Universidade da Beira Interior
Departamento de Informática

GammaWatch Project - Monitoring Radiation in the Portuguese Environment

Elaborado por:

Marco Bernardes 45703

Orientador:

Professor Doutor João Manuel da Silva Fernandes Muranho

3 de janeiro de 2024

Agradecimentos

A realização do presente relatório não seria possível sem o apoio, contributo e esforço de várias pessoas que direta ou indiretamente influenciaram o rumo deste trabalho. Assim sendo, pretendo agradecer a todos que sempre me encorajaram a seguir e a fazer sempre melhor.

Ao professor doutor João Manuel da Silva Fernandes Muranho, pela sua orientação, total colaboração e disponibilidade, pelos seus ensinamentos que contribuíram para o meu enriquecimento educacional, por toda a ajuda prestada, não apenas neste trabalho, mas fundamentalmente ao longo de todo o semestre, sendo um privilégio e uma honra ser seu educando. A ele, o meu muito Obrigado!

A todos os docentes que contribuem para a minha formação e crescimento académico ao longo desta licenciatura, pelo saber que transmitem, os valiosos conhecimentos que me incutem e pela excelência da qualidade técnica de cada um.

E por fim, agradeço de forma especial à minha família, pela confiança no meu progresso, pelo apoio incondicional e por incentivarem e apoiarem em todas as áreas da minha vida.

A todos o meu sincero e profundo Muito obrigado!

Conteúdo

Conteúdo **iii**

Lista de Figuras **v**

1 Introdução **1**
1.1 Introdução 1
1.2 Enquadramento 1
1.3 Organização do Documento 1

2 Desenvolvimento da Aplicação Cliente/Servidor **3**
2.1 Introdução 3
2.2 Aplicação Cliente/Servidor 3
2.3 SQL Server 4
2.4 Configuração do Acesso ao Servidor 4
2.5 Ambiente de Desenvolvimento 4

3 Modelação **5**
3.1 Introdução 5
3.2 Modelo de dados 5
3.3 Considerações 7
3.3.1 Procedimento de inserção 8
3.3.2 Gatilho de análise de Leitura 9

4 Implementação e Testes **13**
4.1 Introdução 13
4.2 Acesso à base de dados 13
4.3 Interface e Funcionalidades 13

5 Conclusões e Trabalho Futuro **17**
5.1 Conclusões 17
5.2 Trabalho Futuro 17
5.3 Apêndices 17

Lista de Figuras

3.1 Diagrama Entidade Associação. 6
3.2 Esquema Relacional. 7

4.1 PYODBC 13
4.2 Página principal 14
4.3 Página principal (2) 14
4.4 Página da Estação. 15
4.5 Página do Sensor. 15

Acrónimos

UX User Interface

Capítulo

1

Introdução

1.1 Introdução

O trabalho que foi desenvolvido teve como objetivo principal implementar um sistema de informação semelhante ao da RADNET portuguesa, composto por uma base de dados e uma aplicação que permitisse interagir com o sistema desenvolvido.

As principais interações com o sistema incluem a visualização e análise mais simplificadas dos dados obtidos pelos sensores, como também a consciencialização sobre os riscos da radiação e outras informações úteis para o utilizador comum.

1.2 Enquadramento

Este documento relata o processo de desenvolvimento realizado para o projeto da Unidade Curricular de Bases de Dados da Universidade da Beira Interior

1.3 Organização do Documento

De modo a refletir o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. No primeiro capítulo — Introdução — são apresentados o projeto, o enquadramento do mesmo, a enumeração dos objetivos delineados para a conclusão do mesmo e a respetiva organização do documento.

2. No segundo capítulo — Desenvolvimento da Aplicação Cliente/Servidor — são apresentadas e descritas as escolhas e os métodos utilizados no desenvolvimento da plataforma no enunciado.

3. No terceiro capítulo é explorado o processo de modelação da base de dados e é explicado em pormenor todas as escolhas que foram feitas e decisões tomadas

4. No quarto capítulo — Implementação e Testes - É explicado todo o processo de acesso à base de dados, a interface e as funcionalidades introduzidas.

