do presente projeto é composta por seis docentes do curso de graduação Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Campus São Paulo (IFSP-SPO), a saber:

- Alyson César Fumagalli dos Santos Júnior
- Bruno de Almeida Fischer
- Eliel da Silva
- Giovanna Camille Silva Carvalho
- Henrique Santiago Pires
- Henrriky Jhonny de Oliveira Bastos

O grupo de trabalho foi formado logo no início da primeira disciplina de Projeto Integrado de Extensão (PIE) por estudantes que já haviam realizado diversos outros trabalhos em conjunto e, portanto, estavam acostumados a trabalhar em equipe.

Visando uma transmissão de informações clara e centralizada, utilizou-se o *Discord* (DISCORD, 2025) como ferramenta de comunicação para realizar reuniões assíncronas; e o *Taiga* (**??**) para organizar documentos, atribuir tarefas e monitorar o andamento do projeto (juntamente com o *ProjectLibre* (PROJECTLIBRE, 2025)).

3.1.1 Responsabilidades / Papéis / Atividades

Para cada integrante da equipe foi definido um papel contendo responsabilidades e atividades definidas com base em suas respectivas proficiências em diferentes áreas, a fim de distribuir as tarefas do projeto de maneira eficiente.

O Quadro 2 descreve a distribuição dos membros do grupo com base em seus respectivos papéis de uma forma mais geral.

Quadro 2 – Membros e seus respectivos papéis

Membro	**Gestor**	***Tech Lead***	**Desenvolvedor *Full Stack***	**Analista de Doc**
Alyson	✓		✓	
Bruno			✓	
Eliel			✓	
Giovanna				✓
Henrique			✓	
Henrriky		✓	✓	

Fonte: Produzido pelos autores

Assim, constata-se que a equipe conta com 1 (um) gestor, responsável pelo gerenciamento de todo o projeto; 1 (um) *tech lead*, encarregado de guiar a equipe de desenvolvimento; 5 (cinco) desenvolvedores *full stack* (com 2 deles desempenhando outros papéis paralelos) incumbidos por desenvolver o sistema em todas as suas etapas e 1 (uma) analista de documentação para supervisionar as documentações do projeto.

O Quadro 3 apresenta as atividades desempenhadas pelos membros da equipe nas diversas áreas que contemplaram o desenvolvimento do projeto.

Quadro 3 – Membros e suas atividades

Membro	***Taiga***	***Front-End***	***Back-End***	**Banco de Dados**	**Documer**
Alyson	✓		✓	✓	✓
Bruno	✓		✓	✓	✓
Eliel	✓		✓	✓	✓
Giovanna	✓			✓	✓
Henrique	✓		✓	✓	✓
Henrriky	✓		✓	✓	✓

Fonte: Produzido pelos autores

3.2 Metodologias de Gestão e Desenvolvimento

Para o projeto, foi adotada a metodologia Scrum (AWS, 2024) voltada tanto para a gestão quanto para o desenvolvimento do sistema.

3.2.1 Scrum

O *framework* Scrum foi escolhido por ter sido bastante estudado em disciplinas anteriores e também devido à equipe já ter uma certa familiaridade em trabalhar com ele.

Além disso, para possibilitar a aplicação da metodologia, foi utilizado o Taiga (**??**), uma ferramenta *open-source* voltada justamente para gerenciamento de projetos.

Com o Taiga, foi possível aplicar os elementos do Scrum, como a definição de papéis dos integrantes do grupo, a criação e refinamento do *product backlog* e o estabelecimento dos *sprints* a serem trabalhados.

O Taiga também conta com elementos do *Kanban* (EQUIPE TOTVS, 2023) que, nesse projeto, foram incorporados à metodologia Scrum. Esses recursos incluem:

- Criação de quadros separados para cada sprint voltados ao monitoramento das tarefas definidas;
- Estabelecimento de colunas dentro dos quadros para indicar os estágios das tarefas ("a fazer", "em andamento", "concluído" ou "em revisão");
- Representação de tarefas na forma de cartões, contendo informações como o responsável pela tarefa, seu prazo de conclusão e descrição, que são transitados entre as colunas dos quadros;
- Aplicação de *swimlanes* nos quadros para agrupar tarefas relacionadas de acordo com as *user stories*.

Dessa forma, a integração entre *Kanban* e *Scrum* foi benéfica para a equipe, haja vista que permitiu uma melhor visualização do fluxo de trabalho e andamento do *sprint* trabalhado, além de possibilitar a identificação de dependências entre as tarefas.

Tendo em mente as responsabilidades provenientes do Scrum, o Quadro 4 descreve como os membros do grupo foram distribuídos seguindo os papéis da metodologia.

Quadro 4 – Papéis dos integrantes com base no Scrum

Membro	***Product Owner***	***Scrum Master***	**Equipe de Desenvolvimento**
Alyson		✓	✓
Bruno			✓
Eliel			✓
Giovanna			✓
Henrique			✓
Henrriky	✓		✓

Fonte: Produzido pelos autores

3.2.1.1 *Sprints*

Para o desenvolvimento do projeto, foram determinados *sprints* semanais que se iniciam na quarta-feira e terminam na terça-feira da outra semana. Com esse período de tempo, garante-se que a equipe realize entregas incrementais continuamente.

Dessa forma, os *sprints* e suas tarefas específicas foram definidos ao longo do desenvolvimento com base nos itens do *product backlog*, que por sua vez foram estabelecidos a partir das histórias de usuário levantadas junto à entidade parceira do projeto.

Ademais, as reuniões características dos *sprints* da metodologia Scrum foram marcadas para ocorrer presencialmente nos dias referentes à disciplina de PIE, permitindo os membros se organizarem para definir quais itens seriam trabalhados no próximo *sprint*, bem como os prazos, tarefas, prioridades e responsáveis.

3.3 Repositório da Aplicação

Nessa seção do capítulo, apresenta-se o repositório da aplicação, que contém todos os arquivos relevantes ao projeto (como código-fonte, documentos e diagramas).

3.3.1 Definição do repositório da aplicação

Baseado na familiaridade dos integrantes em utilizar o sistema de controle de versão Git (GIT, 2025), escolheu-se o GitHub (GITHUB, 2025c) para hospedar o repositório remoto da aplicação e tornar o desenvolvimento mais colaborativo.

Visando garantir um repositório organizado e eficiente, estabeleceu-se uma estrutura de diretórios separando código-fonte da documentação do projeto e adotou-se o modelo de fluxo de trabalho *Git Flow* (OBJECTIVE, 2023) para organizar o versionamento de ramificações. Além disso, foi utilizado *Conventional Commits* (CONVETIONAL COMMITS, 2025) para padronizar as mensagens de *commit*, conferindo ainda mais organização.

3.3.1.1 Link do repositório e especificações para acesso

O *QR Code* da Figura 8 contém o link que leva diretamente ao repositório remoto do projeto hospedado no *GitHub*. É possível tanto escanear quanto clicar no código abaixo para acessar o repositório.

Figura 8 – *QR Code* do repositório da aplicação

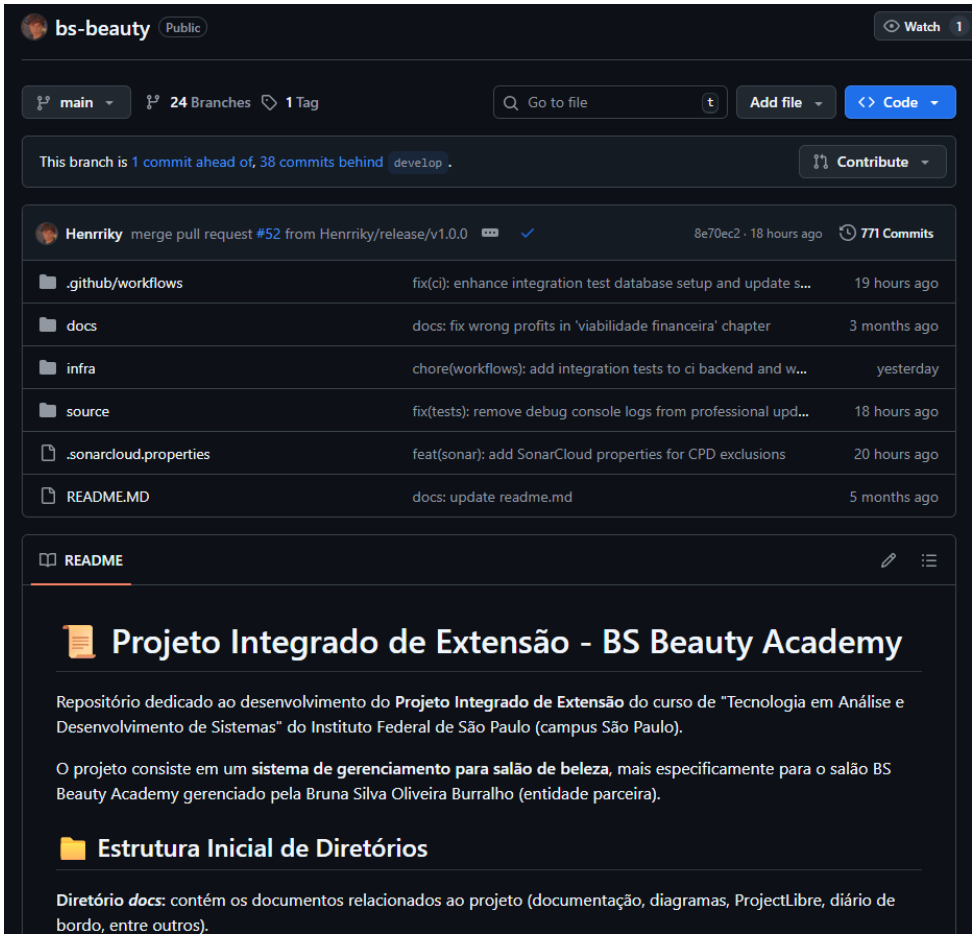
Fonte: Produzido pelos autores

Acessado o repositório, será aberta uma guia no navegador contendo uma página semelhante à Figura 9. Todos os arquivos do projeto podem ser visualizados diretamente

pelo navegador. Como o repositório é público, qualquer pessoa pode acessá-lo e cloná-lo localmente usando *HyperText Transfer Protocol Secure* – Protocolo de Transferência de Hipertexto Seguro (HTTPS). Para tanto, é necessário seguir as seguintes etapas:

1. Instalar o Git na máquina (caso não esteja instalado).
2. Escolher o diretório no qual o repositório será clonado.
3. Abrir o terminal e alterar o diretório atual para o diretório escolhido.
4. Usar o comando "`git clone https://github.com/Henrriky/bs-beauty`".
5. Usar o comando "`cd bs-beauty`" para acessar o repositório clonado.

Figura 9 – Página inicial do repositório



Fonte: Produzido pelos autores

Assim, o repositório estará devidamente clonado na máquina. Para execução local, utilize a *branch* principal ***`main`*** ou a ***`develop`*** e consulte o link do *QR Code* da Figura 10, que contém um tutorial para instalação do sistema.

Figura 10 – *QR Code* do tutorial de instalação

Fonte: Produzido pelos autores

Para utilizar outros métodos de clonagem (como *Secure Shell* (SSH), *GitHub Command Line Interface* – Interface de Linha de Comando (CLI) ou baixar o arquivo `.zip` do projeto), consulte a documentação oficial do *GitHub* (GITHUB, 2025a) referente às formas de clonar um repositório.

3.4 Desafios e Soluções

Durante a execução do projeto, a equipe enfrentou como principal desafio a limitação de tempo para conciliar as atividades acadêmicas e o desenvolvimento das funcionalidades planejadas. Essa restrição impactou diretamente o ritmo de entrega das tarefas e exigiu uma reorganização das prioridades no backlog.

Para superar esse obstáculo, foram adotadas estratégias de gestão ágil, como a redefinição de escopo em cada sprint e a divisão mais clara das responsabilidades entre os membros. Além disso, a comunicação constante por meio de reuniões breves e o uso de ferramentas colaborativas, como o Taiga e o GitHub, contribuíram para manter o alinhamento do time e garantir a entrega dos principais requisitos dentro do prazo estabelecido. Essa experiência reforçou a importância do planejamento contínuo e da adaptação frente a imprevistos no contexto de projetos reais.

4 DESENVOLVIMENTO DO PROJETO

Este capítulo descreve o processo de desenvolvimento do sistema proposto, abordando desde a definição de seu escopo até os testes realizados para garantir sua qualidade. Para isso, são apresentados os elementos que nortearam a construção da solução, como as histórias de usuário, as decisões arquiteturais, as tecnologias adotadas e as práticas de verificação aplicadas.

4.1 Escopo do Projeto

Esta seção apresenta o escopo do sistema desenvolvido, com o objetivo de delimitar suas funcionalidades, comportamentos esperados e restrições de operação. As definições aqui descritas foram elaboradas com base em reuniões realizadas com a gestora do salão de beleza, nas quais foram discutidas as necessidades do negócio e as principais demandas da operação cotidiana.

Com base nessas interações, foram estabelecidas as regras de negócio que orientam o funcionamento do sistema, os requisitos funcionais que especificam os serviços a serem oferecidos aos usuários e os requisitos não funcionais que tratam de aspectos como desempenho, segurança e usabilidade.

A definição clara desses elementos é essencial para garantir a coerência do sistema com as necessidades dos usuários e para orientar a equipe de desenvolvimento ao longo das etapas do projeto.

4.1.1 Regras de Negócio

As regras de negócio definem os comportamentos, restrições e condições que regem o funcionamento do sistema, conforme as particularidades da *BS Beauty Academy*, e representam diretrizes que devem ser respeitadas tanto no uso da aplicação quanto no desenvolvimento das funcionalidades.

A seguir, são listadas as principais regras de negócio identificadas:

	ID
	RN01
	RN02
	RN03

	ID	
	RN04	
	RN05	
	RN06	
	RN07	
	RN08	
	RN09	
	RN10	Preços e duração de serviç
	RN11	
	RN12	
	RN13	
	RN14	
	RN15	
	RN16	
	RN17	
	RN18	
	RN19	
	RN20	
	RN21	
	RN22	
	RN23	A qua
	RN24	Atualmente, o salão c
	RN25	
	RN26	

4.1.2 Requisitos Funcionais

Os requisitos funcionais representam as funcionalidades que o sistema deve implementar a fim de garantir o cumprimento das regras de negócio definidas durante as reuniões entre a equipe de desenvolvimento e a gestora do estabelecimento. Esses requisitos foram organizados de forma a manter uma correspondência clara com as regras de negócio, permitindo maior rastreabilidade e coerência entre as decisões do domínio e a implementação técnica do sistema.

