**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**ETEC DA ZONA LESTE**

**NOVOTEC Desenvolvimento de Sistemas**

**Ana Beatriz Novais Pereira**

**Andrei Nunes Pereira**

**Camilly Demarco dos Santos**

**Carlos Eduardo Fernandes Farias**

**Carchau:**

**Sistema de conectividade na locação de automóveis**

**São Paulo**

**2024**

**Ana Beatriz Novais Pereira**

**Andrei Nunes Pereira**

**Camilly Demarco dos Santos**

**Carlos Eduardo Fernandes Farias**

**CARCHAU:**

**Sistema de conectividade na locação de automóveis**

Trabalho de Conclusão de Curso apresentado ao Novotec Desenvolvimento de Sistemas da Etec Zona Leste, orientado pelo Prof. Esp. Jeferson Roberto de Lima, como requisito parcial para a obtenção do título de técnico em Desenvolvimento de Sistemas.

**São Paulo**

**2024**

**Ana Beatriz Novais Pereira**

**Andrei Nunes Pereira**

**Camilly Demarco dos Santos**

**Carlos Eduardo Fernandes Farias**

**CARCHAU:**

**Sistema de conectividade na locação de automóveis**

Trabalho de Conclusão de Curso apresentado ao Novotec Desenvolvimento de Sistemas da Etec Zona Leste, orientado pelo Prof. Esp. Jeferson Roberto de Lima, como requisito parcial para a obtenção do título de técnico em Desenvolvimento de Sistemas.

Aprovado em:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(título e nome de elemento que compõe a banca examinadora) (data)

Dedicamos este presente trabalho a todas as pessoas que são afligidas pelas dificuldades de locomoção nas metrópoles independentemente de recursos ou falta de acesso, além dos aficionados por tecnologia e inovação.

**Agradecimentos**

“A coisa mais indispensável a um homem é reconhecer o uso que deve fazer do seu próprio conhecimento.”

***Platão***

**Resumo**

Inicialmente há diversos desafios urbanos enfrentados pela população, um dos principais desafios enfrentados pela população é a falta de infraestrutura e gerenciamento para a otimização da mobilidade urbana. O propósito desta pesquisa consiste no desenvolvimento de uma aplicação web e mobile, com o objetivo de contribuir para um maior dinamismo na circulação de pessoas e refletir no desenvolvimento sustentável, econômico e social por meio da locação de automóveis. Os métodos utilizamos no momento do planejamento e documentação se baseiam nas metodologias ágeis, e em pesquisas de viés científico e acadêmico. A aplicação mobile oferece maior mobilidade aos habitantes de São Paulo, compartilhando o ônus da posse de um veículo entre locadores e locatários, e visa minimizar as dificuldades de atingir as regiões centrais às mais periférica. Deste modo, viabilizaremos os transportes por meios dos veículos automotores, com a redução de custos deste tipo de locação.

**Palavras-chave:** Mobilidade urbana. Aplicação mobile. Locação. Veículo. Transporte.

**Abstract**

First of all, there are several urban challenges confront by the population, being a principally the miss infrastructure and gestion to optimize the urban mobility. This research project intends to develop web and mobile application, that the objective is to contribute to taller than dynamism on people circulation and reflect in developing sustainable, economic and social by rent of automobiles. The current research is based on to plan and documentation moments such as Agil methodology, based in search the strand scientific and academic. The main results are the greater mobility of the peoples from São Paulo by means of a mobile application that share in between, the landlords and renters, the onus of the ownership of a vehicle and intend minimize the difficulties to reach the most peripheral regions. Therefore, will have a transport viabilization by means the vehicles, with a reduction of the cost those type of location.

**Keywords:** Urban mobility. Mobile application. Location. Vehicle. Transport.

**Lista de Ilustrações**

[Figura 1 – Tags bases de um arquivo HTML 16](#_Toc171643234)

[Figura 2 – Exemplo de um código HTML para criação da tela de login 17](#_Toc171643235)

[Figura 3 – Resultado de um código HTML para criação da tela de login 17](#_Toc171643236)

[Figura 6 – Tags do formulário completo 21](#_Toc171643237)

[Figura 7 – Resultado da estilização HTML para criação da tela de login 22](#_Toc171643238)

[Figura 8 – Exemplo de validação de formulário 23](#_Toc171643239)

[Figura 9 – Continuação do exemplo de validação de formulário 24](#_Toc171643240)

[Figura 10 – Exemplo de comandos utilizando o NPM 25](#_Toc171643241)

[Figura 11 – Exemplo de estrutura de um Projeto em React 26](#_Toc171643242)

[Figura 12 – Exemplo de estrutura de código para iniciar um Projeto em React 26](#_Toc171643243)

[Figura 13 – Exemplo de estrutura de código da execução a aplicação 27](#_Toc171643244)

[Figura 14 – Formulário em *React* 27](#_Toc171643245)

[Figura 15 – Exemplo de estrutura de um Projeto em *React Native* 28](#_Toc171643246)

[Figura 16 – Exemplo de código para iniciar um projeto em React Native 29](#_Toc171643247)

[Figura 17 – Exemplo para definir elementos de um projeto em React Native 30](#_Toc171643248)

[Figura 18 – Exemplo para definir Botões em um Projeto em React Native 31](#_Toc171643249)

[Figura 19 – Exemplo para definir estilização em um projeto em React Native 32](#_Toc171643250)

[Figura 20 – Formulário em React Native 33](#_Toc171643251)

[Figura 21 – Exemplo de estrutura de um código em Typescript 34](#_Toc171643252)

[Figura 22 – Execução de um código em Typescript 34](#_Toc171643253)

[Figura 23 – Exemplo de estrutura de um código em Typescript com erro 35](#_Toc171643254)

[Figura 24 – Resultado do código em Typescript com erro 35](#_Toc171643255)

[Figura 25 – Exemplo de estrutura de um código em JSON 36](#_Toc171643256)

[Figura 26 – Demonstração gráfica da comunicação com Cloud Firestore 38](#_Toc171643257)

[Figura 27 – Exemplo de Wireframe de baixa fidelidade 39](#_Toc171643258)

[Figura 28 – Exemplo de prototipação do Wireframe de alta fidelidade 40](#_Toc171643259)

[Figura 29 – Exemplo Diagrama de Caso de uso 41](#_Toc171643260)

[Figura 30 – Exemplo de diagrama de atividade 43](#_Toc171643261)

[Figura 31 – Exemplo de diagrama de sequência 44](#_Toc171643262)

[Figura 32 – Exemplo de diagrama de máquina de estado 45](#_Toc171643263)

**Lista de Quadros**

[Tabela 1 – Documentação do Caso de Uso vender ingresso 39](#_Toc168564812)

[Tabela 2 – Continuação da documentação do Caso de Uso vender ingresso 40](#_Toc168564813)

**Lista de Abreviaturas e Siglas**

Application (APP)

Application Programming Interface (API)

Cascading Style Sheets (CSS)

Document Object Model (DOM)

Hypertext Markup Language (HTML)

I Operating System (IOS)

JavaScript (JS)

JavaScript Object Notation (JSON)

Just-in-Time (JIT)

Node Package Manager (NPM)

Regras de Negócio (RN)

Requisitos Funcionais (RF)

Requisitos Não Funcionais (RNF)

São Paulo Transporte (SPTrans)

Sistema de Gerenciamento de Dados (SGBD)

Structured Query Language (NOSQL)

Unified Modeling Language (UML)

**SUMÁRIO**

[**1 INTRODUÇÃO 13**](#_Toc171643430)

[**2 REFERENCIAL TEÓRICO 14**](#_Toc171643431)

[**2.1 Dificuldades na Locomoção 14**](#_Toc171643432)

[**2.2 Tecnologias Utilizadas 14**](#_Toc171643433)

[**2.3 HTML 14**](#_Toc171643434)

[**2.4 CSS 17**](#_Toc171643435)

[**2.5 JavaScript 21**](#_Toc171643436)

[**2.6 Framework 23**](#_Toc171643437)

[**2.7 Node e NPM 23**](#_Toc171643438)

[**2.8 React 24**](#_Toc171643439)

[**2.9 React Native 27**](#_Toc171643440)

