RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 14 Nro. 1 2024
Uni-FACEF Centro Universitário Municipal de Franca

DESENVOLVIMENTO DE APLICATIVO MÓVEL PARA AGENDAMENTO DE SERVIÇOS

Ygor Silva Alves Taveira
Graduando em Sistemas de Informação – Uni-FACEF
ygortaveira@outlook.com

Yuri Fernandes de Lima Caparelli
Graduando em Engenharia de Software – Uni-FACEF
yuricaparelliofc@gmail.com

Claudio Eduardo Paiva
Mestre em Ciência da Computação – UFSCar
claudioeduardopaiva@gmail.com

Resumo
Encontrar bons profissionais é um desafio frequente, muitas vezes marcado pela incerteza em relação à qualidade dos serviços, pela escassez de referências confiáveis e pela disponibilidade limitada de opções acessíveis no momento necessário. Por isso, o objetivo deste trabalho é apresentar o projeto de desenvolvimento de um aplicativo que, por sua vez, visa facilitar o acesso a bons profissionais, além de ampliar oportunidades para trabalhadores com baixa visibilidade no mercado. O projeto delimita o necessário para a criação de um aplicativo prático, eficiente e seguro, que centraliza o gerenciamento de serviços e transações e, para isso, abrange desde a concepção da interface até a implementação de funcionalidades críticas, além da preparação para futuras inclusões, como a integração com sistemas de pagamento e agendamento *online*. A segurança dos dados também será tratada como prioridade e, para isso, serão abordadas as melhores práticas em criptografia e proteção contra ameaças cibernéticas. O projeto foi conduzido sob a visão das metodologias ágeis, permitindo adaptações rápidas e entregas contínuas. O resultado é o projeto de um aplicativo que atende às necessidades dos usuários e estabelece um padrão de eficiência e confiabilidade no gerenciamento de prestadores de serviços.

Palavras-chave: Aplicativo móvel; PHP Laravel; Pagamento *online*; Agendamento de horários.

Abstract
Finding good professionals is a frequent challenge, often marked by uncertainty regarding the quality of services, the scarcity of reliable references and the limited availability of affordable options at the time of need. Therefore, the objective of this work is to present the development project of an application that, in turn, aims to facilitate access to good professionals, in addition to expanding opportunities for workers with low visibility in the market. The project outlines what is necessary to create a practical, efficient and secure application that centralizes the management of services and transactions. To this end, it covers everything from the design of the interface to the implementation of critical functionalities, in addition to preparation for future inclusions, such as integration with payment systems and online scheduling. Data security will also be treated as a priority and, to this end, best practices in encryption and protection against cyber threats will be addressed. The project was

conducted under the vision of agile methodologies, allowing for rapid adaptations and continuous deliveries. The result is the design of an application that meets the needs of users and establishes a standard of efficiency and reliability in the management of service providers.

Keywords: *Mobile application; PHP Laravel; Online payment; Appointment scheduling.*

1. Introdução

A dificuldade em encontrar bons profissionais é um desafio recorrente, marcado pela incerteza na qualidade do serviço, pela ausência de referências confiáveis e pela dificuldade de achar opções acessíveis e disponíveis no momento certo. Esse problema afeta tanto as pessoas que estão em busca de prestadores de serviços quanto os profissionais que enfrentam desafios para se destacar em um mercado competitivo.

Diante desse cenário, podem surgir questões como: qual a melhor abordagem para desenvolver um aplicativo *mobile* que não apenas facilite o acesso a profissionais confiáveis, mas que também amplie as oportunidades para trabalhadores com baixa visibilidade no mercado? Além disso, como garantir que essa solução seja prática, eficiente e segura, atendendo tanto às necessidades de quem busca um prestador de serviços quanto às demandas tecnológicas de uma aplicação moderna? Como facilitar o processo de agendamento *online* e garantir segurança em transações de pagamento virtual?

Por isso, o objetivo deste trabalho é apresentar o projeto para desenvolvimento de um aplicativo *mobile (app)* para gerenciamento de prestação de serviços, com o propósito de facilitar a conexão entre tomadores de serviço e profissionais qualificados. Visando centralizar o gerenciamento de serviços e transações, oferecendo uma plataforma prática e segura que atenda às necessidades de ambos os lados do mercado.

Considerando as dificuldades enfrentadas pelas pessoas na busca por profissionais de confiança e as barreiras que muitos trabalhadores enfrentam para ganhar visibilidade, é proposto preencher essa lacuna ao oferecer uma plataforma centralizada e acessível, tornando o processo de contratação de serviços mais prático, eficiente e seguro.

Este projeto apresenta, entre outras coisas, a implementação do *backend* do *app* por meio do PHP Laravel, escolhido por sua robustez e sintaxe elegante, a facilidade na manutenção e escalabilidade do código, ao mesmo tempo em que oferece segurança avançada para dados sensíveis. Para a construção das interfaces, sugere-se o uso da linguagem Flutter, um *framework* criado pelo Google, que permite a construção de interfaces de usuário consistentes e responsivas para iOS e Android a partir de uma única base de código.

A gestão do projeto se deu por meio de metodologias ágeis, como o Scrum, assegurando um desenvolvimento colaborativo e flexível, capaz de responder rapidamente a mudanças nos requisitos, à medida em que o projeto caminhava.

O *app* ainda está em fase de desenvolvimento e espera-se que ofereça uma solução eficaz para as dificuldades enfrentadas no setor de prestação de serviços, uma vez que tem o objetivo de transformar a experiência dos usuários e ampliar as oportunidades de trabalho para profissionais com baixa visibilidade, criando um impacto positivo tanto para tomadores de serviço quanto para prestadores de serviços.

2. Referencial Teórico

Esta seção aborda temas como dificuldades ao se buscar serviços por meio da Internet, ferramentas para construção de aplicações *mobile* e estudo de caso baseado em dois aplicativos disponíveis no mercado que foram criados com tecnologias semelhantes às propostas para uso neste trabalho.

2.1 Aplicativos Móveis

Nos últimos anos, o desenvolvimento de aplicativos móveis tem evoluído rapidamente, impulsionado pela necessidade crescente de soluções tecnológicas que ofereçam experiências ricas e responsivas aos usuários (AWS, 2024).

Estão disponíveis no mercado diversos tecnologias para desenvolvimento móvel, como Flutter, React Native, Swift e Kotlin, cada um com suas peculiaridades e diferenciais, que podem ser utilizados de forma estratégica em cada aplicação. Um *framework* proeminente neste cenário é o Flutter, desenvolvido pelo Google, que permite a criação de interfaces de usuário para múltiplas plataformas (iOS e Android) a partir de uma única base de código. Sua arquitetura baseada em Dart e a funcionalidade de *"hot reload"* são fundamentais para a construção de interfaces dinâmicas e ágeis, permitindo uma rápida visualização das mudanças de código em tempo real (GOOGLE, 2024).

O recurso *hot reload* do Flutter ajuda criar interfaces de usuário, adicionar recursos e corrigir *bugs*. *Hot reload* funciona injetando arquivos de código-fonte atualizados na máquina virtual (VM) Dart em execução. Depois que a VM atualiza as classes com as novas versões de campos e funções, a estrutura Flutter reconstrói automaticamente a árvore de *widgets*, permitindo que seja visualizado rapidamente os efeitos das alterações (HOT RELOAD, 2024).

