Gestore di Rilevazioni Sensori

A cura di:
Vincenzo Puca MAT 297113
Marco Alrazem MAT 298913

Scopo

Si intende sviluppare un applicativo che gestisca i dati di alcuni sensori luminosità-temperatura, i quali, per finalità di progetto, saranno simulati; si potrà gestire anche i sensori stessi.

Nello specifico verranno istanziati alcuni dati prodotti dai sensori in modo randomico (**INSERT**) tramite apposito pulsante nell'interfaccia utente.

Sarà possibile poi creare un grafico con i dati raccolti (**QUERY SELECT**), modificare o aggiungere specifici record manualmente (**INSERT-DELETE-UPDATE**), o eliminare completamente l'intera tabella dati (**TRUNCATE**), il tutto tramite query sql mandate direttamente al gestore del database (**mariaDB**) mediante modulo **Connect.py**.

L'applicativo è scritto completamente in linguaggio Python, l'installazione e l'istanza del server è posta su server Raspberry Pi (gestito tramite **SSH**), pertanto è possibile accedere alla banca dati dell'applicativo soltanto se si è posti nella stessa rete del server (Infrastruttura **LAN**).

L'applicazione sarà gestita attraverso un numero predefinito di amministratori.

Non sarà prevista l'aggiunta di ulteriori utenti. La creazione degli amministratori viene effettuata durante la prima fase dell'installazione del server tramite SSH.

