

Centro Estadual de Educação Tecnológica Paula Souza
ETEC DA ZONA LESTE
Ensino Médio com Habilitação Profissional de Técnico em
Desenvolvimento de Sistemas

Matheus Curci Romano
Rebeca Matewanga Maria Kamalandua
Revellin Mendes Ferreira

VIA SCHOLAE: Sistema de Acompanhamento de Vans Escolares.

SÃO PAULO-SP
2023

**Matheus Curci Romano
Rebeca Matewanga Maria Kamalandua
Revellin Mendes Ferreira**

VIA SCHOLAE: Sistema de Acompanhamento de Vans Escolares.

Trabalho de Conclusão de curso apresentado ao Curso Ensino Médio com Habilitação Profissional de Técnico em Desenvolvimento de Sistemas da ETEC ZONA LESTE, orientado pelo Prof. Jeferson Roberto de Lima, como requisito parcial para obtenção do título de Técnico em Desenvolvimento de Sistemas.

**São Paulo
2023**

VIA SCHOLAE: Sistema de Acompanhamento de Vans Escolares.

Trabalho de Conclusão de Curso
apresentado ao curso técnico em
administração integrado ao ensino médio da
ETEC Zona Leste orientado pela professora
Ma. Marcia Macário Dos Santos como
requisito parcial para a obtenção do título
Técnico em Administração.

Aprovado em: __/__/__

Banca Examinadora

Professor
ETEC da Zona Leste

Professor
ETEC da Zona Leste

Professor
ETEC da Zona Leste

Este trabalho é dedicado, em primeiro lugar, a Deus, pela força e coragem durante toda esta longa caminhada, as nossas famílias e amigos pelo incentivo e pelo apoio constante.

AGRADECIMENTOS

Expressamos nossa gratidão aos professores e coordenadores, Jeferson Roberto de Lima e Rogério Bezerra do curso de Desenvolvimento de Sistemas, que foram tão importantes na nossa vida acadêmica e no desenvolvimento desta monografia.

*“É a educação que faz o futuro parecer um
lugar de esperança e transformação.”
(Marianna Moreno)*

RESUMO

Sistema com a finalidade de flexibilizar o acesso ao serviço de transporte escolar e melhorar a intercomunicação entre os stakeholders, reorganizando a maneira como ocorre o gerenciamento das vans, além da implementação de medidas de segurança de modo a preservar a integridade, tanto dos estudantes, como dos motoristas; as funcionalidades responsáveis por suprir esses problemas serão notificações em tempo real sobre tempo de chegada, acidentes, emergências, entre outras informações pertinentes sobre o estado das vans e seus respectivos passageiros, além da disposição de informações como a rota dos veículos e por fim um chat para comunicação direta dos pais com os motoristas.

Palavras-chave: Motorista; Vans; Segurança; Transparência; Intercomunicação.

ABSTRACT

System with the purpose of making access to the school transport service more flexible and improving intercommunication between stakeholders, reorganizing the way in which the management of vans occurs, in addition to the implementation of safety measures in order to preserve the integrity of both students and drivers; The functionalities responsible for solving these problems will be real-time notifications about arrival time, accidents, emergencies, among other pertinent information about the status of the vans and their respective passengers, in addition to the provision of information such as the route of the vehicles and finally a chat for direct communication between parents and drivers.

Keywords: Driver; Vans; Security; Transparency; Intercom.

LISTA DE FIGURAS

- Figura 1 - Cronograma das etapas do trabalho de conclusão de curso 15
- Figura 2 - Código base HTML 19
- Figura 3 - Formulário de contato 19
- Figura 4 - Código base CSS 20
- Figura 5 - Formulário de contato com CSS 21
- Figura 6 - Função de alerta JavaScript 22
- Figura 7 - Resultado função JavaScript 22
- Figura 8 - Código base em React Native 23
- Figura 9 - Continuação código básico React Native 23
- Figura 10 - Estilização no código base de React Native 26
- Figura 11 - Resultado do Formulário de Contato 29
- Figura 12 - Exemplo de Banco de Dados no SQLite 31
- Figura 13 - Exemplo de Diagrama de Caso de Uso 32
- Figura 14 - Exemplo de Diagrama de Classe 35
- Figura 15 - Exemplo de Diagrama de Sequência 38
- Figura 16 - Exemplo de Diagrama de Atividade 41
- Figura 17 - Exemplo do NPM para instalação de um modulo 45
- Figura 18 - Exemplo do NPM para iniciar um servidor de desenvolvimento 46

LISTA DE ABREVIATURAS E SIGLAS

Application Programming Interface (API)

Cascading Style Sheet (CSS)

Engenharia de Software Auxiliada por Computador (CASE)

Document Object Model (DOM)

Hypertext Markup Language (HTML)

Integrated Development Environment (IDE)

JavaScript (JS)

Modelo de Entidade Relacionamento (MER)

Node Package Manager (NPM)

Transporte Escolar Gratuito (TEG)

Unified Modeling Language (UML)

Visual Studio Code (VS Code)

SUMÁRIO

1. INTRODUÇÃO.....	12
1.1 Problema de Pesquisa	13
1.2 Hipótese	13
1.3 Objetivos da Aplicação.....	13
1.4 Objetivo Geral	13
1.5 Objetivos Específicos.....	13
1.6 Justificativa.....	14
1.7 Metodologia.....	14
1.8 Cronograma	15
2. REFERENCIAL TEÓRICO	15
2.1 Transporte escolar no Brasil	15
2.2 Como é implementado em escolas públicas	16
2.3 Como é implementado em escolas particulares.....	16
2.4 O impacto do transporte escolar no Brasil	17
2.5 O impacto da aplicação no transporte escolar	17
3. Tecnologias Utilizadas.....	18
3.1 HyperText Markup Language.....	18
3.2 Cascading Style Sheet.....	19
3.3 JavaScript	21
3.4 React Native.....	22
3.5 Node.js	29
3.6 Bootstrap.....	29
3.7 Banco de Dados.....	30
3.8 SQLite	30
3.9 Astah Community	31
3.10 Unified Modeling Language.....	31
3.10.1 Diagrama de Caso de Uso	32
3.10.2 Diagrama de Classe	35
3.10.3 Diagrama de Sequência	37
3.10.4 Diagrama de Atividade	40
3.11 Application Programming Interface	43
3.11.1 Mapbox.....	43

3.12	Figma	44
3.13	Wireframe.....	44
3.14	Visual Studio Code.....	44
3.15	Expo.....	45
3.16	Node Package Manager	45
4.	REFERÊNCIAS	46

1. INTRODUÇÃO

Este trabalho aborda o desenvolvimento de uma aplicação destinada ao acompanhamento das vans escolares, visando flexibilizar o acesso ao serviço de transporte escolar e aprimorar a comunicação entre os pais e os motoristas. O sistema implementará medidas de segurança para preservar a integridade tanto dos estudantes quanto dos motoristas, tornando o trajeto escolar mais seguro e tranquilo para todos os envolvidos.

As funcionalidades planejadas para o aplicativo serão projetadas para suprir os problemas enfrentados no transporte escolar. Isso incluirá notificações em tempo real sobre o tempo de chegada, saída, emergências e outras informações pertinentes, disponibilizando informações detalhadas sobre a rota dos veículos, proporcionando aos responsáveis maior tranquilidade e controle sobre o trajeto de seus filhos. Também está prevista a inclusão de um chat que permitirá a comunicação direta entre os pais e os motoristas, possibilitando uma forma rápida e eficaz de esclarecer dúvidas e contribuindo para uma comunicação segura.

Ademais, será disponibilizado um site explicativo da aplicação, fornecendo informações detalhadas sobre o funcionamento do serviço, políticas de segurança e privacidade. Por meio do site, os pais e motoristas poderão acessar todas as informações necessárias sobre o serviço de transporte escolar e esclarecer suas dúvidas.

Segundo o Detran do Estado de São Paulo (2024), os genitores devem averiguar a estabilidade legal do veículo e do motorista antes de contratar o serviço de transporte escolar para seus filhos.

Diante da necessidade de oferecer uma solução moderna e eficiente para um problema recorrente enfrentado pelos responsáveis pelas crianças, proporcionando um transporte seguro e confiável dos alunos para a escola, além da crescente demanda por praticidade e segurança, um aplicativo dedicado pode suprir essas necessidades, otimizando o uso dos recursos de transporte e contribuindo para a redução do trânsito nas áreas escolares. Ademais, a implementação desse aplicativo

representa uma oportunidade de negócio promissor, alinhada às tendências de mobilidade urbana e tecnologia.

1.1 Problema de Pesquisa

Consoante BALARDIM (2019), no mercado, especificamente nas redes de educação, há um viés que tende a ser adepto de uma supervisão do transporte escolar pela tecnologia, pois seus benefícios são auto evidentes.

Como a integração da tecnologia pode auxiliar e melhorar a segurança e qualidade do transporte escolar?

1.2 Hipótese

A tecnologia pode ser uma solução eficiente para auxiliar na localização das vans escolares em tempo real e a otimização das rotas, proporcionando mais segurança aos usuários e comunicação direta com os pais.

1.3 Objetivos da Aplicação

A seguir serão apresentados os objetivos gerais e específicos do trabalho.

1.4 Objetivo Geral

Desenvolvimento de um sistema escolar que proporcione aos responsáveis maior facilidade de localizar a van escolar por meio de um aplicativo móvel com o propósito de assegurar a segurança e eficiência do transporte dos alunos desde a saída de sua casa para a escola e vice e versa, oferecendo um chat de comunicação direta entre o responsável da criança e o motorista com notificações em tempo real de chegada, saída e emergências. Ademais teremos um site informativo do projeto, resultando em uma solução moderna e conveniente para todos.

1.5 Objetivos Específicos

- Definir as tecnologias que serão aplicadas no projeto;
- Criar um aplicativo que acompanhe a rota das vans escolares.
- Desenvolver um site demonstrativo do projeto.
- Registrar o acompanhamento em tempo real do veículo.

- Aplicar comunicação direta com os pais.
- Implementar alertas e notificações de segurança.
- Restringir a zona de localização específica para a aplicação.
- Apontar a segurança e eficácia da aplicação.

1.6 Justificativa

Segundo o Detran do Estado de São Paulo (2024), os genitores devem averiguar a estabilidade legal do veículo e motorista antes de firmar o serviço de transporte escolar dos seus filhos.

