ALFREDO BENETI DIANI

RELATÓRIO DE ESTÁGIO EM DESENVOLVIMENTO FULLSTACK NA EMPRESA DELTA GLOBAL S.A.

LAVRAS - MG

2022

ALFREDO BENETI DIANI

RELATÓRIO DE ESTÁGIO EM DESENVOLVIMENTO FULLSTACK NA EMPRESA DELTA GLOBAL S.A.

Relatório de estágio supervisionado 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. Dilson Lucas Pereira
Orientador

LAVRAS – MG
2022

ALFREDO BENETI DIANI

RELATÓRIO DE ESTÁGIO EM DESENVOLVIMENTO FULLSTACK NA EMPRESA DELTA GLOBAL S.A.

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

APROVADO em 16 de setembro de 2022
Prof. Dr. Renata Teles Moreira (UFLA)
Me. Samuel Terra Vieira (Delta Global S.A.)

Prof. Dr. Dilson Lucas Pereira
Orientador

LAVRAS – MG
2022

“There is no knowledge that is not power.”

Ralph Waldo Emerson

AGRADECIMENTOS

A Deus pela vida, saúde e sabedoria para enfrentar os obstáculos.

A minha família, meus pais José Roberto e Rose pela educação e suporte, minha irmã Isabella e minha avó Leonor, sem minha família nada disso seria possível.

A minha namorada e parceira Lethícia pelo apoio, paciência e companheirismo.

A Delta Global S.A. pela oportunidade e confiança em meu trabalho.

A Universidade Federal de Lavras e ao corpo docente pelos conhecimentos transmitidos.

E a todos que de alguma forma contribuíram para minha formação educacional e profissional.

RESUMO

O presente trabalho tem como objetivo relatar atividades desenvolvidas durante o período de estágio na área de desenvolvimento de *software* como desenvolvedor *fullstack* atuando principalmente no produto Delta Assist com foco no setor comercial na empresa Delta Global S.A. A companhia tem sua matriz na cidade de Porto Alegre-RS e uma filial de desenvolvimento de *software* em Lavras-MG. Sua atuação tem abrangência nacional no ramo de serviços, monitoramento e assistência veicular por meio da utilização de tecnologias de *software* e hardware. As principais tecnologias utilizadas durante o estágio em desenvolvimento foram: Arquitetura MVC, HTML, CSS, JavaScript, jQuery, PHP, CodeIgniter, MySQL e Git. Com o apoio dessas tecnologias, o estágio consistiu no desenvolvimento de interfaces de usuário, implementação *backend* aplicando regras de negócios e trabalho com o banco de dados a fim de propor soluções para problemas, corrigir *bugs*, implementar melhorias e novas funcionalidades.

Palavras-chave: desenvolvimento *web*; php; mvc; javascript; mysql.

ABSTRACT

The final paper aims to report activities developed during the internship period as a fullstack software developer working mainly on the Delta Assist product with a focus on the commercial sector in the company Delta Global S.A. The company has its headquarters in the city of Porto Alegre-RS and a software development branch in Lavras-MG, its operations have national coverage in the field of services, monitoring and vehicle assistance through the use of technologies such as software and hardware. The main technologies used during the development of the internship were: MVC Architecture, HTML, CSS, JavaScript, jQuery, PHP, CodeIgniter, MySQL, Git. With the support of these technologies, the internship consisted of the development of user interfaces, backend implementation applying business rules and working with the database in order to provide solutions to problems, fix bugs, implement improvements and new functionalities to the system.

Keywords: web development; php; mvc; javascript; mysql.

SUMÁRIO

1 INTRODUÇÃO

A Delta Global S.A. é uma empresa que atua nacionalmente no ramo de serviços, monitoramento e assistência veicular por meio da utilização de tecnologias de *software* e *hardware* próprios. Fundada em 2015, a empresa possui sua matriz em Porto Alegre - RS e uma filial de desenvolvimento de *software* em Lavras-MG, contando no total com cerca de 200 colaboradores e atende cerca de 220 mil veículos.

A empresa opera vendendo esses planos de assistência veicular tanto para pessoas físicas quanto para pessoas jurídicas. A contratação dos planos se dá por intermédio de consultores (corretores) internos ou externos à Delta. Após a contratação dos planos, a Delta presta o serviço de assistência para os clientes por meio do setor interno de Operações que, em caso de sinistros, acionam prestadores de serviços para prestar assistência aos clientes. Como exemplo, caso um cliente tenha um acidente com seu veículo, o mesmo aciona a Delta que irá contatar um prestador de serviço de guincho para remoção do veículo e após o ocorrido a Delta também aciona um prestador de serviços de reparo veicular. Após terminado o serviço, dependendo do tipo de plano contratado, a Delta repassa o valor para o cliente com os devidos custos dos serviços e custos administrativos. Toda a operação ocorre com o suporte do *software* desenvolvido dentro da própria empresa.

A companhia possuí vários produtos na área de *software*, sendo o principal deles o Delta Assist, uma plataforma de assistência veicular que tem suporte 24h por dia, cuidando da gestão dos planos assistenciais, setor de operações, comercial, financeiro além de outras partes da gestão interna.

Outro produto importante é o Delta Fleet, um sistema de gerenciamento de frotas de veículos com rastreamento, telemetria, relatórios de controle de jornadas e emissão de alertas, onde o contratante do serviço pode acessar e gerenciar esses dados.

Entre os demais produtos, a empresa possui o produto Historicar (sistema de consulta de históricos veiculares), a produção de um sistema de *hardware* com sensores instalados nos veículos e integrados aos sistemas da Delta, entre outros projetos.

Os serviços descritos acima são prestados para cerca de 220 mil veículos atendidos pela Delta Global e os sistemas desenvolvidos são utilizados tanto por usuários internos quanto externos.

Este documento relata as atividades desenvolvidas durante o período de estágio que ocorreu entre Junho e Setembro de 2022. As atividades desenvolvidas durante o período de estágio

foram na área de desenvolvimento de *software* como desenvolvedor *fullstack*, atuando principalmente no produto Delta Assist com foco no setor comercial.

O desenvolvedor *fullstack* engloba as atribuições tanto do desenvolvedor *frontend*, que implementa as interfaces de usuário *(client-side)*, quanto as atribuições do desenvolvedor *backend*, que por sua vez implementa as funções, lógicas de negócio e elementos relacionados ao banco de dados *(server-side)*.

Como desenvolvedor *fullstack*, o estagiário atuou com a finalidade de propor soluções para problemas, corrigir *bugs*, implementar melhorias e novas funcionalidades. O estágio foi feito no modelo híbrido, com dois dias da semana em trabalho presencial e três dias em *home office*.

Este documento está organizado em capítulos, sendo que o capítulo 2 descreve as tecnologias utilizadas para desenvolvimento das atividades, o capítulo 3 descreve as atividades desenvolvidas durante o período de estágio, enquanto o capítulo 4 traz as considerações finais relacionadas tanto ao estágio quanto à experiência do aluno no curso de Sistemas de Informação na Universidade Federal de Lavras.

2 TECNOLOGIAS UTILIZADAS

Este capítulo tem como objetivo apresentar as tecnologias utilizadas no desenvolvimento das atividades durante o estágio.

2.1 Arquitetura MVC

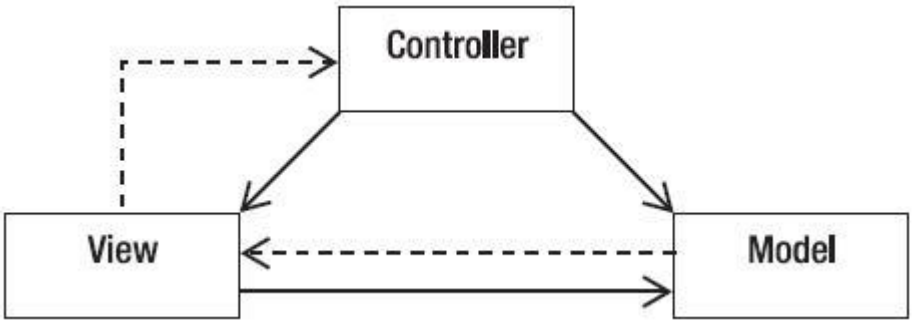
MVC *(Model-View-Controller*) é um padrão de projetos de *software* que separa seu código em 3 partes, *Model* (modelo), *View* (visão) e *Controller* (controladora). A aplicação do padrão visa enfatizar a separação entre a lógica de negócio com a modelagem das entidades e a interface (MOZILLA, 2022c).

Cada parte do MVC é responsável por tratar somente do seu escopo:

1. *Model*: Contem a modelagem das entidades;
2. *View*: Trata do *layout* e da visualização da interface;
3. *Controller*: Encaminha e trata comandos e dados entre a *Model* e a *View* de acordo com a lógica de negócios.

A Figura 1 apresenta um diagrama que demonstra a comunicação entre as 3 partes do modelo MVC.

Figura 1: Objetos utilizados no MVC e suas interações.



Fonte: DEVMEDIA, 2013.

O padrão MVC pode ser implementado diretamente pelo desenvolvedor ou podem ser utilizados *frameworks* que já possuem sua implementação centrada no padrão, como por exemplo Spring MVC, Ruby on Rails, CodeIgniter, Laravel, Django, entre outros.

Entre as vantagens de se utilizar arquitetura MVC, pode-se destacar a possibilidade de modificação de uma de suas partes sem necessitar alterar as outras, como por exemplo, alterar

a interface sem a necessidade de alterar a lógica de negócios, ou mesmo, a reutilização de trechos de código sem a necessidade de reescrita (DEVMEDIA, 2013).

O MVC é largamente utilizado como arquitetura básica de sistemas *web*. Na empresa, o sistema Delta Assist sendo um sistema *web* também utiliza MVC por toda a extensão do projeto. Na *Models* e nas *Controllers* é utilizada a linguagem PHP, enquanto nas *Views* são utilizadas JavaScript, PHP, CSS e HTML como linguagens.

