VINÍCIUS ABREU GONÇALVES

RELATÓRIO TÉCNICO DE ESTÁGIO DTI SISTEMAS

LAVRAS - MG

2023

VINÍCIUS ABREU GONÇALVES

RELATÓRIO TÉCNICO DE ESTÁGIO DTI SISTEMAS

Trabalho de conclusão do curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

Prof. Dr. Antonio Maria Pereira De Resende

Orientador

LAVRAS - MG

2023

VINÍCIUS ABREU GONÇALVES

RELATÓRIO TÉCNICO DE ESTÁGIO DTI SISTEMAS

Trabalho de conclusão do curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

APROVADA em 28/11/2023.

Prof. Dr. Antonio Maria Pereira De Resende	UFLA
Prof. Dr. Janderson Rodrigo De Oliveira	UFLA
Matheus Teixeira Lemos	DTI Digital

gov.br
Documento assinado digitalmente
ANTONIO MARIA PEREIRA DE RESENDE
Data: 13/12/2023 18:31:14-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Antonio Maria Pereira De Resende
Orientador

LAVRAS - MG
2023

Dedico este trabalho a minha familia e principalmente aos meus pais, sem o apoio deles eu não seria capaz de concluir o meu curso.

AGRADECIMENTOS

Aos meus familiares, que me incentivaram a seguir em frente independente das dificuldades encontradas e nunca mediram esforços para quando eu precisei de ajuda ou de algum apoio.

A todos que fizeram parte do meu percurso dentro da área acadêmica, sejam eles professores que me corrigiram e me proporcionaram conhecimento durante a minha jornada, colegas de classe que auxiliaram nos estudos e aprendizagem no geral e, não menos importante, a CompJr, empresa júnior de desenvolvimento de software, que representa o ínicio da minha paixão pela área de desenvolvimento e foi com ela que consegui aprender e ter as portas abertas para o mercado de trabalho.

Aos meus companheiros de casa, que tornaram o meu dia a dia mais leve e realmente exerceram um papel de segunda família durante esses anos.

“Eu acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar.”
(Alan Turing)

RESUMO

A Dti Sistemas visa fornecer serviços de desenvolvimento de programas de computador sob encomenda. O presente relatório de estágio visa descrever a atuação do autor como desenvolvedor durante 1 ano e 7 meses no processo de produção do aplicativo da Ducash, um clube de benefícios do Grupo Zelo, no qual prestou-se serviços de desenvolvimento *full-stack*, abrangendo o contexto *mobile* e *back-end*. Além disso, trabalhou-se nos sistemas da Suppo7, uma empresa que fornece serviços de suporte em equipamentos eletrônicos. Exerceu-se trabalhos semelhantes porém acrescentou-se o *front-end* como *stack*. O desenvolvimento dos sistemas demandou conhecimento em diversas áreas, referentes à programação, incluindo as diretrizes: móvel, *WEB* e *back-end* e banco de dados relacionais e não relacionais. No final do estágio, após 19 meses, a atividade de estágio propiciou a participação no desenvolvimento de 2 sistemas e dentre os principais aprendizados, citam-se: a) Boas práticas de programação; b) Utilização das linguagens e *frameworks* trabalhadas; c) Desenvolvimento de testes unitários. Essa oportunidade de estágio contribuiu significativamente para a formação profissional do aluno, incluindo a aplicação de tecnologias em projetos com efetivo uso no mercado, dentre elas cita-se: React, React Native, .net, Lean Kanban, Scrum e a plataforma de gerenciamento de projetos Azure.

Palavras-chave: Desenvolvimento. Scrum. Lean Kanban. React Native. React. .net. Backend. Frontend. Banco de dados. Aprendizado.

ABSTRACT

Dti Sistemas aims to provide custom computer program development services. This internship report aims to describe the author's role as a developer for 1 year and 7 months in the production process of the Ducash app, a benefits club of Grupo Zelo, where full-stack development services were provided, encompassing both mobile and back-end contexts. Additionally, work was carried out on the systems of Suppo7, a company that provides support services for electronic equipment. Similar tasks were performed, but front-end development was added to the stack. The development of these systems required knowledge in various programming areas, including mobile, WEB, back-end, and relational and non-relational database guidelines. At the end of the internship, after 19 months, it resulted in participation in the development of 2 systems, and among the main takeaways, the following can be highlighted: a) Best programming practices; b) Utilization of the languages and frameworks worked on; c) Development of unit tests. This internship opportunity significantly contributed to the student's professional development, including the application of technologies in projects with real-world market use, including React, React Native, .NET, Lean Kanban, Scrum, and the Azure project management platform.

Keywords: Development. Scrum. Lean Kanban. React Native. React. .NET. Backend. Frontend. Database. Apprenticeship.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O atual capítulo apresenta ao leitor uma breve contextualização a respeito da empresa onde realizou-se o estágio, os principais projetos trabalhados, atividades realizadas e, além disso, a estruturação do documento.

1.1 Empresa

O presente trabalho refere-se ao estágio realizado na Dti Sistemas Ltda[1], pertencente a multinacional britânica WPP Group[2]. Essa, por sua vez, possui 115 mil funcionários, localiza-se em mais de 100 países e dentre as suas diversas intitulações, ocupa atualmente o *ranking* 307 das maiores corporações do mundo pela Fortune Global 500, fato que proporciona uma ampla rede de troca de conhecimentos e auxilia o crescimento das organizações pertencentes ao grupo.

A empresa, fundada em 28/10/2009, possui, como nome fantasia, Dti Digital Crafters e encontra-se registrada, atualmente, no seguinte endereço: Rua Levindo Lopes, 357 5º e 12º Andar - Savassi, Belo Horizonte - MG. A mesma conta hoje com cerca de 1000 funcionários e detém como principais ramos de negócio (de acordo com o registro do CNPJ consultado):

- Desenvolvimento de programas de computador sob encomenda;
- Desenvolvimento e licenciamento de programas de computador customizáveis;
- Consultoria em tecnologia da informação;
- Treinamento em informática.

Dentre os principais *cases* de sucesso da empresa, encontram-se clientes como a MRV, Localiza, Bayer, Vale, Hermes Pardini, RHI Magnesita, Telemont entre outros.

A mesma conta com uma rede de estruturas auto-organizadas, coerentes com o todo, mas com espaço para experimentação, ou seja, independe de um formato específico ou prescrições para cada parte. A estruturação é feita de forma que cada parte da empresa consiga organizar-se da maneira mais coerente e eficaz para o seu contexto, tornando assim, o trabalho mais eficiente.

[1] <https://www.dtidigital.com.br/>
[2] <https://www.wpp.com/pt-br/>

1.2 Atividades realizadas

Realizou-se o estágio no período de 15/06/2021 a 06/03/2023. Originalmente, o contrato encerraria-se no dia 15/06/2023, porém, houve rescisão por motivo de efetivação do autor na empresa.

Durante o processo realizou-se atividades em dois projetos de duas empresas denominadas Ducash, empresa pertencente ao Grupo Zelo, e a Suppo7, pertencente a Telemont.

Ambos os projetos demandaram o conhecimento de desenvolvimento *full-stack*, título dado ao desenvolvedor que trabalha tanto no *front-end* quanto no *back-end* de um *software*.

O primeiro, trata-se de um clube de benefícios utilizado por aqueles afiliados à algum plano funerário pertencente ao Grupo Zelo e também por parceiros externos através do processo de *co-branding*, que baseia-se em uma estratégia de *marketing* onde vários parceiros utilizam um mesmo produto ou serviço em benefício próprio. Por motivos financeiros, encerrou-se sua parceria com a Dti. Neste desenvolveu-se o aplicativo *mobile* da empresa e, dentre as principais atividades realizadas, encontram-se o desenvolvimento do cadastro de clientes, atualizações de *layout* na tela principal, criação da loja virtual dentro da aplicação e os seus respectivos *back-end*;

O segundo, refere-se a uma plataforma de prestação de serviços tecnológicos que objetiva realizar a ponte de conexão entre um prestador e um cliente. Nesse caso, demandou-se conhecimentos em desenvolvimento *mobile*, *front-end* e *back-end*. Desenvolveu-se sistemas referentes à diferentes demandas da empresa, como o *business to business* (B2B), um aplicativo onde pessoas selecionadas poderiam cadastrar-se para realizar demandas recebidas diretamente no celular e um sistema que possibilita administradores a importarem solicitações de serviço recebidas da OI, empresa de telecomunicação, e repassarem aos prestadores registrados. A outra demanda consiste no *business to costumer* (B2C), um sistema semelhante ao anterior que possibilita clientes do dia a dia cadastrarem as suas dificuldades e problemas, com a finalidade de encontrar um prestador adequado para solucionar. Nessa modalidade, aqueles que prestam os serviços são pessoas terceirizadas que possuem um CNPJ ativo para tal atividade.

Em ambos os casos é válido dizer que, seguindo o Dti Flow (fluxo de trabalho definido dentro da Dti), todas as tarefas exigiram a criação de testes unitários.

1.3 Organização do trabalho

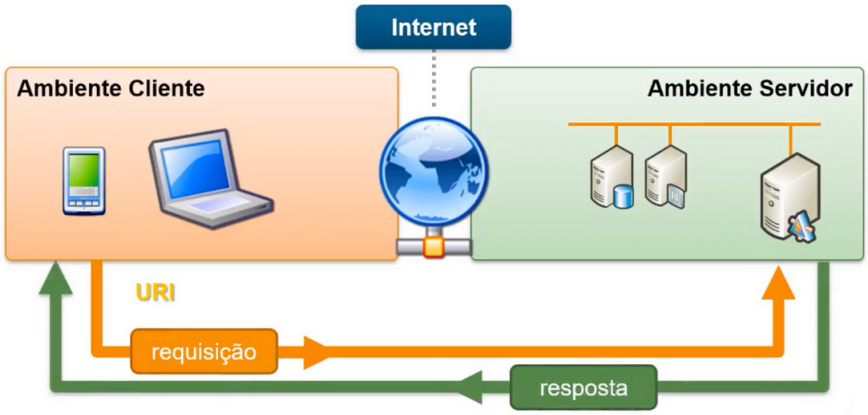
Este trabalho encontra-se organizado como segue: no Capítulo 2 apresenta-se as principais tecnologias utilizadas ao longo do estágio; o Capítulo 3, por sua vez, refere-se as atividades realizadas durante o processo de estágio. Por fim, o Capítulo 4 a conclusão deste relatório.

2 FUNDAMENTAÇÃO TEÓRICA

No presente capítulo apresentam-se conceitos, ferramentas, técnicas e metodologias utilizadas facilitando o entendimento do relatório. Além disso, o capítulo está separado em seções referentes às diferentes áreas exploradas durante o desenvolvimento.

2.1 Desenvolvimento *full-stack*

Figura 2.1 – Representação da arquitetura cliente-servidor



Fonte: (MRLRCH, 2023)

Observa-se na Figura 2.1 a arquitetura cliente-servidor. Ao lado esquerdo está o "Ambiente Cliente", através dele os usuários acessam sites e aplicativos visando realizar determinadas tarefas que sejam do seu interesse. Ao realizar uma ação, é feita uma requisição ao "Ambiente Servidor" que faz todo o processamento e retorna a resposta solicitada para o cliente.

Pode-se classificar o desenvolvedor *full-stack* como aquele que trabalha tanto com o *front-end* (ambiente cliente), quanto *back-end* (ambiente servidor). Neste caso, têm-se uma pessoa que sabe atuar em todas as partes de um sistema.