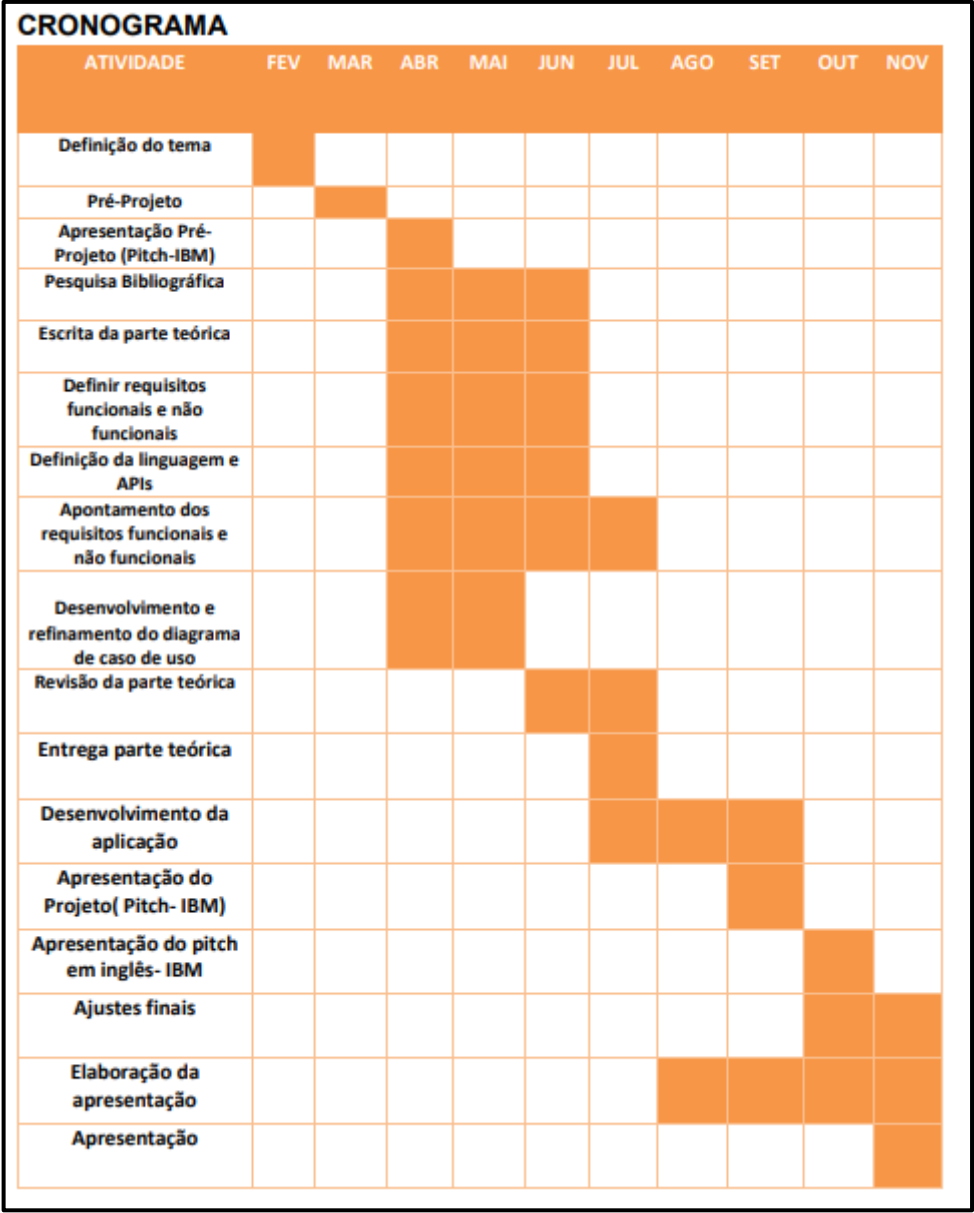
As necessidades de oferecer uma solução moderna e eficiente para um problema recorrente enfrentado pelos responsáveis das crianças, oferecer um transporte seguro e confiável dos alunos para a escola, além da crescente demanda por praticidade e segurança, um aplicativo dedicado pode suprir essas necessidades, otimizando o uso dos recursos de transporte e contribuindo para a redução do trânsito nas áreas escolares. Ademais, a implementação desse aplicativo representa uma oportunidade de negócio promissora, alinhada às tendências de mobilidade urbana e tecnologia.

1.7 Metodologia

- Pesquisa: Livros, artigos, sites e para a construção do referencial teórico sobre os temas abordados neste trabalho.
- Pesquisa Descritiva: Estudo de caso de softwares voltados ao transporte público e localização.
- Teste piloto do formulário: Por meio de um formulário que será compartilhado para vários motoristas de vans escolares e pais antes da finalização do projeto, ajudando a identificar problemas.
- Público e amostra: Pais de crianças de Escolas do 1º ao 6º ano do Ensino Fundamental e motoristas de vans escolares.
- Delimitação do bairro: São Miguel Paulista.

1.8 Cronograma

Figura 1 - Cronograma das etapas do trabalho de conclusão de curso



Fonte: Autoria Própria, 2024.

2. REFERENCIAL TEÓRICO

2.1 Transporte escolar no Brasil

Em conformidade com o G1 (2023), a região de Itaquapecetuba tem enfrentado um grande problema em relação ao transporte escolar, onde muitos alunos acabam sofrendo por falta de transporte.

O transporte escolar no Brasil é fundamental para garantir que milhões de estudantes sejam capazes de frequentar escolas, sejam elas localizadas nas áreas urbanas ou rurais. Vendo em vista como o país tem uma desigualdade social muito grande, e sem o transporte escolar vários alunos ficaram sem o deslocamento para a escola e acaba atrapalhando totalmente o acesso à educação. E é notável que o transporte escolar enfrenta outros problemas, como o a más condições de conservação dos veículos e falta de supervisão, que podem gerar riscos à segurança das crianças.

2.2 Como é implementado em escolas públicas

De acordo com a Prefeitura do Estado de São Paulo (2023), O TEG, Programa de Transporte Escolar Municipal Gratuito, foi instituído com o objetivo de assegurar aos alunos o acesso às escolas da rede municipal sem qualquer custo.

Nas escolas públicas existe um programa chamado TEG (O Transporte Escolar Gratuito) é um programa beneficente criado pelo Governo do Estado de São Paulo para proporcionar aos estudantes de ensino municipal transporte de forma gratuita. O transporte conduz os alunos de suas residências até a instituições aliadas ao programa, o parecer do responsável é de obrigatoriedade a sua presença no momento do embarque dela no transporte e no retorno. Para a criança ser contemplada com esse benefício, ela deve morar a partir de 1,5 quilômetros da instituição municipal onde estar matriculada e ter até 11 anos de idade.

2.3 Como é implementado em escolas particulares

O transporte escolar em escolas privadas desempenha um papel fundamental para diversas famílias, garantindo a chegada segura e pontual dos alunos às instituições de ensino. Geralmente, essa responsabilidade é delegada a empresas especializadas, que disponibilizam veículos e condutores qualificados, possibilitando que as escolas foquem na educação. Contudo, algumas escolas optam por manter sua própria frota visando um maior controle e personalização do serviço. Em ambas as situações, a segurança dos estudantes é priorizada, contando com motoristas capacitados e veículos bem cuidados. Adicionalmente, as escolas estabelecem diretrizes claras para garantir o comportamento adequado dos alunos durante o trajeto. Embora o transporte escolar seja um serviço extra sujeito a taxas separadas,

sua relevância para a experiência educacional é inquestionável, oferecendo comodidade e tranquilidade às famílias.

2.4 O impacto do transporte escolar no Brasil

Se compararmos a disponibilidade de acesso da população estudantil camponesa e a urbana aos centros de ensino veremos que os educandos da cidade têm maior acessibilidade, pois estão mais próximos da instituição, contam com estradas melhores e transportes em melhor condição de trafegar. Desta forma a igualdade de condições de acesso e permanência na escola é diferente. (ARNT e SILVA, 1998, p. 3).

O transporte escolar no Brasil tem uma função essencial em assegurar que estudantes em regiões distantes ou sem transporte público tenham acesso à educação. Seus benefícios abrangem desde a diminuição da evasão escolar e o aprimoramento do desempenho acadêmico até a promoção da segurança, a redução do congestionamento e o impacto econômico positivo. Contudo, é imprescindível que se estabeleçam normas e fiscalização adequadas para garantir a eficiência e a segurança desse tipo de transporte.

2.5 O impacto da aplicação no transporte escolar

A aplicação proposta pelo projeto poderá proporcionar e auxiliar na maior segurança e fiscalização do trajeto das vans escolares, desde o caminho da casa do contratante para a escola até sua volta. Essa iniciativa visa suprir os problemas encontrados com a pesquisa e amplas entrevistas realizadas, e será a iniciativa de uma possível solução para os problemas encontrados.

Além da segurança das crianças, a veracidade e transparência do trabalho do condutor, a aplicação também visa aumentar o contato e conversa direta entre o contratante e o contratado, viabilizando e flexibilizando o acesso do prestador de serviço ao responsável e o contato entre eles.

Além disso, a aplicação também contará com notificações e alertas de segurança, como por exemplo, de chegada e saída, atrasos ou acidentes ocorridos durante o percurso. Foi noticiado pelo G1(2023) um acidente envolvendo uma van escolar e um carro, o acidente deixou crianças e o motorista feridos. Os pais dessas crianças só

foram informados do incidente após algumas horas, causando desespero e até mesmo complicações de saúde. Com a implementação das notificações, os responsáveis ficarão mais seguros e menos preocupados, sendo avisados imediatamente sobre qualquer intercorrência, auxiliando maior credibilidade e transparência sobre qualquer Incidente.

3. TECNOLOGIAS UTILIZADAS

Para o desenvolvimento da aplicação, foram utilizadas as seguintes tecnologias:

3.1 HyperText Markup Language

Em conformidade com Samy (2008), HyperText Markup Language (Html) é uma linguagem de marcação de hipertexto, sua maior característica seria a facilidade de fazer integração com outros documentos web.

Segundo Pereira e Poupa (2011) a linguagem HTML tem sua estrutura que utiliza tags, que identificam a função e o conteúdo de cada elemento da linguagem. Tendo como suas principais tags:

- **<!DOCTYPE html>**. Declara o tipo de documento como HTML5.
- **<html lang="en">** Determina a raiz do documento e especifica o idioma como inglês (en).
- **<head>** Contém informações sobre o documento, como meta tags e o título.
- **<meta charset="UTF-8">** Define o conjunto de caracteres como UTF-8 para suportar caracteres especiais.
- **<meta name="viewport" content="width=device-width, initial-scale=1.0">** Declara a viewport para dispositivos móveis.
- **<title>Document</title>** Decreta o título da página.
- **<body>** Inclui todo o conteúdo visível do documento.