5. No quinto capítulo — Conclusões e Trabalho Futuro — apresenta-se uma reflexão do trabalho e conhecimentos adquiridos ao longo do desenvolvimento do projeto prático e um contrabalanço com a possibilidade de existirem objetivos não alcançados e que se podem explorar no futuro.

Capítulo

2

Desenvolvimento da Aplicação Cliente/Servidor

2.1 Introdução

Neste capítulo, discutiremos o desenvolvimento da aplicação cliente/servidor que utiliza a base de dados como a sua principal fonte de dados. Abordaremos tanto a criação da interface do utilizador da aplicação cliente quanto a implementação da lógica no lado do servidor. Também exploraremos vários aspetos da comunicação entre o cliente e o servidor.

2.2 Aplicação Cliente/Servidor

Para o desenvolvimento do presente trabalho foram usadas diversas frameworks, sendo as principais o **ReactJS**, escolhido para a interface do projeto e o **Flask** (*framework* python), para o servidor e o **SQL Server** que foi escolhido a ferramenta principal de gestão da base de dados da aplicação.

Dado a arquitetura do sistema. Foi necessário o desenvolvimento de um ***Backend*** e um ***Frontend***. O ***Backend***, também chamado de ***Server-Side*** foi construído usando o ***Python***, sendo responsável pelo tratamento dos dados em trânsito do ***frontend*** para a base de dados, o tratamento das ***requests*** recebidas e outras funcionalidades importantes. O ***frontend***, ou ***client-side***, foi desenvolvida usando o JavaScript. O ***frontend*** é responsável pela renderização da ***User Interface (UX)*** e pelo tratamento das interações do utilizador.

2.3 SQL Server

SQL Server é uma ferramenta potente e muito utilizada na gestão de bases de dados relacionais. É um sistema projetado para gerir grandes quantidades de dados de maneira eficiente e fornecer acesso rápido a esses dados através de consultas SQL.

2.4 Configuração do Acesso ao Servidor

2.5 Ambiente de Desenvolvimento

A aplicação foi desenvolvida utilizando várias ferramentas e tecnologias que nos permitiram criar e gerir o projeto de maneira eficiente.

Para o *frontend* da aplicação, foi utilizado ReactJS com Javascript. O ReactJS é uma *framework* JavaScript de código aberto utilizada para criar interfaces de utilizador complexas e interativas. É projetado para permitir a construção de aplicações *web* de forma rápida e eficiente, fornecendo uma série de ferramentas para criar componentes reutilizáveis e gerir o estado da aplicação de maneira eficiente.

Para o backend da aplicação, foi utilizado Flask, que é uma *framework* de código aberto baseada em Python para criar aplicações web de alto desempenho. Foi também utilizado SQL como linguagem de base de dados principal e Git como controlador de versões. O ERDPlus foi outra ferramenta online importante utilizada para criar o modelo de dados da aplicação.

Resumindo, o ambiente de desenvolvimento da aplicação inclui uma combinação de tecnologias desde o *frontend* ao *backend* que permitiram criar uma aplicação de qualidade e gerir o projeto de maneira eficiente.

Capítulo

3

Modelação

3.1 Introdução

A modelação de dados é um processo de criação de um modelo lógico de dados para representar a estrutura de uma base de dados. Esse modelo é criado para representar de maneira precisa e lógica os dados armazenados na base de dados e as relações entre os mesmos.

A modelação de dados é uma etapa fundamental na criação de uma base de dados, pois permite que os dados sejam armazenados de maneira lógica e consistente, garantindo assim a sua integridade. Além disso, a modelação de dados também é importante porque permite entender como estão estruturados os dados e como se relacionam, o que facilita o acesso e a manipulação.

