

ALSE 2023

# PROYECTO

---

Santiago Cortes Tovar

Juan Andres Abella Ballen

# INTRODUCCIÓN

README

## PROYECTO\_ALSE

Proyecto desarrollado por Santiago Cortes Tovar y Juan Andrés Abella Ballen, utilizando C/C++, SQLite, Git, GitHub, CMake y Qt5.

Nota: Este proyecto tiene como objetivo utilizar C/C++, SQLite, Git, GitHub, CMake y Qt5.

### Descripción del Proyecto

Se ha creado una clase base llamada "Sensor" que será heredada por los siguientes 6 sensores. Estos sensores comparten atributos y métodos similares.

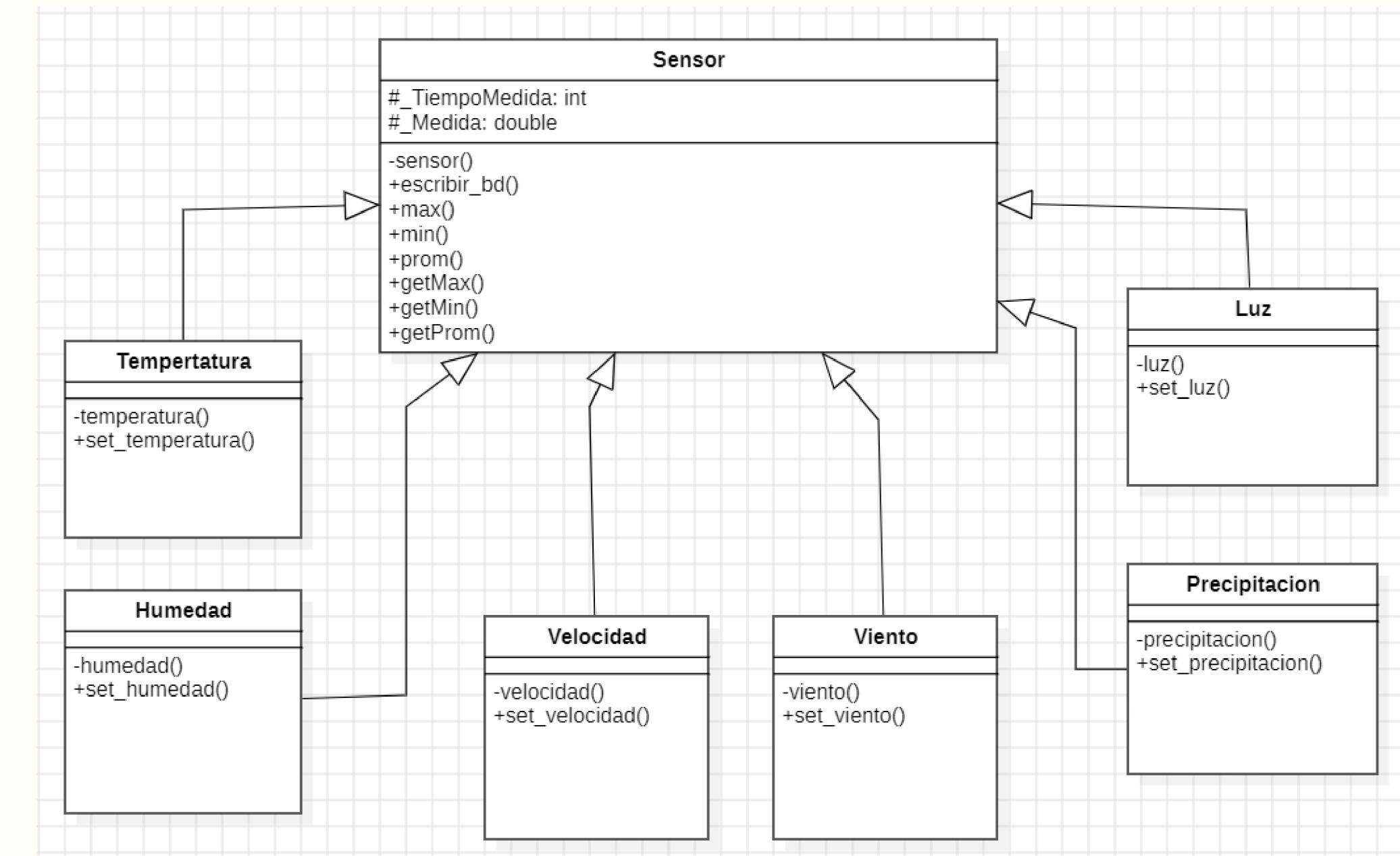
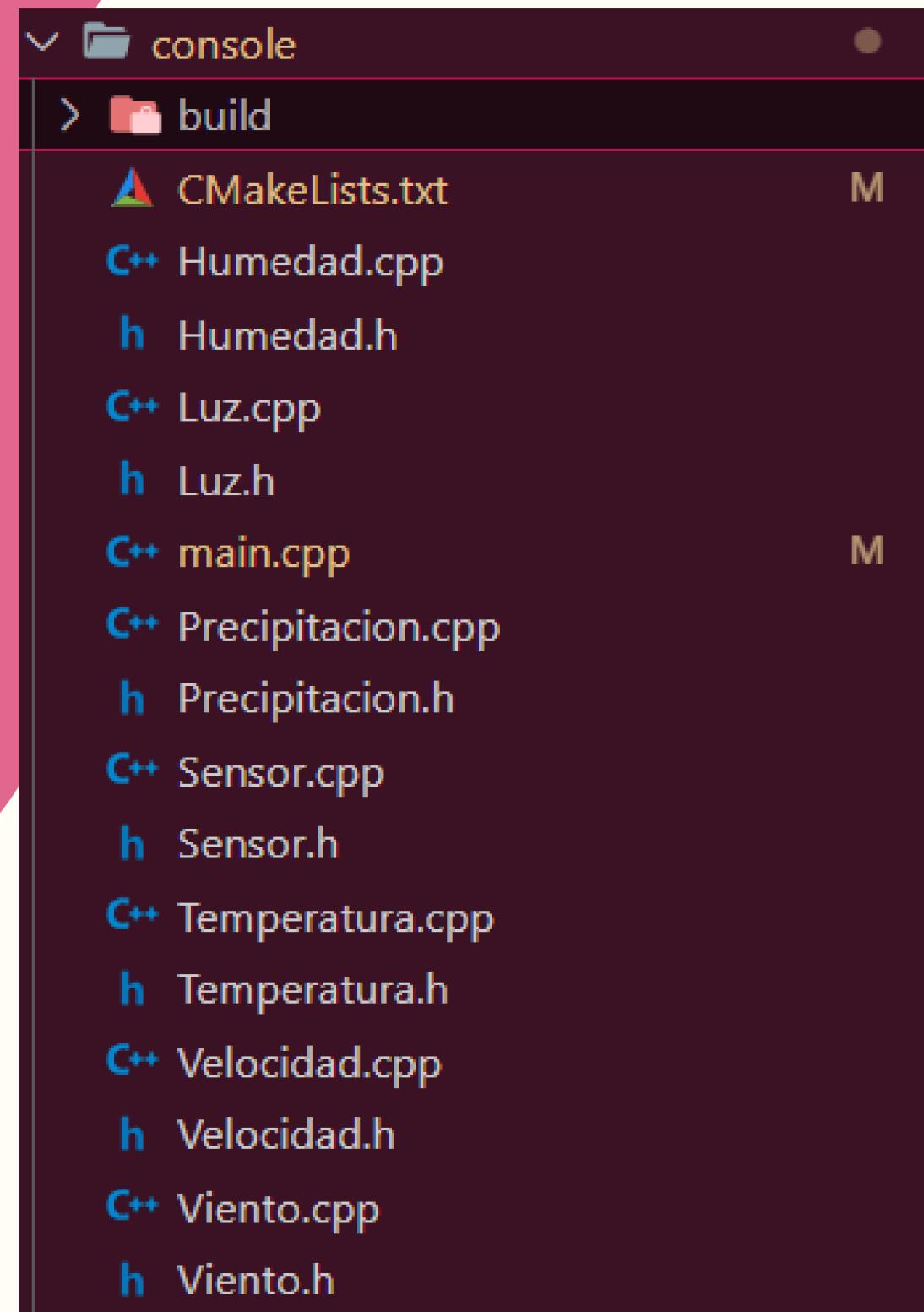
Importante: A continuación se describen los sensores y sus respectivos rangos de medición.

