Documentacion WEB

1

Generado por Doxygen 1.12.0

1. Biometría y Medio Ambiente - Web

Este proyecto es una plataforma web diseñada para el monitoreo de datos biométricos y ambientales. Permite la visualización y análisis de estos datos en tiempo real a través de una interfaz web amigable. Está construida utilizando tecnologías modernas como PHP para el backend, JavaScript para el frontend y Docker para la gestión de contenedores.

1.1. Tecnologías Utilizadas

- **Backend:** PHP
- **Frontend:** JavaScript, HTML, CSS
- **Contenedores:** Docker, Docker Compose
- **Gestor de dependencias:** npm
- **Base de datos:** MySQL

1.2. Instalación y Configuración

1.2.1. Prerrequisitos

1. **Docker**: Asegúrate de tener Docker instalado en tu sistema. `Instalar Docker`.
2. **Node.js y npm**: Instala `Node.js` y npm para gestionar las dependencias del frontend.

1.2.2. Instalación

1. **Clona el repositorio**:
   ```
   git clone https://github.com/SentoMarcos/Biometr-a-y-Medio-Ambiente-Docker-Web-DB.git
   ```
2. **Accede al directorio del proyecto**:
   ```
   cd Biometr-a-y-Medio-Ambiente-Docker-Web-DB
   ```
3. **Instala las dependencias del frontend**: Ejecuta el siguiente comando para instalar las dependencias necesarias:
   ```
   npm install
   ```
4. **Construye y ejecuta los contenedores con Docker**: Usa Docker Compose para construir y levantar todos los servicios:
   ```
   docker-compose up --build
   ```
5. **Accede a la aplicación**: Una vez que los contenedores estén en funcionamiento, puedes acceder a la aplicación web en tu navegador en la URL: `http://localhost:8000` (o el puerto especificado en el archivo `docker-compose.yml`).

1.2.3. Configuración de Variables de Entorno

Crea un archivo `.env` en la raíz del proyecto con las variables de entorno necesarias para la configuración de la base de datos y otros servicios. Aquí tienes un ejemplo:

```
DB_HOST=db
DB_PORT=3306
DB_USER=root
DB_PASSWORD=password
DB_NAME=biometria
```

Estas variables de entorno serán utilizadas por Docker para la configuración de los servicios internos.

1.3. Autores

- SentoMarcos

1.4. Proyectos Relacionados

- Biometría y Medio Ambiente - Android
- Biometría y Medio Ambiente - Arduino

2. Índice de clases

2.1. Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

3. Índice de archivos

3.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

4. Documentación de clases

4.1. Referencia de la interface Botón

la última medición

Diagrama de colaboración de Botón:

4.1.1. Descripción detallada

la última medición

Interfaz para obtener la última medición registrada en la base de datos.

La documentación de esta interface está generada del siguiente archivo:

- medicionesgetter.js

4.2. Referencia de la interface Texto

la última medición

Diagrama de colaboración de Texto:

4.2.1. Descripción detallada

la última medición

Interfaz para mostrar la última medición registrada en la base de datos.

La documentación de esta interface está generada del siguiente archivo:

- medicionesgetter.js

5. Documentación de archivos

5.1. Referencia del archivo README.md

5.2. Referencia del archivo index.js

Servidor web para el proyecto de Biometría y Medio Ambiente.

Variables

- import express from express
- import { createPool } from 'mysql2/promise'
- import cors from cors
- import path from path
- const app = express()

  Crear la aplicación de Express.
- const pool

  Crear el pool de conexiones con MySQL.
- const __filename = fileURLToPath(import.meta.url)

  Se redirige la ruta principal a "/index.html".
- const __dirname = path.dirname(__filename)

5.2.1. Descripción detallada

Servidor web para el proyecto de Biometría y Medio Ambiente.

Definición en el archivo index.js.

5.2.2. Documentación de variables

__dirname

```
const __dirname = path.dirname(__filename)
```

Definición en la línea 49 del archivo index.js.

__filename

```
const __filename = fileURLToPath(import.meta.url)
```

Se redirige la ruta principal a "/index.html".

Se redirige la ruta principal a "/index.html" para que al ingresar a la dirección raíz de la aplicación se cargue el archivo "index.html" de la carpeta "public".