3.2 Modelo de dados

O processo de desenvolvimento do modelo de dados para o trabalho em questão iniciou-se durante as aulas práticas da Unidade Curricular de Base de Dados. Neste período, o grupo começou a criar rascunhos simples com o objetivo de entender que entidades seriam necessárias para atender às necessidades da tarefa proposta e quais seriam os seus atributos. Esses rascunhos foram criados com o intuito de auxiliar o processo de modelação de dados e de definir a estrutura lógica da base de dados. A partir daí, o grupo pôde entender qual seria a melhor forma de representar os dados e as relações entre eles, o que foi fundamental para o desenvolvimento do sistema.

Dado a descrição do contexto em que a aplicação irá estar inserida, foram assumidas as seguintes entidades como as principais:

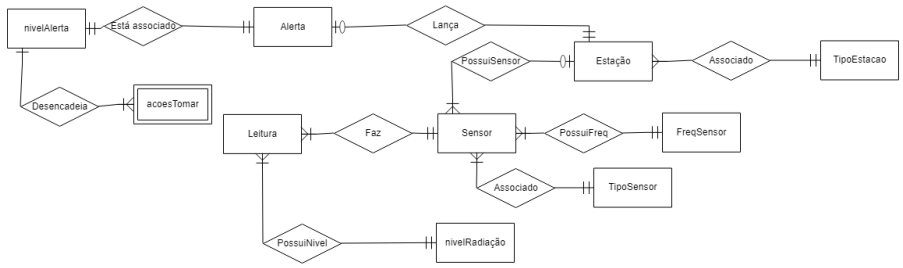

Figura 3.1: Diagrama Entidade Associação.

1. Estação
2. Sensor
3. Leitura
4. Alerta
5. Nível de Radiação
6. Nível de Alerta

Além disso, foram tomadas precauções para garantir que o modelo se mantenha eficaz face a possíveis mudanças nos níveis de radiação e nas ações a serem tomadas. Essas medidas permitem que, ao modificar os níveis, todos os dados na base de dados sejam atualizados de acordo com essa mudança, assegurando a sua relevância e precisão.

Outra decisão crucial foi a de determinar o papel atribuído a cada sensor. Como o enunciado não especificava essa função, foi preciso optar entre um sensor inteligente capaz de armazenar cada leitura ou um sensor que enviasse todos os dados para processamento na estação.

Para este trabalho em particular, optou-se por capacitar cada sensor individualmente, permitindo que cada um armazene os seus dados e, só em situações de alarme, os envia para a estação com uma frequência predefinida.

No final da idealização as entidades e relações necessárias para a aplicação foram as seguintes:

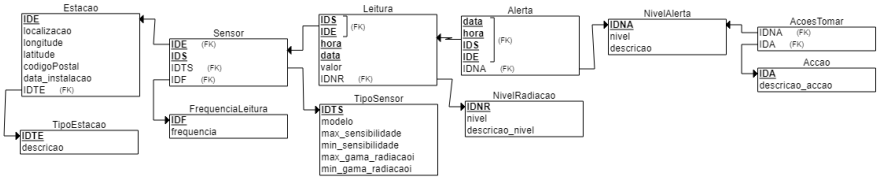

Figura 3.2: Esquema Relacional.

1. Estação (**IDE**, Localizacao, Longitude, Latitude, CodigoPostal, DataInstalação, IDTE (FK))
2. TipoEstacao (**IDTE**, descricao)
3. Sensor (**IDE** (FK), **IDS**, IDTS (FK), IDF (FK))
4. TipoSensor (**IDTS**, modelo, min__ensibilidade, max_sensibilidade, min_gama_radiaçao, max_gama_radiaçao)
5. FrequenciaLeitura (**IDF**, frequencia)
6. Leitura (**IDE** (FK), **IDS** (FK), **data**, **hora**, IDNR (FK))
7. NívelRadiacao (**IDNR**, nivel, descrição_nivel)
8. Alerta (**IDE** (FK), **IDS** (FK), **data**, **hora**, IDNA (FK))
9. NivelAlerta (**IDNA**, nivel, descrição_nivel)
10. AccoesTomar (IDNA (FK), IDA (FK))
11. Accao (**IDA**, descrição_accao)

Após a criação desse modelo "completo", foi então possível avançar no desenvolvimento da aplicação, pois já havia uma base sólida para a construção da base de dados e da aplicação.

