DOCUMENTAÇÃO TÉCNICA DO PROJETO – API DOAÇÃO DE SANGUE

TÍTULO: Sistema de cadastro doadores de sangue.

Grupo:

Nome: Amanda Ferreira RA: 39248151

Nome: Ligia Maria RA: 39106021

Nome: Luiz Storrer RA: 41859863

Nome: Wellington Pedro RA: 38414392

1. **OBJETIVO DO PROJETO:** O objetivo deste projeto é desenvolver uma API REST voltada ao cadastro e gerenciamento de doadores de sangue. A API permite realizar operações de listagem, consulta, adição e exclusão de registros de doadores, utilizando boas práticas de desenvolvimento. A aplicação pode ser utilizada por instituições responsáveis por bancos de sangue, facilitando o controle de dados pessoais e do tipo sanguíneo dos doadores.

2. **ESTRUTURA DA SOLUÇÃO:** A modelagem de dados e aplicação foi centrada na entidade "DOADOR", contendo os seguintes atributos listados:

   Id: identificador único
   Nome: nome completo do doador
   TipoSanguineo: classificação do sangue (ex: O+, A-)
   DataUltimaDoacao: data da última doação registrada

   Esses dados foram estruturados em uma classe Doador.cs, que serviu como base para o mapeamento do banco e definição das rotas.

3. **ENDPOINTS DA API**

MÉTODO	ROTA	DESCRIÇÃO
GET	/api/doadores	Retorna todos os doadores
GET	/api/doadores/{id}	Retorna um doador por ID
POST	/api/doadores	Adiciona um novo doador
DELETE	/api/doadores/{id}	Remove um doador por ID

4. **ORGANIZAÇÃO DO CÓDIGO:** O código da aplicação está dividido da seguinte forma:

   a. Models/Doador.cs – Estrutura da entidade principal
   b. Data/DoadoresContext.cs – Contexto do banco de dados com Entity Framework
   c. Rotas/GetRoutes.cs – Métodos de listagem e consulta
   d. Rotas/PostRoutes.cs – Cadastro de novos doadores
   e. Rotas/DeleteRoutes.cs – Remoção de doadores
   f. Program.cs – Configuração principal da API

5. **JUSTIFICATIVA TÉCNICA:** A modelagem da entidade Doador foi construída com foco nos dados essenciais ao controle de bancos de sangue. A separação em arquivos distintos para rotas GET, POST e DELETE favorece a organização, clareza e manutenção do código.
   A integração com o Entity Framework e uso de banco SQLite atendem às necessidades de persistência com simplicidade, facilitando o uso em ambiente local ou portável. A base de dados foi populada com alguns registros iniciais de doadores para fins de teste e validação.