2.2 Desenvolvimento *frond-end*

O desenvolvimento *front-end* classifica-se como toda a parte visual de um site. A partir dele cria-se as interfaces gráficas previamente pensadas por um *designer* e, além disso,

realizam-se as consultas ao *back-end* e apresenta os dados retornados para o possível usuário do sistema.

2.2.1 HTML, CSS e JavaScript

Essas três tecnologias trabalham em conjunto para criar a experiência de usuário em um *website*. A Linguagem de Marcação de HiperTexto (HTML) fornece a estrutura básica do conteúdo, o *Cascading Style Sheets* (CSS) cuida do estilo e a aparência visual, e o JavaScript adiciona interatividade e funcionalidade à página.

O HTML[1] (BALLERINI, 2023), criado em 1991 por Tim Berners-Lee, objetivava possibilitar o compartilhamento de documentos de forma prática e rápida. Porém, em 1992 criou-se a *World Wide Web* (WWW ou WEB), uma rede utilizada para diversas finalidades por aqueles que desejarem, tornando assim, uma ferramenta amplamente difundida. A mesma permite estruturar e organizar o conteúdo de páginas web, possibilitando definir diversos elementos para exibição em tela do usuário que acessar.

O CSS[2] surgiu como uma consequência do crescimento do HTML. Utiliza-se para cuidar da parte estética de um *website*. Trata-se de uma linguagem de estilização que controla a aparência visual, tornando-se possível a personalização, contribuindo, além da beleza, na usabilidade de um sistema, facilitando a identificação de elementos.

O JavaScript[3] trata-se uma linguagem de programação *"client-side"*, ou seja, executada no próprio navegador que acessa a página. Com a utilização desta, torna-se possível adicionarmos interatividade, funcionalidade, responder ações que o usuário executou, manipular elementos, adicionar animações e realizar consultas ao *back-end*.

2.2.2 TypeScript

TypeScript[4] (MELO, 2021) refere-se a uma linguagem de programação considerada uma extensão do JavaScript.

Desenvolvida pela Microsoft, adiciona recursos e ferramentas inexistentes de forma nativa. Executável em qualquer ambiente que o JavaScript tenha suporte e a sua principal característica atribui-se a adição de uma tipagem estática ao mesmo, que, nativamente, trabalha com a

[1] <https://developer.mozilla.org/en-US/docs/Web/HTML>
[2] <https://developer.mozilla.org/en-US/docs/Web/CSS>
[3] <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
[4] <https://www.typescriptlang.org/docs/>

tipagem dinâmica. Isso significa que as variáveis, objetos e retornos devem possuir definições de forma explicita no código.

Com o seu uso temos algumas vantagens, como por exemplo o aumento da legibilidade, confiabilidade e facilidade na detecção de erros no sistema em tempo de compilação.

Figura 2.2 – Demonstração de uso do TypeScript

```
interface User = {
 name: string,
 email: string,
 address: {
   city: string,
   state?: string
 }

}

function showWelcomeMessage(user: User) {
 return `Welcome ${user.name}, your e-mail is ${user.email}. Your city is ${user.address.city}
    and your state is ${user.address.state}`;
}

showWelcomeMessage({
  name: 'Andressa',
  email: 'andressa@soft.com',
  address: {
   city: 'Porto Alegre',
   state: 'RS'
  }
})
```

Fonte: (TYPESCRIPTLANG.ORG, 2023)

Na Figura 2.2 é possível visualizar a utilização do TypeScript. No início, defini-se uma interface, a "User", que representa um tipo criado pelo desenvolvedor. Em seguida, têm-se a definição de uma função que espera receber um parâmetro que possua a mesma tipografia criada e retorna uma mensagem com os dados recebidos. Ao final, representa-se a chamada dessa função, nessa passa-se os dados solicitados na forma de um objeto do tipo "User". Caso insira-se um dado não compatível com o esperado, é disparado um erro em tempo de compilação para o programador, fato este que não ocorre com o JavaScript, acontece apenas quando realizada a tentativa de execução do código.

2.2.3 React

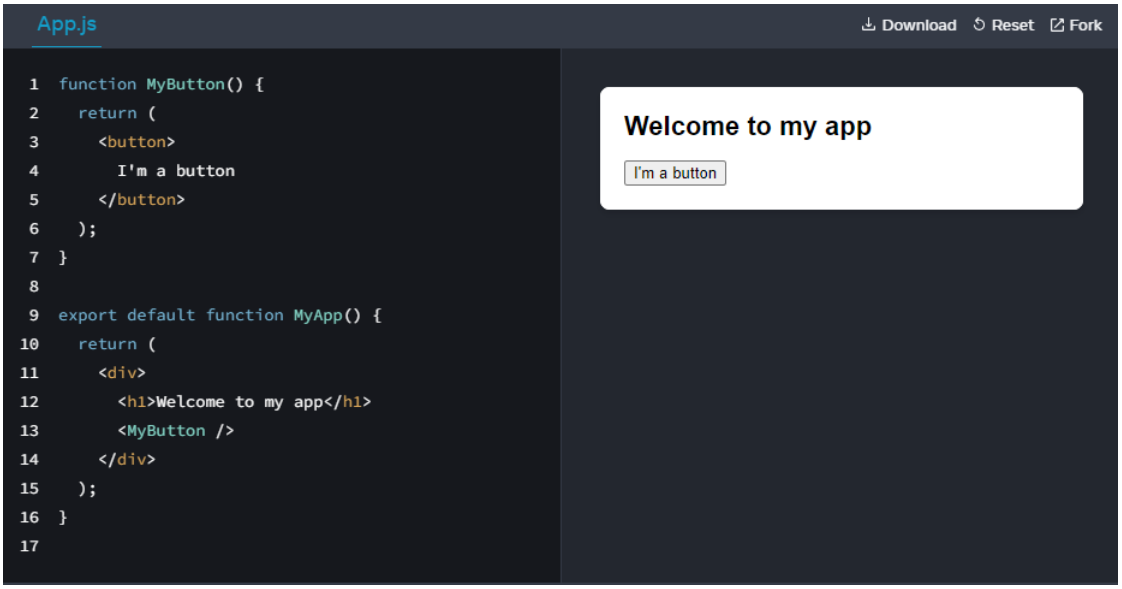
O React[5] (NEVES, 2023) é uma biblioteca baseada no JavaScipt criada pelo Facebook para lidar com situações internas da empresa. Trata-se de um projeto de código aberto, ou seja, significa que qualquer pessoa pode utilizar da tecnologia em seus projetos e também contribuir para a evolução dela.

[5] <https://react.dev/learn>

Dentre as suas maiores características, é válido citar a flexibilidade, escalabilidade e a comunidade ativa, resultando a uma ampla gama de conteúdo de estudo. A ferramenta trabalha com o conceito de componentes reutilizáveis, ou seja, é possível criar uma peça isolada de código a fim de utilizá-la em diferentes contextos de um sistema, descomplicando a construção de uma interface.

Além disso, é possível escrever códigos JavaScript, utilizando a sintaxe JSX, ou TypeScript, com o TSX. Ambas necessitam de conhecimento prévio nas linguagens base da programação WEB, sendo elas: HTML, CSS e JavaScript.

Figura 2.3 – Demonstração de uso do React



Fonte: (REACT.DEV, 2023)

Observa-se na Figura 2.3 um código React no padrão JSX e o seu retorno ao lado. Da linha 1 a 7 é feita a criação do componente "MyButton", que representa um botão utilizado no sistema e, internamente, pode-se visualizar a utilização do HTML para esta criação. Das linhas 9 a 16, encontra-se o componente exportado do arquivo. Dentro, nota-se que utilizou "MyButton"como uma parte deste, representando o comportamento de componentização.

2.2.4 React Native

O React Native[6] (CUNHA, 2023) é uma biblioteca utilizada no desenvolvimento de aplicações móveis. Este é baseado no React, que por sua vez, utiliza o Javascipt, facilitando o processo de aprendizado devido à linguagem difundida e utilizada.

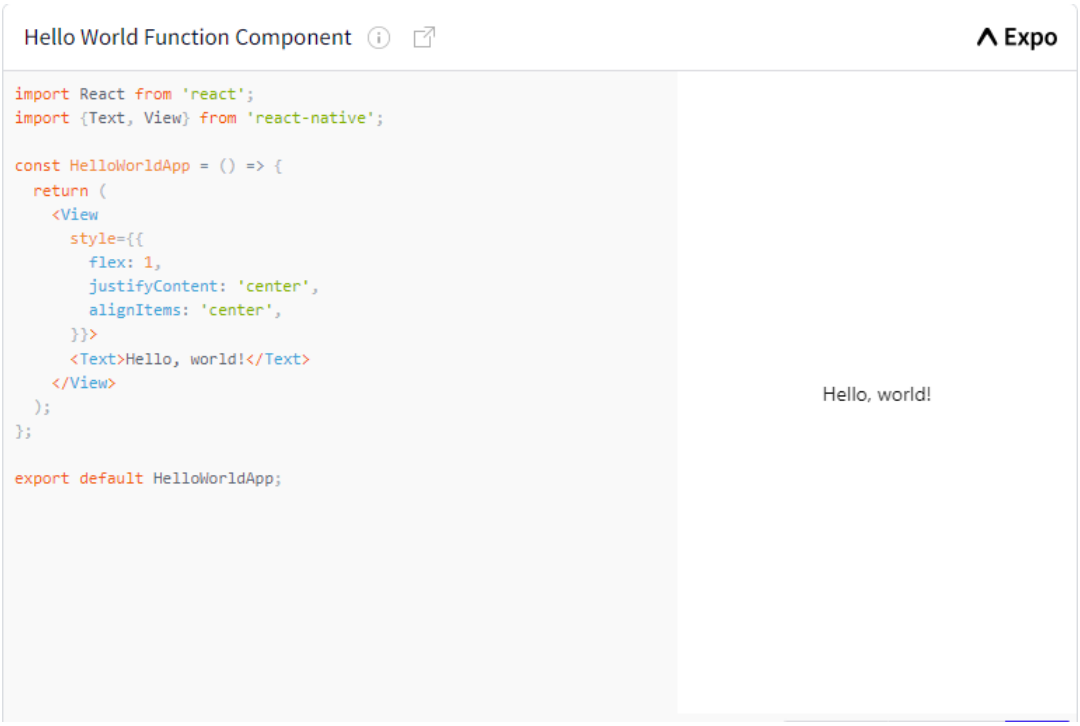
[6] <https://reactnative.dev/docs/getting-started>

Uma das principais características é tratar-se de uma programação *mobile* híbrida, ou seja, um único código funciona em diferentes plataformas, otimizando o processo por necessitar de menos esforço e recursos para a criação um aplicativo compatível com Andoid e iOS. Além disso, permite que o usuário utilize de componentes nativos para o desenvolvimento, podendo ter acesso a recursos como câmera, GPS e outros.

Lançada em 2015 pelo Facebook como um projeto de código aberto, desde então ganhou uma popularidade e é utilizada por diversas empresas como o Discord, Instagram e Facebook.

Existem duas abordagens de uso, a criação de componentes de função e a criação de componentes de classe.

