**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**ETEC DA ZONA LESTE**

**NOVOTEC Desenvolvimento de Sistemas AMS**

**Alex Expedito Silva Santos**

**Danilo Santos Soares**

**Endrigo Gustavo Brandão de Oliveira**

**DAELINK:**

**Vaga de Emprego para Deficientes**

**São Paulo**

**2024**

**Alex Expedito Silva Santos**

**Danilo Santos Soares**

**Endrigo Gustavo Brandão de Oliveira**

**DAELINK:**

**Vaga de Emprego para Deficientes**

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas da Etec Zona Leste, orientado pelo Prof. Jeferson Roberto de Lima, como requisito parcial para conclusão de curso integrado ao ensino médio.

**São Paulo**

**2024**

**RESUMO**

Este trabalho aborda a inclusão de pessoas com Deficiência (PCD) no mercado de trabalho por meio de um sistema baseado em plataformas digitais que promovem conectividade. As cotas para PCD frequentemente não são preenchidas devido à falta procura das empresas e ao preconceito. Embora existam ações para estabelecercotaspara PCD*,* a inclusão enfrenta desafios significativos, resultando em menores taxas de participação no mercado de trabalho comparadas as de pessoas sem deficiência. O objetivo do estudo é desenvolver uma plataforma digital que promove inclusão profissional para facilitar a integração de PCD no mercado de trabalho. A metodologia empregadainclui a análise das necessidades de PCD e empresas. Os resultados indicam a criação de um protótipo de sistema, composto por um site e um aplicativo que pode ser utilizado para empresas preencherem vagas remanescentes dentro de suas instituições.A sua construção é pensada essencialmente para empresas, consistindo em um site e aplicativo que mostram os principais candidatos para vagas remanescentes de determinadas áreas. Portanto demonstra-se que a plataforma tem a capacidade de estabelecer uma melhor eficácia de inclusão de PCD no mercado de trabalho.

**Palavras-Chaves**: Inclusão; PCD; Mercado de trabalho; Plataformas digitais; Ferramentas

**ABSTRACT**

This work addresses the inclusion of People with disability (PWD) in the labor market through a system based on digital platforms that promote connectivity. Quotas for PWD are often not filled due to a lack of demand from companies and prejudice. Although there are actions to establish quotas for PWD, inclusion faces significant challenges, resulting in lower participation rates in the labor market compared to people without disabilities he objective of the study is to develop a digital platform that promotes professional inclusion to facilitate the integration of PWD into the job market. The methodology used includes the analysis of the needs of PWD and companies. The results indicate the creation of a prototype system, consisting of a website and an application that can be used by companies to fill remaining vacancies within their institutions. Its construction is essentially designed for companies, consisting of a website and an application that showcase the main candidates for remaining vacancies in certain areas. Therefore, the project can be qualified to enhance the inclusion of PWD in the Labor Market.

**Key words:** Inclusion; PWD; Labor Market; Digital platform; Tools

**Lista de Figuras**

[Figura 1 – Exemplo de Diagrama de Caso de Uso 13](#_Toc169813687)

[Figura 2 – Exemplo Documentação Caso de Uso 14](#_Toc169813688)

[Figura 3 – Exemplo Diagrama de Atividade 16](#_Toc169813689)

[Figura 4 – Exemplo Diagrama de Sequência 18](#_Toc169813690)

[Figura 5 – Exemplo de Diagrama de Máquina de Estado 20](#_Toc169813691)

[Figura 6 – Exemplo Wireframde de Baixa Fidelidade 22](#_Toc169813692)

[Figura 7 – Exemplo Wireframe Baixa 2 22](#_Toc169813693)

[Figura 8 – Exemplo Html 24](#_Toc169813694)

[Figura 9 – Exemplo HTML 3 25](#_Toc169813695)

[Figura 10 – Exemplo Formulario 26](#_Toc169813696)

[Figura 11 – Exemplo tag “link” 27](#_Toc169813697)

[Figura 12 – Exemplo CSS parte 1 28](#_Toc169813698)

[Figura 13 – Exemplo CSS parte 2 29](#_Toc169813699)

[Figura 14 – Exemplo CSS parte 3 30](#_Toc169813700)

[Figura 15 – Exemplo Formulário Estilizado 31](#_Toc169813701)

[Figura 16 – Exemplo de formulário com javascript 32](#_Toc169813702)

[Figura 17 – Exemplo alerta de teste 33](#_Toc169813703)

[Figura 18 – Exemplo javascript sucesso 34](#_Toc169813704)

[Figura 19 – Exemplo Javascript CANCELAR 34](#_Toc169813705)

[Figura 20 – Exemplo Tailwind Css 35](#_Toc169813706)

[Figura 21 – Tailwind formulário 35](#_Toc169813707)

[Figura 22 – Exemplo Vite 36](#_Toc169813708)

[Figura 23 – React Vite exemplo 37](#_Toc169813709)

[Figura 24 – Exemplo React 38](#_Toc169813710)

[Figura 25 – React Formulario 38](#_Toc169813711)

[Figura 26 – React Native Exemplo 40](#_Toc169813712)

[Figura 27 – React Native 2 41](#_Toc169813713)

[Figura 28 – Exemplo Expo 42](#_Toc169813714)

[Figura 29 – Exemplo Node 43](#_Toc169813715)

[Figura 30 – Exemplo teste de senha Python 45](#_Toc169813716)

[Figura 31 – Exemplo Python com importe 47](#_Toc169813717)

[Figura 32 – Exemplo de instalação de pacotes 48](#_Toc169813718)

[Figura 33 – Exemplo de emoji com Python 49](#_Toc169813719)

[Figura 34 – Exemplo Jupyter 49](#_Toc169813720)

[Figura 35 – Exemplo Pandas 51](#_Toc169813721)

[Figura 36 – Exemplo Sckit Learn 52](#_Toc169813722)

[Figura 37 – Exemplo Der 53](#_Toc169813723)

[Figura 38 – Exemplo Mer 54](#_Toc169813724)

[Figura 39 – Exemplo de banco de dados com o Firebase 55](#_Toc169813725)

**Lista de abreviaturas e siglas**

Cross-Origin Resource Sharing (CORS)

Cascading Style Sheets (CSS)

Diagrama Entidade Relacionamento (DER)

Document Object Model (DOM)

Chave Estrangeira (FK)

HyperText Markup Language (HTML)

Hypertext Transfer Protocol (HTTP)

JavaScript (JS)

JavaScript Object Notation (JSON)

Modelo Entidade Relacionamento (MER)

Não apenas SQL (NoSQL)

Node Package Manager (NPM)

Node Package Executor (NPX)

Pessoa Com Deficiência (PCD)

Chave Primaria (PK)

Sistema de Gerenciamento de Banco de Dados (SGBD)

Linguagem de Consulta Estruturada (SQL)

Unified Modeling Language (UML)

**SUMÁRIO**

[**1 INTRODUÇÃO 10**](#_Toc168865539)

[**2 REFERENCIAL TEÓRICO 12**](#_Toc168865540)

[2.1 Problemas encontrados no mercado de Trabalho 12](#_Toc168865541)

[**2.2 UML 13**](#_Toc168865542)

[**2.2.1 Caso De Uso 13**](#_Toc168865543)

[**2.2.2 Documentação de Caso de Uso 15**](#_Toc168865544)

[**2.2.3 Diagrama de Atividade 18**](#_Toc168865545)

[**2.2.4 Caso de Sequência 20**](#_Toc168865546)

[**2.2.5 Diagrama de Máquina de Estado 22**](#_Toc168865547)

[**2.3 Wireframe 23**](#_Toc168865548)

[**2.3.1 Design Centrado No Usuário (DCU) 25**](#_Toc168865549)

[**2.3.2 Experiencia Do Usuário (UX) 25**](#_Toc168865550)

[**2.4 HTML 26**](#_Toc168865551)

[**2.5 CSS 30**](#_Toc168865552)

[**2.6 JavaScript 35**](#_Toc168865553)

[**2.7 TailwindCSS 38**](#_Toc168865554)

[**2.8 React 40**](#_Toc168865555)

[**2.8.1 Vite 40**](#_Toc168865556)

[**2.8.2 Axios 44**](#_Toc168865557)

[**2.9 React Native 45**](#_Toc168865558)

[**2.10 Expo 47**](#_Toc168865559)

[**2.11 Node.JS 48**](#_Toc168865560)

[**2.11.1 Express 49**](#_Toc168865561)