[**2.10 Expo Go 31**](#_Toc171643441)

[**2.11 Typescript 32**](#_Toc171643442)

[**2.12 JSON 34**](#_Toc171643443)

[**2.13 API 36**](#_Toc171643444)

[**2.14 Banco de Dados 36**](#_Toc171643445)

[**2.15 Firebase 36**](#_Toc171643446)

[**2.16 Wireframe 37**](#_Toc171643447)

[**2.17 UML 39**](#_Toc171643448)

[**3 DESENVOLVIMENTO 45**](#_Toc171643449)

[**3.1 Carchau: Concepção e documentação 45**](#_Toc171643450)

[**3.2 Levantamento de Requisitos 46**](#_Toc171643451)

[**3.3 Requisitos Funcionais 46**](#_Toc171643452)

[**4 CONSIDERAÇÕES FINAIS 48**](#_Toc171643453)

[**REFERÊNCIAS 49**](#_Toc171643454)

1 INTRODUÇÃO

A mobilidade urbana trata-se da capacidade de locomoção das pessoas nas cidades e áreas urbanizadas (GALINDO; LIMA, 2019). Sendo um dos problemas cotidianos atuais da mobilidade urbana, a dificuldade em obter acesso e qualidade no transporte público atual, composto por frotas de ônibus que não conseguem suprir a alta demanda. De acordo com o São Paulo Transporte (SPTrans) (2021 apud PESCARINI 2023) responsável pelo sistema de ônibus municipais, os usuários fizeram 17.928 queixas em relação ao transporte público entre os meses de janeiro e março de 2023. De acordo com o Universidade da Califórnia (2018 apud G1 Em Movimento 2018) um carro compartilhado retira, em média, de nove a 13 automóveis das ruas.

Portanto, há diversos fatores que levam as pessoas a utilizar os meios de transportes público, devido à alta demanda e atualmente se mostra inviável atender a toda população, além de que com o passar dos anos está havendo um aumento do uso dos transportes particulares. Dessa forma, qual seria a alternativa tecnológica além do transporte público, capaz de suprir essa necessidade?

Logo a ideia para amenizar o problema em questão, é a otimização da mobilidade da população por meio de uma aplicação mobile que simule o compartilhamento entre os locadores e locatários, da responsabilidade da posse de um veículo através da sua locação. Uma aplicação mobile que visa impactar uma mudança no fluxo e na comercialização de veículos, viabilizando o trâmite da mobilidade e possibilitando sua abrangência em locais de difícil acesso, oferecendo oportunidades para a população utilizar um transporte de alto custo e ofertando uma manutenção mais viável ao locador por meio da utilização de tal serviço. Complementando com um sistema web informativo do projeto que forneça as instruções de download da aplicação.

Contudo para a construção do projeto, será necessário a realização de pesquisas em livros, artigos e sites, seguida pela análise dos dados obtidos para interpretação e aplicação no mesmo, além do estudo aplicado ao mercado atual de ramo semelhante.

Assim visando minimizar as dificuldades de atingir as regiões centrais às mais periféricas, viabilizando o uso dos veículos automotores, com a redução de custos deste tipo de locação.

2 REFERENCIAL TEÓRICO

Neste presente capítulo, será apresentado as tecnologias e processos condizentes ao desenvolvimento, sobretudo dispõe das informações utilizadas como respaldo teórico na elaboração, documentação e execução deste projeto.

2.1 Dificuldades na Locomoção

As consequências da pandemia na sociedade e os impactos na mobilidade pública, onde com a necessidade de se proteger, coatou os cidadãos a migrarem para meios de locomoção mais seguros, confortáveis e limpos AVELEDA (2023 apud G1 2023)

Uma pesquisa realizada mostra que 33,6 milhões de brasileiros não possuem acesso ao transporte coletivo IBGE (2017 apud Folha de S.Paulo 2023). Grande parte da população de grandes cidades reside em locais distantes de seus respectivos empregos, além de relatar que em média as pessoas levam cerca de mais de duas horas para cumprir o seu trajeto diário Agência EY (2023 apud Terra 2023).

Visto a degradação da locomoção urbana comum, permite a idealização de um projeto consistente a uma aplicação mobile, capaz de facilitar e agilizar a mudança para os automóveis particulares. Sendo esta, construída e planejada com capacidade de abranger a todos aqueles que precisam se deslocar nas metrópoles ou não.

2.2 Tecnologias Utilizadas

A partir deste momento será apresentado as tecnologias em específico, utilizadas na produção do projeto, nas quais consistem em linguagens e documentos.

2.3 HTML

Com o intuito de construir páginas web o *Hyper Text Markup Language* (HTML), refere se a uma linguagem de marcação de hipertexto, foi concebida com o intuito de estruturar páginas da web (EIS; FERREIRA, 2012).

O conceito de hipertextos, é todo elemento dentro de uma página web que pode se correlacionar com outro documento web por meio de links interligados (SILVA, 2008). De forma a serem lidas universalmente pelos navegadores, onde antes de sua criação era necessária uma tecnologia específica para cada navegador (EIS; FERREIRA, 2012). Sendo isto possível, através das *tags* HTML, são marcações essenciais para estruturar e ditar como os elementos, serão dispostos na tela do usuário (ALVES, 2021). As *tags* e instruções principais para a criação de documentos HTML são:

* “*<!DOCTYPE html>”* informa para o navegador a versão em que a marcação foi escrita para um melhor processamento digital da página.
* “*<html>*” é a principal, iniciada após o *Doctype*, cerca todas as marcações indicando o início e o fim da codificação.
* “*<head>*” é referente ao cabeçalho do site, ela armazena informações importantes para a configuração da página.
* “*<body>*” é o corpo do documento, é aqui que os elementos visíveis aos usuários são construídos.

Estas tags, estão sendo apresentadas na estrutura inicial de uma página criado em HTML na figura 1.

Figura 1 – Tags bases de um arquivo HTML



Fonte: Do próprio autor, 2024.

Em continuação a construção utilizando HTML, a figura 2 a seguir, mostra um exemplo da construção de uma página contendo um formulário para login:

Figura 2 – Exemplo de um código HTML para criação da tela de login

Fonte: Do próprio autor, 2024.

As *tags* adicionadas e estruturadas para criação de um design web na figura 2, resultaram na seguinte tela na figura 3:

Figura 3 – Resultado de um código HTML para criação da tela de login

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Em continuidade, após a demonstração do funcionamento de um design com HTML, a análise do código, com foco em cada uma das tags utilizadas.

* “*<form>”* dentro desta *tag* é construído os elementos de um formulário para a coleta e direcionamento de dados do usuário.
* “*h1>*” serve para indicar o título principal. As *tags* *h* podem ir de “*<h1>*” sendo a maior e “*<h6>*” a menor fonte para diferenciar os textos dentro de um site.
* “*<p>*” tem a função de criar um parágrafo.
* Essa tag “*<br>”* trata-se da realização da quebra de linha, ou seja, o conteúdo é mandado para uma linha abaixo.
* *“<input>” é* responsável por criar as caixas de textos para a entrada de dados dos usuários.
* “<a>” serve para criar um *hyperlink* específico para referenciar elementos externos ou internos de um documento através da URL ou o nome de um arquivo, informado no atributo “*href* “da *tag*.
* “*<button>*” é permitido criar botões que realizem as mais diversas funções para uma maior interação do usuário com o site.

2.4 CSS

Com foco na estilização das páginas web, o *Cascading* *Style* *Sheet* (CSS) tem o objetivo de formatar e estilizar elementos escritos por uma linguagem de formatação como HTML, (EIS; FERREIRA, 2012). Sendo uma linguagem declarativa e flexível, que separa o material do respectivo site de sua representação visual, podendo alterar fontes e suas colorações, (SCHEIDT, 2015). A estilização da página advém de três fontes responsáveis diversificadas entre si, que são: autor, usuário e agente de usuário, respectivamente o desenvolvedor, o visitante da página e o navegador (SILVA, 2007).

