Julio Veganos e Hijos interface

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

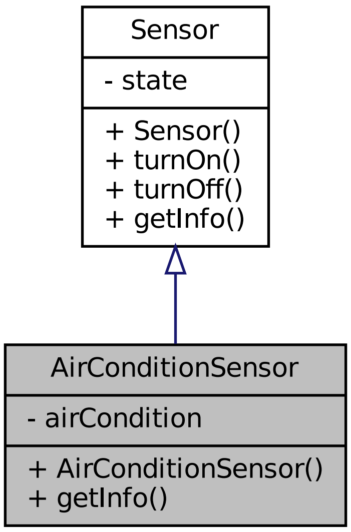
Chapter 4

Class Documentation

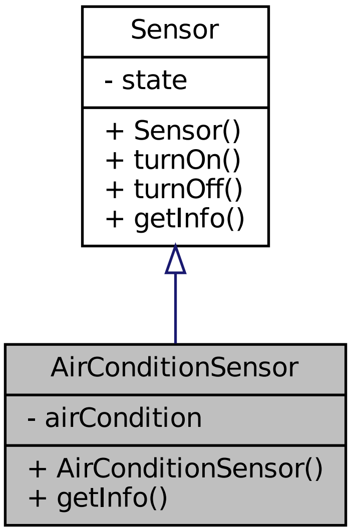
4.1 AirConditionSensor Class Reference

```
#include <AirConditionSensor.h>
```

Inheritance diagram for AirConditionSensor:



Collaboration diagram for AirConditionSensor:



Public Member Functions

- AirConditionSensor ()
- void getInfo ()

Private Attributes

- float airCondition

4.1.1 Detailed Description

Definition at line 17 of file AirConditionSensor.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AirConditionSensor()

```
AirConditionSensor::AirConditionSensor ( )
```

Definition at line 9 of file AirConditionSensor.cpp.

```
9 :Sensor() {} //we define the constructor indicating that it is an inherit class of sensor
```

4.1.3 Member Function Documentation

4.1.3.1 getInfo()

```
void AirConditionSensor::getInfo ( )  [virtual]
```

Creates a random number to simulate de current air condition and displays it

Reimplemented from Sensor.

Definition at line 11 of file AirConditionSensor.cpp.

```
                              {
  srand(time(NULL)+3);
  airCondition = rand()%301;
  cout « "\nCurrent air condition level: " « airCondition « " ppm" « endl;
}
```

References airCondition.

4.1.4 Member Data Documentation

4.1.4.1 airCondition

```
float AirConditionSensor::airCondition  [private]
```

Definition at line 29 of file AirConditionSensor.h.

Referenced by getInfo().

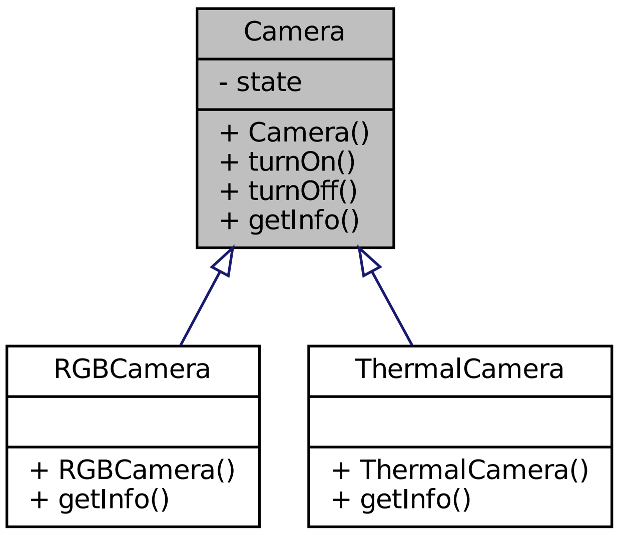
The documentation for this class was generated from the following files:

- AirConditionSensor.h
- AirConditionSensor.cpp

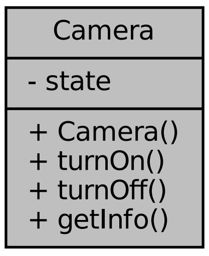
4.2 Camera Class Reference

```
#include <Camera.h>
```

Inheritance diagram for Camera:



Collaboration diagram for Camera:



Public Member Functions

- Camera ()
- void turnOn ()
- void turnOff ()
- virtual void getInfo ()

Private Attributes

- bool state = true

4.2.1 Detailed Description

Definition at line 15 of file Camera.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Camera()

```
Camera::Camera ( )
```

Definition at line 5 of file Camera.cpp.

```
5 {} //we define the constructor
```

4.2.3 Member Function Documentation

4.2.3.1 getInfo()

```
void Camera::getInfo ( )  [virtual]
```

Displays the camera info (polymorphism)

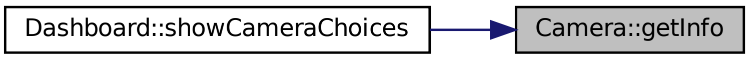
Reimplemented in ThermalCamera, and RGBCamera.

Definition at line 27 of file Camera.cpp.

```
27 {};
```

Referenced by Dashboard::showCameraChoices().

Here is the caller graph for this function:



4.2.3.2 turnOff()

```
void Camera::turnOff ( )
```

Depending on the state of the camera, we turn it off and indicate its state

Definition at line 17 of file Camera.cpp.

```
                       {
  //Depending on the state of the camera, we turn it off and indicate its state
  if (state == true){
    cout « "\nCamera status: Off" « endl;
    state = false;
  }else{
    cout « "\nThe camera is already off" « endl;
  }
}
```

Referenced by Dashboard::showCameraChoices().

Here is the caller graph for this function:

Dashboard::showCameraChoices
Camera::turnOff

4.2.3.3 turnOn()

```
void Camera::turnOn ( )
```

Depending on the state of the camera, we turn it on and indicate its state

Definition at line 7 of file Camera.cpp.

```
                      {
 //Depending on the state of the camera, we turn it on and indicate its state
 if (state == false){
    cout « "\nCamera status: On" « endl;
    state = true;
  }else{
    cout « "\nThe camera is already on" « endl;
  }
}
```

Referenced by Dashboard::showCameraChoices().

Here is the caller graph for this function:

Dashboard::showCameraChoices
Camera::turnOn

4.2.4 Member Data Documentation

4.2.4.1 state

```
bool Camera::state = true  [private]
```

Definition at line 37 of file Camera.h.

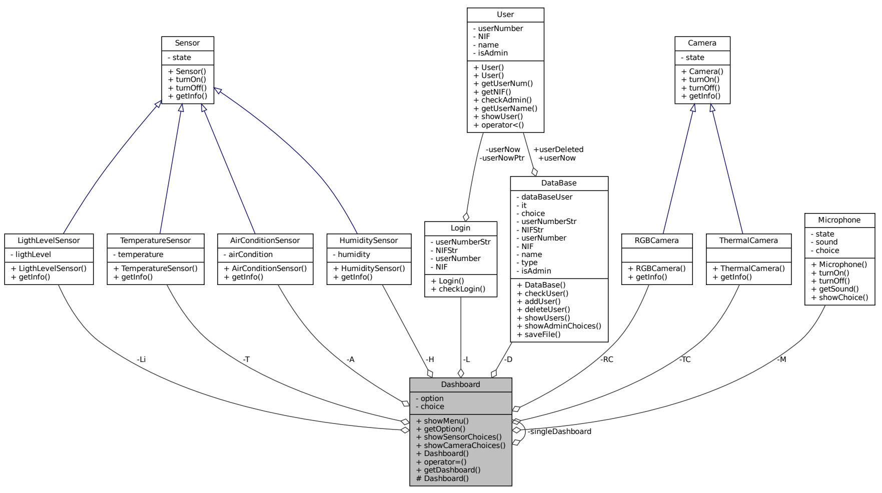
The documentation for this class was generated from the following files:

- Camera.h
- Camera.cpp

4.3 Dashboard Class Reference

```
#include <Dashboard.h>
```

Collaboration diagram for Dashboard:



Public Member Functions

- void showMenu ()
- bool getOption ()
- void showSensorChoices (Sensor ∗)
- void showCameraChoices (Camera ∗)
- Dashboard (Dashboard &otherDashboard)=delete
- void operator= (const Dashboard &)=delete

Static Public Member Functions

- static Dashboard ∗ getDashboard ()

Protected Member Functions

- Dashboard ()

Private Attributes

- int option
- int choice
- Login L
- DataBase D
- TemperatureSensor ∗ T = new TemperatureSensor()
- HumiditySensor ∗ H = new HumiditySensor()
- LigthLevelSensor ∗ Li = new LigthLevelSensor()
- AirConditionSensor ∗ A = new AirConditionSensor()
- RGBCamera ∗ RC = new RGBCamera()
- ThermalCamera ∗ TC = new ThermalCamera()
- Microphone M

Static Private Attributes

- static Dashboard ∗ singleDashboard = nullptr

4.3.1 Detailed Description

Definition at line 16 of file Dashboard.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Dashboard() [1/2]

```
Dashboard::Dashboard (
            Dashboard & otherDashboard )   [delete]
```

4.3.2.2 Dashboard() [2/2]

```
Dashboard::Dashboard ( )   [protected]
```

Definition at line 20 of file Dashboard.cpp.

```
                    { //we define the constructor
  this -> option = option;
}
```

4.3.3 Member Function Documentation

4.3.3.1 getDashboard()

```
Dashboard * Dashboard::getDashboard ( )  [static]
```

Definition at line 225 of file Dashboard.cpp.

```
                                      {
  if(singleDashboard == nullptr){
    singleDashboard = new Dashboard();
  }else{
    cout « "Error: trying to get another instance of a Ball singleton class!\n";
  }

  return singleDashboard;
}
```

Referenced by main().

Here is the caller graph for this function:

main
Dashboard::getDashboard

4.3.3.2 getOption()

```
bool Dashboard::getOption ( )
```

Depending on the option entered, performs an action

Definition at line 119 of file Dashboard.cpp.

```
                             {
  //depending on the option entered, performs an action
  cout « "Enter the number of the action you want to do: ";
  cin » option;

  switch (option)
  {
    case 1:
      system("clear");
      showSensorChoices(T);
      break;

    case 2:
      system("clear");
      showSensorChoices(H);
      break;

    case 3:
      system("clear");
      showSensorChoices(Li);
      break;

    case 4:
      system("clear");
      showSensorChoices(A);
      break;

    case 5:
      system("clear");
      showCameraChoices(RC);
      break;

    case 6:
      system("clear");
      showCameraChoices(TC);
      break;

    case 7:
      system("clear");
      M.showChoice();
      break;

    case 8:
      return true;
      break;

    case 9:
      D.saveFile();
      exit(EXIT_SUCCESS);
      break;

    case 10:
      system("clear");
      D.showAdminChoices();
      break;
  }
  return false;
}
```

4.3.3.3 operator=()

```
void Dashboard::operator= (
              const Dashboard &  )  [delete]
```

4.3.3.4 showCameraChoices()

```
void Dashboard::showCameraChoices (
              Camera * camera )
```

Displays the camera options menu and performs the action depending on the option entered

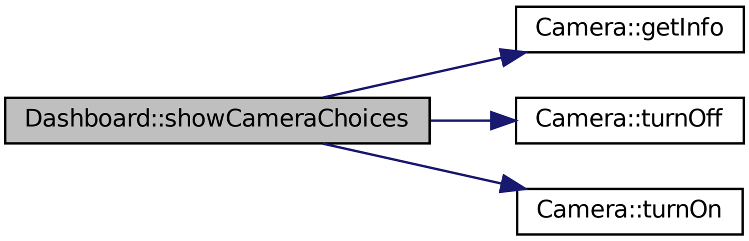
Definition at line 72 of file Dashboard.cpp.

```
                                               {
    //displays the temperature sensor options menu
    while (true) {
      cout « "                                   __  __  _____  _   _  _   _ " « endl
            « "                                  |  \\/  || ____|| \\ | || | | |" « endl
            « "                                  | |\\/| ||  _|  |  \\| || | | |" « endl
            « "                                  | |  | || |___ | |\\  || |_| |" « endl
            « "                                  |_|  |_||_____||_| \\_| \\___/ " « endl
            « "                                                                                   " « endl

            «
      "----------------------------------------------------------------------------------------------" «
      endl
            «
      "oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo\n" «
      endl;
      cout « "                                       1. TURN ON" « endl
          « "                                      2. TURN OFF" « endl
```

```
        « "                                   3. SHOW DATA INFO" « endl
        « "                                   4. BACK TO MENU" « endl;
    cout « "\nEnter your choice number: ";
    cin » choice;

    //Depending on the option that is entered, it calls its respective function
    switch (choice)
    {
    case 1:
      camera->turnOn();
      system("sleep 3");
      system("clear");
      break;

    case 2:
       camera->turnOff();
       system("sleep 3");
       system("clear");
       break;

     case 3:
       camera->getInfo();
       system("sleep 3");
       system("clear");
       break;

     case 4:
       //go back to the main menu
       return;
       break;
     }
   }
 }
```

References Camera::getInfo(), Camera::turnOff(), and Camera::turnOn().