2.2 HTML

Sistemas desenvolvidos para a plataforma *web*, utilizam a linguagem HTML para a apresentação do conteúdo. O HTML *(HyperText Markup Language)* é uma linguagem de marcação e estruturação de hipertexto utilizada como base para descrever páginas *web* que serão exibidas em um navegador permitindo o uso de semântica e acessibilidade (SARAIVA et al, 2018; MOZILLA, 2022a).

As páginas podem conter marcações que permitem visualizar e acessar conteúdos *web* em geral, como textos, imagens, áudios, vídeos entre outros (Silva, 2015), para isto, a linguagem tem como base a utilização de *tags* para os elementos que são descritos por um nome entre os sinais de “<” e “>” . De modo geral, as *tags* aparecem em pares, uma indicando início e a outra iniciando o final da marcação (Miletto et al., 2014), por exemplo: o elemento que indica a abertura de um parágrafo enquanto indica seu fechamento e <img> que indica a inserção de uma imagem no documento.

Os elementos do HTML podem possuir atributos que proporcionam informações adicionais sobre os elementos. Por exemplo: o elemento <a> que define um *hiperlink* possui o atributo *href=””* que define o destino a qual o *hiperlink* irá apontar mostrado no Quadro 1.

Quadro 1: Exemplo de *hiperlink.*

```
<a href="https://www.ufla.br">UFLA</a>
```

Fonte: Do autor (2022).

Um documento HTML básico pode ser definido com algumas *tags* principais para descrever sua estrutura, conforme o Quadro 2.

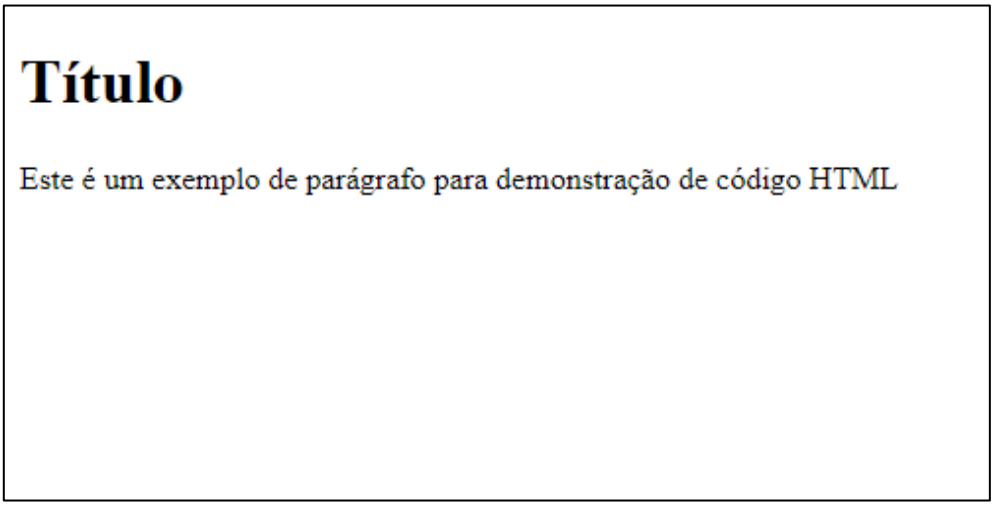
Quadro 2: Exemplo básico de documento HTML.

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8">
        <title>Título da página</title>
    </head>
    <body>
        <div id="fundo">
            <h1>Título</h1>
            <p class="paragrafo">Este é um exemplo de parágrafo para
                demonstração de código HTML
            </p>
        </div>
    </body>
</html>
```

Fonte: Do autor (2022).

Ao abrir o documento contendo o código do Quadro 2, seus elementos são interpretados e mostrados na tela como uma página *web* (Figura 2).

Figura 2: Página *web*.



Fonte: Do autor (2022).

2.3 CSS

O CSS (*Cascading Style Sheets*) é uma linguagem de estilo usada para descrever, organizar e estilizar a apresentação de um documento HTML por meio de instruções e comandos, assim, padronizando as páginas de acordo com um arquivo de estilos (.css).

A utilização de um arquivo CSS pode ser aplicada em mais de uma página, proporcionando redução de esforço e aumento de produtividade, assim como, organização nos códigos-fonte das páginas *web*, separando a estilização do conteúdo HTML (Saraiva, 2018). Tais benefícios se dão principalmente pelo fato de que o código de estilos poderá ser escrito somente uma vez e reaproveitado com sua inclusão nas demais páginas criadas ou a serem criadas.

A aplicação de estilos pode ser feita definindo atributos de estilos para uma *tag*, para uma classe ou para um elemento específico (id), conforme demonstrado no Quadro 3.

Quadro 3: Código CSS.

```css
h1{
    font-family: "Arial Black";
    font-style: italic;
    font-size: 45px;
    color: white;
    text-align: center;
}
#fundo{
    background-color: #34568B;
}
.paragrafo{
    color: white;
    text-align: justify;
}
.rodape{
    color:white;
    font-size: 10px;
}
```

Fonte: Do autor (2022).

Ao incluir o arquivo do Quadro 3, pode-se criar um documento HTML que utilizará a estilização descrita no arquivo CSS, como demonstrado no Quadro 4.

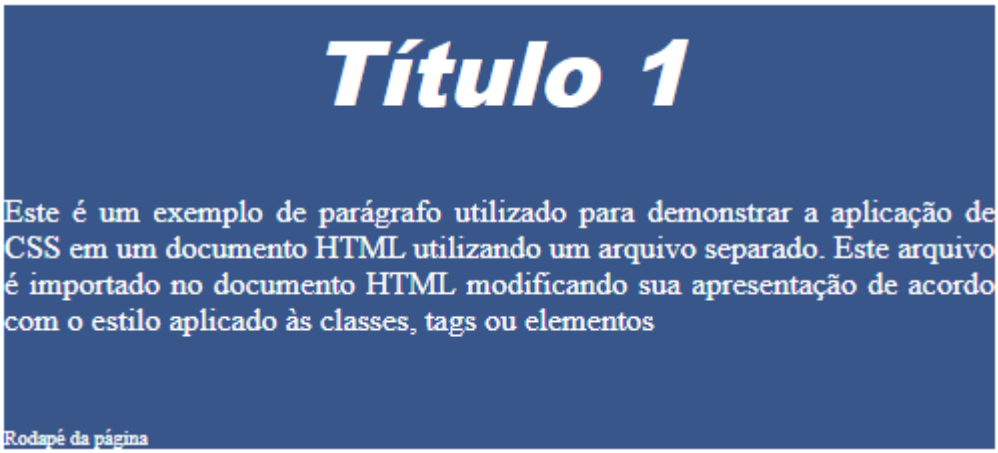
Quadro 4: Documento HTML utilizando CSS.

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8">
        <title>Título da página</title>
        <link rel="stylesheet" href="exemplo.css">
    </head>
    <body>
        <div id="fundo">
            <h1>Título 1</h1>
            <p class="paragrafo">Este é um exemplo de parágrafo
                utilizado para demonstrar a aplicação de CSS em
                um documento HTML utilizando um arquivo separado.
                Este arquivo é importado no documento HTML
                modificando sua apresentação de acordo com o
                estilo aplicado às classes, tags ou elementos.
            </p>
            <br>
            <p class="rodape">Rodapé da página</p>
        </div>
    </body>
</html>
```

Fonte: Do autor (2022).

Ao abrir o documento do Quadro 4 em um navegador, temos o resultado apresentado na Figura 3.

Figura 3: Exemplo do documento do Quadro 4 aberto em um navegador *web*.



Fonte: Do autor (2022).

2.4 Javascript

JavaScript é uma linguagem interpretada baseada em objetos e orientada a eventos, com tipagem dinâmica. A linguagem é comumente utilizada no *front-end* com a função de controlar o comportamento da página de forma dinâmica, permitindo por exemplo a manipulação de estilos e de propriedades dos elementos entre outras operações junto ao navegador (MILETTO, 2014).

Além da aplicação no *front-end*, o JavaScript também pode ser utilizado como linguagem *server-side* utilizando Node.js.

A linguagem JavaScript é baseada em ECMAScript e padronizada pela ECMA *International* (*European Computer Manufacturers Assosciation*) em conformidade com as especificações ECMA-262 (MILETTO, 2014).

No Quadro 5 é mostrado um exemplo de documento HTML para a aplicação de Javascript na manipulação das cores dos elementos de um documento HTML.