	ID
RN01	
	RF01-01
	RF01-02

	ID
RN02	
	RF02-01
	RF02-02
	RF02-03
	RF02-04
	RF02-05
RN03	
	RF03-01
	RF03-02
	RF03-03
	RF03-04
RN04	
	RF04-01
	RF04-02
	RF04-03
	RF04-04
RN05	
	RF05-01
	RF05-02
	RF05-03
	RF05-04
RN06	
	RF06-01
	RF06-02
	RF06-03
	RF06-04
	RF06-05
RN07	
	RF07-01
	RF07-02
	RF07-03
	RF07-04
	RF07-05
RN08	
	RF08-01
	RF08-02
	RF08-03

	ID
	RF08-04
RN09	
	RF09-01
	RF09-02
	RF09-03
	RF09-04
	RF09-05
RN10	
	RF10-01
	RF10-02
	RF10-03
	RF10-04
RN11	
	RF11-01
	RF11-02
	RF11-03
	RF11-04
	RF11-05
RN12	
	RF12-01
	RF12-02
	RF12-03
	RF12-04
	RF12-05
RN13	
	RF13-01
	RF13-02
	RF13-03
	RF13-04
	RF13-05
	RF13-06
RN14	
	RF14-01
	RF14-02
	RF14-03
	RF14-04
	RF14-05

	ID
RN15	
	RF15-01
	RF15-02
	RF15-03
	RF15-04
RN16	
	RF16-01
	RF16-02
	RF16-03
	RF16-04
RN17	
	RF17-01
	RF17-02
	RF17-03
	RF17-04
	RF17-05
RN18	
	RF18-01
	RF18-02
	RF18-03
	RF18-04
	RF18-05
RN19	
	RF19-01
	RF19-02
	RF19-03
	RF19-04
	RF19-05
RN20	
	RF20-01
	RF20-02
	RF20-03
	RF20-04
	RF20-05
RN21	
	RF21-01
	RF21-02

	ID
	RF21-03
	RF21-04
	RF21-05
RN22	
	RF22-01
	RF22-02
	RF22-03
	RF22-04
	RF22-05
	RF22-06
RN23	
	RF23-01
	RF23-02
	RF23-03
	RF23-04
	RF23-05
RN24	
	RF24-01
	RF24-02
	RF24-03
	RF24-04
	RF24-05
	RF24-06
RN25	
	RF25-01
	RF25-02
	RF25-03
	RF25-04
	RF25-05
	RF25-06
	RF25-07
	RF25-08
RN26	
	RF26-01
	RF26-02
	RF26-03
	RF26-04

	ID
	RF26-05
	RF26-06
	RF26-07
	RF26-08
	RF26-09
	RF26-10
	RF26-11

4.1.3 Requisitos Não Funcionais

A seguir, são apresentados os requisitos não funcionais, relativos à qualidade e à experiência do sistema, garantindo aspectos essenciais como usabilidade, segurança e design.

Tabela 3 – Requisitos Não Funciona

	ID	
	RNF01	
	RNF02	
	RNF03	
	RNF04	O s
	RNF05	
	RNF06	

4.1.4 Histórias de Usuário

Esta seção apresenta as histórias de usuário elaboradas com o objetivo de descrever os requisitos funcionais do sistema sob a perspectiva dos usuários finais. As histórias foram definidas com base nas necessidades identificadas, e visam representar, de forma simples e objetiva, as funcionalidades esperadas por cada tipo de usuário do sistema: *customer* (cliente), *professional* (profissional) e *manager* (gerente).

Na tabela a seguir, cada história é redigida utilizando a estrutura "Como [tipo de usuário], eu quero [ação] para [objetivo]", permitindo evidenciar o papel do usuário, a funcionalidade desejada e o benefício esperado.

	ID	**Título**
	HIST-1	Cancelamento de agendamento
	HIST-2	Não aplicação de penalidades

ID	Título
HIST-3	Agendamento de múltiplos serviços
HIST-4	Resumo do agendamento
HIST-5	Visualização de horários disponíveis
HIST-6	Pagamento presencial
HIST-7	Registro de pagamento
HIST-8	Visualização das formas de pagamento
HIST-9	Avaliação de serviços
HIST-10	Geração de estatísticas com base nas avaliações
HIST-11	Não sobreposição de agendamentos
HIST-12	Antecedência no agendamento
HIST-13	Cadastro e edição de profissionais
HIST-14	Personalização de perfil
HIST-15	Definição de turnos
HIST-16	Bloqueio de horários
HIST-17	Precificação de serviços
HIST-18	Duração de serviços
HIST-19	Comissão de serviços
HIST-20	Horário de funcionamento do salão
HIST-21	Aquisição de novos clientes
HIST-22	Relatórios das fontes de captação de clientes
HIST-23	Programa de indicação
HIST-24	Mensagem de aniversário
HIST-25	Desconto aniversário
HIST-26	Notificação de novo atendimento
HIST-27	Notificação de atendimento cancelado
HIST-28	Notificação de atendimento próximo
HIST-29	Notificações sobre agendamentos
HIST-30	Prevenção de agendamentos no passado
HIST-31	Relatórios de produtividade
HIST-32	Relatórios financeiros
HIST-33	Relatórios de ocupação
HIST-34	Histórico de atendimentos
HIST-35	Histórico de agendamentos
HIST-36	Agendamento por profissional
HIST-37	Agendamento por serviço
HIST-38	Lista de profissionais
HIST-39	Lista de serviços

	ID	Título
	HIST-40	Agrupamento de serviços
	HIST-41	Filtro de busca de serviços
	HIST-42	Informações sobre serviços
	HIST-43	Cadastro de categorias de serviços
	HIST-44	Criação de serviços
	HIST-45	Aprovação de novo serviço
	HIST-46	Monitoramento de funcionárias
	HIST-47	Visualização da agenda
	HIST-48	Cancelamento de atendimento
	HIST-49	Login simplificado
	HIST-50	Conclusão de atendimento
	HIST-51	Aceitação de atendimento
	HIST-52	Oferecimento de serviços

4.2 Arquitetura

Nessa seção, apresenta-se a arquitetura da aplicação, que define como os componentes do sistema interagem entre si e com o usuário.

4.2.1 Definições da arquitetura

O sistema foi estruturado com base no modelo cliente-servidor, no qual o *front-end* e o *back-end* operam como aplicações independentes que se comunicam por meio de requisições *HyperText Transfer Protocol* – Protocolo de Transferência de Hipertexto (HTTP), seguindo a arquitetura RESTful (**??**). Essa característica adota um padrão baseado no *Model-View-Controller* – Modelo-Visão-Controlador (MVC) (**??**) de forma adaptada, aproximando-se de uma arquitetura em camadas. Essa escolha organiza o código de maneira modular, o que facilita a manutenção, testabilidade e escalabilidade do sistema.

Historicamente, o padrão MVC foi desenvolvido em um contexto em que a lógica de negócio e a interface de visualização eram implementados na mesma base de código, como em aplicações *desktop* ou *server-side* com PHP. Nesse modelo tradicional, o *Controller* recebia a requisição do usuário, interagia com o *Model* para processar os dados e, em seguida, atualizava a *View*, tudo no mesmo ambiente de execução.

Com a evolução das tecnologias web, houve um crescimento significativo de desacoplamento da camada de visualização, que hoje é frequentemente implementada como uma *Single Page Application* (SPA) utilizando frameworks modernos como *React*, *Angular* ou *Vue.js*. O que antes era considerado *Model* e *Controller* passou a ser visto como o *back-end*,

enquanto a *View* tornou-se o *front-end*, que executa diretamente no navegador do usuário.

Por esse motivo, considera-se que o sistema desenvolvido adota uma versão adaptada do padrão MVC, onde o *Controller* e o *Model* residem no *back-end*, enquanto a *View* é inteiramente gerenciada pelo *front-end*. Essa abordagem promove uma clara separação de responsabilidades:

- ***View:*** Implementada no *front-end* com a biblioteca *React*, é responsável por toda a interface do usuário, incluindo a apresentação dos dados e a captura das interações do usuário. Ela consome os dados fornecidos pela *Application Programming Interface* – Interface de Programação de Aplicações (API) do *back-end* para renderizar as telas de forma dinâmica.
- ***Controller:*** Responsável por receber e tratar as requisições provenientes do cliente (*front-end*), encaminhando-as para a camada de serviço correspondente. Esta camada lida diretamente com aspectos de infraestrutura externa, como servidores de borda ou *gateways* de entrada da aplicação, servindo como ponto inicial de processamento das solicitações.
- ***Service:*** Centraliza a lógica de negócio da aplicação. É responsável por processar as regras do domínio e pode tanto consumir outras funções de serviço quanto interagir com a camada de persistência.
- ***Repository:*** Trata das operações relacionadas à persistência de dados. Atua como uma interface entre os serviços e os mecanismos de armazenamento, como bancos de dados relacionais ou caches, promovendo o desacoplamento da lógica de negócio em relação à camada de dados.

A adaptação do MVC para uma arquitetura em camadas proporciona uma série de vantagens importantes. Primeiramente, há um claro desacoplamento entre a interface do usuário e a lógica de negócio, o que permite que diferentes equipes possam realizar o desenvolvimento do *front-end* e do *back-end* de forma independente, utilizando tecnologias e ferramentas específicas para cada camada. Além disso, cada camada pode ser mantida e evoluída de forma independente, o que facilita a manutenção do sistema ao longo do tempo. Essa separação melhora significativamente a testabilidade, permitindo a criação de testes isolados para cada camada, o que contribui para a qualidade do software. Por fim, o modelo favorece a escalabilidade e o reuso de componentes, uma vez que cada camada pode ser dimensionada conforme a demanda e reutilizada em diferentes contextos, garantindo maior modularidade e flexibilidade na evolução do sistema.

4.2.2 Diagrama da arquitetura

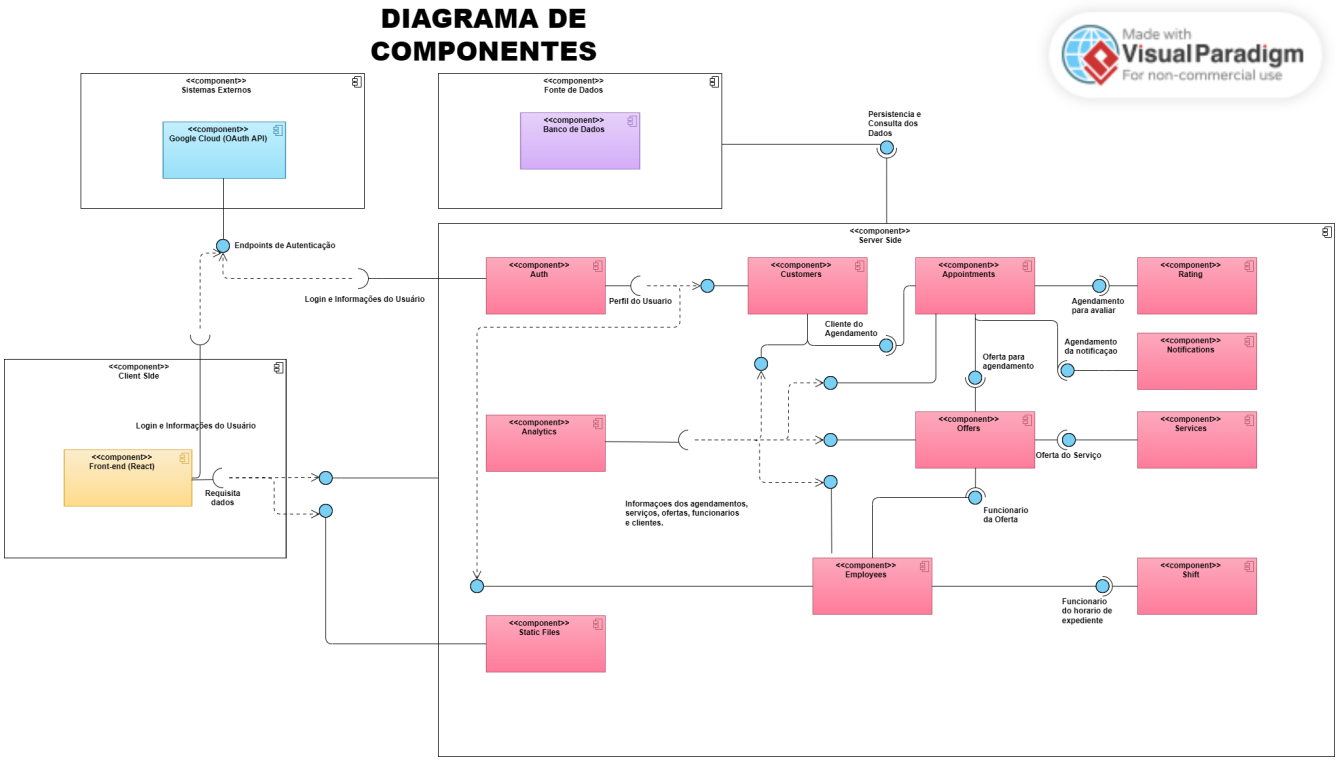
Esta subseção apresenta dois diagramas que representam a arquitetura do sistema desenvolvido: o diagrama de componentes e o diagrama de implantação. Esses diagramas auxiliam na visualização do relacionamento entre os componentes do sistema, bem como a sua distribuição nos ambientes computacionais.

4.2.2.1 Diagrama de componentes

O diagrama de componentes representa a estrutura modular dos principais módulos do sistema, evidenciando as dependências e as formas de comunicação entre eles, como comentado em (**??**). Ele demonstra como os componentes interagem por meio de interfaces — que definem contratos de uso — e implementações — que oferecem as funcionalidades esperadas.

Esse tipo de diagrama é útil para visualizar a estrutura modular da aplicação, facilitando o entendimento da separação de responsabilidades e da reutilização de código, além de apoiar decisões relacionadas à manutenção e evolução do sistema.

Figura 11 – Diagrama de componentes da aplicação



Fonte: Produzido pelos autores

No diagrama de componentes proposto, a aplicação é composta por diversos módulos que representam funcionalidades distintas e organizadas de forma modular. Esses

componentes são responsáveis por encapsular regras de negócio e oferecer interfaces bem definidas para comunicação entre si. A seguir, são descritos os principais módulos do sistema:

- ***Auth***: Componente responsável pela autenticação de usuários e integração com serviços externos, como o *Google OAuth* (Google Developers, 2025). Garante um processo facilitado de login para os usuários e autenticação no acesso das funcionalidades protegidas da aplicação.
- ***Analytics***: Responsável por gerar relatórios e fornecer estatísticas baseadas nas informações do sistema. Auxilia principalmente no acompanhamento de desempenho da plataforma por parte dos gerentes do salão.
- ***Appointments***: Gerencia os agendamentos realizados pelos clientes. Engloba tanto a criação, listagem e atualização dos agendamentos quanto a associação com os serviços ofertados.
- ***Customers***: Controla os dados relacionados aos clientes da plataforma. Permite o cadastro, consulta e edição de informações do perfil dos usuários.
- ***Professionals***: Administra os dados dos profissionais que prestam serviços na aplicação, incluindo informações cadastrais, disponibilidade e associação a serviços específicos.
- ***Notifications***: Responsável por enviar notificações aos usuários, como lembretes, confirmações de agendamento e atualizações importantes. Pode incluir o envio por *e-mail* ou outros canais.
- ***Offers***: Define a relação entre profissionais e os serviços que eles oferecem. Cada oferta especifica o tempo estimado e o valor cobrado por um profissional para realizar determinado serviço. Esse componente é fundamental para a composição de um agendamento, pois determina quais combinações de profissional e serviço estão disponíveis.
- ***Services***: Representa os serviços oferecidos pela empresa, armazenando informações descritivas como nome e descrição. Este módulo não define valores ou tempos de execução, pois esses dados são especificados nas ofertas individuais de cada profissional (por meio do módulo *Offers*).
- ***Shift***: Trata do controle de turnos de trabalho dos profissionais, possibilitando a definição de horários disponíveis para realização dos agendamentos.
- ***Rating***: Permite que os clientes avaliem os serviços e os profissionais após os atendimentos, promovendo um sistema de *feedback* contínuo.