3.3 Considerações

De modo a automatizar alguns processos na base de dados foi necessário recorrer a procedimentos e gatilhos.

3.3.1 Procedimento de inserção

Desde o início foi idealizado que seria necessário criar um procedimento (***InsertRadiationReading***) que fornecesse um método estruturado e automatizado para a inserção de leituras de radiação na base de dados, associando-as aos sensores apropriados e armazenando-as para posterior análise e uso. Este procedimento aceita como parâmetros o valor recebido da leitura de radiação, a data e hora da leitura, bem como o nível de radiação. Ele começa por determinar o IDTS (ID de Tipo de Sensor) com base no valor recebido, verificando em que intervalo de radiação este valor se encontra na tabela TipoSensor. Em seguida, procura pelo IDS (ID de Sensor) correspondente ao IDTS na tabela Sensor. Se encontrar um IDS válido, insere os dados da leitura na tabela Leitura, associando-os ao sensor identificado. Caso contrário, gera uma mensagem informando que o IDS do sensor não foi encontrado ou que o valor recebido não está dentro do intervalo de radiação de nenhum sensor, permitindo assim uma gestão ou registo adequado destas situações no sistema. Este procedimento oferece um método estruturado para a inserção de leituras de radiação, garantindo a associação correta dos dados com os sensores apropriados na base de dados.

```
PROCEDURE InsertRadiationReading(
    @received_value DECIMAL(10, 2),
    @reading_date varchar(20),
    @reading_time varchar(20),
    @radiation_level int(1)
)
AS
BEGIN
    DECLARE @sensor_IDS INT;

    -- Encontra o sensor a receber os dados dado a gama de radiacao da
        leitura
    DECLARE @IDTS INT;
    SELECT @IDTS = IDTS
    FROM TipoSensor
    WHERE @received_value BETWEEN min_gama_radiacao AND
        max_gama_radiacao;

    IF @IDTS IS NOT NULL
    BEGIN
        SELECT @sensor_IDS = IDS
        FROM Sensor
        WHERE IDTS = @IDTS;

        IF @sensor_IDS IS NOT NULL
        BEGIN
```

```
            INSERT INTO Leitura (IDS, IDE, dia, hora, valor, IDNR)
            VALUES (@sensor_IDS, @IDTS, @reading_date, @reading_time,
                @received_value, @radiation_level);
        END
        ELSE
        BEGIN
            PRINT 'Sensor IDS nao encontrado';
        END
    END
    ELSE
    BEGIN
        PRINT 'Valor nao se enquadra em nehuma gama de radiacao';
    END
END;
```