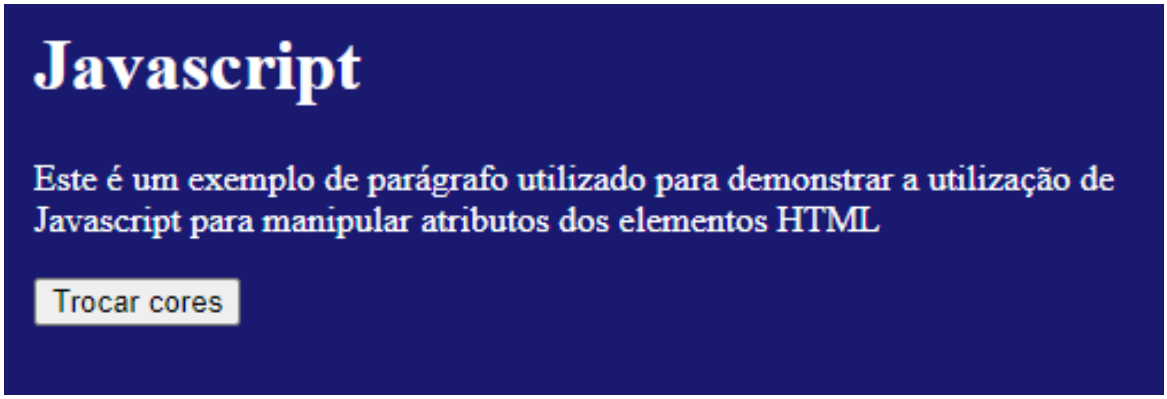
Quadro 5: Exemplo de documento HTML.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
    <title>Título da página</title>
    <script src="exemplo_javascript.js"></script>
    <style>
      p{color: white;}
      h1{color:white}
    </style>
  </head>
  <body>
    <div id="fundo" style="background-color: MidnightBlue;">
      <h1 id="titulo" style="color: white;">Javascript</h1>
      <p id="paragrafo" style="color: white;">
        Este é um exemplo de parágrafo
        utilizado para demonstrar a utilização de Javascript
        para manipular atributos dos elementos HTML
      </p>
      <button id="botao" class="btn">Trocar cores</button>
    </div>
  </body>
</html>
```

Fonte: Do autor (2022).

O código HTML apresentado no Quadro 5 produz uma página com a aparência mostrada na Figura 4.

Figura 4: Documento HTML aberto em um navegador *web*.



Fonte: Do autor (2022).

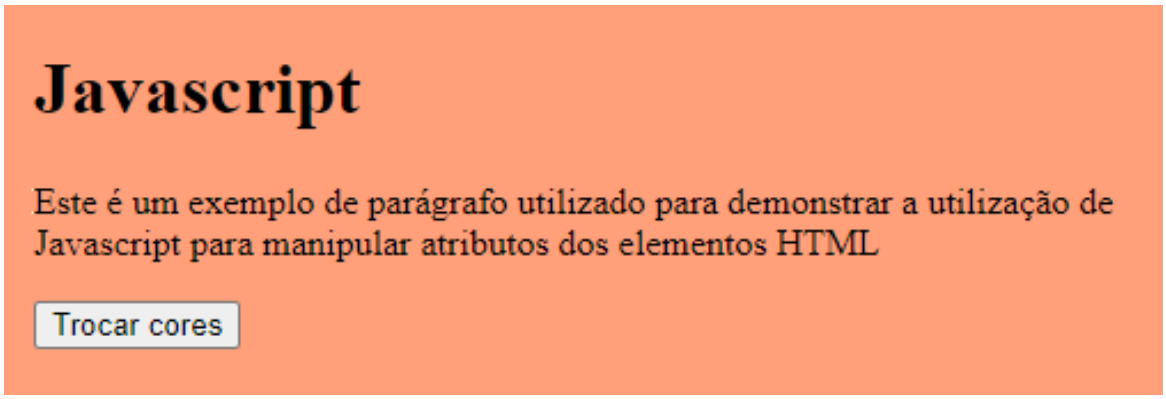
Quadro 6: Exemplo de código JavaScript.

```
document.addEventListener("DOMContentLoaded", () => {
    document.getElementById("botao").onclick = function(){
        const divfundo = document.getElementById("fundo")
        const paragrafo = document.getElementById("paragrafo")
        const titulo = document.getElementById("titulo")
        divfundo.style.backgroundColor = "LightSalmon"
        paragrafo.style.color = "black"
        titulo.style.color = "black"
    }
});
```

Fonte: Do autor (2022).

Utilizando o código de JavaScript apresentado no Quadro 6 e clicando no botão “Trocar cores”, o script é executado, alterando a aparência dos elementos da página sem a necessidade de recarregá-la e gerando um resultado apresentado na Figura 5.

Figura 5: Exemplo de aplicação de Javascript em um documento HTML.



Fonte: Do autor (2022).

2.4.1 jQuery

jQuery é uma biblioteca da linguagem JavaScript focada em simplificar a manipulação de objetos DOM (*Document Object Model*), requisições AJAX (*Asynchronous JavaScript And XML*) e manipulação de eventos (MOZILLA, 2022b).

Tomando como exemplo o código Javascript do Quadro 6, pode-se trocar sua implementação por uma mais simples aplicando jQuery, mantendo a mesma funcionalidade conforme o código exibido no Quadro 7.

Quadro 7: Exemplo de código jQuery.

```
$(document).ready(function(){
    $("#botao").click(function(){
        $("#fundo").css("background-color","LightSalmon")
        $("#titulo").css("color","black")
        $("#paragrafo").css("color","black")
    });
});
```

Fonte: Do autor (2022).

2.5 PHP

PHP (acrônimo recursivo para "*PHP: Hypertext Preprocessor*") é uma linguagem de programação interpretada, *serverside*, criada em 1995. A linguagem PHP é completa, pois oferece recursos como a criação de variáveis e constantes, funções e conexões à bancos de

dados, que propiciam a criação de programas capazes de satisfazer uma série de tarefas no desenvolvimento de um sistema. A linguagem é comumente é utilizada no desenvolvimento *web*, podendo ser mesclada dentro do código HTML permitindo o desenvolvimento de páginas *web* geradas dinamicamente (THE PHP GROUP, 2022; SARAIVA, 2018; MILETTO, 2014).

Levando-se em consideração o desenvolvimento em PHP utilizando a arquitetura MVC, a linguagem também pode ser utilizada na *Model, View* e na *Controller*.

A possibilidade de proporcionar dinamismo às páginas *web* trouxe um grande crescimento para a linguagem, de acordo com dados publicados pela W3Techs (2022) o PHP é utilizado por cerca de 77,4% dos *websites* em que a linguagem de programação *serverside* é conhecida.

Para a identificação da linguagem pelo servidor *web*, os trechos de código PHP precisam ser delimitados por *tags* iniciais e finais, <?php e ?>, a fim de diferenciá-los do HTML ou JavaScript. (MILETTO, 2014).

No Quadro 8 é apresentado um exemplo de aplicação de PHP para a criação de uma tabela (Figura 6) onde suas linhas são geradas por meio de uma estrutura de controle *"for"* e onde os cálculos são gerados por cálculos de multiplicação:

Quadro 8: Exemplo de código PHP para criar uma tabela.