Para que os atributos de CSS sejam aplicados é necessário a conexão do HTML com o CSS através da *tag* link inserida dentro da tag “*<head>*”, a linha contendo esta tag, está selecionada em azul claro, tal como exemplifica a figura 4:

Figura 4 – Tag do cabeçalho do formulário

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Acima temos a *tag* link referenciando o arquivo CSS que será chamado, e em casos em que o arquivo estiver em uma subpasta do projeto, deve-se adicionar o nome da pasta seguido de barra antes do nome do arquivo.

A seguir na figura 5, teremos a imagem com a demonstração da estilização do HTML, já mostrado anteriormente com alguns atributos.

Figura 5 – Tags básicas de CSS

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Junto de seus atributos, as respectivas funções de cada individualmente, que seriam estes a seguir:

* “*<body>*” de HTML, onde todas as propriedades selecionadas configuram o corpo da página.
* “*display”* define os tipos de exibição externos e internos de um elemento, nesse caso a *tag* *flex* rearranja os elementos responsivos dentro do *container,* para que sejam preparados automaticamente de acordo com o tamanho da tela.
* *“justify-content”* é usada para alinhar os elementos, cuja direção (seja em linhas ou colunas) é definida a partir da propriedade *flex-direction*.
* “*align-items*” tem a função de alinhar os elementos no eixo transversal, nesse caso no centro da página.
* *“height”* define o tamanho vertical ocupado por uma *tag* na tela do navegador.
* *“margin”*: a propriedade *margin* do CSS define a área de margem nos quatro lados do elemento. É uma abreviação que define todas as margens individuais de uma só vez: *margin-top*, *margin-right*, *margin-bottom*,e *margin-lef.*
* “*background-color*” define a cor de fundo de um elemento, nesse exemplo a página inteira.
* “*font-family*” permite que se faça uma lista de prioridades de famílias de fontes e/ou nomes genéricos de famílias a serem especificados para um elemento selecionado.
* *“width*” a *tag* determina a largura da área de conteúdo de um elemento.
* *“padding”* define uma a distância entre o conteúdo de um elemento e suas bordas e um atalho que define uma distância para cada lado em conjunto ao invés de defini-los separadamente *(padding-top,* *padding-right,* *padding-bottom,* *padding-left).*
* *“box-**shadow”* é uma ferramenta que permite criar sombras de forma simples e flexível.
* *“container*” é a *tag* criada para referenciar uma *div* em HTML com a *tag* *class*, assim permitindo fazer alterações nos elementos da própria.
* *“form”* é a *tag* criada para referenciar o formulário construído no HTML.
* “*text-align*” permite alinhar textos dentro de um container no centro e aos lados.
* “*font-size*” altera diretamente o tamanho da fonte de um texto.
* “*text-decoration*” a propriedade desenha em elementos descendentes, como a linha inferior vista em links.
* “*border*” é uma propriedade que adiciona um contorno ao redor de um elemento HTML, mas possui a possibilidade de contornar uma única extremidade separadamente utilizando, *border-top, border-bottom, border-left, border-right*.

Logo após na figura 6, temos o código completo do CSS do formulário de login, em conjunto com *tags* adicionais:

Figura 6 – Tags do formulário completo

Texto

Descrição gerada automaticamente  
Fonte: Do próprio autor, 2024.

Logo a seguir na figura 7, o resultado da página estilizada, combinando as tecnologias já mostradas:

Figura 7 – Resultado da estilização HTML para criação da tela de login

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

2.5 JavaScript

Caracterizada por sua linguagem simples e fácil de compreender, o *JavaScript* é uma das bases para o desenvolvimento web, uma linguagem orientada a objetos, suportando uma variedade de estilos de programação, (BIFFI, 2018).

Criada pela Netscape na fase inicial da Web e licenciada pela Sun Microsystems, com o intuito de fazer interseções além de proporcionar uma página web mais dinâmica, sendo uma linguagem de alto nível, (FLANAGAN, 2013). Após múltiplas pesquisas sobre linguagem de programação web o *JavaScript* se destaca mundialmente sendo assim uma linguagem muito valorizada no mercado de trabalho junto com HTML e CSS, (IEPSEN, 2022).

A seguir um exemplo de codificação na figura 8 com *JavaScript,* em continuidade ao exemplo desenvolvido nos capítulos anteriores:

Figura 8 – Exemplo de validação de formulário

****

Fonte: Do próprio autor, 2024.

Seletores e Manipulação do Docment Object Model (DOM) fazem uso da biblioteca *jQuery* para selecionar elementos do DOM e manipulá-los. Isso é evidente na linha onde “$('.input100')” é usado para selecionar todos os elementos com a classe “*input100*”. O *jQuery* simplifica a manipulação do DOM, oferecendo uma sintaxe mais concisa e amigável do que o *JavaScript* puro.

Os Eventos de Formulário respondem a eventos de formulário, como “*blur*” e “*submit*”.

* O evento “*blur*” é acionado quando um elemento perde o foco.
* Enquanto o evento “*submit”* é acionado quando um formulário é enviado. Isso demonstra a interatividade do código em resposta às ações do usuário.

Logo em sequência, a continuação do código em JavaScript, que irá realizar a validação dos inputs, na figura 9.

Figura 9 – Continuação do exemplo de validação de formulário

****

Fonte: Do próprio autor, 2024.

A validação do formulário é realizada no lado do cliente, antes que os dados sejam enviados ao servidor. Isso é uma prática recomendada para fornecer feedback imediato ao usuário e reduzir a carga no servidor. A validação é feita verificando se os campos estão vazios ou se o formato do campo de email está correto.

O código utiliza expressões regulares para validar o formato do campo de email. Expressões regulares são padrões de texto que descrevem um conjuntode *strings* possíveis.

* No código, a expressão regular /^.+@.+\..+$/` valida se uma *string* corresponde ao formato de um endereço de e-mail válido.

2.6 Framework

Um *framework* é a abstração, logo uma estrutura que tem como objetivo representar o início do desenvolvimento de um software (GABARDO, 2017). *Esta estrutura possui códigos pré-estabelecidos, a fim de agilizar o processo de desenvolvimento de aplicações de pequeno a grande porte* (TURINI, 2015).

Como exemplo de *Framework*, temos o *React Native* que se trata de um conjunto de ferramentas que tornam viável a criação de aplicações mobile para IOS e Android (ESCUDELARIO; PINHO 2020, p.1).

2.7 Node e NPM

Uma ferramenta de código aberto chamada de *Node.js, que* conduz aplicações *JavaScript*, a possuir um bom desempenho em relação ao processamento do lado dos servidores, permitindo que o sistema responda de forma eficiente (PEREIRA, 2013).

O *Node.js* possui um gerenciador de pacotes, *Node Package Manager* (NPM), que tem como objetivo armazenar bibliotecas e dependências, criada pela comunidade de desenvolvedores (ALMEIDA, 2015). Tal como exemplifica a figura 10, alguns dos comandos mais utilizados durante o desenvolvimento usando Node.JS.

Figura 10 – Exemplo de comandos utilizando o NPM

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Em suas novas versões, foram desenvolvidas para o uso da V8 engine, que é compilada pelo Just-in-Time(JIT), logo a compilação de código JavaScript são traduzidas diretamente para linguagem de máquina pelo navegador (POWERS, 2019).

2.8 React

Baseada em *JavaScript* e desenvolvida pelo Facebook, a biblioteca React trata-se de simplificar a criação de Interfaces com o uso da linguagem de marcação HTML, capacitando a criação simples de aplicações de grande porte (STEFANOV, 2019). Otimiza a integração de diversas tecnologias, possibilitando aplicar mais facilmente outros *frameworks* de *JavaScript*, além de estabelecer melhores conexões com banco de dados *real-time* (SILVA, 2021).

Quando lançada em 2015, sem contar a inovação com o projeto de linguagem nativa, adentrou no mercado com objetivo de competir com as existentes no mercado, como Vue e Angular (SOUZA, 2019).

Quando se inicia um projeto usando a linguagem *React*, ele realiza a criação do mesmo com uma arquitetura de pastas, onde realiza o *download* de dependências, configurações e telas a serem definidos, conforme exemplificado a seguir na figura 11.

Figura 11 – Exemplo de estrutura de um Projeto em React

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Posteriormente, partindo para um exemplo de projeto desenvolvido utilizando esta biblioteca, nas primeiras linhas, são realizadas as importações, de elementos, bibliotecas e configurações, no arquivo App.js, como mostra a figura 12.