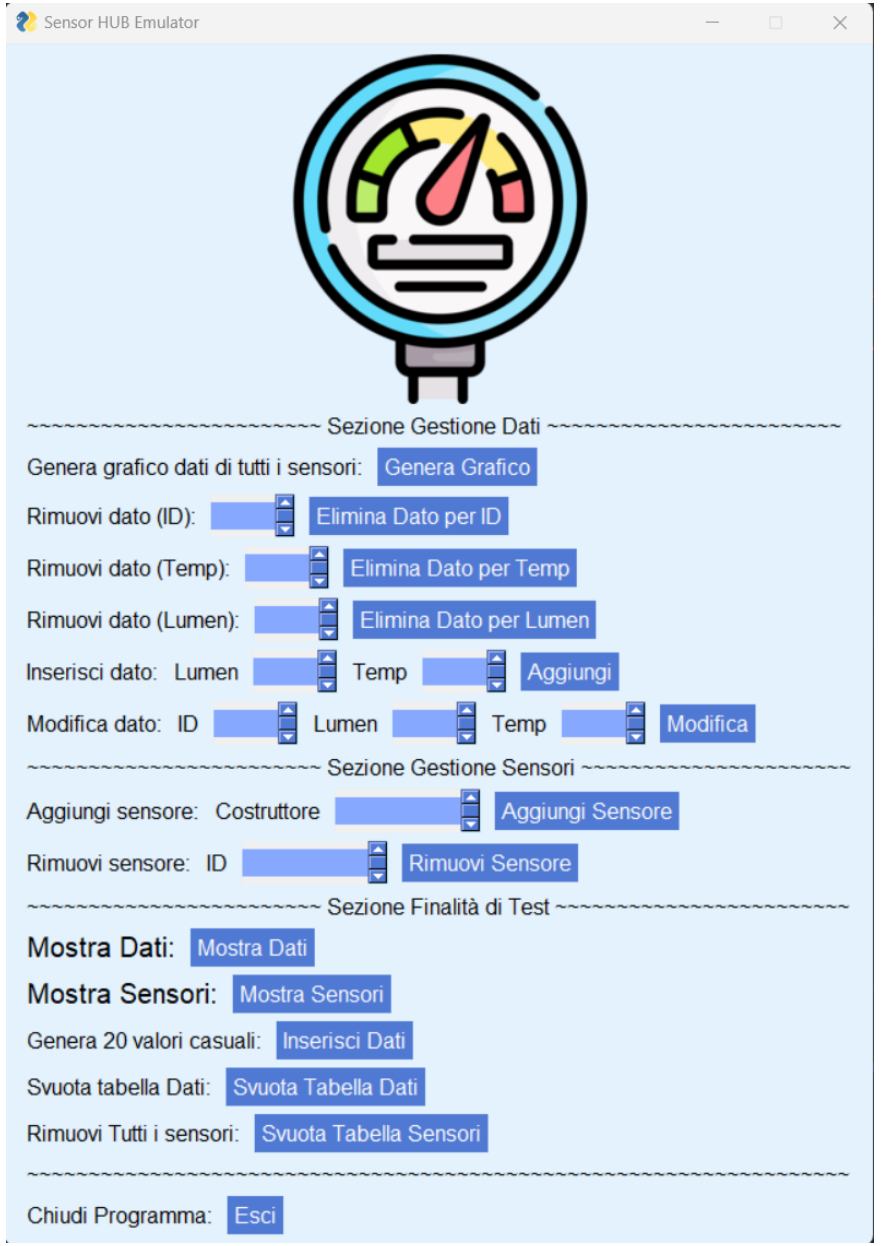
Struttura Applicativo

Login

L'applicativo viene eseguito tramite **GUI** locata in MainGUI.py, al lancio dell'applicazione verrà visualizzata la schermata login nella quale sarà obbligatorio per l'utente inserire il proprio ID e l'ID del sensore che vorrà gestire, nella home-page poi, sarà possibile eseguire il comando scelto tramite interfaccia.

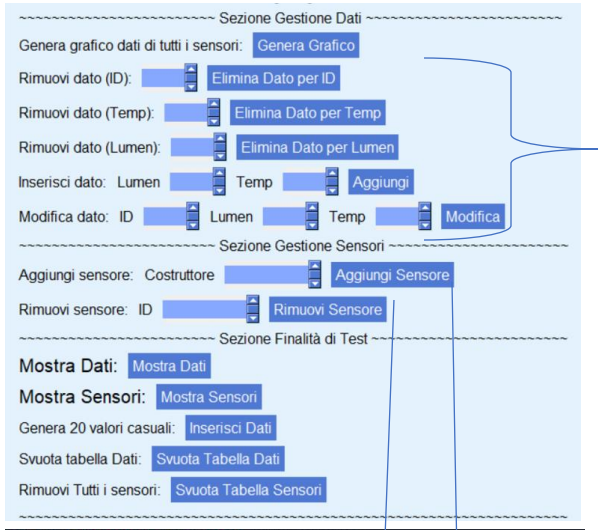
Alla pressione dell'apposito pulsante, infatti, sarà possibile: vedere lo **stato del database** in tempo reale, **creare un grafico** a partire dai dati nel database stesso o **modificare/aggiungere** qualsiasi dato a piacimento.

Il tutto attraverso **GUI** che richiamerà le apposite query locate nel file connect.py.