Here is the call graph for this function:



4.3.3.5 showMenu()

```
void Dashboard::showMenu ( )
```

Displays the login screen and, depending on whether the user exists, displays the main menu Checks that the user entered is of type Admin Allows the showMenu() function to be called again when you want to log out and re-enter a user in case of not putting it, it does not verify that the user exists

Definition at line 178 of file Dashboard.cpp.

```
                          {
     try{
       //displays the login screen and, depending on whether the user exists, displays the main menu
```

```
181       if (L.checkLogin(&D)){
182         while (true){
183           system("clear");
184           cout « "                  __  __     _     ___  _   _       __  __  _____  _   _  _   _ " « endl
185           « "              |  \\/  |    / \\    |_ _|| \\ | |     |  \\/  || ____|| \\ | || | | |" « endl
186           « "              | |\\/| |   / _ \\    | | |  \\| |     | |\\/| ||  _|  |  \\| || | | |" « endl
187           « "              | |  | |  / ___ \\   | | | |\\  |     | |  | || |___ | |\\  || |_| |" « endl
188           « "              |_|  |_|/_/   \\_\\|___||_| \\_|     |_|  |_||_____||_| \\_| \\___/ " « endl
189           « "                                                                                  " « endl

190           «
       "-----------------------------------------------------------------------------------------------" «
       endl
191           «
       "°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n" «
       endl;
192
193
194           cout « "                                    1. TEMPERATURE SENSOR" « endl « "
                    2. HUMIDITY SENSOR" « endl
195           « "                                    3. LIGTH LEVEL SENSOR" « endl « "
              4. AIR CONDITION SENSOR" « endl
196           « "                                    5. RGB CAMERA" « endl « "
        6. THERMAL CAMERA" « endl
197           « "                                    7. MICROPHONE" « endl « "
        8. SIGN OFF" « endl
198           « "                                    9. EXIT" « endl;
199
200           //checks that the user entered is of type Admin
201           if(D.userNow.checkAdmin()){
202             cout « "                                    10. MANAGE USERS\n" « endl;
203           }
204
205           //allows the showMenu() function to be called again when you want to log out and re-enter a
       user
206           //in case of not putting it, it does not verify that the user exists
207           if (getOption()){
208             return;
209           }
210         }
211       }
212     }
213
214     catch(UserNumException &except){
215       cout « "Exception: " « except.what() « endl;
216     }
217
218     catch(NIFException &except){
219       cout « "Exception: " « except.what() « endl;
220     }
221 }
```

4.3.3.6 showSensorChoices()

```
void Dashboard::showSensorChoices (
               Sensor * sensor )
```

Displays the sensor options menu and performs the action depending on the option entered

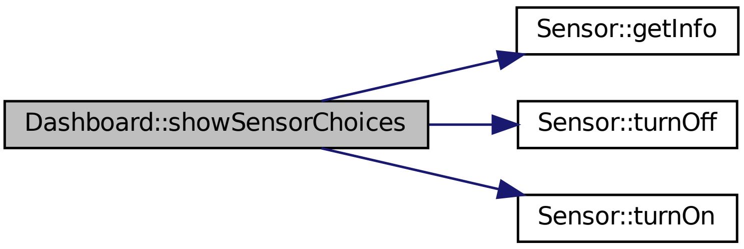
Definition at line 24 of file Dashboard.cpp.

```
24                                               {
25    //displays the temperature sensor options menu
26    while (true) {
27      cout « "                                  __  __  _____  _   _  _   _ " « endl
28           « "                              |  \\/  || ____|| \\ | || | | |" « endl
29           « "                              | |\\/| ||  _|  |  \\| || | | |" « endl
30           « "                              | |  | || |___ | |\\  || |_| |" « endl
31           « "                              |_|  |_||_____||_| \\_| \\___/ " « endl
32           « "                                                                                  " « endl

33           «
       "-----------------------------------------------------------------------------------------------" «
       endl
34           «
       "°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n" «
       endl;
35      cout « "                                  1. TURN ON" « endl
```

```
          « "                                   2. TURN OFF" « endl
          « "                                   3. SHOW DATA INFO" « endl
          « "                                   4. BACK TO MENU" « endl;
      cout « "\nEnter your choice number: ";
      cin » choice;

      //Depending on the option that is entered, it calls its respective function
      switch (choice)
      {
      case 1:
        sensor->turnOn();
        system("sleep 3");
        system("clear");
        break;

      case 2:
        sensor->turnOff();
        system("sleep 3");
        system("clear");
        break;

      case 3:
        sensor->getInfo();
        cout.flush();
        system("sleep 3");
        system("clear");
        break;

      case 4:
        //go back to the main menu
        return;
        break;
      }
    }
  }
```

References Sensor::getInfo(), Sensor::turnOff(), and Sensor::turnOn().

Here is the call graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 A

`AirConditionSensor* Dashboard::A = new AirConditionSensor() [private]`

Definition at line 61 of file Dashboard.h.

4.3.4.2 choice

```
int Dashboard::choice [private]
```

Definition at line 53 of file Dashboard.h.

4.3.4.3 D

```
DataBase Dashboard::D [private]
```

Definition at line 57 of file Dashboard.h.

4.3.4.4 H

```
HumiditySensor* Dashboard::H = new HumiditySensor() [private]
```

Definition at line 59 of file Dashboard.h.

4.3.4.5 L

```
Login Dashboard::L [private]
```

Definition at line 56 of file Dashboard.h.

4.3.4.6 Li

```
LigthLevelSensor* Dashboard::Li = new LigthLevelSensor() [private]
```

Definition at line 60 of file Dashboard.h.

4.3.4.7 M

```
Microphone Dashboard::M [private]
```

Definition at line 64 of file Dashboard.h.

4.3.4.8 option

```
int Dashboard::option  [private]
```

Definition at line 52 of file Dashboard.h.

4.3.4.9 RC

```
RGBCamera* Dashboard::RC = new RGBCamera()  [private]
```

Definition at line 62 of file Dashboard.h.

4.3.4.10 singleDashboard

```
Dashboard * Dashboard::singleDashboard = nullptr  [static], [private]
```

Definition at line 54 of file Dashboard.h.

4.3.4.11 T

```
TemperatureSensor* Dashboard::T = new TemperatureSensor()  [private]
```

Definition at line 58 of file Dashboard.h.

4.3.4.12 TC

```
ThermalCamera* Dashboard::TC = new ThermalCamera()  [private]
```

Definition at line 63 of file Dashboard.h.

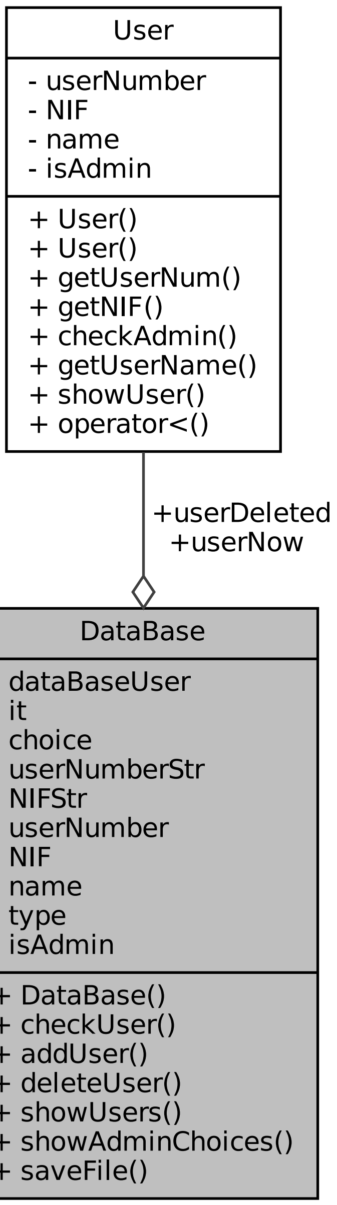
The documentation for this class was generated from the following files:

- Dashboard.h
- Dashboard.cpp

4.4 DataBase Class Reference

```
#include <DataBase.h>
```

Collaboration diagram for DataBase:



Public Member Functions

- DataBase ()
- bool checkUser (int, int)

- void addUser (string, string, string, bool)
- void deleteUser (int)
- void showUsers ()
- void showAdminChoices ()
- void saveFile ()

Public Attributes

- User userNow
- User userDeleted

Private Attributes

- set< User > dataBaseUser
- set< User >::iterator it
- int choice
- std::string userNumberStr
- std::string NIFStr
- int userNumber
- int NIF
- string name
- int type
- bool isAdmin

4.4.1 Detailed Description

Definition at line 18 of file DataBase.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DataBase()

```
DataBase::DataBase ( )
```

Definition at line 17 of file DataBase.cpp.

```
                      { //we define the constructor with the initial users that we are gonna have
  /*this->dataBaseUser.insert(User(1,13172409,"Ana",true));
  this->dataBaseUser.insert(User(2,13172408,"Paula",false));
  this->dataBaseUser.insert(User(3,12345678,"Carlos",false));*/

  // Reads the users found in the users.dat file and dumps them into the database set
  ifstream inUsersFile ("users.dat", ios::in | ios::binary);

  if (!inUsersFile) { // fstream could not open file
    cerr « "File could not be opened." « endl;
    exit (1);
  }

  User user;
  inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
  while (inUsersFile && !inUsersFile.eof()) {
    this->dataBaseUser.insert(user);
    inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
  }
}
```

4.4.3 Member Function Documentation

4.4.3.1 addUser()

```
void DataBase::addUser (
            string userNumberStr,
            string NIFStr,
            string name,
            bool isAdmin )
```

Checks that both the user number and password are the correct size Adds a new user that is entered by an admin in the terminal

Definition at line 55 of file DataBase.cpp.

```
                                                                                   {
    //checks that both the user number and password are the correct size
    try{
      for(int n = 0; n < userNumberStr.length(); n++){
        if(int(userNumberStr[n]) < 47 || int(userNumberStr[n] > 57)){
          throw StringException();
        }
        userNumber = stoi(userNumberStr);
      }

      if (userNumber < 1 || userNumber > 99999){
        throw UserNumException();
      }

      for(int n = 0; n < NIFStr.length(); n++){
        if(int(NIFStr[n]) < 47 || int(NIFStr[n] > 57)){
          throw StringException();
        }
        NIF = stoi(NIFStr);
      }

      if (NIF < 9999999 || NIF > 99999999){
        throw NIFException();
      }

      if(isAdmin != 1 && isAdmin != 0){
        throw TypeError();
      }

      //adds a new user that is entered by an admin in the terminal
      this->dataBaseUser.insert(User(userNumber, NIF, name, isAdmin));
    }

    catch(UserNumException &except){
      cout « "Exception: " « except.what() « endl;
    }

    catch(NIFException &except){
      cout « "Exception: " « except.what() « endl;
    }

    catch(TypeError &except){
      cout « "Exception: " « except.what() « endl;
    }

     catch(StringException &except){
       cout « "Exception: " « except.what() « endl;
     }
}
```

4.4.3.2 checkUser()

```
bool DataBase::checkUser (
             int userNumber,
             int NIF )
```

Checks if the user exixts

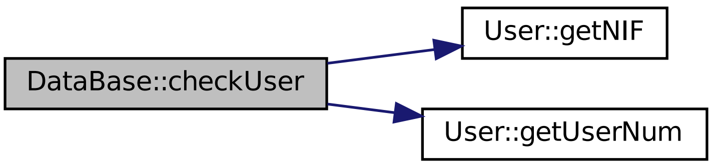
Definition at line 38 of file DataBase.cpp.

```
                                              {
    bool authentication = false;
    //checks if the user exixts
    do{
      for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
        User user = *it;
        if(userNumber==user.getUserNum() && NIF==user.getNIF()){
            userNow = user;
            return true;
            authentication = true;
        }
      }
    }while(authentication == false);

    return false;
}
```

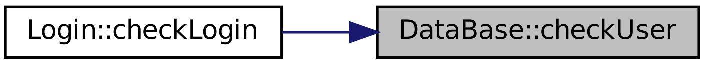
References User::getNIF(), and User::getUserNum().

Referenced by Login::checkLogin().

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.3.3 deleteUser()

```
void DataBase::deleteUser (
             int userNumber )
```

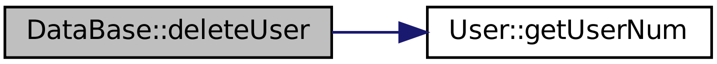
Deletes the user that is entered by an admin in the terminal

Definition at line 105 of file DataBase.cpp.

```
                                            {
  //deletes the user that is entered by an admin in the terminal
  for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
    User user = *it;
    if(userNumber==user.getUserNum()){
      userDeleted = user;
    }
  }
  this->dataBaseUser.erase(userDeleted);
  cout « "El usuario ha sido eliminado" « endl;
}
```

References User::getUserNum().

