

## Julio Veganos e Hijos interface

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AirConditionSensor Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 AirConditionSensor()	8
4.1.3 Member Function Documentation	9
4.1.3.1 getInfo()	9
4.1.4 Member Data Documentation	9
4.1.4.1 airCondition	9
4.2 Camera Class Reference	10
4.2.1 Detailed Description	11
4.2.2 Constructor & Destructor Documentation	11
4.2.2.1 Camera()	11
4.2.3 Member Function Documentation	11
4.2.3.1 getInfo()	11
4.2.3.2 turnOff()	12
4.2.3.3 turnOn()	12
4.2.4 Member Data Documentation	13
4.2.4.1 state	13
4.3 Dashboard Class Reference	13
4.3.1 Detailed Description	14
4.3.2 Constructor & Destructor Documentation	14
4.3.2.1 Dashboard() [1/2]	14
4.3.2.2 Dashboard() [2/2]	14
4.3.3 Member Function Documentation	15
4.3.3.1 getDashboard()	15
4.3.3.2 getOption()	15
4.3.3.3 operator=()	16
4.3.3.4 showCameraChoices()	16
4.3.3.5 showMenu()	17
4.3.3.6 showSensorChoices()	18
4.3.4 Member Data Documentation	19
4.3.4.1 A	19
4.3.4.2 choice	20

4.3.4.3 D	20
4.3.4.4 H	20
4.3.4.5 L	20
4.3.4.6 Li	20
4.3.4.7 M	20
4.3.4.8 option	21
4.3.4.9 RC	21
4.3.4.10 singleDashboard	21
4.3.4.11 T	21
4.3.4.12 TC	21
4.4 DataBase Class Reference	22
4.4.1 Detailed Description	23
4.4.2 Constructor & Destructor Documentation	23
4.4.2.1 DataBase()	23
4.4.3 Member Function Documentation	24
4.4.3.1 addUser()	24
4.4.3.2 checkUser()	25
4.4.3.3 deleteUser()	26
4.4.3.4 saveFile()	26
4.4.3.5 showAdminChoices()	27
4.4.3.6 showUsers()	27
4.4.4 Member Data Documentation	28
4.4.4.1 choice	28
4.4.4.2 dataBaseUser	28
4.4.4.3 isAdmin	28
4.4.4.4 it	29
4.4.4.5 name	29
4.4.4.6 NIF	29
4.4.4.7 NIFStr	29
4.4.4.8 type	29
4.4.4.9 userDeleted	29
4.4.4.10 userNow	30
4.4.4.11 userNumber	30
4.4.4.12 userNumberStr	30
4.5 HumiditySensor Class Reference	30
4.5.1 Detailed Description	31
4.5.2 Constructor & Destructor Documentation	31
4.5.2.1 HumiditySensor()	31
4.5.3 Member Function Documentation	32
4.5.3.1 getInfo()	32
4.5.4 Member Data Documentation	32
4.5.4.1 humidity	32

4.6 LigthLevelSensor Class Reference	33
4.6.1 Detailed Description	34
4.6.2 Constructor & Destructor Documentation	34
4.6.2.1 LigthLevelSensor()	34
4.6.3 Member Function Documentation	34
4.6.3.1 getInfo()	34
4.6.4 Member Data Documentation	34
4.6.4.1 ligthLevel	35
4.7 Login Class Reference	35
4.7.1 Detailed Description	36
4.7.2 Constructor & Destructor Documentation	36
4.7.2.1 Login()	36
4.7.3 Member Function Documentation	36
4.7.3.1 checkLogin()	37
4.7.4 Member Data Documentation	37
4.7.4.1 NIF	37
4.7.4.2 NIFStr	38
4.7.4.3 userNow	38
4.7.4.4 userNowPtr	38
4.7.4.5 userNumber	38
4.7.4.6 userNumberStr	38
4.8 Microphone Class Reference	39
4.8.1 Detailed Description	39
4.8.2 Constructor & Destructor Documentation	39
4.8.2.1 Microphone()	40
4.8.3 Member Function Documentation	40
4.8.3.1 getSound()	40
4.8.3.2 showChoice()	40
4.8.3.3 turnOff()	41
4.8.3.4 turnOn()	41
4.8.4 Member Data Documentation	41
4.8.4.1 choice	42
4.8.4.2 sound	42
4.8.4.3 state	42
4.9 NIFException Class Reference	42
4.9.1 Detailed Description	43
4.9.2 Constructor & Destructor Documentation	43
4.9.2.1 NIFException()	43
4.10 RGBCamera Class Reference	44
4.10.1 Detailed Description	45
4.10.2 Constructor & Destructor Documentation	45
4.10.2.1 RGBCamera()	45

4.10.3 Member Function Documentation	45
4.10.3.1 getInfo()	45
4.11 Sensor Class Reference	46
4.11.1 Detailed Description	46
4.11.2 Constructor & Destructor Documentation	47
4.11.2.1 Sensor()	47
4.11.3 Member Function Documentation	47
4.11.3.1 getInfo()	47
4.11.3.2 turnOff()	48
4.11.3.3 turnOn()	48
4.11.4 Member Data Documentation	49
4.11.4.1 state	49
4.12 StringException Class Reference	49
4.12.1 Detailed Description	50
4.12.2 Constructor & Destructor Documentation	50
4.12.2.1 StringException()	50
4.13 TemperatureSensor Class Reference	51
4.13.1 Detailed Description	52
4.13.2 Constructor & Destructor Documentation	52
4.13.2.1 TemperatureSensor()	52
4.13.3 Member Function Documentation	52
4.13.3.1 getInfo()	52
4.13.4 Member Data Documentation	52
4.13.4.1 temperature	53
4.14 ThermalCamera Class Reference	53
4.14.1 Detailed Description	54
4.14.2 Constructor & Destructor Documentation	54
4.14.2.1 ThermalCamera()	54
4.14.3 Member Function Documentation	54
4.14.3.1 getInfo()	55
4.15 TypeError Class Reference	55
4.15.1 Detailed Description	56
4.15.2 Constructor & Destructor Documentation	56
4.15.2.1 TypeError()	56
4.16 User Class Reference	57
4.16.1 Detailed Description	57
4.16.2 Constructor & Destructor Documentation	58
4.16.2.1 User() [1/2]	58
4.16.2.2 User() [2/2]	58
4.16.3 Member Function Documentation	58
4.16.3.1 checkAdmin()	58
4.16.3.2 getNIF()	59

4.16.3.3 getUsername()	59
4.16.3.4 getUserNum()	59
4.16.3.5 operator<()	60
4.16.3.6 showUser()	60
4.16.4 Member Data Documentation	61
4.16.4.1 isAdmin	61
4.16.4.2 name	61
4.16.4.3 NIF	61
4.16.4.4 userNumber	61
4.17 UserNumException Class Reference	62
4.17.1 Detailed Description	63
4.17.2 Constructor & Destructor Documentation	63
4.17.2.1 UserNumException()	63
<b>5 File Documentation</b>	<b>65</b>
5.1 AirConditionSensor.cpp File Reference	65
5.2 AirConditionSensor.h File Reference	65
5.2.1 Detailed Description	66
5.3 Camera.cpp File Reference	67
5.4 Camera.h File Reference	67
5.4.1 Detailed Description	68
5.5 Dashboard.cpp File Reference	68
5.6 Dashboard.h File Reference	69
5.6.1 Detailed Description	70
5.7 DataBase.cpp File Reference	70
5.8 DataBase.h File Reference	70
5.8.1 Detailed Description	71
5.9 HumiditySensor.cpp File Reference	72
5.10 HumiditySensor.h File Reference	72
5.10.1 Detailed Description	73
5.11 LigthLevelSensor.cpp File Reference	74
5.12 LigthLevelSensor.h File Reference	74
5.12.1 Detailed Description	75
5.13 Login.cpp File Reference	76
5.14 Login.h File Reference	76
5.14.1 Detailed Description	77
5.15 main.cpp File Reference	78
5.15.1 Function Documentation	78
5.15.1.1 main()	79
5.16 Microphone.cpp File Reference	80
5.17 Microphone.h File Reference	80
5.17.1 Detailed Description	81

---

5.18 NIFException.cpp File Reference . . . . .	81
5.19 NIFException.h File Reference . . . . .	82
5.19.1 Detailed Description . . . . .	82
5.20 RGBCamera.cpp File Reference . . . . .	83
5.21 RGBCamera.h File Reference . . . . .	83
5.21.1 Detailed Description . . . . .	84
5.22 Sensor.cpp File Reference . . . . .	85
5.23 Sensor.h File Reference . . . . .	85
5.23.1 Detailed Description . . . . .	86
5.24 StringException.cpp File Reference . . . . .	86
5.25 StringException.h File Reference . . . . .	86
5.25.1 Detailed Description . . . . .	87
5.26 TemperatureSensor.cpp File Reference . . . . .	88
5.27 TemperatureSensor.h File Reference . . . . .	88
5.27.1 Detailed Description . . . . .	89
5.28 ThermalCamera.cpp File Reference . . . . .	90
5.29 ThermalCamera.h File Reference . . . . .	90
5.29.1 Detailed Description . . . . .	91
5.30 TypeError.cpp File Reference . . . . .	92
5.31 TypeError.h File Reference . . . . .	92
5.31.1 Detailed Description . . . . .	93
5.32 User.cpp File Reference . . . . .	93
5.33 User.h File Reference . . . . .	94
5.33.1 Detailed Description . . . . .	94
5.34 UserNumException.cpp File Reference . . . . .	95
5.35 UserNumException.h File Reference . . . . .	95
5.35.1 Detailed Description . . . . .	96
<b>Index</b>	<b>97</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Camera . . . . .	10
RGBCamera . . . . .	44
ThermalCamera . . . . .	53
Dashboard . . . . .	13
DataBase . . . . .	22
Login . . . . .	35
Microphone . . . . .	39
std::runtime_error	
NIFException . . . . .	42
StringException . . . . .	49
TypeError . . . . .	55
UserNumException . . . . .	62
Sensor . . . . .	46
AirConditionSensor . . . . .	7
HumiditySensor . . . . .	30
LigthLevelSensor . . . . .	33
TemperatureSensor . . . . .	51
User . . . . .	57



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AirConditionSensor</a>	7
<a href="#">Camera</a>	10
<a href="#">Dashboard</a>	13
<a href="#">DataBase</a>	22
<a href="#">HumiditySensor</a>	30
<a href="#">LigthLevelSensor</a>	33
<a href="#">Login</a>	35
<a href="#">Microphone</a>	39
<a href="#">NIFException</a>	42
<a href="#">RGBCamera</a>	44
<a href="#">Sensor</a>	46
<a href="#">StringException</a>	49
<a href="#">TemperatureSensor</a>	51
<a href="#">ThermalCamera</a>	53
<a href="#">TypeError</a>	55
<a href="#">User</a>	57
<a href="#">UserNumException</a>	62



