

Julio Veganos e Hijos interfaz

Generated by Doxygen 1.9.1

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 actualUserException Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 actualUserException()	8
4.2 Admin Class Reference	9
4.2.1 Detailed Description	10
4.2.2 Constructor & Destructor Documentation	10
4.2.2.1 Admin()	11
4.2.3 Member Data Documentation	11
4.2.3.1 ADMIN	11
4.3 AirQuality Class Reference	11
4.3.1 Detailed Description	12
4.3.2 Constructor & Destructor Documentation	12
4.3.2.1 AirQuality()	12
4.3.3 Member Function Documentation	13
4.3.3.1 getData()	13
4.4 Brightness Class Reference	13
4.4.1 Detailed Description	15
4.4.2 Constructor & Destructor Documentation	15
4.4.2.1 Brightness()	15
4.4.3 Member Function Documentation	15
4.4.3.1 getData()	15
4.5 credentialException Class Reference	16
4.5.1 Detailed Description	17
4.5.2 Constructor & Destructor Documentation	17
4.5.2.1 credentialException()	17
4.6 Dashboard Class Reference	18
4.6.1 Detailed Description	19
4.6.2 Constructor & Destructor Documentation	19
4.6.2.1 Dashboard()	19
4.6.3 Member Function Documentation	19
4.6.3.1 addUserDashboard()	20
4.6.3.2 cabecera()	21

4.6.3.3 dashboard()	21
4.6.3.4 eraseUserDashboard()	22
4.6.3.5 goBack()	23
4.6.3.6 goOut()	23
4.6.3.7 goToSecurity()	24
4.6.3.8 goToSensors()	25
4.6.3.9 set_dashboardAdmin()	26
4.6.3.10 set_dashboardEmployer()	27
4.6.4 Member Data Documentation	27
4.6.4.1 database	27
4.6.4.2 opcion	28
4.6.4.3 pantalla	28
4.7 DataBase Class Reference	28
4.7.1 Detailed Description	29
4.7.2 Constructor & Destructor Documentation	29
4.7.2.1 DataBase()	30
4.7.3 Member Function Documentation	30
4.7.3.1 addUser()	30
4.7.3.2 eraseUser()	31
4.7.3.3 readUsers()	31
4.7.3.4 userList()	32
4.7.3.5 validUser()	33
4.7.3.6 writeUsers()	33
4.7.4 Member Data Documentation	34
4.7.4.1 actualID	34
4.7.4.2 admin	35
4.7.4.3 isAdmin	35
4.7.4.4 nuevoID	35
4.7.4.5 nuevoNIF	35
4.7.4.6 valided	35
4.7.4.7 vectorUser	35
4.8 Employer Class Reference	36
4.8.1 Detailed Description	37
4.8.2 Constructor & Destructor Documentation	37
4.8.2.1 Employer()	38
4.8.3 Member Data Documentation	38
4.8.3.1 ADMIN	38
4.9 Humidity Class Reference	38
4.9.1 Detailed Description	39
4.9.2 Constructor & Destructor Documentation	39
4.9.2.1 Humidity()	39
4.9.3 Member Function Documentation	40

4.9.3.1 getData()	40
4.10 instructionException Class Reference	41
4.10.1 Detailed Description	42
4.10.2 Constructor & Destructor Documentation	42
4.10.2.1 instructionException()	42
4.11 Login Class Reference	42
4.11.1 Detailed Description	44
4.11.2 Constructor & Destructor Documentation	44
4.11.2.1 Login()	44
4.11.3 Member Function Documentation	44
4.11.3.1 comprobar_user_data()	45
4.11.3.2 get_ID()	45
4.11.3.3 get_NIF()	46
4.11.3.4 go_back()	46
4.11.3.5 go_to_dashboard()	46
4.11.3.6 login()	47
4.11.3.7 set_login()	47
4.11.4 Member Data Documentation	47
4.11.4.1 contador	48
4.11.4.2 data_valid	48
4.11.4.3 database	48
4.11.4.4 intentos	48
4.11.4.5 temp_ID	48
4.11.4.6 temp_NIF	48
4.12 Security Class Reference	49
4.12.1 Detailed Description	50
4.12.2 Constructor & Destructor Documentation	50
4.12.2.1 Security()	50
4.12.3 Member Function Documentation	50
4.12.3.1 alarm()	50
4.12.3.2 camera()	51
4.12.3.3 door()	52
4.12.3.4 generate_random()	52
4.12.3.5 goBackSecurity()	53
4.12.3.6 open_close()	53
4.12.3.7 security()	53
4.12.3.8 window()	54
4.12.4 Member Data Documentation	54
4.12.4.1 alarma_status	55
4.12.4.2 open	55
4.12.4.3 seguridad	55
4.13 Sensor Class Reference	55