Excerto de Código 3.1: Procedimento InsertRadiationReading

3.3.2 Gatilho de análise de Leitura

Desde o início, foi pensado que seria necessário criar um gatilho (***trg_Leitura_AfterInsert***) para analisar cada leitura a ser inserida. Nesse sentido, o gatilho identifica o valor inserido e realiza uma comparação com leituras anteriores dos últimos seis meses para o mesmo sensor. Caso o novo valor exceda 10% do valor máximo das leituras anteriores, um alerta é gerado na tabela **Alerta** e o parâmetro **IDF** do sensor correspondente é atualizado. Se o novo valor estiver abaixo de 80% do valor máximo recente e for menor que um valor de referência, o parâmetro **IDF** é ajustado para cima. Este gatilho tem como objetivo monitorar e ajustar dinamicamente os parâmetros do sensor com base nas novas leituras inseridas na tabela, contribuindo para um controle adaptativo e automatizado do sistema de monitoramento de radiação.

```
TRIGGER trg_Leitura_AfterInsert
ON [dbo].[Leitura]
AFTER INSERT
AS
BEGIN
    DECLARE @IDNR INT

    -- encontra o IDNR
    SELECT TOP 1 @IDNR = IDNR
    FROM inserted

    -- altera o IDF do sensor consoante o valor lido e as leituras
        anteriores
    DECLARE @CurrentValue FLOAT

    SELECT @CurrentValue = valor
```

```
    FROM inserted

    DECLARE @CurrentIDF INT

    DECLARE @IDNA INT

    SELECT @CurrentIDF = MAX(IDF)
    FROM [dbo].[Sensor]
    WHERE IDS = (SELECT IDS FROM inserted)
        AND IDE = (SELECT IDE FROM inserted)

    DECLARE @LastSixMonthsMaxValue FLOAT

    -- Descobro o valor medio das leituras dos ultimos 6 meses
    SELECT @LastSixMonthsMaxValue = AVG(valor)
    FROM [dbo].[Leitura]
    WHERE CONVERT(DATE, dia, 103) >= DATEADD(MONTH, DATEDIFF(MONTH, 0,
        GETDATE()) - 6, 0)
        AND CONVERT(DATE, dia, 103) < DATEADD(MONTH, DATEDIFF(MONTH, 0,
            GETDATE()), 0)
        AND IDE = (SELECT IDE FROM inserted)  -- Filter by the inserted
            IDE

        PRINT 'Last Six Months Max Value: ' + CAST(
            @LastSixMonthsMaxValue AS NVARCHAR(20))
PRINT 'CurrentValue: ' + CAST(@CurrentValue AS NVARCHAR(20))
PRINT 'CurrentValue: ' + CAST(@LastSixMonthsMaxValue * 1.1 AS NVARCHAR
    (20))

    --Verifica se o valor e maior que 10% das leituras medias dos 6
        meses anteriores e Nivel Radiacao > 1
    IF (@CurrentValue > (@LastSixMonthsMaxValue * 1.1) AND @IDNR > 1)
    BEGIN
        PRINT 'Adding Alerta for exceeding 10% threshold'

        select @IDNA =
            CASE
                WHEN @IDNR < 4 THEN 1
                WHEN @IDNR >= 4 AND @IDNR < 6 THEN 2
                WHEN @IDNR >= 6 THEN 3
            END;

        INSERT INTO [dbo].[Alerta] (IDS, IDE, dia, hora, IDNA)
        SELECT IDS, IDE, dia, hora, @IDNA
        FROM inserted;

        -- Decresce o IDF se o valor da leitura for maior que 10%
        IF @CurrentIDF > 1
        BEGIN
```

```
            PRINT 'Decreasing IDF by 1'
            UPDATE [dbo].[Sensor]
            SET IDF = @CurrentIDF - 1
            WHERE IDS = (SELECT IDS FROM inserted)
                AND IDE = (SELECT IDE FROM inserted)
        END
        ELSE
        BEGIN
            PRINT 'Setting IDF to 1 (minimum)'
            UPDATE [dbo].[Sensor]
            SET IDF = 1
            WHERE IDS = (SELECT IDS FROM inserted)
                AND IDE = (SELECT IDE FROM inserted)
        END
    END
    ELSE
    BEGIN
        DECLARE @HighValue FLOAT

        SELECT @HighValue = MAX(valor)
        FROM [dbo].[Leitura]
        WHERE IDE = (SELECT IDE FROM inserted)
            AND valor > @LastSixMonthsMaxValue

        PRINT 'High Value: ' + CAST(@HighValue AS NVARCHAR(20))
        PRINT '@CurrentValue: ' + CAST(@CurrentValue AS NVARCHAR(20))
        PRINT '(@HighValue * 0.8): ' + CAST(@HighValue * 0.8 AS NVARCHAR
            (20))

        -- aumenta o IDF se o valor da leitura for menor que 20%
        IF @CurrentValue < (@HighValue * 0.8) AND @HighValue IS NOT NULL
        BEGIN
            PRINT 'Updating IDF based on lower threshold'

            IF @CurrentIDF < 3
            BEGIN
                PRINT 'Increasing IDF by 1'
                UPDATE [dbo].[Sensor]
                SET IDF = @CurrentIDF + 1
                WHERE IDS = (SELECT IDS FROM inserted)
                    AND IDE = (SELECT IDE FROM inserted)
            END
            ELSE
            BEGIN
                PRINT 'Setting IDF to 3 (maximum)'
                UPDATE [dbo].[Sensor]
                SET IDF = 3
                WHERE IDS = (SELECT IDS FROM inserted)
                    AND IDE = (SELECT IDE FROM inserted)
```

```
                END
            END
        END
END
```