Here is the call graph for this function:



4.4.3.4 saveFile()

```
void DataBase::saveFile ( )
```

Saves the users found in the set to the file users.dat

Definition at line 181 of file DataBase.cpp.

```
                         {
  fstream outUsersFile ("users.dat", ios::in | ios::out | ios::trunc | ios::binary); // ios::in will
   require an existing file
  // Uses: "users.dat", generated in previous example
  if (!outUsersFile) { // fstream could not open file
    cerr « "File could not be opened." « endl;
    exit (1);
  }

  int position = 0;

  for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
    User user = *it;
    outUsersFile.seekp (position * sizeof (User));
    outUsersFile.write (reinterpret_cast <const char *> (&user), sizeof (User));
    position ++;
  }
}
```

4.4.3.5 showAdminChoices()

```
void DataBase::showAdminChoices ( )
```

Displays the manage users options menu, that only an Admin can use

Definition at line 130 of file DataBase.cpp.

```
                              {
  //displays the manage users options menu, that only an Admin can use
  while (true) {
    cout « "\n                             MANAGE USERS" « endl
      « "----------------------------------------------------------------------" « endl;
    cout « "                  1. ADD USER" « endl
        « "                  2. DELETE USER" « endl
        « "                  3. SHOW USERS LIST" « endl
        « "                  4. BACK TO MENU" « endl;
    cout « "Enter your choice number: ";
    cin » choice;

    switch (choice)
    {
    case 1:
      cout « "Enter a new user" « endl;
      cout « "User number: ";
      cin » userNumberStr;
      cout « "User password: ";
      cin » NIFStr;
      cout « "User name: ";
      cin » name;
      cout « "User type (1=Admin, 0=User): ";
      cin » type;

      if (type == 1){
        isAdmin = true;
      }else{  isAdmin = false; }

      addUser(userNumberStr, NIFStr, name, isAdmin);
      break;

    case 2:
      cout « "Enter the user number that you want to delete: ";
      cin » userNumber;
      deleteUser(userNumber);
      break;

    case 3:
      cout « "USERS LIST: " « endl;
      cout « "Nº\tNAME\t\tTYPE" « endl;
      showUsers();
      break;

    case 4:
      return;
      break;
    }
  }
}
```

4.4.3.6 showUsers()

```
void DataBase::showUsers ( )
```

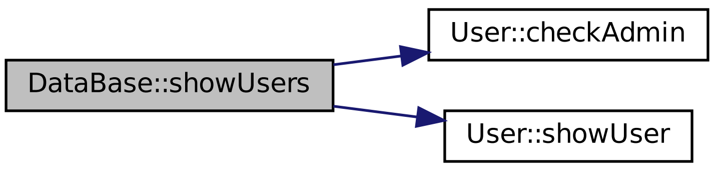
Shows the users that are saved in the database

Definition at line 117 of file DataBase.cpp.

```
                       {
  //shows the users that are saved in the database
  for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
    User user = *it;
    user.showUser();
    if(user.checkAdmin()){
      cout « "\t\tAdmin" « endl;
    }else{
      cout « "\t\tEmployee" « endl;
    }
  }
}
```

References User::checkAdmin(), and User::showUser().

Here is the call graph for this function:



4.4.4 Member Data Documentation

4.4.4.1 choice

`int DataBase::choice [private]`

Definition at line 62 of file DataBase.h.

4.4.4.2 dataBaseUser

`set<User> DataBase::dataBaseUser [private]`

Definition at line 60 of file DataBase.h.

4.4.4.3 isAdmin

`bool DataBase::isAdmin [private]`

Definition at line 69 of file DataBase.h.

4.4.4.4 it

```
set<User>::iterator DataBase::it  [private]
```

Definition at line 61 of file DataBase.h.

4.4.4.5 name

```
string DataBase::name  [private]
```

Definition at line 67 of file DataBase.h.

4.4.4.6 NIF

```
int DataBase::NIF  [private]
```

Definition at line 66 of file DataBase.h.

4.4.4.7 NIFStr

```
std::string DataBase::NIFStr  [private]
```

Definition at line 64 of file DataBase.h.

4.4.4.8 type

```
int DataBase::type  [private]
```

Definition at line 68 of file DataBase.h.

4.4.4.9 userDeleted

```
User DataBase::userDeleted
```

Definition at line 56 of file DataBase.h.

4.4.4.10 userNow

```
User DataBase::userNow
```

Definition at line 55 of file DataBase.h.

4.4.4.11 userNumber

```
int DataBase::userNumber  [private]
```

Definition at line 65 of file DataBase.h.

4.4.4.12 userNumberStr

```
std::string DataBase::userNumberStr  [private]
```

Definition at line 63 of file DataBase.h.

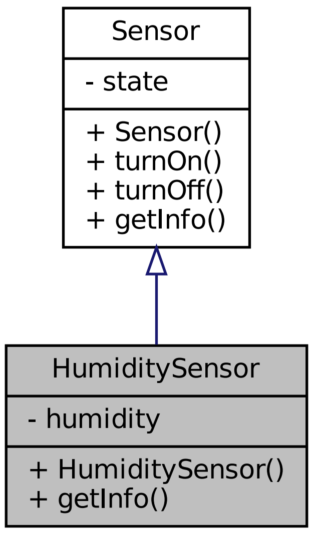
The documentation for this class was generated from the following files:

- DataBase.h
- DataBase.cpp

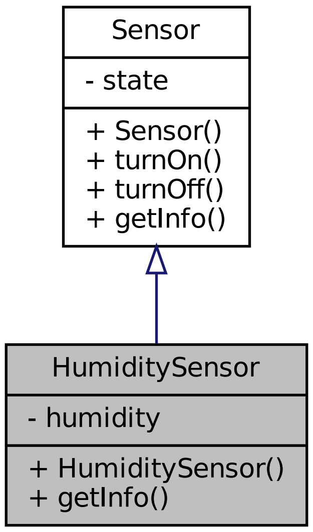
4.5 HumiditySensor Class Reference

```
#include <HumiditySensor.h>
```

Inheritance diagram for HumiditySensor:



Collaboration diagram for HumiditySensor:



Public Member Functions

- HumiditySensor ()
- void getInfo ()

Private Attributes

- float humidity

4.5.1 Detailed Description

Definition at line 17 of file HumiditySensor.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 HumiditySensor()

```
HumiditySensor::HumiditySensor ( )
```

Definition at line 9 of file HumiditySensor.cpp.

```
9 :Sensor() {} //we define the constructor indicating that it is an inherit class of sensor
```

4.5.3 Member Function Documentation

4.5.3.1 getInfo()

```
void HumiditySensor::getInfo ( )  [virtual]
```

Creates a random number to simulate de current humidity and displays it

Reimplemented from Sensor.

Definition at line 11 of file HumiditySensor.cpp.

```
                             {
    srand(time(NULL)+1);
    humidity = rand()%101;
    cout « "\nCurrent humidity: " « humidity « "%" « endl;
}
```

References humidity.

4.5.4 Member Data Documentation

4.5.4.1 humidity

```
float HumiditySensor::humidity  [private]
```

Definition at line 29 of file HumiditySensor.h.

Referenced by getInfo().

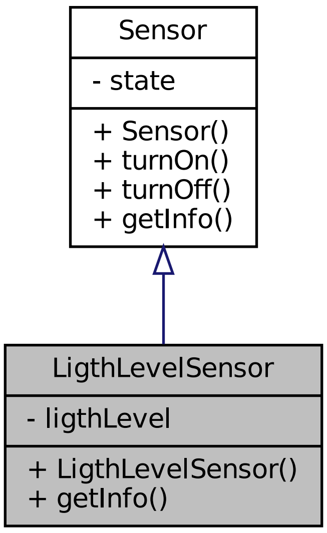
The documentation for this class was generated from the following files:

- HumiditySensor.h
- HumiditySensor.cpp

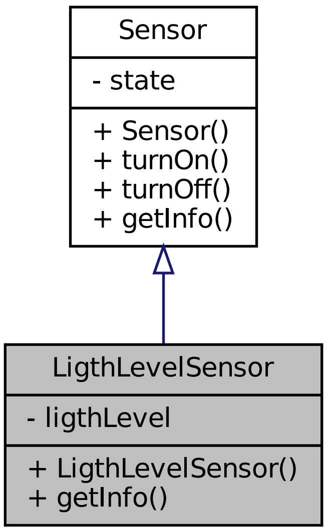
4.6 LigthLevelSensor Class Reference

```
#include <LigthLevelSensor.h>
```

Inheritance diagram for LigthLevelSensor:



Collaboration diagram for LigthLevelSensor:



Public Member Functions

- LigthLevelSensor ()
- void getInfo ()

Private Attributes

- float ligthLevel

4.6.1 Detailed Description

Definition at line 17 of file LigthLevelSensor.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 LigthLevelSensor()

```
LigthLevelSensor::LigthLevelSensor ( )
```

Definition at line 9 of file LigthLevelSensor.cpp.

```
:Sensor() {} //we define the constructor indicating that it is an inherit class of sensor
```

4.6.3 Member Function Documentation

4.6.3.1 getInfo()

```
void LigthLevelSensor::getInfo ( ) [virtual]
```

Creates a random number to simulate de current ligth level and displays it

Reimplemented from Sensor.

Definition at line 11 of file LigthLevelSensor.cpp.

```
                              {
    srand(time(NULL)+2);
    ligthLevel = rand()%101;
    cout « "\nCurrent ligth level: " « ligthLevel « "%" « endl;
}
```

References ligthLevel.

4.6.4 Member Data Documentation

4.6.4.1 ligthLevel

```
float LigthLevelSensor::ligthLevel  [private]
```

Definition at line 29 of file LigthLevelSensor.h.

Referenced by getInfo().

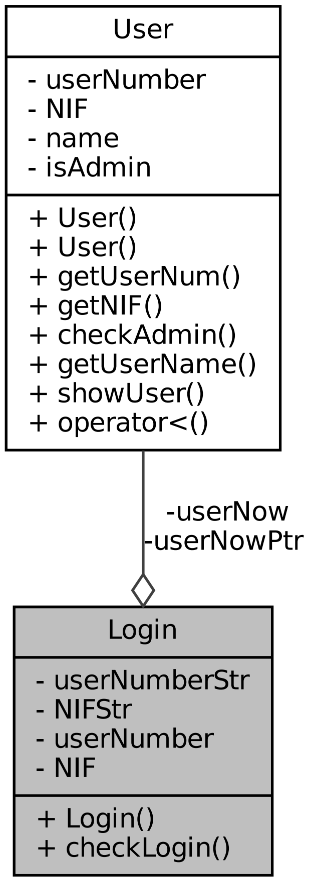
The documentation for this class was generated from the following files:

- LigthLevelSensor.h
- LigthLevelSensor.cpp

4.7 Login Class Reference

```
#include <Login.h>
```

Collaboration diagram for Login:



Public Member Functions

- Login ()
- bool checkLogin (DataBase ∗)

Private Attributes

- std::string userNumberStr
- std::string NIFStr
- int userNumber
- int NIF
- User userNow
- User ∗ userNowPtr

4.7.1 Detailed Description

Definition at line 17 of file Login.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Login()

```
Login::Login ( )
```

Definition at line 8 of file Login.cpp.

```
8 {} //we define the constructor
```

4.7.3 Member Function Documentation

4.7.3.1 checkLogin()

```
bool Login::checkLogin (
               DataBase * d )
```

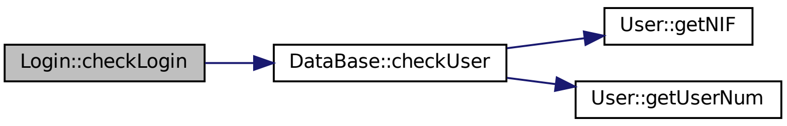
Returns if the login is correct

Definition at line 10 of file Login.cpp.

```
10                                     {
11    bool excep = true;
12    while (excep == true){
13      try{
14        cout«"\n                               User: ";
15        cin»userNumberStr;
16        for(int n = 0; n < userNumberStr.length(); n++){
17          if(int(userNumberStr[n]) < 47 || int(userNumberStr[n]) > 57)){
18            throw StringException();
19            excep = false;
20          }
21          userNumber = stoi(userNumberStr);
22        }
23
24        //returns the password entered
25        cout«"                                Password: ";
26        cin»NIFStr;
27        for(int n = 0; n < NIFStr.length(); n++){
28          if(int(NIFStr[n]) < 47 || int(NIFStr[n] > 57)){
29            throw StringException();
30            excep = false;
31          }
32          NIF = stoi(NIFStr);
33        }
34      }
35
36      catch(StringException &except){
37        cout « "Exception: " « except.what() « endl;
38        continue;
39      }
40
41      return d->checkUser(userNumber, NIF);
42    }
43    return false;
44  }
```

References DataBase::checkUser().

Here is the call graph for this function:



4.7.4 Member Data Documentation

4.7.4.1 NIF

```
int Login::NIF  [private]
```

Definition at line 33 of file Login.h.

4.7.4.2 NIFStr

```
std::string Login::NIFStr  [private]
```

Definition at line 31 of file Login.h.

4.7.4.3 userNow

```
User Login::userNow  [private]
```

Definition at line 34 of file Login.h.

4.7.4.4 userNowPtr

```
User* Login::userNowPtr  [private]
```

Definition at line 35 of file Login.h.

4.7.4.5 userNumber

```
int Login::userNumber  [private]
```

Definition at line 32 of file Login.h.

4.7.4.6 userNumberStr

```
std::string Login::userNumberStr  [private]
```

Definition at line 30 of file Login.h.