4.13.1 Detailed Description	56
4.13.2 Constructor & Destructor Documentation	56
4.13.2.1 Sensor()	56
4.13.3 Member Function Documentation	57
4.13.3.1 cabeceraSensor()	57
4.13.3.2 getData()	57
4.13.3.3 goBackSensor()	58
4.13.3.4 randomData()	58
4.13.4 Member Data Documentation	58
4.13.4.1 back_1	59
4.14 stringException Class Reference	59
4.14.1 Detailed Description	60
4.14.2 Constructor & Destructor Documentation	60
4.14.2.1 stringException()	60
4.15 Temperature Class Reference	61
4.15.1 Detailed Description	62
4.15.2 Constructor & Destructor Documentation	62
4.15.2.1 Temperature()	62
4.15.3 Member Function Documentation	62
4.15.3.1 getData()	62
4.16 User Class Reference	63
4.16.1 Detailed Description	66
4.16.2 Constructor & Destructor Documentation	66
4.16.2.1 User()	66
4.16.3 Member Function Documentation	66
4.16.3.1 getNumRecord()	66
4.16.3.2 getUserADMIN()	67
4.16.3.3 getUserID()	67
4.16.3.4 getUserNIF()	67
4.16.3.5 operator<()	68
4.16.3.6 operator==()	68
4.16.3.7 setNumRecord()	68
4.16.3.8 setUserID()	69
4.16.3.9 setUserNIF()	69
4.16.3.10 setUserRole()	69
4.16.4 Member Data Documentation	69
4.16.4.1 ADMIN	69
4.16.4.2 ID	70
4.16.4.3 NIF	70
4.16.4.4 numRecord	70

5 File Documentation

71

5.1 include/Admin.h File Reference	71
5.1.1 Detailed Description	71
5.2 include/AirQuality.h File Reference	72
5.2.1 Detailed Description	73
5.3 include/Brightness.h File Reference	73
5.3.1 Detailed Description	74
5.4 include/Dashboard.h File Reference	75
5.4.1 Detailed Description	75
5.5 include/DataBase.h File Reference	76
5.5.1 Detailed Description	77
5.6 include/Employer.h File Reference	77
5.6.1 Detailed Description	78
5.7 include/Exceptions.h File Reference	78
5.7.1 Detailed Description	79
5.8 include/Humidity.h File Reference	79
5.8.1 Detailed Description	80
5.9 include/Login.h File Reference	81
5.9.1 Detailed Description	81
5.10 include/main.h File Reference	82
5.10.1 Detailed Description	83
5.10.2 Variable Documentation	83
5.10.2.1 d	83
5.10.2.2 l	83
5.11 include/Security.h File Reference	84
5.11.1 Detailed Description	84
5.12 include/Sensor.h File Reference	85
5.12.1 Detailed Description	86
5.13 include/Temperature.h File Reference	86
5.13.1 Detailed Description	87
5.14 include/User.h File Reference	87
5.14.1 Detailed Description	88
5.15 src/AirQuality.cpp File Reference	88
5.16 src/Brightness.cpp File Reference	89
5.17 src/Dashboard.cpp File Reference	89
5.17.1 Variable Documentation	90
5.17.1.1 airq	90
5.17.1.2 brigh	90
5.17.1.3 database	90
5.17.1.4 hum	90
5.17.1.5 opcion	91
5.17.1.6 s	91
5.17.1.7 sen	91

5.17.1.8 temp	91
5.18 src/DataBase.cpp File Reference	91
5.18.1 Variable Documentation	92
5.18.1.1 admin	92
5.18.1.2 nuevoID	92
5.18.1.3 nuevoNIF	92
5.18.1.4 valided	92
5.19 src/Humidity.cpp File Reference	93
5.20 src/Login.cpp File Reference	93
5.20.1 Variable Documentation	94
5.20.1.1 contador	94
5.20.1.2 data_valid	94
5.20.1.3 intentos	94
5.20.1.4 temp_ID	95
5.20.1.5 temp_NIF	95
5.21 src/main.cpp File Reference	95
5.21.1 Function Documentation	95
5.21.1.1 main()	96
5.22 src/Security.cpp File Reference	96
5.22.1 Variable Documentation	97
5.22.1.1 alarm_status	97
5.22.1.2 back	97
5.22.1.3 db	98
5.22.1.4 entrada	98
5.22.1.5 open	98
5.22.1.6 seguridad	98
5.23 src/Sensor.cpp File Reference	98
5.23.1 Variable Documentation	99
5.23.1.1 back_1	99
5.24 src/Temperature.cpp File Reference	99
5.25 src/User.cpp File Reference	100
5.25.1 Variable Documentation	100
5.25.1.1 ADMIN	100
5.25.1.2 ID	100
5.25.1.3 NIF	100