[**2.11.2 CORS 50**](#_Toc168865562)

[**2.11.3 Node Package Manager 50**](#_Toc168865563)

[**2.11.4 Node Package Executor 50**](#_Toc168865564)

[**2.12 Python 51**](#_Toc168865565)

[**2.13 Jupyter 56**](#_Toc168865566)

[**2.18 Banco De Dados 60**](#_Toc168865567)

[**2.18.1 Diagrama Entidade Relacionamento 60**](#_Toc168865568)

[**2.18.2 Modelo Entidade Relacionamento 61**](#_Toc168865569)

[**2.18.3 Banco De Dados Não Relacional 61**](#_Toc168865570)

[**2.19 Firebase 62**](#_Toc168865571)

[**3 DESENVOLVIMENTO 63**](#_Toc168865572)

[REFERÊNCIAS 64](#_Toc168865573)

1 INTRODUÇÃO

A inclusão de Pessoas com Deficiência (PCD) no mercado de trabalho através de plataformas digitais de conectividade é um tema de grande relevância. Apesar do surgimento de ações sociais que visam a inclusão dos PCD à sociedade de forma geral, as cotas destinadas a essas pessoas não são preenchidas devido à falta de procura das empresas. (G1, 2019)

Desta forma pessoas com deficiência de 14 anos ou mais apresentam taxas de participação no mercado de trabalho (23,8%) e de formalização (34,3%) menores do que as pessoas sem deficiência, cujas taxas são de 66,3% e 50,9% (IBGE apud CNN Brasil, 2023).

O preconceito é um fator significativo que limita o acesso dos PCD ao mercado de trabalho. Estudos indicam que muitas empresas ainda não estão dispostas a contratar pessoas com deficiência, devido à falta de conhecimento sobre as capacidades desses profissionais (CNN Brasil, 2023).

A inclusão social por meio de ações de programas de qualificação profissional tem mostrado resultados positivos, com a origem da Lei n º 8.213, que define benefícios da Previdência social, mas ainda há um longo caminho (CNN Brasil, 2024).

Portanto, é essencial explorar soluções tecnológicas que facilite a integração entre empresas e PCD, aumentando as oportunidades de emprego e promovendo uma sociedade mais inclusiva. Pensando neste sentido surge a questão do porquê as cotas destinadas aos PCDs no mercado de trabalho não são preenchidas, e como uma plataforma digital baseada em redes sociais profissionais pode auxiliar na integração dessas pessoas às empresas? Sendo assim a hipótese formulada é que a utilização de um sistema digital pode aumentar a taxa de preenchimento das cotas passa as pessoas com deficiência nas corporações, ao facilita o processo de recrutamento e quebrar barreiras.

Para alcançar esse objetivo, foram estabelecidos através de pesquisas, revisões de literatura sobre a inclusão de pessoas no mercado de trabalho, juntamente com análises de plataformas digitais existentes que promovem a inclusão social e profissional de PCDs.

2 REFERENCIAL TEÓRICO

Esse capítulo possuí como meta, abstrair todas as etapas de fundamentação teórica para a construção do projeto junto da apresentação de conceitos e tecnologias. Visando demonstrar todo o embasamento teórico da plataforma DAELink.

2.1 Problemas encontrados no mercado de Trabalho

De acordo com G1 (2023) cerca de 18,6 milhões de brasileiros com dois anos ou mais possuem algum tipo de deficiência, sendo a inclusão um desafio que persiste no Brasil devido à falta de acessibilidade e apoio adequado. Dados do Instituto Brasileiro de Geografia e Estatística (IBGE, 2022) comprova que essas pessoas enfrentam maiores dificuldades para entrar no mercado de trabalho cerca de (28,3%) deficientes e (66,3%) sem deficiência.

O Brasil estabelece que empresas com cem empregados ou mais são obrigadas a preencher de 2% a 5% com pessoas com deficiência, conhecida como Lei de Cotas, conforme o artigo 1º da Lei n º 8.213, de 24 de julho de 1991:

Art. 1º A Previdência Social, mediante contribuição, tem por fim assegurar aos seus beneficiários meios indispensáveis de manutenção, por motivo de incapacidade, desemprego involuntário, idade avançada, tempo de serviço, encargos familiares e prisão ou morte daqueles de quem dependiam economicamente. (BRASIL, 1991.).

Apesar de estar em vigor há quase trinta anos, segundo UNICAMP (2020) ainda não é plenamente cumprida, ainda existe desafios para que a inclusão aconteça devido a pouca especificidade da legislação e a qualificação.

Para G1 (2017) a tecnologia está cada vez mais presente no cotidiano, muitas empresas não atendem a essa necessidade. Isso ocorre devido à falta de um sistema acessíveis, gerando uma lacuna na conexão de pessoas com deficiência. Portanto, há uma necessidade de criação de um sistema que facilite essa integração digital. Uma solução de uma plataforma digital de conectividade pode promover a inclusão social e melhorar o acesso dessas pessoas ao mercado de trabalho.

2.2 UML

O Unified Modeling Language é uma representação visual utilizada para auxiliar a compreensão do sistema em sua parte física e lógica, sendo um padrão internacional aceito para a documentação de um software (GUEDES, 2018).

Utilizado para a realização de um projeto completo, e preciso que possa ser alterado posteriormente e permitindo um melhor entendimento entre os clientes e os desenvolvedores (PEREIRA, 2011).

Em termos gerais, através dos diagramas que constituem a UML, o projeto todo é abordado de diferentes formas técnicas para obter um melhor resultado em sua conclusão (GUEDES, 2018).

2.2.1 Caso De Uso

O caso de uso é o primeiro e mais simples diagrama de um sistema, com intuito de demonstrar todas as suas funcionalidades de forma compreensível entre profissionais e leigos (GUEDES, 2018).

Sendo considerado o diagrama mais importante que um sistema possuí, por ser a base dos demais diagramas e servindo como base para o software com a aplicabilidade da análise de sistemas (PRESSMAN; MAXIM, 2021).

Na Figura 1, pode ser vista um exemplo simples de um diagrama de caso de uso.

Figura 1 – Exemplo de Diagrama de Caso de Uso

Diagrama

Descrição gerada automaticamente

Fonte: Guedes, 2018.

O exemplo abordado acima retrata o diagrama de caso de uso de um sistema veterinário que tem como principais elementos:

* Atores: São representados por um boneco palito que são pessoas ou sistemas externos que interagem diretamente com o sistema, no exemplo foi utilizado os atores Secretária, Veterinário e Cliente.
* Levantamento de Requisitos: São as principais funcionalidades que o sistema precisa ter, sendo dividido entre Requisito Funcional, Requisito não Funcional e as Regras de Negócio.
* Caso de Uso: São os reflexos dos requisitos funcionais que são representados por uma elipse com verbo no infinitivo que no exemplo acima foi utilizado “Gerencias Veterinários” e “Marcar Consulta”.
* Associação: Serve para demonstrar a interação entre Ator e Caso de Uso, é representado por uma reta.
* Generalização: Conhecido como Herança, demonstra o processo das heranças que pode ser utilizado entre Ator para Ator ou Caso de uso para Caso de uso, representado por uma reta com uma seta.
* Inclusão: Representado por uma reta pontilhada com “<<Inclide>>” em cima, representa quando uma ação do sistema é considerada obrigatório.
* Extensão: Representado por uma reta pontilhada com “<<Extend>>” em cima, pode ser utilizado para definir validações, Consequências ou Decisões podendo ser opcional.

2.2.2 Documentação de Caso de Uso

A documentação de um caso de uso tem como função descrever, por meio de informações em linhas gerais do caso de uso, como os atores interagem com ele e suas etapas para serem executadas pelo ator e sistema (GUEDES,2011).

Desta maneira não se existe um formato específico de documentação para os casos de uso definido pela UML, porém há formatos correspondentes a ela, deste modo o caso de uso é flexível para realização de sua documentação (GUEDES,2018).

Na figura 2 demonstra um exemplo de documentação de um caso de uso chamado Encerrar conta onde fornecerá características.

Figura 2 – Exemplo Documentação Caso de Uso

Tabela

Descrição gerada automaticamente

Fonte: Guedes, 2018.

* Nome do Caso de Uso: Identifica claramente o objetivo do caso de uso no exemplo "Encerrar Conta".
* Ator Principal: Indica quem inicia o processo “Cliente”.
* Atores Secundários: Mostrar quem auxilia no processo “Funcionário”.
* Resumo: Tem como objetivo descrever brevemente o objetivo do caso de uso.