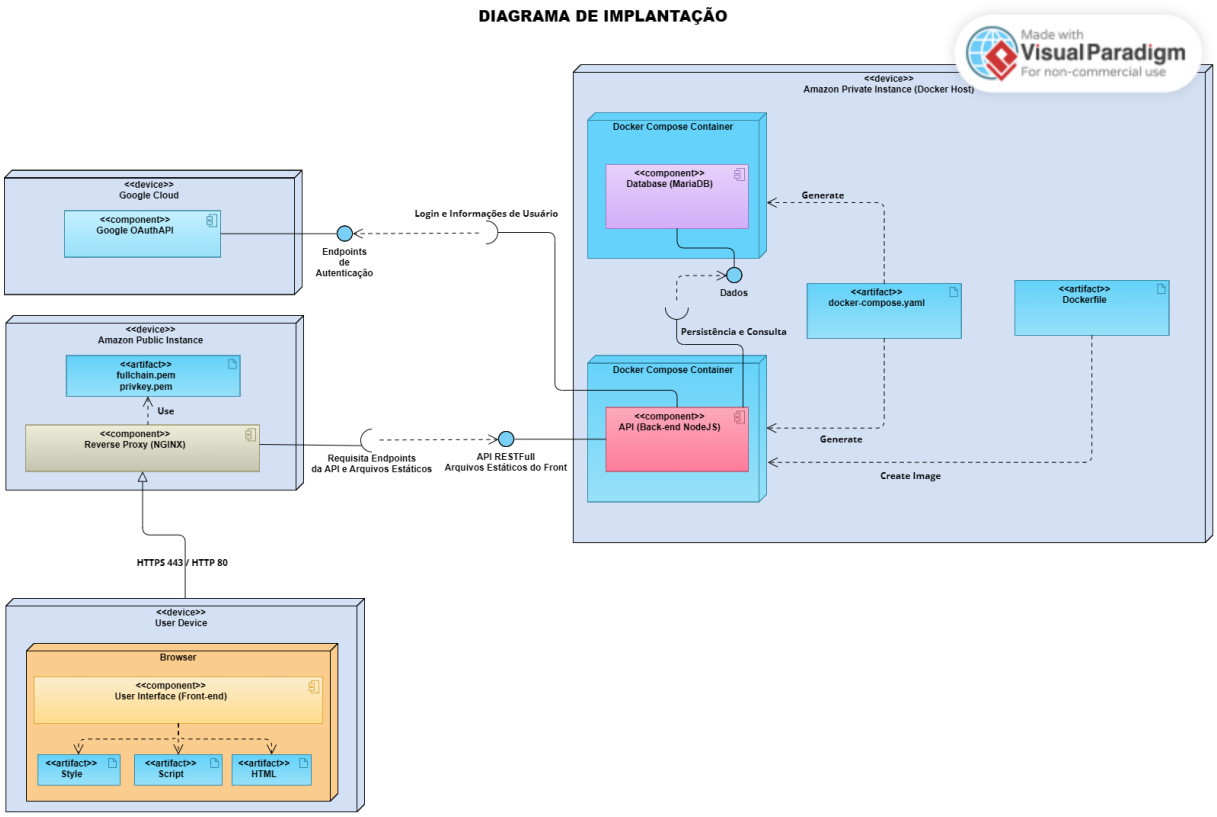
- ***Static Files (Front-end)***: Componente responsável por servir os arquivos estáticos da interface do usuário, gerados após o processo de *build* do projeto *front-end* (em *React*). Inclui arquivos *HyperText Markup Language* - Linguagem de Marcação de Hipertexto (HTML), *Cascading Style Sheets* (CSS) e *JavaScript* que são entregues ao navegador do usuário final via servidor *NGINX*.

4.2.2.2 Diagrama de implantação

O diagrama de implantação mostra como os componentes do sistema estão distribuídos em termos de infraestrutura (**??**), seja em servidores físicos ou ambientes virtuais. Ele ajuda a entender onde cada parte da aplicação está rodando, como os serviços se conectam entre si e quais recursos são necessários para que tudo funcione da forma adequada no ambiente de produção.

Esse tipo de representação é especialmente útil para quem for implantar ou manter o sistema, pois facilita a visualização de elementos como servidores, banco de dados, *gateways* de rede, e outras dependências da aplicação. Além disso, o diagrama contribui para o planejamento de permissões, acessos e políticas de segurança que precisam ser configuradas na infraestrutura.

Figura 12 – Diagrama de implantação da aplicação



Fonte: Produzido pelos autores

O diagrama acima é composto pelos seguintes componentes:

- ***User Device:*** No diagrama proposto, por se tratar de uma aplicação *web*, o dispositivo do usuário será responsável por executar a aplicação *client-side*, que interpreta através do navegador os arquivos CSS, *JavaScript* e HTML gerados no empacotamento ou *build* do projeto feito com a biblioteca *React*. Ademais, esse dispositivo é composto por alguns artefatos importantes que são obtidos pelo navegador por meio de uma requisição ao *Proxy Reverso*: CSS (*style)*, *JavaScript* (*script)* e HTML.

- ***Amazon Public Instance (Device):*** A *Amazon Public Instance* é uma instância *Elastic Compute Cloud* – Nuvem de Computação Elástica (EC2) que possui um *Internet Protocol* – Protocolo de Internet (IP) público, o que permite que ela seja acessada diretamente pelo usuário ou resolvida por *Domain Name System* – Sistema de Nomes de Domínio (DNS). Por esse motivo, nela são executados apenas os componentes que devem estar disponíveis publicamente ao cliente final:

  - *Componente NGINX*: Serviço que atua como *Proxy Reverso* (**??**) para a aplicação que está sendo executada em uma instância privada na arquitetura proposta. Para viabilizar conexões HTTPS, o *NGINX* utiliza certificados digitais emitidos gratuitamente pelo serviço *Let's Encrypt*, utilizando a ferramenta *Certbot* (FOUNDATION, 2025), que automatiza todo o processo de emissão e renovação dos certificados. Este serviço é instalado como um **artefato** adicional no ambiente da instância pública, sendo integrado ao próprio ciclo de configuração e inicialização do *NGINX*.

- ***Amazon Private Instance (Device):*** Por outro lado, a *Amazon Private Instance* é composta por uma instância EC2 com restrições de rede, o que significa que seu acesso é limitado à rede interna e não possui IP público. Nessa instância, componentes da arquitetura que não precisam estar disponíveis de forma pública são o caso de uso perfeito, uma vez que garante maior segurança e isolamento dos aspectos internos da aplicação. Em seu interior, ela é composta pelos seguintes componentes:

  - A *API (Back-end em NodeJS)*: Principal serviço da aplicação, sendo o responsável por prover os *"endpoints"* que fornecem os dados e arquivos estáticos para o *front-end* através de uma *API RestFull*, se comunicando com o banco de dados, um componente que é executado no mesmo dispositivo.

  - *Banco de Dados (Sistema Gerenciador de Banco de Dados (SGBD) MariaDB)*: Serviço de SGBD que provê os dados para o *back-end* da aplicação, o que possibilita a persistência e consulta de forma eficiente. Para garantir a persistência dos dados gerenciados pelo MariaDB, o contêiner utiliza volumes *Docker*

montados na instância EC2 privada. Isso assegura que os dados não sejam perdidos em reinicializações do contêiner.

Além dos serviços em execução, a instância privada também contém os seguintes artefatos essenciais para o empacotamento e execução dos serviços via contêineres *Docker* (**??**):

- *Dockerfile*: Esse artefato descreve as instruções necessárias para criar a imagem *Docker* da aplicação *back-end*, especificando o ambiente base (como a imagem do Node.js), os arquivos a serem copiados, dependências a serem instaladas e os comandos de inicialização da aplicação.
- *docker-compose.yaml*: Esse arquivo é utilizado como ferramenta de orquestração para os serviços *Docker* (**??**) da aplicação. Ele define a configuração dos contêineres da aplicação, como o contêiner da API e o do banco de dados, bem como as variáveis de ambiente, volumes, redes e dependências entre os serviços. É a partir deste artefato que os contêineres são gerados e executados de forma integrada.

- ***Google Cloud (Device):*** Localizada na nuvem pública da Google (*Google Cloud*), essa API é utilizada pelo *back-end* da aplicação para realizar a autenticação de usuários através do protocolo *OAuth* 2.0 (Google Developers, 2025). Esse processo ocorre quando o usuário opta por fazer *login* com sua conta Google. Nesse cenário, a aplicação redireciona o usuário para a tela de autenticação da Google, e após a confirmação, a API recebe um *token* de acesso que é utilizado para obter as informações do usuário autenticado. Esse fluxo garante uma autenticação segura, delegando a responsabilidade da validação de identidade à Google.

O fluxo de execução típico da aplicação baseado no diagrama de implantação acima segue os seguintes passos:

1. O usuário acessa a aplicação pelo navegador, requisitando os arquivos HTML/CSS/*JavaScript* (JS) ao servidor *NGINX*.
2. O *NGINX*, atuando como proxy reverso, redireciona essas requisições para o serviço de *back-end* na instância privada.
3. A API processa a requisição, acessa o banco de dados quando necessário e retorna os dados.
4. Em caso de autenticação via Google, a API redireciona o usuário para o serviço *Google OAuth* (Google Developers, 2025), que retorna um *token* de acesso após o *login*.

5. Esse *token* é utilizado pela API para obter os dados do usuário autenticado e estabelecer uma sessão.

Além da organização dos componentes do diagrama, a arquitetura também prioriza a segurança da comunicação e do acesso. O tráfego entre o navegador do usuário e a instância pública é realizado por meio do protocolo HTTPS, o que garante a confidencialidade e a integridade dos dados transmitidos. Para isso, foi utilizado o serviço gratuito de certificação digital *Let's Encrypt* em conjunto com a ferramenta *Certbot*, que automatiza a emissão, renovação e instalação dos certificados *Transport Layer Security* – Segurança da Camada de Transporte (TLS) no servidor *NGINX*.

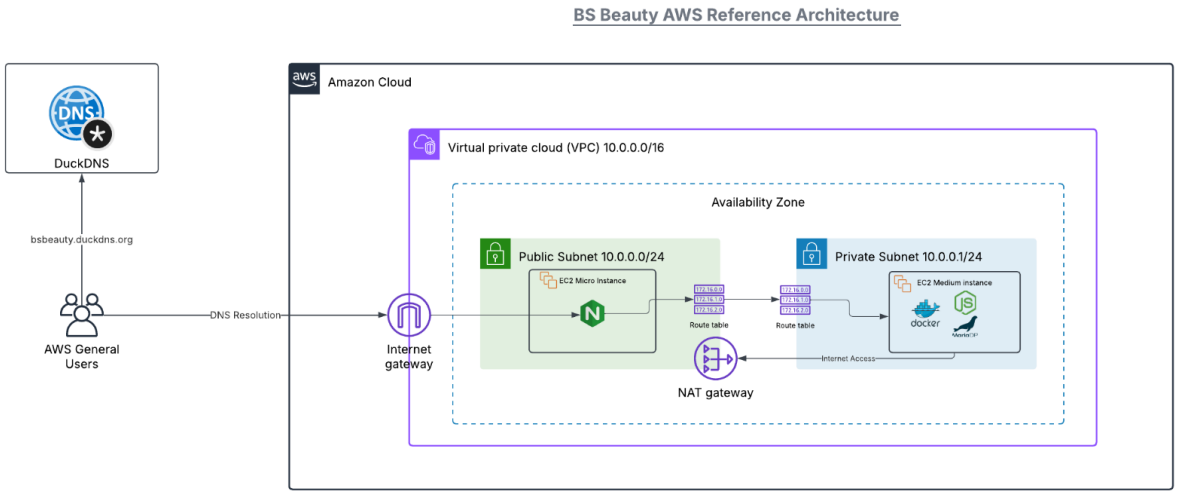
Internamente, a comunicação entre o *NGINX* e os serviços da instância privada ocorre seguindo regras específicas de segurança definidas na *Virtual Private Cloud* – Nuvem Privada Virtual (VPC), utilizando mecanismos como *Security Groups* e *Route Tables*. Isso reduz significativamente a superfície de ataque da aplicação e assegura uma camada adicional de proteção para os aspectos internos.

4.2.2.3 Diagrama de referência na AWS

Com o objetivo de fornecer uma visão mais aprofundada da infraestrutura da aplicação na nuvem, o diagrama apresenta a disposição dos principais componentes implantados na arquitetura da AWS.

Este diagrama ilustra elementos de infraestrutura fundamentais como sub-redes públicas e privadas, resolução de DNS, VPC, *Bastion Server*, *Network Address Translation* – Tradução de Endereços de Rede (NAT) *Gateway*, *Internet Gateway*, banco de dados, entre outros recursos. A representação facilita a compreensão técnica da topologia de rede e da distribuição dos serviços, evidenciando como a aplicação foi projetada para atender requisitos de segurança, escalabilidade e disponibilidade no ambiente da AWS. A seguir, descreve-se brevemente cada elemento presente no diagrama.

