**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**

**Palavras-Chaves**:

**ABSTRACT**

**Key words:**

**Lista de Figuras**

[Figura 1 – Exemplo de HTML 10](#_Toc166791545)

[Figura 2 – Exemplo de Formulário HTML 12](#_Toc166791546)

[Figura 3 – Exemplo de formulário no navegador 13](#_Toc166791547)

[Figura 4 – Exemplo tag “link” 14](#_Toc166791548)

[Figura 5 – Exemplo de código CSS parte 1 15](#_Toc166791549)

[Figura 6 – Exemplo de código CSS parte 2 16](#_Toc166791550)

[Figura 7 – Exemplo de formulário estilizado 17](#_Toc166791551)

[Figura 8 – Exemplo de Formulário com JavaScript 18](#_Toc166791552)

[Figura 9 – Mensagem de condição Javascript 19](#_Toc166791553)

[Figura 10 – Mensagem de sucesso Javascript 20](#_Toc166791554)

[Figura 11 – Mensagem de cancelamento Javascript 20](#_Toc166791555)

[Figura 12 – Exemplo de código de tailwind 21](#_Toc166791556)

[Figura 13 – Exemplo de formulário com tailwind 22](#_Toc166791557)

[Figura 14 – Exemplo de estrutura básica React em código 24](#_Toc166791558)

[Figura 15 – Exemplo de estrutura básica React no navegador 25](#_Toc166791559)

[Figura 16 – Exemplo de formulário no React 26](#_Toc166791560)

[Figura 17 – Exemplo de formulário no navegador em React 27](#_Toc166791561)

[Figura 18 – Exemplo de Home em React Native 28](#_Toc166791562)

[Figura 19 – Exemplo de estilização em React Native 30](#_Toc166791563)

[Figura 20 - Exemplo de Funcionamento Node.js 31](#_Toc166791564)

[Figura 21 – Exemplo de Verificação de Senha em Python 33](#_Toc166791565)

[Figura 22 – Exemplo de código com a utilização de pacote 35](#_Toc166791566)

[Figura 23 – Exemplo de instalação de pacote emoji 36](#_Toc166791567)

[Figura 24 – Demonstração de emoji no cmd 37](#_Toc166791568)

[Figura 25 – Tela de criação do Jupyter Notebook 38](#_Toc166791569)

[Figura 26 - Exemplo de análise de dados com Pandas 39](#_Toc166791570)

[Figura 27 -Exemplo de análise de uma base de dados de um sistema de recomendação de filmes 40](#_Toc166791571)

[Figura 28 – Exemplo de DER 42](#_Toc166791572)

[Figura 29 – Exemplo de MER 42](#_Toc166791573)

**Lista de abreviaturas e siglas**

Cascading Style Sheets (CSS)

Diagrama Entidade Relacionamento (DER)

Document Object Model (DOM)

Chave Estrangeira (FK)

HyperText Markup Language (HTML)

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 8](#_Toc166831399)

[2 REFERENCIAL TEÓRICO 9](#_Toc166831400)

[2.1 Plataforma Para Conectar as Empresas aos PCD 9](#_Toc166831401)

[2.2 Tecnologias Utilizadas 9](#_Toc166831402)

[2.2.1 Hypertext Markup Language (Html) 10](#_Toc166831403)

[2.2.2 Cascading Style Sheets (CSS) 14](#_Toc166831404)

[2.2.3 Javascript (JS) 18](#_Toc166831405)

[2.2.4 Tailwind CSS 21](#_Toc166831406)

[2.2.5 React 23](#_Toc166831407)

[2.2.5.1 Vite 23](#_Toc166831408)

[2.2.6 React Native 28](#_Toc166831409)

[2.2.7 Node.JS 30](#_Toc166831410)

[2.2.7.1 Express 31](#_Toc166831411)

[2.2.7.2 Node Package Manager (npm) 32](#_Toc166831412)

[2.2.7.3 Node Package Executor (NPX) 32](#_Toc166831413)

[2.2.8 Python 33](#_Toc166831414)

[2.2.9 Jupyter 38](#_Toc166831415)

[2.2.12 Banco De Dados 41](#_Toc166831416)

[2.2.12.1 Diagrama Entidade Relacionamento 41](#_Toc166831417)

[2.2.12.2 Modelo Entidade Relacionamento 42](#_Toc166831418)

[2.2.12.3 Banco De Dados Relacional 43](#_Toc166831419)

[2.2.12.4 Banco De Dados Não Relacional 43](#_Toc166831420)

[2.2.13 Firebase 43](#_Toc166831421)

[REFERÊNCIAS 44](#_Toc166831422)

1 INTRODUÇÃO

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 toda a fundamentação da plataforma DAELink.