Figura 12 – Exemplo de estrutura de código para iniciar um Projeto em React

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Logo em sequência, a criação de uma função formulário onde define-se as variáveis de armazenamento dos dados digitados em um formulário HTML, logo abaixo na figura 13.

* “*handleLogin*” que irá exibir as informações digitadas.
* “*limparDados*” irá limpar os dados inseridos nas caixas de texto.

Figura 13 – Exemplo de estrutura de código da execução a aplicação

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

A função principal que executa a aplicação, com uma estrutura HTML, contudo com o uso da função Formulário, sendo criada propriamente para requisitar o formulário, ao adendo que sua estilização ocorre em CSS no arquivo App.css.

Posteriormente o resultado, por meio da demonstração e estrutura e código de um formulário, incluindo as tecnologias de HTML e CSS, na figura 14.

Figura 14 – Formulário em *React*

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

2.9 React Native

Em busca de uma maior adaptabilidade aos sistemas operacionais diversos e melhor aplicabilidade surge-se um *framework* de *JavaScript* chamado *React Native* (ESCUDELARIO; PINHO, 2020).

Nascido na conferência da empresa Facebook em 2015, o *React Native*  tinha por objetivo simplificar o mercado de programação de aplicativos mobile e unificar os sistemas de execução nativos (SILVA, 2022). Capaz de suportar a criação de apps híbridos, ou seja, executáveis tanto em IOS quanto em Android (ESCUDELARIO; PINHO, 2020). Tal tecnologia extingue a necessidade de interpretadores, por meio da técnica *cross-plataform*, logo sendo reconhecida com linguagem nativa, contendo uma base forte em seu antecessor *React* (BEZERRA, VIANA, 2021).

Nesta seção, será descrito o funcionamento de um projeto, criado na linguagem *React Native*. A começar pela estrutura, logo que se cria um projeto, ele estabelece uma arquitetura de pastas, contendo as dependências, configurações e telas do Projeto, como visto na figura 15 abaixo.

Figura 15 – Exemplo de estrutura de um Projeto em *React Native***Texto

Descrição gerada automaticamente**

Fonte: Do próprio autor, 2024.

Já na estrutura de codificação, por ser um *framework* baseado em *React*, é necessário importar o esqueleto e os elementos de código da linguagem pai, nas linhas *import, tal como mostra a figura 16*.

Figura 16 – Exemplo de código para iniciar um projeto em React Native

**Texto

Descrição gerada automaticamente**

Fonte: Do próprio autor, 2024.

* “*App()*” função que delimita a criação da aplicação, e define as variáveis constantes, que são usadas para manipular os dados inseridos nas caixas de manipulação e texto.
* “*handlelogin*” função que determina a criação de uma caixa de diálogo suspensa, onde lista os dados digitados,
* **“***limpardados*"deleta os textos das caixas de texto.

A seguir na figura 17, a criação da tela principal da aplicação utilizando as tags, do React Native.

Figura 17 – Exemplo para definir elementos de um projeto em React Native

**Texto

Descrição gerada automaticamente**

Fonte: Do próprio autor, 2024.

* “*return*” função que retorna a função principal o que deve ser executado, e irá enviar os componentes dos formulários, que funcionam com *tags* semelhantes a linguagem HTML.
* “*ScrollView*” elemento que descreve uma tela que possa ser rolada de baixo para cima e vise e versa, dentro dele.
* “*Text*” onde dentro da *tag*, comporta o texto a ser apresentado, com o atributo “*style***”**, que define a estilização, requisitando a função de estilização do objeto “*label*” na seção de estilização.
* “TextInput”, representa as caixas de entradas de entradas de informações, que apresenta atributos, O atributo “*value***”** atribui a uma variável o valor digitado na caixa, a fim de ser manipulado.
* "*secureTextEntry***”**, que transforma os textos digitados em asterisco, por se tratar de senha, um campo de segurança.

Em sequência a apresentação dos botões do formulário, tanto de envio quanto para limpar as caixas de entrada de dados, por meio da figura 18.

Figura 18 – Exemplo para definir Botões em um Projeto em React Native

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

* O “*Button***”** *representa* um botão na tela, porém não é personalizável, é declarado as características, “*title*” e “*onPress*”, onde respectivamente definem, o texto que será exibido no botão e a função que será acionada ao pressioná-lo.
* Diferente do “*Button***”**, o “*TouchableOpacity***”**, é um botão pode ser personalizado, e considera os atributos “*style*”e “*onPress*”, contudo dentro da *tag*, é definido o texto a ser exibido por meio da tag “*Text***”**.

Já na figura 19, o código estabelece uma constante “*styles*”, e atribui um método que permite a estilização de objetos, com a mesma sintaxe do CSS para web. Com base nos tipos de elementos, no arquivo StyleSheet, a estilização dos elementos dos formulários.

Figura 19 – Exemplo para definir estilização em um projeto em React Native

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

2.10 Expo Go

Uma das formas de se executar as aplicações feitas utilizando React Native, é usando o Expo CLI, comumente usada por desenvolvedores inexperientes, onde para utilizar esta interface, necessita-se de comandos no prompt de comando (ARAUJO, 2019). A plataforma Expo Go, trata-se de um ambiente de compilação, execução, build e teste de softwares, logo um compilador, com versionamento em mobile e desktop para a linguagem *React Native* (ESCUDELARIO; PINHO, 2020).

Tal compilador de *JavaScript* e *Typescript*, possibilita a inclusão de API’s externas facilmente, sem alteração do código nativo e instalação de dependências (FUENTES, 2023).

Por fim na figura 20, obtém-se o resultado do formulário, codificado anteriormente em React Native.

Figura 20 – Formulário em React Native

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

2.11 Typescript

Em meados de 2012, é lançada ao público, uma linguagem que se traduz em um pré-processador de códigos chamado *Typescript*, mantida pela Microsoft, possui o proposito alcançar o desenvolvimento simples ao complexo (ADRIANO, 2021).

Tal linguagem permite a mutação de *JavaScript* de *tipagem* dinâmica para *tipagem* estática, assegurando a integridade das informações que são manipuladas, entre funções e variáveis (FUENTES, 2023). Os códigos produzidos em Typescript, são feitos em orientação a objetos e depois faz uma conversão do código Typescript para o Javascript (ESTEFANI, 2019).

Conforme fora relatado, neste exemplo da figura 21, se utiliza o objeto de tipo *interface* com nome Pessoa, propriamente *Typescript*.

Figura 21 – Exemplo de estrutura de um código em Typescript

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Logo após a definição do objeto pessoa se requisita os atributos em específico, do objeto na função saudação. Este exemplo exige que o objeto requisitado possua todos os atributos de acordo com sua definição antecedente, onde é demonstrado na figura 22.

Figura 22 – Execução de um código em Typescript



Fonte: Do próprio autor, 2024.

Já no exemplo seguinte da figura 23, o objeto de Maria foi definido sem o atributo idade, ignorando as declarações do objeto.

Figura 23 – Exemplo de estrutura de um código em Typescript com erro

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

No editor de código é reconhecido como erro, e logo após é identificado na execução no terminal, devido a sua alta sensibilidade, a linguagem não executa, com um campo nulo, assim como exibe a figura 24.

Figura 24 – Resultado do código em Typescript com erro

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

2.12 JSON

A linguagem *JavaScript Object Notation* (JSON) é uma forma de notação de objetos [*JavaScript*](https://www.alura.com.br/artigos/javascript), de modo que eles possam ser representados de uma forma igual a diversas linguagens (SMITH, 2015). Esta é uma linguagem que tem por função realizar o intercâmbio de dados, logo efetua a conexão entre diversas linguagens dentro de um mesmo projeto ou entre diversos sistemas, assim como XML (BASSETT, 2015).

Normalmente utilizada em conjunto com *JavaScript* para construção de aplicações web, ou outros tipos de sistema, suas conexões entre linguagens condizem até com a implementação de Banco de Dados (SMITH, 2015).

Através do código da figura 25, é demonstrado a configuração do aplicativo desenvolvido em *React Native*, como nome, ícone, versionamento, dentre outras ordenações.