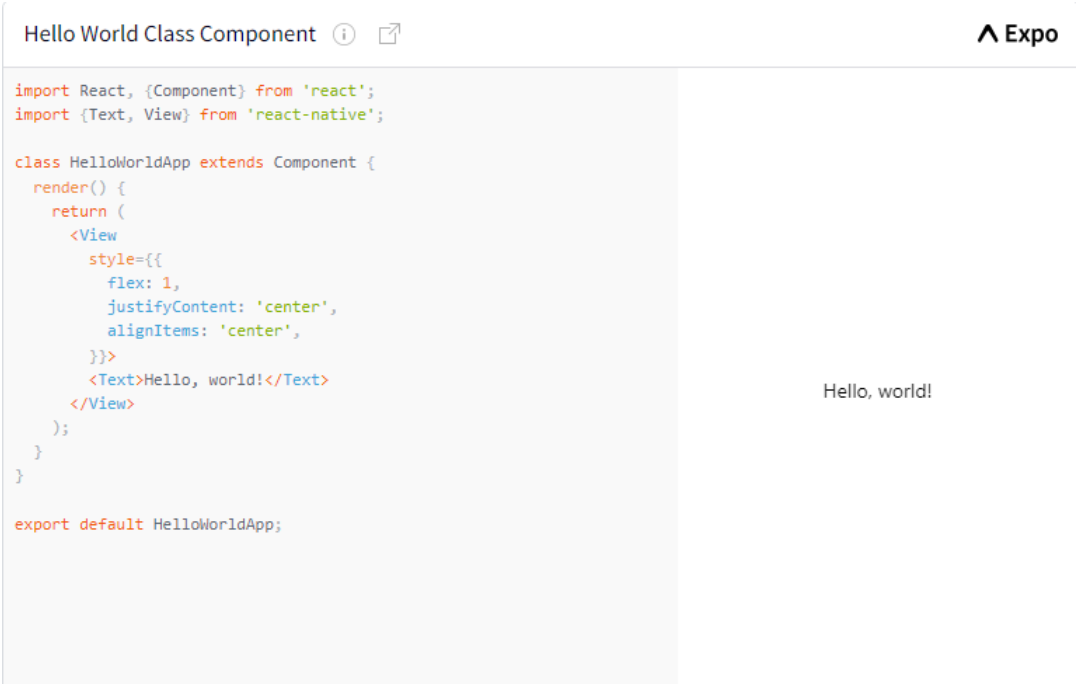
Figura 2.4 – Demonstração de uso do React Native - componentes por função



Fonte: (REACTNATIVE.DEV, 2023)

É apresentado na Figura 2.4 um exemplo de utilização do React Native com a abordagem de componentes por função. Seguindo de cima para baixo expõe-se a importação das dependências necessárias, logo após a definição da função e, dentro dela, o seu retorno e a exportação.

Figura 2.5 – Demonstração de uso do React Native - componentes por classe



Fonte: (REACTNATIVE.DEV, 2023)

Observa-se, na Figura 2.5, um exemplo de utilização do React Native com a abordagem de componentes por classe. A sua estruturação segue um padrão semelhante a de componentes por função, possuindo alterações na definição da classe em comparação a definição da função.

2.2.5 Expo

O Expo[7] (FERNANDES, 2018) é uma ferramenta utilizada no desenvolvimento de aplicativos móveis em conjunto ao React Native e permite um fácil acesso a recursos nativos do dispositivo sem necessitar de alterações significativas de código. Diversos detalhes de configuração e compilação são tratados pela ferramenta, possibilitando que os desenvolvedores se concentrem na criação lógica do sistema.

Como benefício da utilização deste, é valido citar o descarte da necessidade de uso de um MacBook para a depuração da aplicação iOS. Com o expo, torna-se possível a criação de um ambiente capaz de conectar um dispositivo físico ou um emulador pelo aplicativo Expo Go. Basta escanear um "QR Code", visualizar e testar aquilo desenvolvido.

Além disso, existe uma série de recursos oferecidos pela plataforma, como por exemplo a integração com serviços de notificação, autenticação, análise, entre outros.

[7] <https://docs.expo.dev/>

2.3 Desenvolvimento *back-end*

Trata-se, como o próprio nome sugere, daquilo por trás de um sistema. Por ele, obtêm-se as informações apresentadas pelo *front-end* em aplicações não estáticas, ou seja, os dados não são fixos. Um exemplo é o Facebook, as postagens são feitas por usuários, pode-se editar o perfil, dentre outras funcionalidades possíveis devido à existência de um *back-end* que armazena os dados, trata e retorna para o usuário.

2.3.1 .NET

.NET[8] (MICROSOFT, 2023b) trata-se de uma plataforma de desenvolvimento de software criada pela Microsoft. Ela fornece um ambiente para construir, implantar e executar aplicativos em diversos sistemas operacionais e arquiteturas, permitindo o desenvolvimento de soluções para diferentes plataformas, como Windows, macOS e Linux.

Por se tratar de uma plataforma de código aberto, existe um amplo e crescente número de bibliotecas e ferramentas disponíveis que auxiliam no dia a dia do desenvolvimento. Nessa, é possível desenvolver em diferentes linguagens de programação, como por exemplo o C Sharp, o Visual Basic.NET, o F Sharp entre outras.

2.3.2 C Sharp

C Sharp[9] (MICROSOFT, 2023c) refere-se a uma linguagem de programação orientada a objetos desenvolvida pela Microsoft. Essa existe desde 2000 como parte da plataforma .NET Framework e tornou-se uma das linguagens mais populares de uso no *back-end*. A mesma, possui capacidade de lidar com uma ampla gama de cenários de desenvolvimento e sua sintaxe é semelhante a outras linguagens como C, C++ e Java, o que facilita a curva de aprendizado.

A linguagem teve uma ampla evolução desde a sua origem, adicionando recursos para suportar novas cargas de trabalho e práticas emergentes de design de software. Pode-se definir os tipos e comportamentos de cada objeto do sistema. Nesta, existem algumas características principais, sendo elas a tipagem segura, basicamente significa que verificações de tipo dos objetos e variáveis são feitas durante a compilação, garantindo uma maior confiabilidade ao código, o coletor de lixo, faz o gerenciamento de alocação e liberação de memória do sistema de forma

[8] <https://learn.microsoft.com/pt-br/dotnet/>
[9] <https://learn.microsoft.com/en-us/dotnet/csharp/>

automática e a integração com o .NET, possibilitando o acesso a uma ampla gama de bibliotecas e ferramentas de desenvolvimento..

Figura 2.6 – Demonstração de uso do *C Sharp*

Fonte: (MICROSOFT, 2023c)

É representado na Figura 2.6 o uso da linguagem. Nela visualiza-se a definição de uma classe denominada "Program", detentora de um método "Main()", um ponto de entrada do programa. Dentro deste, existe outro método denominado "Console.WriteLine()" que exibe a mensagem "Hello, World" no console da aplicação.

2.4 Banco de dados

Um banco de dados (ORACLE, 2021) consiste em um sistema projetado para armazenar, gerenciar e recuperar informações de forma eficiente, é amplamente utilizado em várias aplicações, desde aplicativos de *desktop* até sistemas empresariais e serviços WEB.

Existem diferentes estruturas e tipos de bancos, dentre estes, é válido citar os dois principais utilizados:

- Banco de Dados Relacional: Organiza-se os dados em tabelas e as relações entre elas é estabelecida por meio de chaves primárias e estrangeiras. Podem ser manipulados e consultados utilizando a linguagem SQL;
- Banco de Dados Não-Relacional: Projetados para lidar com dados não-estruturados ou semiestruturados. Por não possuirem uma estrutura padrão, aumenta-se a flexibilidade e escalabilidade.

O banco relacional deve ser utilizado quando os dados que deseja-se armazenar possuam uma estrutura definida, imutável e relacionamentos complexos entre eles. Dentre suas principais vantagens estão a integridade e consistência nos dados e a flexibilidade nas consultas. Quanto

as desvantagens, é valido citar a baixa escalabilidade e alto custo. Quanto aos bancos não relacionais, devem ser utilizados quando os dados não possuem uma estrutura bem definida e deseja-se uma alta escalabilidade. As vantagens que destacam-se são a escalabilidade e o desempenho. Já as desvantagens estão a baixa consistência nos dados e a quantidade reduzida de recursos para consulta.

2.4.1 SQL

O *Structured Query Language* (SQL) (ORACLE, 2021), linguagem utilizada para gerenciar e manipular bancos de dados relacionais, fornece um conjunto de comandos e instruções para criar, consultar, alterar e excluir dados. Padronizada e amplamente utilizada em sistemas de gerenciamento de banco de dados relacional (SGBD) como o MySQL e SQL Server.

2.4.2 SQL Server

O SQL Server[10] consiste em um SGBD desenvolvido pela Sysbase em parceria com a Microsoft. Essa parceria durou até 1994 e hoje a manutenção e evolução é feita somente pela segunda.

Trata-se de uma das opções mais populares e baratas utilizadas atualmente e fornece um conjunto abrangente de recursos para o gerenciamento dos dados, segurança, desempenho e escalabilidade. Sua popularidade está atribuída, além de seus recursos, ao suporte da Microsoft.

2.4.3 Cosmos DB

O banco de dados Cosmos DB[11] (MICROSOFT, 2023a) refere-se a um serviço oferecido pela Microsoft Azure. Oferece tempo de resposta de milissegundos de um dígito, escalabilidade automática e instantânea e velocidade garantida em qualquer escala, permitindo que aplicativos sejam executados de forma rápida e eficiente em diferentes regiões geográficas.

Trata-se de um banco de dados *Not Only SQL* (NoSQL) que oferece suporte a vários modelos de dados. Como sua estrutura não é necessáriamente engessada, como por exemplo o SQL Server, podem haver dados de diferentes formatos em uma mesma área. No caso do presente relatório de estágio, utilizou no intuito de criar uma funcionalidade de notificações

[10] <https://learn.microsoft.com/pt-br/sql/sql-server>
[11] <https://learn.microsoft.com/pt-br/azure/cosmos-db/introduction>

devido à estrutura de cada uma não ser fixa, podendo assim, armazenar esses dados sem maiores problemas de compatibilidade.

2.5 Teste unitário

Estes testes (SILVA, 2021b) devem ser feitos em um nível mais baixo do projeto, realizados pelos próprios programadores envolvidos e possuem, como objetivo, assegurar a qualidade de cada unidade de forma individual. As unidades em questão tratam-se das menores partes do código, como funções, métodos ou classes.

Além de garantir a qualidade do código, os testes unitários também oferecem outros benefícios, como ajudar a documentar o comportamento das unidades de código, facilitando a compreensão e o trabalho em equipe, além de fornecer um feedback ágil aos desenvolvedores, permitindo a detecção precoce de problemas e a agilidade na resolução.

2.5.1 Jest

O Jest[12] (ESTEVAO, 2020) consiste em uma ferramenta criada pelo Facebook, inicialmente utilizado para testes unitários e de integração no React. Com sua evolução, passou a ser utilizado em diversas plataformas JavaScript, como Node, Redux, TypeScript entre outras. Trata-se de um framework de código aberto e amplamente utilizado, apresenta uma sintaxe compreensível e a curva de aprendizado é pequena.

Com o seu uso, é possível obter alguns relatórios de cobertura de código e, além disso, tornar o sistema mais robusto e confiável.