A seguir, teremos um exemplo básico do código fonte em HTML:

Figura 2 - Código base HTML

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="UTF-8">
5  <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2Bxj"
7  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3tHB60NlkxXC5s9FDVZLEaAA53NDzOxhy9Gk1s1k1eH7l"
8  <link rel="stylesheet" href="style.css">
9  <title>Formulário</title>
10 </head>
11 <body>
12 <div class="container">
13   <h2>Formulário de Contato</h2>
14   <form id="contactForm">
15     <label for="name">Nome:</label>
16     <input type="text" id="name" name="name" required>
17     <label for="email">Email:</label>
18     <input type="email" id="email" name="email" required>
19     <label for="message">Mensagem:</label>
20     <textarea id="message" name="message" rows="4" required></textarea>
21     <input type="submit" value="Enviar">
22   </form>
23 </div>
24 </body>
25 </html>

```

Fonte: Autoria Própria, 2024.

Na figura 2, foi criado um formulário de contato obtendo tem campos para preenchimento sendo eles: Nome, E-mail e Mensagem.

Após o salvamento do código fonte e o acesso a um navegador da internet aparecera um formulário simples visto que ainda não há nenhuma estilização nele. A figura abaixo exhibe como o código se apresenta:

Figura 3 - Formulário de contato

Fonte: Autoria Própria, 2024.

3.2 Cascading Style Sheet

Como destaca Samy (2007), Cascading Style Sheet (CSS) ou seja, folhas de estilo em cascata serve para trazer estilo ao código web, adicionando elementos para o código permitindo uma melhor visualização do código web.

De acordo com Quierelli (2012), para que o desenvolvedor consiga ter um código mais limpo ele deve acionar o CSS visto que ele se aplica a essa função deixando o código fonte menor.

Nas palavras de Jobstraibizer (2009), usar o CSS traz muitos benefícios, sendo assim estilizar o layout a partir de uma única folha de estilo, por exemplo: textos, imagens e entre outros.

A seguir, teremos um exemplo da estilização em CSS no formulário de contato:

Figura 4 - Código base CSS

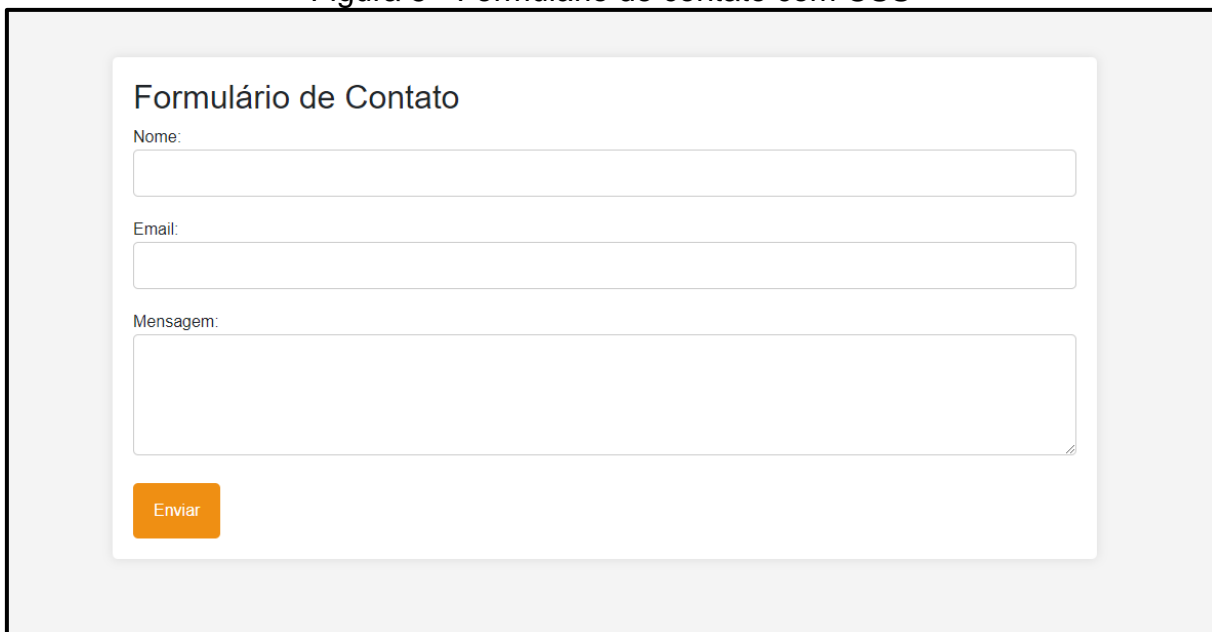
```
# style.css > input[type="submit"]:hover
1  body {
2      font-family: Arial, sans-serif;
3      background-color: #f4f4f4;
4      margin: 0;
5      padding: 0;
6  }
7
8  .container {
9      width: 50%;
10     margin: 50px auto;
11     background-color: #fff;
12     padding: 20px;
13     border-radius: 5px;
14     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
15 }
16
17 input[type="text"],
18 input[type="email"],
19 textarea {
20     width: 100%;
21     padding: 10px;
22     margin-bottom: 20px;
23     border: 1px solid #ccc;
24     border-radius: 5px;
25 }
26
27 input[type="submit"] {
28     background-color: #EF8F13;
29     color: #fff;
30     border: none;
31     padding: 15px 20px;
32     cursor: pointer;
33     border-radius: 5px;
34 }
35
36 input[type="submit"]:hover {
37     background-color: #EF8F13;
38 }
```

Fonte: Autoria Própria, 2024.

Na figura 4, foi implementada toda a estilização do formulário de contato.

Após o salvamento do código básico de CSS ao acessar um navegador de internet será exibido o formulário já estilizado. A figura abaixo representa o resultado da estilização:

Figura 5 - Formulário de contato com CSS

A imagem mostra um formulário de contato estilizado com CSS. O formulário é branco e está centralizado em um fundo cinza claro. No topo, o título "Formulário de Contato" é exibido em uma fonte preta. Abaixo dele, há três campos de entrada: "Nome:" com um campo de texto único, "Email:" com um campo de texto único, e "Mensagem:" com um campo de texto de área. Cada campo tem uma borda cinza clara. No canto inferior esquerdo do formulário, há um botão laranja com o texto "Enviar" em branco.

Fonte: Autoria Própria, 2024.

3.3 JavaScript

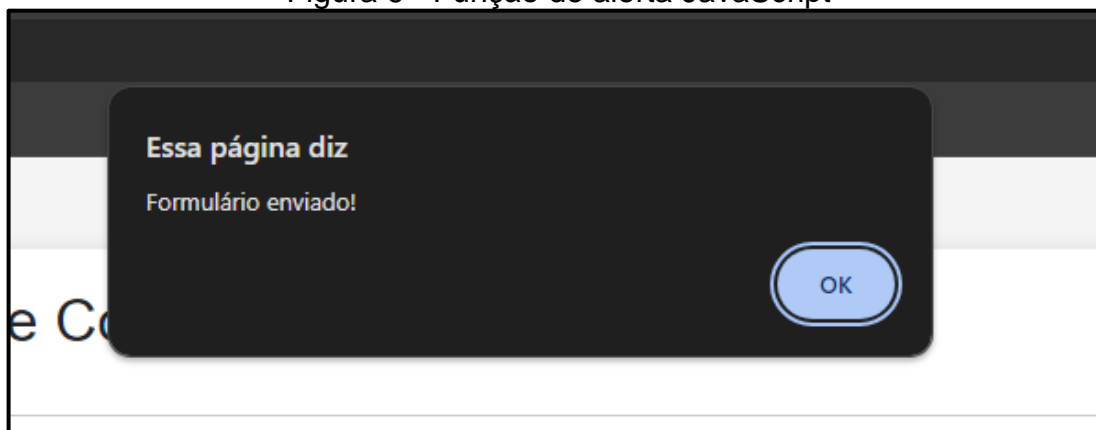
Como aponta Prescott (2016), você nunca poderá confundir Java com Javascript, o Java é uma complexa linguagem de programação, já o Javascript é apenas uma linguagem de scripts, estando mais envolvida com a linguagem de programação C.

Segundo Samy (2010), a criação da linguagem Javascript foi feita para rodar ao lado do cliente, seu funcionamento depende diretamente da sua hospedagem e das funcionalidades programadas.

Para Flanagan (2012), o Javascript é uma linguagem de programação Web, que a maioria dos sites que existem hoje utilizam ele e todos os navegadores modernos.

A seguir, teremos um exemplo da função de notificação em Javascript no formulário de contato:

Figura 6 - Função de alerta JavaScript



Fonte: Autoria Própria, 2024.

Na figura 6, foi implementada uma função em Javascript com a finalidade de que quando enviado o formulário apareça uma notificação no canto da tela.

Após o salvamento da função em Javascript e envio do formulário aparecerá uma notificação no canto da tela informando que o formulário foi enviado. A figura abaixo representa a notificação em Javascript:

Figura 7 - Resultado função JavaScript

```
// Script JavaScript
document.getElementById("contactForm").addEventListener("submit", function(event) {
    event.preventDefault();
    alert("Formulário enviado!");
});
```

Fonte: Autoria Própria, 2024.

3.4 React Native

Conforme Pin (2024), o React Native é uma biblioteca JavaScript que foi desenvolvida pelo Facebook tendo facilidade na criação de vários aplicativos móveis multiplataforma.

Consoante Escudelar e Pinho (2020), o React é a base de toda tecnologia que envolve o React Native, mas você não é obrigado ter esse conhecimento.

A Figura a seguir, representa um exemplo básico de código em react native:

Figura 8 - Código base em React Native

```

app > (tabs) > index.tsx > default
1 import React, { useState } from 'react';
2 import { View, Text, TextInput, Button } from 'react-native';
3 import styles from './formularioStyles'; // Importando os estilos
4
5 const Formulario = () => {
6   const [nome, setNome] = useState('');
7   const [sobrenome, setSobrenome] = useState('');
8   const [email, setEmail] = useState('');
9   const [cidade, setCidade] = useState('');
10  const [estado, setEstado] = useState('');
11
12  const handleSubmit = () => {
13    // Aqui você pode adicionar a lógica para lidar com os dados do formulário
14    console.log('Dados do formulário:', { nome, sobrenome, email, cidade, estado });
15  };
16
17  return (
18    <View style={styles.container}>
19      <View style={styles.formWrapper}>
20        <Text style={styles.label}>Nome</Text>
21        <TextInput
22          style={styles.input}
23          value={nome}
24          onChangeText={setNome}
25          placeholder="Digite seu nome"
26        />
27        <Text style={styles.label}>Sobrenome</Text>
28        <TextInput
29          style={styles.input}
30          value={sobrenome}
31          onChangeText={setSobrenome}
32          placeholder="Digite seu sobrenome"
33        />
34        <Text style={styles.label}>Email</Text>
35        <TextInput
36          style={styles.input}
37          value={email}
38          onChangeText={setEmail}
39          placeholder="Digite seu email"
40          keyboardType="email-address"
41        />

```

Fonte: Autoria Própria, 2024.

Na figura 8, um formulário de contato foi criado com 5 campos: Nome Sobrenome, Email, Cidade e Estado.

A figura abaixo mostra a continuação do código base feito em React Native:

Figura 9 - Continuação código básico React Native