Excerto de Código 3.2: Gatilho $\text{trg}_{Leitura_{A}fterInsert}$

Capítulo

4

Implementação e Testes

4.1 Introdução

Este capítulo vai abordar questões mais concretas no que diz respeito à solução desenvolvida. Irá abordar mais especificamente, o acesso à base de dados, e todas as funcionalidades da aplicação desenvolvida.

4.2 Acesso à base de dados

Para aceder à base de dados pelo ***Python***, teve de ser instalado localmente um *driver* de modo a permitir a conexão entre o servidor e a base de dados ***SQL***.

4.3 Interface e Funcionalidades

```
def database_connection():
    server = '192.168.0.12,1600'
    username = ''
    password = ''
    database_name = 'gammawatch'
    db_connection_string = "Driver={ODBC Driver 17 for SQL Server};Server="
    + server + ";Database=" + database_name + ";UID=" + username + ";PWD=" + password + ";"
```

Figura 4.1: PYODBC

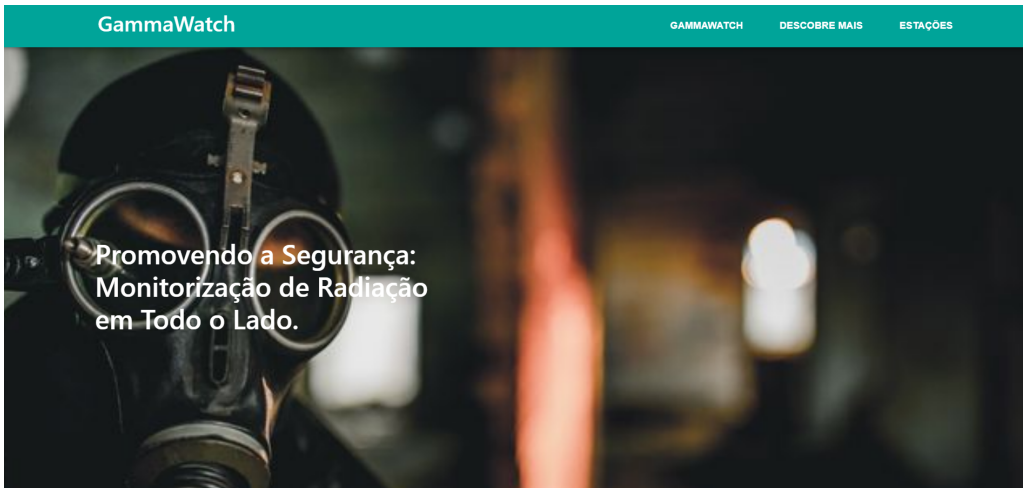

Figura 4.2: Página principal

Figura 4.3: Página principal (2)