```
<table>
    <tr>
        <th>x</th>
        <th>x*1</th>
        <th>x*2</th>
        <th>x*3</th>
    </tr>
    <?php for ($x=1; $x<=5; $x++) { ?>
        <tr>
            <td><?= $x ?></td>
            <td><?= $x * 1?></td>
            <td><?= $x * 2?></td>
            <td><?= $x * 3?></td>
        </tr>
    <?php } ?>
</table>
```

Fonte: Do autor (2022).

Figura 6: HTML gerado.

x	x*1	x*2	x*3
1	1	2	3
2	2	4	6
3	3	6	9
4	4	8	12
5	5	10	15

Fonte: Do autor (2022).

2.6 CodeIgniter

CodeIgniter é um *framework* para desenvolvimento de aplicações -um kit de ferramentas- para desenvolvedores que constroem sites usando PHP. Seu objetivo é permitir que seja possível desenvolver projetos com mais rapidez do que ao escrever códigos partindo do zero. O *framework* fornece um rico conjunto de bibliotecas para tarefas comumente necessárias, bem como uma interface simples e estrutura lógica para acessá-las (CODEIGNITER FOUNDATION, 2022). O CodeIgniter permite a minimização da quantidade de código necessária para a implementação de uma determinada tarefa.

O *framework* utiliza-se da arquitetura MVC para construção das aplicações, porém não obriga o desenvolvedor a utilizar esse padrão de arquitetura, mantendo-se o mais flexível possível, permitindo que o desenvolvedor escolha a arquitetura que desejar. Sua estrutura pode ter partes centrais facilmente estendidas ou completamente substituídas para que o sistema funcione da maneira que for necessária.

Atualmente o *framework* possui como principais alternativas o Laravel e o Symfony, dois frameworks bem consolidados e que, assim como o CodeIgniter, proporcionam facilidades no desenvolvimento *web* em PHP.

O CodeIgniter conta com várias *features* como:

- Bibliotecas para trabalhar com manipulação de imagens, paginação, gerenciamento de cookies, criptografia, manipulação de arquivos, entre outros;
- Segurança, gerenciando tokens de autenticação, utiliza também de filtros para validar e sanitizar *inputs*;
- Maior abstração da camada de persistência (banco de dados), utilizando-se de funções prontas para inserir, atualizar, deletar registros no banco de dados, também utilizando-

se de uma classe de *query builder* facilitando a montagem de uma requisição ao banco de dados.

Em resumo, o CodeIgniter é um framework maleável que fornece ferramentas que visam facilitar o trabalho do desenvolvedor.

2.7 MySQL

MySQL é um sistema de gerenciamento de bancos de dados (SGBD) que utiliza modelo relacional na organização dos dados. Emprega a linguagem SQL (*Structured Query Language*) para consulta, manipulação, controle e definição dos dados e suas estruturas (ELMASRI & NAVATHE, 2011; ORACLE 2022a)

Trabalhando como um sistema cliente/servidor com *multithread*, o MySQL dá suporte a diferentes linguagens de programação, bibliotecas de conexão e ferramentas administrativas, o que proporciona uma grande flexibilidade na sua utilização e explica o fato de ser o 2º SGBD no *ranking* de popularidade, se posicionando na frente de outros SGBDs como Microsoft SQL server, PostgreSQL, MongoDB, atrás somente do Oracle conforme publicado em Db-Engines (2022).

Outro fato é que o MySQL é desenvolvido e distribuído pela companhia Oracle, que também provê suporte ao SGBD, além de ser distribuído em um modelo duplo de licenças, para atender tanto necessidades comerciais quanto de projetos *open source* (ORACLE, 2022b).

2.8 Apache HTTP Server

Apache HTTP Server é um *software open-source* e multiplataforma que tem a função de um servidor *web* que utiliza o protocolo HTTP em sua comunicação (THE APACHE SOFTWARE FOUNDATION, 2022).

Com sua primeira versão criada em 1995, o servidor apache pode ser estendido com a instalação de módulos para que possa acrescentar novas funcionalidades e a capacidade de servir páginas dinâmicas feitas em linguagens como PHP, Pearl, Python, entre outras. (LONGEN, 2022)

O Apache HTTP Server pode ser utilizado como ferramenta tanto em servidor *web* em ambiente de produção, quanto também em um ambiente de desenvolvimento para servir páginas

localmente, possibilitando que desenvolvedores realizem testes sem depender acesso à internet. (SARAIVA, 2018)

Além de sua distribuição *standalone*, o Apache HTTP Server também pode ser encontrado como componente de alguns pacotes de *software* como o XAMPP, WampServer, EasyPHP, entre outros.

2.9 XAMPP

XAMPP é um pacote de *softwares* de código aberto criado pelo projeto sem fins lucrativos Apache Friends para prover um ambiente de desenvolvimento de fácil instalação e configuração (APACHE FRIENDS, 2022).

O pacote é multiplataforma e instala o Apache HTTP Server como servidor *web*, MariaDB (compatível com MySQL) como banco de dados e interpretadores das linguagens PHP e Pearl. É instalado também, um painel de controle onde pode-se configurar, executar e parar os serviços instalados pelo pacote. Com isso, o desenvolvedor tem um ambiente de desenvolvimento *web* local que pode ser utilizando tanto em desenvolvimento quanto também em testes.

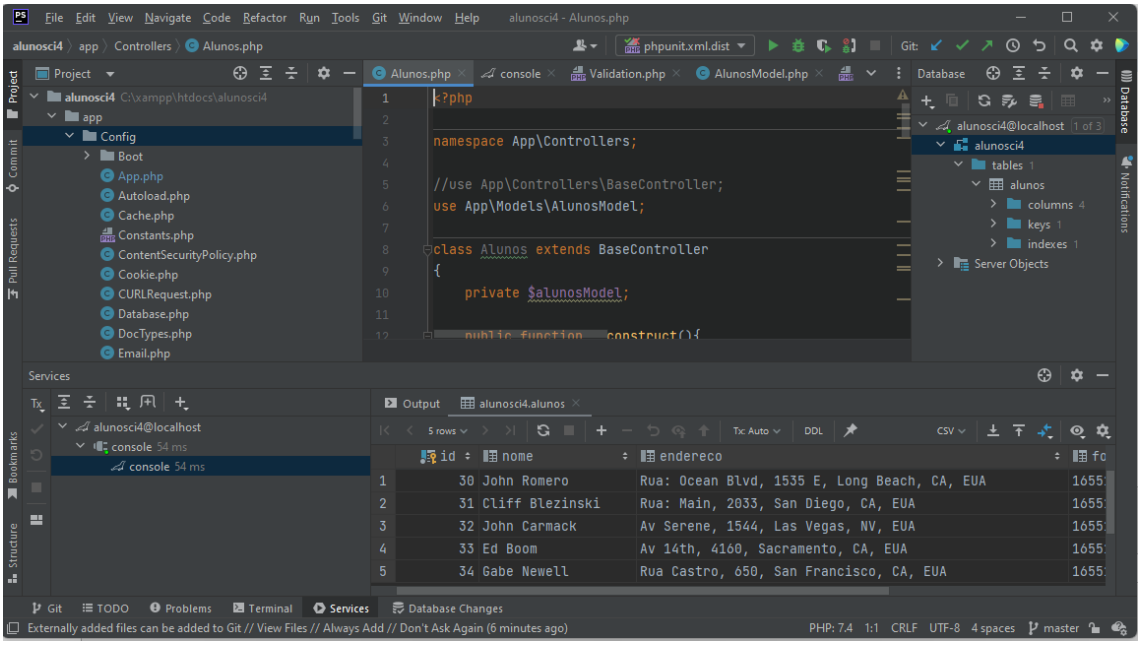
> O nome XAMPP é resultado de uma abreviação de:
> - X: significa que funciona em diferentes plataformas e sistemas operacionais.
> - A: servidor *web* Apache.
> - M: banco de dados MySQL, que será substituído no pacote pelo MariaDB.
> - P: linguagem de script PHP.
> - P: linguagem de script Perl. (SARAIVA, 2018. p.58).

2.10 PhpStorm IDE

O PHPStorm é uma IDE *(Integrated Development Enviroment)*, em português, Ambiente de Desenvolvimento Integrado, com foco em desenvolvimento *web* utilizando a linguagem PHP. Além do PHP, a IDE também tem suporte à edição de código em HTML, CSS, JavaScript e SQL.

A IDE traz em um único *software* uma interface (Figura 7) com as ferramentas básicas para o desenvolvimento *web*, como editor de código, cliente SQL, integração com sistemas de controle de versionamento, ferramentas de linha de comando, Docker, entre outras funcionalidades e ferramentas utilizadas em um ciclo de desenvolvimento (JETBRAINS, 2022).

Figura 7: Interface PHP Storm



Fonte: Do autor (2022).

2.11 Git

Git é um sistema de controle de versionamento de código criado por Linus Torvads, utilizado para rastrear e gerenciar alterações em códigos fonte.

Este sistema de controle de versão permite que equipes de vários desenvolvedores trabalhem em conjunto no desenvolvimento de um *software*, oferecendo ferramentas para resolver problemas gerados por conflito de código quando mais de um desenvolvedor trabalha no mesmo projeto, além de proteger o código-fonte contra falhas humanas que podem comprometer suas partes (ATLASSIAN, 2022).

Para resolver alguns desses problemas, um uso muito comum do Git é a criação de novas ramificações (*Branch*) independentes do código, permitindo que o mesmo seja alterado por um desenvolvedor sem impactar na criação ou modificação desse código por parte de outros membros da equipe. Mais tarde, após modificações feitas, esse código é enviado para o servidor de versionamento e cria-se uma requisição (*Pull request*) para a mescla (*merge*) entre o código criado com o código anterior, incorporando as novas funcionalidades ao *software*.

2.12 Bitbucket