Figura 2.7 – Demonstração de uso do Jest

```
25  it('Click on answer form button - Button should be functional and callback function should be called', () => {
26      const { getByText } = render(
27          <PaperProvider>
28              <SkillFormScreenContent {...props} />
29          </PaperProvider>,
30      );
31      const button = getByText(i18n.t('ANSWER_FORM'));
32
33      fireEvent.press(button);
34
35      expect(props.onPressFormButton).toBeCalledTimes(1);
36  });
37
```

Fonte: Arquivo pessoal

Na Figura 2.7 pode-se visualizar um exemplo de uso desta. Na linha 25 têm-se a criação do teste, inicia-se com "it", uma descrição daquilo testado e uma *iron function* que dá inicio ao

[12] <https://jestjs.io/docs/getting-started>

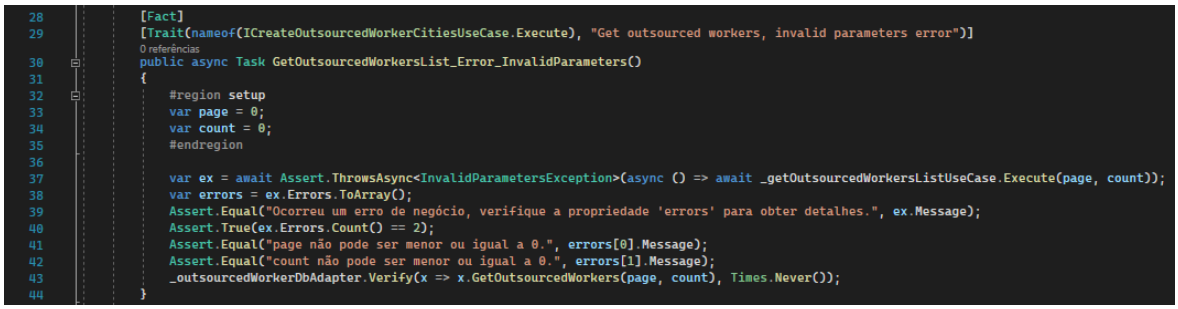
código em si. Na linha 26 é feita a renderização do componente testado, utiliza-se o "render" para tal feito e, dentro das chaves é definido o que deseja-se utilizar do *framework*, no caso, o "getByText". A linha 31 representa a obtenção do botão existente na tela pelo seu texto, na 33 este é pressionado e na 35 espera-se que seja feita a chamada da função denominada "onPressFormButton", passada como propriedade no momento da renderização do componente da linha 26.

2.5.2 xUnit

O xUnit[13] (SILVA, 2021a) é uma biblioteca de código aberto utilizada na realização de testes unitários para o .NET. Esta surgiu a partir do NUnit, já que os criadores buscavam um recurso mais amplo que o criado. O seu objetivo principal é fornecer uma estrutura padronizada para escrever e executar testes automatizados, segue uma abordagem orientada a objetos e fornece uma série de convenções e métodos para a criação de testes unitários e de integração.

Dentre suas características principais, estão a utilização de métodos chamados "asserts" para verificar os resultados dos testes e os relatórios gerados após a execução, esses são bem detalhados e incluem informações sobre falhas, erros, cobertura de código e os casos que passaram.

Figura 2.8 – Demonstração de uso do xUnit



Fonte: Arquivo pessoal

Observa-se na Figura 2.8 a apresentação de um exemplo do uso do xUnit em um contexto prático. Nas linhas 28 e 29 têm-se o inicio do teste definido pelo "[Fact]" e a definição do "Trait" desse, que basicamente é uma forma de tornar mais fácil a identificação e rastreamento. Já a linha 30, é a definição do método do teste, nesse caso, utilizou-se uma nomenclatura padronizada do projeto, a mesma segue com o nome do método a ser testado, um identificador de erro ou sucesso e o tipo de exceção disparada se for o caso. Entre as linhas 32 e 35 há uma

[13] <https://xunit.net/>

região definida como *setup*, nela são criadas variáveis, *mocks* entre outros necessários para a execução. Da linha 37 a 43 estão os usos dos "Asserts" para verificar os resultados do teste. Na 37 analisa-se se o método testado vai disparar um erro do tipo "InvalidParametersException" ao ser executado com os parametros definidos; A 38 recupera os erros disparados e os transforma em um *array*; As demais linhas visam analisar se dispararam corretamente e se métodos posteriores não executaram.

2.6 Metodologias Ágeis

Metodologias ágeis (GOVERNO, 2021) são abordagens de gerenciamento de projetos e desenvolvimento de *software* que enfatizam a flexibilidade, a colaboração e a adaptação contínua, ou seja, têm-se um estreitamento na conexão com o cliente, possibilitando mudanças no contexto do projeto. Além disso, as entregas são feitas em partes ao invés do todo, podendo assim quebrar o sistema em contextos e incrementa-lo quando possível, enquanto os métodos tradicionais são feitos de forma procedural, mais rígidos e menos adaptáveis as mudanças nas necessidades dos clientes. Baseiam-se em um conjunto de princípios e valores descritos no "Manifesto Ágil" (BRASILEIRO, 2018), uma declaração criada em fevereiro de 2001, objetivando guiar as melhores práticas no ramo.

Dentre os princípios e valores estão:

- Indivíduos e interações em vez de processos e ferramentas: valoriza-se a comunicação e a colaboração entre as pessoas envolvidas no projeto;
- Software funcionando em vez de documentação abrangente: o foco está na entrega de software de valor para os clientes em intervalos curtos de tempo, em vez de se concentrar na criação de uma documentação extensa;
- Colaboração com o cliente em vez de negociação de contratos: busca-se uma colaboração estreita e contínua com o cliente para entender suas necessidades em constante mudança e adaptar o produto em conformidade;
- Responder a mudanças em vez de seguir um plano rígido: as metodologias ágeis reconhecem que os requisitos e as circunstâncias podem mudar ao longo do projeto, e a flexibilidade é valorizada para se adaptar a essas mudanças.

2.6.1 Framework Scrum

O Scrum[14] (AMAZON, 2021) é um framework que representa uma das formas mais populares de aplicar o desenvolvimento ágil em um projeto. Objetiva auxiliar na gestão e no desenvolvimento de projetos que tenham um curto prazo de entrega.

O trabalho é dividido em iterações curtas e de tempo predefinido, chamadas de "*sprints*". Cada uma geralmente tem uma duração de duas a quatro semanas. Nesse tempo, a equipe concentra-se em um escopo definido de tarefas, denominadas "*sprint backlog*", devem ser realizadas e finalizadas dentro deste tempo para que possam se tornar um incremento no sistema.

A equipe é composta por diferentes papéis. O "*Product Owner*" (PO), responsável por definir as prioridades do projeto e pelas funcionalidades a serem desenvolvidas. O "*Scrum Master*" (SM), facilitador do processo Scrum, ajuda a equipe a entender e implementar os princípios do Scrum e a equipe de desenvolvimento, composta por profissionais que realizam o trabalho de criar o código produto.

Existem vários ritos (ou reuniões) que ajudam a equipe a colaborar e manter um bom fluxo de trabalho. Os principais são:

- *Planning*: Reunião realizada no início de cada ciclo, o PO e a equipe de desenvolvimento definem quais itens do *backlog* serão incluídos no *sprint*. Eles discutem os objetivos, prioridades e a definição de conclusão das tarefas abordadas;

- *Daily*: Reunião diária curta, geralmente de 15 minutos; A equipe encontra-se para sincronizar o progresso do trabalho. Cada membro compartilha o feito desde a última reunião, o planejado para fazer até a próxima reunião e quaisquer obstáculos que esteja enfrentando;

- *Review*: Reunião realizada ao final de cada *sprint*, a equipe demonstra o trabalho concluído ao PO e as outras partes interessadas. Eles revisam as funcionalidades desenvolvidas durante o ciclo e coletam *feedback*. Com base nesse, pode-se ajustar as prioridades e a direção do projeto;

- *Retrospective*: Reunião realizada após a *review* da *sprint*, a equipe reflete sobre o ocorrido na anterior. Eles analisam o que funcionou bem, o que pode melhorar e identificam ações para implementar melhorias na próxima. A retrospectiva é uma oportunidade para a equipe aprender e adaptar-se.

[14] <https://www.scrum.org/resources/scrum-guide>

2.7 Plataforma Azure

A Azure[15] (GARRETT, 2020) refere-se a uma plataforma da Microsoft que possui mais de 200 produtos e serviços de nuvem, permitindo que indivíduos e organizações construam, implantem e gerenciem suas aplicações em uma escala global. Ou seja, ele proporciona toda a infraestrutura necessária para que um *software* seja gerenciado, através do *backlog*, controle de *sprints*, métricas e outras funcionalidades, além disso, também possibilita a hospedagem em um servidor.

Dentre as funcionalidades da plataforma, é válido citar algumas das principais utilizadas no decorrer do processo de estágio em questão:

- Computação em nuvem: Recursos para hospedar e executar aplicativos em máquinas virtuais (VMs), contêineres, funções sem servidor e outras opções;
- Armazenamento: Variedade de serviços de armazenamento para atender às necessidades de diferentes tipos de dados, como armazenamento de arquivos, bancos de dados, armazenamento em bloco, dentre outros;
- Banco de Dados: O Azure possui vários serviços de banco de dados, como o Azure SQL Database, Azure Cosmos DB, Azure Database for MySQL, Azure Database for PostgreSQL e outros que fornecem soluções escaláveis e gerenciadas para armazenar e acessar dados;
- Redes: Permite criar redes virtuais privadas, conectar redes locais à nuvem, implementar balanceadores de carga, serviços de DNS e outros recursos;
- DevOps: O Azure oferece ferramentas e serviços para suportar práticas de desenvolvimento e operações ágeis (DevOps), como integração contínua, entrega contínua, automação de infraestrutura, dentre outros.

2.8 Ritos internos da empresa

2.8.1 Microsoft Teams

O Microsoft Teams[16] (KOVACS, 2021) é a plataforma de comunicação utilizada. Toda a interação entre os colaboladores acontece nesta, em conversas por *chat* ou por voz. Possui

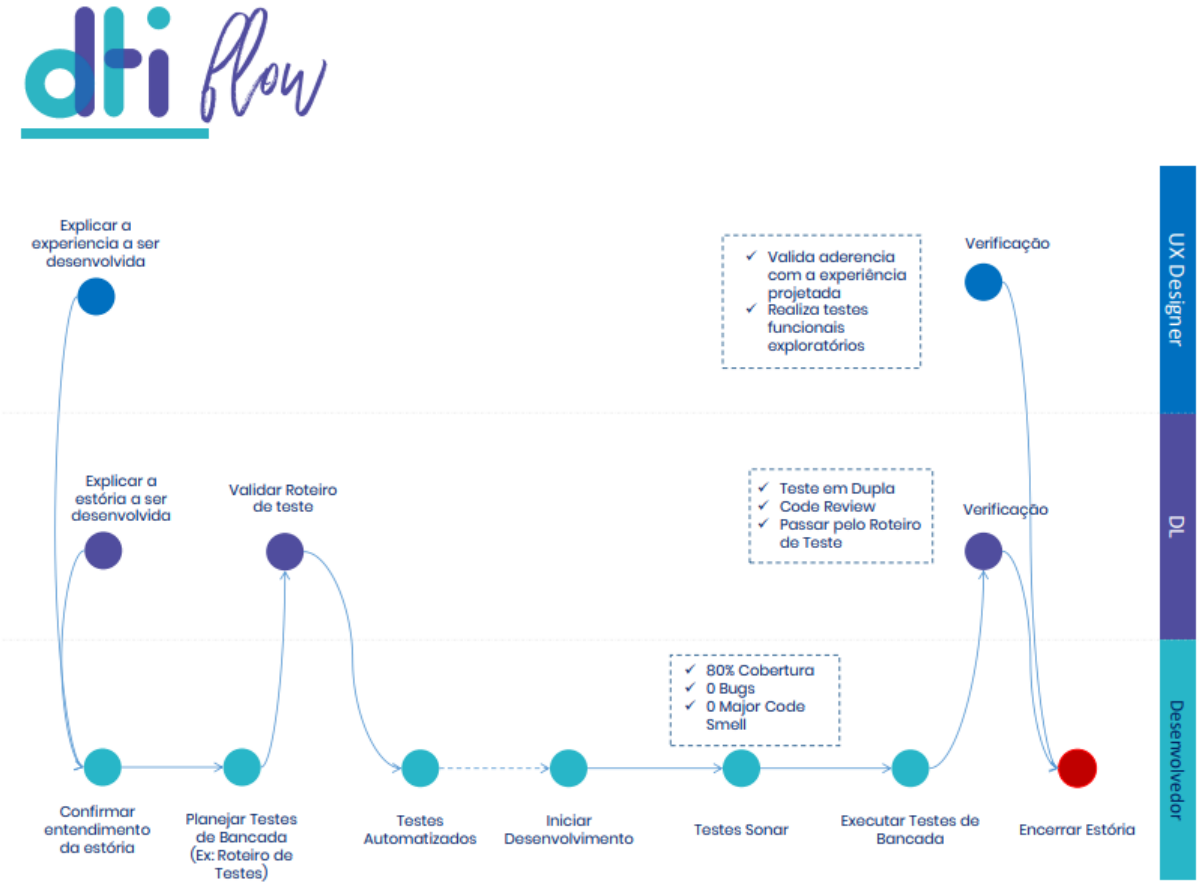
[15] <https://learn.microsoft.com/en-us/azure/?product=popular>
[16] <https://www.microsoft.com/pt-br/microsoft-teams/>

uma ampla gama de ferramentas que possibilitam a organização das partes da empresa, compartilhando informações de forma eficiente e independente da localização física individual.