Exemplo: Este caso de uso descreve as etapas necessárias para que um cliente encerre uma conta.

* Pré-Condições: Especifica as condições que devem ser verdadeiras antes de iniciar o caso de uso “É necessário existir uma conta ativa”.
* Pós-Condições: Descreve o estado esperado após a conclusão do caso de uso no exemplo “A conta será encerrada”.
* Fluxo Principal: Sua função é detalhar os passos padrão que descrevem como o caso de uso é executado normalmente.

Exemplo:

Cliente solicita encerramento da conta fornecendo o número da conta.

Sistema emite saldo.

Sistema encerra a conta.

* Restrições/Validações: Defini regras que limitam como o caso de uso pode ser executado.

“A conta só pode ser encerrada pelo seu titular e o saldo deve estar zerado”.

* Fluxos Alternativos: Demonstra variações do fluxo principal para diferentes cenários.

Exemplo:

* Saldo Positivo: Cliente deve realizar saque antes do encerramento.
* Saldo Negativo: Cliente deve fazer depósito para zerar o saldo.
* Manutenção do Cadastro do Cliente: Se for a única conta do cliente, o cadastro deve ser atualizado para inativo.

2.2.3 Diagrama de Atividade

O Diagrama de atividade dentre todos da UML, é utilizado para a representação da lógica seguida para a realização de um procedimento, permitindo processos em separado demonstrado todo fluxo de trabalho (FOWLER; TORTELLO, 2005)

Na figura 3, possuí um exemplo de um diagrama de atividade e explicação de seus elementos.

Figura 3 – Exemplo Diagrama de Atividade

Diagrama

Descrição gerada automaticamente

Fonte: FOWLER; TORTELLO, 2005.

Neste exemplo, é exemplificado a atividade “Realizar Pedido” que é constituída pela separação de processos após o nó de ação “Receber Pedido”.

Depois processos em paralelos são executados em ordem aleatória até a junção respectiva no final.

Na 1 parte, ocorre o “Preenchimento do Pedido” e uma decisão simples de ordem de prioridade sendo o tipo da entrega selecionada, que possuí uma intercalação e após isso se junta ao outro processo.

Enquanto na 2 parte, ocorre apenas o “Envio da Fatura” e o “Receber Pagamento” antes de se juntarem em um único fluxo.

Ao final do diagrama a atividade se termina, no “Nó de atividade Final”.

Os elementos utilizados comummente no diagrama de atividade são:

* Atividade: É a denominação geral dos processos a serem descritos, escrito de forma sucinta.
* Nó Inicial: Utilizado para demarcar o começo da atividade, e seu fluxo de processos.
* Nó de Atividade Final: Utilizado para demarcar o fim dos processos da atividade.
* Nó de Ação: Serve para a representação de uma etapa, consistindo em apenas um processo unitário.
* Fluxo de Controle: São os elementos visuais que mostram o caminho a ser seguido após determinado processo.
* Nós de Decisão: Representa a decisão entre dois ou mais processos, em que certa condição impacta no que deve ser executado.
* Intercalação: Utilizado para interconectar fluxos após um nó de decisão, que possuem a mesma continuação
* Separação ou Nó de Bifurcação: Utilizado para representar a separação de um Nó de Ação, em vários de forma paralela.
* Nó de Junção: Complemento do Nó de Bifurcação, por determinar o ponto de conexão entre os processos que antes estavam paralelos prosseguindo para um mesmo fluxo.

2.2.4 Caso de Sequência

Segundo Guedes (2011), um diagrama de sequência é um diagrama comportamental que ilustra processos, demonstrando a ordem em que eventos ocorrem com base em caso de uso específico.

Após a identificação dos eventos pelo caso de uso, o modelador cria um diagrama de sequência, que representará como esses eventos provocam o fluxo de um objeto para o outro ao longo dos processos. (PRESSMAN; MAXIM, 2021).

Tratando-se de um processo iniciado por um ator, geralmente dependendo do diagrama de classe, devido os objetos descritos no diagrama de sequência. Sendo uma abordagem excelente para completar o diagrama de classes. (GUEDES,2018).

A seguir na figura 4 será demonstrado um exemplo de caso de sequência mostrando um processo de emissão de saldo do sistema de controle bancário.

Figura 4 – Exemplo Diagrama de Sequência

Diagrama

Descrição gerada automaticamente

Fonte: Guedes, 2018.

Nesse exemplo são utilizados alguns conceitos para a composição do caso de sequência sendo eles:

* **Atores:** representam entidades externas que interagem com o sistema e solicitam serviços, iniciando o processo, eles são representados por um boneco magro semelhante com os utilizados em diagrama de casos de uso.
* **Lifelines:** descrevem objetos ou entidades que participam da interação. Geralmente instancias de classes, com uma linha tracejada vertical indicando a duração de sua participação na interação, mostrando na visualização do fluxo dos eventos e ações.
* **Classes de Boundary e de Controle:** As lifelines podem representar instâncias de diferentes tipos de classes, como boundary (responsáveis pela interface com o usuário) e controle (responsáveis pela lógica de controle do sistema). Isso é indicado pelos estereótipos <<boundary>> e <<control>>.
* **Métodos chamados e retornos:** As mensagens trocadas entre as lifelines representam chamadas de métodos e retornos de valores. Por exemplo, o método consultarConta é chamado na lifeline da classe ControleEmitirSaldo e, se bem-sucedido, retorna um valor verdadeiro.
* **Fluxo de interação:** O diagrama de sequência detalha o fluxo de interação entre os participantes, mostrando a sequência de ações realizadas durante o processo de emissão de saldo em um sistema bancário.
* **Retorno de métodos:** Os retornos dos métodos podem ser valores específicos, como verdadeiro ou um saldo bancário representado por um valor numérico. Esses retornos são representados nas mensagens trocadas entre as lifelines.
* **Estereótipos:** Além dos estereótipos padrão para as classes (<<boundary>>, <<control>>, <<entity>>), o diagrama também pode usar estereótipos específicos para indicar o tipo de lifeline, como <<entity>> para lifelines de classes de entidade.

2.2.5 Diagrama de Máquina de Estado

O Diagrama de máquina de estado foi criado para representar o comportamento demonstrando o funcionamento as funções dos elementos composto em um sistema, demonstrando suas funcionalidades individuais e coletiva (GUEDES, 2018).

Na figura 5 abaixo, é demonstrado a representação de um diagrama de máquina de estado de “Realizar Deposito” com seus devidos processos:

Figura 5 – Exemplo de Diagrama de Máquina de Estado

Diagrama

Descrição gerada automaticamente

Fonte: Guedes, 2018.

Este diagrama é composto por diversos elementos gráficos em sua composição, e os principais dele são:

* Estado: Tem como função representar a situação de um elemento de um sistema, demonstrando os possíveis períodos e processos aplicados sobre ele.
* Transições: Representado por uma seta ligando dois estados, ele demonstra um evento e a mudança de estado de um determinado objeto.
* Estado Inicial: Determina onde que a função do sistema se inicia, representado por um círculo completamente preenchido.
* Estado Final: tem o mesmo objetivo do Estado inicial com a particularidade de determinar o fim do sistema, sendo representado por um círculo parcialmente preenchido.
* Atividades Internas: São atividades que determinam o tempo em que o Estado irá fazer a sua tarefa, sendo dividido por:
  + Entry: Executa a ação no primeiro momento em contato com o estado.
  + Exit: Executa a ação no final de contato com o estado.
  + Do: Executa a ação em todo o momento.
* Pseudoestado de Escolha: É utilizado como método de validação, condição ou escolha sobre possíveis ações que um estado pode fazer, representado por um losango.
* Barra de Bifurcação/União: Representa quando uma ação dentro de um sistema ocorre de forma paralela, é utilizado uma barra preta para a sua representação visual.

2.3 Wireframe

O wireframe é um esboço simplificado de estrutura de uma interface que ilustra as funções do produto. Isso permite estimar o esforço necessário de dados até a modelagem de teste de front-end conforme a proposta (TEIXEIRA, 2014). Para Lucidchart o termo “wireframe” descreve uma representação visual de desenvolvimento de páginas *web* e aplicativos moveis, sendo utilizados no início da criação de estrutura básica de uma tela antes de acrescentar o design final.