Figura 13 – Diagrama Geral da Arquitetura



Fonte: Produzido pelos autores

- **VPC (Virtual Private Cloud):** A aplicação opera dentro de uma VPC personalizada com o bloco *Classless Inter-Domain Routing* – Roteamento Interdomínio sem Classes (CIDR) `10.0.0.0/16`, que abriga duas sub-redes: uma pública e outra privada, seguindo o princípio de segmentação de rede recomendado pela própria AWS (Amazon Web Services, 2025).
- **Sub-rede pública (10.0.0.0/24):** Contém uma instância EC2 de pequeno porte (`t2.micro`) que executa o serviço *NGINX*. Esse servidor atua como *proxy* reverso, roteando as requisições provenientes da internet para os serviços internos hospedados em uma sub-rede privada.
- **Sub-rede privada (10.0.0.1/24):** Hospeda uma instância EC2 de médio porte (`t2.medium`), na qual são executados os contêineres da aplicação via *Docker Compose*, incluindo o serviço de *back-end* (Node.js) e o banco de dados relacional MariaDB.
- **NAT *Gateway*:** Permite que os recursos da sub-rede privada (como a instância EC2 que executa os contêineres) realizem atualizações e acessos à internet de forma segura, sem que sejam diretamente acessíveis externamente.
- **Internet *Gateway*:** Responsável por permitir o tráfego de entrada e saída entre a VPC e a internet pública. Está associado à sub-rede pública e NAT *Gateway*, permitindo que o *NGINX* receba requisições externas e o NAT receba um IP.
- ***DuckDNS*:** Utilizado como serviço de DNS dinâmico gratuito (**??**), permitindo que a aplicação seja acessada por um domínio estável (`bsbeauty.duckdns.org`), mesmo

que o endereço IP público da instância EC2 varie. A resolução de nome é feita de forma transparente para o usuário final, facilitando o acesso à aplicação.

- **Usuários externos (*AWS General Users*):** Representam os clientes que acessam a aplicação via navegador. O tráfego HTTP/HTTPS chega inicialmente ao serviço *NGINX* na sub-rede pública, que encaminha as requisições para a instância privada onde os serviços da aplicação estão efetivamente em execução.

A separação entre sub-rede pública e privada segue boas práticas de segurança e isolamento de ambiente, recomendadas tanto pela documentação oficial da AWS quanto por autores renomados da área de arquitetura de software em nuvem, como em (Amazon Web Services, 2025). Ao manter os serviços internos em uma sub-rede privada, reduz-se a superfície de ataque da aplicação e melhora a resistência contra acessos não autorizados.

Além disso, a utilização do *DuckDNS* simplifica a exposição da aplicação para o ambiente externo sem a necessidade de configurar manualmente um serviço de DNS ou pagar por domínios personalizados, o que se alinha aos objetivos de custo deste projeto.

4.3 Tecnologias e Ferramentas

Nesta seção, são apresentadas as principais tecnologias e ferramentas utilizadas no desenvolvimento do sistema. Elas foram escolhidas com base em critérios de desempenho, escalabilidade, facilidade de manutenção e ampla adoção pela comunidade de desenvolvedores. As ferramentas foram organizadas em quatro categorias: *Front-End*, *Back-End*, *Infraestrutura* e *Qualidade de Software e Testes*.

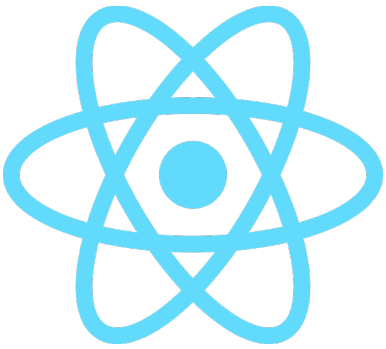
4.3.1 *Front-End*

O *Front-End* é responsável pela interface do usuário e pela interação com o sistema. As tecnologias selecionadas visam oferecer uma experiência fluida e responsiva, com código limpo e de fácil manutenção.

4.3.1.1 *React*

O *React* (**??**) é uma biblioteca *JavaScript* para construção de interfaces de usuário. Atualmente, se coloca como uma das ferramentas mais populares nesse aspecto (**??**). Sua utilização se foca na criação de componentes encapsulados, reutilizáveis e que gerenciam seus próprios estados.

Figura 14 – Logo do *React*



Fonte: (**??**)

4.3.1.2 *TailwindCSS*

Framework de CSS utilitário que permite a criação rápida de *layouts* e estilizações diretamente nas classes HTML. Parte da premissa do desenvolvedor não deixar o arquivo HTML para estilizar a página com CSS, embutindo as duas tecnologias em um único arquivo e também removendo código CSS inútil, diminuindo o tamanho do arquivo final enviado ao usuário final (**??**).

Figura 15 – Logo do *Tailwind*

Fonte: (**??**)

4.3.1.3 *Redux* e *RTK Query*

Biblioteca que facilita o gerenciamento de estados no *React* e outras bibliotecas de *interface* (**??**). Já o *RTK Query* oferece funcionalidades de cache e requisições assíncronas de forma otimizada (**??**). Ele facilita a criação de código para requisição de dados, evitando sua escrita manual, que torna-se muito repetitiva no desenvolvimento de aplicações.

Figura 16 – Logo do *Redux*

Fonte: (**??**)

4.3.2 *Back-End*

O *Back-End* compreende os componentes responsáveis pela lógica de negócio, persistência de dados e comunicação com o *Front-End*. Para isso, foram utilizadas ferramentas que oferecem alta performance e simplicidade no desenvolvimento.

4.3.2.1 *NodeJS*

Ambiente de execução *JavaScript* no lado do servidor, baseado no motor V8 do navegador *Chrome*. É ideal para a construção de aplicações escaláveis e que funcionem em tempo real (**??**). O fato de funcionar independentemente do navegador torna-o performático e atraente para os desenvolvedores.

Figura 17 – Logo do *Node*

Fonte: (**??**)

4.3.2.2 *Express*

Framework de roteamento minimalista para Node.js que torna a criação de APIs e rotas HTTP simples e eficiente. Ele possui uma enxuta gama de ferramentas e recursos,

que providenciam uma forma facilitada para criação de aplicações sem comprometer a já aclamada performance do Node (**??**), no entanto, suas funcionalidades ainda podem ser ampliadas pelos módulos de *middleware*.

Figura 18 – Logo do *Express*

Fonte: (**??**)

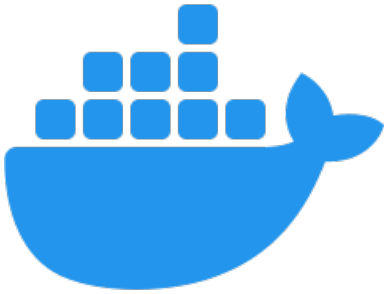
4.3.3 Infraestrutura

A infraestrutura do sistema foi projetada para garantir portabilidade, escalabilidade e facilidade de implantação. Sendo assim, foram empregadas tecnologias modernas de contêinerização, hospedagem em nuvem e gerenciamento de bancos de dados.

4.3.3.1 *Docker*

Plataforma aberta de contêineres que permite empacotar aplicações e suas dependências de forma isolada e reprodutível. Essa divisão entre infraestrutura e aplicação culmina na entrega mais veloz de *software*, e o encapsulamento de aplicações elimina os problemas que surgem por diferenças em relação a *hardware* ou sistema operacional. Um contêiner *Docker* funciona para qualquer pessoa da mesma forma (**??**).

Figura 19 – Logo do *Docker*



Fonte: (**??**)

4.3.3.2 *Amazon Web Services* (AWS)

É uma plataforma que provê computação em nuvem. Esses serviços são utilizados para hospedagem, armazenamento, orquestração de infraestrutura e muito mais (**??**).

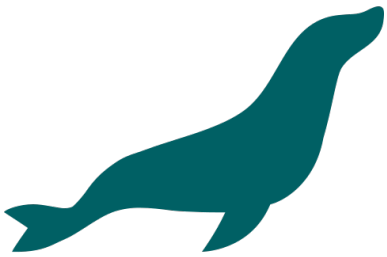
Figura 20 – Logo do *AWS*

Fonte: (**??**)

4.3.3.3 Banco de Dados MariaDB

Notável e popular Sistema de gerenciamento de banco de dados relacional *Open Source* usado para armazenamento persistente e estruturado de dados. Trata-se de um *fork* do *MySQL*, feito pelos seus desenvolvedores originais após a aquisição deste último pela *Oracle* (**??**).

Figura 21 – Logo do *MariaDB*



Fonte: (**??**)

4.3.4 Qualidade de software e testes

Para garantir a qualidade do código e a confiabilidade das funcionalidades desenvolvidas, foram aplicadas ferramentas de análise estática e testes automatizados.

4.3.4.1 *SonarQube*

Ferramenta para análise contínua da qualidade do código, identificando problemas como *bugs*, vulnerabilidades e *code smells* (**??**).

Figura 22 – Logo do *SonarQube*

Fonte: (**??**)

4.3.4.2 *Vitest*

Framework moderno de testes para aplicações *JavaScript* e *TypeScript*, com integração nativa ao ecossistema do Vite (VITEST, 2025).

Figura 23 – Logo do *Vitest*

Fonte: (**??**)

4.4 Testes e Manutenibilidade

Nesta seção, apresenta-se os mecanismos e ferramentas adotados para garantir a qualidade de software do projeto ao longo do desenvolvimento. Será abordado o plano de testes, assim como cada teste que está incluso. Além disso, assuntos como infraestrutura de testes e convenções de código (*coding convention*) serão detalhados, evidenciando práticas que promovem a manutenibilidade, padronização e confiabilidade do sistema ao longo de sua evolução.

4.4.1 Plano de Testes

Tendo em vista que a arquitetura do *back-end* é constituída por *controllers*, *services* e *repositories* usando o *framework Express*, é necessário garantir um ótimo funcionamento da comunicação entre essas camadas. Portanto, conforme os recursos da *RESTful API* são desenvolvidos (agendamentos, serviços de beleza, clientes, profissionais etc) com as respectivas camadas que foram comentadas, urge-se a demanda de serem testadas em paralelo, cobrindo os cenários possíveis de sucesso/falha.

Pretende-se atingir, no mínimo, 70% de cobertura nos testes unitários aplicados sobre as camadas de *controllers* e *services* da API.

As métricas de testes para o *back-end* são obtidas por meio de ferramentas integradas ao ambiente de testes, como o *framework* ***Vitest*** (VITEST, 2025) com suporte a geração de relatórios. Embora a cobertura de testes não garanta por si só a ausência de falhas, ela serve como um forte indicador de que a maior parte da lógica de negócio está sendo exercitada e validada durante a execução dos testes. Essa prática contribui diretamente para a robustez do sistema e facilita a detecção precoce de regressões ao longo do desenvolvimento.

Quanto aos testes do *front-end*, o foco principal do plano é realizar testes de usabilidade tanto em dispositivos de menor resolução (celulares e tablets) quanto dispositivos grandes (desktops e notebooks).

Ademais, os testes desenvolvidos nas duas divisões do sistema terão foco, inicialmente, nas funcionalidades essenciais para o funcionamento do sistema, tais como autenticação, criação e gerenciamento de agendamentos, relatórios, serviços e ofertas. Essa abordagem inicial permite validar as principais regras de negócio desde as primeiras entregas, contribuindo para uma base de código mais robusta e confiável à medida que novas funcionalidades forem sendo integradas.

Quadro 5 – Planejamento de Testes por Fase Funcional

Fase
Fase 1 – Preparação da Esteira CI/CD e Ambiente de Testes
Fase 2 – Funcionalidades Essenciais
Fase 3 – Monitoramento de qualidade e Medidas de comunicação
Fase 4 – Dados Operacionais
Fase 5 – Estabilização e Qualidade Final

Fonte: Produzido pelos autores

4.4.2 Análise Estática

A análise estática de código é realizada utilizando as ferramentas ***SonarQube*** (SONARQUBE, 2025) e ***ESLint*** (ESLINT, 2025), que permitem detectar problemas de qualidade, como vulnerabilidades, *bugs* e código duplicado, sem a necessidade de executar a aplicação. Essa etapa ajuda a manter o código limpo, seguro e sustentável ao longo do tempo.

4.4.3 Testes funcionais

Os testes funcionais baseiam-se nas especificações dos requisitos do sistema. O principal propósito, portanto, é validar se uma determinada funcionalidade solicitada foi desenvolvida com êxito, e está atendendo as expectativas esperadas dentro do sistema. Dessa forma, serão abordados nessa subseção os testes unitários e de integração.

4.4.3.1 Testes Unitários

Os arquivos de testes unitários são organizados em diretórios específicos com uma nomenclatura padronizada. Um exemplo de nome de arquivo seria `appointments.controller.spec.ts`, indicando que se trata de um teste unitário do módulo de agendamentos. Essa convenção visa facilitar a identificação, localização e manutenção dos testes ao longo do tempo.

4.4.3.2 Testes de Componente

Os testes de componente não serão desenvolvidos neste estágio do projeto, tendo em vista a complexidade envolvida na configuração de ambientes de teste para o *front-end* e o tempo reduzido disponível no cronograma de desenvolvimento. Além disso, a cobertura pretendida pelos testes unitários e de integração já contempla a maior parte das interações entre os módulos do sistema, tornando desnecessário, neste momento, o esforço adicional de implementação de testes específicos de componente. Esses testes poderão ser incorporados futuramente, conforme o sistema amadureça e novas versões do *front-end* sejam disponibilizadas.

4.4.3.3 Testes de Integração

Assim como os testes unitários, os testes integrados seguem uma convenção de nomenclatura para manter a organização do projeto. Um exemplo de nome de arquivo seria `services.integration.spec.ts` que sinaliza tratar-se de um arquivo de testes vinculado ao módulo de serviços. Essa padronização contribui para a clareza estrutural do projeto de testes.

4.4.3.4 Testes *end-to-end*

Os testes *end-to-end* não serão desenvolvidos no momento inicial do projeto devido ao alto custo de configuração e manutenção dessa categoria de testes, que exige a simulação completa do ambiente e da integração entre todas as camadas do sistema. Dada a fase atual do desenvolvimento, o foco está em garantir a estabilidade das funcionalidades essenciais por meio de testes unitários e de integração, que já permitem validar os principais fluxos do sistema com menor complexidade. A implementação de testes *end-to-end* está prevista para fases futuras, quando o produto atingir maior estabilidade e o ciclo de entregas estiver consolidado.

4.4.4 Testes não funcionais

4.4.4.1 Testes de performance

Testes de performance estão fora do escopo do projeto neste momento em razão da ausência de requisitos formais de desempenho e da necessidade de concentrar esforços nas funcionalidades básicas do sistema. A configuração de um ambiente adequado para esse tipo de teste demandaria recursos e tempo adicionais não compatíveis com o cronograma atual. Em futuras versões, quando houver maior volume de dados e tráfego de usuários, serão aplicadas medições específicas de latência, throughput e tempo de resposta para assegurar a escalabilidade do sistema.

4.4.4.2 Testes de carga

Os testes de carga não serão realizados na fase atual, pois o sistema ainda se encontra em estágio inicial e não há volume de usuários suficiente para justificar esse tipo de avaliação. Esses testes requerem simulações realistas de uso simultâneo e infraestrutura específica para coleta de métricas de desempenho, o que elevaria significativamente o custo de desenvolvimento. Sua execução será considerada em fases posteriores, após a consolidação da versão estável do sistema.