2.2 Principais Tecnologias para Desenvolvimento de *Backend* no Mercado Atual

Para o lado do servidor, tecnologias como Node.js, Django, Ruby on Rails e Spring Boot, são destaques no mercado, além da que está proposta no desenvolvimento deste projeto, PHP Laravel.

O Laravel é um *framework* de desenvolvimento web *open-source*, baseado em PHP, que segue o padrão de arquitetura MVC (Model-View-Controller), e simplifica tarefas complexas de *backend*, como autenticação de usuários, roteamento, gestão de sessões e cache, oferecendo uma base sólida para o desenvolvimento de sistemas de gerenciamento de serviços (TRAVERSY, 2020). Uma das suas características principais é seu enfoque em segurança. O *framework* oferece proteção robusta contra vulnerabilidades comuns, como SQL *Injection*, CSRF (*Cross-Site Request Forgery*) e XSS (*Cross-Site Scripting),* através de práticas seguras como o uso de senhas criptografadas. Além disso, o Laravel facilita a implementação de medidas de segurança avançadas, essenciais para aplicações que lidam com dados sensíveis e transações financeiras (LARAVEL, 2024).

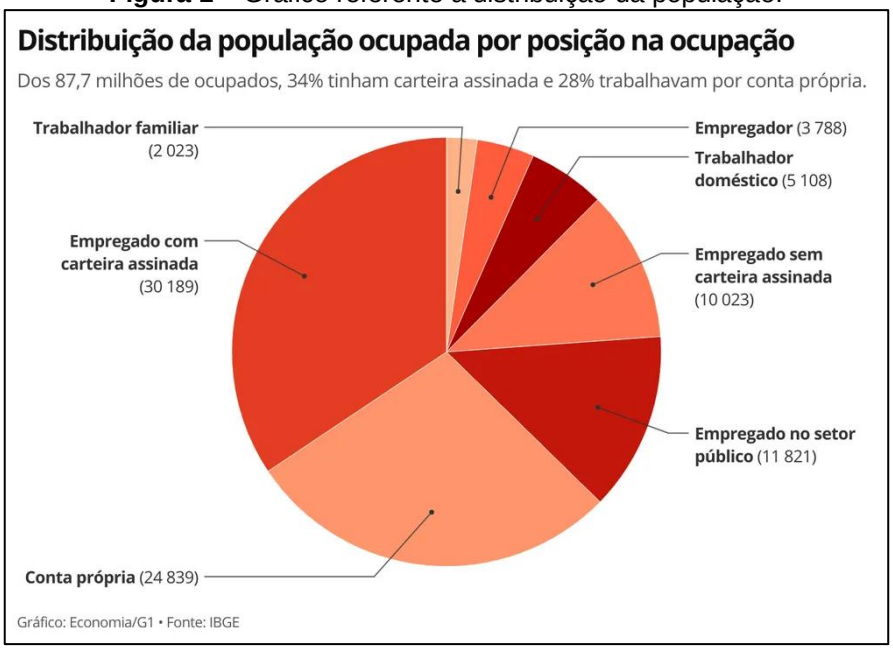
2.3 Prestação de serviços e mercado *online*

O setor de Prestação de Serviço é o que, atualmente, mais gera empregos no Brasil. Ele também representa dois terços da atividade econômica do país e impacta fortemente no PIB. De acordo com dados levantados pelo IBGE, o segmento cresceu cerca de 8,3% no ano de 2022 (AGÊNCIA BRASIL, 2023).

Reforçando essa ideia, o trabalho por conta própria tem aumentado cada vez mais no Brasil, como pode ser visto na Figura 1. Em dezembro de 2021, o número de

trabalhadores por conta própria chegou a 24,8 milhões, segundo o IBGE (Instituto Brasileiro de Geografia e Estatística) – o recorde da série histórica, iniciada em 2012. (G1 ECONOMIA, 2021).

Figura 1 – Gráfico referente a distribuição da população.



FONTE: IBGE

O trabalho autônomo pode ser exercido por um profissional informal ou por um profissional que tem CNPJ. Ele não depende de formação acadêmica e não está atrelado a uma ocupação exclusiva. Cabelereiro, pedreiro, pintor, eletricista, dentista, médico, mecânico, encanador, são exemplos de profissionais autônomos. A Figura 2 permite identificar os principais motivadores do crescimento do trabalho autônomo brasileiro.

Figura 2 - Desemprego é o que mais motiva trabalho autônomo no Brasil

FONTE: CNN Money

Em 2022, o Brasil tinha 1,5 milhão de pessoas que trabalhavam por meio de plataformas digitais e aplicativos de serviços, o equivalente a 1,7% da população

ocupada no setor privado, grande parte desses números são motoristas de aplicativos (CONSULTOR JURIDICO, 2023).

Com base nos dados apresentados, concluímos que é natural que as pessoas busquem serviços autônomos, impulsionadas tanto pelo cenário desfavorável no Brasil, com o aumento significativo do desemprego formal, quanto pela busca por independência e flexibilidade de horários.

É possível notar uma forte tendência do mercado *online* quando se trata de prestação de serviços. As lojas físicas também entenderam essa nova política de vendas, o que potencializou e inovou o mercado de produtos *online*. Motoristas e taxistas também foram impactados por essa tendência e revolucionaram a maneira de trabalhar.

Tudo isso destaca a necessidade de soluções rápidas e práticas para atender às demandas imediatas. Além disso, o cliente precisa ter confiança de que está solicitando e contratando um serviço de qualidade capaz de resolver seu problema.

2.4 Estudos de Caso e Aplicações Similares

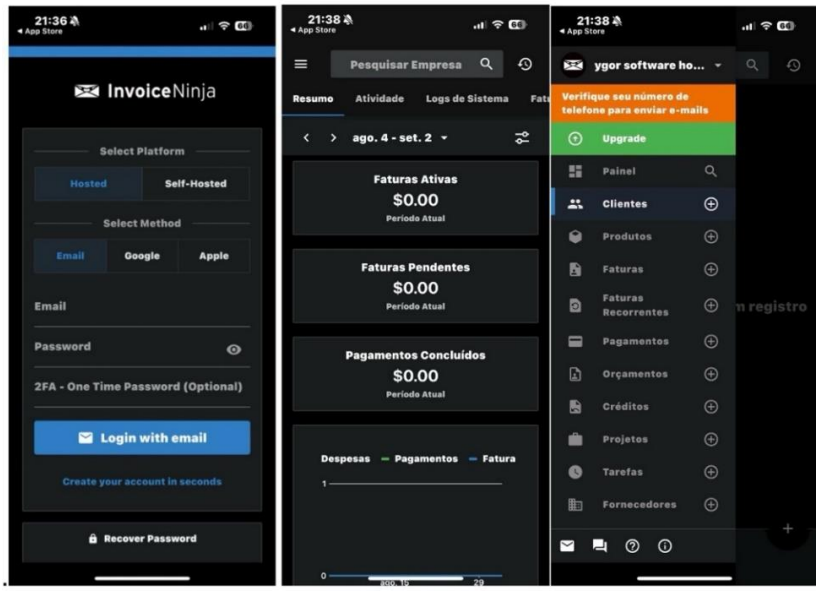
Para contextualizar a relevância do projeto, foram analisadas aplicações similares e que utilizam as tecnologias Flutter e Laravel. As aplicações estudadas são exemplos de sucesso na integração de sistemas de pagamento *online*, agendamento e gerenciamento de dados, fornecendo *insights* valiosos sobre as melhores práticas e desafios enfrentados durante o desenvolvimento.