2.8.2 DTI Flow

O DTI Flow trata-se de um fluxo de trabalho predefinido que rege o processo de desenvolvimento de uma tarefa dentro das equipes, este não é obrigatório ser utilizado pelos colaboradores, normalmente cada equipe organiza-se e adapta-o para um formato que condiz com a sua realidade.

Figura 2.9 – Representação Dti Flow



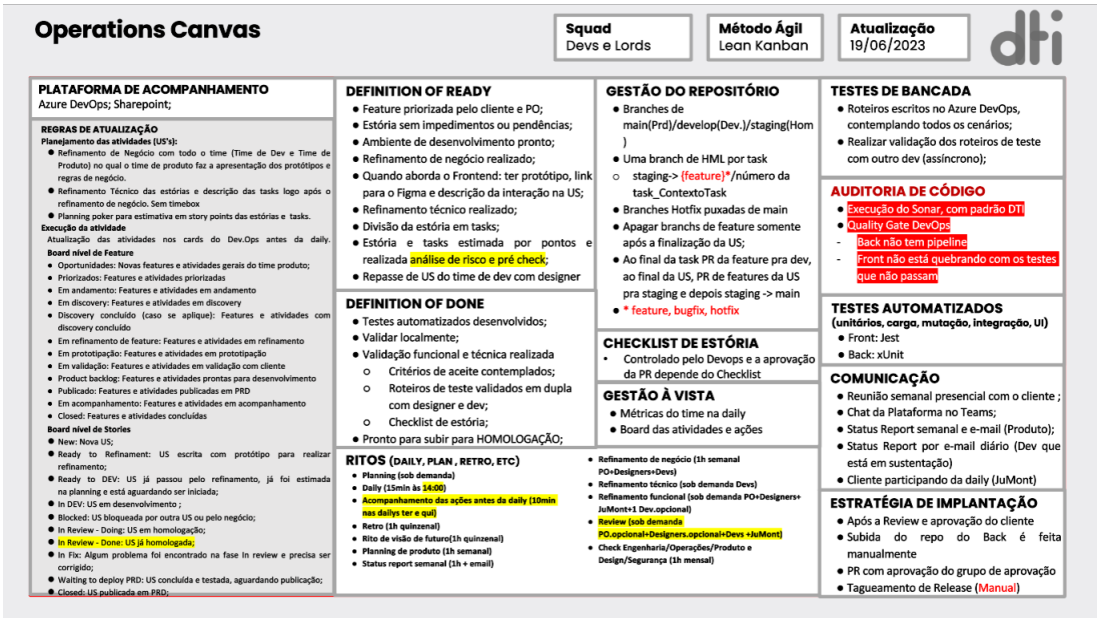
Fonte: Arquivo pessoal

É representado na Figura 2.9 o Dti Flow. Neste, estão definidos os passos que um(a) desenvolvedor(a) deve seguir para entregar sua atividade com um determinado padrão de qualidade. O fluxo de desenvolvimento inicia-se no processo de validação de entendimento da tarefa e criação de um roteiro de testes, seguido da codificação, criação dos testes unitários, utilização de ferramentas de análise estática de código e, ao final, o processo de validação.

2.8.3 Operation Canvas

O Operation Canvas refere-se a um documento criado, no nível organizacional de cada equipe, visando descrever o processo operacional dessa. Nele encontram-se informações como a forma como devem ser criadas as *branchs* para desenvolvimento, ritos existentes para o contexto do time, formas de comunicação, quando uma tarefa está pronta para ser desenvolvida, quando ela está finalizada, entre outras. Deve ser atualizado periodicamente com a presença do *squad* e representa um acordo entre os membros sobre a forma que o dia a dia da equipe é guiado.

Figura 2.10 – Representação Operation Canvas



Fonte: Arquivo pessoal

É representado na Figura 2.10 um exemplo de Operation Canvas. Nele é possível visualizar que, por tratar-se de algo interno de cada time, a linguagem não é formal e, além disso, nesse caso em específico, existem marcações no texto em vermelho e amarelo utilizadas para controle próprio. Apresenta toda a organização da operação da equipe, tornando os membros alinhados com a forma como devem tomar prosseguimento operacional e também auxiliando possíveis novos contribuintes a compreenderem melhor o funcionamento.

2.8.4 Ritos da empresa

Nessa subseção são apresentados alguns ritos internos existentes na empresa, esses fazem parte do cotidiano dos colaboradores e visam tratar de processos organizacionais de cada

time, como por exemplo a saúde do produto, qualidade do código desenvolvido e a segurança. Devem ser realizados periodicamente, geralmente uma vez por mês, e fornecem dados importantes de análise. Estes são divididos entre os membros do time considerando-se, a cada mês, uma pessoa como responsável por guiar os encontros e fomentar as discussões. Dentre estes ritos, pode-se citar:

- Check de engenharia: Voltado para o time técnico. Objetiva tratar de questões de engenharia, ou seja, estrutura dos códigos, qualidade, manutenibilidade, uso de testes, configurações e afins;
- Check de segurança: Trata de questões relacionadas a segurança do produto e sua conformidade com a Lei Geral de Proteção de Dados (LGPD);
- Check de produto e design: Objetiva discutir a respeito do valor que o produto gera, como a equipe compreende este e, além disso, validar os processos de *discovery* e usabilidade promovidos pelo *designer* do time;
- Check de operações: Aborda a forma como está a operação do time, discute-se assuntos como a qualidade dos ritos executados, fluxo de desenvolvimento seguido, integração do time e aprendizado dos membros.

2.8.5 Guildas

As guildas tratam-se de organizações voluntárias guiadas pelos próprios colaboradores da empresa com o apoio desta, visam fomentar a troca de conhecimento através da criação de reuniões entre seus membros. Nestas, pode-se abordar assuntos diversos que estejam relacionados ao seu objetivo. A partir dessa definição, é válido citar que, o autor deste, de forma voluntária, assumiu durante o período de estágio como um dos líderes da Guilda Front-End. Esta visa promover encontros que tratam de assuntos diversos que estejam relacionados a essa modalidade de desenvolvimento. Dentre as atividades da liderança estão a organização, promoção e criação do encontros. As reuniões são realizadas bimensalmente e são abertas para todos os funcionários que interessarem pelo tópico a ser tratado.

3 ATIVIDADES DESENVOLVIDAS

Neste capítulo, apresentam-se exemplos das atividades realizadas pelo autor durante o estágio, abrangendo a contribuição em projetos das empresas Ducash e Suppo7. O primeiro, da Ducash, a alocação inicial, fez parte do treinamento básico fornecido pela empresa, e, por isso, é apresentado de forma mais resumida. A partir dele adquiriu-se conhecimentos no desenvolvimento *mobile*, *back-end* e, além disso, questões operacionais da empresa e do dia a dia de trabalho de um desenvolvedor. O segundo projeto, por ter ocorrido em um período de maior maturidade, representa um passo primordial para o desenvolvimento de proatividade, aprimoramento dos conhecimentos a partir de desafios e fomento de liderança. Neste projeto, da Suppo7, o estagiário teve a oportunidade assumir a frente de alguns ritos da empresa, fato que estimulou o interesse pela parte organizacional e gerencial de um time. Concomitantemente a isso, melhorou-se conhecimentos técnicos nas *stacks* já trabalhadas anteriormente e iniciada a jornada no desenvolvimento *front-end WEB*.

3.1 Projeto da Ducash

O projeto da Ducash objetivou a criação de uma aplicação *mobile* tanto iOS quanto Android. Trata-se de um clube de benefícios que fornece descontos em diversos estabelecimentos e também possui um processo de gamificação onde, conforme o usuário utiliza a sua conta, ganha algumas moedas que pode trocar em cupons em marcas como a Domino's, Drogaria Araujo, Hermes Pardini e diversos outros. Neste atuou-se como um desenvolvedor *full-stack*, ou seja, no desenvolvimento de novas telas, alterações das já existentes, correções de *bugs* no *front-end* e, além disso, na criação, correção e alteração de rotas no *back-end*. O *squad* contava com a presença de 1 *Product Owner*, 1 *Designer*, 1 *Scrum Master* e 4 desenvolvedores em seu inicio e, conforme o projeto foi chegando ao fim, o time recebeu reduções estratégicas. Cita-se, ao longo dessa seção, algumas atividades trabalhadas.

3.1.1 Desenvolvimento do cadastro do aplicativo

O cadastro possibilita que pessoas associadas a algum plano funerário do Grupo Zelo, ou seus parceiros de *co-branding*, participem da plataforma. Nele, solicita-se dados como o nome completo, cidade do usuário, sua inscrição, seus dependentes, email, CPF, entre outros.

A partir dos dados coletados, realiza-se validações diretamente na interface gráfica e, posteriormente, consultas ao *back-end* que visam identificar a aptidão para a utilização da plataforma. O principal dado para essa verificação é o número da inscrição do plano, com ele é consultado o banco de dados do contribuinte, seja este o Grupo Zelo ou outro, para saber a veracidade dos dados. Caso seja retornado um sucesso, o usuário é criado no banco de dados relacional e passa a ter acesso as demais funcionalidades.

Além do *front-end* e do *back-end*, desenvolveu-se testes unitários e de mutação para que o projeto possuisse uma boa cobertura de código, tornando o fluxo mais confiável. Nesse, houve uma dificuldade na criação dos testes devido à arquitetura legada do projeto *front-end*, tornando difícil *mockar* os dados e muitas vezes inviável pelo tempo incrementado. Posteriormente, realizou-se uma refatoração com a finalidade de melhorá-la.

3.1.2 Atualizações de layout na tela principal do aplicativo

Durante o desenvolvimento do projeto, alterou-se a tela inicial do aplicativo três vezes com o intuito de melhorar a usabilidade e a experiência dos usuários. Nesta, na parte superior da tela, exibia-se *banners* que mostravam promoções, funcionalidades novas e descontos da plataforma. Na parte inferior, um carrossel apresentava as principais empresas parceiras e outro as mais próximas da localização do usuário.

Trazia-se esses dados do banco da aplicação por meio da integração com rotas criadas no *back-end* desta e, caso o usuário clique em alguma das três funcionalidades da tela, redirecionaria-o para o fluxo específico desejado.