4.4.4.3 Testes de configuração

Os testes de configuração não serão desenvolvidos devido à maturidade ainda em evolução dos ambientes de implantação e integração contínua. Como o projeto encontra-se em estágio inicial, com constantes ajustes nas variáveis de ambiente e parâmetros de execução, optou-se por validar essas configurações manualmente durante o processo de integração e entrega contínua (*CI/CD*). Assim que os ambientes estiverem estabilizados, pretende-se automatizar parte dessas verificações para garantir maior consistência entre os diferentes estágios de desenvolvimento.

4.4.5 Testes automatizados

Os testes automatizados agrupam os dois tipos de testes que são usados no desenvolvimento do projeto: unitários e integrados, que são executados de forma automática por ferramentas de validação contínua.

A execução dos testes automatizados está ligada ao fluxo de integração contínua por meio da ferramenta ***GitHub Actions*** (GITHUB, 2025b). Essa integração visa garantir que o código entregue atenda a padrões mínimos de qualidade e estabilidade em todas as etapas do desenvolvimento. A cada *push* ou *pull request*, fluxos automatizados são executados para validar o código por meio de testes automatizados, análise estática, e verificação de cobertura. Esse processo assegura que apenas alterações estáveis e em conformidade com os padrões de qualidade sejam incorporadas à base de código principal, promovendo entregas seguras e contínuas ao longo do ciclo de desenvolvimento.

4.4.6 *Logs*

O sistema adota uma estratégia de registro de eventos por meio de mecanismos de *logging* que visam fornecer visibilidade sobre o comportamento da aplicação durante sua execução. Esses registros são essenciais para atividades de depuração, monitoramento e manutenção.

No *back-end* e *front-end*, os logs são implementados utilizando funcionalidades nativas do Node.js, como o uso de `console.log`, `console.error`, `console.warn` e `console.info`.

Os principais pontos de geração de logs incluem:

- A entrada e saída de requisições HTTP (rotas, métodos, status).
- Erros em operações internas, como falhas de banco de dados ou validações.

Além disso, é tido um cuidado com as informações exibidas nos *logs*. Em suma, é priorizado que as mensagens exibidas sejam curtas e objetivas com a cautela de não fornecer dados sensíveis.

4.4.7 *Code Convention*

Para garantir a legibilidade e padronização do código, são adotadas convenções definidas com base em boas práticas da comunidade *JavaScript/TypeScript*. Essas diretrizes ajudam a manter o código uniforme entre os diferentes desenvolvedores do time, reduzindo ambiguidades e facilitando o entendimento do sistema como um todo.

As principais práticas adotadas incluem:

- **Ferramentas de *Linting* e Formatação:**
  - Utilização do ***ESLint*** (ESLINT, 2025) para garantir padrões de estilo e detectar possíveis erros ou práticas inadequadas de codificação.
  - Uso do ***Prettier*** (PRETTIER, 2025) para formatação automática do código, assegurando que todos os arquivos mantenham a mesma estrutura visual (espaçamento, quebras de linha, indentação, etc).
- **Padrões de Nomenclatura:**
  - Uso de `camelCase` para variáveis e funções.
  - Uso de `PascalCase` para componentes e classes.
  - Uso de `UPPER_SNAKE_CASE` para constantes globais.
- **Organização do Código:**
  - Estrutura modular com separação clara entre camadas (*controllers*, *services*, *repositories*).
  - Agrupamento de arquivos por domínio funcional.
- **Boas Práticas:**
  - Escrita de código limpo e legível, evitando duplicações.
  - Utilização de comentários apenas quando necessário, priorizando nomes autoexplicativos.

- **Revisões e Padronização em Equipe:**
  - Adoção de *pull requests* com as devidas descrições das funcionalidades desenvolvidas.
  - Documentação e comunicação clara de decisões técnicas relevantes.

4.4.8 Cobertura de testes

A cobertura de testes indica a porcentagem do código-fonte exercitada pela suíte durante a execução. A cobertura é usada como **indicador de lacunas**, não como garantia de qualidade absoluta.

4.4.8.1 Meta institucional (back-end).

A equipe deverá atingir **70% de cobertura geral no back-end** considerando somente **testes unitários**. Essa meta é agregada no nível do serviço (projeto inteiro), e **não** é exigida por módulo.

4.4.8.2 Como a cobertura será medida.

- **Métrica oficial:** cobertura de **instruções/linhas** (statement/line).
- **Acompanhamento complementar:** cobertura de **ramos** (branch) será reportada, sem meta numérica. Serve para revelar decisões não exercitadas.
- **Escopo do cálculo:** entram apenas arquivos de código de produção do back-end.
- **Exclusões:** arquivos gerados, migrações, *scripts* utilitários e código de teste não entram no denominador.

4.4.8.3 Unitários x integrados.

- **Unitários:** base da meta de 70%. Devem cobrir regras de negócio, validações e tratamentos de erro.
- **Integrados:** não possuem percentual mínimo. Serão priorizados **cenários críticos** do domínio.

4.4.8.3.1 Boas práticas adotadas.

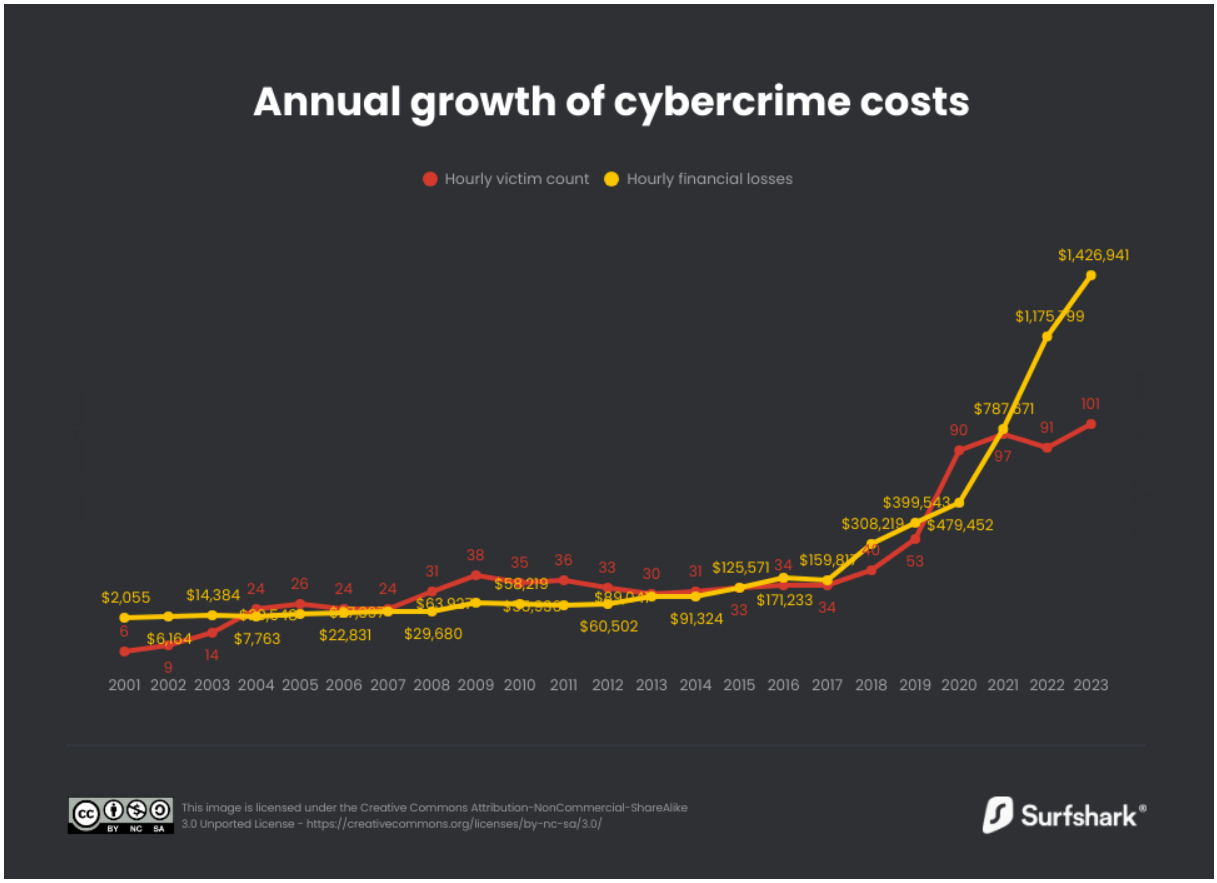
- Escrever testes que verifiquem **resultados e efeitos colaterais** (asserts úteis), não apenas “executar linhas”.
- Dar atenção a **ramos e exceções**: caminhos de erro, validações e *edge cases*.

- Para integrações (BD/filas), usar ***test doubles*** quando possível e poucos testes integrados focados em cenários críticos.

4.5 Segurança, Privacidade e Legislação

Atualmente, há uma crescente no número de usuários utilizando dispositivos conectados a *Internet*, o que, por um lado mostra que essa tecnologia tão importante está sendo difundida a todas as camadas sociais, mas por outro, gera preocupação quanto as implicações do uso inadvertido das redes. Nesse contexto, tornou-se comum pessoas mal intencionadas que usam da ignorância de alguns para cometer ataques digitais, prejudicando ou tomando vantagem de pessoas, grupos ou organizações. A afirmativa anterior é confirmada por um gráfico (Figura 24) elaborado pela *Surfshark* (SURFSHARK, 2023), empresa provedora de serviços de *Virtual Private Network* – Rede Privada Virtual (VPN), que analisa o crescimento anual de custos em cibersegurança.

Figura 24 – Crescimento anual de custos com cibercrime



Fonte: (SURFSHARK, 2023)

Por isso surgiram leis e normas que regularizam como os dados devem ser tratados e também tecnologias que auxiliam a proteger os dois lados da comunicação: os clientes,

que desejam ter seus dados protegidos, e das organizações, que precisam se certificar da identidade do usuário. Isto posto, serão abordadas as técnicas e metodologias adotadas a fim de garantir que o software desenvolvido atenda as demandas da legislação e promova segurança e privacidade a seus utilizadores.

4.5.1 Critérios de Segurança e Privacidade

Na aplicação desenvolvida, foram definidas formas de manter a segurança de todas as partes envolvidas. Foi implementado um método de cadastro e *login* facilitado com geração de *tokens* e também uma infraestrutura de rede que protege o dispositivo que armazena o banco de dados da aplicação.

4.5.1.1 Cadastro e *Login* com Conta Google

Um dos requisitos da entidade parceira, era o desenvolvimento de um sistema de cadastro e *login* facilitados, haja vista que muitos dos clientes tinham dificuldade em acessar a aplicação anterior por constantemente esquecerem sua credenciais (*e-mail* e senha). Considerando essa dificuldade, foi implementado um sistema de registro e *login* utilizando a API de autenticação *OAuth* da Google, pelo fato da maioria dos dispositivos no Brasil apresentarem sistema operacional Android, conforme destaca uma pesquisa (APPMYSITE, 2025), que geralmente requerem uma conta Google para seu funcionamento. E mesmo indivíduos com aparelhos de outro sistema operacional, comumente possuem contas Google para usufruir de seus serviços. Assim sendo, a responsabilidade de identificar os usuários da aplicação foi terceirizada para a Google, e quando estes se cadastram, devem aceitar suas políticas e termos de usuário que definem extensamente como os dados são processados, tratados e protegidos.

4.5.1.2 Infraestrutura de Rede

Os dispositivos (máquinas virtuais) que sustentam a aplicação estão hospedados na AWS, logo sendo de sua inteira responsabilidade protegê-los fisicamente, como diz seu Modelo de Responsabilidade Compartilhada (AWS, 2025), porém no que tange a software e redes, cabe ao time de desenvolvimento proteger. Para evitar acessos indevidos aos sistema interno, criaram-se duas instâncias EC2, uma delas sendo pública e outra privada. Como já explicado, a instância privada não possui IP público, portanto só é possível acessá-la pela rede interna dentro da infraestrutura de rede criada, delimitando uma camada a mais de segurança. O acesso a essa instância é feito pela instância pública por meio do protocolo SSH, que possui seus próprios métodos de segurança com esquema de chaves de acesso.

4.5.1.3 Controle de Acesso Baseado em Papéis

Outra medida de segurança, dessa vez mais relacionada a estrutura do software em si, é o controle de acesso baseado em *roles*, traduzido geralmente como papéis. Na aplicação desenvolvida existem diversas páginas disponíveis, sendo cada uma delas destinada a um tipo de usuário (papel), como gerente, profissional ou cliente. Assim, faz-se necessário uma maneira de bloquear e liberar o acesso a esses recursos conforme o papel do usuário atual, evitando que clientes do salão tenham acesso a páginas de relatório por exemplo. O controle de acesso foi implementado por meio do mapeamento dos papéis e permissões dentro da aplicação. Consequentemente, toda vez que uma página é requisitada, faz-se uma verificação do papel do usuário atual, que recebe ou não permissão para acessá-la. Dessa forma os recursos são disponibilizados de forma consoante ao usuário. Resolvendo o problema de acessos indevidos a partes do sistema e também tornando a experiência do usuário coerente.

4.5.2 Observância à Legislação

No Brasil a legislação que define como os dados devem ser manipulados digitalmente é a Lei Geral de Proteção de Dados, conhecida pelo seu acrônimo LGPD (BRASIL, 2018). De acordo com a LGPD, os dados que coletamos não se enquadram como dados sensíveis, mas apenas como dados pessoais, portanto a aplicação se isenta de muitas restrições legislativas.

Para utilização dos dados pessoais coletados, foi elaborada uma política de usuário que deve ser aceita antes que se conclua o cadastro na plataforma, provendo informações sobre quais dados estão sendo coletados, para qual finalidade e como são tratados. Esses dados não são utilizados de forma a prejudicar o usuário, excluindo ou tratando-o de maneira diferente por motivos pessoais, políticos ou étnicos.

Ademais, o usuário pode visualizar e alterar esses dados a qualquer momento dentro da aplicação sem quaisquer tipo de restrição e como já discutido, diversos mecanismos de segurança foram implementados e empresas consolidadas no ramo de tecnologia são responsáveis pelas questões de infraestrutura física e autenticação, garantindo a integridade, confidencialidade e disponibilidade dos recursos.

4.6 Modelo de Banco de Dados

Para o desenvolvimento da aplicação foram elaborados previamente modelos de representação do banco de dados. Tais modelos auxiliam a ter uma visão da aplicação antes que seja de fato desenvolvida.