Essas referências demonstram como a combinação de Flutter e Laravel pode resultar em soluções robustas, seguras e escaláveis, atendendo às demandas de mercados diversos. O sucesso dessas aplicações valida a escolha das tecnologias e metodologias adotadas neste projeto, evidenciando seu potencial para estabelecer novos padrões de eficiência e confiabilidade na gestão de serviços. São destacados neste trabalho:

- ***Invoice Ninja*:** uma aplicação popular para *freelancers* e pequenas empresas com serviços de faturamento, gerenciamento de clientes e pagamentos *online*. É desenvolvida com Flutter e apresenta uma experiência de usuário fluida e eficiente para a criação e gestão de faturas (Figura 3). O *backend* da aplicação utiliza PHP Laravel, que gerencia a lógica de negócios, autenticação de usuários e integrações com *gateways* de pagamento (INVOICE NINJA, 2024).

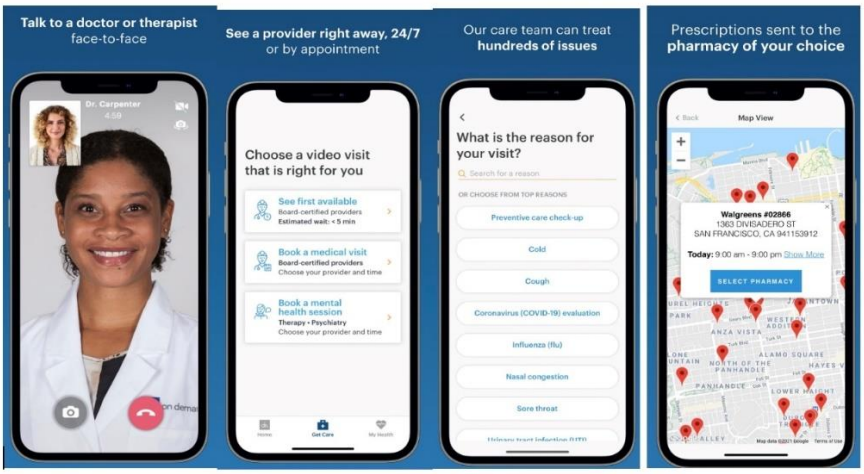
- ***Doctor On Demand:*** uma plataforma de telemedicina que conecta pacientes a médicos e outros profissionais de saúde para consultas *online*. A aplicação móvel, desenvolvida com Flutter, permite que os usuários agendem consultas, façam videochamadas com médicos e gerenciem suas informações de saúde. O *backend* é gerido por PHP Laravel, que assegura a segurança e privacidade dos dados dos pacientes, além de facilitar a integração com sistemas de pagamento para cobrança de consultas (CODES ORBIT, 2024). Com uma interface mais sofisticada, há até um mapa em sua aplicação, que permite localizar a farmácia mais próxima, por exemplo, conforme apresentado na Figura 4.

Figura 3 - Telas do app InvoiceNinja: Tela de login, tela principal e tela de ferramentas**.**



FONTE: os autores

Figura 4 - Telas do app Doctor on Demand apresentando algumas funcionalidades



FONTE: AppStore

3. Métodos e Ferramentas

Nessa seção serão apresentadas as ferramentas, estratégias de desenvolvimento e testes do aplicativo em plataformas Android e iOS, abordando desde a criação da estrutura inicial do projeto até a implementação de funcionalidades essenciais, como a integração com APIs externas, além de explorar conceitos de segurança e a organização do *backend* com PHP Laravel.

3.1 Ferramentas

O agendamento *online* é uma funcionalidade oferecida no aplicativo, permitindo que os usuários marquem compromissos e reservem serviços de forma prática e precisa. A interface desta funcionalidade poderá ser desenvolvida com Flutter, garantindo uma experiência rica e intuitiva.

No *backend*, a escolha do PHP Laravel se justifica por sua sintaxe de fácil usabilidade e por seguir o padrão arquitetural MVC (Model-View-Controller), que favorece a manutenção e o desenvolvimento de aplicações web robustas e escaláveis. A lógica de agendamento, gerida pelo *backend* em Laravel, assegura que

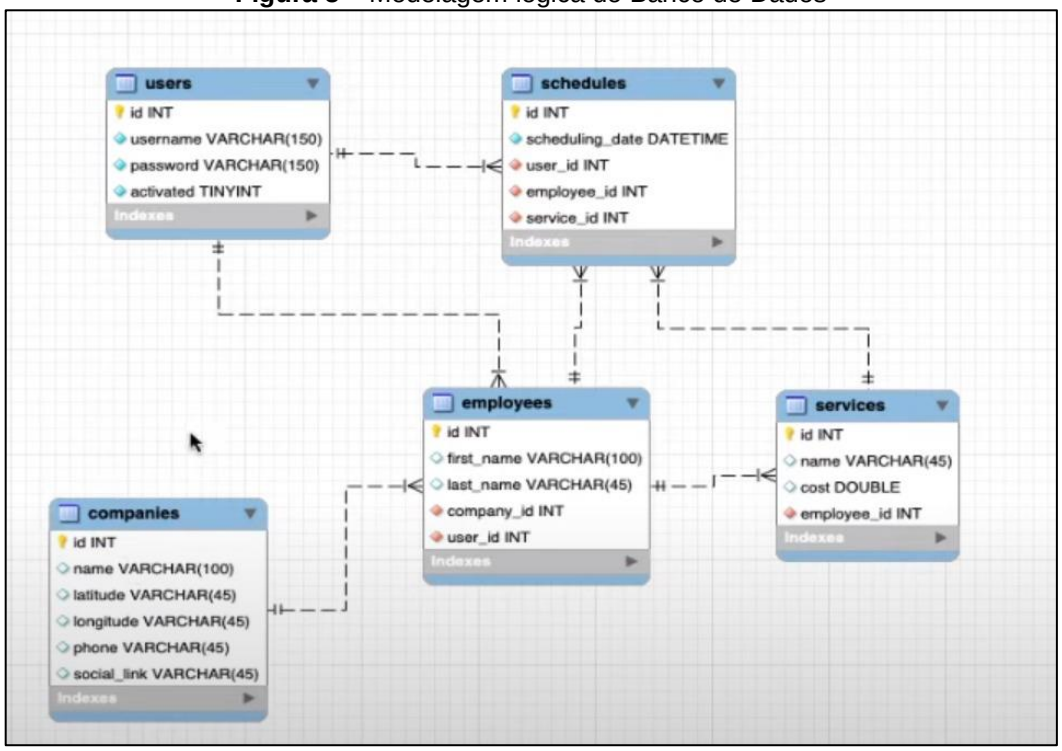
as reservas sejam processadas de forma eficiente, evitando conflitos e otimizando a gestão de tempo e recursos para prestadores de serviços.