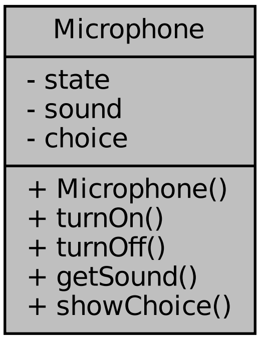
The documentation for this class was generated from the following files:

- Login.h
- Login.cpp

4.8 Microphone Class Reference

```
#include <Microphone.h>
```

Collaboration diagram for Microphone:



Public Member Functions

- Microphone ()
- void turnOn ()
- void turnOff ()
- void getSound ()
- void showChoice ()

Private Attributes

- bool state = true
- string sound = "\nListening command..."
- int choice

4.8.1 Detailed Description

Definition at line 15 of file Microphone.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Microphone()

```
Microphone::Microphone ( )
```

Definition at line 7 of file Microphone.cpp.

```
{} //we define the constructor
```

4.8.3 Member Function Documentation

4.8.3.1 getSound()

```
void Microphone::getSound ( )
```

Prints that a command is being listened for

Definition at line 9 of file Microphone.cpp.

```
                          {
  //prints that a command is being listened for
  cout « sound « endl;
}
```

4.8.3.2 showChoice()

```
void Microphone::showChoice ( )
```

Displays the microphone options menu Depending on the option that is entered, it calls its respective function.

Definition at line 34 of file Microphone.cpp.

```
                            {
  //displays the microphone options menu
  while (true) {
    cout « "\n              ___  ___ _                            _                        " « endl
    « "               |  \\/  |(_)                          | |                      " « endl
    « "               | .  . | _   ___  _ __  ___   _ __  | |__    ___   _ __    ___ " « endl
    « "               | |\\/| || | / __|| '__|/ _ \\ | '_ \\ | '_ \\  / _ \\ | '_ \\  / _ \\" « endl
    « "               | |  | || || (__ | |  | (_) || |_) || | | || (_) || | | ||  __/" « endl
    « "               \\_|  |_/|_| \\___||_|   \\___/ | .__/ |_| |_| \\___/ |_| |_| \\___|" « endl
    « "                                             | |                              " « endl
    « "                                             |_|                              " « endl
    « "                                                                              " « endl
    « "-----------------------------------------------------------------------------------------------" «
    endl
    « "°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n"
    « endl;
    cout « "                                       1. TURN ON" « endl
        « "                                      2. TURN OFF" « endl
        « "                                      3. RECORD COMMAND" « endl
        « "                                      4. BACK TO MENU" « endl;
    cout « "Enter your choice number: ";
    cin » choice;

    switch (choice)
    {
    case 1:
      turnOn();
      system("sleep 3");
      system("clear");
      break;

    case 2:
      turnOff();
      system("sleep 3");
      system("clear");
      break;

    case 3:
      getSound();
      system("sleep 3");
      system("clear");
      break;

    case 4:
      return;
      break;
    }
  }
}
```

4.8.3.3 turnOff()

`void Microphone::turnOff ( )`

Depending on the state of the microphone, we turn it off and indicate its state

Definition at line 24 of file Microphone.cpp.

```
                          {
  //Depending on the state of the microphone, we turn it off and indicate its state
  if (state = true){
    cout « "\nMicrophone status: Off" « endl;
    state = false;
  }else{
    cout « "\nThe microphone is already off" « endl;
  }
}
```

4.8.3.4 turnOn()

`void Microphone::turnOn ( )`

Depending on the state of the microphone, we turn it on and indicate its state

Definition at line 14 of file Microphone.cpp.

```
                         {
  //Depending on the state of the microphone, we turn it on and indicate its state
  if (state == false){
    cout « "\nMicrophone status: On" « endl;
    state = true;
  }else{
    cout « "\nThe microphone is already on" « endl;
  }
}
```

4.8.4 Member Data Documentation

4.8.4.1 choice

```
int Microphone::choice  [private]
```

Definition at line 45 of file Microphone.h.

4.8.4.2 sound

```
string Microphone::sound = "\nListening command..."  [private]
```

Definition at line 44 of file Microphone.h.

4.8.4.3 state

```
bool Microphone::state = true  [private]
```

Definition at line 43 of file Microphone.h.

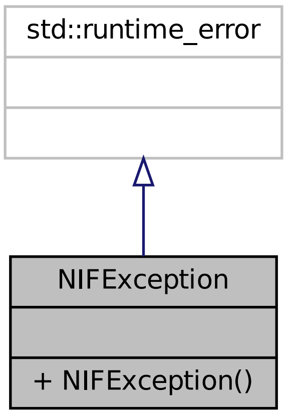
The documentation for this class was generated from the following files:

- Microphone.h
- Microphone.cpp

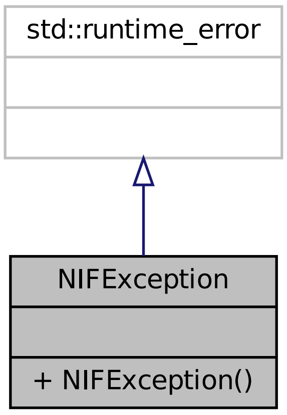
4.9 NIFException Class Reference

```
#include <NIFException.h>
```

Inheritance diagram for NIFException:



Collaboration diagram for NIFException:



Public Member Functions

- NIFException ()

4.9.1 Detailed Description

Definition at line 9 of file NIFException.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 NIFException()

```
NIFException::NIFException ( )
```

Definition at line 3 of file NIFException.cpp.

```
4   :std::runtime_error ("the number of digits for password must be 8."){};
```

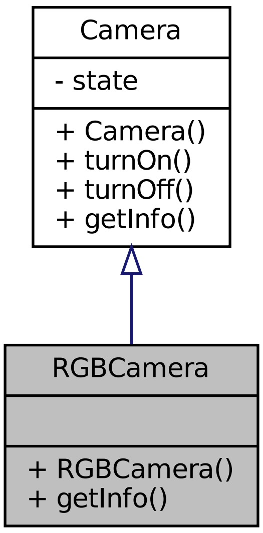
The documentation for this class was generated from the following files:

- NIFException.h
- NIFException.cpp

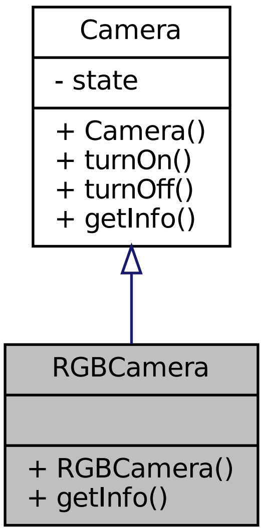
4.10 RGBCamera Class Reference

```
#include <RGBCamera.h>
```

Inheritance diagram for RGBCamera:



Collaboration diagram for RGBCamera:



Public Member Functions

- RGBCamera ()
- void getInfo ()

4.10.1 Detailed Description

Definition at line 17 of file RGBCamera.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 RGBCamera()

```
RGBCamera::RGBCamera ( )
```

Definition at line 9 of file RGBCamera.cpp.

```
9 :Camera() {} //we define the constructor indicating that it is an inherit class of camera
```

4.10.3 Member Function Documentation

4.10.3.1 getInfo()

```
void RGBCamera::getInfo ( )  [virtual]
```

Displays the "thermal image"

Reimplemented from Camera.

Definition at line 11 of file RGBCamera.cpp.

```
11                         {
12    //prints that an image is being displayed
13    cout « " ---------------------------------------------" « endl « "-
       LIVE... -" « endl « "-                                             -" « endl « "-
                               -" « endl « "-                                             -" « endl « "-
                                               -" « endl « "-                                             -"
       « endl « "-                                             -" « endl « "-
                   -" « endl « "-                RGB image                -" « endl « "-
                                   -" « endl « "-                                             -" « endl «
       "-                                             -" « endl « "-
           -" « endl « "-                                             -" « endl « "-
                           -" « endl « "-                                             -" « endl « "-
                                           -" « endl « " ---------------------------------------------"«
       endl;;
14 }
```

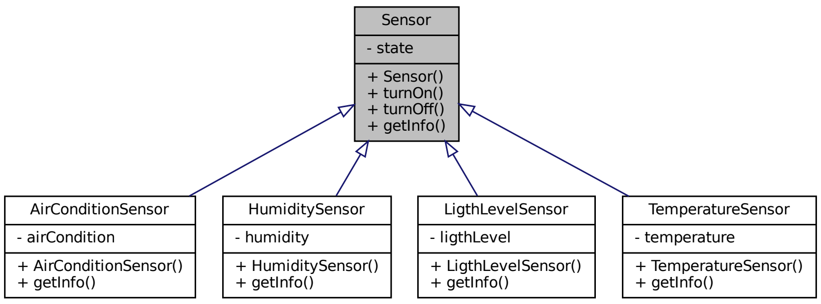
The documentation for this class was generated from the following files:

- RGBCamera.h
- RGBCamera.cpp

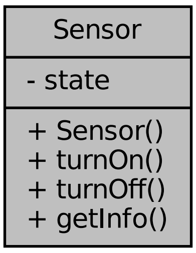
4.11 Sensor Class Reference

```
#include <Sensor.h>
```

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



Public Member Functions

- Sensor ()
- void turnOn ()
- void turnOff ()
- virtual void getInfo ()

Private Attributes

- bool state = true

4.11.1 Detailed Description

Definition at line 15 of file Sensor.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Sensor()

```
Sensor::Sensor ( )
```

Definition at line 5 of file Sensor.cpp.

```
5 {} //we define the constructor
```

4.11.3 Member Function Documentation

4.11.3.1 getInfo()

```
void Sensor::getInfo ( ) [virtual]
```

Displays the sensor info (polymorphism)

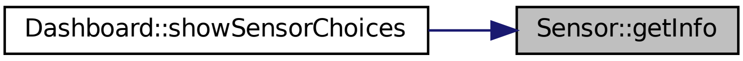
Reimplemented in TemperatureSensor, LigthLevelSensor, HumiditySensor, and AirConditionSensor.

Definition at line 27 of file Sensor.cpp.

```
27 {}
```

Referenced by Dashboard::showSensorChoices().

Here is the caller graph for this function:



4.11.3.2 turnOff()

```
void Sensor::turnOff ( )
```

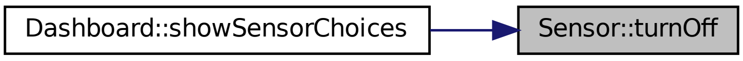
Depending on the state of the sensor, we turn it off and indicate its state

Definition at line 17 of file Sensor.cpp.

```
                        {
    //Depending on the state of the sensor, we turn it off and indicate its state
    if (state == true){
      cout « "\nSensor Status: Off" « endl;
      state = false;
    }else{
      cout « "\nThe sensor is already off" « endl;
    }
  }
```

Referenced by Dashboard::showSensorChoices().

Here is the caller graph for this function:



4.11.3.3 turnOn()

```
void Sensor::turnOn ( )
```

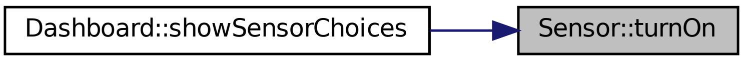
Depending on the state of the sensor, we turn it on and indicate its state

Definition at line 7 of file Sensor.cpp.

```
                       {
   //Depending on the state of the sensor, we turn it on and indicate its state
   if (state == false){
      cout « "\nSensor status: On" « endl;
      state = true;
    }else{
      cout « "\nThe sensor is already on" « endl;
    }
  }
```

Referenced by Dashboard::showSensorChoices().

Here is the caller graph for this function:



4.11.4 Member Data Documentation

4.11.4.1 state

```
bool Sensor::state = true  [private]
```

Definition at line 37 of file Sensor.h.

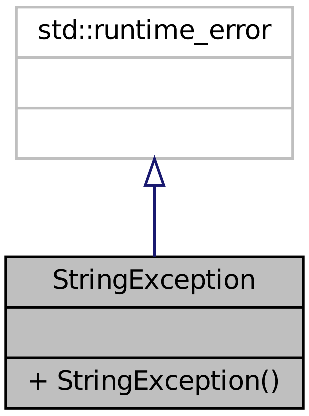
The documentation for this class was generated from the following files:

- Sensor.h
- Sensor.cpp

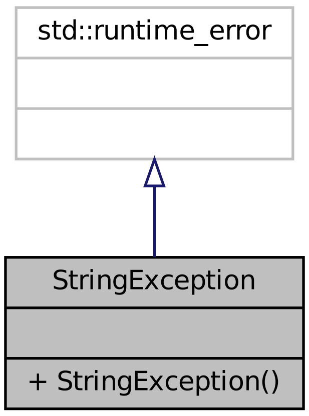
4.12 StringException Class Reference

```
#include <StringException.h>
```

Inheritance diagram for StringException:



Collaboration diagram for StringException:



Public Member Functions

- StringException ()

4.12.1 Detailed Description

Definition at line 9 of file StringException.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 StringException()

```
StringException::StringException ( )
```

Definition at line 3 of file StringException.cpp.

```
4    :std::runtime_error ("you've entered a character when only integers are allowed"){};
```

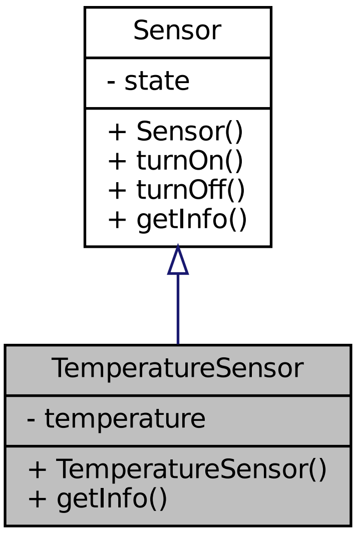
The documentation for this class was generated from the following files:

- StringException.h
- StringException.cpp

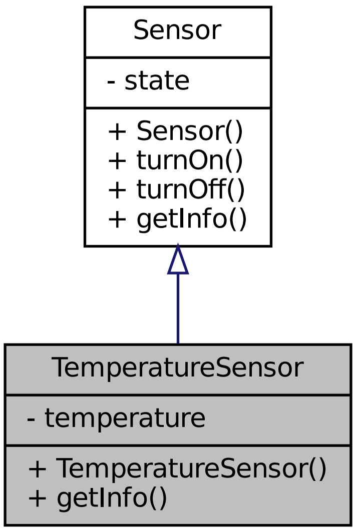
4.13 TemperatureSensor Class Reference

```
#include <TemperatureSensor.h>
```

Inheritance diagram for TemperatureSensor:



Collaboration diagram for TemperatureSensor:



Public Member Functions

- TemperatureSensor ()
- void getInfo ()

Private Attributes

- float temperature

4.13.1 Detailed Description

Definition at line 17 of file TemperatureSensor.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TemperatureSensor()

```
TemperatureSensor::TemperatureSensor ( )
```

Definition at line 9 of file TemperatureSensor.cpp.

```
:Sensor() {} //we define the constructor indicating that it is an inherit class of sensor
```

4.13.3 Member Function Documentation

4.13.3.1 getInfo()

```
void TemperatureSensor::getInfo ( )  [virtual]
```

Creates a random number to simulate de current temperature and displays it

Reimplemented from Sensor.