3.1.3 Desenvolvimento da tela de exibição dos parceiros próximos

Nesse fluxo disponibiliza-se uma lista dos parceiros mais próximos da localização do usuário, com esta ele consegue ter acesso aos dados de descontos nesses estabelecimentos e as informações como endereço, nome, valores, distância, entre outros.

Além da lista que apresenta diversos cards com as logomarcas e nomes para selecionar, existe também a visualização a partir do mapa. Utilizou-se uma biblioteca do React Native para que, a partir de uma lista de dicionário de latitudes e longitudes, fossem exibidos pins em um mapa que indicavam onde encontram-se os parceiros mais próximos. Estes poderiam ser selecionados e redirecionavam para a tela que exibia os dados do estabelecimento.

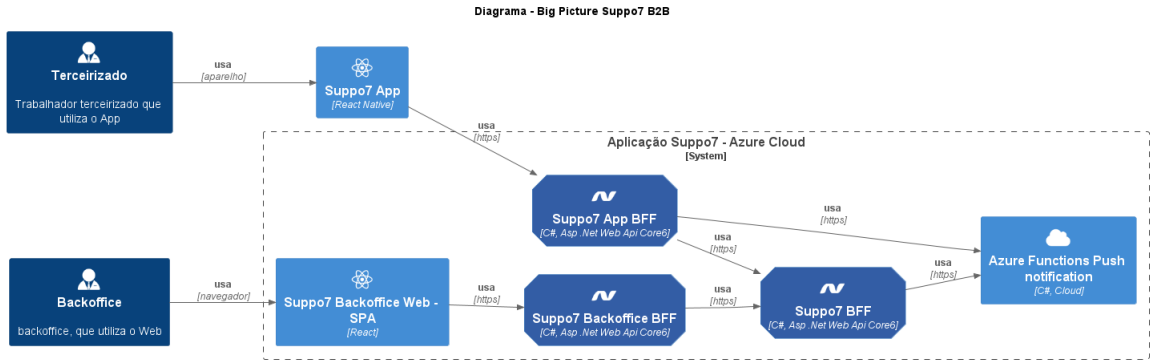
A maior dificuldade deste fluxo, além da criação dos testes unitários devido à arquitetura legada, tratou-se do desenvolvimento de uma rota que obtém todas as localizações e distâncias a partir da posição geográfica do usuário. Esta recebe a latitude e longitude, obtém os mesmos dados dos parceiros e, a partir disso, realiza cálculos euclidianos para obter a distância aproximada entre os dois pontos e ordena-los para a listagem correta de forma crescente na interface gráfica.

3.2 Projeto da Suppo7

O projeto da Suppo7 possui duas vertentes que levam o mesmo nome do modelo de negócio delas. A finalidade de ambas é fornecer um sistema de prestação de serviços tecnológicos. A diferença é o usuário final. Neste, o estagiário atuou no contexto de desenvolvimento, ou seja, no *mobile*, *front-end WEB* e *back-end*, além de assumir a responsabilidade por alguns ritos como check de operações e retrospectivas, guiando-os para buscar sua maior eficiência. O projeto possuia dois times atuando, estruturalmente contavam no início com a presença de 1 *Product Owner*, 1 *Designer* e o time de desenvolvimento, posteriormente foram realizadas alterações que culminaram na unificação dos *squads*, tornando-se um único grande time. Ao decorrer da seção cita-se algumas *user stories* trabalhadas pelo autor, nos quatro diferentes ambitos do projeto.

A primeira destas é o *business to business* (B2B). Trata-se de uma plataforma criada com a finalidade de suprir a necessidade de atendimento de empresas em cidades longe das regiões metropolitanas, nesses casos, não compensa ter um profissional fichado devido às poucas demandas. Atualmente existem dois *softwares* para essa vertente, um aplicativo móvel, para aparelhos Android e iOS, destinado aos prestadores terceirizados que desejam realizar esses atendimentos fornecidos a eles, e um sistema administrativo utilizado por pessoas internas que, atualmente, recebe planilhas com as solicitações de serviço vindas diretamente da Oi, empresa de telecomunicação, e repassa para aqueles que cadastraram-se pelo aplicativo para executar esses agendamentos.

Figura 3.1 – Representação do fluxo do B2B

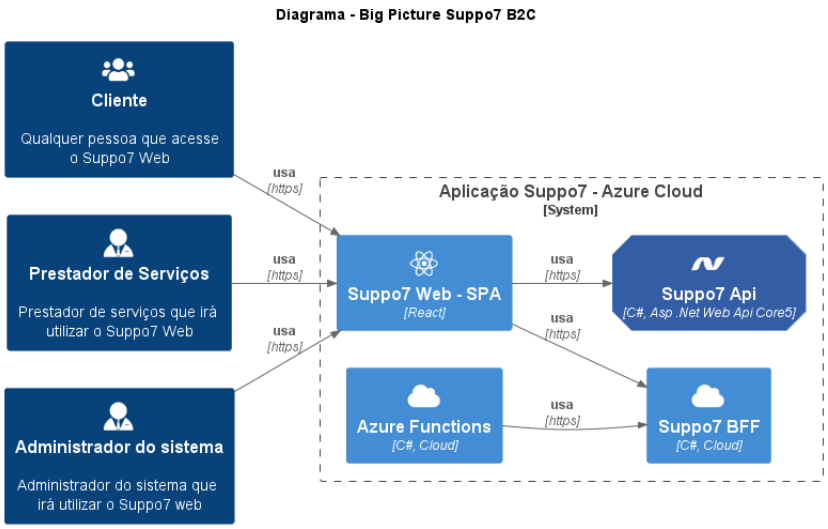


Fonte: Arquivo pessoal

Observa-se na Figura 3.1 a forma como funciona o B2B. Os retângulos a esquerda, em azul escuro, intitulados "Terceirizado" e "Backoffice", representam as personas visadas pela plataforma. Estas, interagem com as interfaces gráficas desenvolvidas, "Suppo7 App" e "Suppo7 Backoffice Web" respectivamente, que, por sua vez, comunicam-se com seus *Back-end for Front-end* (BFF's) que possuem como objetivo apenas realizar a conexão com as demais *Application Programming Interface* (API's) principais e retornar esses dados para o *front-end*. Nesse caso só existe uma principal, a "Suppo7 BFF", que realiza todas as conexões com banco, envio de notificações, tratamento dos dados, entre outros objetivos, sendo assim, o ponto principal que fornece a corretude funcional. A direita estão as "Azure Functions Push Notification", trata-se de um conjunto de funções que não são chamadas com frequência mas possuem uma determinada periodicidade para executarem de tempo em tempo. Ou seja, são programadas para realizar chamadas a determinadas funcionalidades da API principal, sem a necessidade de uma execução manual, cumprindo com alguma finalidade.

A outra vertente é o *business to costumer* (B2C). Trata-se de um sistema criado com a finalidade de suprir a necessidade de consumidores finais dos serviços, ou seja, clientes podem se cadastrar e realizar solicitações de serviços pela plataforma e, da mesma forma, prestadores de serviço também podem acessá-la para atenderem os primeiros. Além disso, existe também um terceiro tipo de usuário, o administrador do sistema que consegue fazer o controle do fluxo de atendimentos e de prestadores cadastrados e aptos para atuar.

Figura 3.2 – Representação do fluxo do B2C



Fonte: Arquivo pessoal

A Figura 3.2 é uma representação de como funciona o fluxo no B2C. Os retângulos a esquerda, nomeados como "Cliente", "Prestador de Serviços" e "Administrador" do sistema representam as personas visadas pela plataforma. Dentro do quadrado pontilhado encontram-se os serviços utilizados. Os usuários acessam o sistema por meio do *front-end* da plataforma, denominado "Suppo7 Web - SPA". Este, por sua vez, comunica-se com as API's, "Suppo7 Api" e "Suppo7 BFF" e retorna para o primeiro as informações solicitadas. Além disso, existem também as "Azure Functions" que conectam-se com um dos *back-end* periodicamente para realizar determinadas tarefas.

3.2.1 Desenvolvimento do histórico de atendimentos aplicativo B2B

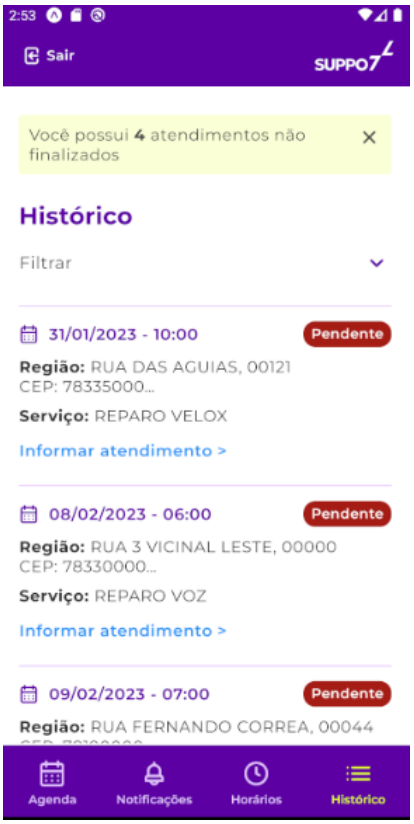
O desenvolvimento do histórico de atendimentos fornece ao prestador B2B a funcionalidade de visualizar os agendamentos aceitos por ele em suas diferentes etapas de ciclo de vida, sendo elas: "Pendente", "Finalizado" e "Cancelado". Criou-se a tela no aplicativo, acessada pelo menu inferior e, além disso, desenvolveu-se duas rotas, uma que busca todos os atendimentos relacionados ao prestador que a utiliza e outra que retorna a quantidade destes que precisam ser finalizados. Com a finalidade de melhorar a experiência do usuário, propôs-se uma ordenação que segue as seguinte regras:

- Os apresentados no topo referem-se aos agendamentos "Pendentes", seguidos dos "Finalizados" e posteriormente dos "Cancelados";

- Os "Pendentes" são ordenados do mais antigo para o mais atual, de forma que o prestador consiga visualizar melhor as necessidades de finalização. Os demais status do mais atual para o mais antigo.

Além dessa medida de usabilidade, existe um modal que informa ao usuário caso possua algum atendimento pendente de finalização, estes são aqueles na etapa de "Pendente" mas a data do agendamento expirou. Além do exposto, há um filtro que permite a exibição dos dados da tela de acordo com a necessidade do prestador.

Figura 3.3 – Representação da tela de histórico do B2B



Fonte: Arquivo pessoal

A Figura 3.3 trata-se de uma representação da forma como a tela é exibida para o prestador e suas funcionalidades existentes. Ressalta-se, naqueles atendimentos marcados como "Pendente", caso a data esteja no passado, é disponibilizado um link direto para o fluxo de finalização deste, acessado pelo botão de "Informar atendimento".