4.6.1 Modelo Entidade Relacionamento - MER

O Modelo Entidade-Relacionamento (MER) explica muito sobre a aplicação e já aponta algumas regras de negócio. Nele foram definidas as seguintes entidades e relacionamentos:

- *CUSTOMER*: Entidade que representa um cliente.
  - ***refers:*** um cliente (*CUSTOMER*) pode ou não indicar (*refers*) outros clientes e um cliente é indicado por nenhum ou um único cliente.
  - ***makes:*** um cliente faz (*makes*) ou não agendamentos (*APPOINTMENT*).
- *APPOINTMENT*: Entidade que representa um agendamento.
  - ***makes***: um agendamento (*APPOINTMENT*) é feito (*makes*) por um cliente (*CUSTOMER*).
  - ***has***: um agendamento (*APPOINTMENT*) tem (*has*) ou não uma avaliação (*RATING*).
  - ***sends***: um agendamento (*APPOINTMENT*) envia (*sends*) ao menos uma notificação (*NOTIFICATION*).
  - ***includes***: um agendamento (*APPOINTMENT*) inclui (*includes*) uma oferta (*OFFER*).
- *RATING*: Entidade que representa a avaliação de um agendamento.
  - ***has:*** uma avaliação (*RATING*) é tida (*has*) por um agendamento (*APPOINTMENT*).
- *NOTIFICATION*: Entidade que representa uma notificação.
  - ***sends:*** uma notificação (*NOTIFICATION*) é enviada por (*sends*) um agendamento (*APPOINTMENT*).
- *OFFER*: Entidade associativa que representa a oferta (*offers*) de um serviço (*SERVICE*) por um profissional (*PROFESSIONAL*).
  - ***includes***: uma oferta (*OFFER*) pode ou não ser incluida (*includes*) em muitos agendamentos (*APPOINTMENT*).
- *PROFESSIONAL*: Entidade que representa um profissional.
  - ***offers***: um profissional (*PROFESSIONAL*) oferece (*offers*) ou não muitos serviços (*SERVICE*).

  - ***has***: um profissional (*PROFESSIONAL*) tem (*has*) ou não muitos turnos (*SHIFT*).
  - ***has***: um profissional (*PROFESSIONAL*) tem (*has*) ao menos um papel (*ROLE*).

- *SERVICE*: Entidade que representa um serviço no salão de beleza.
  - ***offers***: um serviço (*SERVICE*) é oferecido (*offers*) ou não por muitos funcionários (*PROFESSIONAL*).

- *SHIFT*: Entidade que representa os turnos de trabalho de um profissional.
  - ***has***: um turno (*SHIFT*) é tido (*has*) por um profissional (*PROFESSIONAL*).

- *ROLE*: Entidade que representa os papéis que um profissional possui na plataforma.
  - ***has***: um papel (*ROLE*) é tido ou não (*has*) por muitos profissionais (*EMPLOYEE*).
  - ***has***: um papel (*ROLE*) tem (*has*) ao menos uma permissão (*PERMISSIONS*).

- *PROFESSIONAL_ROLE*: Entidade associativa auxiliar para *PROFESSIONAL* e *ROLE*.

- *PERMISSIONS*: Entidade que representa as permissões que cada papel provê.
  - ***has***: uma permissão (*PERMISSIONS*) é tida (*has*) ou não por muitos papeis (*ROLE*).

- *ROLE_PERMISSION*: Entidade associativa auxiliar para *ROLE* e *PERMISSIONS*.

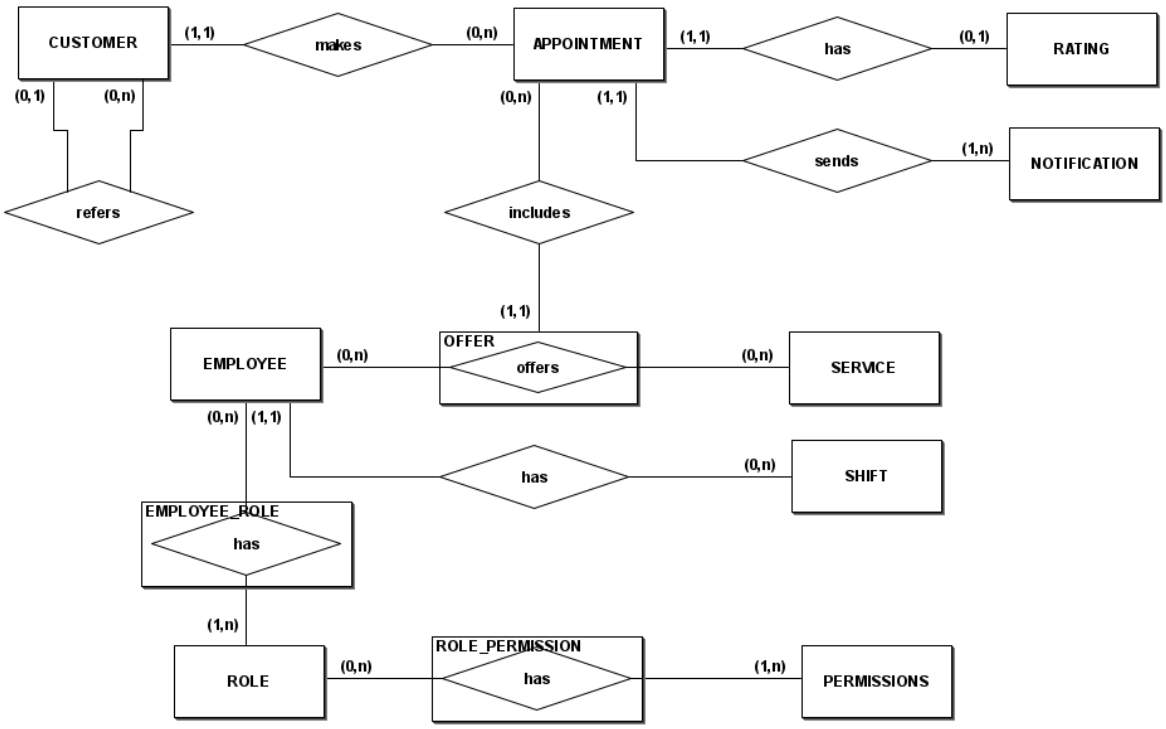
4.6.2 Diagrama Entidade Relacionamento - DER

Após a elaboração do MER conforme o entendimento dos requisitos e necessidades da entidade parceira, foi produzido um Diagrama Entidade-Relacionamento (DER), a partir do MER, que contém a listagem de todos os atributos das entidades modeladas, apresentando seus nomes, o tipo de dado e sua possível categorização como chave primária (identificadora) ou estrangeira.

4.6.3 Dicionário de Dados

Pensando no modelo físico de banco de dados, também foi elaborado um dicionário de dados referente ao sistema que se enquadra como uma poderosa ferramenta de documentação. Com esse instrumento, qualquer pessoa pode entender como foi pensado os atributos do sistema, como são armazenados e qual seria a finalidade de cada um. O dicionário descreve:

Figura 25 – MER

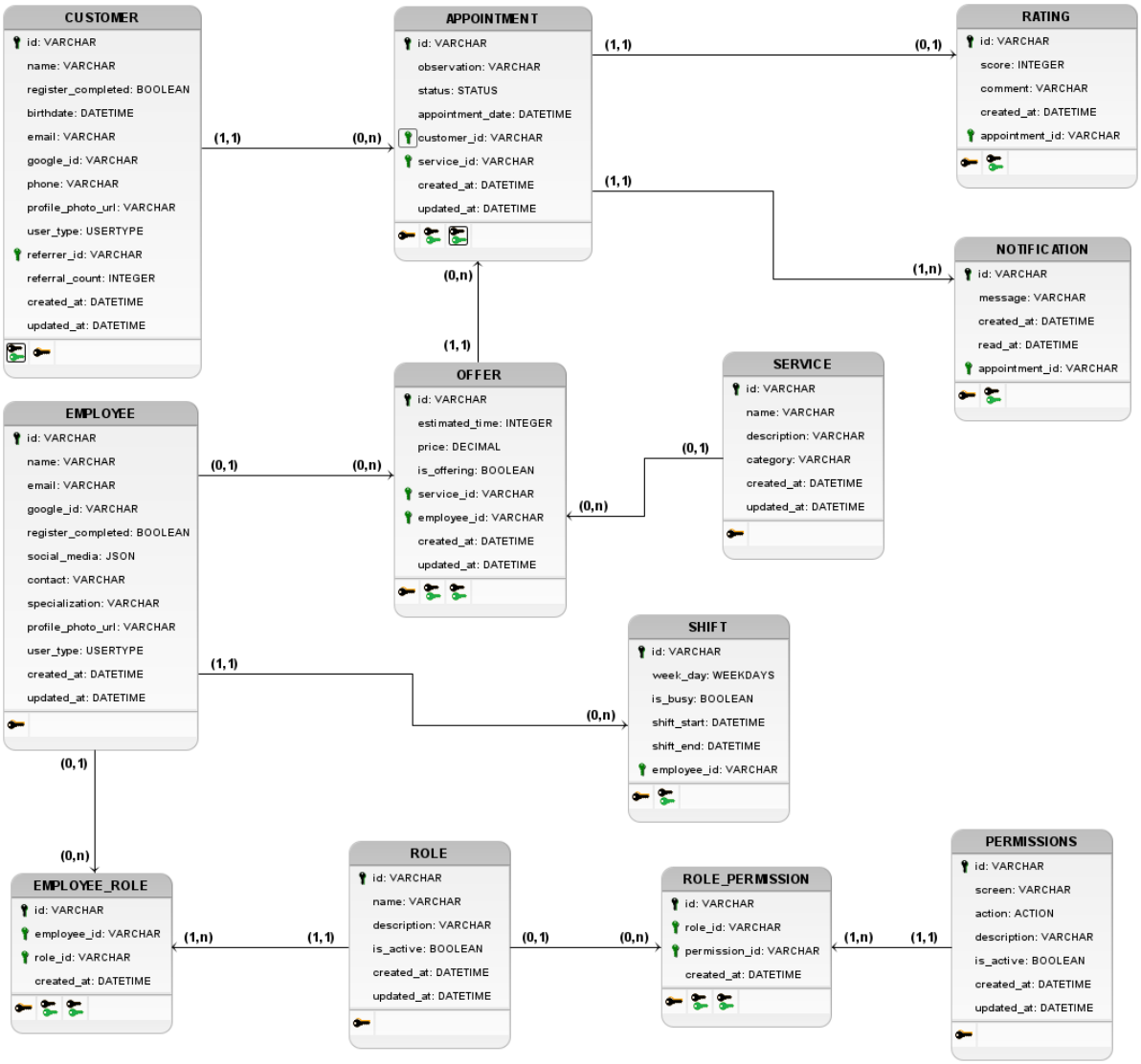


Fonte: Produzido pelos autores

- ***Primary Key* (PK)/*Foreign Key* (FK):** Se um atributo é chave primária ou estrangeira.
- **Nome do Campo:** Qual o nome do atributo.
- **Tipo:** Qual o tipo de dado do atributo, como *INT* para valores numéricos inteiros.
- **Descrição:** A descrição do atributo, explicando o que representa na prática.
- ***Null:*** Se o campo pode ser nulo, ou seja, não ter valor atribuído.
- **Tamanho:** O tamanho do atributo, como quantidade de caracteres (*bytes*).
- **Valores permitidos:** Quais os valores permitidos para atributos. Alguns campos utilizam enumerações, por exemplo, que possuem valores definidos previamente.
- **Observações:** Algumas especificidades, como valores padrão, ou de onde uma FK foi tomada.

Abaixo se encontra um *QR Code* que leva à planilha onde foi elaborado o dicionário de dados desta aplicação.

Figura 26 – DER



Fonte: Produzido pelos autores

Figura 27 – *QR Code* do Dicionário de Dados

Fonte: Produzido pelos autores

4.7 Duração / Cronograma

Esta seção tem como propósito descrever a estimativa de tempo necessária para a conclusão do desenvolvimento do projeto. A definição da duração fundamenta-se no uso do *framework* Scrum (AWS, 2024) e da ferramenta *ProjectLibre* (PROJECTLIBRE, 2025).

4.7.1 Análise da duração do projeto

Conforme o Quadro 6, o projeto — iniciado em março de 2025 — possui uma duração estimada de 9 meses com seu fim estabelecido em novembro considerando todas as etapas de planejamento, análise, desenvolvimentos, testes e *deploy*.

Desses 9 meses, os quatro primeiros foram dedicados ao planejamento, análise e documentação do projeto, bem com o desenvolvimento do *Minimum Viable Product* (MVP). Posteriormente a um mês de recesso, os quatro meses restantes foram voltados ao desenvolvimento de funcionalidades avançadas, testagem e *deploy* da aplicação.

Quadro 6 – Cronograma de atividades do projeto

ETAPAS	MESES									
	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Ou**
PLANEJAMENTO			■	■	■	■				
MVP				■	■	■				
RECESSO							■			
DESENVOLVIMENTO								■	■	■
TESTES								■	■	■
ENTREGA FINAL										
DOCUMENTAÇÃO			■	■	■	■	■	■	■	■

Fonte: Produzido pelos autores

Devido à flexibilidade da metodologia ágil Scrum adotada, o cronograma do projeto não apresenta um comportamento sequencial. Isso é evidenciado por meio do paralelismo existente entre diferentes etapas trabalhadas ao longo do andamento do projeto.

O Quadro 7 apresenta uma duração mais detalhada para cada etapa do projeto junto com as principais atividades desenvolvidas em cada fase.

Segundo definido na Seção 3.2, foram estabelecidos *sprints* semanais na etapa de desenvolvimento; portanto, a fase conta com 8 *sprints* contemplando a implementação de novas funcionalidades e a validação direta com a entidade parceira.

Quadro 7 – Estimativa de duração das etapas do projeto

Etapa	
1 - Planejamento	Entendimento
2 - MVP	
3 - Desenvolvimento	M
4 - Testes	
5 - Documentação	
6 - Entrega Final	

Fonte: Produzido pelos autores

Com o uso do *framework* Scrum, os *sprints* — e, portanto, o cronograma do projeto como um todo — estiveram constantemente sujeitos a mudanças conforme a complexidade das demandas e o retorno dos *stakeholders* ao longo do projeto.

Em uma outra análise, com as tarefas definidas no *ProjectLibre*, estima-se uma duração total de 188 dias para o desenvolvimento completo do projeto considerando os dias úteis em que a equipe dedicou tempo para a realização das atividades do projeto. Para mais detalhes do cronograma estabelecido no *ProjectLibre*, o arquivo `.pod` da ferramenta pode ser acessado no repositório remoto do projeto apresentado na Seção 3.3.

5 VIABILIDADE FINANCEIRA

Este capítulo apresenta uma visão geral dos custos envolvidos no desenvolvimento do sistema, detalhando os gastos com infraestrutura, equipe e ferramentas, além da receita gerada. Também são descritos três cenários financeiros (realista, otimista e pessimista), permitindo visualizar os riscos e oportunidades relacionados ao investimento.

Por se tratar de um projeto acadêmico em que não houve custos nem retorno financeiro de fato, cada seção do capítulo foi analisada com base em valores estimados através de pesquisas realizadas pela equipe. Ao todo, são feitas duas análises diferentes: uma considerando que o projeto será entregue para um cliente específico (a entidade parceira do projeto) e a outra em um eventual SaaS.

