UNIFEOB
ESCOLA DE NEGÓCIOS

2023

PROJETO DE CONSULTORIA EMPRESARIAL

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GESTÃO DE TECNOLOGIA DA INFORMAÇÃO

PROJETO INTEGRADO

Portifólio Web: Estúdio de Tatuagem

SÃO JÃO DA BOA VISTA, SP

JUNHO 2023

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GESTÃO DE TECNOLOGIA DA INFORMAÇÃO

PROJETO INTEGRADO

Portifólio Web: Estúdio de Tatuagem

MÓDULO DESENVOLVIMENTO DESKTOP

Ferramentas de Desenvolvimento Web – Prof. Nivaldo de Andrade

Sistemas Operacionais – Prof. Rodrigo Marudi de Oliveira

Projeto de Desenvolvimento Web – Prof. Nivaldo de Andrade

Estudantes:

Thais de morais vieira, RA 1012022100664
Rafael Felipe Cabello Fiore, RA 1012023100499
Marcelo Augusto Camargo Souza dias, RA 1012023100332
Lucas Balbino Paulino, RA 1012023100315
João Pedro Conde de Oliveira, RA 1012023100356
Rebeca Sara de Oliveira Biazoto, RA 1012023100510

SÃO JOÃO DA BOA VISTA, SP
JUNHO, 2023

SUMÁRIO

1 INTRODUÇÃO

O Projeto Integrado desenvolvido neste trimestre, terá como principal objetivo desenvolver um sistema de cadastro de dados e um site portfólio, para um estúdio de tatuagem que fica localizado na cidade de Aguaí-SP. Com o intuito de armazenar dados de clientes, facilitar o controle dos proprietários, aumentando a visão estratégica deles, já que terão um controle numérico de pessoas que passaram pelo estúdio. Além disso, o site portfólio irá facilitar a divulgação das artes no meio online, proporcionando assim a atração de mais clientes.

Com o advento da internet, houve uma mudança no perfil dos consumidores, atualmente eles não saem mais de casa para escolher o produto, preferem escolher digitalmente. Assim como afirma Alves (2014), as empresas têm, na atualidade, a necessidade de ter seu próprio site, sua página em uma rede social e ter a divulgação de seu portfólio, pois o tempo é algo precioso para os consumidores e a internet proporcionou essa praticidade. Um site portfólio atrativo aumenta a chance de captação de clientes e aumenta a proporção de divulgação dos produtos e serviços oferecidos pela empresa.

Outro fator que irá agregar o estúdio será o banco de dados, através dele será possível localizar os clientes que já se tatuaram ou que demonstraram interesse em alguma arte. Para Silva (2019), as tecnologias da informação aumentam a confiabilidade e integridade sobre os dados armazenados, além disso melhoram o desempenho quando se torna necessário gerar informações ou relatórios.

Por fim, o presente projeto integrado promoverá o trabalho em equipe entre alunos que estão cursando a matéria referente e que buscam alcançar o mesmo nível de aprendizagem durante o desenvolvimento do software que agregará para a gestão da empresa.

2 DESCRIÇÃO DA EMPRESA

O estúdio de tatuagem denominado FaelFelipe Tattoo, fica localizado no seguinte endereço: Rua Alexandrino de Alencar, 978, CEP 13863092, Aguaí-SP.

O público-alvo que frequenta o estabelecimento são jovens e adultos, de ambos os sexos. Os serviços oferecidos pelo estúdio são criação de artes para tatuar e o ato de tatuar, podendo ser realizado em uma ou mais sessões. O estúdio possui somente um funcionário, denominado Rafael, além disso ele também é o proprietário do estabelecimento.

3 PROJETO DE CONSULTORIA EMPRESARIAL

Para o desenvolvimento deste Projeto Integrado, foram utilizados os conteúdos teóricos das matérias “Ferramentas de Desenvolvimento Web” e “Sistemas Operacionais”. Dentro do conteúdo referente a matéria “Ferramentas de Desenvolvimento Web”, está o desenvolvimento do website e API (Interface de Programação de Aplicação). E dentro do conteúdo referente a matéria “Sistemas Operacionais”, está a publicação do website e da API e seu banco de dedos.