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Dashboard	18
DataBase	28
Login	42
std::runtime_error	
actualUserException	7
credentialException	16
instructionException	41
stringException	59
Security	49
Sensor	55
AirQuality	11
Brightness	13
Humidity	38
Temperature	61
User	63
Admin	9
Employer	36

Chapter 2

Class Index

2.1 Class List

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

actualUserException	7
Admin	9
AirQuality	11
Brightness	13
credentialException	
If the ID or NIF are incorrect, the exception will be thrown	16
Dashboard	18
DataBase	28
Employer	36
Humidity	38
instructionException	
If the instruction of the menu is wrong, the exception appears	41
Login	42
Security	49
Sensor	55
stringException	
If a string is introduced when an int is needed an exception will be thrown	59
Temperature	61
User	63

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/Admin.h	71
include/AirQuality.h	
Air Quality sensor	72
include/Brightness.h	
Luminosity sensor	73
include/Dashboard.h	
Main menu	75
include/DataBase.h	
DataBase implementation and validate users's methods	76
include/Employer.h	
Object employer	77
include/Exceptions.h	
Exception that may occur in the interface	78
include/Humidity.h	
Humidity sensor	79
include/Login.h	
Login interface	81
include/main.h	
Main program	82
include/Security.h	
Security methods	84
include/Sensor.h	85
include/Temperature.h	
Temperature sensor	86
include/User.h	
User object's file	87
src/AirQuality.cpp	88
src/Brightness.cpp	89
src/Dashboard.cpp	89
src/DataBase.cpp	91
src/Humidity.cpp	93
src/Login.cpp	93
src/main.cpp	95
src/Security.cpp	96
src/Sensor.cpp	98
src/Temperature.cpp	99
src/User.cpp	100

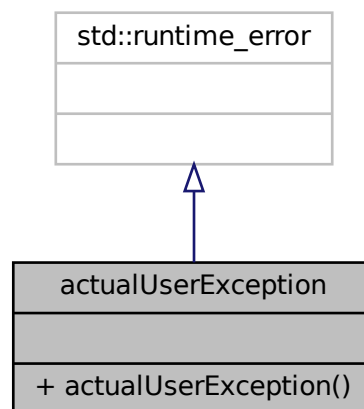
Chapter 4

Class Documentation

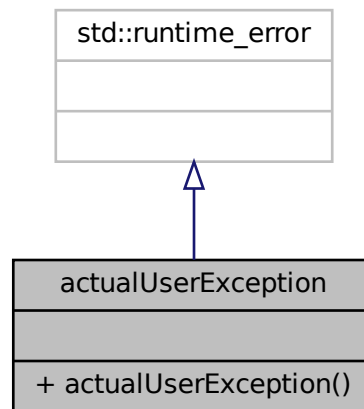
4.1 actualUserException Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for actualUserException:



Collaboration diagram for `actualUserException`:



Public Member Functions

- [actualUserException\(\)](#)

4.1.1 Detailed Description

Definition at line 47 of file `Exceptions.h`.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 actualUserException()

```
actualUserException::actualUserException ( ) [inline]
```

Definition at line 49 of file `Exceptions.h`.

```
50 : std::runtime_error ("No puedes borrar el user usado actualmente") {}
```

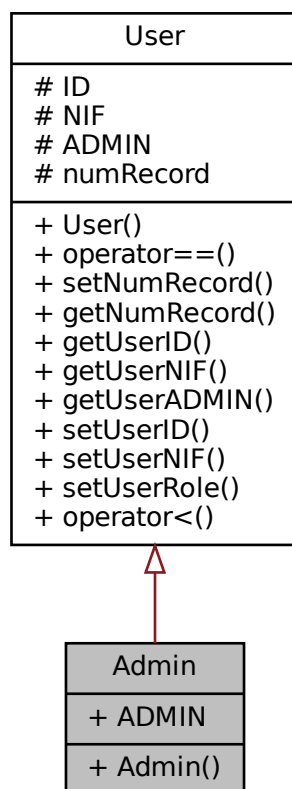
The documentation for this class was generated from the following file:

- [include/Exceptions.h](#)

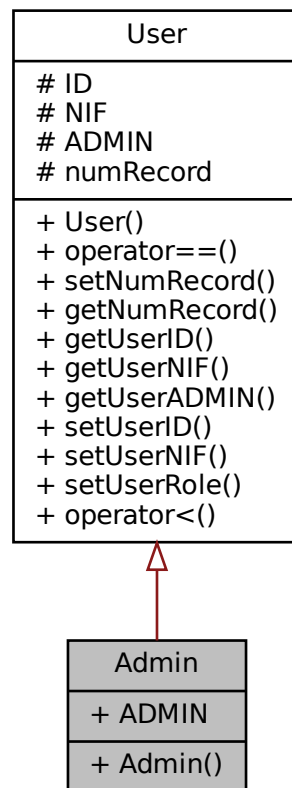
4.2 Admin Class Reference

```
#include <Admin.h>
```

Inheritance diagram for Admin:



Collaboration diagram for Admin:



Public Member Functions

- [Admin](#) ()

Public Attributes

- int [ADMIN](#) = 1

Additional Inherited Members

4.2.1 Detailed Description

Definition at line 20 of file Admin.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Admin()

```
Admin::Admin ( )
```

4.2.3 Member Data Documentation

4.2.3.1 ADMIN

```
int Admin::ADMIN = 1
```

Definition at line 23 of file Admin.h.