Definition at line 11 of file TemperatureSensor.cpp.

```
                           {
  srand(time(NULL));
  temperature = 25+rand()%(40 +1 - 25);
  cout « "\nCurrent temperature: " « temperature « " °C" « endl;
}
```

References temperature.

4.13.4 Member Data Documentation

4.13.4.1 temperature

```
float TemperatureSensor::temperature  [private]
```

Definition at line 29 of file TemperatureSensor.h.

Referenced by getInfo().

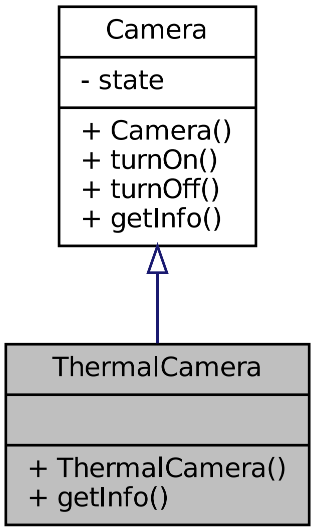
The documentation for this class was generated from the following files:

- TemperatureSensor.h
- TemperatureSensor.cpp

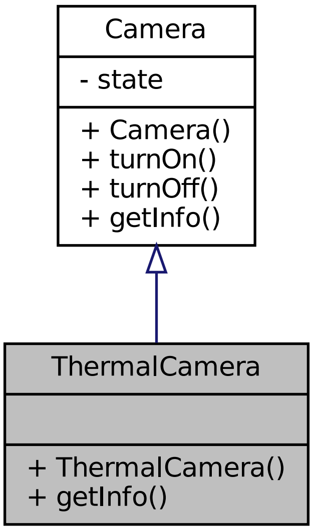
4.14 ThermalCamera Class Reference

```
#include <ThermalCamera.h>
```

Inheritance diagram for ThermalCamera:



Collaboration diagram for ThermalCamera:



Public Member Functions

- ThermalCamera ()
- void getInfo ()

4.14.1 Detailed Description

Definition at line 17 of file ThermalCamera.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ThermalCamera()

```
ThermalCamera::ThermalCamera ( )
```

Definition at line 8 of file ThermalCamera.cpp.

```
8 :Camera() {} //we define the constructor indicating that it is an inherit class of camera
```

4.14.3 Member Function Documentation

4.14.3.1 getInfo()

```
void ThermalCamera::getInfo ( ) [virtual]
```

Displays the "thermal image"

Reimplemented from Camera.

Definition at line 10 of file ThermalCamera.cpp.

```
                          {
    //prints that an image is being displayed
    cout « " ---------------------------------------------" « endl « "-
      LIVE... -" « endl « "-                                             -" « endl « "-
                          -" « endl « "-                                             -" « endl « "-
                                          -" « endl « "-                                             -"
      « endl « "-                                             -" « endl « "-
                 -" « endl « "-                Thermal image                -" « endl « "-
                                 -" « endl « "-                                             -" « endl «
      "-                                             -" « endl « "-
          -" « endl « "-                                             -" « endl « "-
                          -" « endl « "-                                             -" « endl « "-
                                          -" « endl « " ---------------------------------------------"«
      endl;
}
```

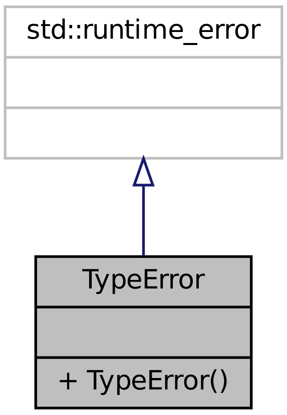
The documentation for this class was generated from the following files:

- ThermalCamera.h
- ThermalCamera.cpp

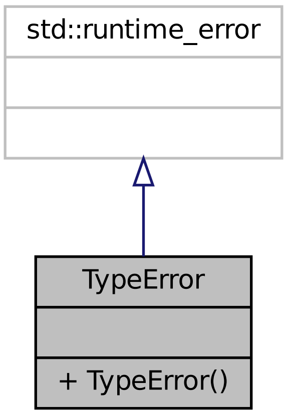
4.15 TypeError Class Reference

```
#include <TypeError.h>
```

Inheritance diagram for TypeError:



Collaboration diagram for TypeError:



Public Member Functions

- TypeError ()

4.15.1 Detailed Description

Definition at line 9 of file TypeError.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 TypeError()

```
TypeError::TypeError ( )
```

Definition at line 3 of file TypeError.cpp.

```
4   :std::runtime_error ("the number must be 1 or 0!"){};
```

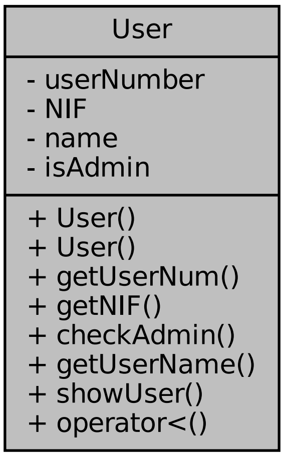
The documentation for this class was generated from the following files:

- TypeError.h
- TypeError.cpp

4.16 User Class Reference

```
#include <User.h>
```

Collaboration diagram for User:



Public Member Functions

- User ()
- User (float userNumber, float NIF, string name, bool isAdmin)
- int getUserNum ()
- int getNIF ()
- bool checkAdmin ()
- string getUserName ()
- void showUser ()
- bool operator< (const User &) const

Private Attributes

- int userNumber
- int NIF
- char name [10]
- bool isAdmin

4.16.1 Detailed Description

Definition at line 15 of file User.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 User() [1/2]

```
User::User ( )
```

Definition at line 9 of file User.cpp.

```
{} //we define the default constructor
```

4.16.2.2 User() [2/2]

```
User::User (
            float userNumber,
            float NIF,
            string name,
            bool isAdmin )
```

Definition at line 11 of file User.cpp.

```
                                                                    { //we define the constructor with
      parameters
    this -> userNumber = userNumber;
    this -> NIF = NIF;
    strcpy(this->name,name.c_str());
    this -> isAdmin = isAdmin;
};
```

4.16.3 Member Function Documentation

4.16.3.1 checkAdmin()

```
bool User::checkAdmin ( )
```

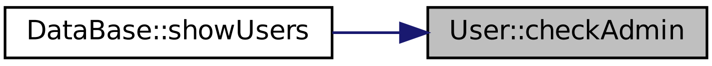
Returns if a user is admin o not by true or false

Definition at line 31 of file User.cpp.

```
                        {
    return isAdmin;
}
```

Referenced by DataBase::showUsers().

Here is the caller graph for this function:



4.16.3.2 getNIF()

```
int User::getNIF ( )
```

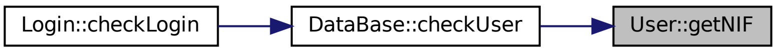
Returns the NIF

Definition at line 24 of file User.cpp.

```
                    {
  if (NIF < 9999999 || NIF > 99999999){
    throw NIFException();
  }
  return NIF;
}
```

Referenced by DataBase::checkUser().

Here is the caller graph for this function:



4.16.3.3 getUserName()

```
string User::getUserName ( )
```

Returns the user name

Definition at line 35 of file User.cpp.

```
                            {
  return name;
}
```

4.16.3.4 getUserNum()

```
int User::getUserNum ( )
```

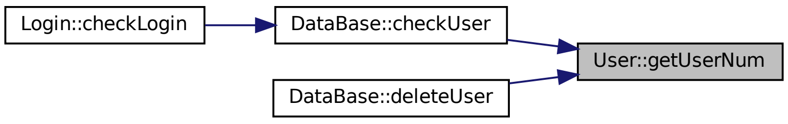
Returns the user number

Definition at line 18 of file User.cpp.

```
                        {
  if (userNumber < 1 || userNumber > 99999){
    throw UserNumException();
  }
  return userNumber;
}
```

Referenced by DataBase::checkUser(), and DataBase::deleteUser().

Here is the caller graph for this function:



4.16.3.5 operator<()

```
bool User::operator< (
             const User & user ) const
```

Overloads the < operator to compare in the set

Definition at line 43 of file User.cpp.

```
                                         {
    return userNumber < user.userNumber;
}
```

References userNumber.

4.16.3.6 showUser()

```
void User::showUser ( )
```

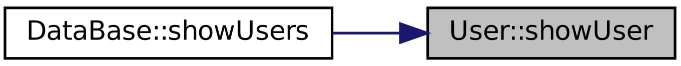
Returns the user name

Definition at line 39 of file User.cpp.

```
                    {
    cout « userNumber « "\t" « name;
}
```

Referenced by DataBase::showUsers().

Here is the caller graph for this function:



4.16.4 Member Data Documentation

4.16.4.1 isAdmin

```
bool User::isAdmin  [private]
```

Definition at line 56 of file User.h.

4.16.4.2 name

```
char User::name[10]  [private]
```

Definition at line 55 of file User.h.

4.16.4.3 NIF

```
int User::NIF  [private]
```

Definition at line 54 of file User.h.

4.16.4.4 userNumber

```
int User::userNumber  [private]
```

Definition at line 53 of file User.h.

Referenced by operator<().

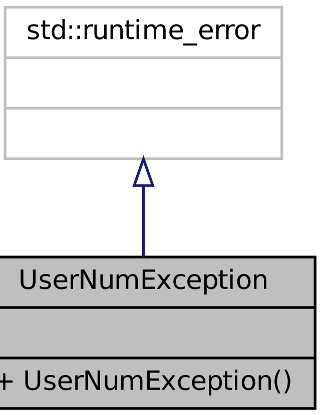
The documentation for this class was generated from the following files:

- User.h
- User.cpp

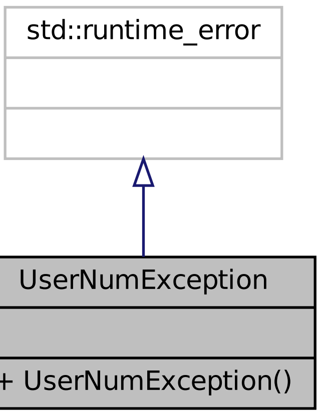
4.17 UserNumException Class Reference

```
#include <UserNumException.h>
```

Inheritance diagram for UserNumException:



Collaboration diagram for UserNumException:



Public Member Functions

- UserNumException ()

4.17.1 Detailed Description

Definition at line 9 of file UserNumException.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 UserNumException()

```
UserNumException::UserNumException ( )
```

Definition at line 3 of file UserNumException.cpp.

```
    :std::runtime_error ("the number of digits in user must be between 1 and 5."){};
```

The documentation for this class was generated from the following files:

- UserNumException.h
- UserNumException.cpp

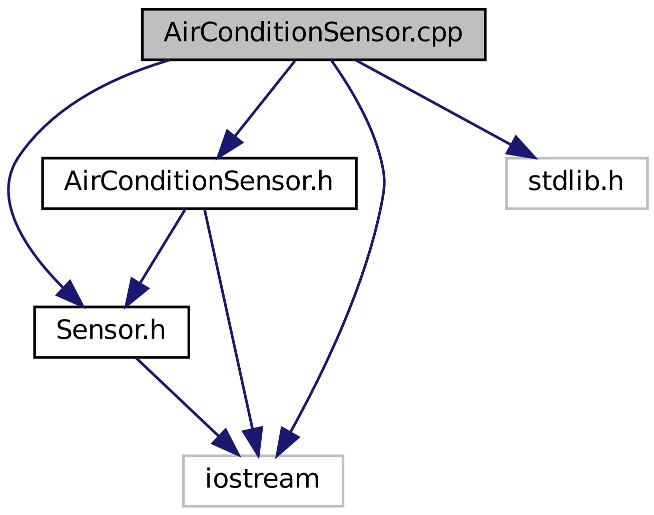
Chapter 5

File Documentation

5.1 AirConditionSensor.cpp File Reference

```
#include "AirConditionSensor.h"
#include "Sensor.h"
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for AirConditionSensor.cpp:

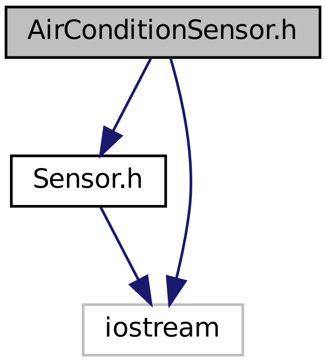


5.2 AirConditionSensor.h File Reference

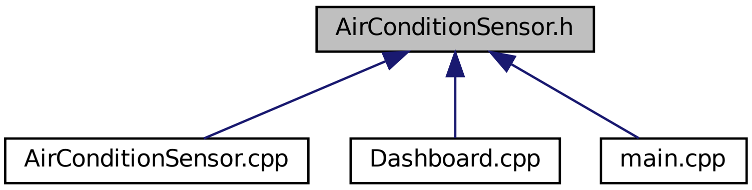
defines the AirConditionSensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

```
#include "Sensor.h"
#include <iostream>
```

Include dependency graph for AirConditionSensor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AirConditionSensor

5.2.1 Detailed Description

defines the AirConditionSensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

Author

Ana Martínez Albendea

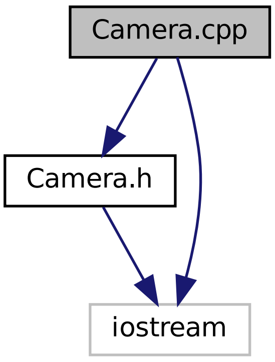
Date

2022-11-23

5.3 Camera.cpp File Reference

```
#include "Camera.h"
#include <iostream>
```

Include dependency graph for Camera.cpp:

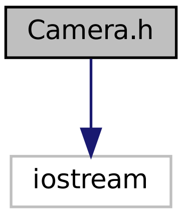


5.4 Camera.h File Reference

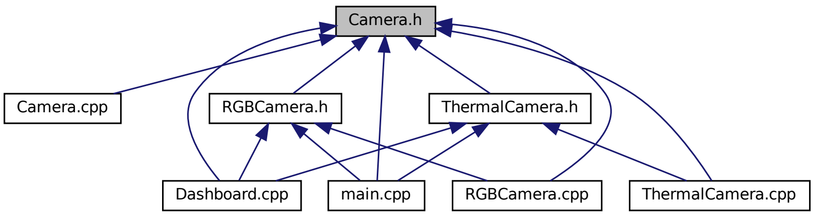
defines the camera class with its attributes, methods, and constructor

```
#include <iostream>
```

Include dependency graph for Camera.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Camera

5.4.1 Detailed Description

defines the camera class with its attributes, methods, and constructor

Author

Ana Martínez Albendea

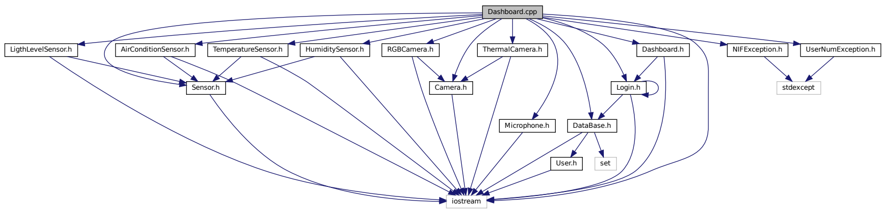
Date

2022-11-23

5.5 Dashboard.cpp File Reference

```
#include "TemperatureSensor.h"
#include "HumiditySensor.h"
#include "LigthLevelSensor.h"
#include "AirConditionSensor.h"
#include "RGBCamera.h"
#include "ThermalCamera.h"
#include "Microphone.h"
#include "DataBase.h"
#include "Login.h"
#include "Dashboard.h"
#include "Sensor.h"
#include "Camera.h"
#include "UserNumException.h"
#include "NIFException.h"
#include <iostream>
```

Include dependency graph for Dashboard.cpp:

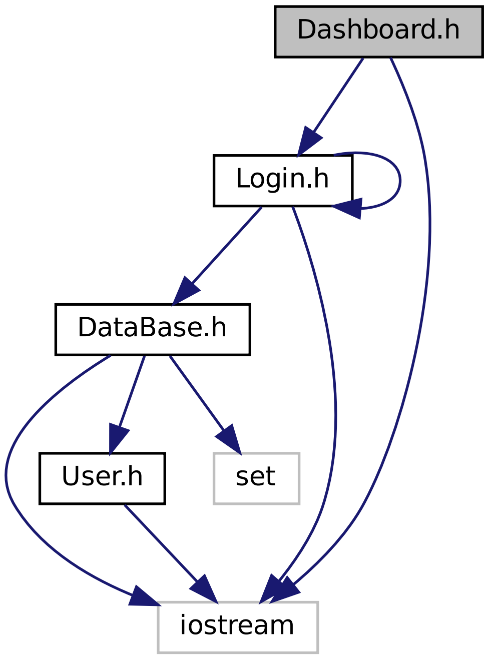


5.6 Dashboard.h File Reference

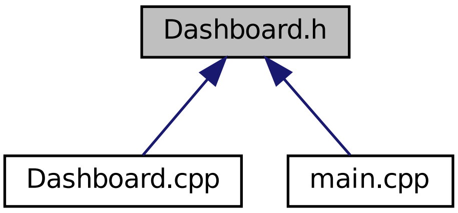
defines the dashboard class with its attributes, methods, and constructor

```
#include "Login.h"
#include <iostream>
```

Include dependency graph for Dashboard.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Dashboard

5.6.1 Detailed Description

defines the dashboard class with its attributes, methods, and constructor

Author

Ana Martínez Albendea

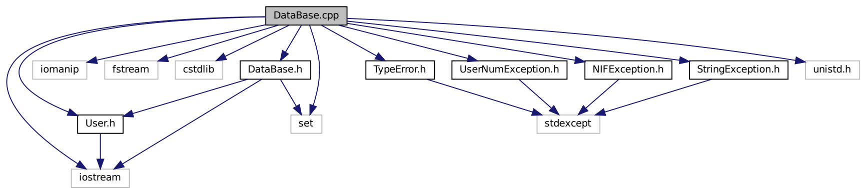
Date

2022-11-23

5.7 DataBase.cpp File Reference

```
#include "User.h"
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include "DataBase.h"
#include "TypeError.h"
#include "UserNumException.h"
#include "NIFException.h"
#include "StringException.h"
#include <unistd.h>
#include <iostream>
#include <set>
```

Include dependency graph for DataBase.cpp:

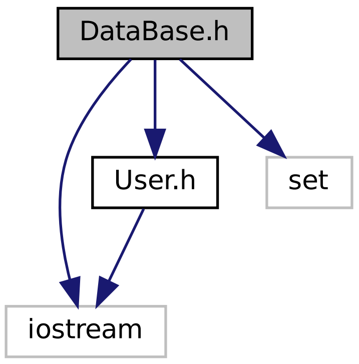


5.8 DataBase.h File Reference

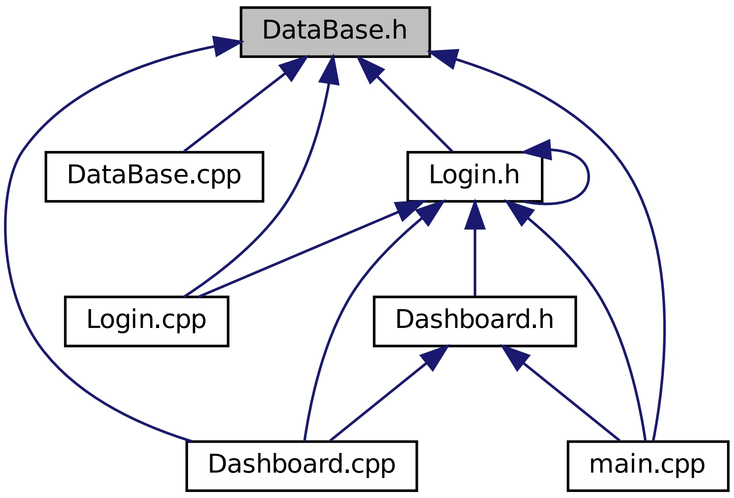
defines the database class with its attributes, methods, and constructor

```
#include <iostream>
#include "User.h"
```

`#include <set>`
Include dependency graph for DataBase.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DataBase

5.8.1 Detailed Description

defines the database class with its attributes, methods, and constructor

Author

Ana Martínez Albendea

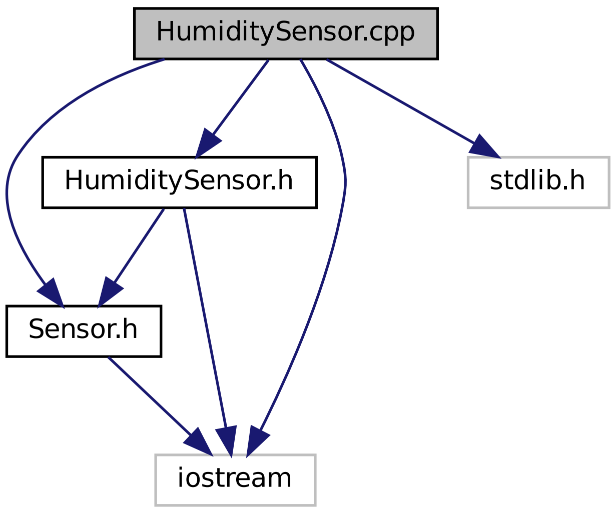
Date

2022-11-23

5.9 HumiditySensor.cpp File Reference

```
#include "HumiditySensor.h"
#include "Sensor.h"
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for HumiditySensor.cpp:

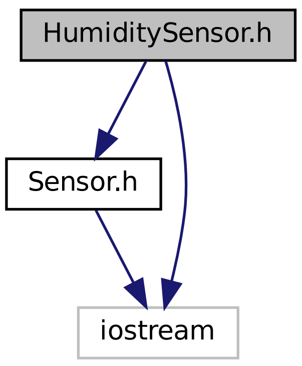


5.10 HumiditySensor.h File Reference

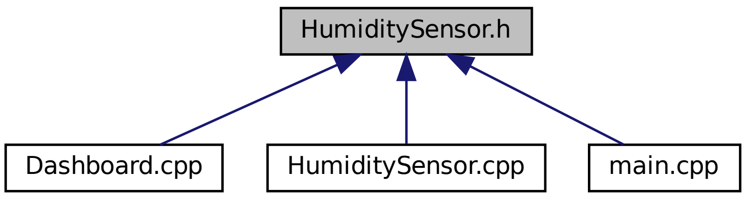
defines the HumiditySensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

```
#include "Sensor.h"
#include <iostream>
```

Include dependency graph for HumiditySensor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class HumiditySensor

5.10.1 Detailed Description

defines the HumiditySensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

Author

Ana Martínez Albendea

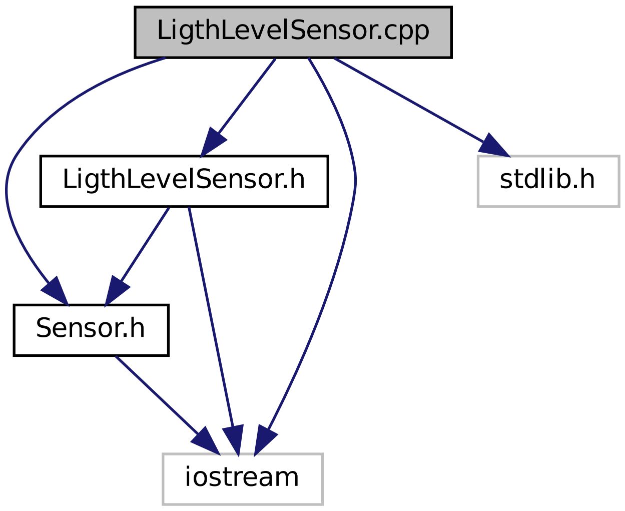
Date

2022-11-23

5.11 LigthLevelSensor.cpp File Reference

```
#include "LigthLevelSensor.h"
#include "Sensor.h"
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for LigthLevelSensor.cpp:

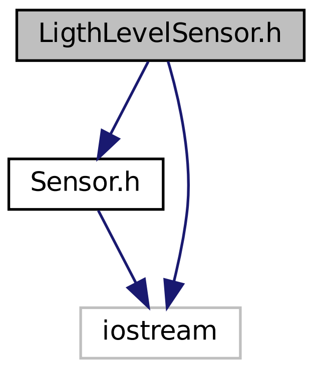


5.12 LigthLevelSensor.h File Reference

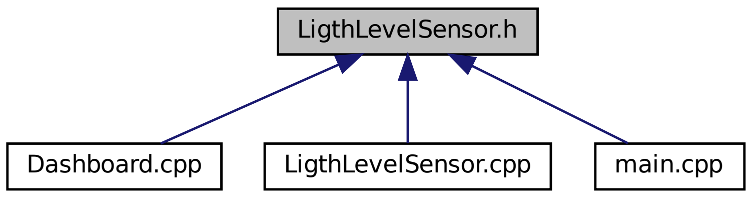
defines the LigthLevelSensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