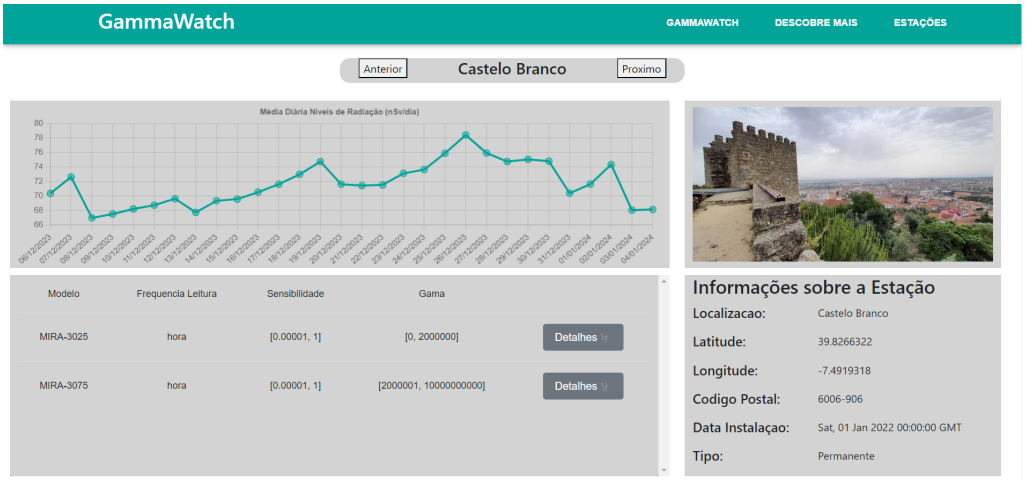

Figura 4.4: Página da Estação.

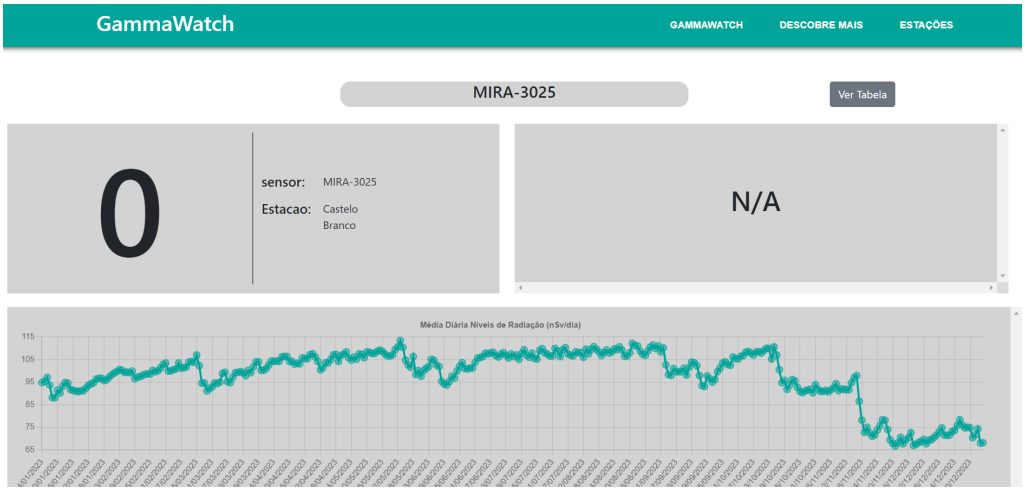

Figura 4.5: Página do Sensor.

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões

Este projeto permitiu adquirir um melhor conhecimento acerca da linguagem de base de dados SQL e da arquitetura cliente/servidor, como também permitiu evoluir em temas recorrentes na Unidade Curricular de Base de Dados. Foi uma excelente maneira de aplicar o conhecimento ensinado na parte teórica desta cadeira.

O projeto encorajou também uma constante pesquisa, no que diz respeito à busca de soluções para problemas que surgiram, fazendo com que o autor se deparasse com várias abordagens para o mesmo problema escolhendo a que achava mais adequada. Promovendo também o exercicio de raciocinio.

5.2 Trabalho Futuro

Atualmente, o projeto atende às necessidades básicas, no entanto, existe margem para melhorias. É crucial otimizar tanto o procedimento quanto o gatilho para reduzir os seus tempos de execução e melhorar a eficiência do sistema. Além disso, para evoluir o projeto, poderia ser considerada a implementação de novas funcionalidades e interações. Por exemplo, expandir a capacidade do sistema para gerir estações e sensores através de um sistema de contas de utilizador com diferentes níveis de permissão. Estas medidas ajudariam a aprimorar o sistema.