Para a criação do website, foi utilizado HTML, TailwindCSS e JQuery + AJAX e outros plugins que adicionaram funções necessárias para qualquer formulário em um website. O TailwindCSS é uma biblioteca JavaScript que gera um arquivo .CSS compilado através da busca por classes em todos os arquivos HTML e JavaScript disponíveis para busca no projeto. O JQuery é uma biblioteca JavaScript utilizada para manipular a DOM (Modelo de Documento por Objeto), o AJAX é um módulo nativo do JQuery responsável por realizar requisições há API’s, e o JQuery Mask Plugin é um plugin feito por terceiros que é largamente utilizado para adicionar máscaras aos campos de formulários.

Para a criação da API, foi utilizado TypeScript, Fastify, Prisma e Zod. O TypeScript é uma biblioteca JavaScript que adiciona uma nova sintaxe ao JavaScript, o tornado uma linguagem tipada. O Fastify é um eficiente framework JavaScript, com módulo de interpretação para o TypeScript, que serve de servidor HTTP (Hypertext Transfer Protocol) para a aplicação, nele podemos gerenciar as rotas de execução de código e diversos outros parâmetros. O Prisma é uma biblioteca de JavaScript/TypeScript que auxilia o desenvolvimento e conexão com o banco de dados, tratando as tabelas como objetos para auxiliar no desenvolvimento. O Zod é uma biblioteca TypeScript que serve para manter os dados tipados no JavaScript, então se desenvolve usando o Zod no TypeScript e após o processo de conversão de TypeScript em JavaScript o Zod se mantém no código para mantê-lo tipados.

3.1 FERRAMENTAS PARA DESENVOLVIMENTO WEB

Para o desenvolvimento deste projeto optamos por fazer o desenvolvimento em duas fases que se comunicam, o website e a API. Utilizando o GitHub como plataforma de versionamento de todo o projeto, utilizamos a Vercel como hospedagem para o Website e a Render para hospedagem da API e do banco de dados PostgreSQL. A vantagem de utilizar plataformas Web para a publicação de sites e serviços é que não há uma limitação de Sistema

Operacional, então o desenvolvimento se torna muito mais fácil. Tanto a Vercel quanto a Render possuem ferramentas para facilitar a publicação de projetos, então quando ocorre um novo “commit” a plataforma identifica e já faz a atualização automaticamente. A Render, para facilitar a publicação de aplicações focadas em back-end as envelopa em um container Docker e roda o container ao invés de rodar a aplicação diretamente em uma máquina, assim removendo a responsabilidade de haver um sistema operacional específico para executar a aplicação.

3.1.1 PROJETANDO A FERRAMENTA

O primeiro passo para o desenvolvimento foi a confecção de um design no projeto figma com as ideias de todos do grupo, juntamente com o dono da empresa escolhida, após apurarmos os designs chegamos a versão final do projeto, o qual nos fornecia toda a identidade e paleta de cores do portifólio.

Após isso passamos para a etapa de desenvolvimento na ferramenta VSCode, ultizando o HTML para estruturar as informações, TailwindCSS para aplicação de estilo e JavaScript para algumas ações.

3.1.2 PROJETO DE INTERFACE COM O USUÁRIO

Para desenvolvermos nosso projeto nos baseamos em 3 fatores importantes de projeto de interface que são análise de usuário, análise de tarefa e análise ambiental. Analisando cautelosamente estes aspectos concluiu-se que o público-alvo seria jovens/ adultos que são os mais interessados no mundo da tatuagem.

Já na análise de tarefa, chegou-se à conclusão de que o público-alvo teria se somente fornecer alguns dados (Nome, Email, Telefone) para ficar por dentro das promoções do estúdio, assim como teriam de forma facilitada o acesso ao Instagram da empesa para possíveis cotação de preços e agendamento de horário.

No aspecto ambiente, foi definido que este portifólio seria disponibilizado nas redes sociais, tendo mais alcance de clientes.

Visto isso, juntamente com conceito UI/UX o projeto manteve um design simples e moderno, seguindo toda identidade da empresa, mas sem perder a funcionalidade trazendo de forma ilustrativa os trabalhos para facilitar assim a interação entre os usuários e o portifólio.