Figura 3.4 – Representação da tela de histórico filtrado do B2B

Fonte: Arquivo pessoal

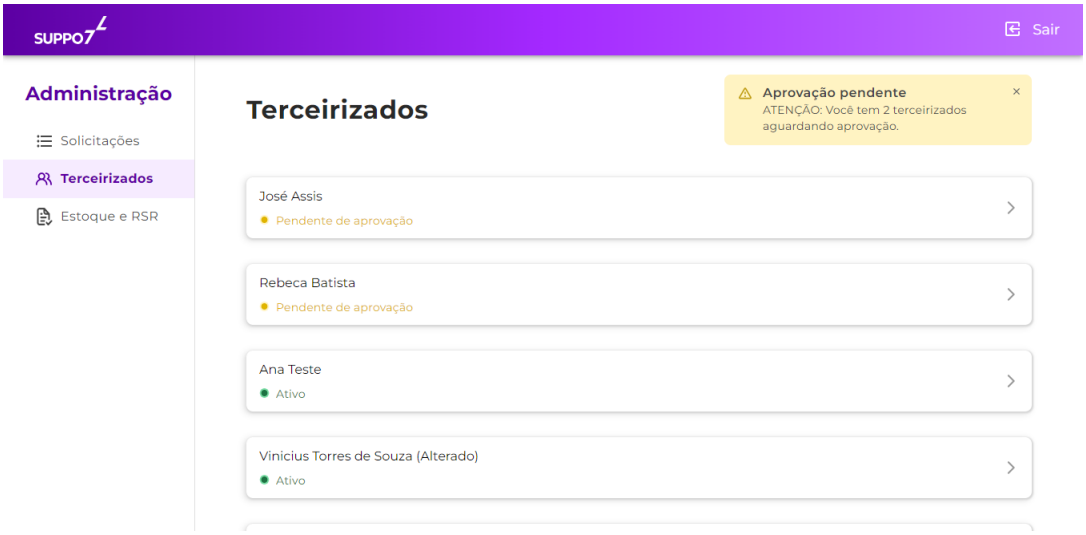
Já nesta Figura 3.4, é possível visualizar o comportamento após a aplicação do filtro. Esta só apresenta os dados filtrados e, quando trata-se de um atendimento marcado como "Finalizado", existe um link para a visualização do relatório de finalização deste, acessado ao clicar em "Ver relatório".

A dificuldade desse fluxo consistiu no desenvolvimento do seu *back-end*. Tratam-se de duas rotas, uma trazendo apenas a quantidade de pendentes de finalização, a fim de controlar a exibição do modal de aviso e a outra controla a obtenção dos dados dos cards, ou seja, a data e hora do agendamento, local, serviço prestado e a situação cadastral que encontra-se. O desafio refere-se a filtragem dos dados quando estes estiverem aplicados.

3.2.2 Desenvolvimento da tela de aprovação de prestadores B2B

O fluxo de aprovação trata-se de uma funcionalidade atribuída ao *backoffice* do sistema, nele o administrador consegue visualizar os prestadores que realizaram cadastro pelo aplicativo, objetivando trabalharem na plataforma, atendendo aquelas solicitações importadas de outras empresas.

Figura 3.5 – Representação da tela inicial de aprovação do prestador B2B



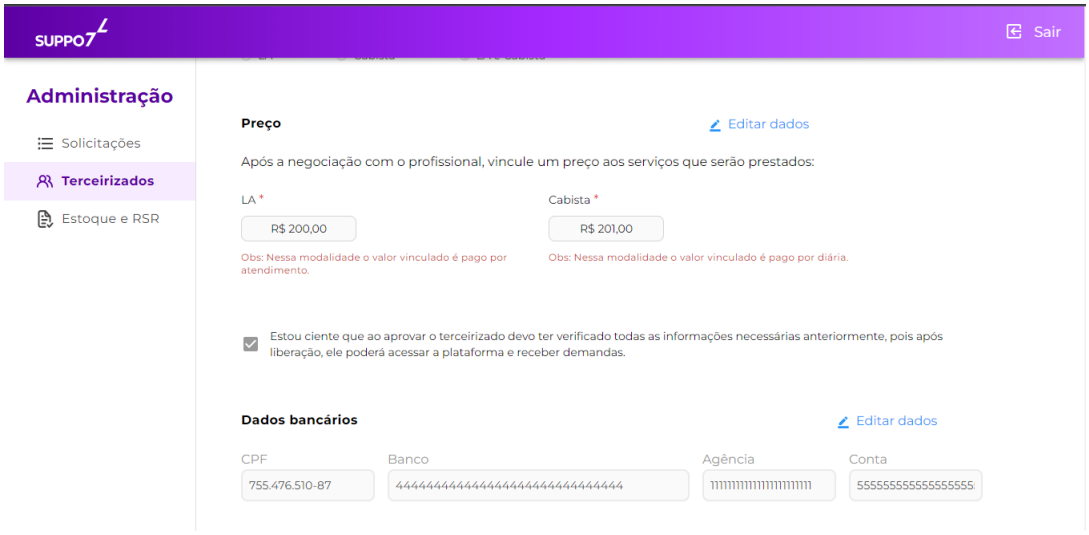
Fonte: Arquivo pessoal

A Figura 3.5 representa a tela inicial de aprovação do prestador B2B, nela é possível visualizar que existe uma listagem dos usuários que realizaram o cadastro e um modal de aviso que informa quantos ainda precisam ser analisados. Com o intuito de melhorar a usabilidade do sistema, existe uma ordenação seguida para a exibição dos dados, segundo a regra:

- Os apresentados no topo são os "Pendente de aprovação", seguidos dos "Ativos" e posteriormente dos "Reprovados".

Caso seja selecionado um que já passou pelo processo de avaliação, é exibida apenas a tela que permite a visualização dos dados preenchidos quando tomada a decisão, sendo estes os dados bancários, habilidades e valores dos serviços. Para aqueles que encontram-se ativos, é possível editar os dados.

Figura 3.6 – Representação da tela de edição do prestador já ativo B2B

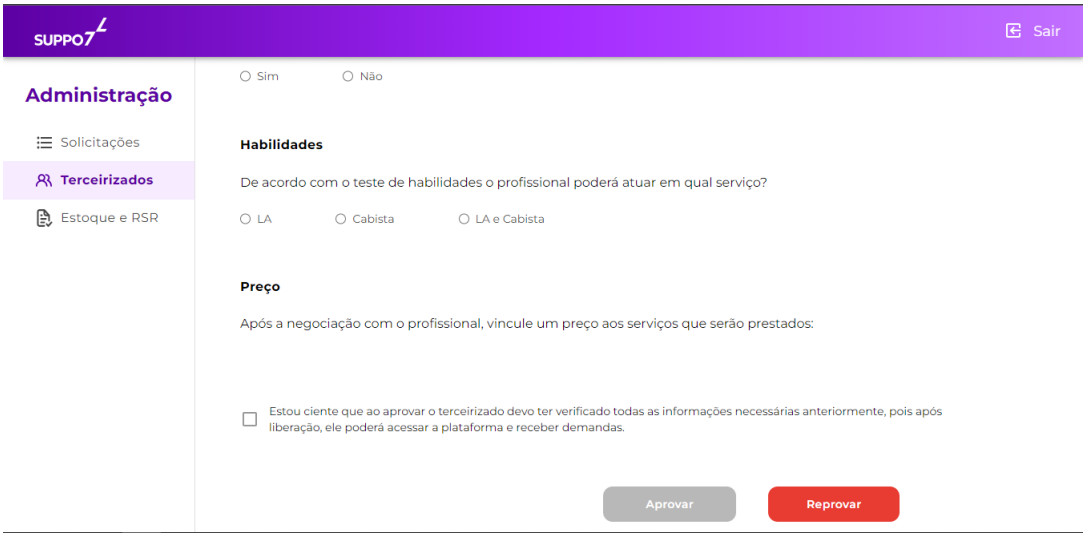


Fonte: Arquivo pessoal

É possível visualizar, através da Figura 3.6, as possibilidades de edição dos dados de preço cobrado por cada serviço e dos dados bancários daqueles prestadores que já estejam aptos na plataforma.

Caso selecionado um pendente, será aberto um formulário a preencher para que a decisão de torná-lo apto a atuar ou não seja tomada.

Figura 3.7 – Representação da tela de aprovação do prestador pendente B2B



Fonte: Arquivo pessoal

Na Figura 3.7 é possível visualizar parte do formulário a preencher pelo administrador para reprovar ou aprovar um terceirizado. No primeiro caso, basta que clique no botão em

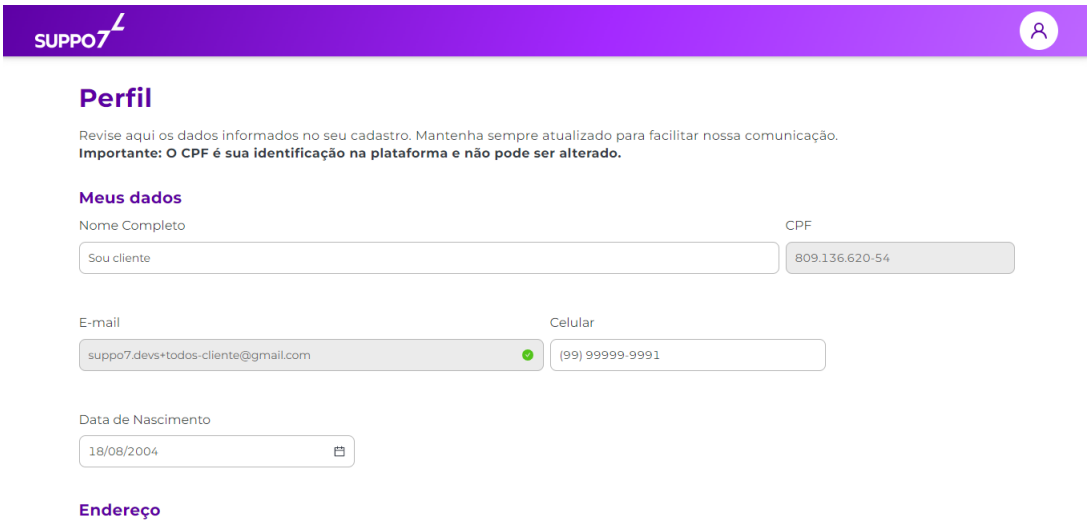
vermelho para seguir com a decisão, já o segundo, deve-se preencher todas as informações incluindo o preço cobrado pelos seus serviços na plataforma.

Por tratar-se de um fluxo WEB relativamente simples, não houveram grandes dificuldades no seu desenvolvimento além da extensão do código e o prazo que deveria ser cumprido.

3.2.3 Desenvolvimento da tela de edição dos dados do cliente B2C

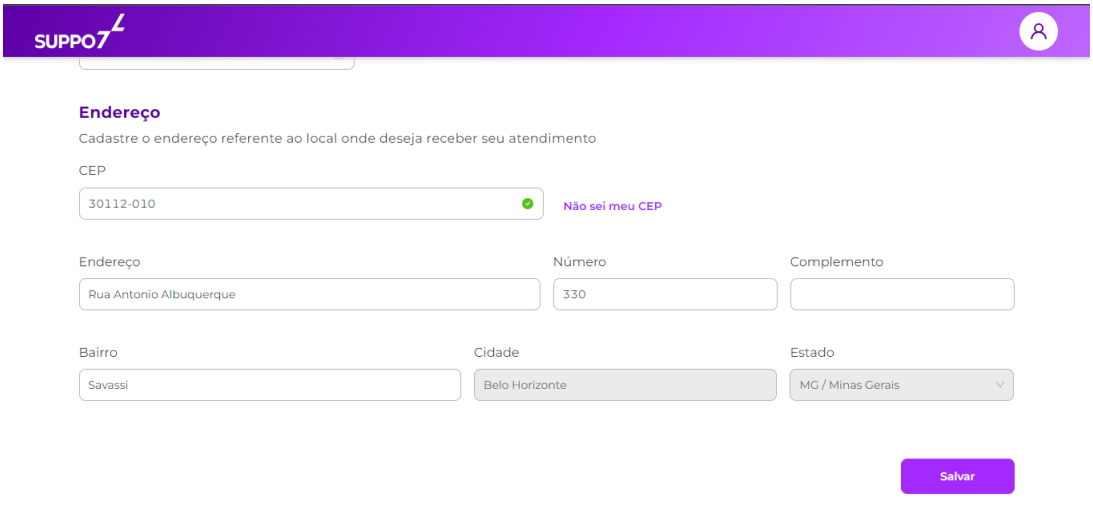
Esta subseção diz respeito a funcionalidade da plataforma que possibilita que os clientes, ou seja, aqueles que realizam as solicitações de serviço, editem os seus dados pessoais, como o nome completo, o celular, data de nascimento e o seu endereço.