#### Sensores

1. Temperatura: Rango de medición: 10°C a 45°C.
2. Humedad: Rango de medición: 0% a 100%.
3. Velocidad: Rango de medición: 0 m/s a 40 m/s.
4. Dirección del Viento: Rango de medición: -180° a 180° (Norte).
5. Precipitación: Rango de medición: 0 mm a 50 mm.
6. Intensidad de Luz: Rango de medición: 0 a 2000 lúmenes.

Advertencia: Todas las mediciones se realizarán a intervalos regulares y se almacenarán en una base de datos junto con la fecha de captura de los datos.

# ESTRUCTURA



```
/* crear la tabla de sensores */
rc = sqlite3_open("db_name.db", &db);
// verificar si se abrio la base de datos
if (rc != 0)
{
    fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
    return 1;
}
else
{
    fprintf(stdout, "Opened database successfully\n\n");
}
// creamos la tabla sesores en la base de datos db:name.db
sqlstr = "CREATE TABLE IF NOT EXISTS sensores_pc (ID INTEGER PRIMARY KEY AUTOINCREMENT, SENSOR TEXT NOT NULL,"
          "MINIMO INTEGER NOT NULL, PROMEDIO INTEGER NOT NULL,"
          "MAXIMO INTEGER NOT NULL, FECHA TEXT NOT NULL);"

rc = sqlite3_exec(db, sqlstr.c_str(), 0, 0, &zErrMsg);
```

```

// crear los sensores
vector<float> temp; // vector que almacena las temperaturas
vector<float> hum; // vector que almacena las humedades
vector<float> vel; // vector que almacena las velocidades
vector<float> viento; // vector que almacena los vientos
vector<float> prec; // vector que almacena las precipitaciones
vector<float> luz; // vector que almacena las luces

```

```

float minTemp = Sensor::min(60 / ts, temp); // minimo de las tem
float maxTemp = Sensor::max(60 / ts, temp); // maximo de las tem
float promTemp = Sensor::prom(60 / ts, temp); // promedio de las te
float minHum = Sensor::min(60 / ts, hum); // minimo de las hume
float maxHum = Sensor::max(60 / ts, hum); // maximo de las hume
float promHum = Sensor::prom(60 / ts, hum); // promedio de las hu
float minVel = Sensor::min(60 / ts, vel); // minimo de las vele
float maxVel = Sensor::max(60 / ts, vel); // maximo de las vele
float promVel = Sensor::prom(60 / ts, vel); // promedio de las ve
float minViento = Sensor::min(60 / ts, viento); // minimo de los vie
float maxViento = Sensor::max(60 / ts, viento); // maximo de los vie
float promViento = Sensor::prom(60 / ts, viento); // promedio de los v
float minPrec = Sensor::min(60 / ts, prec); // minimo de las pre
float maxPrec = Sensor::max(60 / ts, prec); // maximo de las pre
float promPrec = Sensor::prom(60 / ts, prec); // promedio de las pre
float minLuz = Sensor::min(60 / ts, luz); // minimo de las luces
float maxLuz = Sensor::max(60 / ts, luz); // maximo de las luces
float promLuz = Sensor::prom(60 / ts, luz); // promedio de las lu