Bitbucket é uma ferramenta de hospedagem e colaboração de código baseada em Git criada pela Atlassian e projetada para possibilitar que grandes equipes de desenvolvedores executem um projeto. A ferramenta é baseada em nuvem, atuando como um servidor de versionamento, sendo possível a criação, gerenciamento ou acesso aos repositórios através da internet (ATLASSIAN, 2022a).

O sistema da Atlassian possui compatibilidade com a ferramenta Git (CLI ou integrado a outra ferramenta), possibilitando sua utilização para tarefas de gerenciamento de código fonte, como a clonagem de um repositório, criação de *Commits, Branches, Forks, Pull Requests* e outras funcionalidades, a partir de linha de comando, outra ferramenta ou IDE com integração ao Git.

2.13 Kanban

Kanban é um método de gestão de trabalho, originado do Sistema Toyta de Produção. Esta ferramenta objetiva alcançar a produção *Just-In-Time*, ou seja, a produção é baseada na demanda dos clientes. O *Kanban*, literalmente, é um quadro de sinais visuais. Geralmente, os sistemas mais simples são compostos por três colunas, sendo elas: Pedido/Requisição, Produção e Concluído. Quando gerido adequadamente, é possível detectar falhas ou gargalos nas etapas. Embora originado no centro de produção, o setor de desenvolvedores de *software* aderiu ao *Kanban* visando mudar a forma de entrega de produtos e serviços, deixando mais eficiente as etapas do desenvolvimento (MONDEN, 2015; KANBANIZE, 2022).

2.14 Trello

Trello é uma ferramenta visual de colaboração que organiza projetos em cards e boards com uma disposição e apresentação muito similar a um modelo *Kanban* (ATLASSIAN, 2022b).

A ferramenta possui a capacidade de adicionar atributos a cada card, como descrição, data de entrega, membros, checklists, anexos e comentários. Além disso, a Atlassian também disponibiliza uma API que pode ser utilizada para integrar o Trello a outros sistemas (ATLASSIAN, 2022b).

3 ATIVIDADES DESENVOLVIDAS

Durante o estágio, o estagiário participou de uma equipe focada no produto Delta Assist e no desenvolvimento de soluções para o setor Comercial. A equipe que o estagiário foi inserido é formada por 4 desenvolvedores, 1 analista de TI/requisitos e 1 *tech lead*.

Esse capítulo detalha as principais atividades desenvolvidas explicando o fluxo de trabalho, correção de *bugs* e as implementações de melhorias e novas funcionalidades dentro do sistema.

3.1 Workflow

O *workflow* ou fluxo de trabalho de um desenvolvedor se inicia com criação de um *card* no Trello, descrevendo uma demanda devido à necessidade de implementação de melhorias, correção de *bugs* ou desenvolvimento de novas funcionalidades.

O *card* é criado pelo analista de TI, o qual é preenchido com os requisitos levantados em um primeiro contato com o usuário, podendo haver incrementos de requisitos durante o processo de desenvolvimento dessa demanda. Após a criação, esse *card* é atribuído a um desenvolvedor. A partir desse momento, inicia-se um contato entre o analista e o desenvolvedor, onde detalhes da demanda são passados, lógicas de negócio são explicadas e dúvidas iniciais são sanadas.

Com os detalhes iniciais definidos, é iniciado o desenvolvimento. Durante essa etapa, o desenvolvedor faz a codificação e ao surgir novas dúvidas, é iniciado um contato com outros desenvolvedores, com o analista ou com o *techlead*, a fim de resolver falhas do levantamento de requisitos, do entendimento da lógica de negócios ou da definição de estratégias para a resolução dos problemas na codificação.

Na etapa de codificação, o desenvolvedor não utiliza um banco local, utiliza-se uma conexão com banco de homologação, porém em casos muito específicos (sem alteração de dados na *database*) pode ser utilizada uma conexão com o banco de produção visando a acurácia das informações mostradas nas *Views*.

Toda a codificação é feita em um ambiente de desenvolvimento local, no computador do desenvolvedor. Para isso, uma nova *branch* é criada a partir da *branch* PRODUÇÃO, que é referente ao código que está rodando no ambiente de produção.

Após o término da codificação, o desenvolvedor deve fazer um *push* de seus *commits* na *branch* criada e então conectar no servidor de homologação/validação, onde deve fazer *checkout* nessa *branch* e então fazer um *pull*, deixando assim o ambiente preparado para acesso

pelo analista que fará testes e validação em conjunto com o usuário que requisitou esta demanda.

Caso exista algum problema, como *bugs*, não conformidade com a demanda ou adição de novos itens a serem desenvolvidos, este *card* volta para o desenvolvimento para a realização das correções.

Caso o *card* seja validado, o desenvolvedor deve criar um *pull request* desta *branch* para mesclar (*merge*) com a *branch* de PRODUÇÃO. O *pull request* passa por revisores que podem pedir ao desenvolvedor a correção de algum problema encontrado na revisão, ou, caso seja validado pelo revisor, o *merge* é executado e o servidor de produção será atualizado. Esse processo encontra-se ilustrado na Figura 8.

Figura 8: Fluxograma do *workflow*.

Fonte: Do autor (2022)

3.2 Correção de *bugs*

No vocabulário de desenvolvimento de *software*, *bug* é um erro, um defeito ou um comportamento indesejado executado pelo *software*.

Esta seção descreve atividades de correção de *bugs* executadas pelo estagiário durante o período de estágio.

3.2.1 Corrigir valores "inf" e "nan"

No algoritmo em que se criava as páginas utilizadas para a criação dos arquivos PDF e XLSX de “Contratos vencidos e a vencer” era calculado um valor chamado “Sinistralidade”, que consiste na divisão do valor indenizado pelo valor recebido de um contrato.

Em alguns casos, na coluna de “Sinistralidade”, ao invés de um valor numérico eram impressas as *strings* “inf” (*Infinite*) ou “nan” (*Not A Number*). Então foi requisitada a correção desse *bug*.

Com uma análise da função, foi detectado pelo desenvolvedor que tais valores eram gerados quando ocorria a divisão por zero ou quando um dos valores do cálculo era *null*. Para a correção do *bug*, o cálculo foi reescrito levando em consideração os dois casos indesejados e imprimindo o caractere “-“ nesses casos. Os dados referentes ao desenvolvimento desta correção são apresentados no Quadro 9.

Quadro 9: Dados do desenvolvimento da “correção dos valores *inf* e *nan”*

Commits	1
Arquivos alterados	2
Linhas de código excluídas	18
Linhas de código incluídas	44

Fonte: Do autor (2022).

3.2.2 Desativar mensagem de gatilho

Quando um contrato do tipo “contrato de gatilho” (dispara mensagens de e-mail avisando sobre o alto gasto com sinistros) atinge mais de 85% do teto de gastos, o sistema encaminha mensagens para alertar o cliente sobre a situação e trava o contrato para que não seja possível a abertura de novos sinistros quando atingir 100% dos gastos.

Porém, o sistema continuava a mandar mensagens para o cliente, mesmo após seu contrato ser travado. Então, houve a necessidade de parar o envio de mensagens quando o contrato atingisse 100% do seu teto.

Para a correção do problema, um ajuste foi feito na condicional adicionando um operador lógico “and” para tratar o caso como demonstrado no Quadro 10. No Quadro 11 são apresentados os dados referentes a este desenvolvimento.

Quadro 10: Código da correção.