Figura 25 – Exemplo de estrutura de um código em JSON

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

* “*Expo*” é um objeto definindo o aplicativo em geral, tal qual o nome, versão, ícone, dentre outros.
* “*assets*” trata-se do diretório no qual será importado imagens e outros componentes semelhantes.
* “*ios, android e web*” condizem com as definições dos dispositivos suportados.
* “*plugins*” dependências externas.
* “e*xperiments*” definem os tipos rotas de navegações entre as telas.

2.13 API

As Interfaces de Programação de Aplicações (API), condizem com códigos prontos, disponibilizados para desenvolvedores, realizarem requisições das tais, a fim de implementar novos recursos em suas aplicações (PAVANELI, 2023). De forma simples, as API são formas de conectar diversas aplicações utilizando um meio de comunicação comum, logo uma linguagem de programação (SILVA, 2023).

Os recursos providos pelas *API’s* são disponibilizados na internet para reutilizá-las em diversos projetos sem a necessidade de escrever os códigos novamente, como pré-requisito é necessária conexão com a internet (CAMPOS, 2013).

2.14 Banco de Dados

Expressando de uma forma simples e direta, um banco de dados é a junção de dados relacionados entre si, dados esses que seriam fatos, que podem ser guardados e possuem uma definição subjetiva (ROB; CORONEL, 2010). A criação de um sistema de banco de dados visa suprimir as necessidades do homem de registrar dados de diversas situações, como datas e eventos, com suas devidas classificações (ALVES, 2020). 

Uma entidadeé algo do mundo real que possua uma existência independente, como uma empresa, em alguns casos até uma aplicação, enquanto atributos são aspectos da entidade (COSTA, 2011). A chave primária é um campo especial que define uma identidade única para cada registro e contribui para a integridade dos registros alocados no banco e trazendo segurança para as aplicações (NIELD, 2016).

Sistema de Gerenciamento de Dados (SGBD) condiz com um *software* que faz o intermédio da comunicação entre o armazenamento e manutenção de um volume de dados, por meios de códigos (Ramakrishnan; Gehrke, 2019).

Um Banco de Dados Não Relacionalfoi desenvolvido para trabalhar com as demandas de aplicações da nuvem e criado para ultrapassar a média de desempenho, modelagem de dados e limitações de bancos relacionais (GONÇALVES, 2016).

2.15 Firebase

*Firebase* é uma *suit* de base de dados hospedada em nuvem que oferece uma infraestrutura com ferramentas que possibilitam uma melhor qualidade e escalabilidade (MACHADO, 2021). O *Firebase* pode ser seccionado em partes, onde dentre elas estão, o Cloud Firestore e o Real-Time database, assim a o Cloud Firestore por também possuir uma base de dados offline, acaba por se sobressair (FIREBASE, 2022).

Enquanto uma aplicação estiver sendo usada, com conexão na internet, o Cloud irá realizar atualizações em tempo real na base de dados, e quando não houver conexão, os dados serão armazenados localmente, e atualizados depois (ALVES, 2021). O *Cloud Firestore* é um banco de dados flexível e escalonável para desenvolvimento focado em dispositivos móveis, Web e servidores pelo *Firebase* e o Google Cloud (FIREBASE, 2022).

Como fora descrito a respeito da comunicação entre a aplicação e o cloud firestore, pode ser melhor compreendido, pela figura 26.

Figura 26 – Demonstração gráfica da comunicação com Cloud Firestore

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

2.16 Wireframe

Os *wireframes* são um rascunho dos elementos e requisitos essenciais de uma determinada tela, bem como a definição da hierarquia e posição dos conteúdos na interface (MEMÓRIA, 2014). Os *wireframes* são desenvolvidos de forma sucinta, pois o objetivo é determinar as funções que permanecerão no layout final e não o seu design (SILVA; STATI, 2022).

Assim como é mostrado na figura 27, um exemplo de wireframe de baixa fidelidade de uma aplicação mobile.

Figura 27 – Exemplo de Wireframe de baixa fidelidade

Uma imagem contendo Padrão do plano de fundo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Além da estrutura da tela, estes protótipos evitam atrasos e permite ajustes rápidos, o que diminui as alterações no design final, ou seja, ele o rascunho que dará origem ao *layout* final (TEIXEIRA, 2014).

Posteriormente após os esboços iniciais, é necessário a construção das telas, o mais próximo possível do resultado, em continuidade a figura anterior, a 28 mostra a mesma tela, porém mais elaborada.

Figura 28 – Exemplo de prototipação do Wireframe de alta fidelidade

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Para a prototipação dos *wireframes* usaremos o software figma, uma plataforma bastante eficaz para a confecção das interfaces, principalmente por conter componentes parecidos com os que o *React* utiliza (MARQUES, 2023).

2.17 UML

Para a diagramação do projeto será usado os conceitos da *Unified Modeling Language* (UML), linguagem de modelagem padrão com a função de modelar um *software* baseada no paradigma de orientação a objeto (GUEDES, 2018). Sendo composta por diversos diagramas que atuam na compreensão de um sistema de *software* ou processo de negócios, além de ajudar transmitir ideias sobre o projeto (FOWLER, 2007). Os Diagramas são representações gráficas de um conjunto de elementos (Itens e relacionamentos), que são desenhados para uma visualização do sistema de diferentes perspectivas (BOOCH; RUMBAUGH; JACOBSON, 2012).

De acordo com de UML, utilizaremos o diagrama de caso de uso, documentação de caso de uso, diagrama de sequência, diagrama de atividade e diagrama de máquina de estado, para documenta a construção do projeto.

O Diagrama de caso de uso possui uma linguagem simples e de fácil compreensão para que os usuários possam entender como o sistema irá funcionar (BOOCH; RUMBAUGH; JACOBSON, 2012).

A figura 29 é a demonstração simples de caso de uso, aplicada no planejamento de um sistema de controle de Cinema.

Figura 29 – Exemplo Diagrama de Caso de uso

Diagrama

Descrição gerada automaticamente

Fonte: Guedes, 2018.

O exemplo acima ilustra a interação dos atores com o sistema, sendo o funcionário responsável por todos os casos de uso e o cliente associado ao vender ingresso pois é o único em que interage diretamente (GUEDES, 2018).

Os diagramas de caso de uso servem para analisar como um sistema irá se comportar ao realizar determinada função ao invés de ditar como ele deverá realizá-la (BOOCH; RUMBAUGH; JACOBSON, 2012).

É através da documentação de casos de uso que definimos as etapas que um ator ou sistema deve realizar para a execução de um caso de uso, como no exemplo abaixo, a documentação do caso de uso vender ingresso (GUEDES, 2018). As documentações de caso de uso servem para analisar como um sistema irá se comportar ao realizar determinada função ao invés de ditar como ele deverá realizá-la (BOOCH; RUMBAUGH; JACOBSON, 2012).

Em sequêcia na tabela 1, abrange a documentação de um caso de uso, aplicada no caso de uso, UC01 – Vender Ingresso.

Tabela 1 – Documentação do Caso de Uso vender ingresso

Tabela

Descrição gerada automaticamente

Prosseguindo com o restante das restrições e os possíveis cenários de exceção, segue abaixo a continuação da documentação, por meio da tabela 2.

Tabela 2 – Continuação da documentação do Caso de Uso vender ingresso

Texto

Descrição gerada automaticamente

Fonte: Guedes, 2018.

Em sequência será apresentado, o próximo diagrama a ser utilizado na documentação, o diagrama de atividade. O Diagrama de atividade tem o intuito de descrever os passos a serem percorridos para a conclusão de uma tarefa especifica, sendo está representada por um método com certo grau de dificuldade (FOWLER, 2007).

Na figura 30, a o exemplo de diagrama de atividade possui indicações em cinza claro, que explicam os processos de leitura deste diagrama.

Figura 30 – Exemplo de diagrama de atividade

Diagrama

Descrição gerada automaticamente

Fonte: Booch, 2012

O exemplo acima ilustra como funciona as atividades que devem ser seguidas em uma construção, inicializado na escolha do local, onde também exemplifica como funciona uma ramificação caso o orçamento não seja aceito (BOOCH; RUMBAUGH; JACOBSON, 2012).