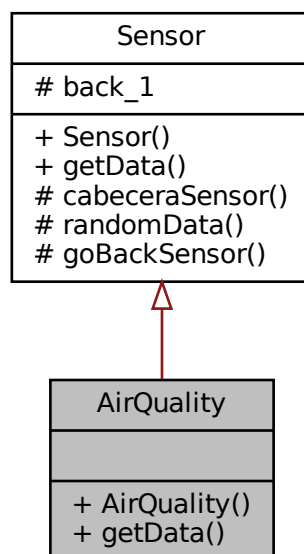
The documentation for this class was generated from the following file:

- [include/Admin.h](#)

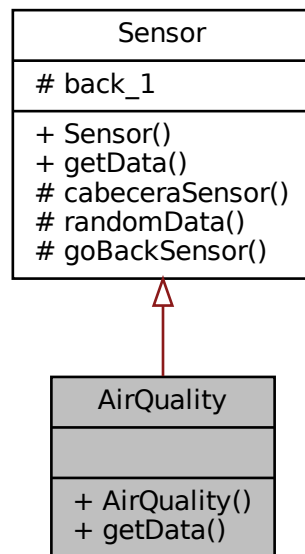
4.3 AirQuality Class Reference

```
#include <AirQuality.h>
```

Inheritance diagram for AirQuality:



Collaboration diagram for AirQuality:



Public Member Functions

- [AirQuality \(\)](#)
- void [getData \(\)](#)

Additional Inherited Members

4.3.1 Detailed Description

Definition at line 20 of file `AirQuality.h`.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 AirQuality()

```
AirQuality::AirQuality ( )
```

Definition at line 8 of file `AirQuality.cpp`.

```
8 {}
```

4.3.3 Member Function Documentation

4.3.3.1 getData()

```
void AirQuality::getData ( )
```

Definition at line 11 of file AirQuality.cpp.

```
11         {
12     while (true) {
13         int random = this->randomData();
14         cout << "\n\t\tAir quality:\t\t" << (random + 30) << " ppm\n" << endl;
15         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
16         cin >> this->back_1;
17         switch (this->back_1) {
18
19             case 0:
20                 return;
21                 break;
22
23             default:
24
25                 cout << "Orden incorrecta" << endl;
26                 system("sleep 1");
27                 this->cabeceraSensor();
28                 break;
29         }
30     }
31 }
```

References back_1.

Here is the caller graph for this function:



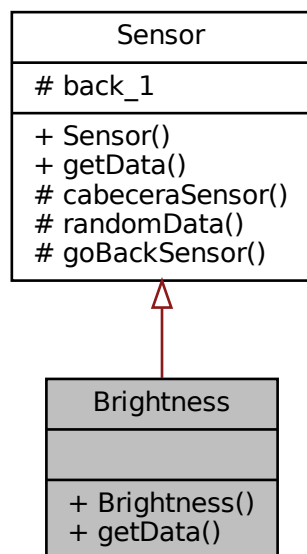
The documentation for this class was generated from the following files:

- include/AirQuality.h
- src/AirQuality.cpp

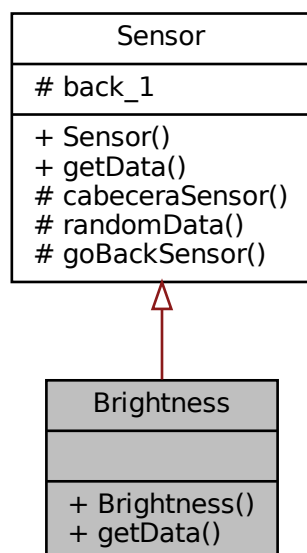
4.4 Brightness Class Reference

```
#include <Brightness.h>
```

Inheritance diagram for Brightness:



Collaboration diagram for Brightness:



Public Member Functions

- [Brightness](#) ()
- void [getData](#) ()

Additional Inherited Members

4.4.1 Detailed Description

Definition at line 21 of file Brightness.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Brightness()

```
Brightness::Brightness ( )
```

Definition at line 8 of file Brightness.cpp.

```
8 {}
```

4.4.3 Member Function Documentation

4.4.3.1 getData()

```
void Brightness::getData ( )
```

Definition at line 11 of file Brightness.cpp.

```
11         {
12     while (true) {
13         int random = this->randomData();
14         cout << "\n\t\tBrightness:\t\t" << (random + 10) << " lmen/m2\n" << endl;
15         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
16         cin >> this->back_1;
17         switch (this->back_1) {
18
19             case 0:
20                 return;
21                 break;
22
23             default:
24
25                 cout << "Orden incorrecta" << endl;
26                 system("sleep 1");
27                 this->cabeceraSensor();
28                 break;
29         }
30     }
31 }
```

References [back_1](#).