```
if ($porcentagem >= 85 && $porcentagem < 100) {
    $mandar_email = true;
}
```

Fonte: Do autor (2022).

Quadro 11: Dados do desenvolvimento “desativar mensagem de gatilho”.

Commits	1
Arquivos alterados	1
Linhas de código excluídas	1
Linhas de código incluídas	1

Fonte: Do autor (2022).

3.2.3 Fix botão Excel Base Atual

Demanda gerada pela ocorrência de um *bug* ao clicar no botão "Excel Base Atual" na tela de visualização de dados do contrato. O botão deveria criar um arquivo de Excel com a base atual de veículos de um dado contrato e disponibilizá-lo para download, porém com o *bug*, ao clicar nesse botão, o sistema mostrava uma tabela chamada “histórico de propostas”, ação a qual não deveria ocorrer.

Após análise do código, foi constatado que o erro estava ocorrendo devido ao valor incorreto do atributo “id” do elemento *button* no HTML.

A correção foi efetuada trocando o “id” do botão que estava com o mesmo valor do “id” do botão "Ver" da sessão de “histórico de propostas” e os dados do desenvolvimento são apresentados no Quadro 12.

Quadro 12: Dados do desenvolvimento “*fix* botão Excel base atual”

Commits	1
Arquivos alterados	1
Linhas de código excluídas	1
Linhas de código incluídas	1

Fonte: Do autor (2022).

3.2.4 PDF sem extensão

Na tela de "Visualizar Cotação", os arquivos PDF eram gerados sem extensão. *Bug* solucionado concatenando ".pdf" ao final da *string* que dava nome ao arquivo gerado.

Durante a leitura do código para entender o *bug*, foi descoberto outro *bug* na mesma *View*. Foi detectado que uma mensagem sobre o produto Delta 24h aparecia em todos os casos em que a idade do veículo era maior que 15 anos, porém a mensagem em questão só deveria aparecer quando o plano fosse parte do Delta 24h e não para os planos individuais. Ajustado para mostrar a mensagem somente quando o tipo de proposta for COTACAO_DELTA24H. Estas correções geraram alterações no código conforme apresentado no Quadro 13.

Quadro 13: Dados do desenvolvimento "PDF sem extensão".

Commits	1
Arquivos alterados	2
Linhas de código excluídas	16
Linhas de código incluídas	19

Fonte: Do autor (2022).

3.3 Implementação de melhorias e novas *features* no sistema

Esta seção destina-se a apresentar as melhorias e novas *features* implementadas pelo estagiário no sistema Delta Assist.

3.3.1 Cotação Delta 24h – Edição

Dentro do sistema, um de seus módulos é responsável pela cotação dos planos ofertados no produto Delta 24H. Nessa cotação são preenchidas informações do corretor, cliente, veículo e plano.

A *View* responsável pela cotação trabalha de forma totalmente dinâmica utilizando JavaScript com JQuery e requisições AJAX (para *Controllers* em PHP) a fim de não precisar recarregar a página durante o processo de preenchimento e de geração da cotação. Devido ao grande número de *inputs* de formulário e informações mostradas de modo dinâmico, a navegação e entendimento do fluxo executado pelo código dessa funcionalidade se torna difícil. A *View* é carregada dinamicamente em etapas, ou seja, após o preenchimento de um formulário, ocorre uma requisição AJAX e o sistema abre novos elementos na mesma tela para

preenchimento de um novo formulário. Toda a manipulação da tela é feita dentro do código de JavaScript com uma lógica diferente de todas as outras *Views* do sistema. Tal fato dificultou o entendimento do código e o que cada evento influenciaria na dinâmica da interface.

Um problema que era enfrentado pelos usuários é o fato de que, se durante o preenchimento de uma das etapas algo fosse preenchido erroneamente, não era possível voltar as etapas para fazer a edição, forçando o usuário a recarregar a tela e recomeçar o preenchimento desde o início.

Devido ao problema citado, foi requisitada a inclusão de botões de edição que possibilitassem voltar as etapas do preenchimento, a fim de corrigir informações digitadas erroneamente.

Foram, então, adicionados botões para navegar pelo fluxo e reativar os formulários para Editar Plano e Editar Veículo, o último consequentemente leva para a edição de planos que é dependente dos dados do veículo.

Durante o desenvolvimento também foram encontrados alguns *bugs* como o não preenchimento automático do "ano modelo" do veículo e um problema com a requisição dos dados do veículo, levando à necessidade de ajustes e da implementação de uma função utilizando AJAX para buscar dados do veículo por placa e preencher novamente os campos do formulário. O Quadro 14 apresenta dados referentes ao desenvolvimento desta funcionalidade.

Quadro 14: Dados do desenvolvimento da *feature* "cotação Delta 24h- edição".

Horas aplicadas (hora)	16
Commits	1
Arquivos alterados	6
Linhas de código excluídas	33
Linhas de código incluídas	109

Fonte: Do autor (2022).

3.3.2 Contratos a Renovar Excel

Um dos relatórios emitidos pelo sistema é o de "Contratos a Renovar" que é gerado em um arquivo PDF o qual pode ser baixado pelo usuário. Porém, existia a necessidade de que também fosse gerado uma planilha XLSX do Excel para que os usuários pudessem manipular os dados conforme a necessidade.

De acordo com a necessidade apresentada, foi requisitado o desenvolvimento de uma funcionalidade que gerasse o relatório em planilha XLSX.

Com isso, foi criado um botão na *View* que gera uma página (contratos_a_vencer.php) com os dados necessários organizados em tabelas e a envia para uma função que gera um arquivo XLSX a partir da página, extraindo os dados necessários das tabelas. Os dados referentes ao desenvolvimento desta melhoria são descritos no Quadro 15.

Quadro 15: Dados do desenvolvimento da *feature* "contratos a renovar Excel".

Horas aplicadas (hora)	3
Commits	1
Arquivos alterados	5
Linhas de código excluídas	19
Linhas de código incluídas	184

Fonte: Do autor (2022).

3.3.3 Campo para Tipo de Carga - Cotação Individual

No módulo "Cotação individual" surgiu a necessidade de adicionar um campo para veículos pesados (caminhões) para descrever o tipo de carga transportada pelo veículo.

Com a demanda, houve a necessidade de adicionar uma nova coluna na tabela de veículos do banco de dados e ajustar a *Model*, a *Controller* e a *View* para que o campo possa ser lido, alterado e mostrado como informação para o usuário. Dados referentes a esta implementação são descritos no Quadro 16.

Quadro 16: Dados do desenvolvimento da *feature* "campo para tipo e carga-cotação individual".

Horas aplicadas (hora)	6
Commits	3
Arquivos alterados	5
Linhas de código excluídas	1
Linhas de código incluídas	28

Fonte: Do autor (2022).

3.3.4 Kanban Fleet Operação - atualizar board

É utilizado um *Kanban* desenvolvido dentro do sistema da Delta para controle de alguns processos e operações, entre eles toda a parte do setor de operações do Delta Fleet. Uma dificuldade que os usuários tinham é que a página desse board precisava ser atualizada manualmente para que os novos dados fossem mostrados. Com isso, foi gerada uma demanda

para que fosse implementada uma funcionalidade onde o usuário pudesse ligar um contador que atualizasse a página automaticamente a cada 5 minutos. Os dados referentes a esta implementação estão apresentados no Quadro 17.

Quadro 17: Dados do desenvolvimento da *feature* "*Kanban Fleet* Operação- atualizar *board*".

Horas aplicadas (hora)	1
Commits	1
Arquivos alterados	2
Linhas de código excluídas	6
Linhas de código incluídas	145

Fonte: Do autor (2022).