2.1 Plataforma Para Conectar as Empresas aos PCD

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, 2019) comprova que essas pessoas enfrentam maiores dificuldades para entrar no mercado de trabalho cerca de (28,3%) deficientes e (66,3%) sem deficiência.

No Brasil, a Lei nº 8.213/91 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.

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

Posto que 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 (G1,2017). Portanto, há uma necessidade de criação de um sistema que facilite essa integração digital. Uma solução baseada em redes sociais profissionais pode promover a inclusão social e melhorar o acesso dessas pessoas ao mercado de trabalho.

2.2 Tecnologias Utilizadas

Nesta seção estão comentadas de forma expositiva as tecnologias utilizadas na criação do sistema DAELink, sendo elas linguagens, *frameworks* e outros elementos utilizados na modelagem do projeto.

2.2.1 Hypertext Markup Language (Html)

*HTML* é uma linguagem de marcação de hipertexto, 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 1 – Exemplo de HTML

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Neste exemplo 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 principal, como 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* fica a parte visual que o navegador executa, ficando todo o conteúdo dentro dela.
* “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 2 – Exemplo de Formulário HTML

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 imagem a seguir:

Figura 3 – Exemplo de formulário no navegador

A screenshot of a computer

Description automatically generated

Fonte: Autoria Própria, 2024.

2.2.2 Cascading Style Sheets (CSS)

O *CSS*, ou em português folha de estilo em cascata, 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 4 – 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 5 – Exemplo de código CSS parte 1

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Deste exemplo, 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 6 – Exemplo de código CSS parte 2

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 7 – Exemplo de formulário estilizado

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.2.3 Javascript (JS)

O Javascript se originou pela Netscape, sendo documentado pela Sun Microsystems (Oracle), 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 funcional.

Na Figura 8 veremos o funcionamento do Javascript dentro do formulário mostrado anteriormente, exemplificando o envio de componentes digitados dentro das caixas de textos.

Figura 8 – 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 9 – Mensagem de condição Javascript

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

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

Figura 10 – Mensagem de sucesso Javascript

Texto

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

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

Figura 11 – Mensagem de cancelamento Javascript

Forma, Retângulo

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.2.4 Tailwind CSS

Tailwind CSS é um framework de estilo CSS código aberto, cujo intuito é 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 12, será demonstrado a realização de um código utilizando tailwind CSS em um formulário semelhante ao mostrado anteriormente.

Figura 12 – Exemplo de código de tailwind

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 13 – Exemplo de formulário com tailwind

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

2.2.5 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).

2.2.5.1 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 14 – Exemplo de estrutura básica React em código

A screen shot of a computer program

Description automatically generated

Fonte: Autoria Própria, 2024.

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

Figura 15 – Exemplo de estrutura básica React no navegador

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 16 – Exemplo de formulário no React

A screen shot of a computer

Description automatically generated

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 imagem abaixo.

Figura 17 – Exemplo de formulário no navegador em React

A black square with white text and red letters

Description automatically generated

Fonte: Autoria Própria, 2024.

2.2.6 React Native

De acordo com Escudelario e Pinho (2020), o React Native é uma plataforma baseada no 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 no exemplo abaixo.

Figura 18 – Exemplo de Home em React Native

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 19 – Exemplo de estilização em React Native

A screen shot of a computer program

Description automatically generated

Fonte: Autoria Própria, 2024.

A imagem anterior 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.

2.2.7 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 20 - Exemplo de Funcionamento Node.js

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”.

2.2.7.1 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 renderizando em telas HTML dinâmicas (BARSOTI; GIBERTONI, 2020).

Conforme a documentação do Express, ele utiliza de recursos para criação de rotas agíeis com uma fácil manipulação de dados, trazendo códigos mais dinâmicos e flexíveis.

2.2.7.2 Node Package Manager (npm)

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).

2.2.7.3 Node Package Executor (NPX)

NPX é um comando incorporado ao NPM, que faz a execução de módulos sem instalá-los no projeto, servindo para requisições rápidas, através da remoção automática desses módulos após seu uso, economizando memória. (MORAES, 2021).

2.2.8 Python

Segundo Matthers (2016) Python é uma linguagem de programação extremamente eficientes pelo fato dos 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.