3.1.3 LINGUAGEM DE DESENVOLVIMENTO

projeto-integrado-web-main (1) - index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Fael Felipe Tattoo Studio - Home</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link href="./styles/output.css" rel="stylesheet" />
    <script src="https://unpkg.com/@phosphor-icons/web"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
    <script type="text/javascript" src="scripts/buttonAction.js"></script>
  </head>
  <body class="flex flex-col h-screen bg-[#122630] text-[#C9926B] font-serif">
    <header class="flex flex-row flex-nowrap h-20 top-0">
      <div class="flex h-20 w-40 items-center">
        <a target="_blank" href="https://www.instagram.com/faelfelipetattoo/" class="pl-10 hover:text-[#F9B486] align-middle">
          <i class="ph-bold ph-instagram-logo text-6xl"></i>
        </a>
      </div>
      <div class="flex flex-row h-20 items-center w-6/12 justify-evenly text-3xl font-semibold">
        <a href="views/portifolio.html" class="hover:text-[#F9B486] hover:border-b-2 border-[#F9B486]"><h1>Meu Portifólio</h1></a>
        <i class="ph-bold ph-dot"></i>
        <a href="views/flashes.html" class="hover:text-[#F9B486] hover:border-b-2 border-[#F9B486]"><h1>Flashs</h1></a>
        <i class="ph-bold ph-dot"></i>
        <a href="views/about.html" class="hover:text-[#F9B486] hover:border-b-2 border-[#F9B486]"><h1>Sobre</h1></a>
      </div>
      <div class="absolute flex h-20 right-10 items-center">
        <button id="signInButton" class="w-60 h-10 bg-[#C9926B] text-black rounded-full text-2xl uppercase font-medium" style="z-index: 1000">Cadastre-se</button>
      </div>
    </header>
    <div class="w-9/12 bg-[#C9926B] h-0.5"></div>

    <content class="flex flex-col h-screen w-screen">
      <div class="absolute flex h-14 right-10">
        <h6 class="text-center w-60">Cadastre-se e receba promoções em seu e-mail</h6>
      </div>
      <div class="absolute flex flex-col mt-80 ml-96">
        <h1 class="text-center font-bold text-9xl">Fael Felipe</h1>
        <h1 class="text-center font-bold text-9xl">Tattoo Studio</h1>
      </div>
      <div class="absolute flex right-10 top-0 h-full">
        <img src="assets/LOGO.svg" class="" />
      </div>
    </content>
  </body>
</html>

```

Figura 1 - Código da página inicial, utilizando HTML e TailwindCSS

projeto-integrado-web-main (1) - buttonAction.js

```
// const { createClient } = require("main.js");

$(document).ready(function () {
  $("#signInButton").click(function (event) {
    event.preventDefault();
    window.location.href = "./views/signin.html";
  });

  $("#signInButtonViews").click(function (event) {
    event.preventDefault();
    window.location.href = "./signin.html";
  });

  $("#phone").mask("(00) 0 0000-0000");

  $("#signin").click(function (event) {
    event.preventDefault();

    let fullname = String($("#fullname").val());
    let email = String($("#email").val());
    let password = String($("#password").val());
    let phone = String($("#phone").val().replace(/(\D)/g, ""));

    if (fullname || email || password || phone) {
      let cliente = {
        fullname: fullname,
        email: email,
        password: password,
        cellphone: phone,
      };

      $.ajax({
        type: "POST",
        dataType: "json",
        contentType: "application/json",
        crossDomain: true,
        data: JSON.stringify(cliente),
        url: "https://projeto-integrado-server.onrender.com/clientes",
      });

      $("#signin").val("Cadastrado");
    } else {
      $("#signin").val("Preencha o Formulário");
    }
  });
});

```

Figura 2 - Código do JavaScript responsável por controlar os botões, máscaras do formulário e fazer requisições a uma API utilizando o AJAX (JavaScript e XML Assíncrono)

projeto-integrado-server-main - schema.prisma

```
// This is your Prisma schema file,
// learn more about it in the docs: https://pris.ly/d/prisma-schema