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AirConditionSensor.cpp</a>	65
<a href="#">AirConditionSensor.h</a>	
Defines the <a href="#">AirConditionSensor</a> class, wich is is inherited from the sensor class, with its attributes, methods, and constructor	65
<a href="#">Camera.cpp</a>	67
<a href="#">Camera.h</a>	
Defines the camera class with its attributes, methods, and constructor	67
<a href="#">Dashboard.cpp</a>	68
<a href="#">Dashboard.h</a>	
Defines the dashboard class with its attributes, methods, and constructor	69
<a href="#">DataBase.cpp</a>	70
<a href="#">DataBase.h</a>	
Defines the database class with its attributes, methods, and constructor	70
<a href="#">HumiditySensor.cpp</a>	72
<a href="#">HumiditySensor.h</a>	
Defines the <a href="#">HumiditySensor</a> class, wich is is inherited from the sensor class, with its attributes, methods, and constructor	72
<a href="#">LigthLevelSensor.cpp</a>	74
<a href="#">LigthLevelSensor.h</a>	
Defines the <a href="#">LigthLevelSensor</a> class, wich is is inherited from the sensor class, with its attributes, methods, and constructor	74
<a href="#">Login.cpp</a>	76
<a href="#">Login.h</a>	
Defines the login class with its attributes, methods, and constructor	76
<a href="#">main.cpp</a>	78
<a href="#">Microphone.cpp</a>	80
<a href="#">Microphone.h</a>	
Defines the microphone class with its attributes, methods, and constructor	80
<a href="#">NIFException.cpp</a>	81
<a href="#">NIFException.h</a>	
Defines the NIF exception class with its constructor	82
<a href="#">RGBCamera.cpp</a>	83
<a href="#">RGBCamera.h</a>	
Defines the <a href="#">RGBCamera</a> class, wich is is inherited from the camera class, with its attributes, methods, and constructor	83

<a href="#">Sensor.cpp</a>	85
<a href="#">Sensor.h</a>	
Defines the sensor class with its attributes, methods, and constructor	85
<a href="#">StringException.cpp</a>	86
<a href="#">StringException.h</a>	
Defines the string exception class with its constructor	86
<a href="#">TemperatureSensor.cpp</a>	88
<a href="#">TemperatureSensor.h</a>	
Defines the <a href="#">TemperatureSensor</a> class, which is inherited from the sensor class, with its attributes, methods, and constructor	88
<a href="#">ThermalCamera.cpp</a>	90
<a href="#">ThermalCamera.h</a>	
Defines the <a href="#">ThermalCamera</a> class, which is inherited from the camera class, with its attributes, methods, and constructor	90
<a href="#">TypeError.cpp</a>	92
<a href="#">TypeError.h</a>	
Defines the user type exception class with its constructor	92
<a href="#">User.cpp</a>	93
<a href="#">User.h</a>	
Defines the user class with its attributes, methods, and constructor	94
<a href="#">UserNumException.cpp</a>	95
<a href="#">UserNumException.h</a>	
Defines the user number exception class with its constructor	95

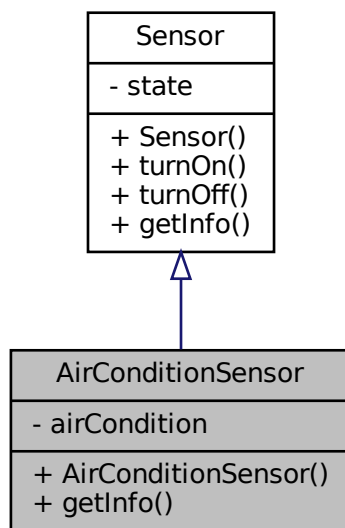
## Chapter 4

# Class Documentation

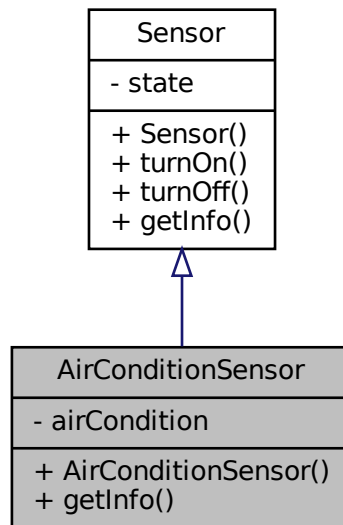
### 4.1 AirConditionSensor Class Reference

```
#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.

```
11     {  
12         srand(time(NULL)+3);  
13         airCondition = rand()%301;  
14         cout << "\nCurrent air condition level: " << airCondition << " ppm" << endl;  
15     }
```

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\(\)](#).

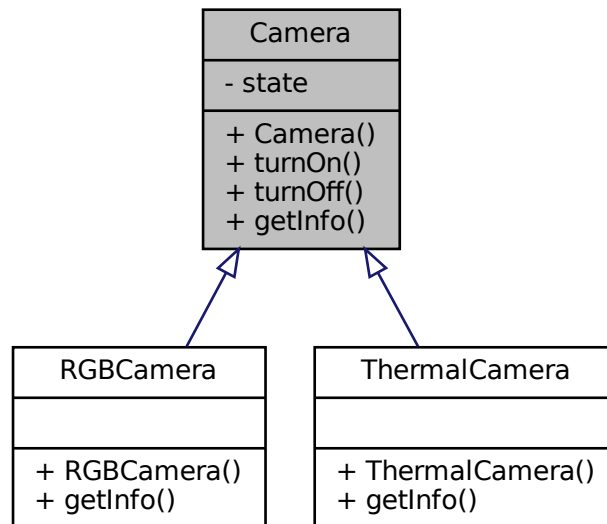
The documentation for this class was generated from the following files:

- [AirConditionSensor.h](#)
- [AirConditionSensor.cpp](#)

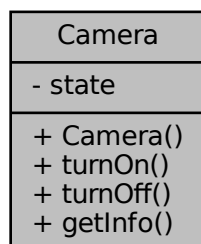
## 4.2 Camera Class Reference

```
#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)

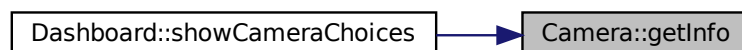
Reimplemented in [ThermalCamera](#), and [RGBCamera](#).

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 ( )
```

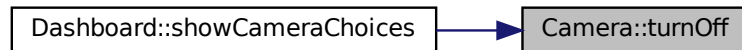
Depending on the state of the camera, we turn it off and indicate its state

Definition at line 17 of file Camera.cpp.

```
17     {  
18     //Depending on the state of the camera, we turn it off and indicate its state  
19     if (state == true){  
20         cout << "\nCamera status: Off" << endl;  
21         state = false;  
22     }else{  
23         cout << "\nThe camera is already off" << endl;  
24     }  
25 }
```

Referenced by Dashboard::showCameraChoices().

Here is the caller graph for this function:



#### 4.2.3.3 turnOn()

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

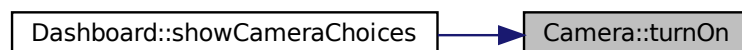
Depending on the state of the camera, we turn it on and indicate its state

Definition at line 7 of file Camera.cpp.

```
7     {  
8     //Depending on the state of the camera, we turn it on and indicate its state  
9     if (state == false){  
10         cout << "\nCamera status: On" << endl;  
11         state = true;  
12     }else{  
13         cout << "\nThe camera is already on" << endl;  
14     }  
15 }
```

Referenced by Dashboard::showCameraChoices().

Here is the caller graph for this function:



## 4.2.4 Member Data Documentation

### 4.2.4.1 state

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

Definition at line 37 of file Camera.h.

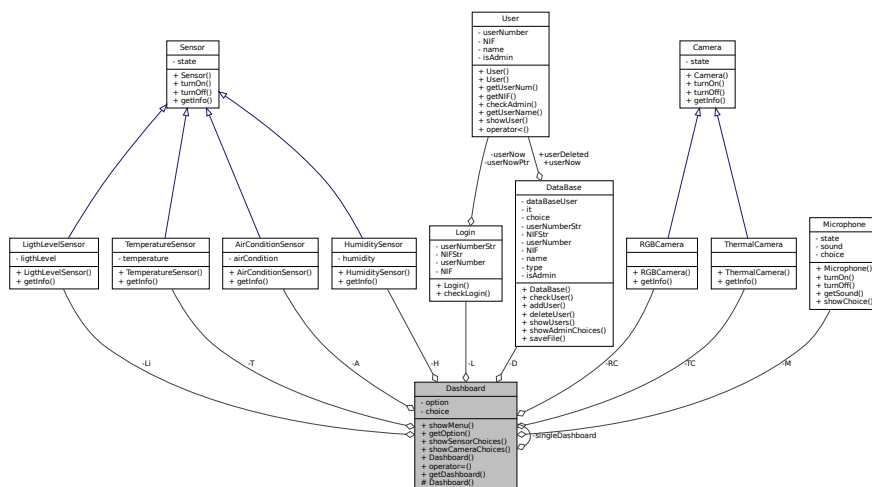
The documentation for this class was generated from the following files:

- [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](#).

```
20         { //we define the constructor
21     this -> option = option;
22 }
```

### 4.3.3 Member Function Documentation

#### 4.3.3.1 getDashboard()

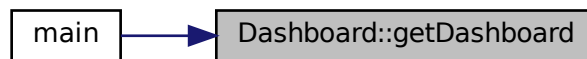
```
Dashboard * Dashboard::getDashboard ( ) [static]
```

Definition at line 225 of file Dashboard.cpp.

```
225 {
226     if(singleDashboard == nullptr){
227         singleDashboard = new Dashboard();
228     }else{
229         cout << "Error: trying to get another instance of a Ball singleton class!\n";
230     }
231
232     return singleDashboard;
233 }
```

Referenced by main().

Here is the caller graph for this function:



#### 4.3.3.2 getOption()

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

Depending on the option entered, performs an action

Definition at line 119 of file Dashboard.cpp.

```
119 {
120     //depending on the option entered, performs an action
121     cout << "Enter the number of the action you want to do: ";
122     cin >> option;
123
124     switch (option)
125     {
126     case 1:
127         system("clear");
128         showSensorChoices(T);
129         break;
130
131     case 2:
132         system("clear");
133         showSensorChoices(H);
134         break;
135
136     case 3:
137         system("clear");
138         showSensorChoices(Li);
139         break;
```





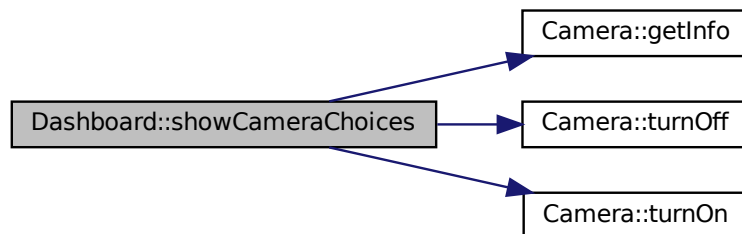
```

85         « "                                     3. SHOW DATA INFO" « endl
86         « "                                     4. BACK TO MENU" « endl;
87     cout << "\nEnter your choice number: ";
88     cin >> choice;
89
90     //Depending on the option that is entered, it calls its respective function
91     switch (choice)
92     {
93     case 1:
94         camera->turnOn();
95         system("sleep 3");
96         system("clear");
97         break;
98
99     case 2:
100        camera->turnOff();
101        system("sleep 3");
102        system("clear");
103        break;
104
105     case 3:
106        camera->getInfo();
107        system("sleep 3");
108        system("clear");
109        break;
110
111     case 4:
112        //go back to the main menu
113        return;
114        break;
115     }
116 }
117 }

```

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`.