Em continuidade como mostra a figura 31, o próximo diagrama a ser realizado durante o processo de documentação, será diagrama de sequência.

O diagrama de sequência captura o comportamento de um único cenário mostrando vários exemplos de objetos e mensagem que são passadas dentro dos casos de uso (GUEDES, 2018).

Figura 31 – Exemplo de diagrama de sequência

*Diagrama, Esquemático

Descrição gerada automaticamente*Fonte: Fowler, 2007

O exemplo acima descreve o processo que o sistema realiza quando há um novo pedido, calculando o preço de um produto e retornando com o devido desconto para o cliente (FOWLER, 2007).

E por fim o último diagrama, o Diagrama de máquina de estado tem como intuito modelar o comportamento individual de um objeto, especificando como ele muda de estado em resposta a eventos ao longo do tempo (BOOCH; RUMBAUGH; JACOBSON, 2012).

Este diagrama, está ilustrado na figura 32, demonstrando visualmente o diagrama de Máquina de estado e seu funcionamento.

Figura 32 – Exemplo de diagrama de máquina de estado

Diagrama

Descrição gerada automaticamente

Fonte: Guedes, 2018

O exemplo acima demonstra os eventos para consulta de um saldo, em que o sistema a partir do número da conta, requisita os processos para executar a função, só executando esses passos, se cada informação estiver correta (GUEDES, 2018).

3 DESENVOLVIMENTO

A partir desta secção estará disposto os processos e etapas realizadas durante o processo de planejamento, em relação a documentação, construção de telas e aprofundamento nos conceitos do projeto, estudando como deverá ser construído cada funcionalidade na aplicação.

3.1 Carchau: Concepção e documentação

Para o sistema de conectividade na locação de automóveis, se vê necessário descrever seus requisitos e funcionalidades para uma compreensão melhor do sistema. Este servirá de ponte para análise e interpretação, a fim de modelar o sistema e construí-lo.

Um aplicativo que permite os usuários alugarem seus carros para outras pessoas. Inicialmente é necessário que os usuários, sejam eles locadores ou locatários, criarem uma conta no aplicativo, fornecendo informações pessoais, como CPF, Nome Completo, Nº do registro de habilitação, detalhes de contato e outros. Por parte do Locador, anúncios detalhados que incluem informações sobre seus carros disponíveis para aluguel, fotos do com o estado de conservação, informações de seguro, disponibilidade, laudo cautelar e preferências.

Partindo do ponto de vista do locatário, eles serão capazes de pesquisar carros disponíveis para aluguel em sua área com base em critérios como localização, modelo, preço e disponibilidade de datas. Eles também podem reservar o carro desejado diretamente pelo aplicativo, para que assim que realizem o pagamento da caução e alugue.

Para que haja o aluguel, será exigido um processo de verificação de identidade seguro, para garantir que tanto o locatário quanto o proprietário do carro sejam quem afirmam ser. Partindo do ponto em que o aluguel, já foi feito e o caução depositado, após chegarem a um ponto de encontro, será disponibilizado o botão em que confirme a entrega do veículo para o locatário e para o mesma o botão de confirmação de recebido. Já após o período de aluguel, posteriormente ao pagamento, será exibido as telas de confirmação de entrega para o locatário e de recebido para o locador, encerrando o processo de locação.

Com o propósito de realizar a manutenção do ambiente, haverá um sistema de avaliação e comentários onde os locatários e proprietários podem avaliar uns aos outros após cada transação. Isso ajuda a construir uma comunidade confiável e permite que os usuários façam escolhas informadas ao escolher um carro para alugar.

Todos os processos relacionados a locação de automóveis, estarão em função da concordância em que locatários e proprietários assinem contratos de locação e concordem com os termos de serviço diretamente pelo aplicativo, detalhando responsabilidades, condições de seguro e demais pendências importantes no tramite.

A aplicação pode enviar notificações e lembretes automáticos aos usuários sobre datas de locação, devolução do carro, pagamentos pendentes e outras informações relevantes. Com a relevância de um sistema de suporte ao cliente integrado para ajudar os usuários com quaisquer dúvidas, problemas ou questões que possam surgir durante o processo de locação.

3.2 Levantamento de Requisitos

Após ser detalhado a concepção do projeto Carchau, se vê necessário descrever as funcionalidades, extraindo do contexto da ideia as funcionalidades que devem estar presentes na aplicação, tantos as que devem interagir diretamente com os usuários, quanto as que devem estar presentes em forma de conceitos aplicados.

3.3 Requisitos Funcionais

Os requisitos funcionais descrevem as funções da aplicação que irão interagir diretamente com o usuário, de forma análoga a botões, telas pop-up, cadastros ou alterações. Onde estas funcionalidades estão descritas e enumeradas, com seus respectivos códigos, na qual será ferramenta de auxílio na compreensão da documentação dos casos de uso.

RF01 – Cadastrar Conta

Esta funcionalidade se relaciona com realizar o cadastro de usuários, tanto locadores quanto locatário na plataforma. Fazendo o registro das informações pessoais, descrições dos veículos e entre outros dados.

RF02 – Fazer login

Esta funcionalidade descreve a entrada na aplicação após ter realizado a criação de uma conta, com o CPF ou CNPJ e a sua senha de acesso.

RF03 – Exibir Mapa

A partir da tela do locatário, inicialmente estará disposto um mapa com os veículos disponíveis para aluguel nas proximidades, para uma melhor compreensão dos endereços jonde se encontram os carros.

RF04 – Visualizar Veículo

Na tela inicial, estará disposto, além do mapa, uma listagem dos veículos com algumas respectivas informações sobre eles.

RF05 – Pesquisar Veículo

Em caso de preferência de algum veículo, será possível pesquisá-lo, em uma barra de pesquisa, para alugá-lo.

RF06 – Realizar Solicitação de Aluguel

Após o momento de seleção de um veículo, o locatário terá a possibilidade de solicitar a locação do veículo ao locador.

RF07 – Alocar Veículo

Este será um botão exibido ao locador, para atualizar o estado do anúncio de locação, de disponível para alugado.

RF08 – Realizar Caução

Esta funcionalidade descreve a função de realizar o pagamento de um valor como garantia, antes de alugar um veículo.

RF09 – Visualizar Chat

Antes mesmo de realizar a solicitação do aluguel, estará disponível um chat para negociação entre locador e locatário.

RF10 – Confirmar Entrega

Após a confirmação do aluguel, no momento do ponto de encontro, o locador poderá confirmar a entrega do automóvel ao locatário. O mesmo vale para o momento de devolução, com a diferença de que os papeis irão se inverter.

RF11 – Confirmar Recepção

No momento do tramite do aluguel, após o veículo ser entregue, tanto a locatário quanto ao locador, será exibido o botão para confirmar a recepção do automóvel.

RF12 – Realizar Pagamento

A tela de posterior a de realizar o aluguel será a de realizar o pagamento, pois para que seja possível devolver o veículo, será necessário pagar conforme a modalidade acordada entre os envolvidos.

RF13 – Devolver Caução

Após o encerramento do tramite do aluguel, a contar um período, o dinheiro será devolvido ao locatário.

RF14 – Cadastrar Avaliação

Depois do período de aluguel haverá possibilidade de avaliar o locador e o locatário.

RF15 – Exibir Solicitação de Locação

Partindo para as telas do locador, ao iniciar o aplicativo, estará disposto as solicitações de aluguel feitas a respeitos dos seus veículos.

RF16 – Aceitar Solicitação

Para cada Solicitação, haverá duas opções, uma delas a de aceitar a solicitação, para depois realizarem negócio.

RF17 – Rejeitar Solicitação

A segunda opção é a de rejeitar a solicitado, onde ela será deletada.

RF18 – Manter Veículo

Um locador poderá cadastrar mais de um veículo, e administrar os perfis de cada veículo, logo deletar, alterar, criar ou exibir os perfis de seus veículos.

RF19 – Enviar Laudo Cautela

Para que cada veículo cadastrado, será necessário enviar a plataforma o seu respectivo laudo cautelar, para que esteja apto a operar.

RF20 – Enviar Imagens do Estado do Veículo

Para cadastrar um veículo, o locador deverá realizar o cadastro das imagens do estado de conservação do automóvel.