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model cliente {
  id String @id @db.VarChar(64) @default(cuid())
  fullname String @db.VarChar(64)
  email String @unique
  password String @db.Char(60)
  cellphone String @db.Char(11)
  isAdmin Boolean @default(false)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  agenda agenda[]
}

model agenda {
  id String @id @db.VarChar(64) @default(cuid())
  data DateTime @db.Timestamp()
  clienteId String @db.VarChar(64)
  cliente cliente @relation(fields: [clienteId], references: [id])
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
```

Figura 3 – Código de configuração do Prisma e criação das tabelas do banco de dados.

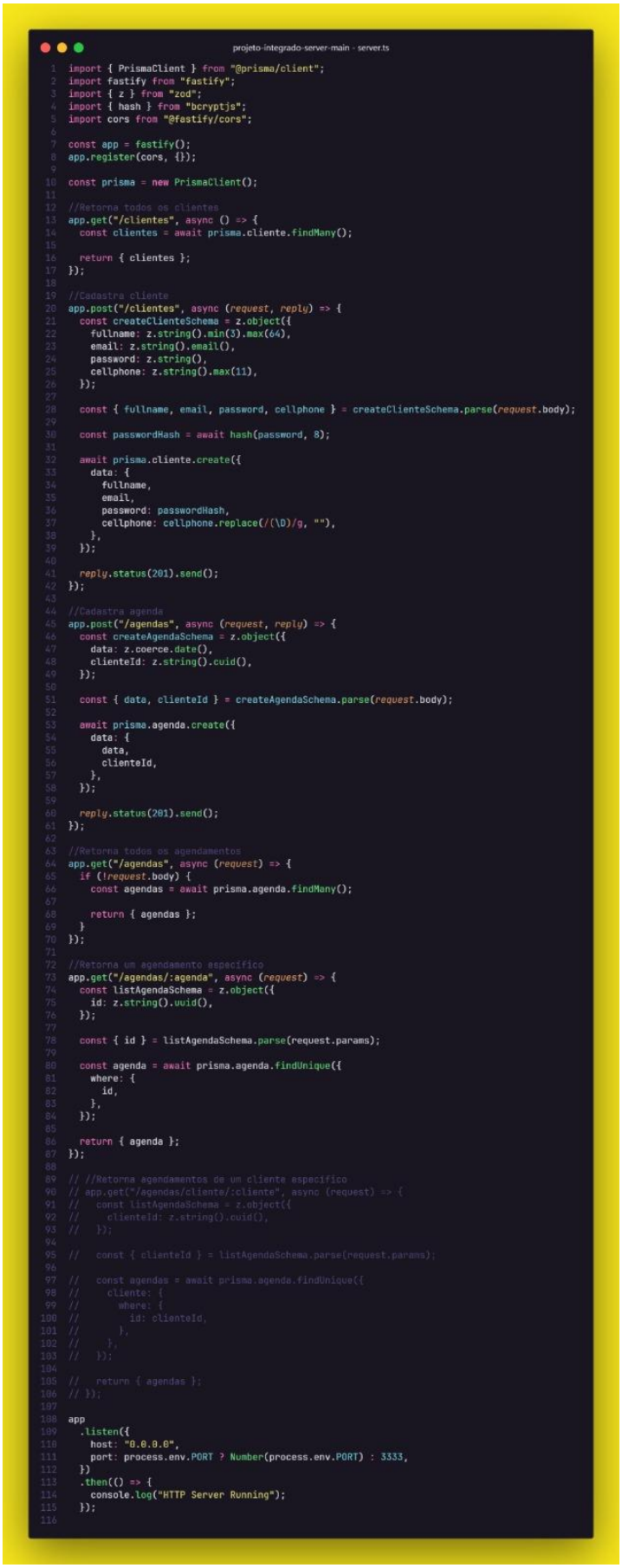

Figura 4 - Servidor escrito em TypeScript

projeto-integrado-server - server.js