Home Page

Struttura Applicativo



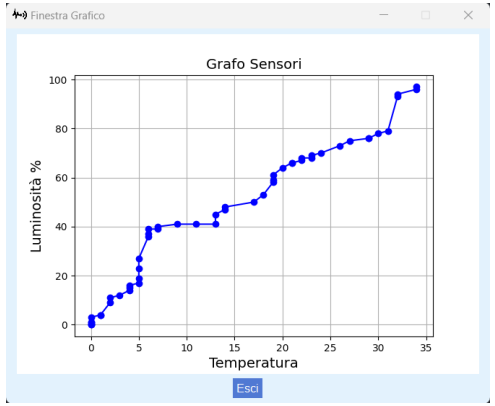

```python
def AddSens(Curs,ConnToDB,SensProd): #add a new sensor
  try:
   Curs.execute("INSERT INTO Sensore (Produttore) VALUES (?)", (SensProd,))
  except mariadb.Error as e:
   print(f"Errore inserimento: {e}")

  ConnToDB.commit()

def RemoveSensID(Curs,ConnToDB,IDSens): #delete one record from given idsens (textbox)
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("DELETE FROM Sensore WHERE IDSensore = ?",(IDSens,))
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione dato dal DB: {e}")

  ConnToDB.commit()
```

```python
def InjectInfo(Curs,ConnToDB): #inject 20 records into the DB (pseudo-random)

datarand = datetime.date.today()
for x in range(20):
  datotemp = random.randint(0,35)
  datolum = random.randint(0,100)
  try:
   Curs.execute("INSERT INTO Dato (Temperatura,LuminositàPerc) VALUES (?, ?)", (datotemp,datolum))
  except mariadb.Error as e:
   print(f"Errore Aggiunta dati al DB: {e}")

ConnToDB.commit()

def PurgeRecords(Curs,ConnToDB): #remove all records into the DB
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("TRUNCATE TABLE Dato")
   Curs.execute("TRUNCATE TABLE Gestione")
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione dati dal DB: {e}")
  ConnToDB.commit()

def PurgeAllSensors(Curs,ConnToDB): #remove all sensors into the DB
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("TRUNCATE TABLE Sensore")
   Curs.execute("TRUNCATE TABLE Produzione")
   Curs.execute("TRUNCATE TABLE Installazione")
   Curs.execute("TRUNCATE TABLE Dato")
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione sensori dal DB: {e}")
  ConnToDB.commit()
```

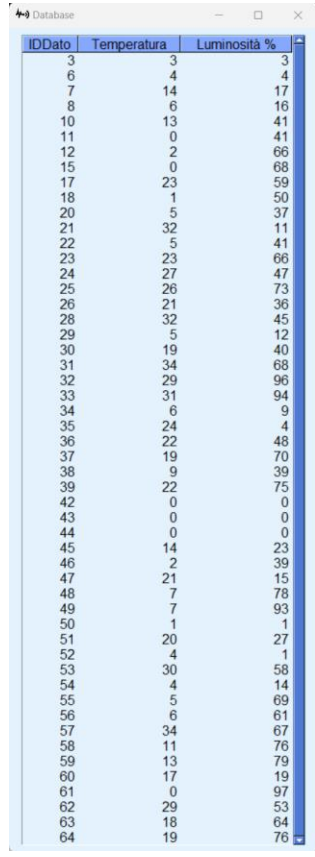
```python
def DeleteOneFromID(Curs,ConnToDB,ID): #delete one record from given id (textbox)
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("DELETE FROM Dato WHERE IDDato = ?",(ID,))
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione dato dal DB: {e}")

  ConnToDB.commit()

def DeleteOneFromTemp(Curs,ConnToDB,Temp): #delete one record from given temp (textbox)
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("DELETE FROM Dato WHERE Temperatura = ?",(Temp,))
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione dato dal DB: {e}")

  ConnToDB.commit()

def DeleteOneFromLumen(Curs,ConnToDB,Lumen): #delete one record from given Lumen (textbox)
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("DELETE FROM Dato WHERE LuminositàPerc = ?",(Lumen,))
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione dato dal DB: {e}")

  ConnToDB.commit()

def InjectInfoTempLumen(Curs,ConnToDB,temp,lumen): #inject info with temp lumen given
  try:
   Curs.execute("INSERT INTO Dato (Temperatura,LuminositàPerc) VALUES (?, ?)", (temp,lumen))
  except mariadb.Error as e:
   print(f"Errore inserimento: {e}")

  ConnToDB.commit()

def EditData(Curs,ConnToDB,id,temp,lumen): #edit data from given id, than edit values lumen and temp
  try:
   Curs.execute("UPDATE Dato SET Temperatura = ?, LuminositàPerc = ? WHERE IDDato = ?", (temp,lumen,id))
  except mariadb.Error as e:
   print(f"Errore modifica: {e}")

  ConnToDB.commit()
```