No que diz respeito ao gerenciamento de dados, o MySQL foi escolhido como o sistema de banco de dados relacional para o aplicativo devido à sua robustez, desempenho e flexibilidade. O MySQL é reconhecido por sua capacidade de lidar com grandes volumes de dados e consultas complexas de maneira eficiente, sendo uma escolha popular para aplicações web e móveis.

A integração entre o Laravel e o MySQL foi facilitada pelo uso do Eloquent ORM (*Object-Relational Mapping),* que abstrai a complexidade das operações SQL e permite a interação com o banco de dados de forma facilitada. Além disso, o Laravel oferece ferramentas para migrações de banco de dados, validação de dados e proteção contra injeção de SQL, assegurando que as operações no *backend* sejam seguras e eficientes (TRAVERSY, 2020).

Através da análise detalhada do diagrama de entidade-relacionamento, foi possível identificar as principais relações e entidades envolvidas na modelagem do banco de dados (Figura 5). Este diagrama facilita a compreensão da estrutura e das interações dos dados.

Figura 5 – Modelagem lógica do Banco de Dados



FONTE: os autores

Para a gestão do projeto, escolheu-se o SCRUM, que facilitou a colaboração e a flexibilidade, permitindo que os requisitos fossem atendidos. Neste trabalho, a adoção da gestão por *Sprints* foi fundamental para a entrega contínua de incrementos funcionais. Além disso, a prática de revisões de código e a implementação de testes unitários asseguraram a qualidade do código e a estabilidade do software (SOMMERVILLE, 2011).

No desenvolvimento do aplicativo móvel cada ferramenta ou software utilizado desempenhou um papel específico no processo, contribuindo para diferentes aspectos do desenvolvimento.

O Visual Studio Code (VS Code) foi o ambiente de desenvolvimento integrado (IDE) utilizado para a escrita do código do projeto. Ele é popular entre os desenvolvedores por sua interface de fácil uso, extensibilidade e suporte para diversas linguagens de programação. Ele oferece recursos como depuração, controle de versão integrado e uma vasta biblioteca de extensões que facilitam o desenvolvimento e aumentam a produtividade.

Para o envio de requisições e automação de APIs, foi utilizado o Insomnia. Esta ferramenta permite testar e depurar APIs de forma eficiente, facilitando a comunicação entre o *frontend* e o *backend*. O Insomnia é especialmente útil para garantir que as integrações de API funcionem corretamente, fornecendo uma interface intuitiva para a criação e envio de requisições HTTP.

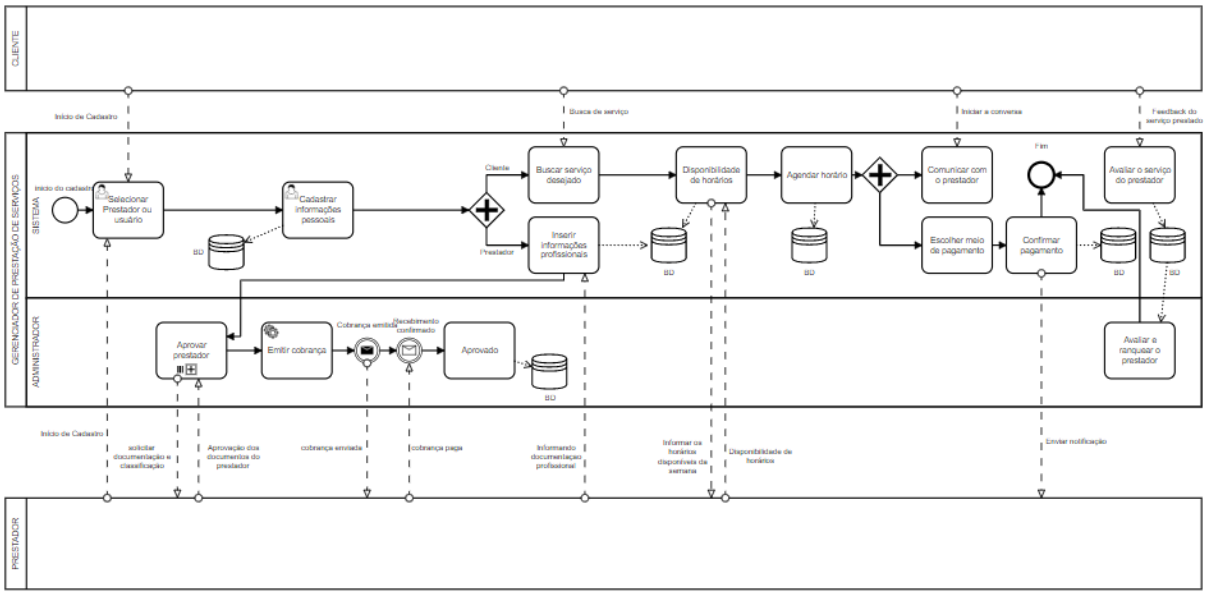
O sistema de controle de versão Git foi utilizado para gerenciar o código-fonte do projeto. O Git permite que os desenvolvedores rastreiem mudanças no código, colaborem de maneira eficiente e revertam para versões anteriores quando necessário. Ele é essencial para manter a integridade do código e facilitar o trabalho em equipe, garantindo que todos os membros do projeto estejam sincronizados e possam contribuir simultaneamente.

Estas ferramentas combinadas proporcionaram uma base sólida para o desenvolvimento do app, garantindo que cada aspecto do projeto fosse tratado com a devida atenção à qualidade e eficiência.

3.2 Artefatos da Engenharia de Software

Neste projeto a notação BPMN (*Business Process Model and Notation*) foi utilizada para mapear de forma precisa e visual todo o fluxo de trabalho do sistema, desde o cadastro do usuário até a avaliação final do serviço prestado. O diagrama, ilustrado na Figura 6, facilita a compreensão das etapas envolvidas e das responsabilidades de cada ator no processo, permitindo a análise detalhada e a identificação de possíveis melhorias no fluxo.