```
"use strict";
var __create = Object.create;
var __defProp = Object.defineProperty;
var __getOwnPropDesc = Object.getOwnPropertyDescriptor;
var __getOwnPropNames = Object.getOwnPropertyNames;
var __getProtoOf = Object.getPrototypeOf;
var __hasOwnProp = Object.prototype.hasOwnProperty;
var __copyProps = (to, from, except, desc) => {
  if (from && typeof from === "object" || typeof from === "function") {
    for (let key of __getOwnPropNames(from))
      if (!__hasOwnProp.call(to, key) && key !== except)
        __defProp(to, key, { get: () => from[key], enumerable: !(desc = __getOwnPropDesc(from, key)) || desc.enumerable });
  }
  return to;
};
var __toESM = (mod, isNodeMode, target) => (target = mod != null ? __create(__getProtoOf(mod)) : {}, __copyProps(
  // If the importer is in node compatibility mode or this is not an ESM
  // file that has been converted to a CommonJS file using a Babel-
  // compatible transform (i.e. "__esModule" has not been set), then set
  // "default" to the CommonJS "module.exports" for node compatibility.
  isNodeMode || !mod || !mod.__esModule ? __defProp(target, "default", { value: mod, enumerable: true }) : target,
  mod
));

// src/server.ts
var import_client = require("@prisma/client");
var import_fastify = __toESM(require("fastify"));
var import_zod = require("zod");
var import_bcryptjs = require("bcryptjs");
var import_cors = __toESM(require("@fastify/cors"));
var app = (0, import_fastify.default)();
app.register(import_cors.default, {});
var prisma = new import_client.PrismaClient();
app.get("/clientes", async () => {
  const clientes = await prisma.cliente.findMany();
  return { clientes };
});
app.post("/clientes", async (request, reply) => {
  const createClienteSchema = import_zod.z.object({
    fullname: import_zod.z.string().min(3).max(64),
    email: import_zod.z.string().email(),
    password: import_zod.z.string(),
    cellphone: import_zod.z.string().max(11)
  });
  const { fullname, email, password, cellphone } = createClienteSchema.parse(request.body);
  const passwordHash = await (0, import_bcryptjs.hash)(password, 8);
  await prisma.cliente.create({
    data: {
      fullname,
      email,
      password: passwordHash,
      cellphone: cellphone.replace(/(\D)/g, "")
    }
  });
  reply.status(201).send();
});
app.post("/agendas", async (request, reply) => {
  const createAgendaSchema = import_zod.z.object({
    data: import_zod.z.coerce.date(),
    clienteId: import_zod.z.string().cuid()
  });
  const { data, clienteId } = createAgendaSchema.parse(request.body);
  await prisma.agenda.create({
    data: {
      data,
      clienteId
    }
  });
  reply.status(201).send();
});
app.get("/agendas", async (request) => {
  if (!request.body) {
    const agendas = await prisma.agenda.findMany();
    return { agendas };
  }
});
app.get("/agendas/:agenda", async (request) => {
  const listAgendaSchema = import_zod.z.object({
    id: import_zod.z.string().uuid()
  });
  const { id } = listAgendaSchema.parse(request.params);
  const agenda = await prisma.agenda.findUnique({
    where: {
      id
    }
  });
  return { agenda };
});