```

```

// llenar los sensores
for (int i = 0; i < (60 / ts); i++)
{
    Temperatura t;
    t.set_temperatura();
    temp.push_back(t.get_temperatura());
    Humedad h;
    h.set_humedad();
    hum.push_back(h.get_humedad());
    Velocidad v;
    v.set_velocidad();
    vel.push_back(v.get_velocidad());
    Viento vi;
    vi.set_viento();
    viento.push_back(vi.get_viento());
    Precipitacion p;
    p.set_precipitacion();
    prec.push_back(p.get_precipitacion());
    Luz l;
    l.set_luz();
    luz.push_back(l.get_luz());
}

/*
 * // verificacion de los sensores
 cout << "Temperatura " << i << ":" << temp[i] << endl;
 cout << "Humedad " << i << ":" << hum[i] << endl;
 cout << "Velocidad " << i << ":" << vel[i] << endl;
 cout << "Viento " << i << ":" << viento[i] << endl;
 cout << "Precipitacion " << i << ":" << prec[i] << endl;
 cout << "Luz " << i << ":" << luz[i] << endl;
*/
}

```

```
// verificacion de los datos
cout << "Tiempo de muestreo: " << ts << ".Numero de muestreos: " << 60 / ts << endl;
cout << "Fecha: " << fecha << "\n\n";
// verificacion de los datos
cout << "Temperatura: " << minTemp << " " << promTemp << " " << maxTemp << endl;
cout << "Humedad: " << minHum << " " << promHum << " " << maxHum << endl;
cout << "Velocidad: " << minVel << " " << promVel << " " << maxVel << endl;
cout << "Viento: " << minViento << " " << promViento << " " << maxViento << endl;
cout << "Precipitacion: " << minPrec << " " << promPrec << " " << maxPrec << endl;
cout << "Luz: " << minLuz << " " << promLuz << " " << maxLuz << endl;
// insertar los datos en la base de datos
sql = "INSERT INTO sensores_pc (SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) "
    "VALUES ('Temperatura', " +
        to_string(minTemp) + ", " + to_string(promTemp) + ", " + to_string(maxTemp) + ", '" + fecha + "'); "
"INSERT INTO sensores_pc (SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) "
"VALUES ('Humedad', " +
        to_string(minHum) + ", " + to_string(promHum) + ", " + to_string(maxHum) + ", '" + fecha + "'); "
"INSERT INTO sensores_pc (SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) "
"VALUES ('Velocidad', " +
        to_string(minVel) + ", " + to_string(promVel) + ", " + to_string(maxVel) + ", '" + fecha + "'); "
"INSERT INTO sensores_pc (SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) "
"VALUES ('Viento', " +
        to_string(minViento) + ", " + to_string(promViento) + ", " + to_string(maxViento) + ", '" + fecha + "'); "
"INSERT INTO sensores_pc (SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) "
"VALUES ('Precipitacion', " +
        to_string(minPrec) + ", " + to_string(promPrec) + ", " + to_string(maxPrec) + ", '" + fecha + "'); "
"INSERT INTO sensores_pc (SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) "
"VALUES ('Luz', " +
        to_string(minLuz) + ", " + to_string(promLuz) + ", " + to_string(maxLuz) + ", '" + fecha + "'); ";

/* Execute SQL statement */
rc = sqlite3_exec(db, sql.c_str(), 0, 0, &zErrMsg);
if (rc != SQLITE_OK)
{
    fprintf(stderr, "SQL error: %s\n", zErrMsg);
    sqlite3_free(zErrMsg);
    return 1;
}
else
{
    fprintf(stdout, "Records created successfully\n");
}
```

```

// Conectar a la base de datos remota
if (mysql_real_connect(connection, HOST, USER, PASSWD, DB, 0, NULL, 0) == NULL)
{
    cout << "\n\nFailed to connect to database: " << mysql_error(connection) << endl;
    return 1;
}
else
{
    cout << "\n\nConnected to database successfully" << endl;
}

// Obtener los datos de la tabla sensores de la base de datos local sensores_pc
sql = "SELECT * FROM sensores_pc";
rc = sqlite3_exec(db, sql.c_str(), 0, 0, &zErrMsg);
sqlite3_prepare_v2(db, sql.c_str(), -1, &statement, NULL); // preparar la consulta

// Insertar los datos en la tabla sensores de la base de datos en la nube
while (sqlite3_step(statement) == SQLITE_ROW) // obtener los datos de la consulta fila por fila
{
    int id = sqlite3_column_int(statement, 0); // obtiene el id de cada fila
    string sensor = reinterpret_cast<const char *>(sqlite3_column_text(statement, 1)); // se guarda el sensor
    float minimo = static_cast<float>(sqlite3_column_double(statement, 2)); // se guarda el minimo
    float promedio = static_cast<float>(sqlite3_column_double(statement, 3)); // se guarda el promedio
    float maximo = static_cast<float>(sqlite3_column_double(statement, 4)); // se guarda el maximo
    string fecha = reinterpret_cast<const char *>(sqlite3_column_text(statement, 5)); // se guarda la fecha
    // insertar los datos en la base de datos en la nube
    string insertQuery = "INSERT IGNORE INTO sensores (ID, SENSOR, MINIMO, PROMEDIO, MAXIMO, FECHA) VALUES ('" + to_string(id) + "','" +
    " | | | | " + sensor + "', '" + to_string(minimo) + "', '" + to_string(promedio) + "', '" +
    " | | | | " + to_string(maximo) + "', '" + fecha + "')";
    if (mysql_query(connection, insertQuery.c_str()) != 0)
    {
        fprintf(stderr, "Failed to insert data into cloud database: %s\n", mysql_error(connection));
        return 1;
    }
}

cout << "\n\nBase de datos copiada exitosamente a la nube" << endl;