5.1 Custos

A Tabela 5 apresenta os custos com mão de obra, infraestrutura e ferramentas aplicadas no desenvolvimento do projeto.

Tabela 5 – Custos mensais estimados do projeto

Item de Custo	**Valor Unitário (R$)**	**Valor Total (R$)**
Mão de Obra		
Gestor	1.250,00	1.250,00
Tech Lead	2.000,00	2.000,00
Desenvolvedor *Fullstack* (3x)	750,00	2.250,00
Analista de Documentação	500,00	500,00
Subtotal Mão de Obra		**6.000,00**
Infraestrutura e Ferramentas		
AWS EC2 (múltiplas instâncias)	—	250,00
SonarQube (licença comercial)	—	350,00
Figma Professional (acesso *full* + dev)	—	160,00
Subtotal Infraestrutura		**760,00**
TOTAL MENSAL		**6.760,00**

Fonte: Produzido pelos autores

Para determinar os custos da mão de obra, utilizou-se a plataforma *Glassdoor* (**??**). Nela, pesquisou-se o salário médio de cada um dos cargos definidos na subseção 3.1.1 considerando a cidade de São Paulo.

Conforme aconselhado pelo orientador do projeto, considerou-se um dia de trabalho inteiro equivalente a apenas uma hora para se aproximar da disponibilidade real que os membros da equipe podiam dedicar ao desenvolvimento do projeto. Assim, o custo mensal de cada cargo foi obtido considerando o valor da hora trabalhada.

O preço da infraestrutura das instâncias EC2, operando 24 horas por dia, foi calculado utilizando a própria calculadora de preços da AWS (**??**).

Quanto às ferramentas utilizadas no projeto, levou-se em conta os preços do *SonarQube* (**??**) e os planos do *Figma* (**??**). Os valores em dólar foram convertidos considerando uma cotação média de R$ 5,50 em junho de 2025.

A Tabela 6 apresenta o custo total do projeto considerando os 9 meses de desenvolvimento previstos na Seção 4.7.

Tabela 6 – Custos totais do projeto

Item de Custo	**Valor Mensal (R$)**	**Valor Total (R$)**
Mão de Obra		
Gestor	1.250,00	11.250,00
Tech Lead	2.000,00	18.000,00
Desenvolvedor Fullstack (3x)	2.250,00	20.250,00
Analista de Documentação	500,00	4.500,00
Subtotal Mão de Obra		**54.000,00**
Infraestrutura e Ferramentas		
AWS EC2 (múltiplas instâncias)	250,00	2.250,00
SonarQube (licença comercial)	350,00	3.150,00
Figma Professional (acesso full + dev)	160,00	1.440,00
Subtotal Infraestrutura		**6.840,00**
TOTAL		**60.840,00**

Fonte: Produzido pelos autores

5.2 Receitas

Com base nos custos mensais e totais da Seção 5.1, constata-se que o projeto não é financeiramente viável para a entidade parceira, pois — mesmo parcelando — ela teria que arcar com todas as despesas do projeto descritas.

Assim sendo, será considerada uma eventual adaptação do sistema para um modelo SaaS, a fim de analisar as receitas que a aplicação poderia gerar. A Tabela 7 apresenta os valores das mensalidades e as principais funcionalidades de cada plano.

A precificação dos planos foi definida com base nos preços praticados pelos concorrentes identificados na Seção 1.4, considerando também os custos operacionais e o

público-alvo do sistema.

Tabela 7 – Projeção de receitas mensais - SaaS para salões de beleza

Plano	
Básico	
Profissional	Voltado a salões de médio

Fonte: Produzido pelos autores

O **Plano Básico** é a porta de entrada para um salão 100% digital e organizado. Com ele, é possível centralizar todos os agendamentos, controlar os serviços e gerenciar uma pequena equipe com poucos cliques, ganhando tempo para focar no que realmente importa: os clientes. É a ferramenta essencial para profissionalizar o atendimento e dar o primeiro passo para o crescimento.

Já o **Plano Profissional**, foi desenhado para salões com maior volume de atendimentos, que buscam crescimento acelerado e máxima lucratividade. Ele automatiza tarefas repetitivas, prepara relatórios visuais e cria campanhas que atraem e fidelizam clientes. É a inteligência que um negócio precisa para tomar as melhores decisões e crescer de forma sustentável e lucrativa.

5.3 Análise Financeira

Esta seção apresenta a avaliação econômica do projeto, considerando diferentes cenários de receita e custos operacionais. O objetivo é demonstrar a viabilidade financeira do sistema quando estruturado como um modelo SaaS, destacando a relação entre investimentos, receitas projetadas e retorno esperado. A análise contempla tanto a perspectiva conservadora quanto cenários mais otimistas, fornecendo uma visão abrangente sobre a sustentabilidade econômica do projeto e os fatores que influenciam o sucesso financeiro da aplicação.

5.3.1 Cenários

Nesta subseção, são detalhados três cenários de desempenho financeiro do projeto: pessimista, realista e otimista. Cada cenário considera diferentes taxas de crescimento da base de clientes e respectivas receitas provenientes das mensalidades, bem como os custos associados à manutenção e operação do sistema. A análise permite identificar o ponto de equilíbrio, projetar lucros ou prejuízos e avaliar indicadores financeiros importantes, como ROI e Payback Period, para cada contexto estudado.

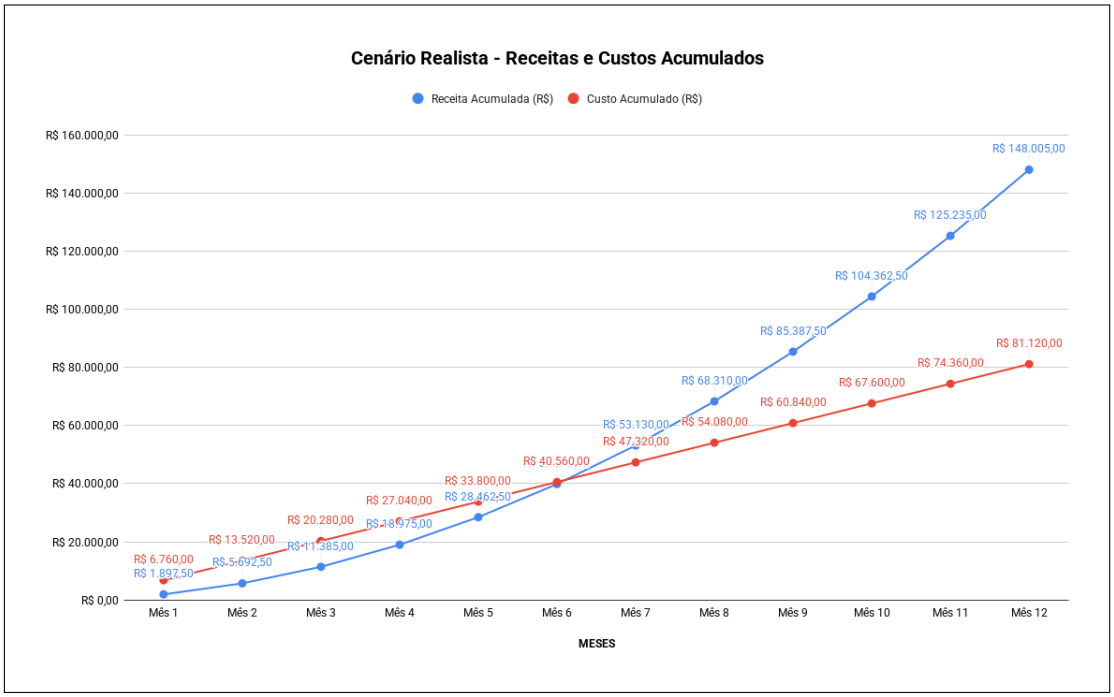
5.3.1.1 Cenário Realista

A Figura 28 apresenta as receitas e custos acumulados no cenário realista considerando o projeto como um SaaS. Nesse contexto, para o cálculo do acúmulo de receitas, estabeleceu-se um aumento mensal razoável de 15 clientes para o plano básico e 10 para o profissional.

Os custos consideram somente os gastos com mão de obra, infraestrutura e mensalidades ou licenças das ferramentas utilizadas no desenvolvimento e manutenção da aplicação.

No cenário realista, o ponto de equilíbrio é atingido no sétimo mês e, ao fim dos doze meses de análise, há um lucro de R$ 66 mil.

Figura 28 – Cenário realista



Fonte: Produzido pelos autores

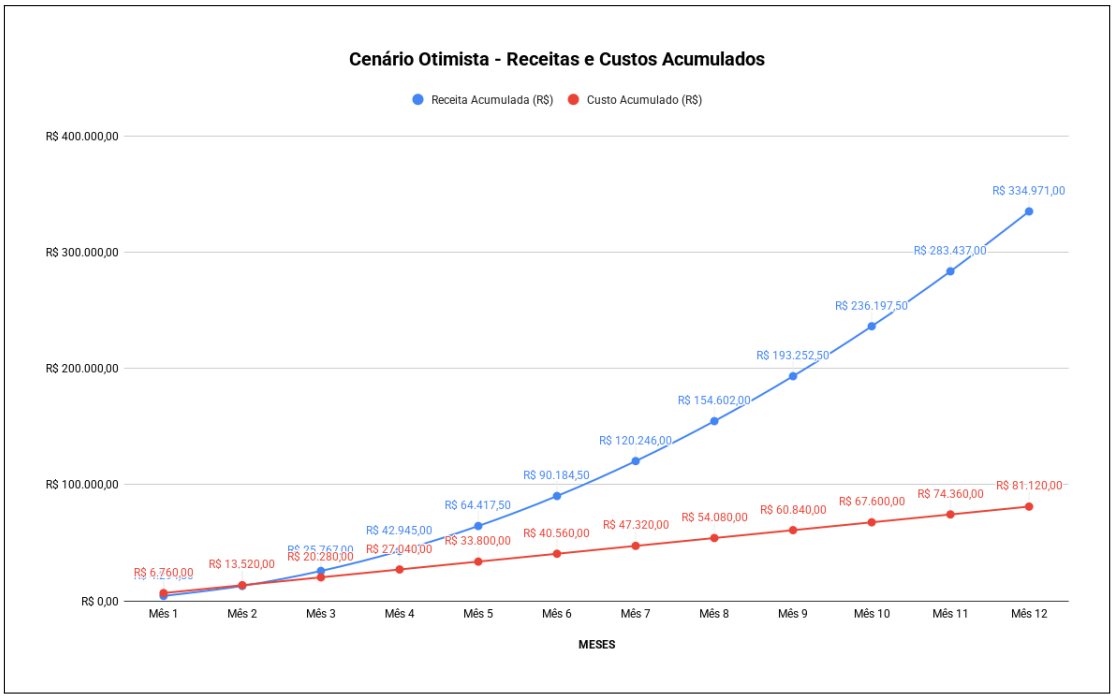
5.3.1.2 Cenário otimista

A Figura 29 apresenta as receitas e custos acumulados no cenário otimista considerando o projeto como um SaaS. Nesse contexto, para o cálculo do acúmulo de receitas, estabeleceu-se um aumento mensal acentuado de 30 clientes para o plano básico e 25 para o profissional.

Os custos consideram somente os gastos com mão de obra, infraestrutura e mensalidades ou licenças das ferramentas utilizadas no desenvolvimento e manutenção da aplicação.

No cenário otimista, o ponto de equilíbrio é atingido no terceiro mês e, ao fim dos doze meses de análise, há um lucro de R$ 253 mil.

Figura 29 – Cenário otimista



Fonte: Produzido pelos autores

5.3.1.3 Cenário pessimista

A Figura 30 apresenta as receitas e custos acumulados no cenário pessimista considerando o projeto como um SaaS. Nesse contexto, para o cálculo do acúmulo de receitas, estabeleceu-se um aumento mensal baixo de 5 clientes para o plano básico e 3 para o profissional.

Os custos consideram somente os gastos com mão de obra, infraestrutura e mensalidades ou licenças das ferramentas utilizadas no desenvolvimento e manutenção da aplicação.

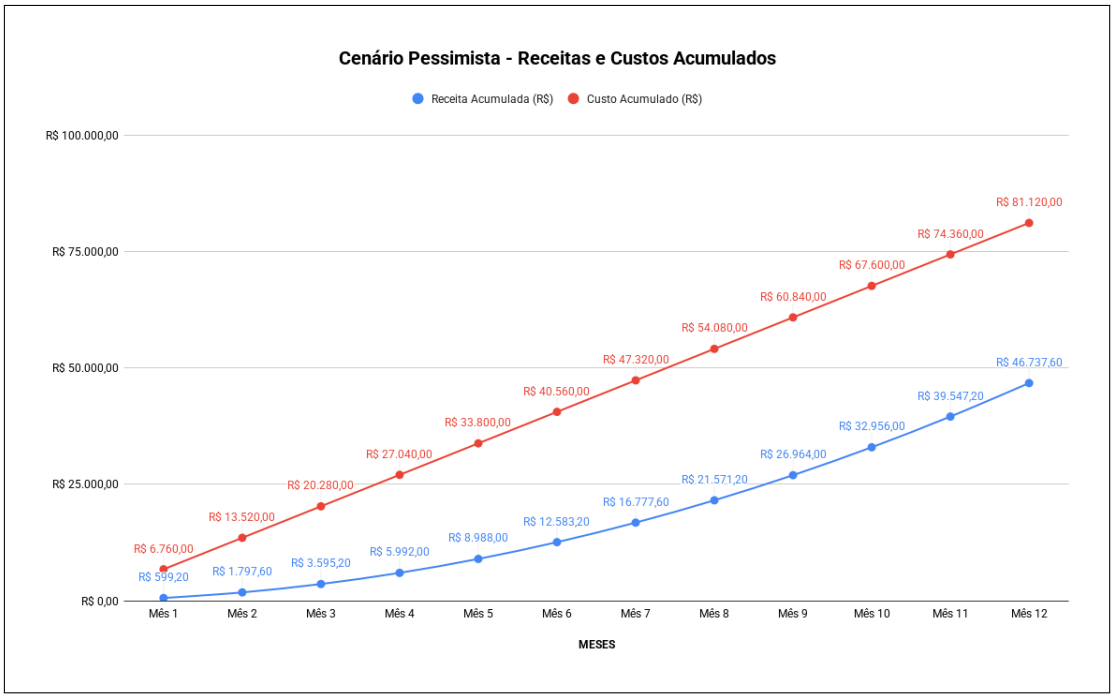
No cenário pessismista, o ponto de equilíbrio não é atingido no doze meses de análise e, ao fim desse intervalo de tempo, há um prejuízo de R$ 34 mil.

5.3.2 Indicadores financeiros

Para complementar a análise dos cenários apresentados, foram utilizadas métricas financeiras amplamente empregadas em estudos de viabilidade econômica, como o **Retorno sobre o Investimento (ROI)** e o **Período de Retorno (Payback Period)**.