app.listen({
  host: "0.0.0.0",
  port: process.env.PORT ? Number(process.env.PORT) : 3333
}).then(() => {
  console.log("HTTP Server Running");
});

```

Figura 5 - Servidor após a transpilação do TypeScript para o JavaScript

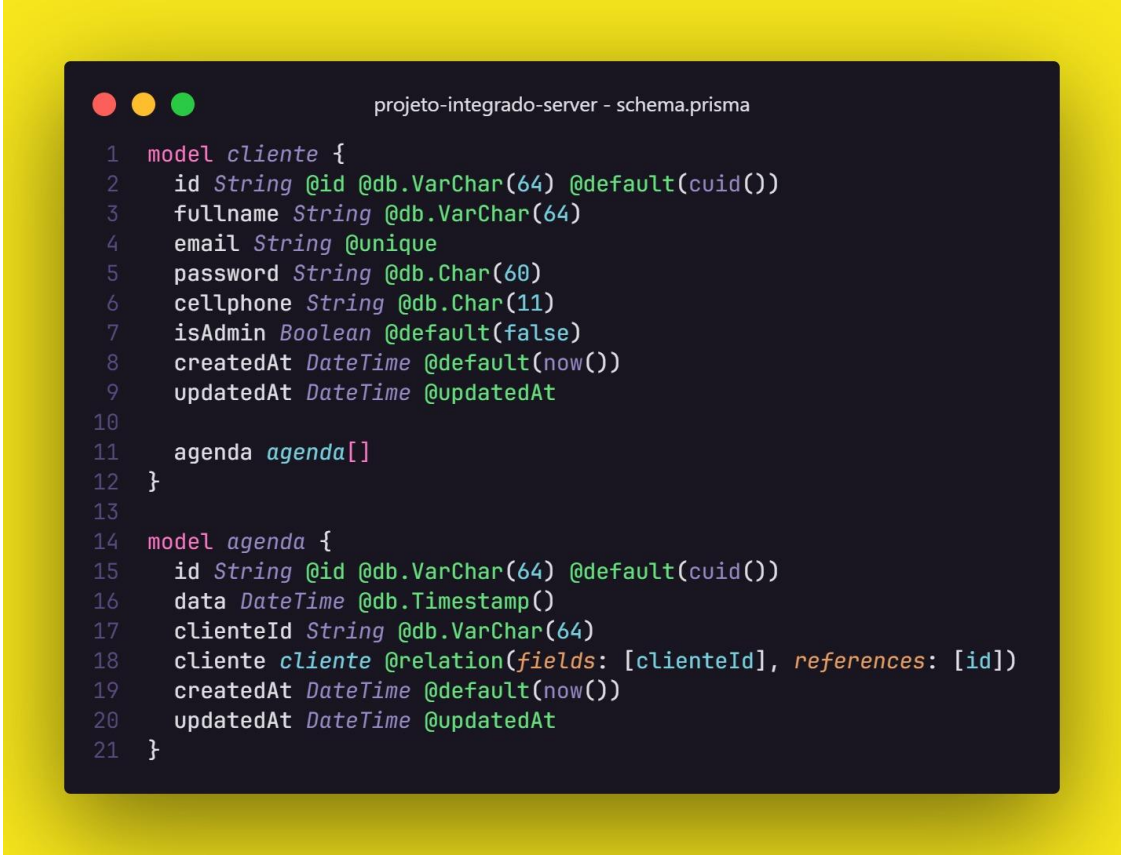

Figura 6 - Código do Prisma que é usado para gerar as tabelas do banco de dados

3.2 SISTEMAS OPERACIONAIS

Devido a escolha de utilizar somente plataformas online para publicar a aplicação, não houve uma direta preocupação com o sistema operacional que receberá a aplicação. A Vercel e Render, plataformas onde o projeto foi publicado, simplificam o processo de publicação pois além de possuírem uma forma automatizada de atualizar a aplicação quando uma alteração é feita na plataforma de versionamento escolhida, o GitHub, também possui uma função que envelopa a aplicação em um container Docker. O container remove totalmente a responsabilidade do sistema operacional de executar a aplicação, pois ela rodará em um envelope específico para o programa. Ou seja, se a aplicação a ser publicada exige um container Node.JS, o container limpo terá somente o que é necessário para executar a aplicação em Node.JS.

3.2.1 COMPONENTES DE SISTEMAS OPERACIONAIS

Devido as linguagens escolhidas para o desenvolvimento do projeto, acredita-se que não há um sistema operacional ideal para publicar o projeto, mas sim há um sistema mais prático para publicar.

Como optou-se por utilizar HTML, CSS e JQuery para o desenvolvimento do front-end e JavaScript/TypeScript do desenvolvimento do back-end, qualquer sistema operacional que fosse capaz de executar um servidor Apache e o Node.JS com os módulos utilizados seria perfeito para publicar o projeto.

Mas como o objetivo era a praticidade de publicação e ser gratuito, optou-se por plataformas online que utilizam métodos mais modernos para isso. O método moderno em questão, é o uso de containers para executar as aplicações, já que ela fica isolada do sistema operacional.

3.2.2 GERENCIAMENTO E FUNCIONALIDADES DO SISTEMA OPERACIONAL

Devido a forma de publicação do projeto a comparação de sistemas operacionais deve ser diferente, já que não houve uma hospedagem em um sistema operacional específico, mas sim em plataformas online (Nuvem).

Como não há, diretamente, um sistema operacional atuando na execução do projeto, a comparação será entre Windows, Linux e Docker.

O Docker, é um programa de virtualização de containers, ou seja ele simula um ambiente virtual e isolado do sistema operacional, garantindo que aquele ambiente tenha somente o que é necessário para executar a aplicação, excluindo qualquer chance de vício ou erro que pode vir do sistema operacional. Sua grande desvantagem, é o tempo que leva para o container ser inicializado. Pois geralmente, para economizar processamento, as hospedagens colocam o container em modo de hibernação caso ele não receba uma requisição em um tempo estipulado.