```
#include "Sensor.h"
#include <iostream>
```

Include dependency graph for LigthLevelSensor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LigthLevelSensor

5.12.1 Detailed Description

defines the LigthLevelSensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

Author

Ana Martínez Albendea

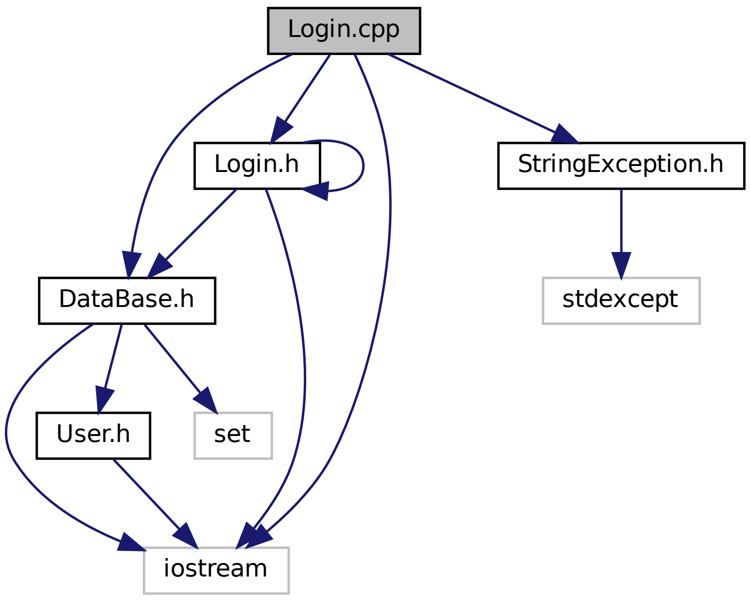
Date

2022-11-23

5.13 Login.cpp File Reference

```
#include "Login.h"
#include <iostream>
#include "StringException.h"
#include "DataBase.h"
```

Include dependency graph for Login.cpp:

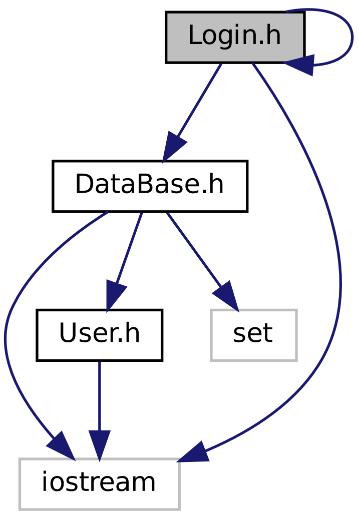


5.14 Login.h File Reference

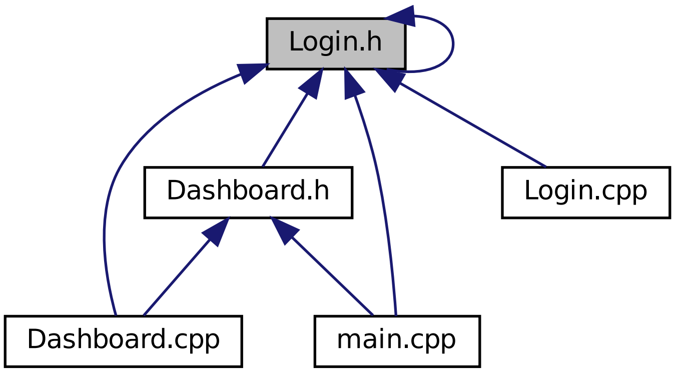
defines the login class with its attributes, methods, and constructor

```
#include "Login.h"
#include "DataBase.h"
#include <iostream>
```

Include dependency graph for Login.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Login

5.14.1 Detailed Description

defines the login class with its attributes, methods, and constructor

Author

Ana Martínez Albendea

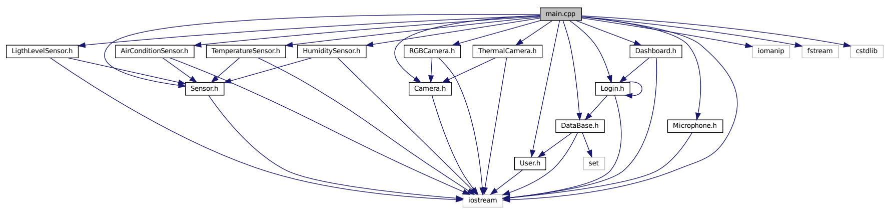
Date

2022-11-23

5.15 main.cpp File Reference

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include "Sensor.h"
#include "TemperatureSensor.h"
#include "HumiditySensor.h"
#include "LigthLevelSensor.h"
#include "AirConditionSensor.h"
#include "Camera.h"
#include "RGBCamera.h"
#include "ThermalCamera.h"
#include "User.h"
#include "Microphone.h"
#include "DataBase.h"
#include "Login.h"
#include "Dashboard.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

5.15.1 Function Documentation

5.15.1.1 main()

```
int main ( )
```

Definition at line 28 of file main.cpp.

```
28         {
29   //title created with ascii art
30   system("clear");
31   cout «
      "ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo" «
      endl;
32   cout «
      "-----------------------------------------------------------------------------------------------" «
      endl;
33   cout « "       _       _ _       __     __                                       _   _ _  _
      " « endl
34   « "     | |_   _| (_) ___   \\ \\   / /__  __ _  __ _ _ __   ___  ___    ___  | | | (_)(_) ___  ___ " «
      endl
35   « "  _  | | | | | | |/ _ \\   \\ \\ / / _ \\/ _` |/ _` | '_ \\ / _ \\/ __|  / _ \\ | |_| | || |/ _ \\/
      __|" « endl
36   « " | |_| | |_| | | | (_) |   \\ V /  __/ (_| | (_| | | | | (_) \\__ \\ |  __/ |  _  | || | (_) \\__
      \\" « endl
37   « "  \\___/ \\__,_|_|_|\\___/     \\_/ \\___|\\__, |\\__,_|_| |_|\\___/|___/  \\___| |_| |_|_|/
      |\\___/|___/" « endl
38   « "                              |___/                                       |__/           " «
      endl;
39   cout «
      "\n---------------------------------------------------------------------------------------------" «
      endl;
40   cout «
      "ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo\n" «
      endl;
41
42
43   cout « "                                Please, log in to continue\n"  « endl;
44   try{
45
46     // Creates the binary file users.dat and save 10 spaces to store users. Can be created without that
      reservation
47     /*ofstream outUsersFile ("users.dat", ios::out | ios::binary);
48
49     if (!outUsersFile) { // ofstream could not open file
50       cerr « "File could not be opened." « endl;
51       exit (1);
52     }
53
54     User user; // fill with zeros each data member
55     for (int i = 0; i < 10; i++){ // write 10 empty records to file
56       outUsersFile.write (reinterpret_cast <const char *> (&user), sizeof (User));
57     }*/
58
59     Dashboard* Ds = Dashboard::getDashboard(); //construction of a dashboard type object
60
61     //while loop allowing to log back in after log out
62     while (true) {
63       Ds->Dashboard::showMenu(); //call to the function that shows the main menu of the dashboard
64     }
65   }
66
67   catch(bad_alloc &except){
68     cout « "Exception: " « except.what() « endl;
69   }
70 }
```

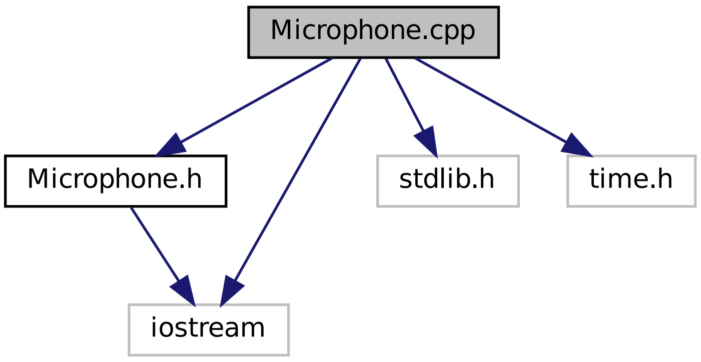
References Dashboard::getDashboard().

Here is the call graph for this function:

5.16 Microphone.cpp File Reference

```
#include "Microphone.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for Microphone.cpp:

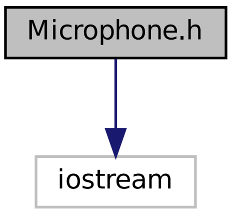


5.17 Microphone.h File Reference

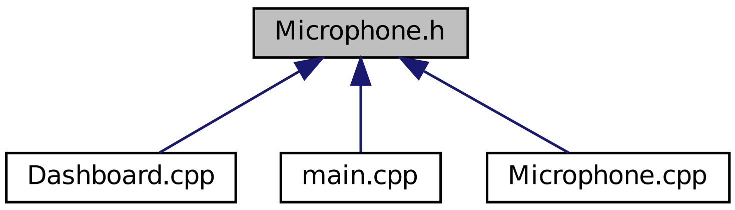
defines the microphone class with its attributes, methods, and constructor

```
#include <iostream>
```

Include dependency graph for Microphone.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Microphone

5.17.1 Detailed Description

defines the microphone class with its attributes, methods, and constructor

Author

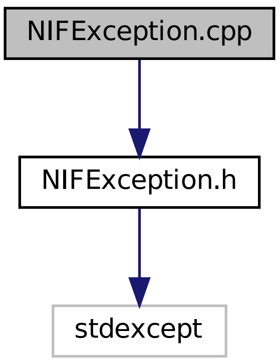
Ana Martínez Albendea

Date

2022-11-23

5.18 NIFException.cpp File Reference

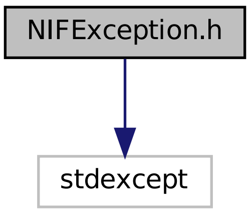
```
#include "NIFException.h"
```
Include dependency graph for NIFException.cpp:



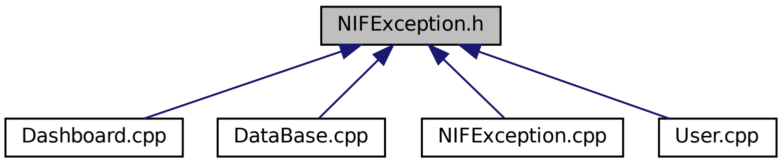
5.19 NIFException.h File Reference

defines the NIF exception class with its constructor

`#include <stdexcept>`
Include dependency graph for NIFException.h:



This graph shows which files directly or indirectly include this file:



Classes

- class NIFException

5.19.1 Detailed Description

defines the NIF exception class with its constructor

Author

Ana Martínez Albendea

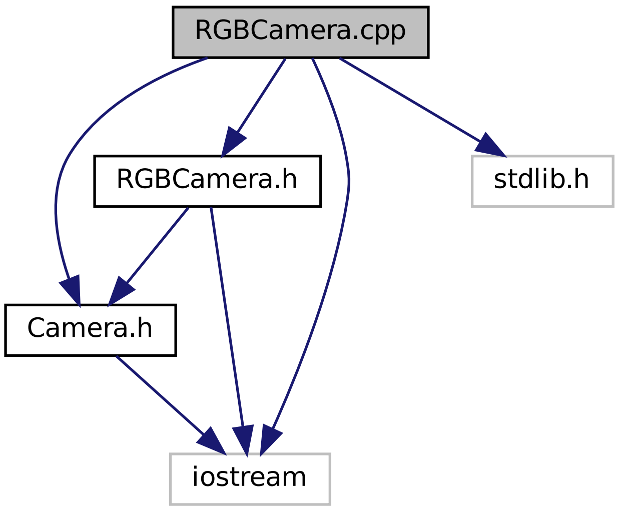
Date

2022-11-23

5.20 RGBCamera.cpp File Reference

```
#include "RGBCamera.h"
#include "Camera.h"
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for RGBCamera.cpp:

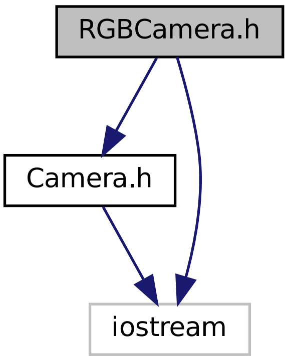


5.21 RGBCamera.h File Reference

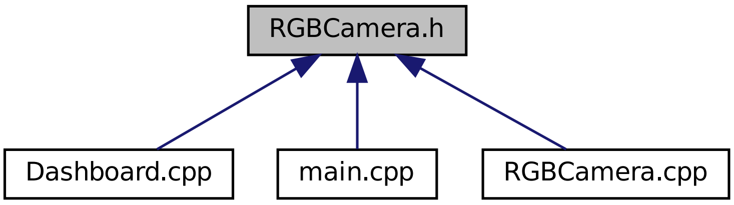
defines the RGBCamera class, wich is is inherited from the camera class, with its attributes, methods, and constructor