Sezione finalità di test

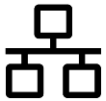
```python
def GetSensData(Curs): #get output data return lists tipo + produttore
    riga=0
    produttore=[]
    idsensore=[]

    #getting Produttore value
    riga=0
    try:
     Curs.execute("SELECT Produttore FROM Sensore GROUP BY IDSensore")
     for riga in Curs:
      produttore.extend(riga)
    except mariadb.Error as e:
      print(f"Errore recupero dati dal DB: {e}")

    #getting IDSensore value
    riga=0
    try:
     Curs.execute("SELECT IDSensore FROM Sensore GROUP BY IDSensore")
     for riga in Curs:
      idsensore.extend(riga)
    except mariadb.Error as e:
      print(f"Errore recupero dati dal DB: {e}")

    return produttore,idsensore
```



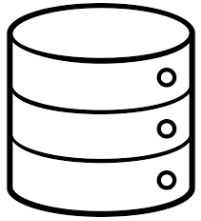

```python
def GetData(Curs): #get output data return double lists tempdata + lumendata
    riga=0
    tempdata=[]
    lumendata=[]
    IDRecord=[]

    #getting temperatura value
    try:
     Curs.execute("SELECT Temperatura FROM Dato GROUP BY IDDato")
     for riga in Curs:
      tempdata.extend(riga)
    except mariadb.Error as e:
      print(f"Errore recupero dati dal DB: {e}")

    #getting percentuale luminosità value
    riga=0
    try:
     Curs.execute("SELECT LuminositàPerc FROM Dato GROUP BY IDDato")
     for riga in Curs:
      lumendata.extend(riga)
    except mariadb.Error as e:
      print(f"Errore recupero dati dal DB: {e}")

    riga=0
    try:
     Curs.execute("SELECT IDDato FROM Dato GROUP BY IDDato")
     for riga in Curs:
      IDRecord.extend(riga)
    except mariadb.Error as e:
      print(f"Errore recupero dati dal DB: {e}")

    return tempdata,lumendata,IDRecord
```




```python
def SendLogin(Curs,ConnToDB,adminID,sensorID):
  try:
   Curs.execute("CREATE OR REPLACE TRIGGER WhoAddData after insert ON dato FOR EACH ROW insert into gestione (IDDato,IDAmministratore)select (select IDDato f
   Curs.execute("CREATE OR REPLACE TRIGGER WhoAddedSensor after insert ON sensore FOR EACH ROW insert into installazione (IDSensore,IDAmministratore) select
   Curs.execute("CREATE OR REPLACE TRIGGER WhatSensorAddedData after insert ON dato FOR EACH ROW insert into produzione (IDSensore,IDDato) select (select IDS
  except mariadb.Error as e:
   print(f"Errore modifica trigger: {e}")

  ConnToDB.commit()
```