Desta maneira os wireframes são definidos em baixa fidelidade que se refere ao rascunho do produto, sendo simples e com poucos detalhes, utilizado para explicar a ideia inicial para seu time, clientes. (MIRO).

Já os de Alta fidelidade oferece uma representação detalhada do produto, incluindo cores e elementos gráficos atraentes, demonstrando uma visão mais clara e precisa de como a interface final ficará demonstrando possíveis funções. (FABRICIO, 2014).

Na figura 6 a seguir será demonstrado um exemplo de wireframes de baixa fidelidade onde demonstra um protótipo de site e seus componentes que irá constitui-lo, mostrando uma pré-visualização de uma página *web*.

Figura 6 – Exemplo Wireframde de Baixa Fidelidade

Forma

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Na figura a seguir demonstrará o wireframe de alta fidelidade do exemplo anterior com componentes mais detalhados, mostrado uma evolução do design anterior representando uma tela mais próxima do produto.

Figura 7 – Exemplo Wireframe Baixa 2

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.3.1 Design Centrado No Usuário (DCU)

O design Centrado no Usuário surgiu da Interação humano-computador (IHC) e consiste em uma metodologia de design de software para desenvolvedores ajudando a criação de aplicativos que atendam às necessidades de seus usuários. (LOWDERMILK, 2013, p.26)

2.3.2 Experiencia Do Usuário (UX)

De acordo tem como função ser uma disciplina do design focada na experiência emocional e sensorial dos usuários ao interagir com produtos, colocando o Usuário como centro no processo de criação, evitando decisões baseadas em suposições.

Focando a Experiência do usuário (UX), envolve a análise de como um usuário navega por um produto ou serviço, ilustrando os caminhos percorridos e as interações realizadas pelos usuários enquanto tentam completar uma tarefa. (MIRO).

2.4 Html

A linguagem de marcação de hipertexto conhecida como *HTML*, sendo necessária para qualquer documento utilizado na *web* no quesito de sites utilizado para estruturar todo o conteúdo de uma página (SILVA, 2015).

Baseado em demarcações denominadas “*tags*”, que podem se referir a qualquer elemento presente na página, sendo interpretados pelo *browser,* mostrando sua representação visual (DUCKETT, 2016).

Figura 8 – Exemplo Html

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Nesta figura 8, demonstra a estrutura básica para a criação de uma página *web* simples, e as principais *tags* que compõem essa estrutura são:

* “DOCTYPE *Html*” :O *Document Type Declaration* é utilizado para informar a versão do *HTML* que, consequentemente, ajuda o navegador a renderizar as informações.
* *<html>*:Esta *tag* é utilizada para definir o esqueleto do site, englobando todos os conteúdos de uma página na internet.
* <head>: É usado para a inclusão de recursos externos ao principal, como por exemplo *links*.
* <title>: Utilizado para definir o título da página, exibido na aba do navegador.
* <body>: A parte principal de um site, entre essas *tag* ficam os conteúdos carregados pelo navegador.
* <atributos>: São utilizados para definição de um comportamento na estrutura da página, como exemplo o atributo “*Class*” utilizado para definir um conjunto de elementos pertencentes a uma classe específica.

A imagem a seguir, demonstra o processo da criação de um formulário simples:

Figura 9 – Exemplo HTML 3

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Para a criação de um formulário, foi utilizado algumas tags de estruturação, sendo elas:

* <h2> Utilizado para definir o tamanho de um texto em uma página *web*, podendo ir de h1 que é o maior, até h6 que é o menor.
* <form> É uma *tag* utilizada para a criação de formulários com intuito de extrair dados, como exemplo o cadastro de uma pessoa, onde é extraído nome, *email*, idade e senha.
* *<label>* É utilizada para definir um rótulo descritivo a um elemento de uma página *web.*
* *<Input>* É a *tag* principal para a utilização de formulários, ela é a caixa de texto que fica dentro dos formulários.
* <button> Utilizado para executar alguma ação dentro do sistema, como visto na imagem abaixo, a função dele vai ser “Cadastrar”.

O resultado desse código pode ser visto na figura 10 a seguir:

Figura 10 – Exemplo Formulario

A screenshot of a computer

Description automatically generated

Fonte: Autoria Própria, 2024.

2.5 CSS

A folha de estilo em cascata, o CSS é uma linguagem concebida para a estilização de páginas *web* mediante atributos pré-estabelecidos que se relacionam às características visuais (SCHEIDT, 2015).

Em que através das demarcações feitas pelo HTML, essas alterações estilísticas podem ocorrer, o restringindo apenas as marcações. (SILVA, 2008).

Figura 11 – Exemplo tag “link”

**Texto

Descrição gerada automaticamente**Fonte: Autoria Própria, 2024.

Neste exemplo, é demonstrado como adicionar um código CSS externo ao arquivo Html.

* *<link> Tag* utilizada para vincular propriedades de outro arquivo, através do “
* <href> (que é seguido do local armazenado do arquivo) e o *rel* para se definir o tipo de relacionamento.

Figura 12 – Exemplo CSS parte 1

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Na figura 12, através da anexação de atributos às classes ou via *id* se restringindo a uma unidade da respectiva classe *(“#formstyle*”). É feita a adição de atributos que afetam toda a página (*html, body, \**), ou elementos específicos trocando fonte, cores, posicionamento horizontal e vertical e de outros detalhes gráficos do formulário.

Figura 13 – Exemplo CSS parte 2

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Continuando no código mudamos através das classes (‘.’ Previamente da classe), ou acessando diretamente as tags a mudando em toda a página.

Figura 14 – Exemplo CSS parte 3

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Ao final do código *CSS*, todos os componentes passam pelo processo de indexação de atributos. Assim realizando a estilização do formulário e dos outros elementos que compõe uma página *web*.

Figura 15 – Exemplo Formulário Estilizado

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente com confiança média

Fonte: Autoria Própria, 2024.

2.6 JavaScript

Originado pela Netscape, e sendo documentado pela Sun Microsystems (Oracle), o Javascript é popularmente conhecido como uma linguagem de programação da *web* tornando-se específica para comportamentos do *HTML* e *CSS* (FLANAGAN ,2012).

Para Groner (2019) a linguagem Javascript é extremamente eficaz, se tornando uma das linguagens mais populares mundialmente, passando a ser usada tanto para *front-end* quanto *back-end* resultando em uma linguagem multifuncional.

Na Figura 16 veremos o funcionamento do Javascript dentro do formulário mostrado anteriormente, exemplificando o envio de componentes digitados dentro das caixas de textos.

Figura 16 – Exemplo de formulário com javascript

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Depois do </form> deve-se implementar, a *tag* <Script> possui como função incorporar ou referenciar código Javascript em um documento HTML.

* *Document*: Utilizado como forma de representação do documento HTML carregado no navegador.
* *GetElementById*: É reconhecido como um método do objeto ‘*document*’ que retorna o elemento que possui um atributo ID, no caso ‘*myForm*’.
* *Onsubmit*: É um evento que ocorre quando um formulário é enviado. Acompanhado com uma função a este evento permitindo que seja executado o código quando o usuário tentar enviar.
* *Function(event):* Possui como função sempre ser chamada quando se utilizar o evento ‘*onsubmit*’. Ela recebe um argumento ‘Event’, que é um objeto contendo detalhes sobre o evento de submissão.
* *Confirm* (‘quer testar o formulário? Se sim, clique em ok!’): ’*confirm*’ é uma função que exibe uma caixa de diálogo com uma mensagem e dois botões: OK e CANCELAR. Retorna ‘*true*’ se o usuário clicar em ’OK’ e ‘*FALSE*’ se clicar em cancelar. Isso é usado para confirmar como o usuário proceder com o envio do formulário.
* Alert: possui como objetivo exibir uma caixa de diálogo com uma mensagem para usuário. Neste caso, é usado para informar que o formulário foi enviado com sucesso, já o outro exemplo informa ao usuário que o envio foi formulário foi cancelado.

Contudo, ao enviar o resultado, o *alert* deverá ser apresentado demonstrando as duas opções, testar o formulário ou cancelar.

Figura 17 – Exemplo alerta de teste

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Na figura 18 a seguir, aparecerá o resultado se for escolhida a opção OK, demonstrando o sucesso do envio do formulário.

Figura 18 – Exemplo javascript sucesso

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Caso seja escolhida a opção CANCELAR, deverá aparecer o envio cancelado do formulário.