Here is the caller graph for this function:



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

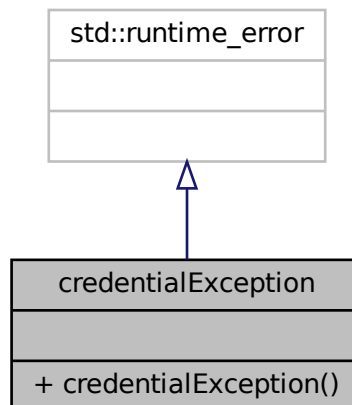
- [include/Brightness.h](#)
- [src/Brightness.cpp](#)

4.5 credentialException Class Reference

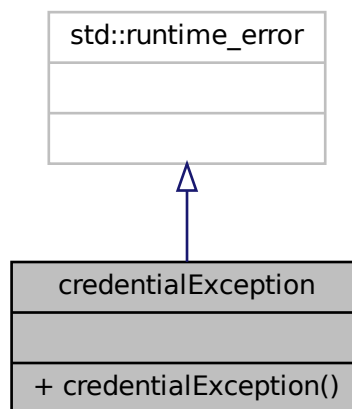
If the ID or NIF are incorrect, the exception will be thrown.

```
#include <Exceptions.h>
```

Inheritance diagram for `credentialException`:



Collaboration diagram for credentialException:



Public Member Functions

- [credentialException\(\)](#)

4.5.1 Detailed Description

If the ID or NIF are incorrect, the exception will be thrown.

Definition at line 31 of file Exceptions.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 credentialException()

```
credentialException::credentialException ( ) [inline]
```

Definition at line 33 of file Exceptions.h.

```
34 : std::runtime_error ("El usuario y/o password son incorrectos") {}
```

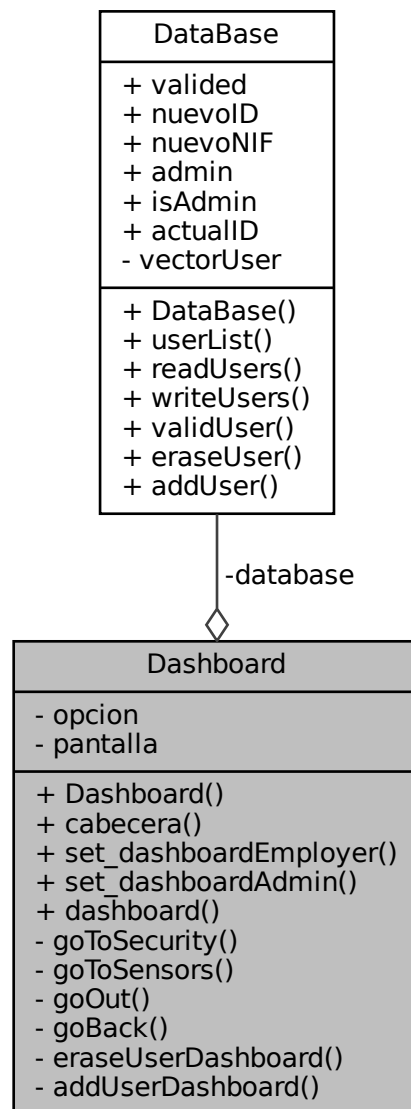
The documentation for this class was generated from the following file:

- include/Exceptions.h

4.6 Dashboard Class Reference

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

Collaboration diagram for Dashboard:



Public Member Functions

- [Dashboard](#) ()
- void [cabecera](#) ()

Principal header with the name of the business.

- void [set_dashboardEmployer](#) ()

Set the menu for the employers.

- void `set_dashboardAdmin` ()

Set the menu for the Admins.

- void `dashboard` (`DataBase` *, bool)

Private Member Functions

- void `goToSecurity` ()

Menu with options for the security.

- void `goToSensors` ()

menu that shows the different sensors. For each instruction a sensor is shown

- void `goOut` ()

Function to exit the program when the option is selected.

- void `goBack` ()

- void `eraseUserDashboard` ()

Method that allow the admin to erase the users.

- void `addUserDashboard` ()

Function that allows the admin to add new users.

Private Attributes

- int `opcion`
- int `pantalla`
- `DataBase` * `database`

4.6.1 Detailed Description

Definition at line 19 of file Dashboard.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Dashboard()

```
Dashboard::Dashboard ( )
```

Definition at line 27 of file Dashboard.cpp.

```
27 {}
```

4.6.3 Member Function Documentation

4.6.3.1 addUserDashboard()

```
void Dashboard::addUserDashboard ( ) [private]
```

Function that allows the admin to add new users.