```

178     {
179     try{
180         //displays the login screen and, depending on whether the user exists, displays the main menu

```



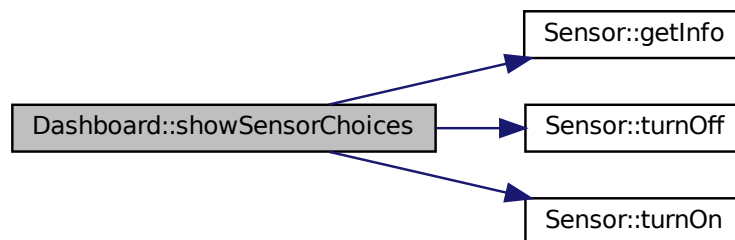
```

36         « "                                2. TURN OFF" « endl
37         « "                                3. SHOW DATA INFO" « endl
38         « "                                4. BACK TO MENU" « endl;
39     cout << "\nEnter your choice number: ";
40     cin >> choice;
41
42     //Depending on the option that is entered, it calls its respective function
43     switch (choice)
44     {
45     case 1:
46         sensor->turnOn();
47         system("sleep 3");
48         system("clear");
49         break;
50
51     case 2:
52         sensor->turnOff();
53         system("sleep 3");
54         system("clear");
55         break;
56
57     case 3:
58         sensor->getInfo();
59         cout.flush();
60         system("sleep 3");
61         system("clear");
62         break;
63
64     case 4:
65         //go back to the main menu
66         return;
67         break;
68     }
69 }
70 }

```

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.

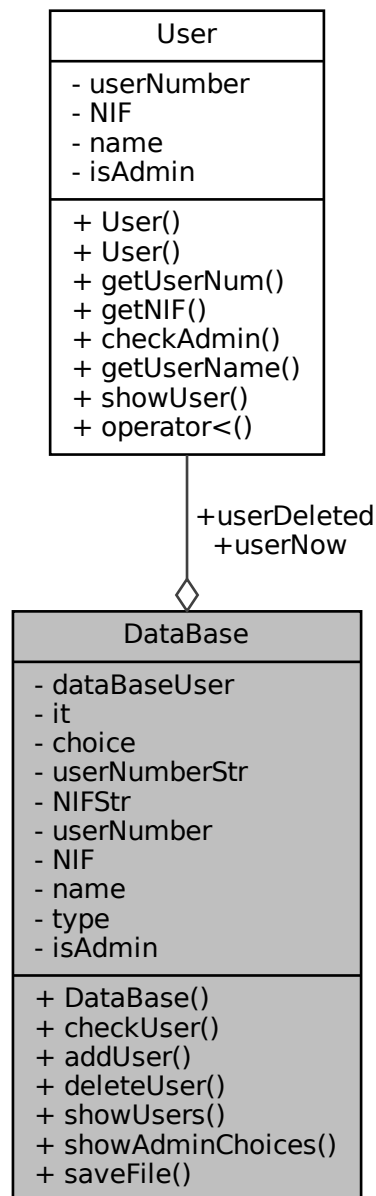
The documentation for this class was generated from the following files:

- [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`.

```

17         { //we define the constructor with the initial users that we are gonna have
18     /*this->dataBaseUser.insert(User(1,13172409,"Ana",true));
19     this->dataBaseUser.insert(User(2,13172408,"Paula",false));
20     this->dataBaseUser.insert(User(3,12345678,"Carlos",false));*/
21
22     // Reads the users found in the users.dat file and dumps them into the database set
23     ifstream inUsersFile ("users.dat", ios::in | ios::binary);
24
25     if (!inUsersFile) { // ifstream could not open file
26         cerr << "File could not be opened." << endl;
27         exit (1);
28     }
29
30     User user;
31     inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
32     while (inUsersFile && !inUsersFile.eof()) {
33         this->dataBaseUser.insert(user);
34         inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
35     }
36 }
```

### 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.

```
55
56 //checks that both the user number and password are the correct size
57 try{
58     for(int n = 0; n < userNumberStr.length(); n++){
59         if(int(userNumberStr[n]) < 47 || int(userNumberStr[n] > 57)){
60             throw StringException();
61         }
62         userNumber = stoi(userNumberStr);
63     }
64
65     if (userNumber < 1 || userNumber > 99999){
66         throw UserNumException();
67     }
68
69     for(int n = 0; n < NIFStr.length(); n++){
70         if(int(NIFStr[n]) < 47 || int(NIFStr[n] > 57)){
71             throw StringException();
72         }
73         NIF = stoi(NIFStr);
74     }
75
76     if (NIF < 9999999 || NIF > 99999999){
77         throw NIFException();
78     }
79
80     if(isAdmin != 1 && isAdmin != 0){
81         throw TypeError();
82     }
83
84     //adds a new user that is entered by an admin in the terminal
85     this->dataBaseUser.insert(User(userNumber, NIF, name, isAdmin));
86 }
87
88 catch(UserNumException &except){
89     cout << "Exception: " << except.what() << endl;
90 }
91
92 catch(NIFException &except){
93     cout << "Exception: " << except.what() << endl;
94 }
95
96 catch(TypeError &except){
97     cout << "Exception: " << except.what() << endl;
98 }
99
100 catch(StringException &except){
101     cout << "Exception: " << except.what() << endl;
102 }
103 }
```



#### 4.4.3.2 checkUser()

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

Checks if the user exists

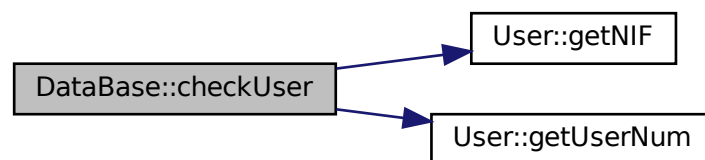
Definition at line 38 of file DataBase.cpp.

```
38 {
39     bool authentication = false;
40     //checks if the user exists
41     do{
42         for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
43             User user = *it;
44             if(userNumber==user.getUserNum() && NIF==user.getNIF()){
45                 userNow = user;
46                 return true;
47                 authentication = true;
48             }
49         }
50     }while(authentication == false);
51     return false;
52 }
53 }
```

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.

```
105 {
106     //deletes the user that is entered by an admin in the terminal
107     for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
108         User user = *it;
109         if(userNumber==user.getUserNum()){
110             userDeleted = user;
111         }
112     }
113     this->dataBaseUser.erase(userDeleted);
114     cout << "El usuario ha sido eliminado" << endl;
115 }
```

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.

```
181 {
182     fstream outUsersFile ("users.dat", ios::in | ios::out | ios::trunc | ios::binary); // ios::in will
183     require an existing file
184     // Uses: "users.dat", generated in previous example
185     if (!outUsersFile) { // fstream could not open file
186         cerr << "File could not be opened." << endl;
187         exit (1);
188     }
189     int position = 0;
190
191     for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
192         User user = *it;
193         outUsersFile.seekp (position * sizeof (User));
194         outUsersFile.write (reinterpret_cast <const char *> (&user), sizeof (User));
195         position ++;
196     }
197 }
```

## 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.

```

130         {
131     //displays the manage users options menu, that only an Admin can use
132     while (true) {
133         cout << "\n\n                                MANAGE USERS" << endl
134         << "-----" << endl;
135         cout << "                                1. ADD USER" << endl
136             << "                                2. DELETE USER" << endl
137             << "                                3. SHOW USERS LIST" << endl
138             << "                                4. BACK TO MENU" << endl;
139         cout << "Enter your choice number: ";
140         cin >> choice;
141
142         switch (choice)
143         {
144         case 1:
145             cout << "Enter a new user" << endl;
146             cout << "User number: ";
147             cin >> userNumberStr;
148             cout << "User password: ";
149             cin >> NIFStr;
150             cout << "User name: ";
151             cin >> name;
152             cout << "User type (1=Admin, 0=User): ";
153             cin >> type;
154
155             if (type == 1){
156                 isAdmin = true;
157             }else{ isAdmin = false; }
158
159             addUser(userNumberStr, NIFStr, name, isAdmin);
160             break;
161
162         case 2:
163             cout << "Enter the user number that you want to delete: ";
164             cin >> userNumber;
165             deleteUser(userNumber);
166             break;
167
168         case 3:
169             cout << "USERS LIST: " << endl;
170             cout << "N°\tNAME\t\tTYPE" << endl;
171             showUsers();
172             break;
173
174         case 4:
175             return;
176             break;
177         }
178     }
179 }
```

## 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.

```

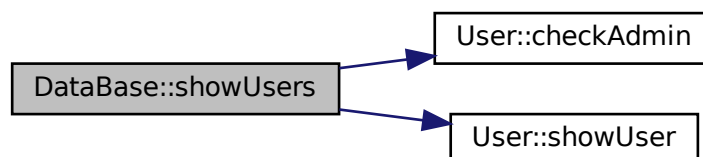
117         {
118     //shows the users that are saved in the database
119     for(it=dataBaseUser.begin(); it!=dataBaseUser.end(); it++){
120         User user = *it;
121         user.showUser();
122         if(user.checkAdmin()){
123             cout << "\t\tAdmin" << endl;
124         }else{
125             cout << "\t\tEmployee" << endl;

```

```
126     }  
127 }  
128 }
```

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.

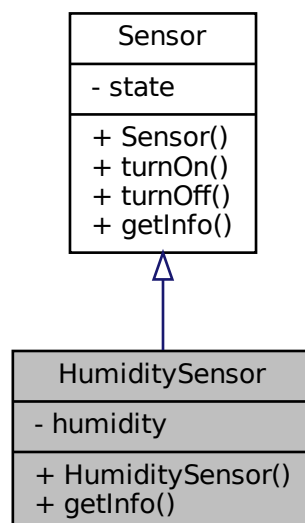
The documentation for this class was generated from the following files:

- [DataBase.h](#)
- [DataBase.cpp](#)

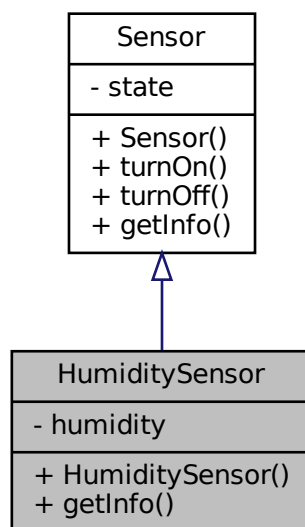
## 4.5 HumiditySensor Class Reference

```
#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.

```
11     {  
12         srand(time(NULL)+1);  
13         humidity = rand()%101;  
14         cout << "\nCurrent humidity: " << humidity << "%" << endl;  
15     }
```

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:

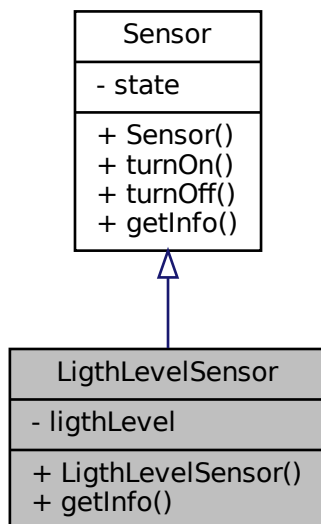
- [HumiditySensor.h](#)
- [HumiditySensor.cpp](#)



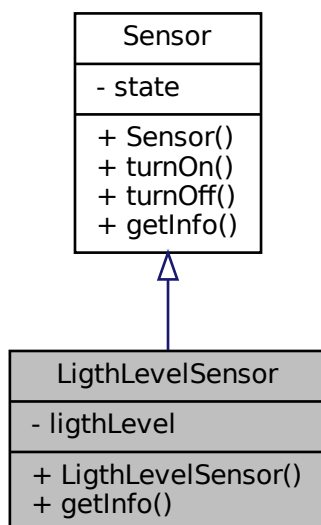
## 4.6 LigthLevelSensor Class Reference

```
#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.