```

app > (tabs) > index.tsx > Formulario
5 const Formulario = () => {
42   <Text style={styles.label}>Cidade</Text>
43   <TextInput
44     style={styles.input}
45     value={cidade}
46     onChangeText={setCidade}
47     placeholder="Digite sua cidade"
48   />
49   <Text style={styles.label}>Estado</Text>
50   <TextInput
51     style={styles.input}
52     value={estado}
53     onChangeText={setEstado}
54     placeholder="Digite seu estado"
55   />
56   <Button title="Enviar" onPress={handleSubmit} />
57 </View>
58 </View>
59 );
60 };
61
62 export default Formulario;

```

Fonte: Autoria Própria, 2024.

Explicação do código acima:

- **import React, { useState } from 'react':** Importa o módulo React e a função `useState` do React para permitir o uso de estado em componentes funcionais.
- **import { View, Text, TextInput, Button } from 'react-native':** Importa os componentes essenciais do React Native necessários para construir a interface do usuário, como `'View'` (contêiner), `'Text'` (texto), `'TextInput'` (campo de entrada de texto) e `'Button'` (botão).
- **import styles from './formularioStyles':** Importa os estilos definidos em um arquivo separado chamado `'formularioStyles'` para aplicá-los aos elementos do formulário.
- **const Formulario = () => {:** Define um componente de função chamado `Formulario`. Este é um componente funcional que encapsula o formulário e seu estado.
- **const [nome, setNome] = useState('):** Define um estado chamado `nome` e uma função `setNome` para atualizar esse estado. O estado inicial é uma string vazia. Isso é repetido para os estados `sobrenome`, `email`, `cidade` e `estado`.
- **useState('):** Utiliza o hook `useState` do React para criar um estado. Ele retorna um array onde o primeiro elemento é o valor atual do estado (`nome`, `sobrenome`, etc.) e o segundo elemento é uma função para atualizar esse estado (`setNome`, `setSobrenome`, etc.).
- **const handleSubmit = () => {:** Define uma função chamada `handleSubmit`. Esta função será chamada quando o formulário for submetido.
- **<View style={styles.container}>:** Renderiza uma `'View'` que usa os estilos definidos em `'formularioStyles.container'`. Esta `'View'` envolve todo o conteúdo do formulário.
- **<View style={styles.formWrapper}>:** Renderiza uma segunda `'View'` que usa os estilos definidos em `'formularioStyles.formWrapper'`. Esta `'View'` envolve todo o formulário em si.
- **<Text style={styles.label}>Nome</Text>:** Renderiza um componente `'Text'` com o estilo definido em `'formularioStyles.label'`. Este é um rótulo para o campo de entrada de texto do nome.

- **<TextInput style={styles.input} value={nome} onChangeText={setNome} placeholder="Digite seu nome" />:** Renderiza um componente 'TextInput' que permite ao usuário inserir texto. Ele usa os estilos definidos em 'formularioStyles.input'. O valor do campo de texto é definido como 'nome' e é atualizado pela função 'setNome'. O atributo 'placeholder' fornece um texto de orientação para o usuário.
- **<Text style={styles.label}>Sobrenome</Text>:** Renderiza um componente 'Text' com o estilo definido em 'formularioStyles.label'. Este é um rótulo para o campo de entrada de texto do sobrenome.
- **<TextInput style={styles.input} value={sobrenome} onChangeText={setSobrenome} placeholder="Digite seu sobrenome" />:** Renderiza um componente 'TextInput' que permite ao usuário inserir texto. Ele usa os estilos definidos em 'formularioStyles.input'. O valor do campo de texto é definido como 'sobrenome' e é atualizado pela função 'setSobrenome'. O atributo 'placeholder' fornece um texto de orientação para o usuário.
- **<Text style={styles.label}>Email</Text>:** Renderiza um componente 'Text' com o estilo definido em 'formularioStyles.label'. Este é um rótulo para o campo de entrada de texto do email.
- **<TextInput style={styles.input} value={email} onChangeText={setEmail} placeholder="Digite seu email" keyboardType="email-address" />:** Renderiza um componente 'TextInput' que permite ao usuário inserir texto. Ele usa os estilos definidos em 'formularioStyles.input'. O valor do campo de texto é definido como 'email' e é atualizado pela função 'setEmail'. O atributo 'placeholder' fornece um texto de orientação para o usuário. Ele também especifica o tipo como endereço de e-mail.
- **<Text style={styles.label}>Cidade</Text>:** Renderiza um componente 'Text' com o estilo definido em 'formularioStyles.label'. Este é um rótulo para o campo de entrada de texto da cidade.
- **<TextInput style={styles.input} value={cidade} onChangeText={setCidade} placeholder="Digite sua cidade" />:** Renderiza um componente 'TextInput' que permite ao usuário inserir texto. Ele usa os estilos definidos em 'formularioStyles.input'. O valor do campo de texto

é definido como 'cidade' e é atualizado pela função 'setCidade'. O atributo 'placeholder' fornece um texto de orientação para o usuário.

- **Text style={styles.label}>Estado</Text>:** Renderiza um componente 'Text' com o estilo definido em 'formularioStyles.label'. Este é um rótulo para o campo de entrada de texto do estado.
- **<TextInput style={styles.input} value={estado} onChangeText={setEstado} placeholder="Digite seu estado" />:** Renderiza um componente 'TextInput' que permite ao usuário inserir texto. Ele usa os estilos definidos em 'formularioStyles.input'. O valor do campo de texto é definido como 'estado' e é atualizado pela função 'setEstado'. O atributo 'placeholder' fornece um texto de orientação para o usuário.
- **<Button title="Enviar" onPress={handleSubmit} />:** Renderiza um botão com o título "Enviar". Quando o botão é pressionado, a função 'handleSubmit' é chamada.
- **export default Formulario:** Exporta o componente Formulario.

Figura 10 - Estilização no código base de React Native

