

Relatório Trabalho Final Sistemas de Apoio a Decisão

Aluno: Pedro Arthur Cunha Anício

1. Descrição das tecnologias utilizadas

Frontend

- **React:** Utilizado para criar a interface do usuário de forma eficiente e reativa. O React permite a construção de componentes reutilizáveis e facilita a criação de interfaces dinâmicas com base no estado da aplicação.
- **Axios:** Biblioteca utilizada para fazer requisições HTTP ao backend. Axios foi escolhido por ser simples de usar e por ser amplamente utilizado para integração com APIs REST.
- **React Router:** Usado para navegação entre as páginas do aplicativo sem a necessidade de recarregar a página, permitindo a construção de uma Single Page Application (SPA).

Backend

- **Node.js:** Ambiente de execução JavaScript utilizado para o desenvolvimento do backend. Ele é ideal para aplicações que precisam de alta performance e escalabilidade.
- **Express:** Framework minimalista para Node.js que facilita a criação de APIs RESTful. Foi usado para configurar as rotas e lidar com as requisições HTTP de forma simples e eficaz.
- **MongoDB:** Banco de dados NoSQL utilizado para armazenar os sintomas e diagnósticos. MongoDB foi escolhido pela sua flexibilidade e escalabilidade, permitindo a armazenagem de documentos JSON.
- **Mongoose:** Biblioteca para Node.js que fornece uma solução elegante para modelar dados no MongoDB.

2. Descrição da arquitetura ou classes utilizadas no sistema

Arquitetura

A arquitetura do sistema segue o padrão cliente-servidor, onde o frontend em React faz requisições HTTP ao backend, que é responsável pelo processamento dos dados e pela comunicação com o banco de dados.

- **Frontend (React):**
 - **Home:** Página principal do sistema com informações introdutórias e links para outras funcionalidades.
 - **Diagnóstico:** Página onde o usuário seleciona sintomas para obter um diagnóstico baseado nas respostas fornecidas.
 - **EditarTabela:** Página para o administrador adicionar, atualizar ou excluir sintomas da base de dados.
 - **Navbar:** Componente de navegação para facilitar o acesso às diferentes funcionalidades.
- **Backend (Node.js + Express):**
 - **Routes:** Definem as rotas de API para gerenciar os sintomas e gerar diagnósticos.
 - GET /sintomas: Retorna todos os sintomas cadastrados.
 - POST /sintomas: Adiciona um novo sintoma.

- PUT /sintomas/:id: Atualiza o nome de um sintoma existente.
- DELETE /sintomas/:id: Exclui um sintoma existente.
- POST /api/diagnostico: Recebe os sintomas e gera um diagnóstico.
- **Banco de Dados (SQLite):**
 - **Sintomas:** Coleção de sintomas cadastrados no sistema, com o campo nome que armazena o nome do sintoma.

Classes no Sistema

- **Sintoma:** Classe que representa um sintoma, com o campo nome.
 - Métodos: adicionarSintoma, atualizarSintoma, excluirSintoma.

3. Trechos de códigos

- **Pegar sintomas cadastrados no banco de dados:**

```
def obter_sintomas(db):
    cursor = db.cursor()
    cursor.execute('SELECT nome FROM sintomas')
    return [row[0] for row in cursor.fetchall()]
```

Executa uma consulta SQL para selecionar todos os nomes de sintomas cadastrados pelo usuário no SQLite.

- **Gerar dados de treinamento:**

```
def gerar_dados_treinamento(sintomas):
    """ Gera dados fictícios associando sintomas a doenças diferentes. """
    doencas_possiveis = ["Gripe", "Covid", "Dengue", "Resfriado", "Alergia"]
    dados_ficticios = []

    for _ in range(500): # 500 exemplos de treino
        sintomas_paciente = {s: random.choice(["Leve", "Médio", "Forte", "Nenhum"]) for s in sintomas}
        doenca = random.choice(doencas_possiveis)
        linha = {**sintomas_paciente, "Doença": doenca}
        dados_ficticios.append(linha)

    return pd.DataFrame(dados_ficticios)
```

Cria um conjunto de dados fictício para treinar o modelo. Para cada um dos 50 exemplos de treinamento gera um conjunto de sintomas aleatórios com intensidades distintas e associa uma doença aleatória a esses sintomas.

- **Treinamento do modelo:**

```
def treinar_modelo(df):
    df_encoded = pd.get_dummies(df.drop('Doença', axis=1)) # gerar colunas "dummies"
    X = df_encoded
    y = df['Doença']
    clf = DecisionTreeClassifier(max_depth=5, min_samples_split=3, min_samples_leaf=2, criterion='entropy')
    clf.fit(X, y)
    return clf, X.columns
```

Treina o modelo de árvore de decisão usando os dados fictícios. A função converte as colunas categóricas em colunas binárias, define as variáveis independentes (x) como os sintomas e a variável depende de (y) como a doença.

- **Previsão da doença:**

```
def prever_doenca(sintomas, clf, colunas_esperadas):  
    """ Ajusta os sintomas recebidos para o formato esperado pelo modelo """  
    sintomas_df = pd.DataFrame([**{s: "Nenhum" for s in colunas_esperadas}, **sintomas])  
  
    sintomas_df = pd.get_dummies(sintomas_df)  
    for col in colunas_esperadas:  
        if col not in sintomas_df:  
            sintomas_df[col] = 0 # se a coluna não existe, preenche com zero  
  
    sintomas_df = sintomas_df[colunas_esperadas]  
    return clf.predict(sintomas_df)[0]
```

Usa o modelo treinado para prever a doença com base nos sintomas recebidos. É criado um DataFrame com os sintomas recebidos, preenchendo os sintomas ausentes com “Nenhum”. Após isso, converte os sintomas em colunas binárias (pd.get_dummies), garante que todas as colunas esperadas pelo modelo estejam presentes, reordena as colunas para corresponder ao formato usado no treinamento e, por fim, faz a previsão usando o modelo treinado.

- **Tabela sintomas SQLite:**

```
const sqlite3 = require('sqlite3').verbose();  
  
const db = new sqlite3.Database('./sintomas.db', (err) => {  
    if (err) {  
        console.error('Erro ao conectar ao banco de dados:', err.message);  
    } else {  
        console.log('Conectado ao banco de dados SQLite.');        db.run(`CREATE TABLE IF NOT EXISTS sintomas (  
            id INTEGER PRIMARY KEY AUTOINCREMENT,  
            nome TEXT NOT NULL  
        )`, (err) => {  
            if (err) console.error('Erro ao criar tabela:', err.message);  
        });  
    }  
});  
  
module.exports = db;
```

4. Imagens das telas

- Tela de diagnostico:

Sistema de Diagnóstico

Diagnóstico

Editar Tabela

Diagnóstico

Dor de Cabeça

Irrelevante

Dor no Peito

Irrelevante

Náusea

Irrelevante

Dor de dente

Irrelevante

Palpitações

Irrelevante

Insônia

Irrelevante

Desmaio

Irrelevante

Espirros

Irrelevante

Tosse

Irrelevante

Diagnosticar

Resultado:

Resfriado

- Tabela de sintomas:

Editar Tabela de Sintomas

ID	Sintoma	Ações
7	<input type="text" value="Dor de Cabeça"/>	<div>Excluir</div>
8	<input type="text" value="Dor no Peito"/>	<div>Excluir</div>
9	<input type="text" value="Náusea"/>	<div>Excluir</div>
10	<input type="text" value="Dor de dente"/>	<div>Excluir</div>
<input type="text" value="Novo sintoma"/>		<div>Adicionar</div>