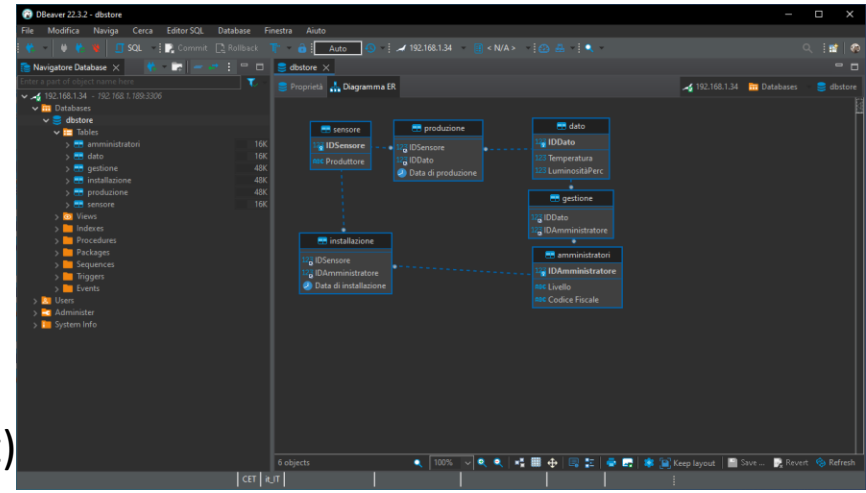
Raspberry Pi

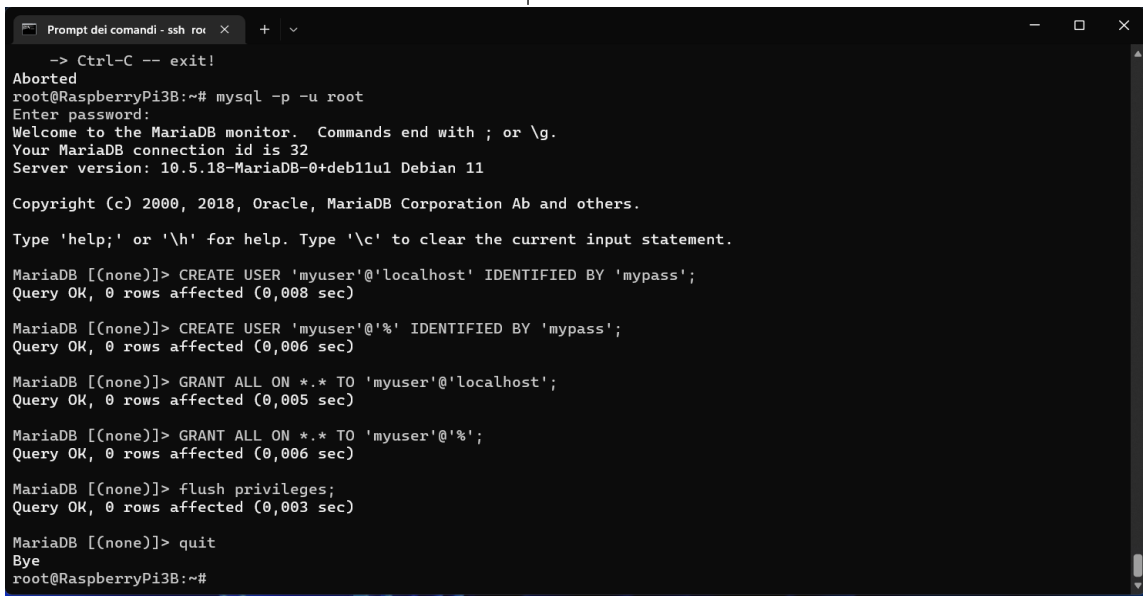


Server:
Database LAN gestito da mariadb
IP LAN 192.168.1.190
(Connect.py)

DBeaver

DBeaver, utilizzato per la gestione del **database** (DBStore), contenente tabelle e campi, impostazione di **chiave primaria/esterna** e dei vari **vincoli** (NOT Null) (Int) (varchar) (Incremental)





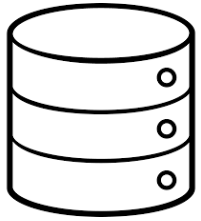
Processo di **creazione database** e **utente gestore**

Connessione da client al server ssh:
ssh numeutente@192.168.1.189

- In allegato log file dei passaggi installazione **mariadb** e creazione utente

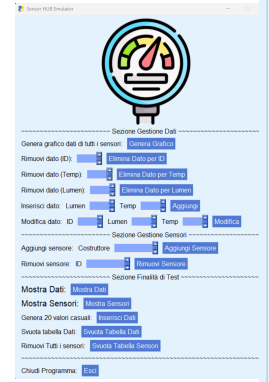
Sezione Server

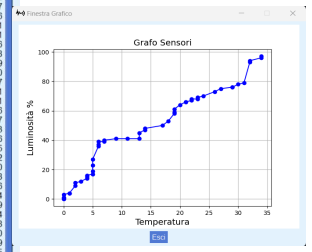
Raspberry Pi



LAN

Client Sensor hub emulator (**MainGUI.py**)