Definición en la línea 48 del archivo index.js.

app

```
const app = express()
```

Crear la aplicación de Express.

Se crea la aplicación de Express y se configura para usar JSON y CORS

Definición en la línea 15 del archivo index.js.

cors

```
import cors from cors
```

Definición en la línea 7 del archivo index.js.

express

```
import express from express
```

Definición en la línea 5 del archivo index.js.

import

```
import { createPool } from 'mysql2/promise'
```

Definición en la línea 6 del archivo index.js.

path

```
import path from path
```

Definición en la línea 8 del archivo index.js.

pool

```
const pool
```

Valor inicial:

```
= createPool({
    host: 'mysqldb',
    user: 'root',
    password: '123456',
    database: 'mydb',
    port: 3306,
})
```

Crear el pool de conexiones con MySQL.

Se crea el pool de conexiones con MySQL y se configura para conectarse a la base de datos "my db" en el host "mysqldb" con el usuario "root" y la contraseña "123456".

Parámetros

host	- El host de la base de datos.
user	- El usuario de la base de datos.
password	- La contraseña de la base de datos.
database	- El nombre de la base de datos.
port	- El puerto de la base de datos.

Definición en la línea 32 del archivo index.js.

5.3. index.js

Ir a la documentación de este archivo.

```
00001
00005 import express from 'express';
00006 import { createPool } from 'mysql2/promise';
00007 import cors from 'cors';
00008 import path from 'path';
00009 import { fileURLToPath } from 'url';
00010
00015 const app = express();
00016 // Use the express.json() middleware to parse JSON bodies
00017 app.use(express.json());
00018 // Usar middleware CORS y Morgan
00019 app.use(cors());
00020
00031 // Crear el pool de conexiones con MySQL
00032 const pool = createPool({
00033     host: 'mysqldb',  // Cambiar a 'localhost' o '127.0.0.1' si es necesario
00034     user: 'root',
00035     password: '123456',
00036     database: 'mydb',  // Asegúrate de que el nombre de la base de datos esté correcto
00037     port: 3306,
00038 });
00039
00040 // Servir archivos estáticos desde la carpeta "public"
00041 app.use(express.static('public'));
00042
00048 const __filename = fileURLToPath(import.meta.url);
00049 const __dirname = path.dirname(__filename);
00050
00051 // Servir archivos estáticos desde la carpeta "public"
00052 app.use(express.static(path.join(__dirname, 'public')));
00053
00054 // Redirigir a index.html
00055 app.get('/', (req, res) => {
00056     res.sendFile(path.join(__dirname, 'public', 'index.html'));
00057 });
00072 app.get('/ping', async (req, res) => {
00073     try {
00074         const [result] = await pool.query('SELECT NOW()');
00075         res.json(result);
00076     } catch (error) {
00077         console.error('Error al hacer ping a la base de datos:', error.message);
00078         res.status(500).send('Error en la base de datos');
00079     }
00080 });
00081
00098 // Ruta para obtener la última medición
00099 app.get('/ultima-medicion', async (req, res) => {
00100     try {
00101         const [result] = await pool.query('SELECT * FROM medidas ORDER BY fecha DESC LIMIT 1');
00102         if (result.length === 0) {
00103             return res.status(404).json({ error: 'No se encontraron mediciones' });
00104         }
00105         res.json(result[0]); // Devuelve el primer resultado
00106     } catch (error) {
00107         console.error('Error al obtener la última medición:', error.message);
00108         res.status(500).send('Error en la base de datos');
00109     }
00110 });
00130 app.post('/insertar', async (req, res) => {
00131     try {
00132         const { Lugar, Gas, Valor } = req.body; // Obtenemos Lugar, Gas, y Valor del body
00133
00134         if (!Lugar || !Gas || !Valor) {
```

```
00135             return res.status(400).send('Faltan datos obligatorios');
00136         }
00137
00138         // Generamos la fecha actual
00139         const fecha = new Date();
00140
00141         // Insertamos la medición en la base de datos
00142         const [result] = await pool.query(
00143             'INSERT INTO medidas (fecha, Lugar, Gas, Valor) VALUES (?, ?, ?, ?)',
00144             [fecha, Lugar, Gas, Valor]
00145         );
00146
00147         res.status(201).send('Medición insertada correctamente');
00148     } catch (error) {
00149         console.error('Error al insertar la medición:', error.message);
00150         res.status(500).send('Error en el servidor');
00151     }
00152 });
00153
00154
00155 // Iniciar el servidor en el puerto 3000
00156 app.listen(3000, () => {
00157     console.log('Server running at http://localhost:3000/');
00158 });
```