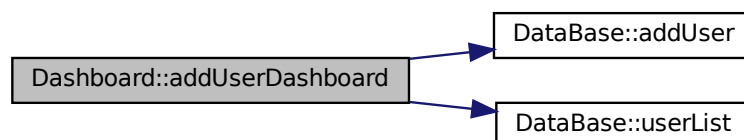
Definition at line 263 of file Dashboard.cpp.

```

263     {
264         while (true) {
265             cabecera();
266             database->userList();
267             cout <<
                "\n-----\n"
            << endl;
268             cout << "Si desea añadir un usuario introduzca 1, para salir introduzca 0" << endl;
269             cout << "»> ";
270             cin >> opcion;
271             try {
272                 switch (opcion)
273                 {
274                     case 1:
275                         cout << "Introduzca ID del nuevo usuario" << endl;
276                         cout << "»> ";
277                         cin >> database->nuevoID;
278                         cout << database->nuevoID << endl;
279                         cout << "Introduzca NIF del nuevo usuario" << endl;
280                         cout << "»> ";
281                         cin >> database->nuevoNIF;
282                         cout << database->nuevoNIF << endl;
283                         cout << "¿El usuario es administrador? (introduzca 0 para NO, 1 para SI" << endl;
284                         cout << "»> ";
285                         cin >> database->admin;
286                         database->addUser();
287                         break;
288                     case 0:
289                         return;
290                     default:
291                         throw instructionException();
292                 }
293             } catch (instructionException &except) {
294                 cout << "Exception: " << except.what() << endl;
295                 system("sleep 2");
296             }
297         }
298     }
299 }
```

References DataBase::addUser(), DataBase::admin, database, DataBase::nuevoID, DataBase::nuevoNIF, opcion, and DataBase::userList().

Here is the call graph for this function:



4.6.3.2 cabecera()

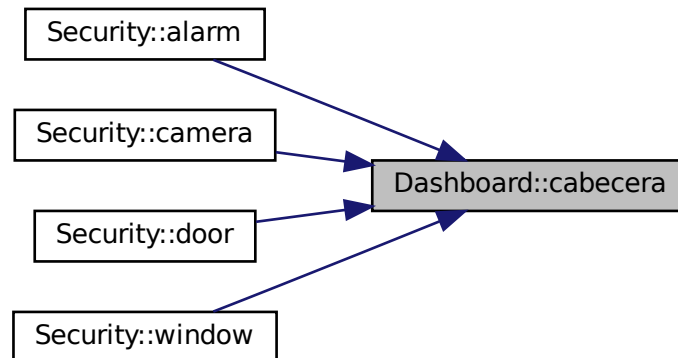
```
void Dashboard::cabecera ( )
```

Principal header with the name of the business.

Definition at line 41 of file Dashboard.cpp.

[illegible]

Here is the caller graph for this function:



4.6.3.3 dashboard()

```
void Dashboard::dashboard (
    DataBase * base,
    bool isAdmin )
```

Definition at line 301 of file Dashboard.cpp.

```

301
302     database = base;
303     if (isAdmin) {
304         set_dashboardAdmin();
305     }
306 }

```

```

305     } else {
306         set_dashboardEmployer();
307     }
308 }

```

References database.

Here is the caller graph for this function:



4.6.3.4 eraseUserDashboard()

```
void Dashboard::eraseUserDashboard ( ) [private]
```

Method that allow the admin to erase the users.

Definition at line 234 of file Dashboard.cpp.

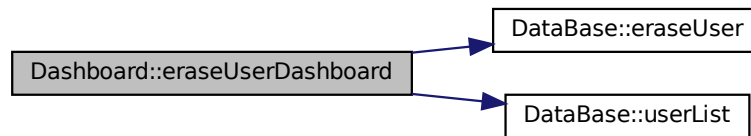
```

234     {
235         while (true) {
236             int opcion;
237             cabecera();
238             database->userList();
239             cout <<
                "\n-----\n"
            << endl;
240             cout << "\nIntroduzca el ID del usuario que desea borrar o introduzca 0 para volver" << endl;
241             cout << database->actualID << endl;
242             cout << ">>> ";
243             cin >> opcion;
244             try {
245                 switch (opcion) {
246                     case 0:
247                         return;
248                     default:
249                         if (database->actualID == opcion) {
250                             throw actualUserException();
251                             break;
252                         } else {
253                             database->eraseUser(opcion);
254                         }
255                 }
256             } catch (actualUserException &except) {
257                 cout << "Exception: " << except.what() << endl;
258                 system("sleep 2");
259             }
260         }
261     }

```

References DataBase::actualID, database, DataBase::eraseUser(), opcion, and DataBase::userList().