```
#include "Camera.h"
#include <iostream>
```

Include dependency graph for RGBCamera.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RGBCamera

5.21.1 Detailed Description

defines the RGBCamera class, wich is is inherited from the camera class, with its attributes, methods, and constructor

Author

Ana Martínez Albendea

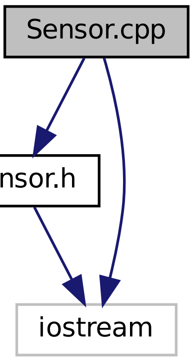
Date

2022-11-23

5.22 Sensor.cpp File Reference

```
#include "Sensor.h"
#include <iostream>
```

Include dependency graph for Sensor.cpp:

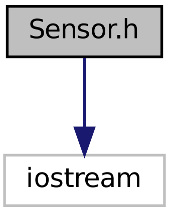


5.23 Sensor.h File Reference

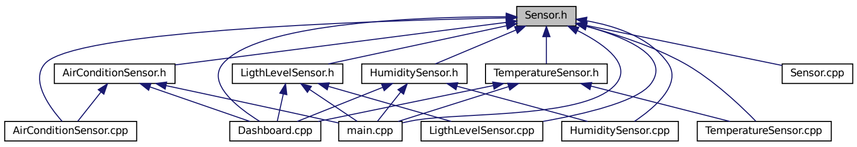
defines the sensor class with its attributes, methods, and constructor

```
#include <iostream>
```

Include dependency graph for Sensor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Sensor

5.23.1 Detailed Description

defines the sensor class with its attributes, methods, and constructor

Author

Ana Martínez Albendea

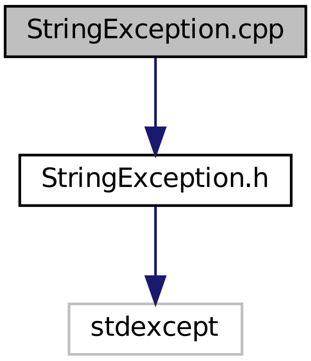
Date

2022-11-23

5.24 StringException.cpp File Reference

```
#include "StringException.h"
```

Include dependency graph for StringException.cpp:

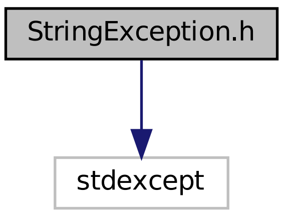


5.25 StringException.h File Reference

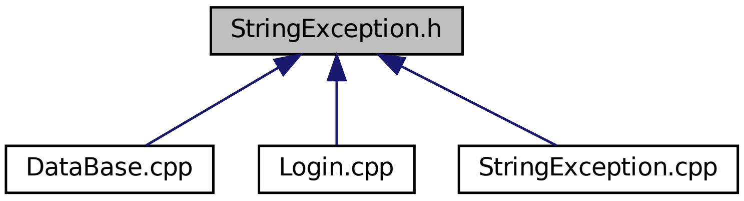
defines the string exception class with its constructor

```
#include <stdexcept>
```

Include dependency graph for StringException.h:



This graph shows which files directly or indirectly include this file:



Classes

- class StringException

5.25.1 Detailed Description

defines the string exception class with its constructor

Author

Ana Martínez Albendea

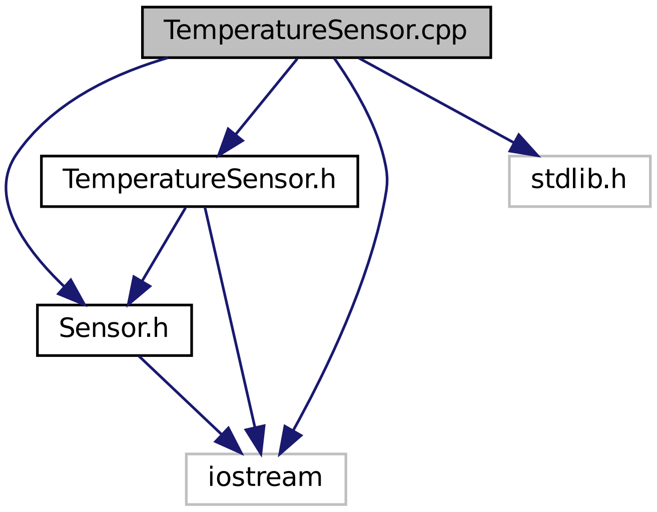
Date

2022-11-23

5.26 TemperatureSensor.cpp File Reference

```
#include "TemperatureSensor.h"
#include "Sensor.h"
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for TemperatureSensor.cpp:

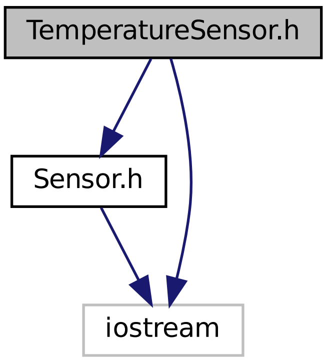


5.27 TemperatureSensor.h File Reference

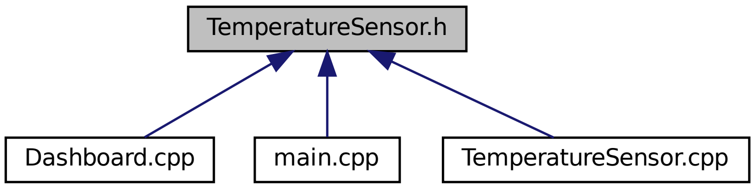
defines the TemperatureSensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

```
#include "Sensor.h"
#include <iostream>
```

Include dependency graph for TemperatureSensor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TemperatureSensor

5.27.1 Detailed Description

defines the TemperatureSensor class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

Author

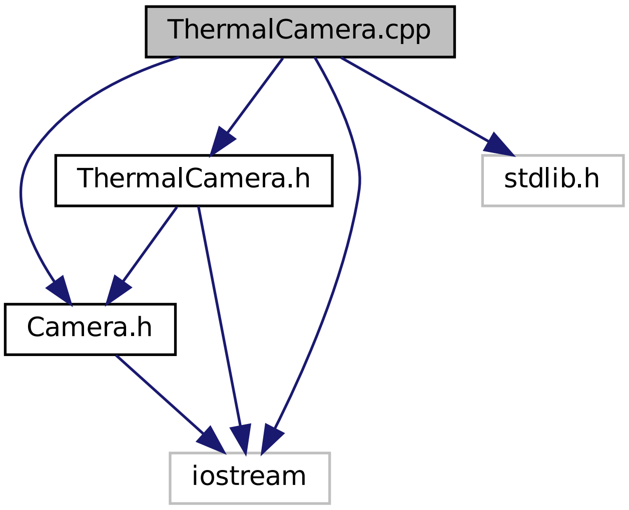
Ana Martínez Albendea

Date

2022-11-23

5.28 ThermalCamera.cpp File Reference

```
#include "ThermalCamera.h"
#include "Camera.h"
#include <iostream>
#include <stdlib.h>
```
Include dependency graph for ThermalCamera.cpp:

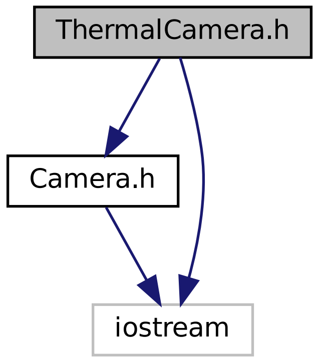


5.29 ThermalCamera.h File Reference

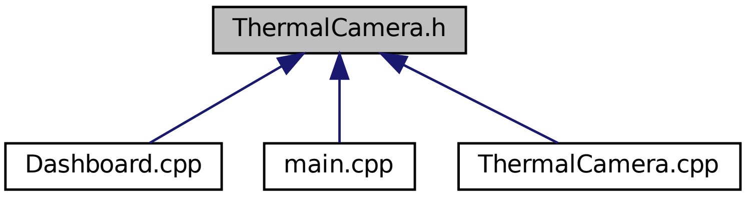
defines the ThermalCamera class, wich is is inherited from the camera class, with its attributes, methods, and constructor

```
#include "Camera.h"
#include <iostream>
```

Include dependency graph for ThermalCamera.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ThermalCamera

5.29.1 Detailed Description

defines the ThermalCamera class, wich is is inherited from the camera class, with its attributes, methods, and constructor

Author

Ana Martínez Albendea

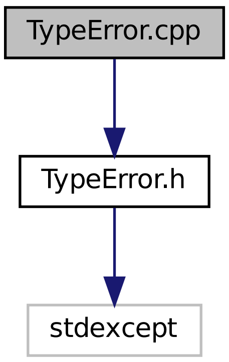
Date

2022-11-23

5.30 TypeError.cpp File Reference

```
#include "TypeError.h"
```

Include dependency graph for TypeError.cpp:

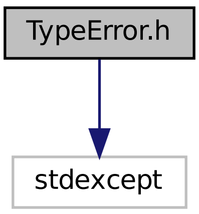


5.31 TypeError.h File Reference

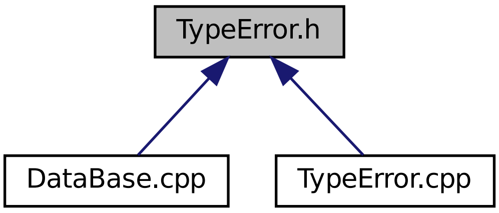
defines the user type exception class with its constructor

```
#include <stdexcept>
```

Include dependency graph for TypeError.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TypeError

5.31.1 Detailed Description

defines the user type exception class with its constructor

Author

Ana Martínez Albendea

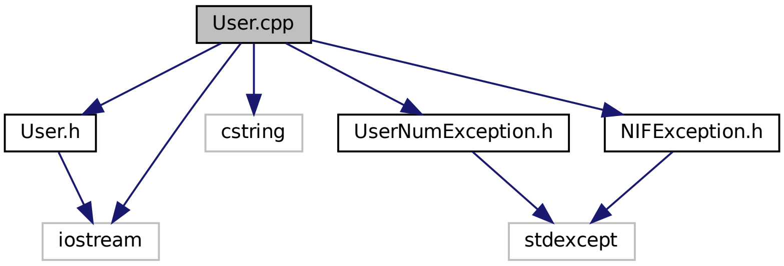
Date

2022-11-23

5.32 User.cpp File Reference

```
#include "User.h"
#include <iostream>
#include <cstring>
#include "UserNumException.h"
#include "NIFException.h"
```

Include dependency graph for User.cpp:

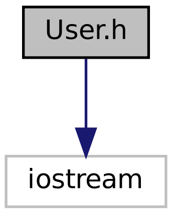


5.33 User.h File Reference

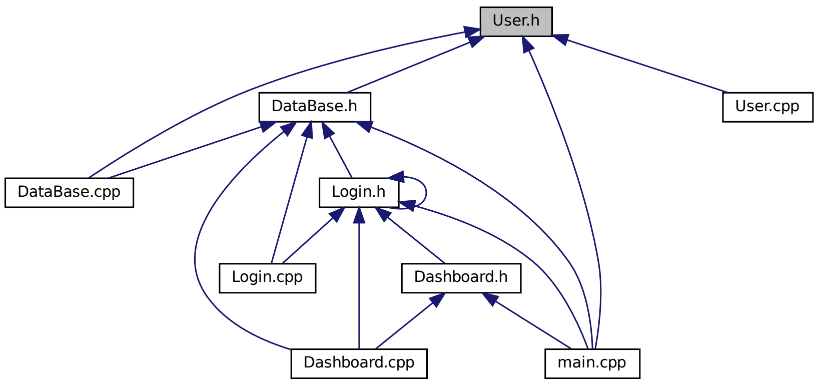
defines the user class with its attributes, methods, and constructor

```
#include <iostream>
```

Include dependency graph for User.h:



This graph shows which files directly or indirectly include this file:



Classes

- class User

5.33.1 Detailed Description

defines the user class with its attributes, methods, and constructor

Author

Ana Martínez Albendea

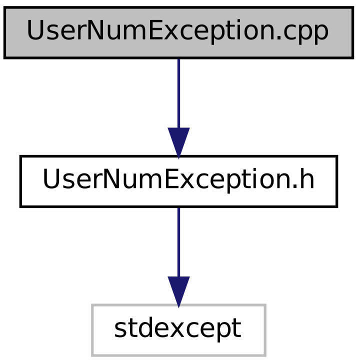
Date

2022-11-23

5.34 UserNumException.cpp File Reference

```
#include "UserNumException.h"
```

Include dependency graph for UserNumException.cpp:

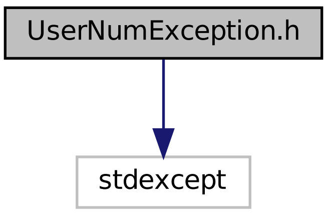


5.35 UserNumException.h File Reference

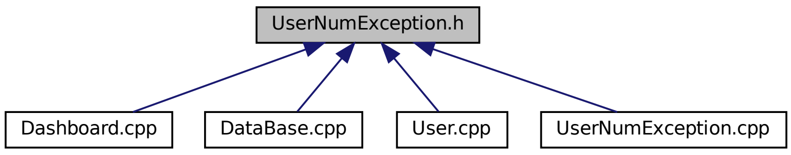
defines the user number exception class with its constructor

```
#include <stdexcept>
```

Include dependency graph for UserNumException.h:



This graph shows which files directly or indirectly include this file:



Classes

- class UserNumException

5.35.1 Detailed Description

defines the user number exception class with its constructor

Author

Ana Martínez Albendea

Date

2022-11-23

Index