O **ROI (Return on Investment)** mede a relação entre o lucro obtido e o investimento inicial, indicando o percentual de retorno gerado pelo projeto. Sua fórmula é:

Figura 30 – Cenário pessimista



Fonte: Produzido pelos autores

$$ROI = \frac{\text{Lucro Líquido}}{\text{Investimento Total}} \times 100$$

O **Payback Period** representa o tempo necessário para recuperar o investimento inicial a partir das receitas acumuladas, sendo calculado como:

$$\text{Payback} = \frac{\text{Investimento Inicial}}{\text{Receita Líquida Média Mensal}}$$

5.3.2.1 Cenário realista

Considerando o cenário realista, em que o aumento mensal é de 15 clientes no plano básico e 10 no profissional, ao fim de 12 meses, o lucro acumulado obtido somente pelas mensalidades é de aproximadamente R$ 59 mil. Com investimento inicial de R$ 60.840 (Tabela 6), o ROI estimado é de cerca de 108% e o *payback* ocorre antes de completar um ano.

5.3.2.2 Cenário otimista

No cenário otimista, com crescimento de 30 clientes no básico e 25 no profissional por mês, ao fim de 12 meses, o lucro acumulado pelas mensalidades é de R$ 253 mil, resultando em ROI de aproximadamente 416% e confirmando o rápido retorno do investimento. O *payback* ocorre em cerca de 3 meses.

5.3.2.3 Cenário pessimista

No cenário pessimista, com aumento de apenas 5 clientes no plano básico e 3 no profissional por mês, ao fim de 12 meses, o projeto apresenta prejuízo de R$ 34 mil, refletindo ROI negativo e ausência de payback no período analisado.

6 CONSIDERAÇÕES FINAIS

A aplicação web *BS Beauty* representa uma contribuição relevante para o mercado de serviços de beleza, sobretudo no modelo de *coworking*. Ao centralizar e digitalizar o agendamento e a gestão do fluxo de trabalho de diferentes profissionais autônomos, a aplicação reduz erros administrativos e melhora a experiência do cliente. Considerando que, segundo estudo do (SENAC-SP, 2022), até 30% do tempo dos pequenos empreendedores é consumido em tarefas manuais, o *BS Beauty* oferece um diferencial competitivo, permitindo que gestores e profissionais dediquem mais tempo ao atendimento do que às operações administrativas.

Durante o desenvolvimento da aplicação, a escolha da metodologia Scrum foi fundamental para o planejamento e a entrega do projeto. Os ciclos de *sprints* permitiram validar rapidamente cada funcionalidade junto à nossa parceira de extensão, garantindo flexibilidade na evolução de requisitos. Durante discussões sobre os requisitos, o módulo de pagamento *on-line* foi estrategicamente descartado do sistema, uma vez que a gestora optou por manter os pagamentos apenas de forma presencial. Além disso, o fluxo de agendamento foi inicialmente pensado como horário→profissional, porém, após sugestões do orientador, foi dividido em três caminhos distintos: agendamento apenas por horário, apenas por profissional ou de forma combinada. Essas mudanças só foram possíveis graças à flexibilidade do Scrum.

A comunicação com a gestora Bruna e a coordenação interna da equipe, apesar de bem-sucedidas, representaram desafios significativos. A necessidade de validações constantes das regras de negócio, aliada a conflitos de agenda, impossibilitou reuniões presenciais com todos os *stakeholders*. Por isso, grande parte das interações foi conduzida por mensagens de texto ou ligações. Ferramentas como *Discord*, *GitHub* e *Taiga* foram essenciais para alinhar demandas, formalizar decisões e manter a coesão no código e na documentação.

Espera-se que, após a implantação, o sistema elimine conflitos de agenda, reduza a taxa de não-comparecimento por meio de lembretes automáticos e aumente a receita mensal em virtude da otimização da ocupação das estações, da clareza nos relatórios e da fidelização de clientes. Além disso, ao disponibilizar *dashboards* de performance e indicadores de satisfação, o *BS Beauty* criará uma base de dados estratégica para decisões futuras de *marketing* e expansão.

Olhando para o futuro, o encerramento deste ciclo inicial abre caminho para novas fases de evolução do projeto. Os próximos passos para o *BS Beauty* podem incluir o desenvolvimento de um aplicativo móvel dedicado aos profissionais, para consulta ágil

de agenda e comissões, e a implementação de um módulo de análise de dados avançado. Este módulo poderá, futuramente, utilizar o histórico de agendamentos para gerar insights preditivos sobre horários de pico e preferências de clientes, consolidando a plataforma como uma ferramenta não apenas operacional, mas também estratégica.

Em síntese, este projeto de extensão cumpriu seu objetivo ao entregar uma solução digital que se destaca da concorrência por meio de diferenciais claros: uma interface minimalista focada no *coworking*, um sistema robusto para a gestão de aluguel de estações e um modelo de negócio transparente, com todas as funcionalidades operacionais necessárias inclusas no plano básico e sem taxas de pagamento. Por fim, o *BS Beauty* não é apenas um sistema de agendamento, mas uma ferramenta de empoderamento para pequenos empreendedores, projetada para apoiar o crescimento sustentável dos *coworkings* de beleza no Brasil.

REFERÊNCIAS

ABIHPEC. *Relatório de Gestão de Salões de Beleza*. 2021. Relatório ABIHPEC. Disponível em: <https://abihpec.com.br/relatorio-gestao-saloes-2021>. Acesso em: 01 jun 2025. Citado na página 16.

Amazon Web Services. *VPC Best Practices*. 2025. <https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html>. Acesso em: 05 junho 2025. Citado 2 vezes nas páginas 49 e 50.

APPMYSITE. *Android vs iOS: Mobile Operating System market share statistics (Updated 2025)*. 2025. Disponível em: <https://www.appmysite.com/blog/android-vs-ios-mobile-operating-system-market-share-statistics-you-must-know/>. Acesso em: 7 jun. 2025. Citado na página 63.

Avec Company. *Avec — plataforma de gestão e marketplace para negócios de beleza*. 2025. Site institucional da Avec. Disponível em: <https://negocios.avec.app/>. Acesso em: 25 maio 2025. Citado na página 21.

AWS. *O que é Scrum?* 2024. Disponível em: <https://aws.amazon.com/pt/what-is/scrum/>. Acesso em: 1 jun. 2025. Citado 2 vezes nas páginas 28 e 69.

AWS. *Modelo de responsabilidade compartilhada*. 2025. Disponível em: <https://aws.amazon.com/pt/compliance/shared-responsibility-model/>. Acesso em: 7 jun. 2025. Citado na página 63.

Beauty Fair. *Coworkings de beleza: entenda o modelo e suas vantagens*. 2024. Blog "Negócios de Beleza" – Beauty Fair. Disponível em: <https://negociosdebeleza.beautyfair.com.br/coworkings-de-beleza/>. Acesso em: 27 maio 2025. Citado na página 13.

BOOKSY. *Global Consumer Trends Report*. 2022. Relatório Booksy. Disponível em: <https://www.booksy.com/global-consumer-trends-2022>. Acesso em: 01 jun 2025. Citado na página 16.

BRASIL. *Lei Geral de Proteção de Dados Pessoais (LGPD)*. 2018. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 7 jun. 2025. Citado na página 64.

CONVETIONAL COMMITS. *Conventional Commits*. 2025. Disponível em: <https://www.conventionalcommits.org/pt-br/v1.0.0/>. Acesso em: 1 jun. 2025. Citado na página 30.

DISCORD. *Discord*. 2025. Disponível em: <https://discord.com>. Acesso em: 31 mai. 2025. Citado na página 27.

ECOMMERCE NA PRÁTICA. *Mercado da beleza no Brasil 2025: tendências e números*. 2025. Disponível em: <https://ecommercenapratica.com/blog/mercado-da-beleza/>. Acesso em: 2025-06-01. Nenhuma citação no texto.

EQUIPE TOTVS. *Kanban: conceito, como funciona, vantagens e implementação*. 2023. Blog. Disponível em: <https://www.totvs.com/blog/negocios/kanban/>. Acesso em: 1 jun. 2025. Citado na página 29.

ESLINT. *ESLint · Find and fix problems in your JavaScript code*. 2025. Disponível em: <https://eslint.org/>. Acesso em: 6 jun. 2025. Citado 2 vezes nas páginas 57 e 60.

FGV. *Estudo de Produtividade em Pequenos Negócios*. 2020. Relatório FGV. Disponível em: <https://fgv.br/estudo-produtividade-pequenos-negocios-2020>. Acesso em: 01 jun 2025. Citado na página 16.

FOUNDATION, E. F. *Certbot - Get HTTPS for free*. 2025. <https://certbot.eff.org/>. Acesso em: 05 junho 2025. Citado na página 46.

Gazeta do Povo. *O que é um coworking de beleza?* 2023. Gazeta do Povo. Disponível em: <https://www.gazetadopovo.com.br/conteudo-publicitario/spazio-bellezza-coworking/o-que-e-um-coworking-de-beleza/>. Acesso em: 18 maio 2025. Citado na página 13.

Gendo Sistemas. *Gendo — Sistema de agendamento completo*. 2025. Site institucional da Gendo. Disponível em: <https://www.gendo.com.br/>. Acesso em: 25 maio 2025. Citado 2 vezes nas páginas 19 e 20.

GIT. *Git*. 2025. Disponível em: <https://git-scm.com>. Acesso em: 1 jun. 2025. Citado na página 30.

GITHUB. *Clonar um repositório*. 2025. Disponível em: <https://docs.github.com/pt/repositories/creating-and-managing-repositories/cloning-a-repository>. Acesso em: 1 jun. 2025. Citado na página 32.

GITHUB. *Documentação do GitHub Actions*. 2025. Disponível em: <https://docs.github.com/pt/actions>. Acesso em: 6 jun. 2025. Citado na página 59.

GITHUB. *GitHub · Build and ship software on a single, collaborative platform*. 2025. Disponível em: <https://github.com>. Acesso em: 1 jun. 2025. Citado na página 30.

Google Developers. *Using OAuth 2.0 to Access Google APIs*. 2025. Documentação oficial – Google Developers. Disponível em: <https://developers.google.com/identity/protocols/oauth2>. Acesso em: 05 junho 2025. Citado 2 vezes nas páginas 44 e 47.

MINDMINERS. *Pesquisa de Satisfação de Serviços Pessoais*. 2022. Relatório MindMiners. Disponível em: <https://mindminers.com.br/pesquisa-satisfacao-servicos-2022>. Acesso em: 01 jun 2025. Citado na página 16.

NOTION. *Notion*. 2025. Disponível em: <https://www.notion.com/pt>. Acesso em: 31 mai. 2025. Nenhuma citação no texto.

OBJECTIVE. *Git Flow: como funciona e quais as vantagens desse fluxo de trabalho*. 2023. Disponível em: <https://www.objective.com.br/insights/git-flow/>. Acesso em: 1 jun. 2025. Citado na página 30.

PRETTIER. *What is Prettier?* 2025. Disponível em: <https://prettier.io/docs/>. Acesso em: 6 jun. 2025. Citado na página 60.

PROJECTLIBRE. *ProjectLibre Desktop*. 2025. Disponível em: <https://www.projectlibre.com/projectlibre-desktop/>. Acesso em: 31 mai. 2025. Citado 2 vezes nas páginas 27 e 69.

Reservio. *Ferramenta de agendamento on-line vs. agendamento tradicional: Qual é o melhor para o seu negócio?* 2024. Acesso em 26 out. 2023. Disponível em: <https://www.reservio.com/pt-br/blog/dicas-de-negocios/ferramenta-de-agendamento-on-line-vs-agendamento-tradicional>. Acesso em: 2025-06-06. Citado na página 25.

SEBRAE. *Coworking de Beleza*. 2023. Disponível em: <https://sebrae.com.br/sites/PortalSebrae/conteudos/posts/coworking-de-beleza,d409d5dc8e166810VgnVCM1000001b00320aRCRD>. Acesso em: 2025-06-01. Citado na página 16.

SEBRAE. *Números mostram a pujança dos negócios de beleza*. 2023. Publicado em 25 ago. 2023. Disponível em: <https://sebrae.com.br/sites/PortalSebrae/artigos/numeros-mostram-a-pujanca-dos-negocios-de-beleza,dc88327896a76810VgnVCM1000001b00320aRCRD>. Acesso em: 2025-06-01. Citado 2 vezes nas páginas 15 e 24.

SEBRAE. *Softwares e aplicativos facilitam e melhoram a gestão do salão*. 2023. Disponível em: <https://sebrae.com.br/sites/PortalSebrae/artigos/softwares-e-aplicativos-facilitam-e-melhoram-a-gestao-do-salao,f06cac941b896810VgnVCM1000001b00320aRCRD>. Acesso em: 2025-06-07. Citado na página 25.

SEBRAE RS. *Beleza em 2025: confira as tendências para o setor*. 2024. Blog Digital SEBRAE RS. Disponível em: <https://digital.sebraers.com.br/blog/mercado/beleza-em-2025-confira-as-tendencias-para-o-setor/>. Acesso em: 27 maio 2025. Citado na página 13.

SEBRAE SC. *Coworking de beleza*. 2025. Publicado em 07 mai. 2025. Disponível em: <https://www.sebrae-sc.com.br/observatorio/relatorio-de-inteligencia/coworking-de-beleza>. Acesso em: 2025-06-01. Citado na página 16.

SENAC. *Panorama do mercado da beleza*. 2023. PDF. Disponível em: <https://forumsetorial.senac.br/assets/images/panorama_mercado_beleza.pdf>. Acesso em: 2025-06-07. Citado 3 vezes nas páginas 15, 16 e 17.

SENAC-SP. *Perfil do Empreendedor de Beleza*. 2022. Relatório SENAC-SP. Disponível em: <https://www.sp.senac.br/perfil-empreendedor-beleza-2022>. Acesso em: 01 jun 2025. Citado 2 vezes nas páginas 16 e 78.

SONARQUBE. *SonarQube Server 2025.3 Documentation*. 2025. Disponível em: <https://docs.sonarsource.com/sonarqube-server/latest/>. Acesso em: 7 jun. 2025. Citado na página 57.

SURFSHARK. *Cybercrime statistics*. 2023. Disponível em: <https://surfshark.com/research/cybercrime-risks/statistics>. Acesso em: 7 jun. 2025. Citado na página 62.

Trinks Tecnologia da Informação. *Plataforma de gestão para salões, barbearias e clínicas de estética*. 2025. Trinks Negócios. Disponível em: <https://negocios.trinks.com/>. Acesso em: 25 maio 2025. Citado na página 18.

VITEST. *Getting Started*. 2025. Disponível em: <https://vitest.dev/guide/>. Acesso em: 7 jun. 2025. Citado 2 vezes nas páginas 55 e 56.