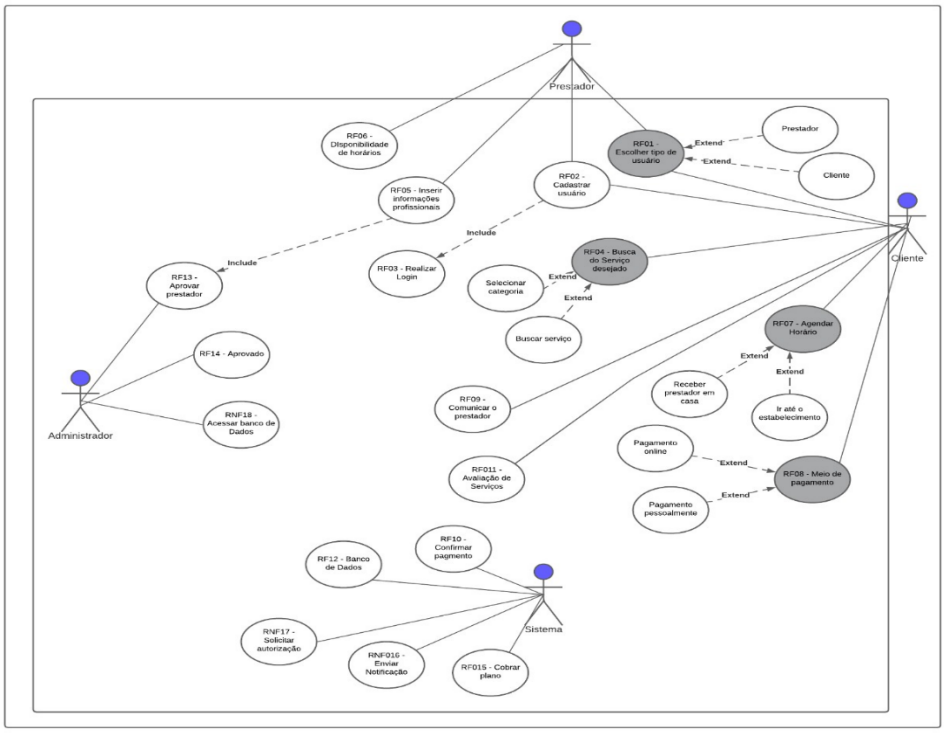
Figura 6 - Diagrama BPMN



FONTE: os autores

Foi utilizada a criação do Diagrama de Caso de Uso, que transmite uma visão geral do sistema, onde todas as funcionalidades estão concentradas (Figura 7).

Figura 7 - Diagrama de Caso de Uso



FONTE: os autores

Um diagrama de caso de uso é importante para ajudar a visualizar de maneira clara as interações entre os operadores (como clientes, administradores e sistemas) e as funcionalidades do sistema (como realizar login, buscar serviços ou confirmar pagamentos). Ele facilita a identificação dos objetivos funcionais, a comunicação entre os membros da equipe de desenvolvimento e as partes interessadas, além de apoiar a validação e refinamento do escopo do projeto. Esse tipo de diagrama também é útil para documentar o comportamento do sistema e garantir que todas as necessidades do usuário (SOMMERVILLE, 2011).

3.3 Hardware (mínimo) utilizado

O desenvolvimento exigiu o uso de bons equipamentos para assegurar um processo eficiente e uma validação adequada do resultado. Os equipamentos foram selecionados para atender às necessidades de desempenho e compatibilidade durante o ciclo de desenvolvimento.

Para a codificação e a execução das ferramentas, foram utilizados computadores pessoais, sendo eles:

- Asus-M570DD 15”, processador Ryzen 5 3500U, placa de vídeo Nvidia GTX 1050 4GB, 16GB RAM, HD 1TB e SSD 256GB NVme.
- MacBook Air 13” processador M1, 8GB RAM, SSD 256GB.

Estas máquinas ofereceram ótimo desempenho para compilar códigos rapidamente, executar ambientes de desenvolvimento integrados e simular dispositivos móveis, essenciais para validar interfaces de usuário, medir performance e acompanhar as funcionalidades do aplicativo em diferentes sistemas operacionais e tamanhos de tela.

3.4 Etapas do Desenvolvimento

O desenvolvimento seguiu uma série de etapas estruturadas. Primeiramente, a configuração do ambiente de desenvolvimento foi realizada com a instalação do Flutter SDK e a configuração das ferramentas Android Studio e Xcode, usados para o desenvolvimento e testes do *app* em plataformas Android e iOS, respectivamente.

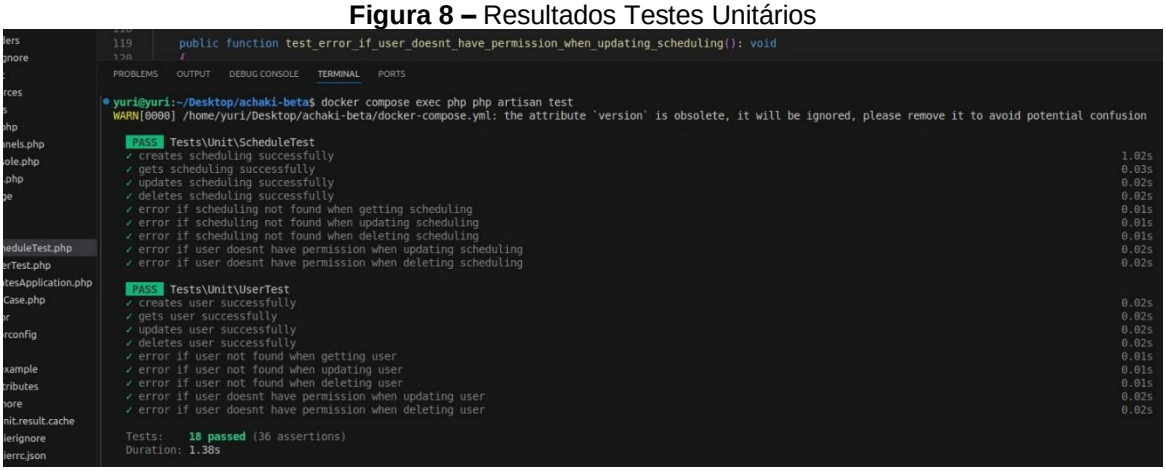
Em seguida, a estrutura inicial do projeto foi criada utilizando o comando *flutter create*, que gera a base do código necessária para iniciar o desenvolvimento. Esta etapa estabelece a organização do projeto e facilita a adição de novas funcionalidades de forma estruturada.

No desenvolvimento da interface de usuário, utilizar *widgets* pré-construídos do Flutter permitem a criação de telas interativas e visualmente atraentes. Esses *widgets* proporcionam componentes reutilizáveis e que podem ser customizados para atender às necessidades específicas do aplicativo.

Para a implementação *backend* foi utilizando o padrão MVC (*Model-View-Controller*), que separa a lógica de negócios (*Model),* a interface de usuário (*View)* e o controle das interações (*Controller*). Essa abordagem promove um código mais organizado e modular, permitindo uma manutenção mais fácil e a escalabilidade do aplicativo, além de facilitar o teste e a reutilização dos componentes.

O processo de teste e *debugging* foi continuamente realizado utilizando-se o recurso "*phpunit*" do Laravel. Este recurso permite testar os processos descritos e desenvolvidos nas funções criadas ao decorrer do fluxo do código. Na Figura 8 é possível observar os resultados obtidos nos testes unitários criados.

Figura 8 – Resultados Testes Unitários



FONTE: os autores

O código exibido na Figura 9 mostra um teste unitário para garantir que um usuário sem permissão não consiga atualizar agendamentos. Utilizando-se autenticação via Sanctum, simula-se a tentativa de alteração e valida-se o retorno de um erro 403 (*Forbidden*). Isso assegura a aplicação correta das permissões no sistema.

A Figura 10 mostra um teste unitário para verificar se a criação de um agendamento é bem-sucedida. Um usuário autenticado tenta criar um agendamento, alimentando-o com dados simulados de serviço, funcionário e data. O teste valida se a resposta do sistema retorna o código HTTP 201 (*Created*) e se a resposta contém o ID do agendamento criado, garantindo que o processo de criação está funcionando corretamente.