5.4. Referencia del archivo GetMediciones.php

Variables

- global $conn
- $sql = "SELECT * FROM medidas ORDER BY id DESC LIMIT 10"
- $result = mysqli_query($conn, $sql)
- if(mysqli_num_rows($result) > 0) else

5.4.1. Documentación de variables

$conn

```
global $conn
```

Definición en la línea 4 del archivo GetMediciones.php.

$result

```
$result = mysqli_query($conn, $sql)
```

Definición en la línea 8 del archivo GetMediciones.php.

$sql

```
$sql = "SELECT * FROM medidas ORDER BY id DESC LIMIT 10"
```

Definición en la línea 6 del archivo GetMediciones.php.

else

```
if (mysqli_num_rows( $result) > 0) else
```

Valor inicial:

```
{
    echo "0 results"
```

Definición en la línea 16 del archivo GetMediciones.php.

5.5. GetMediciones.php

Ir a la documentación de este archivo.

```
00001 <?php
00002 // obtener mediciones
00003 require_once '../Conexion/DBConexion.php';
00004 global $conn;
00005
00006 $sql = "SELECT * FROM medidas ORDER BY id DESC LIMIT 10";
00007
00008 $result = mysqli_query($conn, $sql);
00009
00010 if (mysqli_num_rows($result) > 0) {
00011     $mediciones = array();
00012     while ($row = mysqli_fetch_assoc($result)) {
00013         $mediciones[] = $row;
00014     }
00015     echo json_encode($mediciones);
00016 } else {
00017     echo "0 results";
00018 }
00019 ?>
```

5.6. Referencia del archivo DBConexion.php

Archivo de conexión a la base de datos.

Variables

- $host = 'localhost'
- $usuario = 'root'
- $password = '123456'
- $dbname = 'mydb'
- $port = 3306
- $conn = mysqli_connect($host, $usuario, $password, $dbname, $port)

5.6.1. Descripción detallada

Archivo de conexión a la base de datos.

Este archivo contiene la conexión a la base de datos MySQL

Versión

1.0

Parámetros

string	*$host*	Nombre del host de la base de datos
string	*$usuario*	Nombre del usuario de la base de datos
string	*$password*	Contraseña del usuario de la base de datos
string	*$dbname*	Nombre de la base de datos
int	*$port*	Puerto de la base de datos

Definición en el archivo DBConexion.php.

5.6.2. Documentación de variables

$conn

```
$conn = mysqli_connect($host, $usuario, $password, $dbname, $port)
```

Definición en la línea 20 del archivo DBConexion.php.

$dbname

```
$dbname = 'mydb'
```

Definición en la línea 16 del archivo DBConexion.php.

$host

```
$host = 'localhost'
```

Definición en la línea 13 del archivo DBConexion.php.

$password

```
$password = '123456'
```

Definición en la línea 15 del archivo DBConexion.php.

$port

```
$port = 3306
```

Definición en la línea 17 del archivo DBConexion.php.

$usuario

```
$usuario = 'root'
```

Definición en la línea 14 del archivo DBConexion.php.

5.7. DBConexion.php

Ir a la documentación de este archivo.

```
00001 <?php
00013 $host = 'localhost'; // Use the container name
00014 $usuario = 'root';  // Ajusta esto según tu usuario MySQL
00015 $password = '123456'; // Ajusta esto según tu contraseña
00016 $dbname = 'mydb';
00017 $port = 3306; // Puerto como entero, sin comillas
00018
00019 // Conexión a MySQL
00020 $conn = mysqli_connect($host, $usuario, $password, $dbname, $port);
00021
00022
00023 ?>
```

5.8. Referencia del archivo medicionesgetter.js

Funciones para obtener mediciones de la base de datos y mostrarlas en la página web.