3.3.5 Kanban Fleet Operação – Sensores

Com o crescimento da empresa e a implantação de novos projetos, certas partes do sistema necessitam da adição de novas funcionalidades para acompanhar o fluxo de serviços prestados pela Delta.

Devido a esse fato, foi desenvolvido uma estrutura de *Kanban* com elementos genéricos e customizáveis, com a finalidade de adequar essa estrutura à diversas áreas da empresa e criar *Kanbans* personalizados para setores como Operação, Comercial, Suporte Técnico etc.

A fim de exemplificar a genericidade da estrutura do *Kanban* pode-se utilizar a tabela *checklist* do *Kanban*. Por exemplo, pode-se criar na interface do *Kanban* de "operações", um *checklist* de "itens a serem reparados no veículo" e com isso guardar informações na tabela "checklist" e "itens_checklist" em um relacionamento 1-N. Assim, como pode-se criar na interface do *Kanban* de "relacionamento com o cliente", um *checklist* de "contratos a renovar" e guardar as informações na mesma estrutura. Da mesma maneira, a tabela "parametro" é utilizada para descrever diversos parâmetros diferentes de um card dentro de um *Kanban*.

Dentro do *Kanban* do setor Operações do Delta Fleet, já existia uma maneira de cadastrar os veículos utilizando uma estrutura genérica de *checklist* desenvolvida para o *Kanban*. Com essa estrutura, cria-se um *checklist* de placas para controlar os serviços feitos em cada veículo.

A partir da implantação do projeto de instalação de sensores em veículos, veio a necessidade de controlar as ordens de serviço de instalação de sensores nos veículos. Então, caso um cliente contrate o serviço da Delta para sua frota e deseje instalar sensores de monitoramento em sua frota, há a necessidade de cadastramento e do controle dessa ordem de serviço. Então surgiu a

demanda de desenvolvimento de uma maneira de controlar quais sensores seriam instalados em quais veículos.

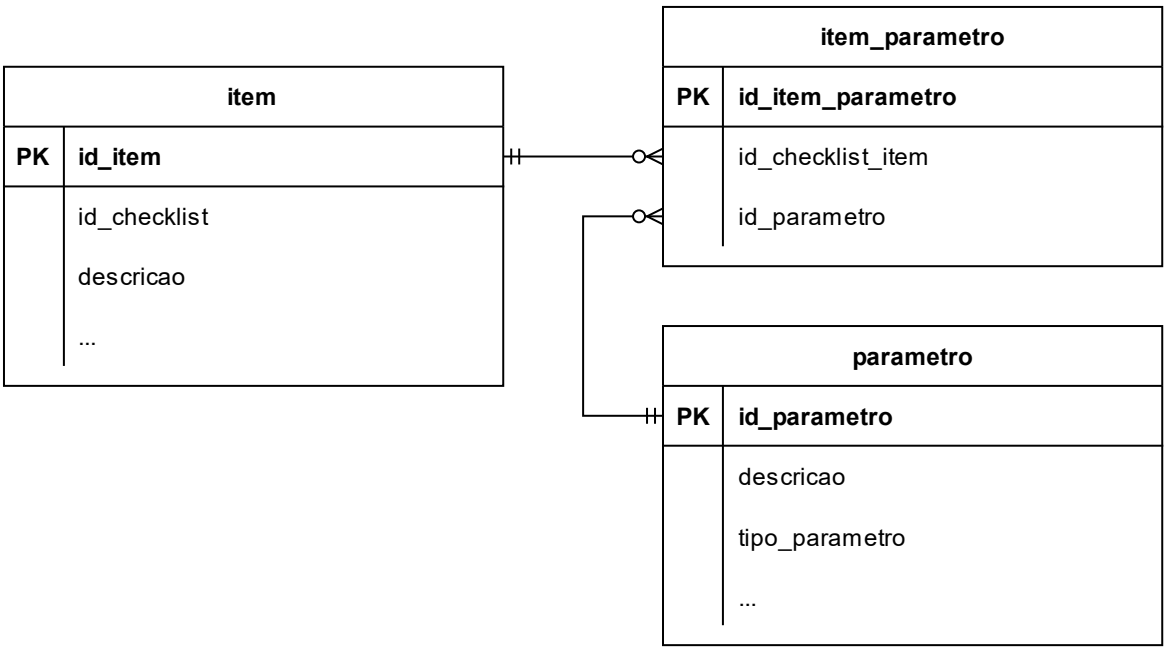
Para isso, foi desenvolvida uma solução onde foram utilizadas as tabelas de itens de *checklist* e a tabela de parâmetros (estrutura genérica para guardar parâmetros utilizados em várias partes do *Kanban*). Essa solução envolve a criação de uma tabela para fazer um relacionamento N – N entre as duas tabelas citadas anteriormente como apresentado na Figura 9.

Para a implantação da solução também foi necessária a criação de funções na *Model* e na *Controller* para manipulação dos dados, além de modificações na *View* para que os dados sejam apresentados em uma nova modal. Essa modificação na *View* necessita que tudo seja feito através de requisições AJAX para evitar que a página seja recarregada durante o cadastramento dos dados, assim, melhorando a experiência do usuário e facilitando seu trabalho.

Toda a parte do *backend* foi desenvolvida utilizando funções escritas em PHP, enquanto o *frontend* teve seu desenvolvimento utilizando JavaScript com JQuery e AJAX com a finalidade de deixar a interface dinâmica.

Esta nova funcionalidade exigiu mais planejamento das alterações, tanto em relação ao banco de dados quanto às requisições assíncronas, demandando mais tempo para ser desenvolvida, como apresentado no Quadro 18.

Figura 9: Diagrama de Relacionamento de Entidade (ER) da solução



Fonte: Do autor (2022)

Quadro 18: Dados do desenvolvimento da *feature* "*Kanban Fleet* Operação- Sensores"

Horas aplicadas (hora)	42
Commits	4
Arquivos alterados	5
Linhas de código excluídas	48
Linhas de código incluídas	237

Fonte: Do autor (2022).

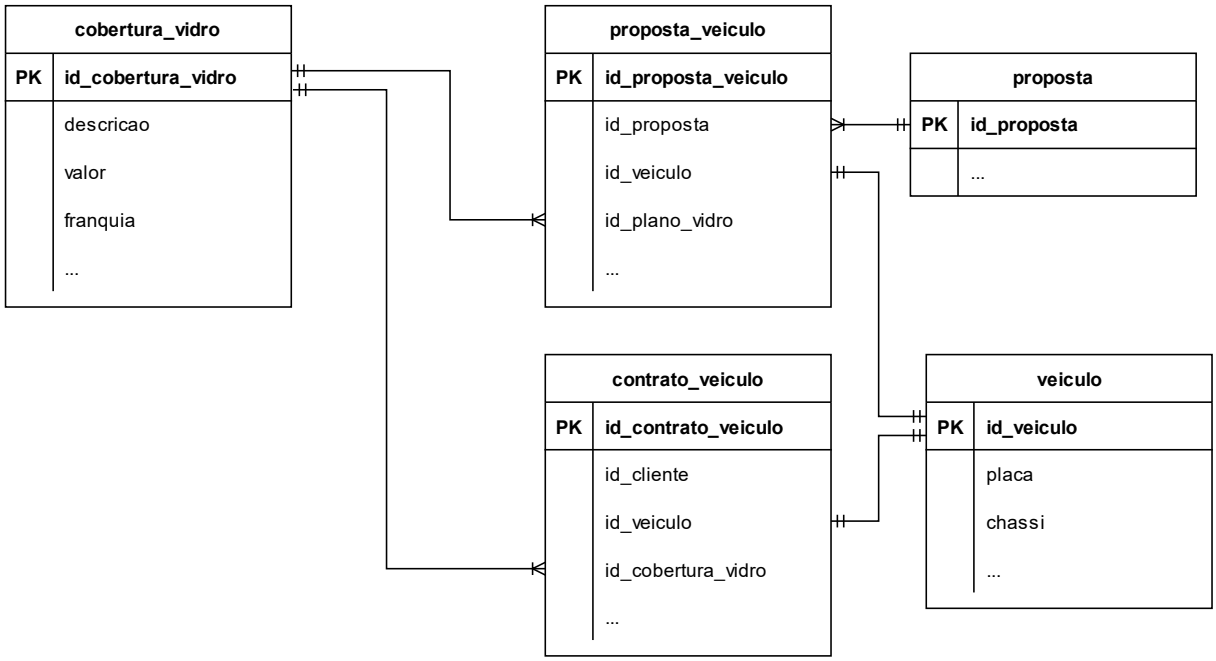
3.3.6 Plano de vidros

A empresa fornece serviços baseados em planos de assistência comercializados com os clientes. Esses planos possuem itens de cobertura, como por exemplo, a troca de vidros.

O setor Comercial da Delta identificou a oportunidade aumentar as opções de vendas de planos, transformando essa cobertura de vidros em planos de assistência que deveriam ser separados por tipos de veículo (leve ou pesado) e por ano de fabricação do veículo, com a finalidade de proporcionar diferentes faixas de preço, de coberturas e de valores de franquia pagos no acionamento de sinistros.

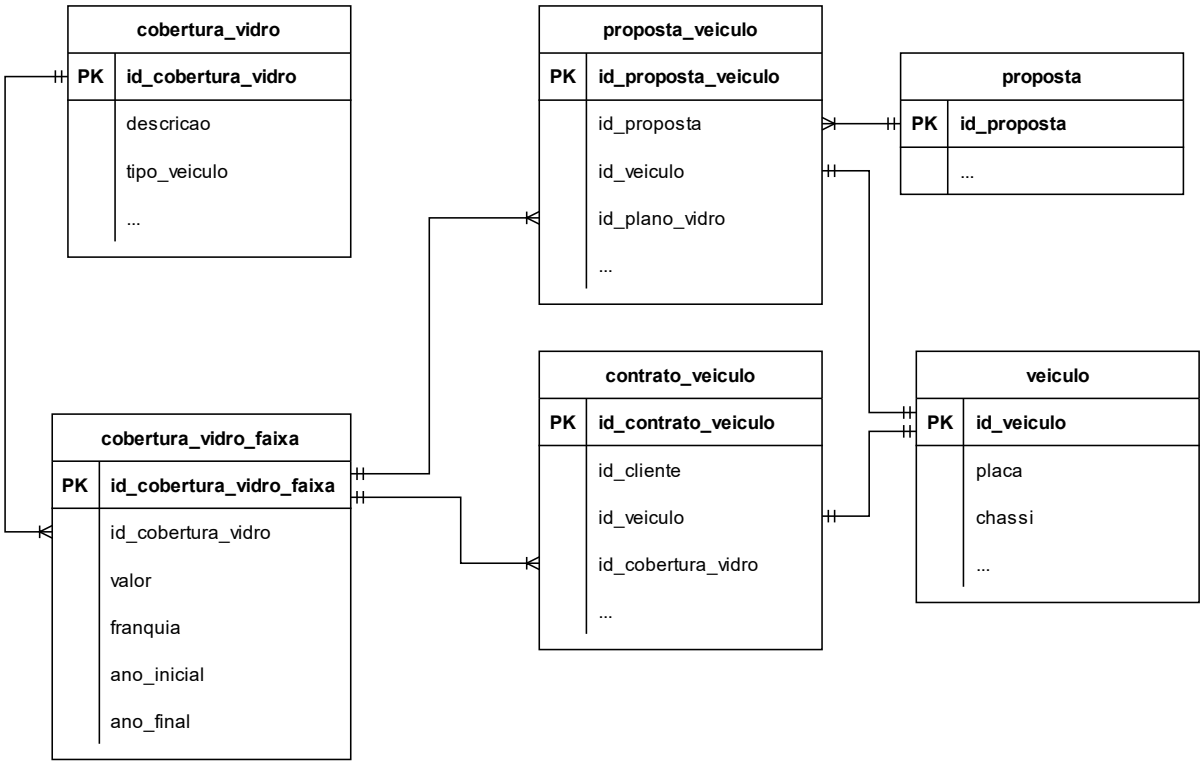
Para a adequação à essa nova regra de negócios da empresa, o desenvolvedor traçou a estratégia de criar uma nova tabela no banco de dados para cadastrar as diferentes faixas. Essa tabela estaria ligada à tabela existente de "planos de vidro". Com isso a ligação que as outras tabelas do sistema ("propostas", "contratos" etc) possuíam com a tabela "planos de vidro" antes da modificação (Figura 10), passariam a estar ligada agora com a tabela "faixas de planos de vidro" (Figura 11).

Figura 10: Diagrama de Relação de Entidade (ER) antes da alteração.



Fonte: Do autor (2022).

Figura 11: Diagrama de Relação de Entidade (ER) após a alteração.

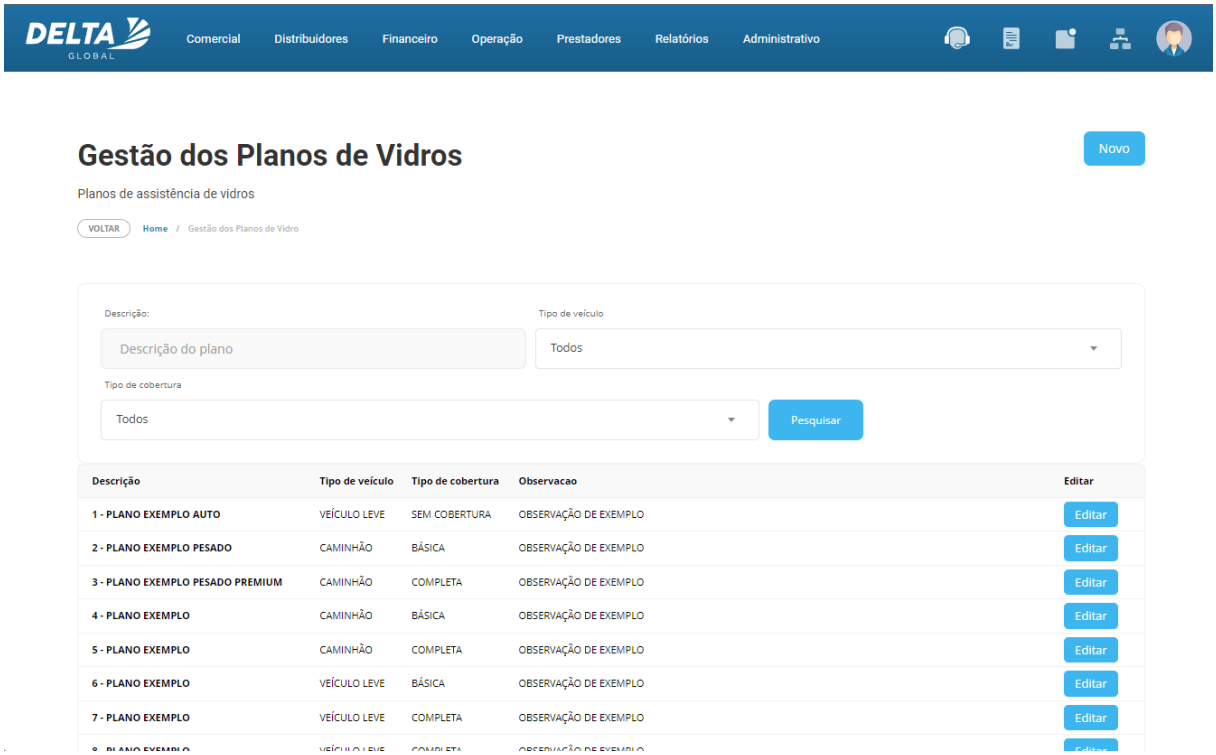


Fonte: Do autor (2022).

Foi tomada a decisão de manter a tabela original de planos de vidro e criar uma nova tabela para suas faixas, pois existem informações dos planos que seriam duplicadas entre as tabelas, como descrição, tipo de veículo, observações, informações de arquivos de modelos de contrato de cada plano etc.

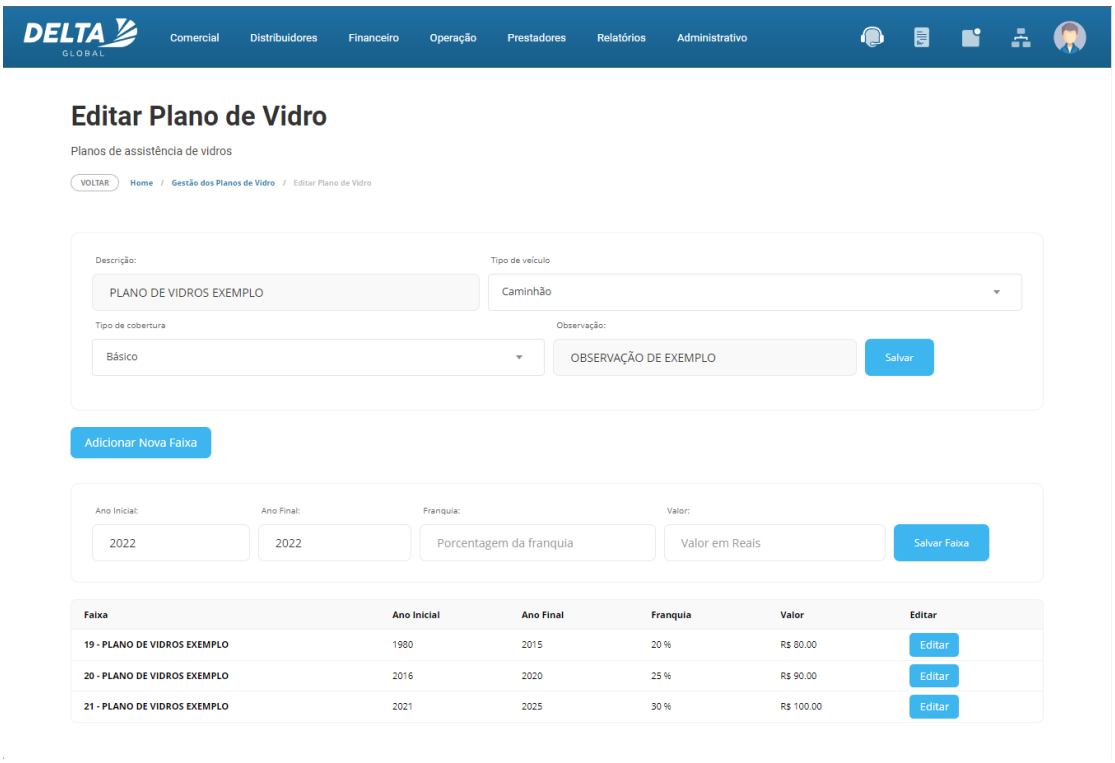
Para o gerenciamento dos planos de vidro e suas faixas, foi implementado um novo CRUD (*Create, Read, Update, Delete*) no sistema, com implementação na *Model*, *Controller* e criação de novas *Views* (Figura 12 e Figura 13).

Figura 12: Interface desenvolvida para o CRUD de Planos de Vidro.



Fonte: Do autor (2022).

Figura 13: Interface desenvolvida para o CRUD de Faixas de Planos de Vidro



Fonte: Do autor (2022).

A criação de uma nova entidade no sistema trouxe impactos em outras partes do sistema, surgindo a partir disso, a necessidade de avaliação desses impactos e a modificação de partes do código já existentes em cada uma das partes impactadas. Alguns dos impactos estão nos contratos, multi-contratos, planos, propostas, coberturas, cotações, relatórios gerados e várias *Views* que se utilizam das entidades citadas.

Muitos dos impactos foram solucionados trocando a chave estrangeira das tabelas que referenciavam a tabela “planos de vidro” para referenciar a tabela “faixas de planos de vidro” e inserindo uma tupla de faixa para cada plano de vidro utilizando o mesmo id da tabela de “planos de vidro”. Com isso ,cada registro das outras tabelas passam a referenciar uma tupla de “faixa de planos de vidro”, que por sua vez, referencia a tabela de “planos de vidro”. Além do citado, foram necessários também ajustes na *Model* e na *Controller* referente às “coberturas”.

A separação de faixas por tipo de veículo foi resolvida utilizando uma coluna na tabela de planos de vidro para tipo de veículo.

O requisito de separação de veículos em faixas por ano de fabricação do veículo foi solucionado utilizando 2 colunas na nova tabela, “ano_inicial” e “ano_final” as quais delimitam qual o limite superior e inferior do ano de modelo de um veículo para ele possa utilizar uma dada faixa de cobertura de vidros de um certo plano.

Um dos maiores desafios durante a implementação foi o fato de encontrar uma solução que atendesse à nova regra de negócios, mas ao mesmo tempo mantivesse consistência de informação com tudo que já estava cadastrado no sistema e com a lógica de negócios anterior. Este fato contribuiu para um grande número de horas aplicadas no desenvolvimento dessa nova funcionalidade (Quadro 19).