Figura 9 – Teste de permissão de usuário.

```php
public function test_error_if_user_doesnt_have_permission_when_updating_scheduling(): void
{
    $user = User::factory()->create();
    Sanctum::actingAs($user);

    $schedulingData = [
        'user_id' => $user->id,
    ];

    $scheduling = Schedules::factory()->create($schedulingData);

    $secondUser = User::factory()->create();
    Sanctum::actingAs($secondUser);

    $response = $this->scheduleService->updateSchedule($scheduling->id, ['schedule_date' => 'aparar a barba']);

    $this->assertEquals(Response::HTTP_FORBIDDEN, $response['code']);
    $this->assertEquals("You don't have permission to perform this action.", $response['response']['data']);
}
```

FONTE: os autores

Figura 10 – Teste de criação de um agendamento.

```php
public function test_creates_scheduling_successfully(): void
{
    $user = User::factory()->create();

    Sanctum::actingAs($user);

    $schedulingData = [
        'user_id' => User::factory()->create()->id,
        'service_id' => Services::factory()->create()->id,
        'employee_id' => Employees::factory()->create()->id,
        'schedule_date' => fake()->dateTimeBetween('now', '+1 day'),
    ];

    $response = $this->scheduleService->createSchedule($schedulingData);

    $this->assertEquals(Response::HTTP_CREATED, $response['code']);
    $this->assertArrayHasKey('id', $response['response']['data']);
}
```

FONTE: os autores

A Figura 11 mostra um teste unitário para verificar se a atualização de um agendamento ocorre com sucesso. Um usuário autenticado cria um agendamento e, em seguida, tenta atualizá-lo com uma nova data. O teste valida se a resposta do sistema retorna o código HTTP 200 (OK) e se a data do agendamento é atualizada corretamente, assegurando que o processo de atualização funciona conforme o esperado e só seja feito caso o usuário esteja autenticado, garantido a segurança.

Figura 11 – Teste de edição de um agendamento.

```php
public function test_updates_scheduling_successfully(): void
{
    $user = User::factory()->create();

    Sanctum::actingAs($user);

    $schedulingData = [
        'user_id' => $user->id,
    ];

    $scheduling = Schedules::factory()->create($schedulingData);

    $newDate = '2024-09-24 15:00:00';

    $response = $this->scheduleService->updateSchedule($scheduling->id, ['schedule_date' => $newDate]);

    $this->assertEquals(Response::HTTP_OK, $response['code']);
    $this->assertEquals($newDate, $response['response']['data']['schedule_date']);
}
```

FONTE: os autores

3.4.1 *Backend* com PHP Laravel

Para esta parte do desenvolvimento houve a instalação do Laravel através do Composer, que é um gerenciador de dependências para PHP que facilita a instalação e atualização de bibliotecas necessárias para um projeto.

Aqui a configuração inicial também incluiu a instalação de todos os pacotes necessários para o funcionamento do Laravel, bem como a configuração de variáveis de ambiente essenciais para o desenvolvimento.

Após a configuração do ambiente, foi realizada a estruturação inicial do projeto, o que envolveu a criação das pastas principais do Laravel: *Models*, *Views* e *Controllers*. A organização do código em pastas específicas para cada camada do MVC facilita a manutenção, a escalabilidade e a clareza do projeto. A estrutura inicial do projeto foi estabelecida de modo a permitir uma fácil navegação e adição de novas funcionalidades. A Figura 12 mostra um exemplo de parte do código de uma *controller* criada no projeto.

Figura 12 – *Controller* responsável pelo registro de uma empresa no banco.

```php
/**
 * Create a new company.
 *
 * @param array $companyDetails - Details of the company.
 * @return array - Returns an array with the created company or an error message.
 */
public function createCompany(array $companyDetails): array
{
    $companyDetails['user_id'] = auth()->user()->id;

    $company = $this->repository->createCompany($companyDetails);

    return $this->created($company);
}

/**
 * Get a specific company.
 *
 * @param int $companyId - The ID of the company.
 * @return array - Returns an array with the company or an error message.
 */
public function getCompanyById(int $companyId): array
{
    $error = $this->checkIfHasError($companyId);

    if (! empty($error)) {
        return $error;
    }

    $company = $this->repository->getCompanyById($companyId);

    return $this->ok($company);
}
```

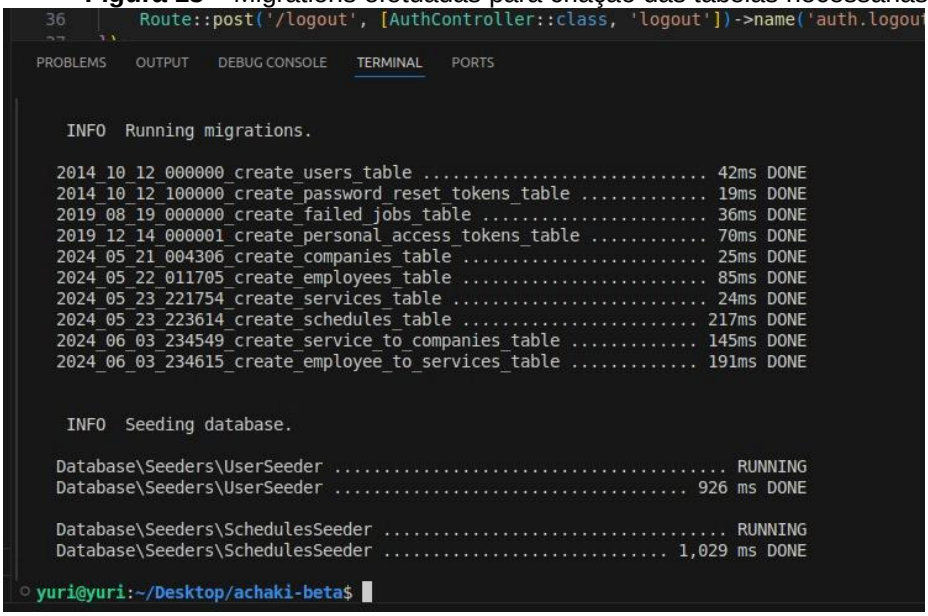
FONTE: os autores

Em seguida houve o desenvolvimento das rotas e controladores. As rotas foram definidas para mapear as URLs do aplicativo para os controladores correspondentes, que gerenciam as solicitações HTTP. Esta etapa envolveu a criação de controladores para manipular as diferentes operações CRUD (*Create, Read, Update, Delete*) e outras funcionalidades específicas do aplicativo.

Foi configurada uma conexão do Laravel com o MySQL e implementadas *migrations* para a criação das tabelas. As *migrations* permitem o versionamento do esquema do banco de dados e facilitam a gestão de alterações no esquema ao longo do tempo. Esta etapa assegurou que os dados fossem armazenados de forma estruturada e acessível. A Figura 13 mostra as *migrations* e a Figura 14 mostra um

exemplo de como se comporta as tuplas registradas em uma tabela do banco de dados.

Figura 13 – Migrations efetuadas para criação das tabelas necessárias.



FONTE: os autores

Figura 14 – Tabela de agendamento de horários

FONTE: os autores

Uma vez estabelecida a infraestrutura básica, a implementação de funcionalidades específicas do aplicativo foi iniciada.

Entre as funcionalidades desenvolvidas, foram criados sistemas de registro e login seguros para garantir a autenticação de usuários, um mecanismo para manter a autenticidade durante as sessões e um sistema de notificações para manter os usuários informados sobre eventos importantes.