```
9 :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.

```
11 {
12     srand(time(NULL)+2);
13     ligthLevel = rand()%101;
14     cout << "\nCurrent ligth level: " << ligthLevel << "%" << endl;
15 }
```

References [ligthLevel](#).

### 4.6.4 Member Data Documentation

#### 4.6.4.1 lighLevel

```
float LigthLevelSensor::lighLevel [private]
```

Definition at line 29 of file LigthLevelSensor.h.

Referenced by getInfo().

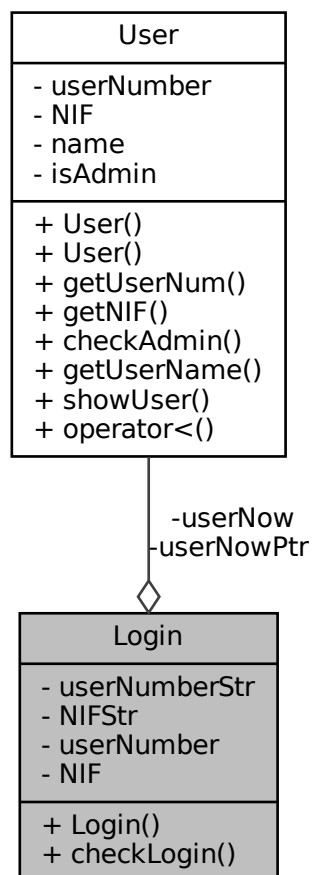
The documentation for this class was generated from the following files:

- [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.

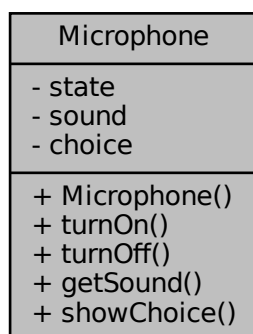
The documentation for this class was generated from the following files:

- [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.

```
7 {} //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.

```
9 {
10     //prints that a command is being listened for
11     cout << sound << endl;
12 }
```

### 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.

```
34 {
35     //displays the microphone options menu
36     while (true) {
37         cout << "\n
38         << "
39         << "
40         << "
41         << "
42         << "
43         << "
44         << "
45         << "
46         << "-----" << endl
47         << ".....\n"
48         << endl;
49         cout << "
50         << "
51         << "
52         cout << "Enter your choice number: ";
53         cin >> choice;
54
55         switch (choice)
56         {
57         case 1:
58             turnOn();
59             system("sleep 3");
60             system("clear");
61             break;
62
63         case 2:
```



```
64     turnOff();
65     system("sleep 3");
66     system("clear");
67     break;
68
69     case 3:
70     getSound();
71     system("sleep 3");
72     system("clear");
73     break;
74
75     case 4:
76     return;
77     break;
78 }
79 }
80 }
```

#### 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.

```
24     {
25     //Depending on the state of the microphone, we turn it off and indicate its state
26     if (state == true){
27         cout << "\nMicrophone status: Off" << endl;
28         state = false;
29     }else{
30         cout << "\nThe microphone is already off" << endl;
31     }
32 }
```

#### 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.

```
14     {
15     //Depending on the state of the microphone, we turn it on and indicate its state
16     if (state == false){
17         cout << "\nMicrophone status: On" << endl;
18         state = true;
19     }else{
20         cout << "\nThe microphone is already on" << endl;
21     }
22 }
```

### 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.

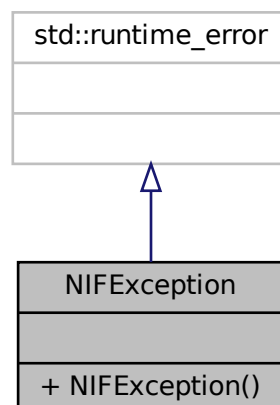
The documentation for this class was generated from the following files:

- [Microphone.h](#)
- [Microphone.cpp](#)

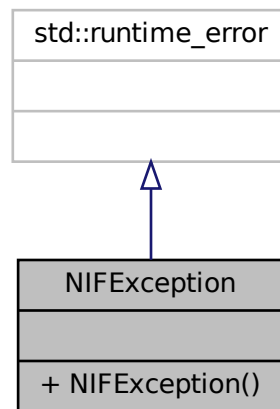
## 4.9 NIFException Class Reference

```
#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."){};
```

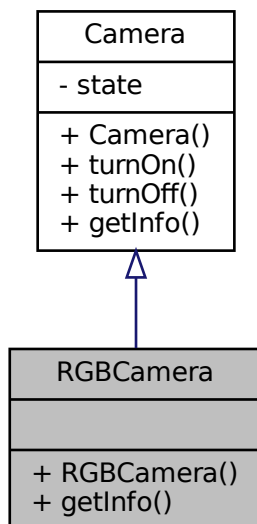
The documentation for this class was generated from the following files:

- [NIFException.h](#)
- [NIFException.cpp](#)

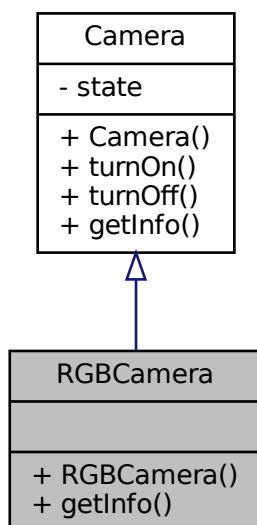
## 4.10 RGBCamera Class Reference

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

Inheritance diagram for RGBCamera:



Collaboration diagram for RGBCamera:

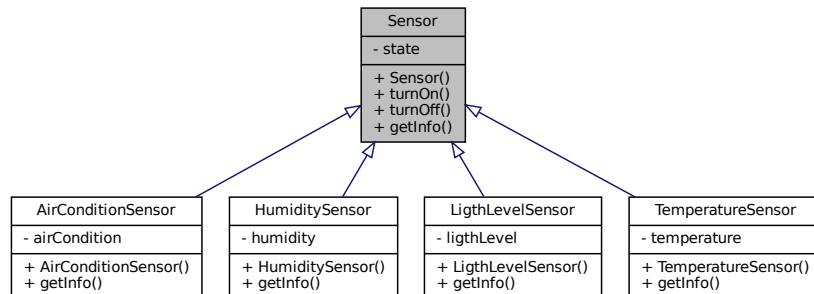




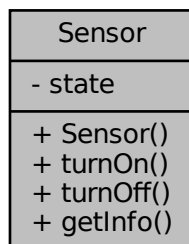
## 4.11 Sensor Class Reference

```
#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)

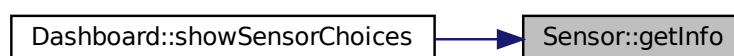
Reimplemented in [TemperatureSensor](#), [LigthLevelSensor](#), [HumiditySensor](#), and [AirConditionSensor](#).

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 ( )
```

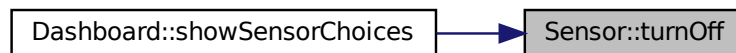
Depending on the state of the sensor, we turn it off and indicate its state

Definition at line 17 of file Sensor.cpp.

```
17     {  
18         //Depending on the state of the sensor, we turn it off and indicate its state  
19         if (state == true){  
20             cout << "\nSensor Status: Off" << endl;  
21             state = false;  
22         }else{  
23             cout << "\nThe sensor is already off" << endl;  
24         }  
25     }
```

Referenced by Dashboard::showSensorChoices().

Here is the caller graph for this function:



#### 4.11.3.3 turnOn()

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

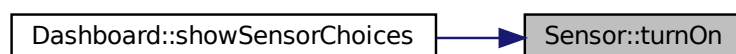
Depending on the state of the sensor, we turn it on and indicate its state

Definition at line 7 of file Sensor.cpp.

```
7     {  
8         //Depending on the state of the sensor, we turn it on and indicate its state  
9         if (state == false){  
10             cout << "\nSensor status: On" << endl;  
11             state = true;  
12         }else{  
13             cout << "\nThe sensor is already on" << endl;  
14         }  
15     }
```

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.

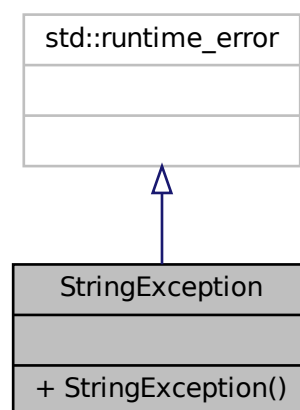
The documentation for this class was generated from the following files:

- [Sensor.h](#)
- [Sensor.cpp](#)

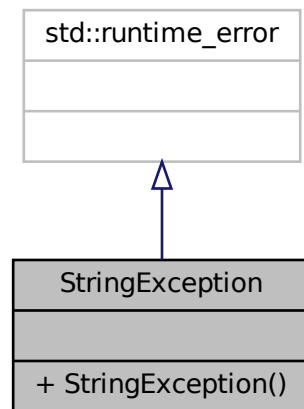
## 4.12 StringException Class Reference

```
#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"){};
```

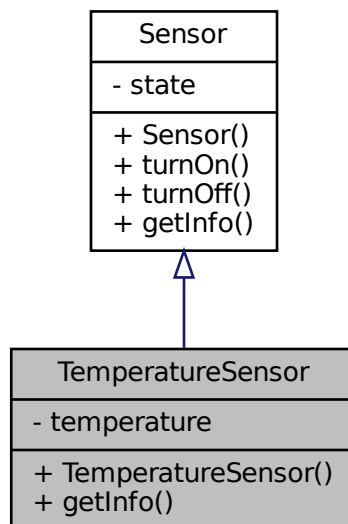
The documentation for this class was generated from the following files:

- [StringException.h](#)
- [StringException.cpp](#)

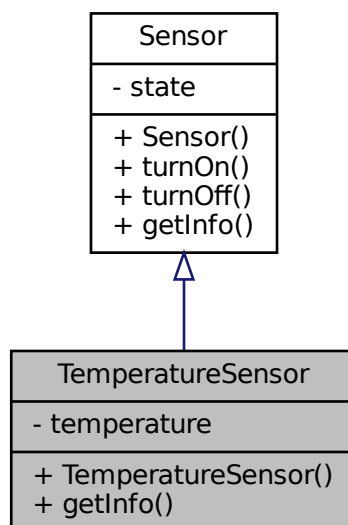
## 4.13 TemperatureSensor Class Reference

```
#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`.

```
9 :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`.