RF21 – Manter Modalidade

Uma das informações cadastradas nos perfis dos veículos é a modalidade, que poderá ser alterada.

RF22 – Visualizar Conta

Os dois tipos de usuário poderão visualizar seus perfis, e suas respectivas informações.

RF23 – Editar Conta

Os dois tipos de usuário poderão editar as respectivas informações de seus perfis.

3.4 Requisitos Não Funcionais

Há também os requisitos não funcionais, que tratam dos conceitos que devem estar presente durante a criação de um sistema, estes representam as funcionalidades que não interagem diretamente com o usuário, mas sua presença é fundamental para uma aplicação eficaz.

RNF01 – Calcular locação

A começar que o sistema deve calcular um possível valor de locação, com base na modalidade, no intuito de auxiliar o locador sugerindo o preço a cobrar.

RNF02 – Velocidade de navegação

A aplicação deve ser otimizada, para não conter delays ou travamentos durante seu uso.

RNF03 – Disponibilidade

A aplicação deve estar disponível a vários tipos de smartphone, tanto aos sistemas operacionais quanto em modelos de celulares.

RNF04 – Responsividade

O sistema deve estar responsivo a diferentes tamanhos de telas, e adaptar os seus componentes de acordo.

RNF05 – Segurança

O sistema tem que assegurar as informações dos usuários e a sua integridade.

RNF06 – Visibilidade

A aplicação deve possuir um design, com formas, cores e tamanhos para tenha uma boa visibilidade.

RNF07 – Acessibilidade

A plataforma deve apresentar seus recursos de maneira intuitiva, para que todos os tipos usuários consigam fazer uso da plataforma.

RNF08 – Assertividade em pesquisa

Nas ferramentas de pesquisa, quando usadas, devem apresentar resultados assertivos com os pesquisados.

3.5 Regras de Negócios

As regras de negócios definem as normas que serão seguidas e como serão seguidas pelo sistema, estruturando a aplicação com base no negócio da empresa ou projeto que estiver sendo seguido. As regras relacionadas com a Carchau estão dispostas a seguir:

RN01 – Para realizar a locação é necessário realizar o Caução.

RN02 – O Caução é devolvido assim que a locação for finalizada.

RN03 – É necessário realizar o upload do laudo cautelar, antes de alugar.

RN04 – Pode ser realizada mais de uma solicitação de locação, mas elas expirarão com o passar de x dias.

RN05 – Realizar o caução antes do aluguel e após a pagamento efetua-se o aluguel.

RN06 – O Cálculo do aluguel tem como base a modalidade escolhida e o tempo de aluguel .

4 CONSIDERAÇÕES FINAIS

REFERÊNCIAS

ADRIANO, Thiago da Silva. **Guia prático de TypeScript**: melhore suas aplicações javascript. São Paulo: Casa do Código, 2021. 209 p.

ALMEIDA, Flávio. **Mean: Full stack Javascript para aplicões web com Mongodb, Express, Angular e Node**. São Paulo: Casa do Código, 2015. 367 p.

ALVES, William Pereira. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020. 368 p.

ALVES, William Pereira. **HTML & CSS - Aprenda como construir páginas web**. [S.L.]: Expressa, 2021.

BASSETT, Lindsay et al. **Introdução ao JSON**: um guia para json que vai direto ao ponto. São Paulo: Novatec, 2015. 152 p.

BIFFI, Kelvin Baumhardt. **Javascript: Básico ao Avançado**. São Paulo, 2018. 122 p. Ebook. Disponível em: <https://www.amazon.com.br/Javascript-Avançado-Kelvin-Baumhardt-Biffi-ebook/dp/B0BS9VQTJW>. Acesso em: 14 de maio de 2024.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML** – Guia do Usuário.

2. ed. Rio de Janeiro: Elsevier Editora Ltda, 2012. 552 p.

DATE, Christopher J. **Introdução a Sistemas de Bancos de Dados**. 8. ed. Rio de Janeiro: Gen, 2004. 896 p.

EIS, Diego. **HTML5 e CSS3 com Farinha e Pimenta**. Santa Catarina: Clube de Autores, 2012. 219 p.

ELMASRI, Ramez; B.NAVATHE, Shamkant. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Education Brasil Ltda, 2010. 808 p.

ESCUDELARIO, Bruna; PINHO, Diego. **React Native**: desenvolvimento de aplicativos mobile com react. São Paulo: Casa do Código, 2020. 189 p.

FLANAGAN, David. **JavaScript**: o guia definitivo. 6 ed. Porto Alegre: Bookman Companhia Editora Ltda, 2012. 1080 p.

FOWLER, Martin Albert. **UML Essencial**: Um breve guia para a linguagem-padrão

de modelagem de objetos. 3. ed. Porto Alegre: Bookman, 2007. 161 p.

GABARDO, Ademir Cristiano. **Laravel Para Ninjas**. São Paulo: Novatec, 2017. 184 p.

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

Novatec Editora Ltda, 2018. 496.p

HEUSER, Carlos Alberto. **Projeto de banco de dados**: volume 4 da série livros didáticos informática ufrgs. 6. ed. São Paulo: Bookman, 2009. 282 p. Disponível em: <https://www.google.com.br/books/edition/Projeto\_de\_banco\_de\_dados\_Volume\_4\_da\_S/UKtB7\_MnWQMC?hl=pt-BR&gbpv=0>. Acesso em: 04 maio 2024.

IEPSEN, Edécio Fernando. **Lógica de Programação e Algoritmos com JavaScript**: uma Introdução à Programação de Computadores com Exemplos e Exercícios Para Iniciantes. 2. ed. São Paulo: Novatec Editora, 2022. 414 p.

MACHADO, Kheronn Khennedy. **Angular 11 e Firebase**: construindo uma aplicação integrada com a plataforma do google. São Paulo: Casa do Código, 2021. 177 p.

MARQUES, Jideon F. **Como ser um desenvolvedor web**. [S.L.]: Clube de Autores, 2023. 207 p.

MEMÓRIA, Felipe. **Design para a internet: projetando a experiência perfeita**. [S.L.]: Gen Ltc, 2005. 192 p.

NIELD, Thomas. **Introdução à Linguagem SQL**: abordagem prática para iniciantes. São Paulo: Novatec, 2016. 144 p. 1ª edição.

PEREIRA, Caio Ribeiro. **Node.js: Aplicações web real-time com Node.js**. São Paulo: Casa do Código, 2013. 186 p.

POWERS, Shelley. **Aprendendo Node**: usando javascript no servidor. 2. ed. São Paulo: Novatec, 2019. 39 p.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistemas de Gerenciamento de Banco de Dados**. 3. ed. São Paulo: Bookman, 2015. 884 p.

ROB, Peter; CORONEL, Carlos. **Sistemas de Banco de Dados**: projeto, implementação e gerenciamento. São Paulo: Cengage, 2010. 744 p. 1ª edição.

SCHEIDT, Felippe Alex. **Fundamentos de CSS**: criando design para sistemas web. Foz do Iguaçu: Outbox Interativa, 2015. 126 p.

SILVA, Jéssica Laisa Dias da; STATI, Cesar**. Prototipagem e Testes de Usabilidade**. [S.L.]: Intersaberes, 2022. 226 p.

SILVA, Maurício Samy. **Criando Sites com HTML**: sites de alta qualidade com html e css. São Paulo: Novatec, 2008. 432 p.

SILVA, Maurício Samy. **Construindo Sites com CSS e (X)HTML**: sites controlados por folhas de estilo em cascata. São Paulo: Novatec, 2007. 448 p.

SILVA, Maurício Samy. **React - Aprenda Praticando**: desenvolva aplicações web reais com uso da biblioteca react e de seus módulos auxiliares. São Paulo: Novatec,

2021. 240 p.

SMITH, Ben. **JSON Básico**: conheça o formato de dados preferido da web. São Paulo: Novatec, 2015. 400 p.

STEFANOV, Stoyan. **Primeiros passos com React**: construindo aplicações web. São Paulo: Novatec, 2016. 248 p.

TEIXEIRA, Fabricio. Introdução e Boas Práticas em Ux Design. São Paulo: Casa do Código, 2014. 262 p.