Quadro 19: Dados do desenvolvimento da *feature* “Plano de Vidros”

Horas aplicadas (hora)	110
Commits	21
Arquivos alterados	18
Linhas de código excluídas	312
Linhas de código incluídas	703

Fonte: Do autor (2022).

4 CONSIDERAÇÕES FINAIS

Durante o período de estágio, habilidades como desenvolvedor são testadas, assim como, novos conhecimentos são adquiridos.

Tratando-se de *soft skills*, o desenvolvimento de relações interpessoais, habilidade de trabalhar em equipe e principalmente a capacidade de se comunicar de forma assertiva e efetiva foram habilidades necessárias e indispensáveis para a execução de tarefas relacionadas com a atuação de um desenvolvedor de *software* no estágio.

Em relação às *hard skills*, o estágio contribuiu para uma ampliação e aprofundamento do conhecimento prévio sobre as linguagens PHP e Javascript, a utilização de um banco de dados de porte empresarial com MySQL, assim como, proporcionou um primeiro contato com a utilização de *frameworks* de desenvolvimento de *software* (CodeIgniter, neste caso) e a aplicação da arquitetura MVC em um sistema real.

A atuação como desenvolvedor de *software fullstack* dentro de uma empresa com um sistema legado que continua em desenvolvimento contínuo traz desafios, como o entendimento de todas as camadas do código com suas funcionalidades, das relações entre as entidades de dados, além da necessidade de assimilar toda a regra de negócios envolvida em cada processo em que o desenvolvedor precisa trabalhar.

Um dos principais problemas notados durante o estágio está no elo entre o desenvolvimento de *software* e as necessidades do negócio, que, nesse caso, seria na função dos analistas de requisitos. Esses por sua vez, têm grande conhecimento da lógica de negócios e de como a empresa trabalha, porém demonstram pouco ou nenhum conhecimento sobre desenvolvimento de *software*. Esse fato impacta na tradução de requisitos do negócio para o time de desenvolvimento, resultando muitas vezes em demandas com pouca explicação do problema a ser solucionado da ótica do desenvolvimento de *software*.

Ao se colocar em foco o papel que a universidade desempenha na formação do profissional e sua atuação como desenvolvedor, fica claro que as disciplinas cursadas durante a formação em BSI (Bacharelado em Sistemas de Informação) são fundamentais. Como principais destaques podem ser citadas as disciplinas de Introdução a Algoritmos e Estruturas de Dados com a construção de uma base sólida de lógica de programação, as disciplinas de Introdução a Sistemas de Bancos de Dados e Sistemas Gerenciadores de Banco de Dados como base para o entendimento da estruturação das informações em bancos de dados e o relacionamento de entidades, Engenharia de *Software*, Processos de *Software* e Gerência de Projetos de *Software* contribuindo para o entendimento do ciclo de desenvolvimento de um *software*, assim como,

as arquiteturas utilizadas em sua implementação e, finalmente, as disciplinas de Sistemas Distribuídos e Computação em Nuvem trazendo o conhecimento de implantação, arquitetura e infraestrutura de sistemas *web*.

Tendo como base o crescente número de soluções de *software* e aplicações que estão sendo desenvolvidas e utilizadas na nuvem, nota-se um certo desfalque na formação do profissional de TI egresso do curso de BSI, pois existe pouco contato com linguagens voltadas para desenvolvimento *web* durante o curso. A única experiência de desenvolvimento *web* real do aluno dentro do curso é um trabalho realizado em PHP na disciplina de Engenharia de *Software*. Para resolver o problema apresentado, uma solução viável é a implantação de uma disciplina obrigatória voltada completamente para o desenvolvimento de sistemas *web*.

Ao final do curso de BSI, do estágio e da redação deste trabalho de conclusão de curso, nota-se que o aluno egresso passa por um amadurecimento profissional e intelectual durante esse processo, terminando essa etapa com capacidade de atuar como desenvolvedor de *software*, assim como, devido a fundação de uma boa base, aprender novas linguagens, tecnologias e conceitos para desempenhar seu papel como profissional.

REFERÊNCIAS

APACHE FRIENDS. **XAMPP Apache + MariaDB + PHP + Perl**. 2022. Disponível em: https://www.apachefriends.org/. Acesso em: 24 ago. 2022.

APACHE. **Apache HTTP Server Project**. 2022. Disponível em: https://httpd.apache.org/ABOUT_APACHE.html. Acesso em: 24 ago. 2022.

ATLASSIAN. **O que é controle de versão?** 2022a. Disponível em: https://www.atlassian.com/br/git/tutorials/what-is-version-control. Acesso em: 24 ago. 2022.

ATLASSIAN. **Introdução ao Trello**. 2022b. Disponível em: https://trello.com/guide. Acesso em: 30 ago. 2022.

CODEIGNITER FOUNDATION (org.). **CodeIgniter4 Overview**. 2022. Disponível em: https://codeigniter.com/user_guide/concepts/index.html. Acesso em: 08 ago. 2022.

DB-ENGINES. **DB-Engines Ranking**. 2022. Disponível em: https://db-engines.com/en/ranking. Acesso em: 29 ago. 2022.

DEVMEDIA. **Introdução ao Padrão MVC**. 2013. Disponível em: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308. Acesso em: 29 ago. 2022.

ELMASRI, Ramez; NAVATHE, Shamkant B.. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Universidades, 2011. 808 p.

JETBRAINS. **PhpStorm**: O IDE PHP super inteligente. Disponível em: https://www.jetbrains.com/pt-br/phpstorm/. Acesso em: 22 ago. 2022.

LONGEN, Andrei Silva. **O que é Apache? Uma Visão Aprofundada do Servidor Apache**. 2022. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-apache). Acesso em: 23 ago. 2022.

MILETTO, Evandro M.; BERTAGNOLLI, Silvia de C. **Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP (Tekne**). São Paulo: Grupo A, 2014. E-book. 9788582601969. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582601969/. Acesso em: 22 ago. 2022.

MONDEN, Yasuhiro. **Sistema Toyota de Produção.** Porto Alegre: Grupo A, 2015. E-book. ISBN 9788582602164. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582602164/. Acesso em: 07 set. 2022.

MOZILLA (org.). **HTML: Linguagem de Marcação de Hipertexto**. 2022a. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML. Acesso em: 22 ago. 2022.

MOZILLA (org.). **JQuery**. 2022b. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/jQuery. Acesso em: 28 ago. 2022.

MOZILLA (org.). **MVC**. 2022c. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/MVC. Acesso em: 28 ago. 2022.

ORACLE. **Commercial License for OEMs, ISVs and VARs**. 2022b. Disponível em: https://www.mysql.com/about/legal/licensing/oem/. Acesso em: 29 ago. 2022.

ORACLE. **MySQL 8.0 Reference Manual**: what is mysql?. What is MySQL?. 2022a. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Acesso em: 29 ago. 2022.

SARAIVA, Maurício de O.; BARRETO, Jeanine dos S. **Desenvolvimento de sistemas com PHP**. São Paulo: Grupo A, 2018. E-book. 9788595023222. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595023222/. Acesso em: 22 ago. 2022.

SILVA, Maurício Samy. **Fundamentos de HTML5 e CSS3**. São Paulo: Novatec Editora, 2015. 304 p.

THE PHP GROUP. **Manual do PHP**. Disponível em: https://www.php.net/manual/pt_BR/preface.php. Acesso em: 22 ago. 2022.

W3TECHS. **Usage statistics of PHP for websites**. Disponível em: https://w3techs.com/technologies/details/pl-php. Acesso em: 22 ago. 2022.