```
11 {
12     srand(time(NULL));
13     temperature = 25+rand()%(40 +1 - 25);
14     cout << "\nCurrent temperature: " << temperature << " °C" << endl;
15 }
```

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()`.

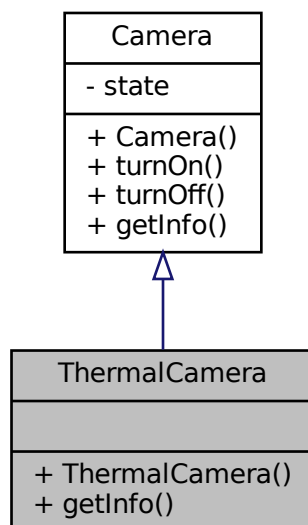
The documentation for this class was generated from the following files:

- [TemperatureSensor.h](#)
- [TemperatureSensor.cpp](#)

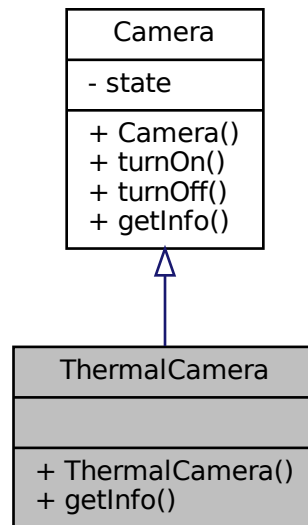
## 4.14 ThermalCamera Class Reference

```
#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 ( )
```

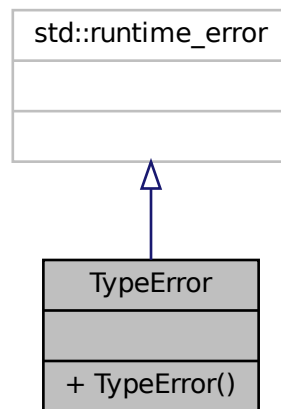
Definition at line 8 of file ThermalCamera.cpp.

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

### 4.14.3 Member Function Documentation



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:

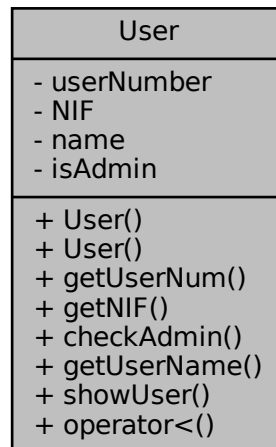
- [TypeError.h](#)
- [TypeError.cpp](#)



## 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.

```
9 {} //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.

```
11                                     { //we define the constructor with
    parameters
12   this-> userNumber = userNumber;
13   this-> NIF = NIF;
14   strcpy(this->name,name.c_str());
15   this-> isAdmin = isAdmin;
16 };
```

## 4.16.3 Member Function Documentation

### 4.16.3.1 checkAdmin()

bool User::checkAdmin ( )

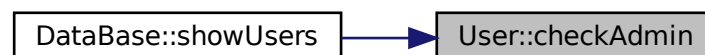
Returns if a user is admin o not by true or false

Definition at line 31 of file User.cpp.

```
31   {
32   return isAdmin;
33 }
```

Referenced by DataBase::showUsers().

Here is the caller graph for this function:



#### 4.16.3.2 getNIF()

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

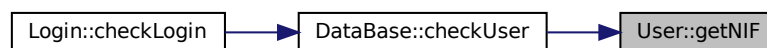
Returns the NIF

Definition at line 24 of file User.cpp.

```
24     {  
25     if (NIF < 99999999 || NIF > 999999999) {  
26         throw NIFException();  
27     }  
28     return NIF;  
29 }
```

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.

```
35     {  
36     return name;  
37 }
```

#### 4.16.3.4 getUserNum()

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

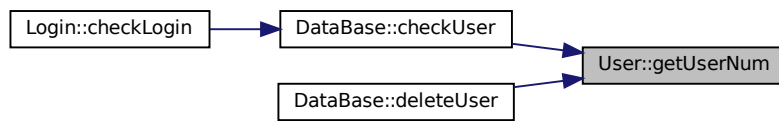
Returns the user number

Definition at line 18 of file User.cpp.

```
18     {  
19     if (userNumber < 1 || userNumber > 99999) {  
20         throw UserNumException();  
21     }  
22     return userNumber;  
23 }
```

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.

```
43 {
44     return userNumber < user.userNumber;
45 }
```

References userNumber.

#### 4.16.3.6 showUser()

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

Returns the user name

Definition at line 39 of file User.cpp.

```
39 {
40     cout << userNumber << "\t" << name;
41 }
```

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<().

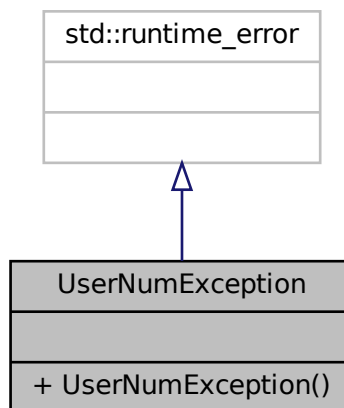
The documentation for this class was generated from the following files:

- [User.h](#)
- [User.cpp](#)

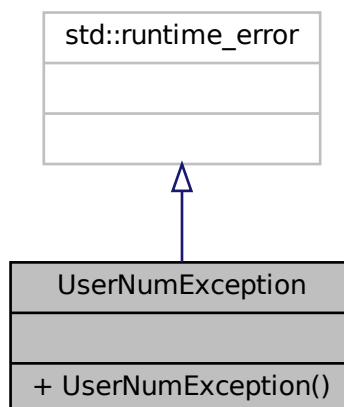
## 4.17 UserNumException Class Reference

```
#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.

```
4 :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](#)





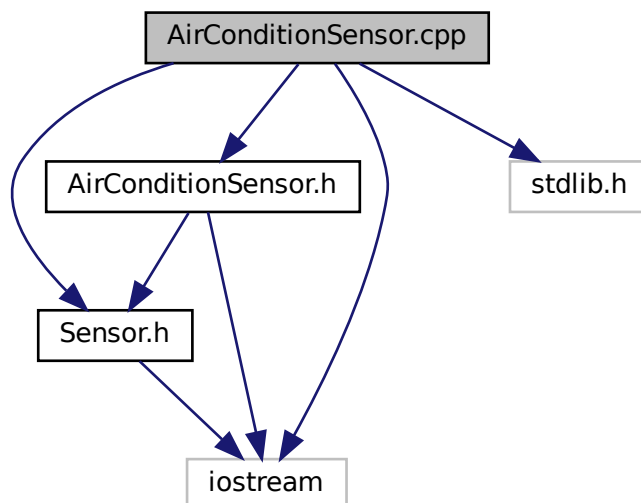
## Chapter 5

# 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:

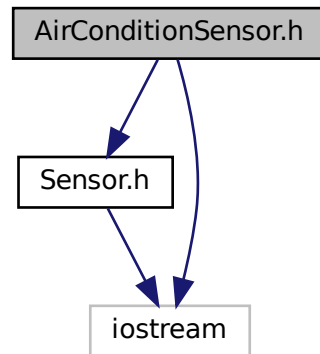


### 5.2 AirConditionSensor.h File Reference

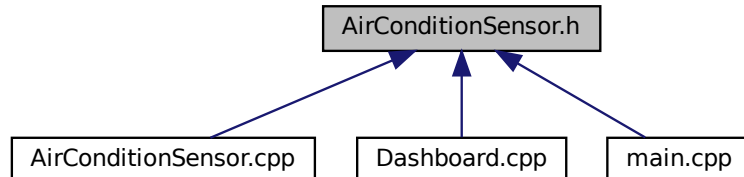
defines the [AirConditionSensor](#) class, which is inherited from the sensor class, with its attributes, methods, and constructor

```
#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, which is inherited from the sensor class, with its attributes, methods, and constructor

#### Author

Ana Martínez Albendea

#### Date

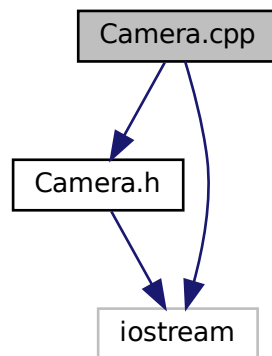
2022-11-23

## 5.3 Camera.cpp File Reference

```
#include "Camera.h"
```

```
#include <iostream>
```

Include dependency graph for Camera.cpp:

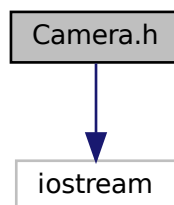


## 5.4 Camera.h File Reference

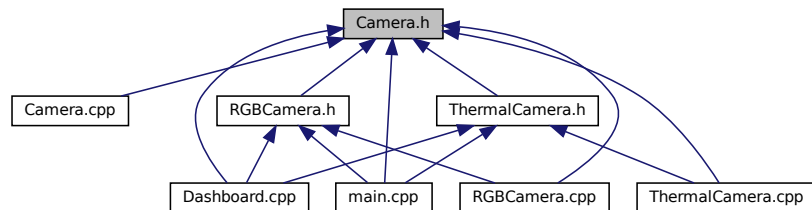
defines the camera class with its attributes, methods, and constructor

```
#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

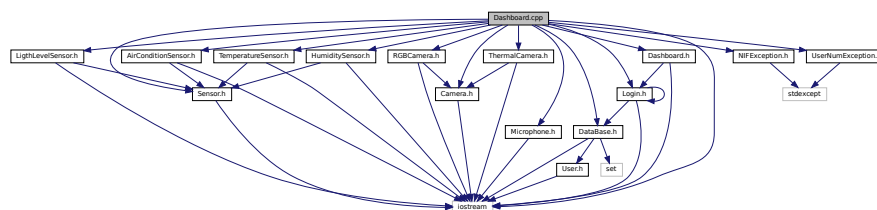
## 5.5 Dashboard.cpp File Reference

```

#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:

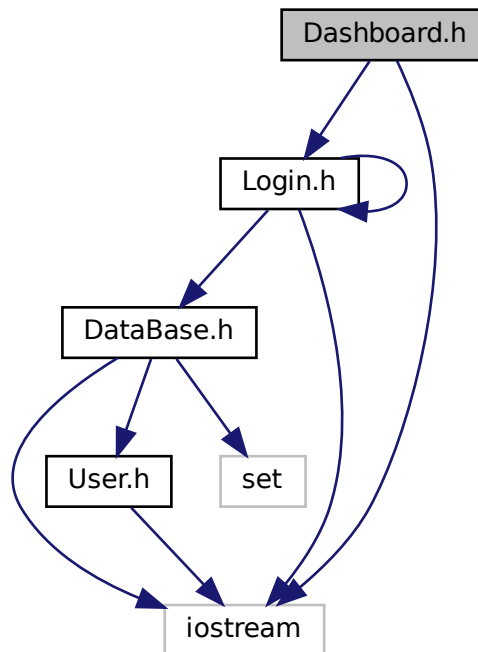


## 5.6 Dashboard.h File Reference

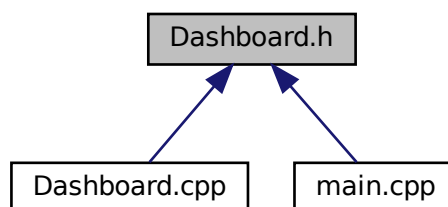
defines the dashboard class with its attributes, methods, and constructor

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

Include dependency graph for Dashboard.h:



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



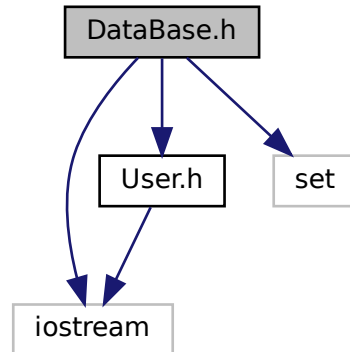
### Classes

- class [Dashboard](#)

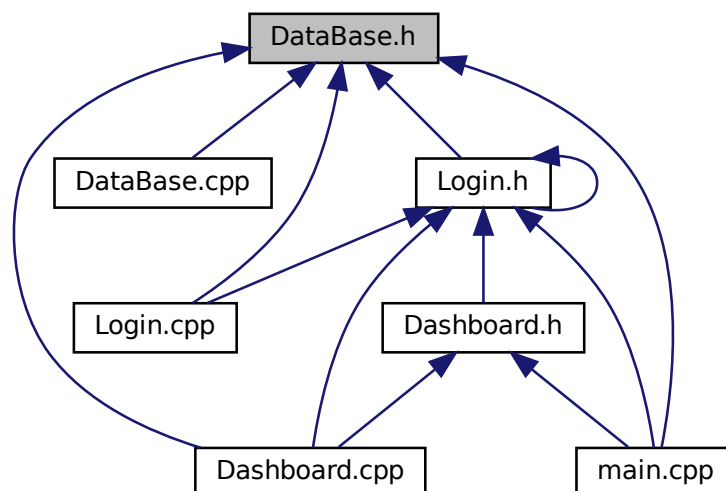