Figura 19 – Exemplo Javascript CANCELAR

Forma, Retângulo

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.7 TailwindCSS

O framework de estilo CSS código aberto, cujo é denominado Tailwind tem o intuito de oferecer uma abordagem única para projetos *web*, dentro do código HTML, gerando estilo eficaz sem a precisão de escrever comando CSS. (CLASEN, 2023).

O tailwind tem como objetivo eliminar a necessidade do uso de CSS personalizado, através de biblioteca de classes pré-definidas para estilos de elementos específicos, como cores, tipografia e muitos outros. (MATOS, 2023; KLIMM, 2021).

Na figura 20, será demonstrado a realização de um código utilizando tailwind CSS em um formulário semelhante ao mostrado anteriormente.

Figura 20 – Exemplo Tailwind Css

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Em seguida, será apresentado o formulário no aspecto visual, mostrando o resultado após a aplicação das estilizações descritas.

Figura 21 – Tailwind formulário

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.8 React

O React é uma biblioteca JavaScript utilizada para a criação de interfaces de forma particionada, que podem ser combinadas em componentes. Abrangendo desde sites a aplicativos móveis (REACT).

Concebida com o intuito de simplificar e agilizar a criação de interfaces, criada por Jordan Walke, um engenheiro do Facebook, em 2011. Se tornando atualmente a biblioteca mais popular de JavaScript (SILVA, 2021).

2.8.1 Vite

Vite vem do francês que significa “Rápido”, demonstrado em sua proposta de ser uma ferramenta que permite a criação de projetos *front-end* de forma acessível, sendo leve e prática, trazendo conceitos criativos para as páginas *web* (REACT VITE).

Para Schmitz e Georgii (2015), o React utiliza a implementação de camadas para a visualização de uma página *web,* utilizando como base componentes para a sua criação.

Utilizando-se Document Object Model (DOM), garantindo um desempenho superior a outras bibliotecas do gênero. E a criação de uma página nesta linguagem pode ser vista na imagem a seguir (REACT VITE).

Figura 22 – Exemplo Vite

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

E o resultado da criação de uma tela básica feita em React pode ser observado na figura 23 a seguir.

Figura 23 – React Vite exemplo

A screenshot of a computer

Description automatically generated

Fonte: Autoria Própria, 2024.

Utilizando desses conceitos, por utilizar como base o *HTML* para criação de páginas, para fazer um formulário não se torna muito distinto, como visto na imagem a seguir.

Figura 24 – Exemplo React

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

A principal diferença da criação de uma página *HTML* para uma React são os atributos como o “*Class*” que fica “ClassName” no React.

Com base na imagem anterior, foi realizada a criação de um formulário simples que pode ser visto na figura 25 abaixo.

Figura 25 – React Formulario

A black square with white text and red letters

Description automatically generated

Fonte: Autoria Própria, 2024.

2.8.2 Axios

O Axios é um framework de NodeJs e de linguagens de front-end como o React, sua principal função é de gerenciar as requisições com o protocolo Hypertext Transfer Protocol (HTTP) dentro de um ambiente *web* (AXIOS, 2024).

2.9 React Native

De acordo com Escudelario e Pinho (2020), o React Native é uma plataforma baseada em React, viabilizando a criação de aplicativos híbridos, executados no iOS (Apple), quanto no Android, sendo criada pelo Facebook em 2013.

O React Native pode ser definido como um framework de código aberto que visa a criação de aplicações nativas, ou seja, não há uma camada *web* como interface, mas sim o próprio aplicativo nativo. (LEITÃO, 2019, apud GRANDE; TANAKA, 2023).

Baseado na criação de sua estrutura em componentes, o React Native utiliza dos mesmos conceitos de criação de tela do React, podendo ser visto na figura 26 abaixo.

Figura 26 – React Native Exemplo

A screen shot of a computer program

Description automatically generated

Fonte: Autoria Própria, 2024.

Os principais componentes utilizados na criação dessa tela são:

* *Import*: utilizado para a importação de bibliotecas do React para facilitar a criação de telas com componentes prontos.
* *Export* *default function*: É a função principal que envia os componentes para a tela principal para ser interpretada.
* *View style*= {}: É um componente utilizado para representar a tela da aplicação, e o *style* é a estilização desse componente.
* *Text*: É basicamente um componente para a criação de textos na tela.
* *ImageBackground*: É uma imagem utilizada como plano de fundo do aplicativo.
* *Source*: Local onde a imagem que vai ser utilizada como plano de fundo.
* *Button*: Serve para a criação de um botão na aplicação.
* *Title:* Representa o nome que esse botão vai ter.
* *OnPress:* Representa qual vai ser a função que o botão vai executar quando for clicado.

Figura 27 – React Native 2

A screen shot of a computer program

Description automatically generated

Fonte: Autoria Própria, 2024.

Na figura 27 utiliza os mesmos conceitos de estilização abordados no *CSS*, que a principal diferença é a sua estruturação que utiliza uma constante para definir o estilo de aplicativo.

2.10 Expo

O Expo assim como o pacote do react “*create-react-app*” fornece a criação de toda a estrutura necessária para desenvolver uma aplicação, fornecendo um ambiente para a criação facilitada de aplicativos (ESCUDELARIO; PINHO, 2020).

Sendo assim o Expo é uma ferramenta utilizada na criação mobile com React Native que permite facilitar o acesso a algumas API’s nativas do dispositivo sem precisar da instalação de qualquer dependência ou alterar código nativo (ROCKETSEAT,2020).

Na Figura 28 será demonstrado um formulário dentro do react-native com expo nas suas dependências.

Figura 28 – Exemplo Expo

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.11 Node.JS

Node.js é um ambiente JavaScript que permite a execução dos comandos de forma externa, exercendo um papel de *back-en*d aos moldes *non-blocking* (não bloqueante) realizando múltiplas execuções ao mesmo tempo, evitando atraso (PEREIRA, 2013).

Possuindo também como qualidades para sua atuação, junto de ser compatível com diversas plataformas, e sendo impulsionado pelas bibliotecas feitas pelos usuários, tornando-se uma ferramenta prática e acessível. (DUARTE JÚNIOR, 2022).

Figura 29 – Exemplo Node

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

No exemplo acima pode ser visto as funcionalidades básicas do node de criar um servidor utilizando o Express em que os principais comandos são:

* *App*: Tem como intuito carregar todas as funcionalidades do Express dentro de si.
* Métodos: Dentro do node existem métodos para diversas funções em uma página web, no exemplo foi utilizado o *Get*.
* Req, Res: O *Request* e o Respons são responsáveis pelo envio de informações entre páginas.
* /Tela: É o local onde as ações dentro da função serão executadas.
* /TelaParametros/:nome: Tem a mesma funcionalidade do tópico anterior com diferença que para ser executado precisa de uma informação que no exemplo acima foi o “Nome”.

2.11.1 Express

O Express é um framework de NodeJs que tem como principal função facilitar o desenvolvimento de aplicações de forma prática, utilizando de middleware e roteamento, renderizado em telas HTML dinâmicas (BARSOTI; GIBERTONI, 2020).

2.11.2 CORS

O Cross-Origin Resource Sharing é responsável pelo compartilhamento de informações em uma página web utilizando do HTTP para a sua distribuição, sendo utilizado como framework de NodeJs (MOZILLA, 2024).

2.11.3 Node Package Manager

NPM é o gerenciador de pacotes do Node, que permite a instalação de diversos módulos que abrangem diversos aspectos como *back-end* e pagamento online, sendo eles realizados e disponibilizados pela comunidade (POWERS, 2019).

2.11.4 Node Package Executor

O Npx é um comando incorporado ao npm, realizando a execução de módulos sem instalá-los, servindo para requisições rápidas, através da remoção automática desses módulos após seu uso, economizando memória. (MORAES, 2021).

2.12 Python

Segundo Matthers (2016) Python é uma linguagem de programação extremamente eficientes pelo fato de os programas conterem menos linhas de códigos, ajudando a construção de um código “limpo”, obtendo uma compreensão e depuração rápida.

Para Menezes (2014) Python vem crescendo em várias áreas da computação, sendo elas bancos de dados, biotecnologia e jogos. Tornado assim uma linguagem de fácil aprendizagem e obtenção de resultados em pouco tempo através de suas bibliotecas.

Para Menezes (2019) a versatilidade, a simplicidade fazem do Python uma das linguagens de programação mais populares do mundo., o que a torna uma escolha valiosa para qualquer programador

Conforme Matthers (2016) a linguagem Python possui um interpretador que executa em uma janela de terminal, o que permite a possibilidade de teste de códigos sem a necessidade de salvar e executar um programa completo.