TURINI, Rodrigo. **PHP e Laravel**: crie aplicações web como um verdadeiro artesão. 2. ed. São Paulo: Casa do Código, 2015. 265 p.

ALVES, Isabel Cristina Araújo Rubim. **VERIFICAÇÃO AUTOMÁTICA GEORREFERENCIADA COM RECONHECIMENTO FACIAL**. 2021. 148 f. TCC (Graduação) - Curso de Engenharia de Computação, Pontifícia Universidade Católica de Goiás, Goiânia, 2021. Disponível em: https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/1622/1/TCC%20II%20-%20Isabel%20Cristina%20Arau%CC%81jo%20Rubim%20Alves.pdf. Acesso em: 10 jul. 2024.

**ARAUJO, Gabriel Rodrigues de.**DESENVOLVIMENTO CROSS-PLATFORM COM REACT NATIVE**: um estudo de caso do aplicativo naveg. 2019. 69 f. TCC (Graduação) - Curso de Sistemas e Mídias Digitais, Universidade Federal do Ceará, Fortaleza, 2019. Disponível em: https://repositorio.ufc.br/bitstream/riufc/45892/3/2019\_tcc\_graraujo.pdf. Acesso em: 10 jul. 2024.**

BEZERRA, Franklyn Seabra Rogério. **Desenvolvimento Nativo vs Ionic vs React Native: uma análise comparativa do suporte à acessibilidade em Android.** 2021. 25 f. Trabalho de Conclusão de Curso (Graduação em Computação)-Universidade Federal do Ceará, Fortaleza, 2021. Disponível em:<https://repositrio.ufc.br/bitstream/

riufc/58979/1/2021\_tcc\_fsrbezerra.pdf>.Acesso em: 15 de Maio 2024.

CAMPOS, Marcos Vinícius Souza. **DESENVOLVIMENTO DE APIS BASEADAS EM REST PARA INTEGRAÇÃO E CONSTRUÇÃO DE APLICAÇÕES**. 2013. 94 f. TCC (Doutorado) - Curso de Análise de Sistemas e Tecnologia da Informação, Faculdade de Tecnologia de Americana, Americana, 2013. Disponível em: http://ric.cps.sp.gov.br/bitstream/123456789/1596/1/20131S\_CAMPOSMarcosViniciusSouza\_CD1589.pdf. Acesso em: 19 maio 2024.

COSTA, Elisângela Rocha da. **BANCOS DE DADOS RELACIONAIS**. 2011. 64 f. TCC (Doutorado) - Curso de Processamento de Dados, Faculdade de Tecnologia de São Paulo, São Paulo, 2011. Disponível em: https://www.fatecsp.br/dti/tcc/tcc0025.pdf. Acesso em: 19 maio 2024.

ESTEFANI, Jonathan da Silva Maciel de. **DESENVOLVIMENTO DE UM PROTÓTIPO DE UM SISTEMA WEB PARA GESTÃO DE TRABALHOS ACADÊMICOS**. 2019. 144 f. TCC (Graduação) - Curso de Tecnologia da Informação e Comunicação, Universidade Federal de Santa Catarina, Araranguá, 2019.

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: 14 de Maio. 2024.

GONÇALVES, André Luiz. **Desenvolvimento de um aplicativo Android utilizando banco de dados não-relacional para organização e controle de presença de um time de futebol**. 2016. 61 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2016 Disponível em: <https://lume.ufrgs.br/bitstream/handle/10183/150930/001009684.pdf

?sequence=1>. Acesso em: 04 maio 2024.

LINS, Gabriel de Souza. **UTILIZANDO REACTJS PARA O DESENVOLVIMENTO DE UM SISTEMA DE ALOCAÇÃO E RESERVA DE SALAS NO CAMPUS DA UFC EM QUIXADÁ**. 2019. 36 f. TCC (Graduação) - Curso de Sistemas de Informação, Computação, Universidade Federal do Ceára, Quixadá, 2019. Disponível em: < https://repositorio.ufc.br/bitstream/riufc/49762/1/2019\_tcc\_gdeslins.pdf>. Acesso em: 14 de Maio 2024.

PAVANELI, Tiago Henrique. **Desenvolvimento de uma API e uma Aplicação Web para auxiliar na análise de dados providos por Aplicação Móvel**. 2023. 75 f. TCC (Graduação) - Curso de Sistemas de Informação, Computação, Universidade Federal de Uberlândia, Uberlândia, 2023. Disponível em: https://repositorio.ufu.br/bitstream/123456789/37886/3/DesenvolvimentoAPIAplicacao.pdf. Acesso em: 17 maio 2024.

SILVA, Felipe Augusto Vasconcelos e. **DESENVOLVIMENTO DE UMA API PARA APLICAÇÃO DE LEARNING ANALYTICS EM DADOS DE EXPERIÊNCIAS DE ESTUDANTES EM PLATAFORMAS DE JOGOS EDUCACIONAIS**. 2023. 45 f. Tese (Doutorado) - Curso de Ciência da Computação, Universidade Federal de Ouro Preto, Ouro Preto, 2023. Disponível em: https://monografias.ufop.br/bitstream/35400000/5380/6/MONOGRAFIA\_DesenvolvimentoAPIAplica%C3%A7%C3%A3o.pdf. Acesso em: 10 jul. 2024.

SILVA, Manoela Gomes da. **Desenvolvimento e avaliação de um aplicativo de celular com uso de inteligência artificial para diagnóstico da etiologia da hemorragia intraparenquimatosa cerebral**. 2022. 86 f. Monografia (Especialização) - Curso de Medicina, Medicina, Universidade de São Paulo, Ribeirão Preto, 2022.Disponível em: <<https://www.teses.usp.br/teses/disponiveis/17/17163/tde-11042023-090144/publico/Dissertacao_Manoela_Gomes_versao_corrigida.pdf>>. Acesso em: 15 de Maio 2024.

FIREBASE. Aprenda os fundamentos. [S.I]. Firebase, 2024. Disponível em:https: https://firebase.google.com/docs?hl=ptbr&\_gl=1\*18pchen\*\_up\*MQ.&gclid=504d29a01bc01aa2d6394f92b0c2ad51&gclsrc=3p.ds. Acesso em 25 maio 2024.

BOCCHINI, Bruno. **SP: transporte público da capital perdeu 30% dos passageiros**: queda foi registrada no período de 2013 a 2022. **Agência Brasil**. 2023. Disponível em: https://agenciabrasil.ebc.com.br/geral/noticia/2023-06/sp-transporte-publico-da-capital-perdeu-30-dos-passageiros. Acesso em: 16 mar. 2024.

GALINDO, Ernesto Pereira; LIMA NETO, Vicente Correia. **A MOBILIDADE URBANA NO BRASIL**: percepções de sua população. Textos Para Discussão, Distrito Federal, v. 2468, n. 2, p. 7-54, abr. 2019. Disponível em: https://portalantigo.ipea.gov.br/agencia/images/stories/PDFs/TDs/td\_2468.pdf. Acesso em: 16 mar. 2024.

PESCARINI, FÁBIO.**Queixas por demora de ônibus disparam na cidade de São Paulo.** Folha de S.Paulo. 2023. Disponível em: https://www1.folha.uol.com.br/cotidiano/2023/04/queixas-por-demora-de-onibus-disparam-na-cidade-de-sao-paulo.shtml. Acesso em: 18 mar. 2024.

**PORTAL G1. O que é carsharing e como essa tendência ajuda a melhorar o trânsito nas cidades.** São Paulo, 13 ago. 2018. Disponível em: https://g1.globo.com/especial-publicitario/em-movimento/noticia/2018/08/13/o-que-e-carsharing-e-como-essa-tendencia-ajuda-a-melhorar-o-transito-nas-cidades.ghtml. Acesso em: 19 mar. 2024.

TERRA: Super app pode amenizar dificuldades do transporte público. São Paulo, 21 set. 2023. Disponível em: https://www.terra.com.br/amp/byte/tecnologia-e-solucoes/super-app-pode-amenizar-dificuldades-do-transporte-publico,245b335fe588

4fec0885648ba50d8e95cpaoxxmg.html. Acesso em: 18 mar. 2024.