5.3 Apêndices

```
USE gammawatch;

CREATE TABLE TipoEstacao
(
  IDTE INT NOT NULL IDENTITY(1,1),
  descricao VARCHAR(255) NOT NULL,
  PRIMARY KEY (IDTE)
);

CREATE TABLE TipoSensor
(
  IDTS INT NOT NULL IDENTITY(1,1),
  modelo  VARCHAR(60) NOT NULL,
  max_sensibilidade FLOAT NOT NULL,
  min_sensibilidade FLOAT NOT NULL,
  min_gama_radiacao FLOAT NOT NULL,
  max_gama_radiacao FLOAT NOT NULL,
  PRIMARY KEY (IDTS)
);

CREATE TABLE NivelRadiacao
(
  IDNR INT NOT NULL,
  descricao_nivel INT NOT NULL,
  PRIMARY KEY (IDNR)
);

CREATE TABLE NivelAlerta
(
  IDNA INT NOT NULL IDENTITY(1,1),
  nivel INT NOT NULL,
  descricao VARCHAR(255) NOT NULL,
  PRIMARY KEY (IDNA)
);

CREATE TABLE Accao
(
  IDA INT NOT NULL IDENTITY(1,1),
  descricao_accao VARCHAR(255) NOT NULL,
  PRIMARY KEY (IDA)
);

CREATE TABLE FreqLeitura
(
  IDF INT NOT NULL IDENTITY(1,1),
  frequencia VARCHAR(60) NOT NULL,
  PRIMARY KEY (IDF)
);
```

```
CREATE TABLE Estacao
(
  IDE INT NOT NULL IDENTITY(1,1),
  localizacao VARCHAR(20) NOT NULL,
  latitude VARCHAR(20) NOT NULL,
  longitude VARCHAR(20) NOT NULL,
  codigoPostal VARCHAR(8) NOT NULL,
  data_instalacao DATE NOT NULL,
  tipo INT NOT NULL,
  PRIMARY KEY (IDE),
  FOREIGN KEY (tipo) REFERENCES TipoEstacao(IDTE)
);

CREATE TABLE Sensor
(
  IDS INT NOT NULL IDENTITY(1,1),
  IDE INT NOT NULL,
  IDTS INT NOT NULL,
  IDF INT NOT NULL,
  PRIMARY KEY (IDS, IDE),
  FOREIGN KEY (IDE) REFERENCES Estacao(IDE),
  FOREIGN KEY (IDTS) REFERENCES TipoSensor(IDTS),
  FOREIGN KEY (IDF) REFERENCES FreqLeitura(IDF)
);

CREATE TABLE Leitura
(
  IDS INT NOT NULL,
  IDE INT NOT NULL,
  dia VARCHAR(20) NOT NULL,
  hora VARCHAR(20) NOT NULL,
  valor FLOAT NOT NULL,
  IDNR INT NOT NULL,
  PRIMARY KEY (IDS, IDE, dia, hora),
  FOREIGN KEY (IDS, IDE) REFERENCES Sensor(IDS, IDE),
  FOREIGN KEY (IDNR) REFERENCES NivelRadiacao(IDNR)
);

CREATE TABLE Alerta
(
  IDS INT NOT NULL,
  IDE INT NOT NULL,
  dia INT NOT NULL,
  hora INT NOT NULL,
  IDNA INT NOT NULL,
  FOREIGN KEY (IDS, IDE, dia, hora) REFERENCES Leitura(IDS, IDE, dia,
      hora),
  FOREIGN KEY (IDNA) REFERENCES NivelAlerta(IDNA)
```

```
);

CREATE TABLE AcoesTomar
(
  IDNA INT NOT NULL,
  IDA INT NOT NULL,
  FOREIGN KEY (IDNA) REFERENCES NivelAlerta(IDNA),
  FOREIGN KEY (IDA) REFERENCES Accao(IDA)
);
```

Excerto de Código 5.1: Script Criação BD