```

app > (tabs) > JS formularioStyles.js > [0] styles > [0] container > [0] backgroundColor
1  import { StyleSheet } from 'react-native';
2
3  const styles = StyleSheet.create({
4    container: {
5      flex: 1,
6      justifyContent: 'center',
7      alignItems: 'center',
8      paddingHorizontal: 20,
9      backgroundColor: '#969696',
10   },
11   formWrapper: {
12     width: '80%',
13     backgroundColor: 'fff',
14     padding: 20,
15     borderRadius: 10,
16     shadowColor: 'black',
17     shadowOffset: {
18       width: 0,
19       height: 2,
20     },
21     shadowOpacity: 0.25,
22     shadowRadius: 3.84,
23     elevation: 5,
24   },
25   label: {
26     fontSize: 16,
27     fontWeight: 'bold',
28     marginBottom: 5,
29   },
30   input: {
31     width: '100%',
32     height: 40,
33     borderWidth: 1,
34     borderColor: 'ccc',
35     borderRadius: 5,
36     paddingHorizontal: 10,
37     marginBottom: 10,
38   },
39 });
40
41 export default styles;

```

Fonte: Autoria Própria, 2024

Explicação do código acima:

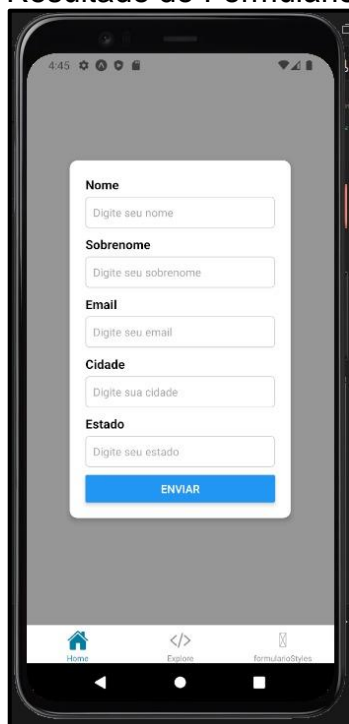
- **import { StyleSheet } from 'react-native':** importa a biblioteca 'StyleSheet' da biblioteca 'react-native'.
- **const styles = StyleSheet.create({ ... }):** Esta criando um objeto 'styles' usando o método 'create()' da classe 'StyleSheet'.
- **container: { ... }:** Define os estilos para um container específico.
- **flex: 1:** Define a propriedade flex como 1, o container irá se expandir para preencher todo o espaço disponível.
- **justifyContent: 'center':** Centraliza os itens verticalmente do container.
- **alignItems: 'center':** Centraliza os itens horizontalmente.
- **paddingHorizontal: 20:** Adiciona um preenchimento horizontal de 20 pixels ao container.
- **backgroundColor: '#969696':** Define a cor de fundo do container como cinza escuro.
- **formWrapper: { ... }:** Define os estilos para o container que envolve o formulário.
- **width: '80%':** Define a largura como 80% da largura do container.
- **backgroundColor: '#fff':** Define a cor de fundo do componente como branco.
- **padding: 20:** Adiciona um preenchimento interno de 20 pixels ao redor do componente.
- **borderRadius: 10:** Define o raio dos cantos do componente como 10 pixels.
- **shadowColor: '#000':** Define a cor da sombra projetada pelo componente como preto.
- **shadowOffset: { width: 0, height: 2 }:** Define o deslocamento da sombra em relação ao componente. Neste caso, a sombra não será deslocada horizontalmente e será deslocada verticalmente para baixo em 2 pixels.
- **shadowOpacity: 0.25:** Define a opacidade da sombra projetada pelo componente como 0.25.
- **shadowRadius: 3.84:** Define o raio da sombra projetada pelo componente como 3.84.
- **elevation: 5:** Define a altura da sombra projetada pelo componente como 5. No React Native, a propriedade elevation é usada para controlar a altura da sombra em relação à superfície do componente.
- **label{:** Define os estilos para a label.

- **fontSize: 16:** Define o tamanho da fonte do texto como 16.
- **fontWeight: 'bold':** Define a fonte como negrito.
- **marginBottom: 5:** Adiciona uma margem inferior de 5 pixels entre o rótulo e o próximo elemento.
- **input:** Este estilo é aplicado aos campos de entrada de texto no formulário.
- **width: '100%':** Define a largura do campo como 100% da largura do container.
- **height: 40:** Define a altura do campo como 40 pixels.
- **borderWidth: 1:** Define a largura da borda do campo como 1 pixel.
- **borderColor: '#ccc':** Define a cor da borda do campo como cinza claro.
- **borderRadius: 5:** Define os cantos do campo como arredondados com um raio de 5 pixels.
- **paddingHorizontal: 10:** Adiciona um preenchimento horizontal de 10 pixels dentro do campo.
- **marginBottom: 10:** Adiciona uma margem inferior de 10 pixels entre o campo e o próximo elemento.

Após o salvamento do código base estilizado ao ser acessado em um navegador de internet será exibido o resultado.

A imagem a seguir, mostra o resultado da tela feita com React Native e CSS.

Figura 11 - Resultado do Formulário de Contato

A smartphone screen displaying a contact form. The form is centered and has a white background with a thin border. It contains five text input fields, each with a label above it: 'Nome' (with placeholder 'Digite seu nome'), 'Sobrenome' (with placeholder 'Digite seu sobrenome'), 'Email' (with placeholder 'Digite seu email'), 'Cidade' (with placeholder 'Digite sua cidade'), and 'Estado' (with placeholder 'Digite seu estado'). Below the last field is a blue button with the text 'ENVIAR' in white. The smartphone's status bar at the top shows the time 4:45 and various icons. At the bottom, there is a navigation bar with three icons: a home icon, a code editor icon, and a form icon, with labels 'Home', 'Editor', and 'Form (autoStyle)' respectively.

Fonte: Autoria Própria, 2024.

3.5 Node.js

Em conformidade com Valero (2019), o NodeJS é uma plataforma construída em cima do motor de Javascript V8 do Google Chrome, permitindo a execução de programas escritos em Javascript fora do navegador.

Como destaca Moraes (2023), o NodeJS utiliza uma arquitetura orientada a eventos, ideal para desenvolver aplicativos Web, usando um modelo de entrada e saída não bloqueante, tornando-o ágil e eficaz.

De acordo com Pereira (2016), o NodeJS é uma plataforma escalável e de baixo nível, podendo programar diretamente com diversos protocolos de rede e internet.

3.6 Bootstrap

O Bootstrap é um framework que ajuda a estilizar os sites e aplicações mais bonitos visivelmente.

Para Samy (2023), o bootstrap é o mais utilizado entre os frameworks, fazendo sites e aplicações web mais responsivas, deixando o front-end mais fácil de fazer.

Conforme Cancinos (2023), o bootstrap facilita a criação de um design, poupando o tempo dos desenvolvedores, da mesma forma mostrando como ele é adaptável para distinguir plataformas.

3.7 Banco de Dados

Nas palavras de Date (2004) um sistema de banco de dados é um conjunto computacional de verificação de registros, tornando-se o proporcional a um grupo de documentos de informações computadorizados.

Conforme descrito por Leite (2008), uma das maneiras de demonstrar como os dados estão armazenados é pelo MER - Modelo Entidade Relacionamento que exhibe as informações em tabelas e a maneira que se relacionam.

Como disse Date (2004), a expressão entidade mostra o objeto representado pelo banco de dados, os relacionamentos servem para ligar os elementos assim como podem ser uma entidade também.

3.8 SQLite

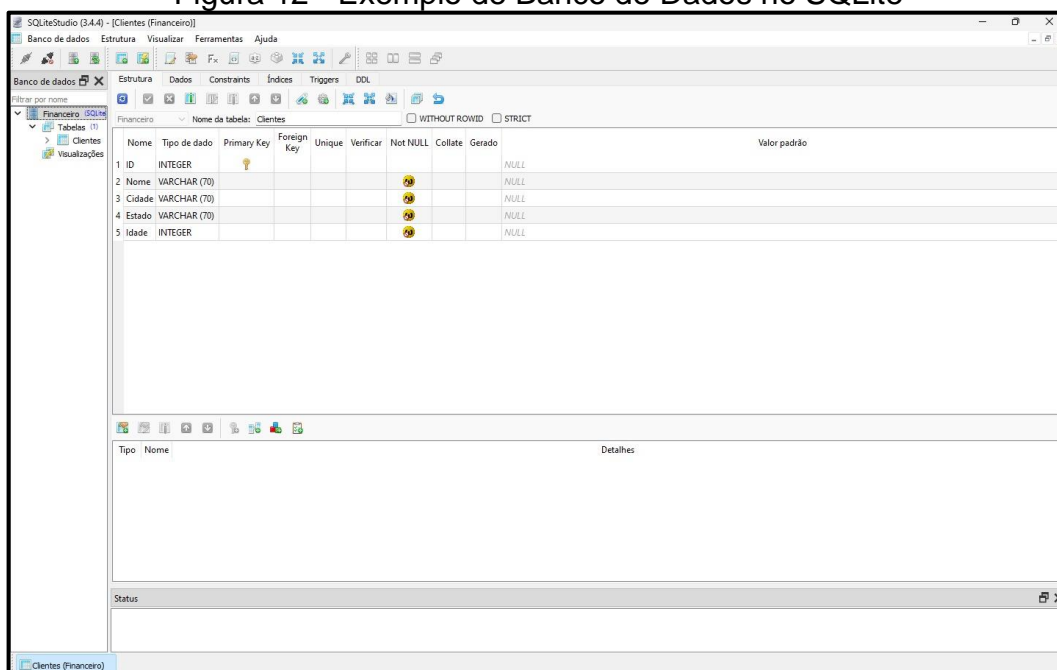
Em conformidade com Adriano (2023), O SQLite é um sistema gerenciador de banco de dados integrado e eficiente, que não requer um servidor separado e é adequado para o uso em aplicativos de computador e dispositivos móveis.

Consoante Rafael (2007), o SQLite é uma solução de banco de dados especialmente voltada para dispositivos móveis. Ele permite que aplicativos acessem bancos de dados SQL diretamente.

Segundo Prescott (2015), o SQL é uma linguagem de consulta, usada para realizar funções em um banco de dados. O SQLite oferece suporte total ao SQL, permitindo que os desenvolvedores utilizem essas funcionalidades em seus aplicativos móveis.

A Figura a seguir é uma representação de um exemplo de banco de dados realizado no SQLite:

Figura 12 - Exemplo de Banco de Dados no SQLite



Fonte: Autoria Própria, 2024.

3.9 Astah Community

Nas palavras de Guedes (2018), o Astah é uma ferramenta CASE (Engenharia de Software Auxiliada por Computador), que colaboram para a execução de uma ou mais atividades durante um processo de engenharia de software.

O Astah é uma ferramenta versátil para modelagem visual, amplamente utilizada no mercado. O Astah conta com recursos avançados e possui uma interface intuitiva, que facilita a interação do usuário com a plataforma. O Astah permite a criação de diversos tipos de diagramas, como diagrama de caso de uso, diagrama de classe, diagrama de sequência, entre outros, proporcionando uma melhor comunicação e compreensão de softwares complexos.