```
#include <set>
```

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

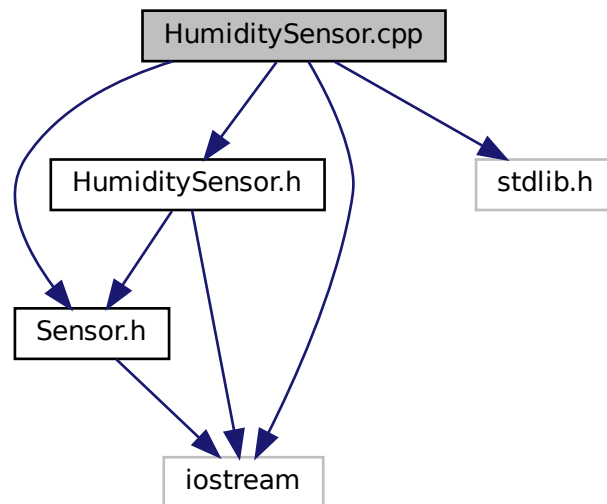
**Date**

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:



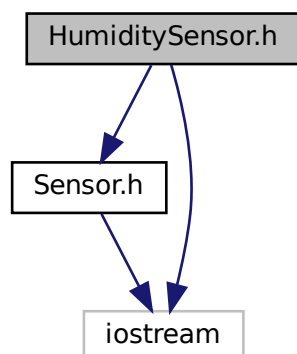
## 5.10 HumiditySensor.h File Reference

defines the [HumiditySensor](#) class, which is inherited from the sensor class, with its attributes, methods, and constructor

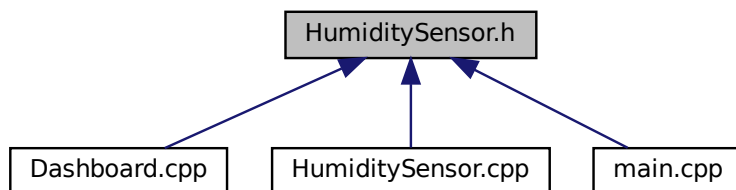
```
#include "Sensor.h"  
#include <iostream>
```



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

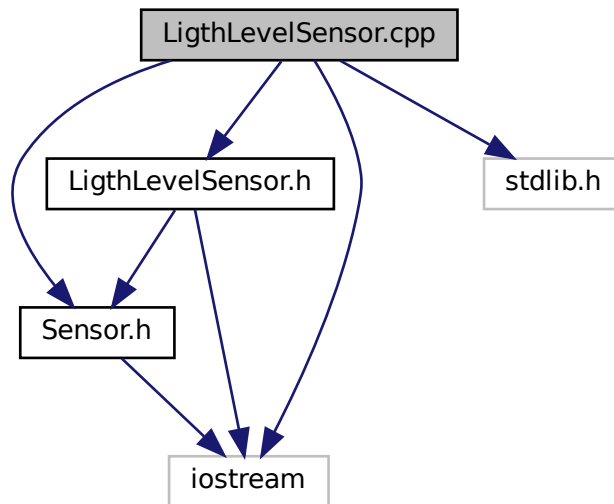
#### Date

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:

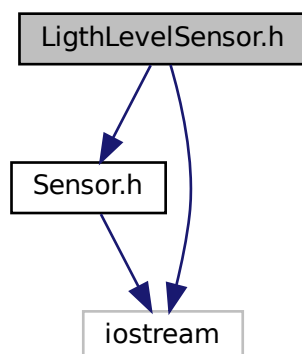


## 5.12 LigthLevelSensor.h File Reference

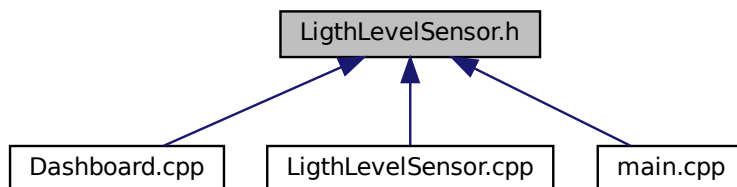
defines the [LigthLevelSensor](#) class, wich is is inherited from the sensor class, with its attributes, methods, and constructor

```
#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

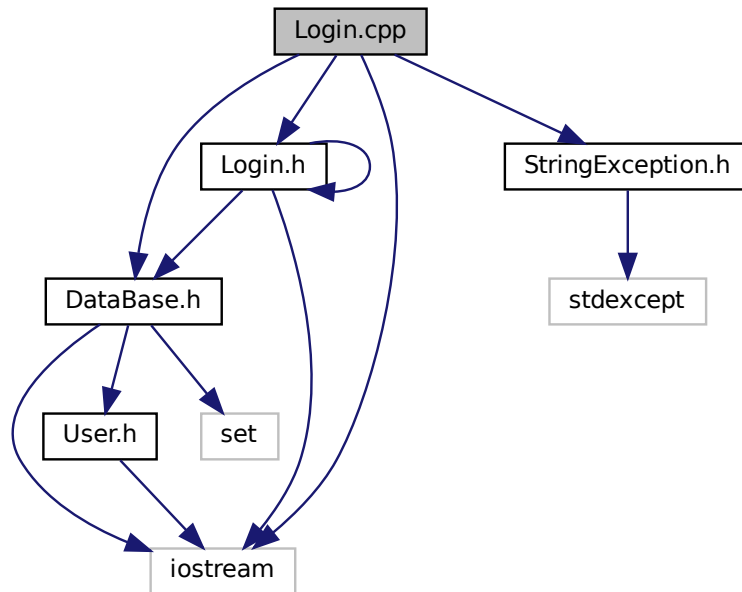
Date

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:

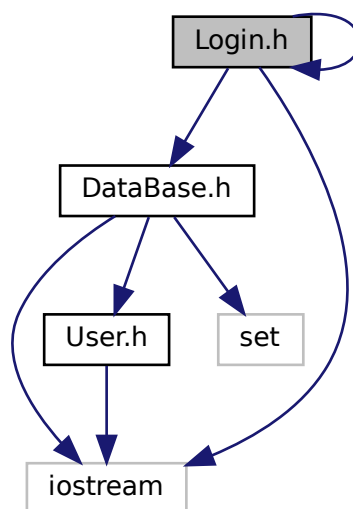


## 5.14 Login.h File Reference

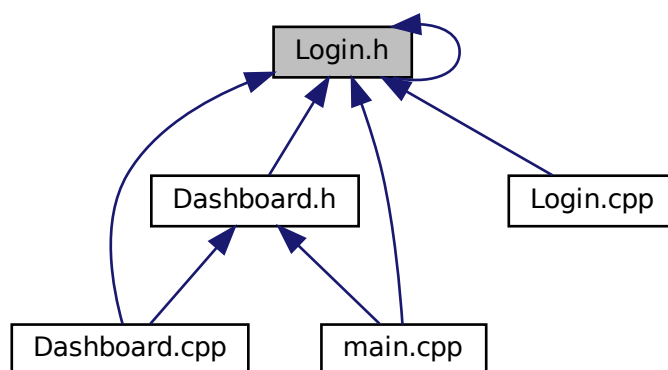
defines the login class with its attributes, methods, and constructor

```
#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

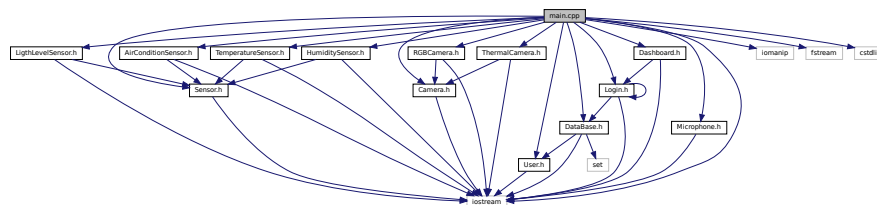
**5.15 main.cpp File Reference**

```

#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include "Sensor.h"
#include "TemperatureSensor.h"
#include "HumiditySensor.h"
#include "LightLevelSensor.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 ( )
```

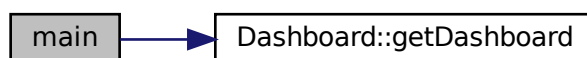
Definition at line 28 of file main.cpp.

```

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

References `Dashboard::getDashboard()`.