Desta maneira a versatilidade, a simplicidade faz do Python uma das linguagens de programação mais populares do mundo. Sendo seu contínuo crescimento em popularidade, o que a torna uma escolha valiosa para qualquer programador (MENEZES, 2019, p26).

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 12 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 21 – Exemplo de Verificação de Senha em Python

**Tela de computador com texto preto sobre fundo branco

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. Sendo assim será armazenado dentro da variável ‘*True’* (verdadeiro) ‘tem\_tamanho\_adequado’, demonstrando que a senha atende esse critério de segurança, mas caso não deverá ser armazenado ‘*False’*(falso).
* *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.
* *If* tem\_tamanho-adequado and tem\_numero and tem\_letra\_maiuscula: a linha inicia-se com uma declaração condicional (‘*if*’) que verifica se todas as três condições anteriores são verdadeiras. Se forem, executa o bloco de código endentado abaixo dela.
* *Return* “Senha segura!”: Se a condição do ‘*if*’ for verdadeira, a função retorna a String “Senha segura!”.
* *Return* dentro do *Else*: Se a condição citada anteriormente não for verdadeira, a função retorna outra *string* explicando por que a senha é considerada fraca.
* *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).

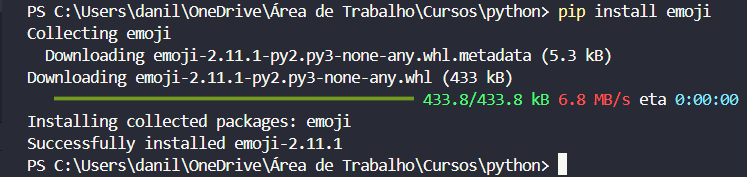
Na figura a seguir demonstrará um exemplo de código que usará um pacote que não está dentro do python então será realizado a instalação

Figura 22 – Exemplo de código com a utilização de pacote

 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. “
* Input: “é uma função embutida que lê uma linha do input do usuário no contexto de uma pergunta, e espera que o usuário pressione Enter para continuar.”
* Print (emojize(random.choice(self.resposta))):“Responsável por imprimir uma resposta aleatória, selecionando uma das strings da lista ‘self.respostas’ através disso a função ‘emojize’ é usada então para converter quaisquer códigos de emoji na string selecionada em emojis reais. ”
* 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. ”

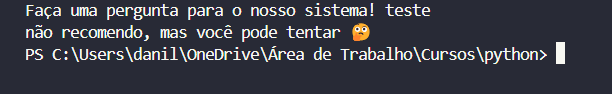
Figura 23 – Exemplo de instalação de pacote emoji



Fonte: Autoria Própria, 2024.

Na figura 22 é usado o pip install emoji para a instalação dos pacotes de emojis, devido não serem incluídos dentro do Python.

Figura 24 – Demonstração de emoji no cmd

 Fonte: Autoria Própria, 2024.

Segundo a figura 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.

2.2.9 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 linhagens de marcação e códigos (VASCONCELOS; GUERRA, 2023).

Figura 25 – Tela de criação do Jupyter Notebook

A screenshot of a computer

Description automatically generated

Fonte: Autoria Própria, 2024.

**2.2.10 Machine Learning**

O Machine Learning utiliza da filtração de dados para a criação de novas informações, gerando resultados significativos, possibilitando a tomada de decisões inteligente através desses 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).

**2.2.11.1 Numpy**

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.

**2.2.11.2 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 26 - Exemplo de análise de dados com Pandas

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Para a criação do exemplo acima foi utilizado o comando “Read\_csv”, 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.

**2.2.11.3 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 27 -Exemplo de análise de uma base de dados de um sistema de recomendação de filmes

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.

2.2.12 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 de 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).

2.2.12.1 Diagrama Entidade Relacionamento

Der são utilizados para visualizar e estruturar as relações entre os dados em 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 28 – Exemplo de DER

Diagrama

Descrição gerada automaticamente

Fonte: Autoria Própria, 2024.

Na figura 28 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.

2.2.12.2 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 29 – Exemplo de 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.

2.2.12.3 Banco De Dados Relacional

Este tipo de banco é baseado em um modelo relacional e emprega um conjunto de tabelas para representar os dados e as relações entre eles, sendo utilizado como base a linguagem de dados SQL, tornando padrão de banco. (SILBERSCHATZ; SUDARSHAN, 2006).

2.2.12.4 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).

2.2.13 Firebase

Firebase Realtime 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).

REFERÊNCIAS

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.

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.

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.

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.

**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.

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.

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

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.

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

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.

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

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.