3.10 Unified Modeling Language

A Linguagem de Modelagem Unificada (UML), também conhecida como Unified Modeling Language, é uma linguagem para modelagem de projetos.

Conforme destacado por Guedes (2018), a UML é uma ferramenta essencial na engenharia de software, oferecendo uma notação padronizada para visualizar, especificar, construir e documentar os projetos.

A UML possui uma variedade de diagramas estruturais e comportamentais, cada um com uma finalidade específica. Estes diagramas permitem que os desenvolvedores descrevam a estrutura e o comportamento de um sistema.

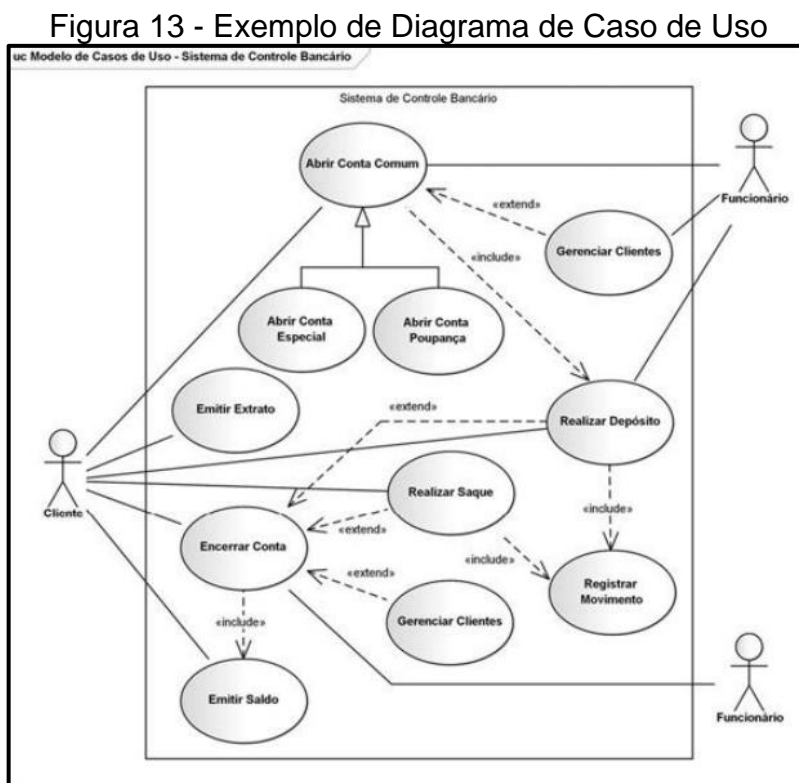
De acordo com Booch (2006), a UML não é uma linguagem de programação, mas sim uma ferramenta para o desenvolvimento de softwares. Os diagramas da UML ajudam a reduzir a complexidade do desenvolvimento de software.

Os principais diagramas utilizados são:

3.10.1 Diagrama de Caso de Uso

Costuma ser aplicado no levantamento e análise de requisitos, sendo de fácil compreensão para todos visualizarem a ideia do projeto.

Para Guedes (2018), o diagrama de caso de uso desempenha um papel importante nas etapas de levantamento e análise de requisitos do sistema. Ele oferece uma representação visual que facilita a compreensão do sistema.



Fonte: Guedes, 2011.

O diagrama de caso de uso mostrado na imagem acima representa as interações entre os clientes e funcionários e um sistema de controle bancário.

Existem dois atores principais:

- **Cliente:** O cliente é a pessoa que utiliza os serviços bancários para realizar diversas operações, como abrir contas, realizar saques e depósitos, entre outras.
- **Funcionário:** Os funcionários são os trabalhadores do banco que gerenciam as operações dos clientes e mantêm o sistema funcionando corretamente.

Casos de Uso

Os casos de uso são as funcionalidades específicas que o sistema oferece. Cada elipse representa um caso de uso.

- **Abrir Conta Comum:** Permite ao cliente iniciar o processo de abertura de uma conta bancária comum.
- **Abrir Conta Especial:** Extensão do caso de uso de abrir conta comum, permite ao cliente iniciar o processo de abertura de uma conta especial.
- **Abrir Conta Poupança:** Extensão do caso de uso de abrir conta comum, permite ao cliente iniciar o processo de abertura de uma conta poupança.
- **Realizar Depósito:** Permite ao cliente fazer um depósito em sua conta. Este caso de uso inclui o caso de uso de Registrar Movimento, que registra a transação.
- **Realizar Saque:** Permite ao cliente retirar dinheiro de sua conta.
- **Emitir Extrato:** O cliente pode solicitar e obter um extrato da sua conta.
- **Encerrar Conta:** O cliente pode fechar sua conta bancária. Este caso de uso inclui o caso de uso de Emitir Saldo, que fornece o saldo atual da conta antes do fechamento.
- **Emitir Saldo:** O cliente pode consultar o saldo atual da sua conta.

- **Gerenciar Clientes:** Este caso de uso é utilizado pelos funcionários para gerenciar as informações e operações dos clientes no sistema bancário.

Relações entre Casos de Uso

As relações entre os casos de uso são indicadas por diferentes tipos de linhas:

- **Include:** Esta relação, indicada por uma linha sólida com uma seta e o rótulo <<includes>>, mostra que um caso de uso necessariamente inclui a funcionalidade de outro caso de uso.
- **Extend:** Esta relação, indicada por uma linha pontilhada com uma seta e o rótulo <<extends>>, mostra que um caso de uso pode ser estendido por funcionalidades adicionais dependendo das condições.

Interações

As linhas que conectam os atores aos casos de uso indicam quais ações cada ator pode realizar. Por exemplo, o cliente pode abrir contas, realizar saques e depósitos, entre outras ações. Os funcionários podem gerenciar as informações dos clientes e auxiliar em operações específicas quando necessário.

Conclusão:

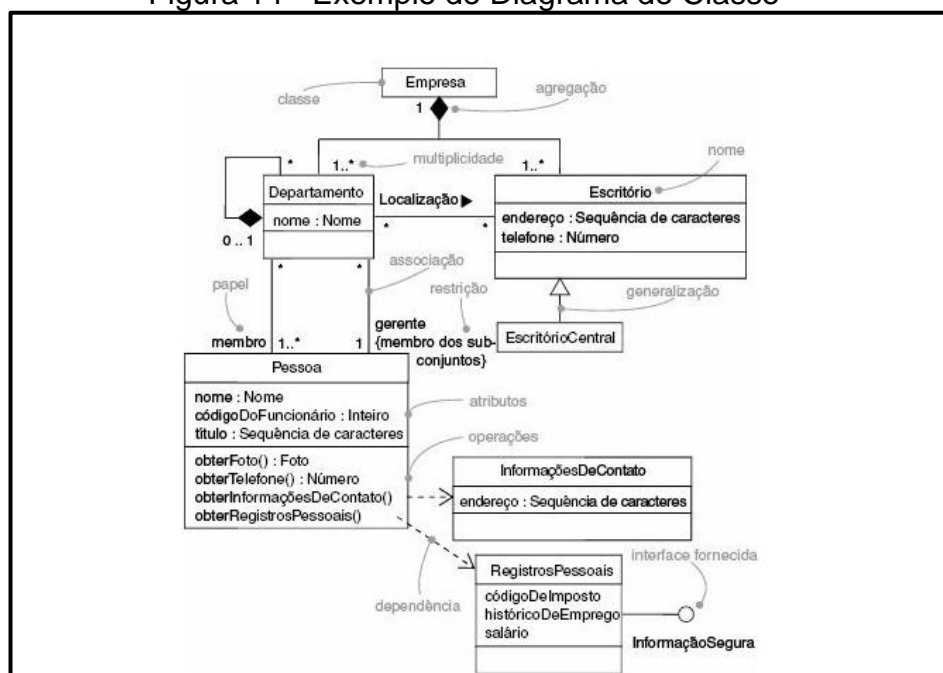
Este diagrama de sequência detalha as etapas envolvidas na abertura de uma conta comum em um sistema bancário, mostrando como diferentes atores e componentes do sistema interagem ao longo do tempo para completar a tarefa.

3.10.2 Diagrama de Classe

É executada na descrição das classes mostrando seus atributos e métodos, identificando a forma como cada um se relaciona.

Como aponta Booch (2006), os diagramas de classes oferecem uma representação visual das classes, interfaces e interações. Permitindo aos desenvolvedores identificar as classes, associações, heranças e dependências.

Figura 14 - Exemplo de Diagrama de Classe



Fonte: Booch, 2006.

A imagem acima apresenta um diagrama de classes que modela a estrutura de uma empresa e suas relações.

Empresa:

- Representada por um losango, possui uma relação de multiplicidade de "1" com "Departamento" e "Escritório", indicando que uma empresa tem um ou mais departamentos e um ou mais escritórios.

Departamento:

- **Atributo:** "nome: Nome".
- Possui uma multiplicidade de "1.." com "Pessoa", indicando que um departamento é composto por uma ou mais pessoas.
- Relaciona-se com "Pessoa" através do papel "membro".
- Relaciona-se com "Localização" com multiplicidade de "0..1", indicando que um departamento pode ou não ter uma localização.

Escritório:

- **Atributos:** "endereço: Sequência de caracteres" e "telefone: Número".
- Relaciona-se com "Localização" com multiplicidade de "1..", indicando que um escritório tem uma ou mais localizações.
- Apresenta uma generalização com "EscritórioCentral", significando que "EscritórioCentral" é uma especialização de "Escritório", mas com características adicionais ou restrições.

Pessoa:

- **Atributos:** "nome: Nome", "códigoDoFuncionário: Inteiro", "título: Sequência de caracteres".
- **Operações:** "obterFoto()", "obterTelefone()", "obterInformaçõesDeContato()", "obterRegistrosPessoais()".

- Relaciona-se com "InformaçõesDeContato" e "RegistrosPessoais".

InformaçõesDeContato:

- **Atributo:** "endereço: Sequência de caracteres".

RegistrosPessoais:

- **Atributos:** "códigoDeImposto", "históricoDeEmprego", "salário", e depende de "InformaçõesDeContato"

Multiplicidade:

- "1.." para várias relações, indicando que uma entidade pode estar relacionada com uma ou mais instâncias de outra entidade.
- "0..1" indicando uma relação opcional, onde a entidade pode ou não estar presente.

Conclusão:

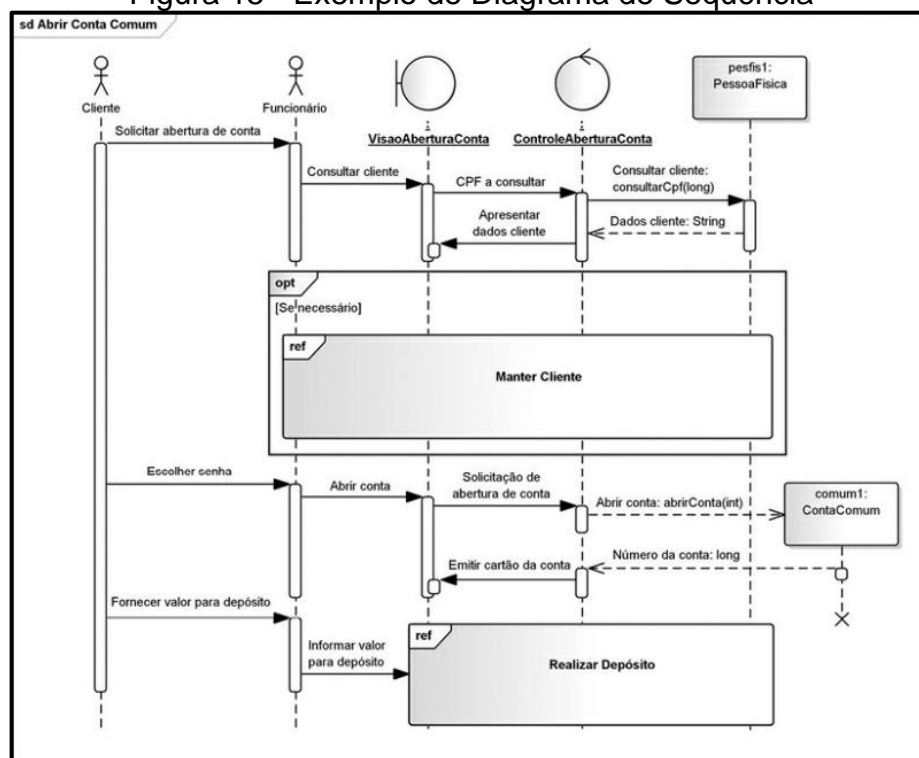
O diagrama mostra como os diferentes elementos da estrutura de uma empresa se relacionam entre si, detalhando suas associações, agregações e generalizações, além de especificar os atributos e operações das classes envolvidas.

3.10.3 Diagrama de Sequência

Exibe um grupo de objetos que são representados por linhas identificando mensagens que estão sendo trocadas durante a interação dos objetos.

Em conformidade com Guedes (2018), o diagrama de sequência representa a interação entre objetos ao longo do tempo. Essa visualização é essencial para compreender o fluxo de execução das funcionalidades do sistema.

Figura 15 - Exemplo de Diagrama de Sequência



Fonte: Guedes, 2011.

O diagrama de sequência acima mostra a interação entre diferentes objetos e atores ao longo do tempo para realizar a ação de "Abrir Conta Comum" em um sistema bancário.

Elementos do Diagrama:

Atores e Objetos:

- **Cliente:** Inicia o processo solicitando a abertura de uma conta.
- **Funcionário:** Interage com o sistema para processar a solicitação do cliente.
- **VisaoAberturaConta:** Um componente de visualização que apresenta informações ao funcionário.
- **ControleAberturaConta:** Um componente de controle que gerencia a lógica de abertura de conta.
- **PessoaFisica:** Representa a entidade de cliente no sistema.