Windows e Linux, são sistemas operacionais completos que servem para executar a aplicação. Neles não tem limitação de execução de programas, exceto no Linux com programas Windows, e nem limitações que podem vir de uma virtualização. Por ser um sistema operacional completo, é possível instalar qualquer binário de execução de qualquer aplicação e até mesmo executar funções mais completas que um container não vai permitir, como a execução de um cronjob. Porém seu maior problema é o vício que pode ser carregado durante a execução de uma aplicação.

3.2.3 GERENCIAMENTO DE HARDWARE PELO SISTEMA OPERACIONAL

O projeto, front-end e back-end, e o banco de dados estão hospedados em duas plataformas diferentes, a Vercel e Render. A Vercel é responsável pela hospedagem do front-end da aplicação, pois possui uma interface simplificada e pensada para a publicação de aplicações front-end. Já a Render, é responsável pela publicação do back-end e do banco de dados, devido a robustez do serviço ofertado e simplicidade na publicação de aplicações back-end.

Visando a segurança, optou-se pelo uso da Render como local para publicar o projeto, pois possui certificados TLS gratuitos, um sistema anti DDoS (Ataque distribuído de negação de Serviço), redes privadas e um sistema automatizado de sincronização com plataformas de versionamento. Por estarmos utilizando um plano gratuito da plataforma, existem algumas limitações de uso de dados, tempo de compilação e hardware, além do banco de dados ser deletado após 90 dias de uso.

Agora, visando a praticidade não houve tanta preocupação com a segurança, por isso optamos pela utilização da Vercel para hospedar o front-end. Onde as limitações que impactam o projeto são o uso de dados e tempo de compilação do projeto.

4 CONCLUSÃO

Durante o processo de desenvolvimento deste projeto foi possível observar a aplicabilidade dos conceitos abordados durante as aulas e identificar a relevância que a criação de portfólios e sites tem para pequenos empreendedores. No decorrer das reuniões do grupo, foram discutidos quais seriam as melhores linguagens de programação a serem utilizadas e o integrante que possuía mais familiaridade conseguiu transmitir o seu conhecimento aos demais integrantes, assim foi atingido o objetivo de todos atuarem de maneira ativa nesse processo. Por fim, esse projeto demonstrou ser de grande aplicabilidade e possível de ser reproduzido em outros segmentos que necessitam de aproximação entre o cliente e o produto.

5 REFERÊNCIAS

ALVES, Andréia et al. **Comunicação Empresarial:** um estudo sobre Marketing Digital. Intercom–Sociedade Brasileira de Estudos Interdisciplinares da Educação. Disponível em <https://www.portalintercom.org.br/anais/sudeste2014/resumos/R43-0308-1.pdf.> Acessado em 11 de jun de 2023.

DA SILVA, Adriel Thainan Izidoro et al. **A IMPORTÂNCIA DO BANCO DE DADOS PARA O CONTROLE EM UMA INDÚSTRIA**. Tekhne e Logos, v. 10, n. 2, p. 76-89, 2019. Disponível em < http://revista.fatecbt.edu.br/index.php/tl/article/view/590.> Acessado em 11 de jun de 2023.

6 ANEXOS

Design do projeto : PÁGINA DO SITE – Figma

Repositório do Projeto Web : RafaM0rais/projeto-integrado-web (github.com)

Repositório do Projeto Server: RafaM0rais/projeto-integrado-server (github.com)

Projeto Disponível em: https://projeto-integrado-web.vercel.app/