Connect.py
Gestore **query** e **connessione server**

```python
def SendLogin(Curs,ConnToDB,adminID,sensorID):
 try:
  Curs.execute("CREATE OR REPLACE TRIGGER WhoAddData after insert ON dato FOR EACH ROW insert into gestione (IDDato,IDAmministratore)select (select IDDato fro
  Curs.execute("CREATE OR REPLACE TRIGGER WhoAddedSensor after insert ON sensore FOR EACH ROW insert into installazione (IDSensore,IDAmministratore) select (s
  Curs.execute("CREATE OR REPLACE TRIGGER WhatSensorAddedData after insert ON dato FOR EACH ROW insert into produzione (IDSensore,IDDato) select (select IDSer
 except mariadb.Error as e:
  print(f"Errore modifica trigger: {e}")

 ConnToDB.commit()
```

```python
def DeleteOneFromID(Curs,ConnToDB,ID): #delete one record from given id (textbox)
 try:
  Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
  Curs.execute("DELETE FROM Dato WHERE IDDato = ?",(ID,))
  Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
 except mariadb.Error as e:
  print(f"Errore eliminazione dato dal DB: {e}")

 ConnToDB.commit()

def DeleteOneFromTemp(Curs,ConnToDB,Temp): #delete one record from given temp (textbox)
 try:
  Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
  Curs.execute("DELETE FROM Dato WHERE Temperatura = ?",(Temp,))
  Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
 except mariadb.Error as e:
  print(f"Errore eliminazione dato dal DB: {e}")

 ConnToDB.commit()

def DeleteOneFromLumen(Curs,ConnToDB,Lumen): #delete one record from given Lumen (textbox)
 try:
  Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
  Curs.execute("DELETE FROM Dato WHERE LuminositàPerc = ?",(Lumen,))
  Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
 except mariadb.Error as e:
  print(f"Errore eliminazione dato dal DB: {e}")

 ConnToDB.commit()

def InjectInfoTempLumen(Curs,ConnToDB,temp,lumen): #inject info with temp lumen given
 try:
  Curs.execute("INSERT INTO Dato (Temperatura,LuminositàPerc) VALUES (?, ?)", (temp,lumen))
 except mariadb.Error as e:
  print(f"Errore inserimento: {e}")

 ConnToDB.commit()

def EditData(Curs,ConnToDB,id,temp,lumen): #edit data from given id, than edit values lumen and temp
 try:
  Curs.execute("UPDATE Dato SET Temperatura = ?, LuminositàPerc = ? WHERE IDDato = ?", (temp,lumen,id))
 except mariadb.Error as e:
  print(f"Errore modifica: {e}")

 ConnToDB.commit()
```

```python
def GetSensData(Curs): #get output data return lists tipo + produttore
 riga=0
 produttore=[]
 idsensore=[]

 #getting Produttore value
 riga=0
 try:
  Curs.execute("SELECT Produttore FROM Sensore GROUP BY IDSensore")
  for riga in Curs:
   produttore.extend(riga)
 except mariadb.Error as e:
   print(f"Errore recupero dati dal DB: {e}")

 #getting IDSensore value
 riga=0
 try:
  Curs.execute("SELECT IDSensore FROM Sensore GROUP BY IDSensore")
  for riga in Curs:
   idsensore.extend(riga)
 except mariadb.Error as e:
   print(f"Errore recupero dati dal DB: {e}")

 return produttore,idsensore
```

```python
def GetCursorConn(): #connect to mariaDB return cursor + conn var
 try:
  ConnToDB = mariadb.connect(
   user="myuser",
   password="mypass",
   host="192.168.1.190",
   port=3306,
   database="DBStore")
 except mariadb.Error as e:
   print(f"Errore di connessione a MariaDB Platform: {e}")
   sys.exit(1)

 Curs = ConnToDB.cursor()
 return Curs,ConnToDB
```