```

```

#include <mysql/mysql.h>

using namespace std;

MYSQL *connection = mysql_init(NULL);
int query_state;

#define HOST "db4free.net"      // host de la base de datos
#define USER "cortua_abellita" // usuario de la base de datos
#define PASSWD "cortua_abellita" // contraseña de la base de datos
#define DB "base_datos_alse"   // nombre de la base de datos

```

Type Here

TEMPERATURA EN C°

mm	mm	mm
mm	mm	mm

HUMEDAD EN %

mm	mm	mm
mm	mm	mm

VELOCIDAD EN m/s

mm	mm	mm
mm	mm	mm

PRECIPITACION EN mm

mm	mm	mm
mm	mm	mm

DIRECCION DEL VIENTO EN ° NORTE

mm	mm	mm
mm	mm	mm

INTENSIDAD DE LUZ EN LUMENES

mm	mm	mm
mm	mm	mm

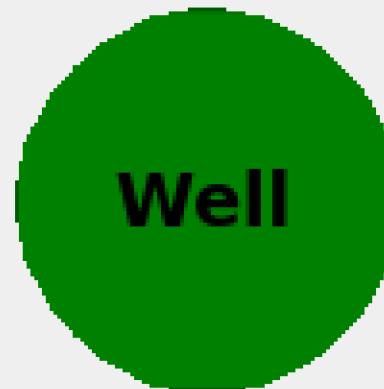
# MainWindow

TEMPERATURA EN C°

-7

1

30



HUMEDAD EN %

0

13.6667

81



VELOCIDAD EN m/s

0

11.6667

28

PRECIPITACION EN mm

16

15.6667

31

DIRECCION DEL VIENTO EN ° NORTE

-175

40.6667

168

INTENSIDAD DE LUZ EN LUMENES

29

26.3333

50





Reciente Favoritas

Nueva

base\_datos\_alse

Nueva

sensores

db4free\_sys

information\_schema

performance\_schema

Servidor: MySQL 8.2 Server:3306 » Base de datos: base\_datos\_alse » Tabla: sensores