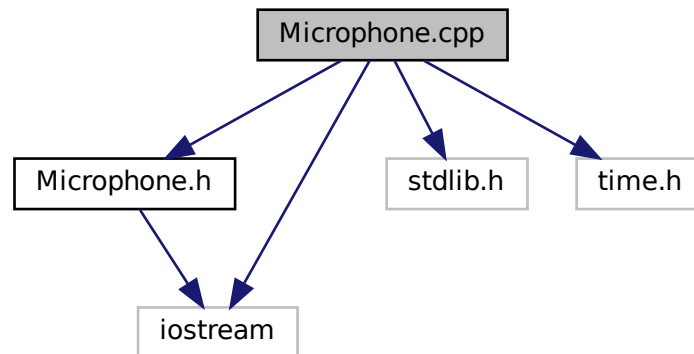
Here is the call graph for this function:



## 5.16 Microphone.cpp File Reference

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

Include dependency graph for Microphone.cpp:

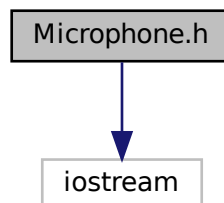


## 5.17 Microphone.h File Reference

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

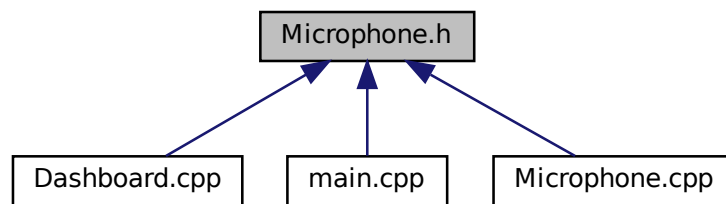
```
#include <iostream>
```

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

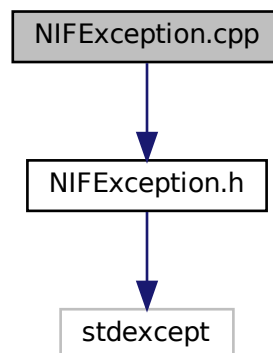
#### Date

2022-11-23

## 5.18 NIFException.cpp File Reference

```
#include "NIFException.h"
```

Include dependency graph for `NIFException.cpp`:

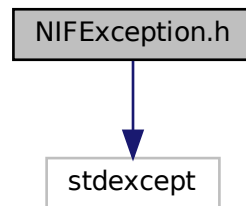


## 5.19 NIFException.h File Reference

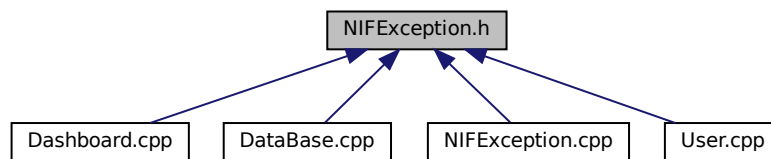
defines the NIF exception class with its constructor

```
#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

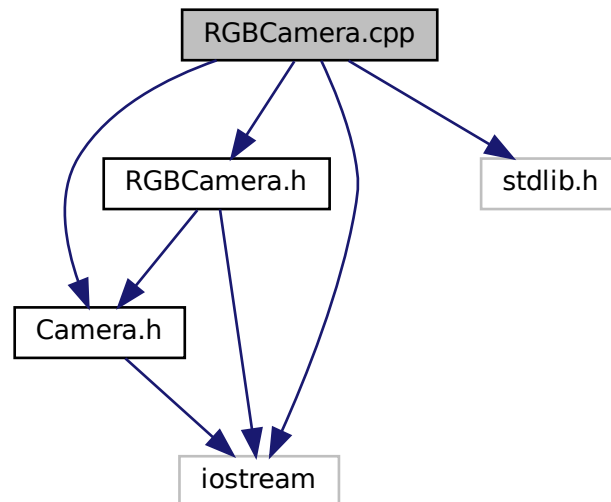
##### Date

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:

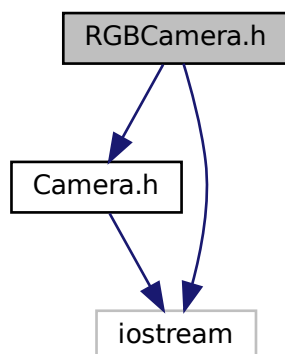


## 5.21 RGBCamera.h File Reference

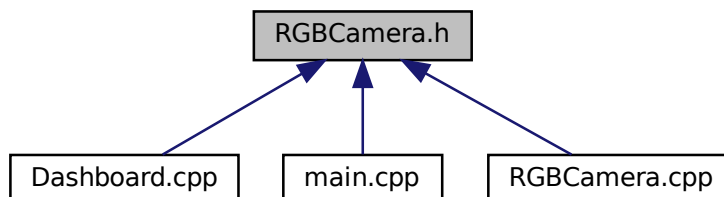
defines the [RGBCamera](#) class, which is inherited from the camera class, with its attributes, methods, and constructor

```
#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, which is inherited from the camera class, with its attributes, methods, and constructor

#### Author

Ana Martínez Albendea

#### Date

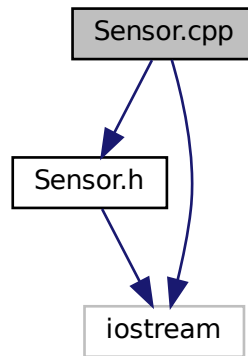
2022-11-23

## 5.22 Sensor.cpp File Reference

```
#include "Sensor.h"
```

```
#include <iostream>
```

Include dependency graph for Sensor.cpp:

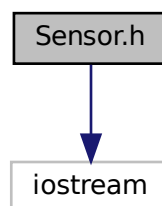


## 5.23 Sensor.h File Reference

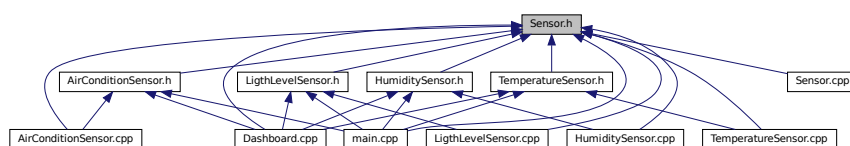
defines the sensor class with its attributes, methods, and constructor

```
#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

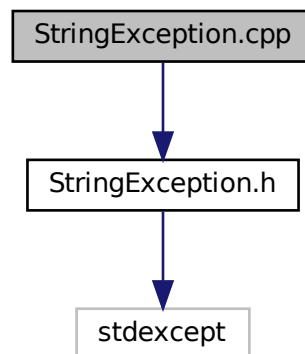
#### Date

2022-11-23

## 5.24 StringException.cpp File Reference

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

Include dependency graph for StringException.cpp:

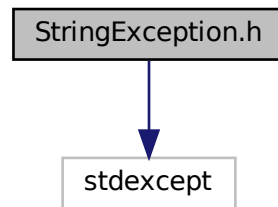


## 5.25 StringException.h File Reference

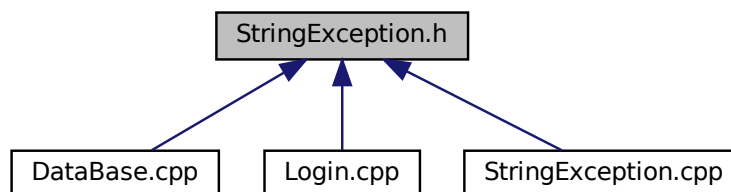
defines the string exception class with its constructor

```
#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

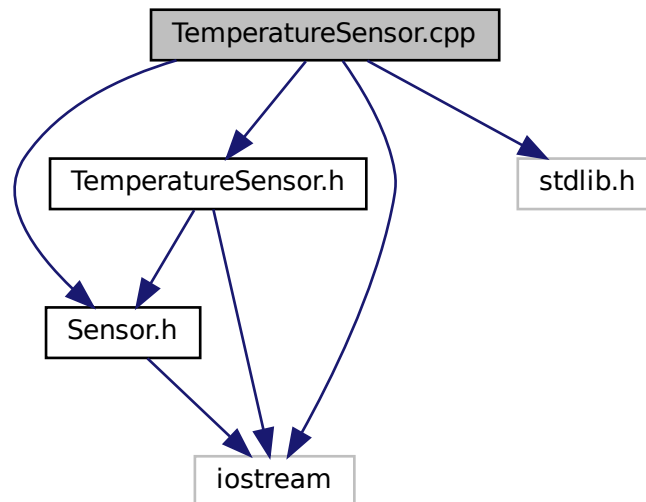
#### Date

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:



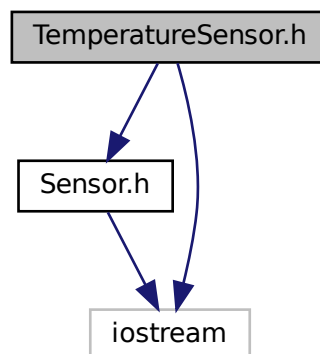
## 5.27 TemperatureSensor.h File Reference

defines the [TemperatureSensor](#) class, which is inherited from the sensor class, with its attributes, methods, and constructor

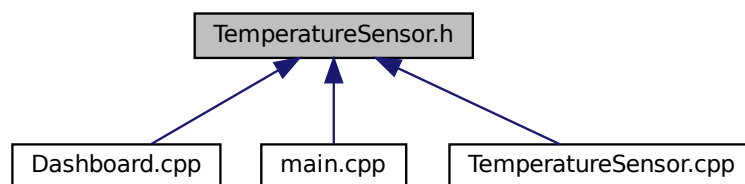
```
#include "Sensor.h"  
#include <iostream>
```



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, which is inherited from the sensor class, with its attributes, methods, and constructor

Author

Ana Martínez Albendea

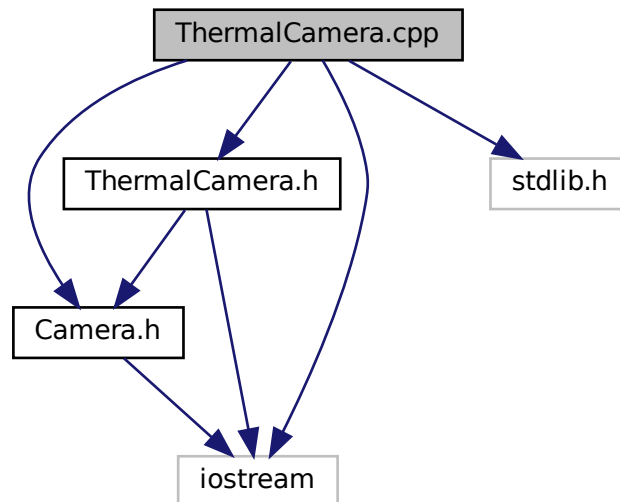
Date

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:

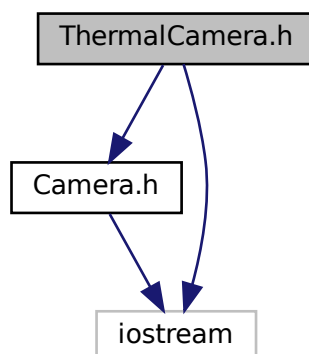


## 5.29 ThermalCamera.h File Reference

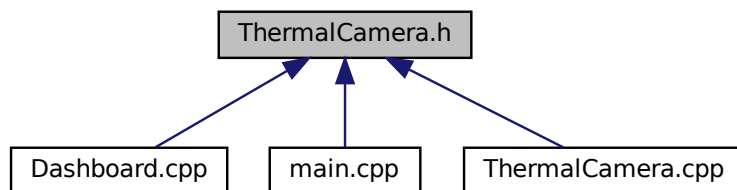
defines the [ThermalCamera](#) class, which is inherited from the camera class, with its attributes, methods, and constructor

```
#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, which is inherited from the camera class, with its attributes, methods, and constructor

#### Author

Ana Martínez Albendea

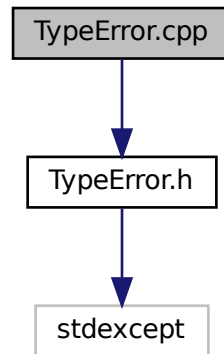
#### Date

2022-11-23

### 5.30 TypeError.cpp File Reference

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

Include dependency graph for TypeError.cpp:

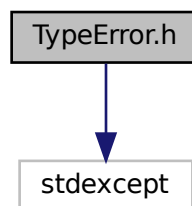


### 5.31 TypeError.h File Reference

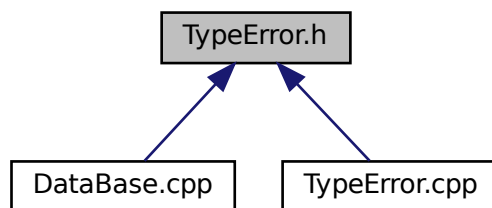
defines the user type exception class with its constructor