- **ContaComum:** Representa a entidade da conta bancária a ser aberta.

Interações:

- **Solicitar abertura de conta:** O cliente inicia o processo pedindo ao funcionário para abrir uma conta.
- **Consultar cliente:** O funcionário consulta a interface “VisaoAberturaConta” para obter informações do cliente.
- **CPF a consultar:** O funcionário insere o CPF do cliente a ser consultado.
- **Apresentar dados cliente:** Retorna os dados do cliente para o funcionário.
- **Manter Cliente:** Se necessário, o sistema realiza operações adicionais de manutenção de dados do cliente.
- **Escolher senha:** O cliente escolhe uma senha para a conta.
- **Abrir conta:** O funcionário solicita a abertura da conta através do componente de controle.
- **Solicitação de abertura de conta:** A solicitação é processada pelo componente de controle.
- **Emitir cartão da conta:** O sistema emite um cartão para a conta.
- **Informar valor para depósito:** O cliente informa o valor a ser depositado na conta.
- **Realizar Depósito:** A operação de depósito é realizada, completando o processo de abertura da conta.

Detalhamentos do Fluxo:**Início do Processo:**

- O cliente solicita a abertura de uma conta ao funcionário.

Consulta de Informações:

- O funcionário consulta o sistema para verificar os dados do cliente utilizando o CPF.

- A interface de visualização apresenta os dados do cliente.

Manutenção de Cliente:

- Se necessário, o sistema pode realizar operações adicionais para manter ou atualizar os dados do cliente.

Abertura da Conta:

- O cliente escolhe uma senha para a nova conta.
- O funcionário utiliza o componente de controle para processar a solicitação de abertura da conta.
- O sistema emite um cartão da conta.

Depósito Inicial:

- O cliente fornece o valor inicial para depósito.
- O sistema realiza a operação de depósito, concluindo o processo.

Fim do Processo:

- O processo de abertura da conta é concluído, indicado pelo "X" no final do diagrama, que representa a finalização da interação.

Conclusão:

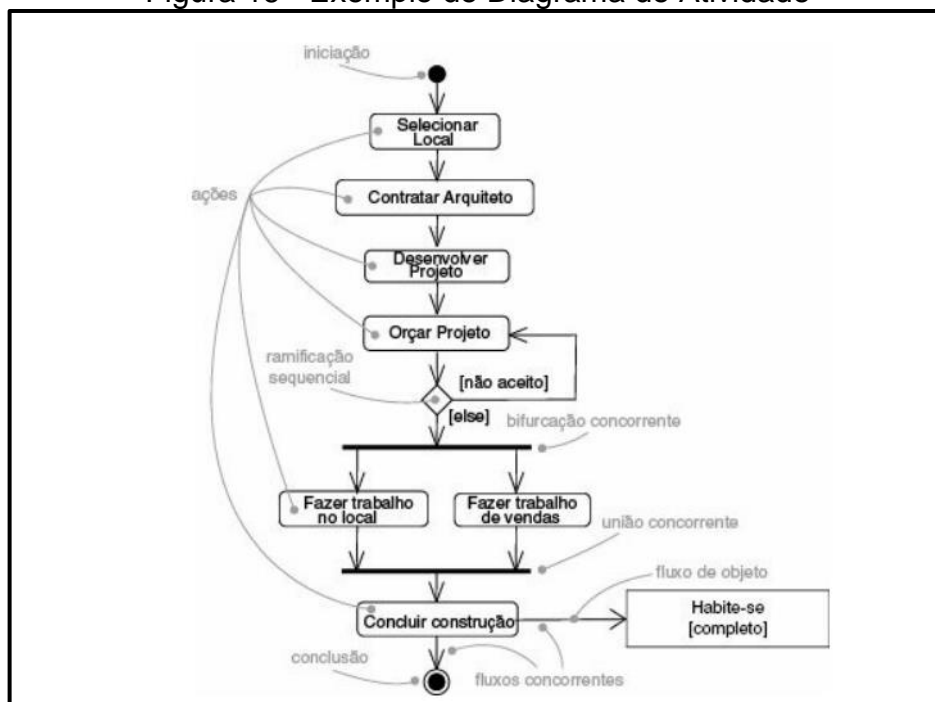
Esse diagrama fornece uma visão clara das interações entre os atores Clientes e Funcionários em um sistema de controle bancário, desde a abertura de contas até a gestão de clientes e movimentações financeiras. Ele mostra como diferentes funcionalidades estão relacionadas e dependem umas das outras.

3.10.4 Diagrama de Atividade

Representa os passos que devem ser executados durante o processo do projeto, podendo ser mostrado como uma rotina.

Nas palavras de Fowler (2005), o diagrama de atividade é eficaz para representar os passos sequenciais e as decisões de um processo, além das interações entre os elementos do sistema. Ele oferece uma compreensão ampla do comportamento do sistema.

Figura 16 - Exemplo de Diagrama de Atividade



Fonte: Booch, 2006.

O diagrama de atividade acima ilustra o fluxo de trabalho de um processo de construção.

Elementos e Fluxo do Diagrama:

Início:

- O diagrama começa com um círculo preto sólido, que representa o ponto de início do processo.

Seleção de Local:

- A primeira atividade é "Selecionar Local", onde se escolhe o local da construção.

Contratação de Arquiteto:

- Após selecionar o local, a próxima etapa é "Contratar Arquiteto", no qual será responsável pelo desenvolvimento do projeto.

Desenvolvimento do Projeto:

- O arquiteto desenvolve o projeto, detalhando como a construção será realizada.

Orçamento do Projeto:

- O "Orçar Projeto" é a atividade em que se calcula o custo total do projeto.

Decisão sobre o Orçamento:

- Após o orçamento, há uma decisão representada por um losango onde:

Se o projeto não for aceito, o processo retorna para a etapa de "Orçar Projeto".

Se o orçamento for aceito, o processo continua para as próximas atividades.

Trabalhos:

- O diagrama se divide em dois fluxos paralelos, sendo eles:

- **Fazer trabalho no local:** Refere-se às atividades de construção física no local.

- **Fazer trabalho de vendas:** Refere-se às atividades de marketing e vendas.

Concluir Construção:

- Após os trabalhos de construção e vendas serem concluídos, a próxima atividade é "Concluir Construção".

Emissão do Habite-se:

- O processo termina com a emissão do "Habite-se", documento que certifica que a construção está completa e em conformidade com as regulamentações.

Fim:

- O diagrama conclui com um círculo preto com um contorno, indicando o fim do processo.

Explicação do Fluxo e dos Conceitos:

- **Atividades:** Cada retângulo representa uma atividade no processo.
- **Decisão:** O losango representa um ponto de decisão, onde o fluxo do processo pode seguir diferentes caminhos baseados em condições.
- **Fluxos Paralelos:** As linhas horizontais representam a bifurcação e a união de fluxos paralelos, indicando que as atividades podem ocorrer simultaneamente.
- **Início e Fim:** Os círculos sólidos representam o início e o círculo com contorno representa o fim do processo.

-Conclusão:

Este diagrama detalha um processo sequencial e paralelo de construção, desde a seleção do local até a conclusão da construção e emissão do Habite-se. Ele é útil para visualizar o fluxo de trabalho, identificar pontos de decisão, e compreender como diferentes atividades interagem no processo de construção.

3.11 Application Programming Interface

Segundo a IBM, uma API (Application Programming Interface) é um conjunto de normas essenciais para facilitar a comunicação segura entre diferentes aplicativos e sistemas, auxiliando a compartilhar dados e funcionalidades de seus aplicativos.

3.11.1 Mapbox

Como aponta Aravena e Delazari (2021), o Mapbox é uma API que possui um plugin chamado “Mapbox Android Map”, que permite adicionar um mapa em aplicativos smartphone com o sistema operacional Android.

De acordo com o Ribeiro e Gonçalves (2020), com o Mapbox é possível fazer a estilização do seu mapa, podendo escolher estilos padrões que a API possui, ou podendo criar o seu próprio estilo.

Conforme descrito por Bocard (2019), a API do Mapbox permite a utilização dos mapas de forma offline, pois é possível fazer o download dos mapas e das rotas.

3.12 Figma

Em conformidade com Villain e Silveira (2023), o figma é uma ferramenta de design onde qualquer pessoa pode ter acesso, criando seu próprio design digital para sites, aplicativos e entre outros...

Para Lopes (2023), O figma pode ser acessado em qualquer lugar por ser em uma plataforma web, basta ter Internet, oferecendo uma forma de fazer design mais acessível a todos.

3.13 Wireframe

De acordo com Mendonça (2017), ao criar um design adequado é necessário planejar com antecedência, visto que fazer um site sem um planejamento adequado é semelhante a construir uma casa sem a icnografia de sua obra.

Em conformidade com Teixeira (2014), o Wireframe é um desenho simples da disposição de uma interface, com o objetivo de representar a aparência final e a funcionalidade do produto oferecido.

3.14 Visual Studio Code

Como aponta a Microsoft (2024), o Visual Studio code é um editor de código-fonte, que dentro dele pode ser utilizado JavaScript, TypeScript e Node.js, possui extensões para outras linguagens.

3.15 Expo

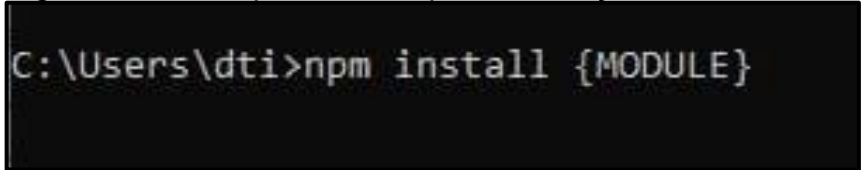
Como destacado por Fuentes (2023), O Expo é uma ferramenta utilizada no desenvolvimento de aplicações moveis com React Native possibilitando uma fácil introdução de APIs nativas sem necessidade de inserir qualquer dependência.

O Expo é uma ferramenta que auxilia no desenvolvimento de aplicativos sendo possível criar aplicativos para IOS, Android e web, baseado em JavaScript ou TypeScript, visto que ele promove suporte para programas no emulador Android ou smartphone.

3.16 Node Package Manager

Segundo Paiva (2018), O Node Package Manager (NPM) é um administrador de pacotes de uma aplicação, que tem o Angular pois possui a mesma base do NodeJs.

Figura 17 - Exemplo do NPM para instalação de um modulo