Examinar Estructura SQL Buscar Insertar Exportar Importar

	ID	SENSOR	MINIMO	PROMEDIO	MAXIMO	FECHA
<input type="checkbox"/>	1	Temperatura	-10	16	42	2023-12-09 17:08:25
<input type="checkbox"/>	2	Humedad	1	26	85	2023-12-09 17:08:25
<input type="checkbox"/>	3	Velocidad	0	8	29	2023-12-09 17:08:25
<input type="checkbox"/>	4	Viento	-169	-82	-4	2023-12-09 17:08:25
<input type="checkbox"/>	5	Precipitacion	2	23	36	2023-12-09 17:08
<input type="checkbox"/>	6	Luz	20	47	94	2023-12-09 17:08
<input type="checkbox"/>	7	Temperatura	-10	7	35	2023-12-09 17:08
<input type="checkbox"/>	8	Humedad	33	48	81	2023-12-09 17:08
<input type="checkbox"/>	9	Velocidad	5	13	28	2023-12-09 17:08
<input type="checkbox"/>	10	Viento	-97	5	137	2023-12-09 17:08
<input type="checkbox"/>	11	Precipitacion	2	16	44	2023-12-09 17:08
<input type="checkbox"/>	12	Luz	5	32	68	2023-12-09 17:08
<input type="checkbox"/>	13	Temperatura	-6	24	43	2023-12-09 17:09
<input type="checkbox"/>	14	Humedad	37	62	100	2023-12-09 17:09
<input type="checkbox"/>	15	Velocidad	0	6	34	2023-12-09 17:09
<input type="checkbox"/>	16	Viento	-125	31	171	2023-12-09 17:09
<input type="checkbox"/>	17	Precipitacion	19	18	34	2023-12-09 17:09
<input type="checkbox"/>	18	Luz	46	53	83	2023-12-09 17:09
<input type="checkbox"/>	19	Temperatura	-6	24	40	2023-12-11 13:34
<input type="checkbox"/>	20	Humedad	3	43	98	2023-12-11 13:34
<input type="checkbox"/>	21	Velocidad	1	18	37	2023-12-11 13:34
<input type="checkbox"/>	22	Viento	-164	-2	167	2023-12-11 13:34
<input type="checkbox"/>	23	Precipitacion	5	25	48	2023-12-11 13:34:32
<input type="checkbox"/>	24	Luz	2	48	99	2023-12-11 13:34:32

ID	SENSOR	MINIMO	PROMEDIO	MAXIMO	FECHA
1	Temperatura	-10	15.8	42	2023-12-09 ...
2	Humedad	1	25.6	85	2023-12-09 ...
3	Velocidad	0	7.8	29	2023-12-09 ...
4	Viento	-169	-81.800003	-4	2023-12-09 ...
5	Precipitacion	2	23	36	2023-12-09 ...
6	Luz	20	47	94	2023-12-09 ...
7	Temperatura	-10	6.8	35	2023-12-09 ...
8	Humedad	33	48	81	2023-12-09 ...
9	Velocidad	5	13.4	28	2023-12-09 ...
10	Viento	-97	4.8	137	2023-12-09 ...
11	Precipitacion	2	15.8	44	2023-12-09 ...
12	Luz	5	31.799999	68	2023-12-09 ...
13	Temperatura	-6	24.333334	43	2023-12-09 ...
14	Humedad	37	62.333332	100	2023-12-09 ...
15	Velocidad	0	6.333333	34	2023-12-09 ...
16	Viento	-125	31	171	2023-12-09 ...

```
MatrixXd MainWindow::abrir_db()
{
    MatrixXd R(6, 3);

    db = QSqlDatabase::addDatabase("QSQLITE"); // agrega la base de datos a qt
    db.setDatabaseName("../db_name.db"); // crea una conexión con la base de datos
    if (db.open()) // abre la base de datos
    {
        qDebug() << "Base de datos abierta";
    }
    else
    {
        qDebug() << "Error al abrir la base de datos";
    }

    QString consulta = "SELECT MINIMO, PROMEDIO, MAXIMO FROM (SELECT ID, MINIMO, PROMEDIO, MAXIMO FROM sensores_pc ORDER BY ID DESC LIMIT 6) ORDER BY ID ASC";
    QSqlQuery query(db); // Asigna la conexión a la consulta
    if (query.prepare(consulta))
    {
        if (query.exec())
        {
            qDebug() << "Consulta realizada";
        }
        else
        {
            qDebug() << "Error al ejecutar la consulta";
        }
    }
    else
    {
        qDebug() << "Error al preparar la consulta";
    }

    for (int i = 0; i < 6; i++)
    {
        query.next(); // Avanza a la siguiente fila
        R.row(i) << query.value(0).toDouble(), query.value(1).toDouble(), query.value(2).toDouble(); // Asigna los valores de la fila a la matriz
    }
    db.close(); // Cierra la conexión
    return R;
}
```

**i GRACIAS!**