Here is the call graph for this function:



4.6.3.5 goBack()

```
void Dashboard::goBack ( ) [private]
```

4.6.3.6 goOut()

```
void Dashboard::goOut ( ) [private]
```

Function to exit the program when the option is selected.

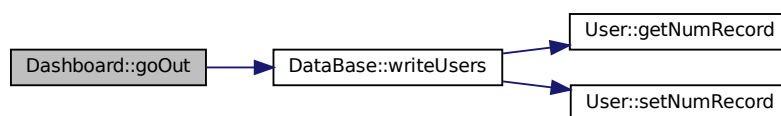
Definition at line 227 of file `Dashboard.cpp`.

```

227     {
228         system("clear");
229         cout << "\x1b[41mSaliendo del sistema...\x1b[0m" << endl;
230         database->writeUsers();
231         exit(0);
232     }
  
```

References database, and `DataBase::writeUsers()`.

Here is the call graph for this function:



4.6.3.7 goToSecurity()

```
void Dashboard::goToSecurity ( ) [private]
```

Menu with options for the security.

Definition at line 187 of file Dashboard.cpp.

```
187         {
188             while (true) {
189                 // Heading and options
190                 cabecera();
191                 cout << "\t\tOPCIONES DISPONIBLES\n\t\t-Puertas\t==> 1\n\t\t-Ventanas\t==> 2\n\t\t-Camaras\t==>
3\n\t\t-Alarma\t\t==> 4\n\t\t-Exit\t\t==> 0" << endl;
192                 cout <<
"\n-----\n"
<< endl;
193                 cout << "Seleccione una opcion escribiendo el número correspondiente.\n" << endl;
194                 cout << ">>> ";
195                 cin >> opcion;
196                 try {
197                     switch (opcion) {
198                         case 0:
199                             return;
200                         case 1:
201                             cabecera();
202                             s.security("puertas");
203                             break;
204                         case 2:
205                             cabecera();
206                             s.security("ventanas");
207                             break;
208                         case 3:
209                             cabecera();
210                             s.security("camaras");
211                             break;
212                         case 4:
213                             cabecera();
214                             s.security("alarma");
215                             break;
216                         default:
217                             throw instructionException();
218                     }
219                 } catch (instructionException &except) {
220                     cout << "Exception: " << except.what() << endl;
221                     system("sleep 2");
222                 }
223             }
224 };
```

References opcion, s, and Security::security().

Here is the call graph for this function:



4.6.3.8 goToSensors()

```
void Dashboard::goToSensors ( ) [private]
```

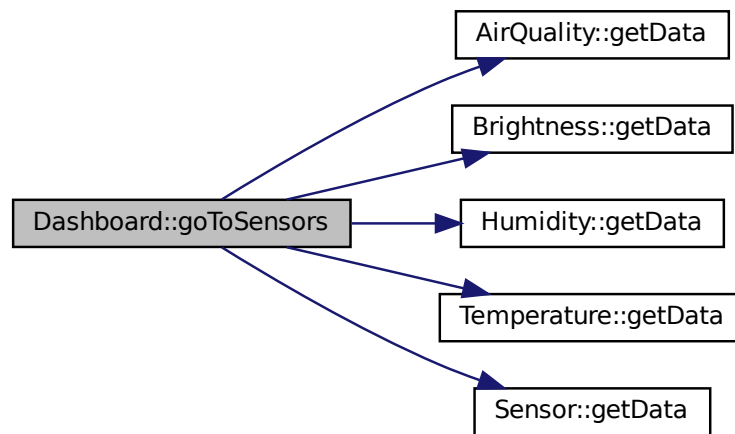
menu that shows the different sensors. For each instruction a sensor is shown

Definition at line 133 of file Dashboard.cpp.

```
133     {
134         while (true) {
135             // Heading and options
136             cabecera();
137             cout << "\t\tOPCIONES DISPONIBLES\n\t\t-Humidity\t==> 1 \n\t\t-Brightness\t==> 2 \n\t\t-Air
Quality\t==> 3 \n\t\t-Temperature\t==> 4 \n\t\t-RBG cam\t==> 5 \n\t\t-Termal cam\t==> 6
\n\t\t-Exit\t==> 0" << endl;
138             cout <<
"\n-----\n"
<< endl;
139             cout << "Seleccione una opcion escribiendo el número correspondiente.\n" << endl;
140             cout << ">> ";
141             cin >> opcion;
142             try {
143                 switch (opcion) {
144                     case 0:
145                         return;
146
147                     case 1:
148                         cabecera();
149                         hum.getData();
150                         break;
151
152                     case 2:
153                         cabecera();
154                         brigh.getData();
155                         break;
156
157                     case 3:
158                         cabecera();
159                         airq.getData();
160                         break;
161
162                     case 4:
163                         cabecera();
164                         temp.getData();
165                         break;
166
167                     case 5:
168                         cabecera();
169                         sen.getData(opcion);
170                         break;
171
172                     case 6:
173                         cabecera();
174                         sen.getData(opcion);
175                         break;
176                     default:
177                         throw instructionException();
178                 }
179             } catch (instructionException &except) {
180                 cout << "Exception: " << except.what() << endl;
181                 system("sleep 2");
182             }
183         }
184     };
```

References `airq`, `brigh`, `AirQuality::getData()`, `Brightness::getData()`, `Humidity::getData()`, `Temperature::getData()`, `Sensor::getData()`, `hum`, `opcion`, `sen`, and `temp`.