A segurança do aplicativo foi uma prioridade durante todo o desenvolvimento e, para isso, foram implementadas práticas de segurança para proteger os dados dos usuários e garantir a integridade do sistema. Entre as medidas adotadas estavam o *hashing* de senhas, utilizando algoritmos para armazenar senhas de forma segura; a

proteção contra CSRF (*Cross-Site Request Forgery*) para evitar que ações não autorizadas fossem executadas em nome dos usuários; e a prevenção contra XSS (*Cross-Site Scripting*) para impedir a injeção de scripts maliciosos nas páginas web. Essas práticas de segurança garantem que o aplicativo seja resistente a diversas ameaças comuns na web.

3.4.2 Sistema de Agendamento *Online*

O sistema de agendamento *online* permite que os usuários organizem as reservas de serviços com facilidade e precisão. As etapas a seguir detalham o processo de desenvolvimento e integração desta funcionalidade presente no app.

Inicialmente, construir uma interface para a visualização de disponibilidade e agendamento de serviços é primordial. Como auxílio da criação desta interface foi efetuada uma prototipação de telas com intenção de serem amigáveis e de fácil utilização, permitindo que os usuários vejam os horários disponíveis e agendem serviços com poucos cliques na tela do *app*. A utilização de *widgets* pré-construídos presentes no Flutter, juntamente com a personalização de componentes, permitem oferecer uma experiência de usuário rica e responsiva em diferentes dispositivos e certificar compatibilidade e usabilidade em múltiplas plataformas.

Para a contrução do *backend* utilizou-se PHP Laravel para implementar *endpoints* específicos de gerenciamento das operações de agendamento, como criação, atualização, cancelamento e visualização. Estes *endpoints* foram projetados para se comunicar eficientemente com o *frontend* que será criado posteriormente, assegurando que todas as operações de agendamento sejam realizadas de forma rápida e segura. A lógica de negócios necessária para validar e processar os agendamentos foi incorporada nos controladores e modelos do Laravel, de acordo com o padrão MVC utilizado.

Estas etapas da implementação garantem que o sistema de agendamento se torne não apenas funcional e eficiente, mas também seguro e confiável, proporcionando aos usuários uma experiência positiva e protegida ao utilizarem o aplicativo e agendar serviços.

4. Resultados da solução proposta

Com o manuseio do Figma, foram criados os protótipos das telas do *app*. Nesta seção são apresentadas a prototipação de *login*, busca, cadastro, avaliação e pagamento.

A Figura 15 mostra a primeira e segunda telas apresentadas após execução do *app*. Elas são: a Tela de Apresentação e Carregamento e a Tela de Dupla Escolha, onde é permitido acessar o *app* como prestador ou cliente. Esta última apresenta ainda uma opção de cadastro ou *login*, caso já possua conta registrada.

A Figura 16 apresenta as telas de cadastro de conta de usuário, seja de um cliente ou um prestador de serviços, onde é necessário a entrada de dados da empresa do profissional.

Figura 15 - Telas Iniciais

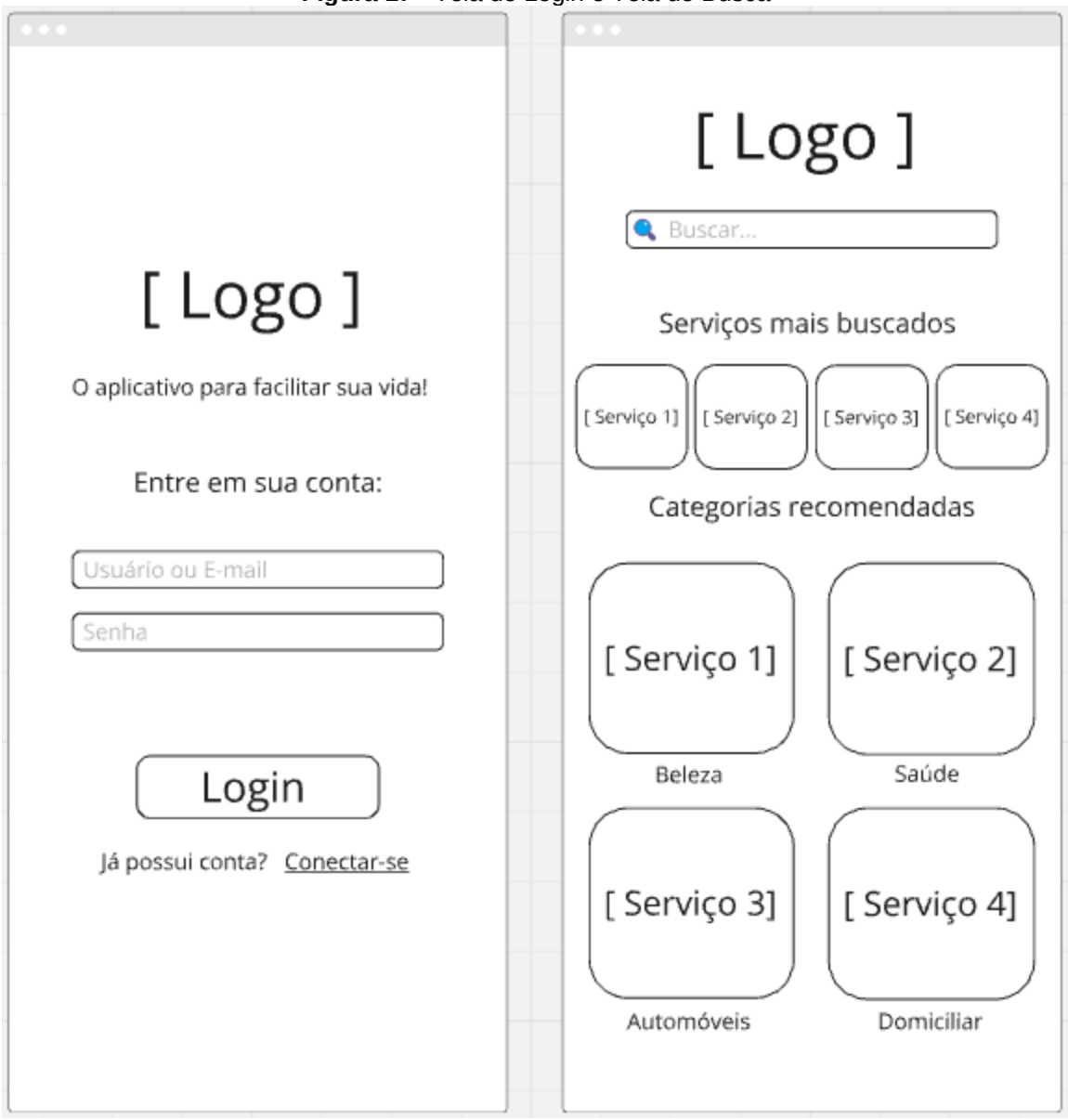
FONTE: os autores.

Figura 16 - Telas de cadastro

FONTE: os autores.

A Figura 17 apresenta a Tela de *Login*, composta pelas informações inseridas no cadastro (e-mail e senha) necessários para a autenticação no *app* e a Tela de Busca, responsável pela categorização e gerenciamento do mecanismo que dividirá as funções e serviços de cada prestador.