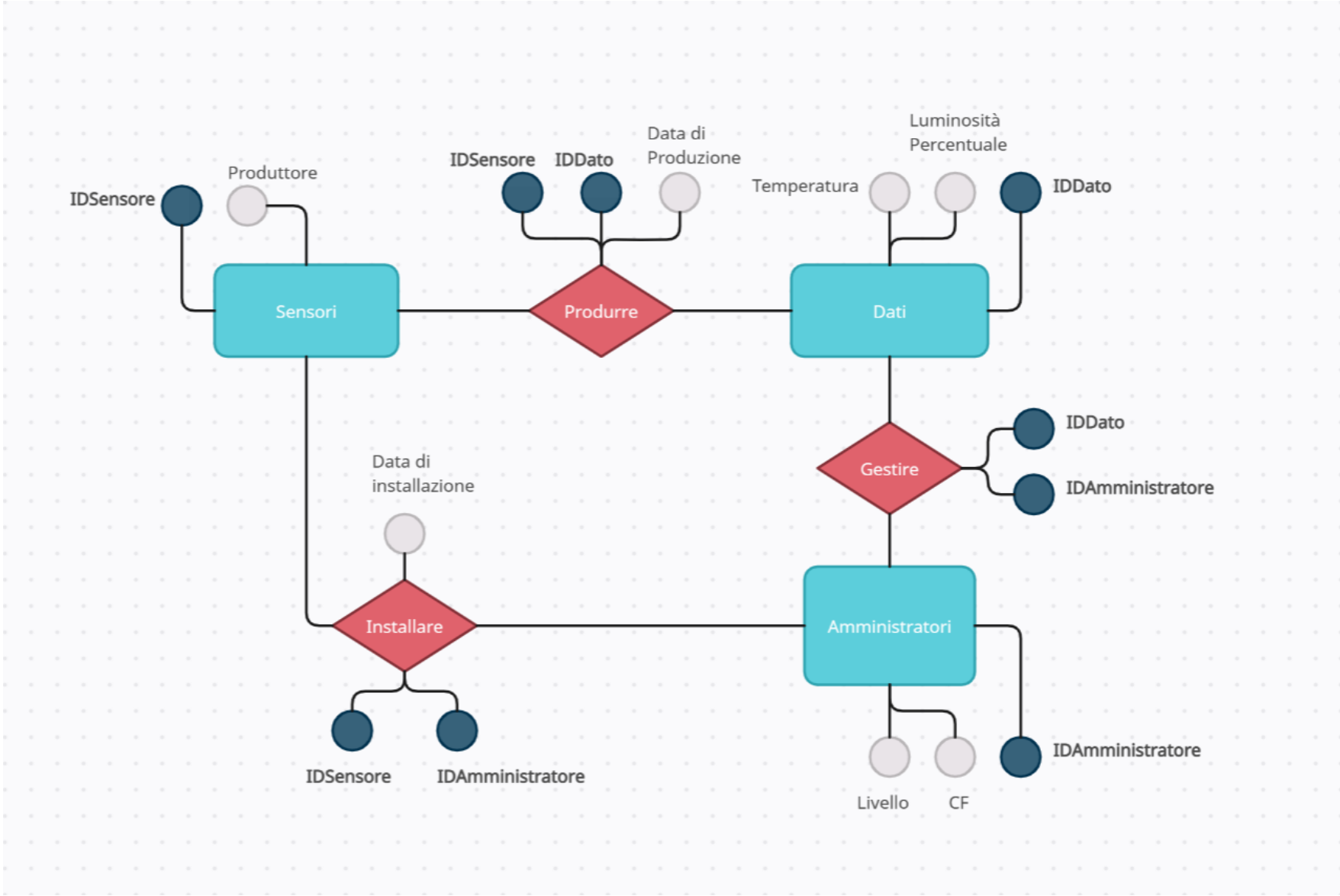
```python
def AddSens(Curs,ConnToDB,SensProd): #add a new sensor
 try:
  Curs.execute("INSERT INTO Sensore (Produttore) VALUES (?)", (SensProd,))
 except mariadb.Error as e:
  print(f"Errore inserimento: {e}")

 ConnToDB.commit()

def RemoveSensID(Curs,ConnToDB,IDSens): #delete one record from given idsens (textbox)
 try:
  Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
  Curs.execute("DELETE FROM Sensore WHERE IDSensore = ?",(IDSens,))
  Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
 except mariadb.Error as e:
  print(f"Errore eliminazione dato dal DB: {e}")

 ConnToDB.commit()
```

```python
def InjectInfo(Curs,ConnToDB): #inject 20 records into the DB (pseudo-random)

 datarand = datetime.date.today()
 for x in range(20):
  datotemp = random.randint(0,35)
  datolum = random.randint(0,100)
  try:
   Curs.execute("INSERT INTO Dato (Temperatura,LuminositàPerc) VALUES (?, ?)", (datotemp,datolum))
  except mariadb.Error as e:
   print(f"Errore Aggiunta dati al DB: {e}")

 ConnToDB.commit()

def PurgeRecords(Curs,ConnToDB): #remove all records into the DB
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("TRUNCATE TABLE Dato")
   Curs.execute("TRUNCATE TABLE Gestione")
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione dati dal DB: {e}")
  ConnToDB.commit()

def PurgeAllSensors(Curs,ConnToDB): #remove all sensors into the DB
  try:
   Curs.execute("SET FOREIGN_KEY_CHECKS = 0")
   Curs.execute("TRUNCATE TABLE Sensore")
   Curs.execute("TRUNCATE TABLE Produzione")