Figura 3.8 – Representação da parte superior da tela de edição de dados do cliente



Fonte: Arquivo pessoal

Figura 3.9 – Representação da parte inferior tela de edição de dados do cliente



Fonte: Arquivo pessoal

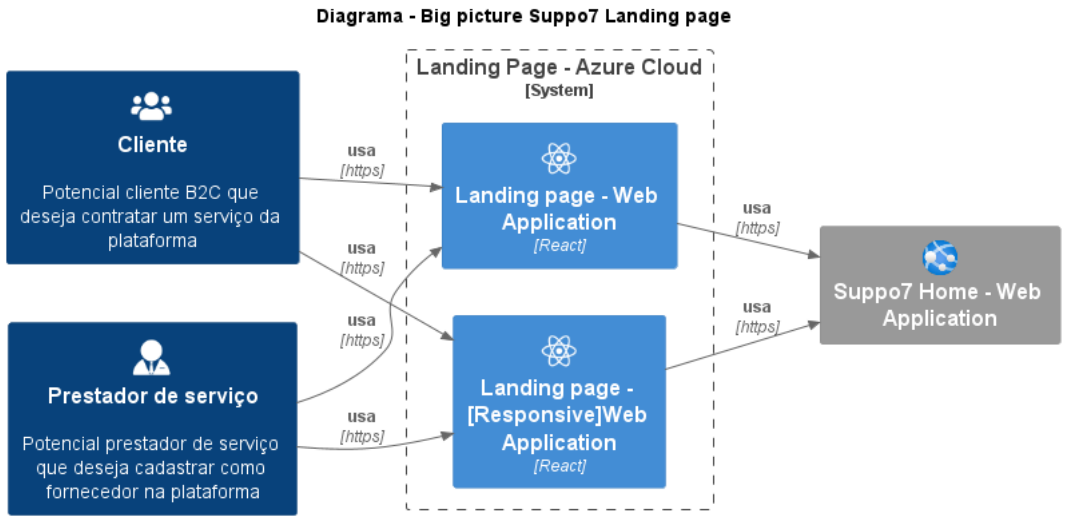
Observa-se nas Figuras 3.8 e 3.9 a representação do fluxo no *front-end*. Trata-se de um formulário que o usuário pode alterar os campos habilitados, os com coloração branca, e posteriormente, ao clicar no botão de salvar, os dados são enviados e a alteração é feita no banco. Caso este não saiba o CEP, basta clicar em "Não sei meu CEP" para redirecionar para uma página onde conseguirá obtê-lo.

O fluxo em questão demandou o desenvolvimento de duas novas rotas no *back-end*, uma delas é a de obtenção das informações do endereço do cliente a partir do CEP. Essa recebe apenas o número do dado e retorna para o *front-end* o endereço associado para o autopreenchimento dos demais campos, com exceção do número da residência. A outra é a de edição dos dados no banco a partir de um objeto contendo o preenchido pelo usuário no formulário. Realiza-se a verificação destes dados e, caso estejam validados, é feita a alteração no banco, caso contrário, é retornado o erro e informado o campo que apresenta a falha.

3.2.4 Desenvolvimento da tela de sobre nós da Landing page Suppo7

A *Landing Page*[1] da Suppo7 trata-se de uma página web estática, ou seja, não realiza chamadas ao *back-end*. Possui o objetivo de atrair novos clientes e prestadores para a plataforma; Essa é a página que aparece quando realizada uma pesquisa pelo nome da empresa em mecanismos de busca.

Figura 3.10 – Representação do fluxo da Landing Page



Fonte: Arquivo pessoal

[1] <https://suppo7.com.br/>

Observa-se na Figura 3.10 a representação da forma como o fluxo se comporta nesse caso. A esquerda, representados por retângulos azul escuro, visualiza-se as personas que acessam a página, sendo essas os clientes e prestadores em potencial da plataforma. Ao centro encontra-se o site propriamente dito, em sua versão para navegadores comuns e a responsiva, para que acessos de dispositivos móveis possam ter uma boa experiência e usabilidade. A direita situa-se a página inicial da plataforma Suppo7 B2C, o final do fluxo, onde os acessos são direcionados com a intenção de obter conversões.

Nesta, é válido citar que, durante o processo de estágio, desenvolveu-se a seção de "Sobre nós"[2] da página, possuindo a finalidade de apresentar para o usuário a respeito da plataforma, trazendo por exemplo a história desta e a sua missão.

Figura 3.11 – Representação da tela inicial de Sobre Nós da Landing Page

Fonte: Arquivo pessoal

Na Figura 3.11 é possível visualizar a tela inicial da seção em questão, nela é feito um breve resumo a respeito da empresa, como por exemplo as áreas de experiência em prestação de serviço e o tempo de existência da Telemont no mercado; Conforme o usuário rola a tela para baixo, são exibidos os demais dados desta.

Por tratar-se de um fluxo estático, não houveram dificuldades significativas no desenvolvimento. Desenvolveu-se a página WEB completa em um período curto, de aproximadamente uma semana.

[2] <https://suppo7.com.br/about-us>

4 CONCLUSÃO

O presente relatório visa abordar o processo de estágio vivenciado pelo autor do mesmo. Atuou-se como um desenvolvedor *full-stack*, ou seja, realizou atividades tanto no *front-end* e *mobile* das aplicações, quanto no *back-end*. Durante o período, trabalhou-se em dois projetos: Ducash e Suppo7, que representam diferentes contextos e dificuldades vivenciadas.

O primeiro projeto citado teve sua parceria encerrada com a DTI por motivos financeiros e, por isso, a alocação do estagiário alterou para o segundo. Ambos promoveram um aprendizado significativo e, a partir das dificuldades e defeitos de cada um, impulsionou-se o processo de discernimento das boas e más práticas de programação.

O acompanhamento da empresa e as trocas de experiência representam significativa importância para a evolução técnica e pessoal. Realizou-se diversos processos de feedback e, além disso, designou-se uma pessoa para acompanhar mais de perto a trajetória traçada, podendo auxiliar com o provimento materiais de estudo, direcionando tarefas do nível adequado e no contexto do dia a dia. O ambiente da organização é bem receptivo, aberto e adaptativo a forma individual que cada colaborador tem de adquirir conhecimento.

Do ponto de vista do escritor, este é um espaço ideal para aperfeiçoamento e aprendizado de técnicas, práticas e habilidades interpessoais. Quando necessário, houve suporte em dificuldades e, caso ocorresse algum erro na tarefa, não havia algum tipo de punição, mas sim o consentimento da naturalidade do processo de aprendizagem, reduzindo a pressão de tentativas falhas.

Algumas disciplinas vistas durante a trajetória acadêmica forneceram uma ampla base, principalmente no contexto de raciocínio lógico. Entre essas, pode-se citar: Introdução aos Algoritmos, Estrutura de Dados, Algoritmos em Grafos, Paradigmas de Linguagens de Programação, Linguagens Formais Autonômas e Práticas de Programação Orientada a Objetos. Quanto a questões operacionais, é válido citar: Engenharia de Software, Qualidade de Software, Interação Humano-Computador e Gestão de Tecnologia da Informação. Além disso, aquelas que ensinam o funcionamento e o uso dos bancos de dados demonstraram importância, sendo elas: Introdução a Sistemas de Banco de Dados e Sistemas Gerenciadores de Banco de Dados.

Como conclusão, torna-se evidente que o estágio supervisionado proporcionou diversas experiências que fomentaram o crescimento do estagiário. Adquiriu-se entendimentos amplos de desenvolvimento, nas instâncias de aplicações *front-end*, *mobile* e *back-end* e, além da parte

técnica, ampliou-se a experiência profissional, *soft skills* e fomentou-se o desejo e ambição por uma maior liberdade financeira e intelectual.

REFERÊNCIAS

AMAZON. **O que é o Scrum?** 2021. Disponível em: <https://aws.amazon.com/pt/what-is/scrum/>.

BALLERINI, R. **HTML, CSS e Javascript, quais as diferenças?** 2023. Disponível em: <https://www.alura.com.br/artigos/html-css-e-js-definicoes>.

BRASILEIRO, R. **Manifesto Ágil, o que é e qual a sua história**. 2018. Disponível em: <https://www.metodoagil.com/manifesto-agil/>.

CUNHA, A. **React Native: o que é e tudo sobre o Framework**. 2023. Disponível em: <https://www.alura.com.br/artigos/react-native>.

ESTEVAO. **Teste unitário com Jest**. 2020. Disponível em: <https://www.devmedia.com.br/teste-unitario-com-jest/41234>.

FERNANDES, D. **Expo: o que é, para que serve e quando utilizar?** 2018. Disponível em: <https://blog.rocketseat.com.br/expo-react-native/>.

GARRETT, F. **O que é Microsoft Azure? Veja como funciona e preços do serviço de nuvem**. 2020. Disponível em: <https://www.techtudo.com.br/listas/2020/07/o-que-e-microsoft-azure-veja-como-funciona-e-precos-do-servico-de-nuvem.ghtml>.

GOVERNO, E. do. **Metodologias Ágeis/Scrum/Lean**. 2021. Disponível em: <https://egov.df.gov.br/metodologias-ageis-scrum-lean/>.

KOVACS, L. **O que é e como funciona o Microsoft Teams?** 2021. Disponível em: <https://tecnoblog.net/responde/o-que-e-e-como-funciona-o-microsoft-teams/>.

MELO, D. **O que é TypeScript? [Guia para iniciantes]**. 2021. Disponível em: <https://tecnoblog.net/responde/o-que-e-typescript-guia-para-iniciantes/>.

MICROSOFT. **Bem-vindo ao Azure Cosmos DB**. 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/cosmos-db/introduction>.

MICROSOFT. **Crie. Teste. Implante.** 2023. Disponível em: <https://dotnet.microsoft.com/pt-br/>.

MICROSOFT. **Um tour pela linguagem C**. 2023. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>.

MRLRCH. **Arquitetura Web — Resumão Completo**. 2023. Disponível em: <https://medium.com/@mrlrocha/o-b%C3%A1sico-que-voc%C3%AA-precisa-saber-sobre-arquitetura-web-para-iniciar-em-web-security-cd465cc6e787>.

NEVES, V. **React: o que é, como funciona e um Guia dessa popular ferramenta JS**. 2023. Disponível em: <https://www.alura.com.br/artigos/react-js>.

ORACLE. **O que é um Banco de Dados?** 2021. Disponível em: <https://www.oracle.com/br/database/what-is-database/>.

REACT.DEV. **Quick Start**. 2023. Disponível em: <https://react.dev/learn>.

REACTNATIVE.DEV. **Introduction**. 2023. Disponível em: <https://reactnative.dev/docs/getting-started?syntax=classical&guide=android>.

SILVA, G. **O que é xUnit?** 2021. Disponível em: <https://coodesh.com/blog/dicionario/o-que-e-xunit/>.

SILVA, G. H. de L. **Tipos de testes: quais os principais e por que utilizá-los?** 2021. Disponível em: <https://www.alura.com.br/artigos/tipos-de-testes-principais-por-que-utiliza-los>.

TYPESCRIPTLANG.ORG. **TypeScript is JavaScript with syntax for types.** 2023. Disponível em: <https://www.typescriptlang.org/>.