Na figura 30 será demonstrado um modelo de código exemplificando as funcionalidades da linguagem Python onde poderá ser inserido uma senha e o programa verificará se a senha é segura ou não.

Figura 30 – Exemplo teste de senha Python

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

* Função: refere-se a um bloco de código que realiza uma tarefa específica, sendo definida com o uso da palavra *def*, tendo então como objetivo organizar e reutilizar um código, promovendo uma eficiência.
* def: possui como função iniciar a definição de uma função chamada ‘verificar\_senha’, que recebe como parâmetro(referência) ‘senha’.
* Len(senha): é uma função que retornará o número de caracteres que existem na senha fornecida. A expressão ‘>= 8’ compara esse número para ver se a senha tem pelo menos 8 caracteres.
* *Any*(char,isdigit() for char in senha): verifica se há pelo menos um digito na senha. O método ‘char,isdigit()’ avaliará se cada caractere na senha é um número. ‘*any*’ é uma função que retorna *‘true’* se qualquer caractere for um digito, caso contrário retorna ‘false’.
* *Any*(char.isupper() for char in senha): Possui uma similaridade anterior, esta linha usa ’*any*’ para determinar se há alguma letra maiúscula na senha. ‘char.isupper() ’é um método que verifica se o caractere ‘char’ é uma letra maiúscula.
* Estrutura Condicional *(If):* é usado para executar uma rotina apenas se uma condição específica for verdadeira.
* *Return* “Senha segura!”: Se a condição do ‘*if*’ for verdadeira, a função retorna a *String* “Senha segura!”.
* *Input* (“Digite sua senha para verificação:): Esta função incorporada exibe uma mensagem e espera que o usuário insira uma *string* através do teclado. Sendo assim a *string* inserida é então atribuída a variável ‘senha’.
* *Print* (resultado): é usado para exibir o valor de ‘resultado’, que informará ao usuário sobre a segurança de sua senha.

Para McKinney, (2018) O python muitas vezes poderá ser instalado pacotes adicionais, não incluídos na distribuição do Anaconda, sendo um deles o pip que é um programa que gerencia e cuida da instalação de pacotes Python.

Nesse sentido existem dois tipos: o conda e pip ambos com propósitos diferentes, o conda oferece um gerenciamento de pacotes geral para uma grande variedade de linguagens no ambiente conda e pip oferece serviços especificamente para Python. (MUELLER, 2020).

Na figura a seguir demonstrará um exemplo de código que usará um pacote que não estão dentro do python então será realizado a instalação

Figura 31 – Exemplo Python com importe

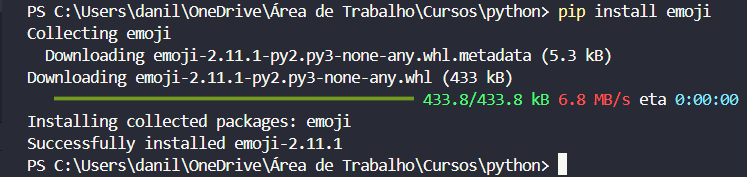
Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

* Import random” Esta linha importa o módulo ‘random’ do Python, usados para gerar números aleatórios. Esse módulo é utilizado no código para selecionar aleatoriamente uma resposta da lista de possíveis respostas.”
* From emoji import emojize” Esta linha importa a função ‘emojize’ do pacote ‘emoji’, que converte strings contendo códigos de texto de emoji em emojis visuais.”
* Class DecidaPorMim” que permite a criação de objetos em Python com atributos (dados) e métodos (funções) associados.”
* def\_\_init\_\_(Self): Este método especial é o construtor da classe, chamado quando uma nova instância da classe é criada, inicializando os atributos do objeto.”
* self.resposta=[...]: ” Dentro do construtor o atributo ‘resposta’ é definido é definido como uma lista de strings, cada uma podendo incluir códigos de emoli. “
* Decida = DecidaPorMim(): “cria uma instância da classe DecidaPormim. O construtor init é chamado para iniciar o novo objeto decida.”
* Decida.iniciar(): “é chamado, iniciando o processo em que o usuário é solicitado a fazer uma pergunta e, em seguida, uma resposta aleatória é exibida.”

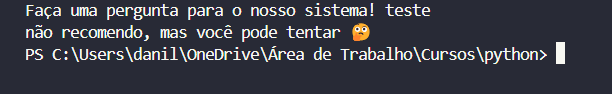
Figura 32 – Exemplo de instalação de pacotes



Fonte: Autoria Própria, 2024.

Na figura 32 é usado o pip install emoji para a instalação dos pacotes de emojis, devido não serem incluídos dentro do Python.

Figura 33 – Exemplo de emoji com Python

 Fonte: Autoria Própria, 2024.

A figura 33 acima, demonstra a pergunta realizada pelo sistema com a resposta do usuário, gerando assim a função do sistema, pegando uma frase dentro da lista e respondendo junto com o emoji.

2.13 Jupyter

Um ambiente de desenvolvimento de manipulação dados, o Jupyter Notebook é utilizado principalmente por conta da sua simplicidade e facilidade de uso e pelo suporte a linguagens de marcação e códigos (VASCONCELOS; GUERRA, 2023).

Figura 34 – Exemplo Jupyter

A screenshot of a computer

Description automatically generated

Fonte: Autoria Própria, 2024.

**2.14 Machine Learning**

O Machine Learning utiliza-se da filtragem de dados para a criação de novas informações, gerando resultados significativos, possibilitando a tomada de decisões inteligente através dos dados gerados (KNEUSEL, 2024).

A tecnologia está em constante evolução, e o aprendizado de máquina se torna crucial para o avanço de diversas áreas comerciais, sendo adotada pelas maiores empresas da atualidade como a Netflix (DOMINGOS, 2017).

**2.15 Numpy**

O NumPy é uma biblioteca da linguagem Python que visa primordialmente a possibilidade da utilização de computação numérica em projetos, criado em 2005 como uma biblioteca de código aberto (NUMPY).

Segundo Pereira (2020), O NumPy possui funções fundamentais para o gerenciamento de dados pelo fato de sua manipulação de matrizes e vetores com funções matemáticas de alto nível.

**2.16 Pandas**

Sendo utilizado como ferramenta para análise de dados, a biblioteca Pandas tem como função primordial o acesso à manipulação de dados utilizando de planilhas ou dados da internet (CHEN, 2018).

Para McKinney (2018), o Pandas obtém uma série de funções para a manipulação de dados sendo essencial para criações de análise de dados e machine learning.

E as principais funcionalidades do Pandas podem ser vistas na imagem a seguir:

Figura 35 – Exemplo Pandas

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Para a criação do exemplo acima foi utilizado o comando “DataFrame”, mas existem diversos comandos para analisar base de dados diferentes como:

* Read\_csv: Lê dados de determinado arquivo ou URL para manipular dados.
* Read\_table: Visualiza dados de determinada tabela de um arquivo ou URL.
* Read\_fwf: Lê arquivos de uma coluna especifica com tamanho fixo.
* Read\_clipboard: Utilizado para conversão de tabelas para páginas web, tendo muita similaridade com o “read\_table”.
* Read\_excel: Analisa dados de um arquivo Excel com extensões de XLS ou XLSX
* Read\_html: Lê dados de uma tabela em um documento HTML.
* Read\_json: Capta informações em um JavaScript Object Notation (JSON).
* Read\_sql: Lê resultados de uma inquisição SQL.

**2.17 Scikit-Learn**

Utilizado como biblioteca de linguagem de máquina, o Scikit-Learn é um software de código livre que tem como intuito a análise de dados com base em algoritmos de classificação, agrupamento e entre outros (INHESTA, 2022).

E com a utilização de uma base de dados simples, pode ser feito a análise como apresentado na figura abaixo.

Figura 36 – Exemplo Sckit Learn

A screenshot of a computer

Description automatically generated

Fonte: Autoria Própria, 2024.

As três linhas de código demonstram ao Scikit-Learn analisando uma base de dados de filmes em que:

* Pd.Dataframe: é o comando onde prepara a base de dados para fazer a análise de recomendação
* Rec\_df[‘x-man’]: analisa o filme “x-men” na base de dados com base na avaliação da base de dados
* sort\_values(ascending=False): Ordena os valores de uma coluna e com o atributo “ascending”, colocando os de ordem decrescente.