   Curs.execute("TRUNCATE TABLE Installazione")
   Curs.execute("TRUNCATE TABLE Dato")
   Curs.execute("SET FOREIGN_KEY_CHECKS = 1")
  except mariadb.Error as e:
   print(f"Errore eliminazione sensori dal DB: {e}")
  ConnToDB.commit()
```

```python
def GetData(Curs): #get output data return double lists tempdata + lumendata
 riga=0
 tempdata=[]
 lumendata=[]
 IDRecord=[]

 #getting temperatura value
 try:
  Curs.execute("SELECT Temperatura FROM Dato GROUP BY IDDato")
  for riga in Curs:
   tempdata.extend(riga)
 except mariadb.Error as e:
   print(f"Errore recupero dati dal DB: {e}")

 #getting percentuale luminosità value
 riga=0
 try:
  Curs.execute("SELECT LuminositàPerc FROM Dato GROUP BY IDDato")
  for riga in Curs:
   lumendata.extend(riga)
 except mariadb.Error as e:
   print(f"Errore recupero dati dal DB: {e}")

 riga=0
 try:
  Curs.execute("SELECT IDDato FROM Dato GROUP BY IDDato")
  for riga in Curs:
   IDRecord.extend(riga)
 except mariadb.Error as e:
   print(f"Errore recupero dati dal DB: {e}")

 return tempdata,lumendata,IDRecord
```

```python
def CloseDB(Curs,ConnToDB): #close all connections
 Curs.close()
 ConnToDB.close()
```

Schema Entità-Relazione



Schema logico

Sensori: (**IDSensore**, Produttore)

Dati: (**IDSensore**, Temperatura, Luminosita_Perc)

Amministratori: (**IDAmministratore**, Livello, CF)

Produrre: (**IDSensore**, **IDDato**, Data_Produzione)

Installare: (**IDSensore**, **IDAmministratore**, Data_Install)

Gestire: (**IDDato**, **IDAmministratore**)

Schemi DBeaver con i relativi vincoli