```
#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

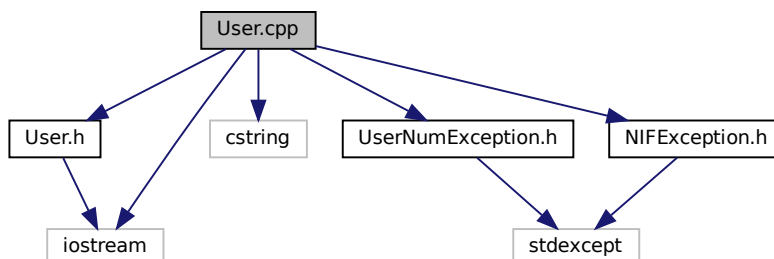
#### Date

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:

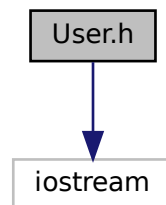


## 5.33 User.h File Reference

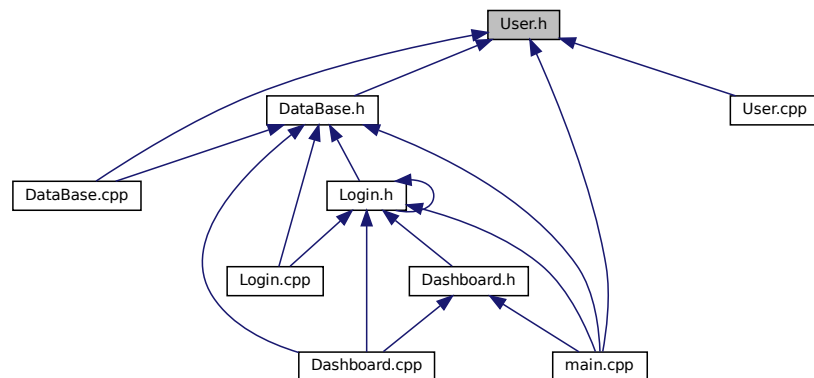
defines the user class with its attributes, methods, and constructor

```
#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

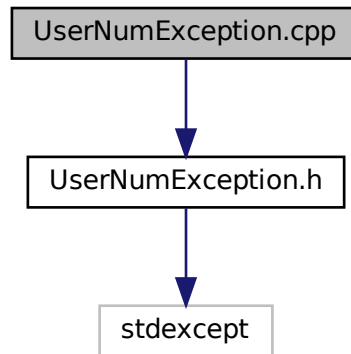
Date

2022-11-23

## 5.34 UserNumException.cpp File Reference

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

Include dependency graph for UserNumException.cpp:

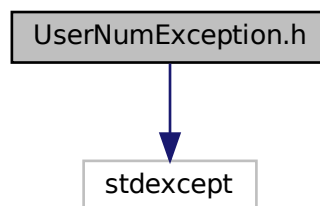


## 5.35 UserNumException.h File Reference

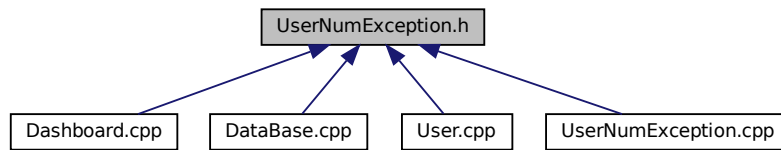
defines the user number exception class with its constructor

```
#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

## A

- Dashboard, [19](#)
- addUser
  - DataBase, [24](#)
- airCondition
  - AirConditionSensor, [9](#)
- AirConditionSensor, [7](#)
  - airCondition, [9](#)
  - AirConditionSensor, [8](#)
  - getInfo, [9](#)
- AirConditionSensor.cpp, [65](#)
- AirConditionSensor.h, [65](#)

## Camera, [10](#)

- Camera, [11](#)
- getInfo, [11](#)
- state, [13](#)
- turnOff, [11](#)
- turnOn, [12](#)

- Camera.cpp, [67](#)

- Camera.h, [67](#)

- checkAdmin
  - User, [58](#)

- checkLogin
  - Login, [36](#)

- checkUser
  - DataBase, [24](#)

- choice
  - Dashboard, [19](#)
  - DataBase, [28](#)
  - Microphone, [41](#)

## D

- Dashboard, [20](#)
- Dashboard, [13](#)
  - A, [19](#)
  - choice, [19](#)
  - D, [20](#)
  - Dashboard, [14](#)
  - getDashboard, [15](#)
  - getOption, [15](#)
  - H, [20](#)
  - L, [20](#)
  - Li, [20](#)
  - M, [20](#)
  - operator=, [16](#)
  - option, [20](#)
  - RC, [21](#)
  - showCameraChoices, [16](#)
  - showMenu, [17](#)

- showSensorChoices, [18](#)
- singleDashboard, [21](#)
- T, [21](#)
- TC, [21](#)

- Dashboard.cpp, [68](#)

- Dashboard.h, [69](#)

- DataBase, [22](#)

- addUser, [24](#)

- checkUser, [24](#)

- choice, [28](#)

- DataBase, [23](#)

- dataBaseUser, [28](#)

- deleteUser, [25](#)

- isAdmin, [28](#)

- it, [28](#)

- name, [29](#)

- NIF, [29](#)

- NIFStr, [29](#)

- saveFile, [26](#)

- showAdminChoices, [26](#)

- showUsers, [27](#)

- type, [29](#)

- userDeleted, [29](#)

- userNow, [29](#)

- userNumber, [30](#)

- userNumberStr, [30](#)

- DataBase.cpp, [70](#)

- DataBase.h, [70](#)

- dataBaseUser

- DataBase, [28](#)

- deleteUser

- DataBase, [25](#)

- getDashboard

- Dashboard, [15](#)

- getInfo

- AirConditionSensor, [9](#)

- Camera, [11](#)

- HumiditySensor, [32](#)

- LigthLevelSensor, [34](#)

- RGBCamera, [45](#)

- Sensor, [47](#)

- TemperatureSensor, [52](#)

- ThermalCamera, [54](#)

- getNIF

- User, [58](#)

- getOption

- Dashboard, [15](#)

- getSound

- Microphone, [40](#)

- getUserName
  - User, 59
- getUserNum
  - User, 59
- H
  - Dashboard, 20
- humidity
  - HumiditySensor, 32
- HumiditySensor, 30
  - getInfo, 32
  - humidity, 32
  - HumiditySensor, 31
- HumiditySensor.cpp, 72
- HumiditySensor.h, 72
- isAdmin
  - DataBase, 28
  - User, 61
- it
  - DataBase, 28
- L
  - Dashboard, 20
- Li
  - Dashboard, 20
- lighLevel
  - LighLevelSensor, 34
- LighLevelSensor, 33
  - getInfo, 34
  - lighLevel, 34
  - LighLevelSensor, 34
- LighLevelSensor.cpp, 74
- LighLevelSensor.h, 74
- Login, 35
  - checkLogin, 36
  - Login, 36
  - NIF, 37
  - NIFStr, 37
  - userNow, 38
  - userNowPtr, 38
  - userNumber, 38
  - userNumberStr, 38
- Login.cpp, 76
- Login.h, 76
- M
  - Dashboard, 20
- main
  - main.cpp, 78
- main.cpp, 78
  - main, 78
- Microphone, 39
  - choice, 41
  - getSound, 40
  - Microphone, 39
  - showChoice, 40
  - sound, 42
  - state, 42
  - turnOff, 41
  - turnOn, 41
- Microphone.cpp, 80
- Microphone.h, 80
- name
  - DataBase, 29
  - User, 61
- NIF
  - DataBase, 29
  - Login, 37
  - User, 61
- NIFException, 42
  - NIFException, 43
- NIFException.cpp, 81
- NIFException.h, 82
- NIFStr
  - DataBase, 29
  - Login, 37
- operator<
  - User, 60
- operator=
  - Dashboard, 16
- option
  - Dashboard, 20
- RC
  - Dashboard, 21
- RGBCamera, 44
  - getInfo, 45
  - RGBCamera, 45
- RGBCamera.cpp, 83
- RGBCamera.h, 83
- saveFile
  - DataBase, 26
- Sensor, 46
  - getInfo, 47
  - Sensor, 47
  - state, 49
  - turnOff, 47
  - turnOn, 48
- Sensor.cpp, 85
- Sensor.h, 85
- showAdminChoices
  - DataBase, 26
- showCameraChoices
  - Dashboard, 16
- showChoice
  - Microphone, 40
- showMenu
  - Dashboard, 17
- showSensorChoices
  - Dashboard, 18
- showUser
  - User, 60
- showUsers
  - DataBase, 27

- singleDashboard
  - Dashboard, [21](#)
- sound
  - Microphone, [42](#)
- state
  - Camera, [13](#)
  - Microphone, [42](#)
  - Sensor, [49](#)
- StringException, [49](#)
  - StringException, [50](#)
- StringException.cpp, [86](#)
- StringException.h, [86](#)
- T
  - Dashboard, [21](#)
- TC
  - Dashboard, [21](#)
- temperature
  - TemperatureSensor, [52](#)
- TemperatureSensor, [51](#)
  - getInfo, [52](#)
  - temperature, [52](#)
  - TemperatureSensor, [52](#)
- TemperatureSensor.cpp, [88](#)
- TemperatureSensor.h, [88](#)
- ThermalCamera, [53](#)
  - getInfo, [54](#)
  - ThermalCamera, [54](#)
- ThermalCamera.cpp, [90](#)
- ThermalCamera.h, [90](#)
- turnOff
  - Camera, [11](#)
  - Microphone, [41](#)
  - Sensor, [47](#)
- turnOn
  - Camera, [12](#)
  - Microphone, [41](#)
  - Sensor, [48](#)
- type
  - DataBase, [29](#)
- TypeError, [55](#)
  - TypeError, [56](#)
- TypeError.cpp, [92](#)
- TypeError.h, [92](#)
- User, [57](#)
  - checkAdmin, [58](#)
  - getNIF, [58](#)
  - getUserName, [59](#)
  - getUserNum, [59](#)
  - isAdmin, [61](#)
  - name, [61](#)
  - NIF, [61](#)
  - operator<, [60](#)
  - showUser, [60](#)
  - User, [58](#)
  - userNumber, [61](#)
- User.cpp, [93](#)
- User.h, [94](#)
- userDeleted
  - DataBase, [29](#)
- userNow
  - DataBase, [29](#)
  - Login, [38](#)
- userNowPtr
  - Login, [38](#)
- userNumber
  - DataBase, [30](#)
  - Login, [38](#)
  - User, [61](#)
- userNumberStr
  - DataBase, [30](#)
  - Login, [38](#)
- UserNumException, [62](#)
  - UserNumException, [63](#)
- UserNumException.cpp, [95](#)
- UserNumException.h, [95](#)