Figura 17 - Tela de Login e Tela de Busca



FONTE: os autores.

A Figura 18 mostra a Tela de Informações cadastradas inicialmente pelos prestadores de serviço, como dados de contato e endereços, e apresenta as Telas com funcionalidades de pagamento *online* e *feedback* onde é oferecida ao cliente a oportunidade de efetuar a avaliação do serviço prestado e assim, dar mais credibilidade ao prestador.

Figura 18 - Telas de finalização

FONTE: os autores.

A partir das telas e funcionalidades descritas, é possível identificar áreas-chave para melhorias e implementações adicionais que podem elevar a experiência do usuário e a eficiência do aplicativo.

5. Implementações e projetos futuros

Sabe-se que a integração de sistemas de pagamento *online* é uma funcionalidade requerida em aplicativos voltados para a gestão de serviços. Neste contexto, a escolha do *Stripe* como plataforma de pagamento é particularmente relevante, dada sua conformidade com as normas PCI DSS (*Payment Card Industry Data Security Standard*), que asseguram a proteção dos dados dos usuários contra fraudes e vazamentos de informações. A versão mais recente do PCI DSS, a 4.0, introduz medidas proativas de segurança, como treinamento obrigatório de conscientização em segurança e autenticação multifator para todos os usuários com acesso a dados sensíveis, o que reforça ainda mais a proteção dos sistemas de pagamento *online* (FINTECH NEWS, 2024).

A implementação de sistemas de pagamento requer uma abordagem cuidadosa tanto do ponto de vista técnico quanto regulatório. Além de garantir a segurança, é essencial que o processo de *checkout* seja rápido e intuitivo, proporcionando uma experiência de usuário sem interrupções. Estudos recentes mostram que a integração eficaz de APIs de pagamento, como as oferecidas pelo Stripe, não só melhora a segurança das transações com a utilização de criptografia avançada e tokenização, mas também aumenta a confiança dos usuários no serviço oferecido, minimizando riscos de fraudes e garantindo a conformidade com os padrões mais exigentes de segurança (STRIPE, 2024).

Para melhorar a comunicação com os usuários, considera-se integrar a funcionalidade de notificações utilizando o *Firebase*. Configurando o *Firebase Cloud Messaging* (FCM) para enviar confirmações automáticas de agendamento e lembretes próximos à data do serviço. Notificações *push* são fundamentais para garantir que os usuários recebam informações importantes em tempo real. A integração com o *Firebase* também permite a personalização das mensagens enviadas, melhorando a experiência do usuário e aumentando a taxa de comparecimento aos serviços agendados (NORMAN, 2013).

6. Considerações finais

Visando o conceito de busca de serviços e prestadores, conforme discutido ao longo deste artigo, acredita-se no potencial da solução aqui apresentada, considerando o mercado de prestação de serviços atual do Brasil.

Embora o desenvolvimento do *app* esteja em andamento, este projeto permitiu a vivência de uma série de experiências valiosas e aprendizados, que nos permitiram entender como estruturar, documentar e iniciar um aplicativo dessa magnitude.

Foram construídas partes fundamentais do *app*, como o sistema de cadastro de usuários e o *backend* e, embora ainda não totalmente finalizados, esses elementos continuam em processo de criação e melhoria e poderão ser reaproveitados em futuras versões do *app*.

Este projeto permitiu a observação de que muitas das aplicações mobile atuais utilizam metodologias semelhantes às que apresentamos ao longo deste artigo, o que reforça a aplicabilidade futura do que já foi construído.

A execução deste projeto permitiu grande crescimento, tanto estudantil, quanto profissional. Aprendemos sobre a importância de planejar, documentar e desenvolver soluções tecnológicas complexas, além de encontrarmos desafios práticos, como a gestão de tempo e a adaptação a imprevistos. Esse processo nos proporcionou uma visão mais clara sobre a utilização de metodologias ágeis e a necessidade de flexibilidade em projetos deste porte.

Durante o desenvolvimento da aplicação, nos deparamos com desafios que exigiram dedicação, especialmente pela complexidade técnica envolvida. Houve momentos em que foi preciso repensar o que havia sido planejado, e isso nos fez ajustar o percurso e encontrar novas formas de seguir em frente. No fim, esses obstáculos trouxeram aprendizados importantes e nos ensinaram a ser mais flexíveis e criativos ao longo do processo.

Para o futuro, partes desenvolvidas, como o sistema de cadastros e o *backend*, nos servirão como base para novas implementações. Estamos considerando reaproveitar esse desenvolvimento e a inclusão de novas funcionalidades, como a criação de sistemas que envolvam geolocalização.

Referências

AGÊNCIA BRASIL. Setor de serviços cresce 8,3% em 2022 e atinge recorde. Agência Brasil, 08 fev. 2023. Disponível em: https://agenciabrasil.ebc.com.br/economia/noticia/2023-02/setor-de-servicos-cresce-83-em-2022-e-atinge-recorde. Acesso em: 17 set. 2024.

AWS. “O que é Flutter?” Amazon Web Services, 2024. Disponível em: https://aws.amazon.com/pt/what-is/flutter/. Acesso em: 31 out. 2024.

CODES ORBIT. On-Demand Doctor App Development. Disponível em: https://code-sorbit.com/doctor-on-demand-app-development. Acesso em: 23 set. 2024

CONSULTOR JURIDICO, Brasil tem 1,5 milhão de trabalhadores por aplicativo, diz IBGE. Disponível em: https://www.conjur.com.br/2023-out-25/brasil-15-milhao-trabalhadores-aplicativo-ibge/. Acesso em: 17 set. 2024.

FINTECH NEWS, Future proofing PCI DSS compliance in online payment systems. Disponível em: https://www.fintechnews.org/future-proofing-pci-dss-compliance-in-online-payment-systems/. Acesso em: 4 set. 2024.

G1 ECONOMIA. Disponível em: https://g1.globo.com/economia/noticia/2021/08/31/trabalho-por-conta-propria-atinge-recorde-de-248-milhoes-de-pessoas.ghtml. Acesso em: 17 set. 2024.

HOT RELOAD. Disponível em: https://docs.flutter.dev/tools/hot-reload. Acesso em 22 set. 2024.

INVOICE NINJA. *Invoice Ninja*. Disponível em: https://github.com/invoiceninja. Acesso em: 23 set. 2024.

LARAVEL. Sobre o Laravel. Disponível em: https://laravel.com/docs. Acesso em: 04 set. 2024.

NORMAN, A. Design of Everyday Things. Nova York: Basic Books, 15 mar. 2013.

STRIPE, Secure payments by Stripe: Everything you need to know. Submigrations, 2024. Disponível em: https://www.submigrations.com/secure-payments-by-stripe. Acesso em: 4 set. 2024.

STRIPE, security explained: A guide for businesses. Disponível em: https://stripe.com/guides/payment-security. Acesso em: 4 set. 2024.

SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 14 jun. 2011.

TRAVERSY, B. PHP Laravel for Beginners: From Zero to Mastery. [S.l.]: CreateSpace Independent Publishing Platform, 23 out. 2020