Variables

- const getUltimaMedicion

  @function getUltimaMedicion
- const boton = document.getElementById('btn-ultima-medicion')
- const texto = document.getElementById('texto-medicion')

5.8.1. Descripción detallada

Funciones para obtener mediciones de la base de datos y mostrarlas en la página web.

Este archivo contiene funciones para obtener mediciones de la base de datos y mostrarlas en la página web.

Definición en el archivo medicionesgetter.js.

5.8.2. Documentación de variables

boton

```
const boton = document.getElementById('btn-ultima-medicion')
```

Definición en la línea 31 del archivo medicionesgetter.js.

getUltimaMedicion

```
const getUltimaMedicion
```

Valor inicial:

```
= async () => {
    try {
        const response = await fetch('/ultima-medicion');
        if (!response.ok) {
            throw new Error('Error al obtener la última medición');
        }
        return await response.json();
    } catch (error) {
        console.error('Error al obtener la última medición:', error);
        return null;
    }
}
```

@function getUltimaMedicion

Obtiene la última medición registrada en la base de datos.

Realiza una petición GET a la ruta "/ultima-medicion" para obtener la última medición registrada en la base de datos.

Devuelve

La última medición registrada en la base de datos.

Excepciones

Error	si hay un error al obtener la última medición.

Definición en la línea 14 del archivo medicionesgetter.js.

texto

```
const texto = document.getElementById('texto-medicion')
```

Definición en la línea 36 del archivo medicionesgetter.js.

5.9. medicionesgetter.js

Ir a la documentación de este archivo.

```
00001
00014 const getUltimaMedicion = async () => {
00015     try {
00016         const response = await fetch('/ultima-medicion');
00017         if (!response.ok) {
00018             throw new Error('Error al obtener la última medición');
00019         }
00020         return await response.json();
00021     } catch (error) {
00022         console.error('Error al obtener la última medición:', error);
00023         return null;
00024     }
00025 };
00030 //darle al boton y cambiar el texto h1 por la ultima medicion
00031 const boton = document.getElementById('btn-ultima-medicion');
00036 const texto = document.getElementById('texto-medicion');
00042 boton.addEventListener('click', async () => {
00043     const medicion = await getUltimaMedicion();
00044     if (medicion) {
00045         texto.textContent = `Ultima medicion: ${medicion.fecha} - Lugar: ${medicion.Lugar} - Gas:
      ${medicion.Gas} - Valor: ${medicion.Valor}`;
00046     } else {
00047         texto.textContent = 'Error al obtener la última medición';
00048     }
00049 } );
```

6. Ejemplos

6.1. GET/ping

@function GET /ping

@function GET /ping Ruta para verificar la conexión a la base de datos (ping)

Se crea una ruta "/ping" que devuelve la fecha y hora actuales de la base de datos para verificar que la conexión está funcionando correctamente.

Devuelve

La fecha y hora actuales de la base de datos.

Excepciones

Error	500 si hay un error al hacer ping a la base de datos.

{ "0": { "NOW()": "2021-10-20T00:00:00.000Z" } }

6.2. GET/ultima-medicion

@function GET /ultima-medicion

@function GET /ultima-medicion Ruta para obtener la última medición

Se crea una ruta "/ultima-medicion" que devuelve la última medición registrada en la base de datos.

Devuelve

La última medición registrada en la base de datos.

Excepciones

Error	404 si no se encontraron mediciones.
Error	500 si hay un error en la base de datos.

{ "fecha": "2021-10-20T00:00:00.000Z", "Lugar": "Laboratorio", "Gas": "CO2", "Valor": 400 }

6.3. POST/insertar

@function POST /insertar

@function POST /insertar Ruta para insertar una medición

Se crea una ruta "/insertar" que permite insertar una nueva medición en la base de datos.

Parámetros

Lugar	- El lugar de la medición.
Gas	- El tipo de gas medido.
Valor	- El valor de la medición.

Devuelve

Un mensaje de éxito o error en la inserción.

Excepciones

Error	400 si faltan datos obligatorios.
Error	500 si hay un error en el servidor.

{ "Fecha": "2021-10-20T00:00:00.000Z", "Lugar": "Laboratorio", "Gas": "CO2", "Valor": 400 }

Índice alfabético