2.18 Banco De Dados

Para Cayres (2015) Banco de dados são coleções de dados que podem ser inseridos, atualizados e recuperados, sendo visto como uma representação de fatos, conceitos de maneira normalizada. Essa afirmação ressalta a importância do banco de dados, esclarecendo a possibilidade de armazenamento de informações tanto em sistemas simples quanto em sistemas complexos (ELMASRI; NAVATHE, 2011).

Segundo Caiut (2015) o Sistema de Gerenciamento de Banco de Dados (SGBD), que consiste na finalidade de gerenciar os Bancos de dados, facilitando assim o processo de definição, construção e manejo de bancos de Dados armazenados.

Seu surgimento iniciou na década de 1960, quando Charles Bachman desenvolveu a base do modelo de dados de rede; mais tarde, no final dessa década, a IBM lançou seu SGBD. (RAMAKRISHNAN; GEHRKE, 2008, p.24).

2.18.1 Diagrama Entidade Relacionamento

O Der são utilizados para visualizar e estruturar as relações entre os dados de um sistema de Banco de Dados, orientando os desenvolvedores e projetistas a entenderem o domínio do problema e a projetar uma base estrutural eficiente. (RAMAKRISHNAN; GEHRKE, 2008, p.24).

Figura 37 – Exemplo Der

Diagrama

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Na figura 37 teremos a representação da sala cujo intuito é demonstrar os espaços físicos disponíveis para o uso. Já os alunos representam as pessoas que utilizam as salas, no contexto, os alunos que ocupam a sala para estudo. O relacionamento será estabelecido por Locação que ligará as salas aos alunos.

2.18.2 Modelo Entidade Relacionamento

Modelo de entidade relacional (MER) representa o banco como uma coleção de relações, onde cada equivale uma tabela de valores de registros lineares, chamados de tuplas, composto por atributos. (SILBERSCHATZ; SUDARSHAN, 2006).

Figura 38 – Exemplo Mer

Diagrama

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Com a demonstração da figura acima é notório as seguintes pautas. No modelo de dados, temos duas entidades principais: Sala e Aluno. A entidade Sala é identificada pelo atributo. Num\_Sala (PK), que serve como chave primária, indicando unicamente cada sala e sua capacidade máxima no atributo Quantidade\_Alunos. Na entidade Aluno, Id\_Aluno (PK) serve como identificador único, enquanto Nome e Ano\_Aluno detalham as informações pessoais e acadêmicas. O atributo num\_Sala (FK) em Aluno estabelece uma conexão direta com a Sala, mostrando onde cada aluno está alocado e garantindo a eficiência na gestão de espaço.

2.18.3 Banco De Dados Não Relacional

Para o termo NoSQL refere-se a um banco de dados que não segue normas de tabelas presente no banco de dados relacional, sendo ideais para lidar com grandes volumes de dados, solucionando problemas de escalabilidade (CAYRES,2015).

2.19 Firebase

O Firebase Database, é um método eficaz de criação de banco de dados, fazendo o uso de JSON para a atualização em tempo real junto de conceitos de armazenamento em nuvem, tornando-lhe versátil para diversos projetos (FIREBASE).

Figura 39 – Exemplo de banco de dados com o Firebase

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

3 DESENVOLVIMENTO

REFERÊNCIAS

AXIOS. **Introdução** [S.I], Axios, 2024. Disponível em: https://axios-http.com/ptbr/docs/intro. Acessado em: 07 jun 2024.

BARSOTI, N.; GIBERTONI, D. IMPACTO QUE O SEQUELIZE TRAZ PARA O DESENVOLVIMENTO DE UMA API CONSTRUÍDA EM NODE.JS COM EXPRESS.JS. **Revista Interface Tecnológica**, v. 17, n. 2, p. 231–243, 18 dez. 2020.

BRASIL. Lei nº 8.213, de 24 de julho de 1991. Dispõe sobre os planos de benefícios da previdência social e dá outras providências. **Da Finalidade e dos Princípios Básicos da Previdência Social**. Disponível em: https://www.planalto.gov.br/ccivil\_03/leis/l8213cons.htm. Acesso em: 07 jun 2024.

CAIUT, Fabio. **Administeação de Banco de Dados**. Rio de Janeiro: Escola Superior de Redes Rnp, 2015. 186 p.

CAYRES, Paulo Henrique. Modelagem de Banco de Dados. Rio de Janeiro: Escola Superior de Redes Rnp, 2015. 186 p.

CHEN, Daniel Y. **Análise de dados com Python e Pandas**. São Paulo: Novatec Editora, 2018. 432 p.

CLASEN, Beatriz. **TailwindCSS e Next.js**: dominando o desenvolvimento web com estilo. dominando o desenvolvimento web com estilo. 2023. Disponível em: https://blog.rocketseat.com.br/tailwindcss-e-next-js-dominando-o-desenvolvimento-com-estilo/. Acesso em: 30 abr. 2024.

CNN Brasil. **IBGE divulga estudo inédito sobre deficiência e desigualdades sociais no Brasil**. CNN Brasil, 22 maio 2023. Disponível em: https://www.cnnbrasil.com.br/nacional/ibge-divulga-estudo-inedito-sobre-deficiencia-e-desigualdades-sociais-no-brasil/. Acesso em: 21 maio 2024.

CNN Brasil**. Lei de cotas para pessoas com deficiência faz 30 anos neste sábado**. Disponível em: https://www.cnnbrasil.com.br/nacional/lei-de-cotas-para-pessoas-com-deficiencia-faz-30-anos-neste-sabado/. Acesso em: 15 jun. 2024.

DOMINGOS, Pedro. **O Algoritmo Mestre**: como a busca pelo algoritmo de machine learning definitivo recriará nosso mundo. São Paulo: Novatec, 2017. 344 p.

DUARTE JÚNIOR, Luiz Fernando. **Programação Web com Node.js**. 2. ed. Gravataí, Rs: Edição do Autor, 2022. 394 p.

DUCKETT, Jon**. HTML e CSS**: projete e construa websites. São Paulo: Alta Books, 2016. 512 p.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistema de banco de dados**. 6. ed. Universida do Texas em Arlington: Pearson Universidades, 2011. 788 p.

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

EXPRESS. **Framework web rápido, flexível e minimalista para node.Js** [S.I]. Express, 2024. Disponível em: https://expressjs.com/pt-br/. Acesso em 07 jun 2024

FIREBASE. **Aprenda os fundamentos.** [S.I]. Firebase, 2024. Disponível em: https://firebase.google.com/docs?hl=pt-br&\_gl=1\*18pchen\*\_up\*MQ.&gclid=504d29a01bc01aa2d6394f92b0c2ad51&gclsrc=3p.ds. Acesso em 07 jun 2024.

FLANAGAN, David. **Javascript**: o guia definitivo. 6. ed. Porto Alegre: Bookman, 2012. 1080 p. Tradução: João Eduardo Nóbrega Tortello.

FOWLER, Martin; TORTELLO, João. **UML Essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3. ed. Porto Alegre: Bookman, 2005.

G1. **Brasil tem 18,6 milhões de pessoas com deficiência, cerca de 8,9% da população**, segundo IBGE. 2023. Disponível em: https://g1.globo.com/economia/noticia/2023/07/07/brasil-tem-186-milhoes-de-pessoas-com-deficiencia-cerca-de-89percent-da-populacao-segundo-ibge.ghtml. Acesso em: 14 maio 2024.

G1. **O desafio de usar tecnologia na inclusão de pessoas com deficiência. Pequenas Empresas & Grandes Negócios**, 11 dez. 2017. Disponível em: https://g1.globo.com/economia/pme/pequenas-empresas-grandes-negocios/noticia/2017/12/o-desafio-de-usar-tecnologia-na-inclusao-de-pessoas-com-deficiencia.html. Acesso em: 13 maio 2024.

G1. Deficientes enfrentam dificuldades para ocupar vagas de trabalho reservadas por lei. G1, 11 set. 2019. Disponível em: https://g1.globo.com/pe/pernambuco/noticia/2019/09/11/deficientes-enfrentam-dificuldades-para-ocupar-vagas-de-trabalho-reservadas-por-lei.ghtml. Acesso em: 20 maio 2024.