```
C:\Users\dti>npm install {MODULE}
```

Fonte: Autoria Própria, 2024.

Na figura 17, foi feito um exemplo do NPM (Node Package Manager) para a instalação de um modulo.

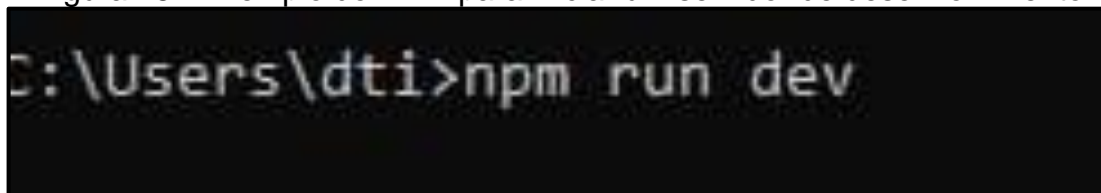
O comando `npm install {MODULE}` é utilizado para instalar um módulo específico no contexto de um projeto Node.js. Para melhor compreensão, abaixo está uma breve explicação de cada parte do código:

`npm`: Este é o comando Node Package Manager, uma ferramenta que facilita a instalação e o gerenciamento de pacotes de software Node.js.

`install`: Esta é a instrução para npm instalar pacotes.

`{MODULE}`: Este é um espaço reservado que deve ser substituído pelo nome do módulo que você deseja instalar.

Figura 18 - Exemplo do NPM para iniciar um servidor de desenvolvimento



Fonte: Autoria Própria, 2024.

Na figura 18, foi feito um exemplo do NPM (Node Package Manager) para a iniciação de um servidor de desenvolvimento.

O comando `npm install {MODULE}` é utilizado para instalar um módulo específico no contexto de um projeto Node.js. Para melhor compreensão, abaixo está uma breve explicação de cada parte do código:

`npm`: Este é o comando Node Package Manager, uma ferramenta que facilita a instalação e o gerenciamento de pacotes de software Node.js.

`install`: Esta é a instrução para npm instalar pacotes.

`{MODULE}`: Este é um espaço reservado que deve ser substituído pelo nome do módulo que você deseja instalar.

4. REFERÊNCIAS

BOOCH, Grady. **UML - Guia do Usuário**. 2ª ed. Tradução. Rio de Janeiro: GEN LTC, 2006.

CANCINOS, M. Bootstrap - **Iniciantes: Design rápido e fácil para programadores de web e aplicativos**. 1. ed. Georgia: White Tower Publishing, 2020.

DATE, Christopher J. **Introdução a sistemas de banco de dados**. 8. ed. Rio de Janeiro: Elsevier. 2004.

ESCUDELARIO, Bruna; PINHO, Diego. React Native: **Desenvolvimento de aplicativos mobile com React**. 1. ed. São Paulo: Casa do Código, 2020.

FLANAGAN, D. **JavaScript: O Guia Definitivo**. 6.ed. São Paulo: Bookman Editora, 2012.

FLATSCHART, Fábio. **HTML5 Embarque imediato**. 1. ed. Rio de Janeiro: Brasport, 2011.

FOWLER, Martin. **UML Essencial: Um Breve Guia para a Linguagem-Padrão de Modelagem de Objetos**. 3ª ed. Porto Alegre: Bookman, 2005.

GUEDES, Gilleanes T. A. **UML 2 - Uma Abordagem Prática**. 3ª ed. São Paulo: Novatec Editora, 2018.

JOBSTRAIBIZER, Flavia. **Criação de Sites com CSS**. 1. ed. São Paulo: Universo dos Livros, 2009.

LEITE, Mario. **Acessando banco de dados com ferramentas RAD: aplicações em Delphi**. 1. ed. Rio de Janeiro: Brasport. 2008.

MORAES, William Bruno. **Construindo aplicações com NodeJS**. 4. ed. São Paulo: Novatec Editora, 2023.

OLIVEIRA, C. L. V., & Zanetti, H. A. P. **JavaScript Descomplicado: Programação para a Web, IOT e Dispositivos Móveis**. 1. ed. Editora Érica, 2020.

PEREIRA, A., & POUPA, C. **Linguagens WEB**. 6. ed. Lisboa: Edições Sílabo, 2017.

PEREIRA, C. R. **Construindo APIs REST com Node.js**. São Paulo: Casa do Código, 2016.

PIN S.D.S. **Guia para iniciantes no desenvolvimento mobile: HTML, CSS, JavaScript e React Native**. 1. ed. São Paulo: Câmara Brasileira do Livro, 2021.

PRESCOTT, P. (R. C. S. Barros, Trad.). **SQL para Iniciantes**. Babelcube Inc, 2015.

PRESCOTT, P (F. Souza, Trad.). **Programação em JavaScript**. Babelcube Inc, 2016

QUIERELLI, D. A. **Criando sites com HTML-CSS-PHP: Construindo um projeto - Iniciante**. 1. ed. Santa Catarina: Clube de Autores, 2012.

SAMY, M. S. **Bootstrap: um Guia Completo Para Construir Aplicativos Responsivos, Modernos e Eficientes**. 1. ed. São Paulo Novatec Editora, 2023.

SAMY, M. S. **Criando Sites com HTML: Sites de Alta Qualidade com HTML e CSS**. São Paulo: Novatec Editora, 2008.

SAMY, M. S. **JavaScript - Guia do Programador: Guia completo das funcionalidades de linguagem JavaScript**. São Paulo: Novatec Editora, 2010.

SAMY, M. S. **Construindo Sites com CSS e (X)HTML**. 1. ed. São Paulo: Novatec, 2007.

TECH, Adrinao. **Apostila de SQLite – Nível Básico**. 2023. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/7748553/mod_resource/content/1/Apostila%20de%20SQLite%20básico.pdf> Acesso em: 27 abril. 2024.

TEIXEIRA, Fabricio. **Introdução e boas práticas em UX Design**. São Paulo: Casa do Código, 2014. E-book.

ARNT, Ana de Medeiros; SILVA, Ester Silmão Lopes. **O acesso às escolas do campo e o transporte escolar**. UNEMAT, MT, 2008. Disponível em:<http://need.unemat.br/4_forum/artigos/ester.pdf>. Acesso em: 01 abr. 2024.

Figma: o que é a ferramenta, design e como usar. Disponível em: <<https://www.alura.com.br/artigos/figma>>. Acesso em: 09 maio. 2024.

FUENTES, Guilherme Cardoso. **LightLow: Aplicativo simulador de consumo energético residencial**. 2023. Trabalho de Conclusão de Curso (Curso de Bacharelado em Sistemas de Informação) - Faculdade De Ciências de Bauru, Bauru, 2023. Disponível em: <<https://repositorio.unesp.br/handle/11449/239454>> Acesso em: 10 mai. 2024.

MENDONÇA, Ewerton. **Introdução a Web Design: Curso Técnico em Informática: Educação a distância**. 2017. Disponível em: <[https://sisacad.educacao.pe.gov.br/bibliotecavirtual/bibliotecavirtual/texto/Caderno_I_NFO\(Introducao_a_Web_Design_2017.1\).pdf](https://sisacad.educacao.pe.gov.br/bibliotecavirtual/bibliotecavirtual/texto/Caderno_I_NFO(Introducao_a_Web_Design_2017.1).pdf)> Acesso em: 01 maio. 2024

PAIVA, Aline Vitor Vaz. **Aplicativo Agrocomputer: Protótipo para Controle de Custeio Baseado em Tempo**. Uberlândia, 2018. Disponível em:

<<https://repositorio.ufu.br/bitstream/123456789/24526/3/AplicativoAgrocomputerPrototipo.pdf>>. Acesso em: 15 maio 2024.

RIBEIRO, Caique da Silva; GONÇALVES, Felipe Santos. **Ônibus: Aplicativo para Rastreamento, Amostra da Rota e Pontos de Parada do Ônibus do Instituto Federal**. *Instituto Federal de Educação, Ciência e Tecnologia do Piauí – IFPI*, 2020.

Disponível em: < https://d1wqtxts1xzle7.cloudfront.net/62466439/OniBus_-_Aplicativo_para_Rastreamento__Amostra_da_Rota_e_Pontos_de_Parada_do_Onibus_do_Instituto_Federal20200324-28282-1ltwhzg-libre.pdf?1585186411=&response-content-disposition=inline%3B+filename%3DOni_Bus_Aplicativo_para_Rastreamento_Amo.pdf&Expires=1715979294&Signature=cD8ddtOCHKZjMQ0d8rmS5PF3hfGAiXxGdogLsX1gBT5QrDAeyl4pFW8ypapmda0PC-safrEM2RVF8gDSMPk9ij1tqWnJitvIdtF5FWUdbCxQIYFmjNGLNXGWWGuQ2rSqFjqhP25L9uJKgn0qv1n-xUvufDMZelZgPI0JOwKMcx~y-LbBFw5Uy5gUmTPcc0CU2V4AoVfMYKGc2Z5gSzXenANB1t5EIlhfIF~MYkle-UFd2BxJg6glUdyl8DfeK5kl8uSuyD6cBMxqbE6pC3IRga9HZOXDcx51ASLO6xT93PGZLoVd9ZPBRJSCdXVodWU456ebPL~5EdEwn~tYGetrA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 15 maio 2024.

libre.pdf?1585186411=&response-content-

disposition=inline%3B+filename%3DOni_Bus_Aplicativo_para_Rastreamento_Amo.pdf&Expires=1715979294&Signature=cD8ddtOCHKZjMQ0d8rmS5PF3hfGAiXxGdogLsX1gBT5QrDAeyl4pFW8ypapmda0PC-

safrEM2RVF8gDSMPk9ij1tqWnJitvIdtF5FWUdbCxQIYFmjNGLNXGWWGuQ2rSqFjqhP25L9uJKgn0qv1n-xUvufDMZelZgPI0JOwKMcx~y-

LbBFw5Uy5gUmTPcc0CU2V4AoVfMYKGc2Z5gSzXenANB1t5EIlhfIF~MYkle-

UFd2BxJg6glUdyl8DfeK5kl8uSuyD6cBMxqbE6pC3IRga9HZOXDcx51ASLO6xT93PGZLoVd9ZPBRJSCdXVodWU456ebPL~5EdEwn~tYGetrA__&Key-Pair-

Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 15 maio 2024.

VALERO, D. M. (2019). **Negação de Serviço em Aplicações Node.js**.

FACULDADE DE TECNOLOGIA DE AMERICANA. Disponível em: <

https://ric.cps.sp.gov.br/bitstream/123456789/3727/1/20191S_VALERODouglasMaria_no_OD0658.pdf > Acesso em: 01 abril. 2024.

VILLAIN, Matheus. SILVEIRA, Maria. **Figma: o que é a ferramenta, Design e uso**. 2023. Disponível em: <<https://www.alura.com.br/artigos/figma>>. Acesso em: 01. Maio. 2024.

ARAVENA, Carolina Aguilar; DELAZARI, Luciene Stamato. Desenvolvimento de aplicativo para auxílio à navegação em ambientes internos. **Revista Brasileira de Cartografia**, Curitiba, PR, v. 73, n. 2, p. 530–541, abr./jun. 2021.

BALARDIM, G. CLIPESCOLA. **Transporte escolar e acesso à escola**. Disponível em: <https://www.clipescola.com/transporte-escolar-e-acesso-a-escola/>. Acesso em: 16 de mar. 2024.

BOCARD, Taysa. **Mapbox ou Google Maps: qual API maps escolher para seu app?** 2019. Disponível em: <https://usemobile.com.br/mapbox-ou-google-maps/#:~:text=O%20Mapbox%20Studio%20permite%20criar,usar%20os%20servi%C3%A7os%20da%20companhia%3E>. Acesso em: 15 maio 2024.

G1. **Falta de transporte escolar dificulta vida de alunos para frequentarem aulas em Itaquaquecetuba**. Disponível em: <https://g1.globo.com/sp/mogi-das-cruzes-suzano/noticia/2023/02/07/falta-de-transporte-escolar-dificulta-vida-de-alunos-para-frequentarem-aulas-em-itaquaquecetuba.ghtml>. Acesso em: 01 mai. 2024.

G1 São Paulo. **Van escolar e carro se envolvem em acidente e crianças ficam feridas na Grande SP**. Disponível em: <https://g1.globo.com/sp/sao-paulo/noticia/2023/12/11/van-escolar-e-carro-se-envolvem-em-acidente-e-criancas-ficam-feridas-na-grande-sp.ghtml>. Acesso em: 03. maio. 2024.

IBM. **O que é API?** Disponível em: <https://www.ibm.com/brpt/topics/api> Acesso em: 13 maio. 2024.

LOPES, M. **Figma: o que é e como usar, suas funcionalidades e vantagens**. Disponível em: <https://ebaonline.com.br/blog/o-que-e-figma-e-como-usar>. Acesso em: 09 maio. 2024.

MICROSOFT. **Visual Studio Code**. [S. l.], 2024. Disponível em: <https://visualstudio.microsoft.com/pt-br/>. Acesso em: 02 mai. 2024.

Prefeitura de São Paulo. “Transporte Escolar Gratuito - TEG.” Disponível em: <https://educacao.sme.prefeitura.sp.gov.br/transporte-escolar-gratuito/>. Acesso em: 1 maio. 2024.

RAFAEL. (2007). **SQLite - O Pequeno Notável**. Disponível em: <https://www.devmedia.com.br/sqlite-o-pequeno-notavel/7249> > Acesso em: 27 abril. 2024.

SÃO PAULO. "Volta às aulas: Detran-SP dá dicas para contratação de serviço de transporte escolar." Disponível em:

<https://www.saopaulo.sp.gov.br/spnoticias/ultimas-noticias/volta-as-aulas-detran-sp-da-dicas-para-contratacao-servico-de-transporte-escolar/>. Acesso em: 16 de mar. 2024.