GRANDE, Carine Casa; TANAKA, Simone Sawasaki. Comparaçao entre o desempenho de aplicações para smartphones desenvolvidas em Flutter e React Native: uma análise utilizando algoritmos de ordenação. **Revista Terra & Cultura: Cadernos de Ensino e Pesquisa**, v. 39, n.T especial, p. 7-17, 16 fev. 2023. Disponível em: http://publicacoes.unifil.br/index.php/Revistateste/article/view/2796/2559. Acesso em: 05 maio 2024.

GRONER, Loiane. **Estrutura de dados e algoritmos em javaScript**: escreva um código javascript complexo e eficaz usando a mais recente ecmascript. 2. ed. São Paulo, Sp-Brasil: Novatec Editora, 2019. 408 p. Tradução: Lúcia A. Kinoshita.

GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec Editora, 2011. 488 p.

GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. 3. ed. São Paulo: Novatec Editora, 2018. 496 p.

PEREIRA, Luiz Antônio Demoraes. **Análise e Modelagem de Sistemas com a UML**: com dicas e exercícios resolvidos. Rio de Janeiro: Edição do Autor, 2011. 282 p.

GRONER, Loiane. **Estrutura de dados e algoritmos em javaScript**: escreva um código javascript complexo e eficaz usando a mais recente ecmascript. 2. ed. São Paulo, Sp-Brasil: Novatec Editora, 2019. 408 p. Tradução: Lúcia A. Kinoshita.

**IBGE - Instituto Brasileiro de Geografia e Estatística. Desemprego e informalidade são maiores entre as pessoas com deficiência. Agência de Notícias IBGE, 22 nov. 2022. Disponível em:** https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/34977-desemprego-e-informalidade-sao-maiores-entre-as-pessoas-com-deficiencia. **Acesso em: 15 maio 2024.**

INHESTA, Danielle Christina Fernandes. **Uso de aprendizado de máquina para determinar as melhores práticas de implementação de chatbots**. 2022. Trabalho de Conclusão de Curso (MBA) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022. Disponível em: https://bdta.abcd.usp.br/directbitstream/5c19ea8d-736e-47c7-927e-9aecf5e80609/Danielle%20Christina%20Fernandes%20Inhesta\_Monografia\_207392.TCC.Danielle%20Inhesta\_Final%20Version\_22092022.Danielle.pdf. Acesso em: 28 abr. 2024.

KNEUSEL, Ronald T. **Como a Inteligência Artificial Funciona**: da magia à ciência. São Paulo: Novatec Editora Ltda, 2024. 256 p.

LOWDERMILK**,** **Travis.** Design Centrado no Usuário: um guia para o desenvolvimento de aplicativos amigáveis. São Paulo: Novatec Editora, 2013. 184 p.

LUCIDCHART. **O que é wireframe?** Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-wireframe. Acesso em: 18 maio 2024.

MENEZES, Nilo Ney Coutinho. **Introdução à Programação com Python**: algoritmos e lógica de programação para iniciantes. 2. ed. São Paulo, Sp-Brasil: Novatec Editora, 2014. 328 p.

MATTHERS, Eric. **Curso intensivo de python**: uma introdução prática e baseada em projetos à programação. São Paulo, Sp-Brasil: Novatec Editora, 2016. 656 p. Tradução: Lúcia A. Kinoshita.

MATOS, Nicole Fernandes Freitas**. Desenvolvimento e validação de Protótipos de Aplicativo para o Autocuidado de Gestantes**. 2023. 53 f. TCC (Graduação) - Curso de Engenharia da Computação, Engenharia da Computação, Universidade Federal do Ceará Seminário de Monografia Curso de Graduação em Engenharia da Computação, Sobral, 2023. Cap. 6. Disponível em: https://repositorio.ufc.br/handle/riufc/76303. Acesso em: 04 maio 2024.

MCKINNEY, Wes. **Python para análise de dados:** tratamento de dados com pandas, numpy e ipython. São Paulo: Novatec Editora, 2018. 616 p.

MENEZES, Nilo Ney Coutinho. **Introdução à Programação com Python**: algoritmos e lógica de programação para iniciantes. 2. ed. São Paulo, Sp-Brasil: Novatec Editora, 2014. 328 p.

MENEZES, Nilo Ney Coutinho. **Introdução à Programação com Python**: algoritmos e lógica de programação para iniciantes. 3. ed. São Paulo, Sp-Brasil: Novatec Editora, 2019. 328 p.

MIRO. **O que é um wireframe?** Como criar? Tipos, exemplos e modelos. Disponível em: https://miro.com/pt/wireframe/o-que-e-wireframe. Acesso em: 18 maio 2024.

MORAES, William Bruno. **Construindo Aplicações com NodeJS**. 3. ed. São Paulo: Novatec, 2021. 272 p.

MOZILLA. **Cross-Origin Resource Sharing** (CORS). Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS. Acesso em: 18 jun. 2024.

MUELLER, John Paul. **Começando a Programar em Python**: para leigos. 2. ed. Rio de Janeiro Rj-Brasil: Alta Books, 2020. 391 p.

NUMPY. **Quem Somos** [S.I]. NumPy, 2024. Disponivel em: https://numpy.org/pt/about/. Acesso em 07 jun 2024

PEREIRA, Caio Ribeiro. **Aplicações web real-time com Node.js**. S. L: Casa do Código, 2013. 2 p.

PEREIRA, Eduardo. **Computação Evolucionária**: aplique os algoritmos genéticos com python e numpy. Flórianopolis: Casa do Código, 2020. 166 p.

POWERS, Shelley. **Aprendendo Node**: usando javascript no servidor. 2. ed. São Paulo: Novatec Editora, 2019. 312 p. Tradução em português autorizada da edição em inglês da obra Learning Node, 2nd Edition, ISBN 9781491943120 © 2016 Shelly Powers. Esta tradução é publicada e vendida com a permissão da O'Reilly Media, Inc., detentora de todos os direitos para publicação e venda desta obra.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia se Software**: uma abordagem profissional. 9. ed. Porto Alegre: Amgh Editora Ltda, 2021. 704 p. Tradução de: Francisco Araújo da Costa.

**Qual a relação entre tecnologia e acessibilidade?** Disponível em: https://esr.rnp.br/temas-diversos/tecnologia-e-acessibilidade/. Acessado em 07 jun 2024

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistema de Gerenciamento de Banco de Dados**. 3. ed. São Paulo,Sp-Brasil: Amgh, 2008. 905 p.

REACT. **Introdução** [S.I], React, 2024. Disponível em: https://pt-br.legacy.reactjs.org/docs/getting-started.html. Acessado em: 07 jun 2024.

REACT VITE. **Visão geral** [S.I], React vite, 2024. Disponivel em: https://pt.vitejs.dev/guide/.Acessado em: 07 jun 2024.

**ROCKETSEAT.** **Expo e React Native**: a união que está transformando o desenvolvimento mobile. Blog Rocketseat, 09 out. 2020. Disponível em: https://blog.rocketseat.com.br/expo-react-native/. Acesso em: 09 jun. 2024.

SCHEIDT, F. A. **Fundamentos de CSS: criando design para sistemas web**. [s.l.] Outbox Livros Digitais, 2015. 6 p.

SCHMITZ, Daniel; GEORGII, Daniel Pedrinha. **React - Guia do Iniciate**: domine a biblioteca javascript utilizada pelo facebook e instagram. São Paulo: Leanpub, 2015. 51 p.

SILBERSCHATZ, Abraham; SUDARSHAN, S. Korth Henry F**. Sistema de Banco de Dados**. 5. ed. Rio de Janeiro Rj-Brasil: Editora Campus, 2006. 781 p.

SILVA, M. S. **Criando Sites com HTML: Sites de Alta Qualidade com HTML e CSS**. [s.l.] Novatec Editora, 2008. 213 p.

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

SILVA, Mauricio Samy. **React Aprenda Praticando:** desenvolva aplicações web reais com uso da biblioteca react e de seus módulos auxiliares. São Paulo: Novatec Editora Ltda, 2021. 240 p.

**Tecnologia e Inclusão: importância e desafios**. Disponível em: https://www.handtalk.me/br/blog/tecnologia-e-inclusao/. Acessado em 07 jun 2024.

TEIXEIRA, Fabricio. **Introdução e Boas Práticas em UX Design.** São Paulo: Casa do Código, 2014. 217 p.

VASCONCELOS, Davi R.; GUERRA, Paulo T. **Ensinando Teoria da Computação com Jupyter Notebook**. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 31., 2023, João Pessoa/PB. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023. p. 9-19. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2023.229496.