Here is the call graph for this function:



4.6.3.9 set_dashboardAdmin()

```
void Dashboard::set_dashboardAdmin ( )
```

Set the menu for the Admins.

Definition at line 93 of file Dashboard.cpp.

```

93         {
94
95         while (true) {
96             cabecera();
97             cout << "\t\tOPCIONES DISPONIBLES\n\t\t-Sensores\t==> 1\n\t\t-Seguridad\t==> 2\n\t\t-Add
Users\t==> 3\n\t\t-Erase Users\t==> 4\n\t\t-Cerrar Sesión\t==> 5\n\t\t-Exit\t\t==> 0\n\t\t" << endl;
98
99             cout <<
"-----\n"
<< endl;
100             cout << "Seleccione una opcion escribiendo el número correspondiente.\n" << endl;
101             cout << ">>> ";
102             cin >> opcion;
103             try {
104                 switch (opcion) {
105                     case 0:
106                         goOut();
107                         break;
108                     case 1:
109                         goToSensors();
110                         break;
111                     case 2:
112                         goToSecurity();
113                         break;
114                     case 3:
115                         addUserDashboard();
116                         break;
117                     case 4:
118                         eraseUserDashboard();
119                         break;
120                     case 5:
121                         return;
122                     default:
123                         throw instructionException();
124                 }

```

```

125         } catch (instructionException &except) {
126             cout << "Exception: " << except.what() << endl;
127             system("sleep 2");
128         }
129     }
130 };

```

References option.

4.6.3.10 set_dashboardEmployer()

```
void Dashboard::set_dashboardEmployer ( )
```

Set the menu for the employers.

Definition at line 61 of file Dashboard.cpp.

```

61     {
62         while (true) {
63             cabecera();
64             cout << "\t\tOPCIONES DISPONIBLES\n\t\t-Sensores\t==> 1\n\t\t-Seguridad\t==> 2\n\t\t-Cerrar
Sesión\t==> 3\n\t\t-Exit\t\t==> 0\n\t\t" << endl;
65
66             cout <<
"-----\n"
<< endl;
67             cout << "Seleccione una opcion escribiendo el número correspondiente.\n" << endl;
68             cout << ">>> ";
69             cin >> opcion;
70             try {
71                 switch (opcion) {
72                     case 0:
73                         goOut();
74                         break;
75                     case 1:
76                         goToSensors();
77                         break;
78                     case 2:
79                         goToSecurity();
80                         break;
81                     case 3:
82                         return;
83                     default:
84                         throw instructionException();
85                 }
86             } catch (instructionException &except) {
87                 cout << "Exception: " << except.what() << endl;
88                 system("sleep 2");
89             }
90         }
91     };

```

References option.

4.6.4 Member Data Documentation

4.6.4.1 database

```
DataBase* Dashboard::database [private]
```

Definition at line 78 of file Dashboard.h.

4.6.4.2 opcion

```
int Dashboard::opcion [private]
```

Definition at line 76 of file Dashboard.h.

4.6.4.3 pantalla

```
int Dashboard::pantalla [private]
```

Definition at line 77 of file Dashboard.h.

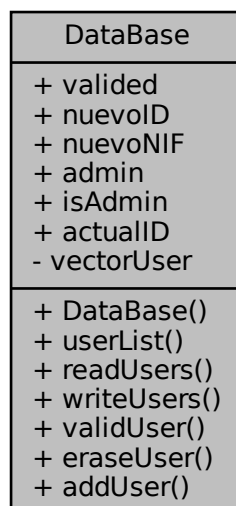
The documentation for this class was generated from the following files:

- [include/Dashboard.h](#)
- [src/Dashboard.cpp](#)

4.7 DataBase Class Reference

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

Collaboration diagram for DataBase:



Public Member Functions

- [DataBase](#) ()
Construct a new Data Base object.
- void [userList](#) ()
list of the current usersd
- void [readUsers](#) ()
Method to read the current users from the file users.dat.
- void [writeUsers](#) ()
Methot used for write in the file users.dat the users at the end of the program.
- bool [validUser](#) (int, int)
Validates the users credentials that are stored in the database.
- void [eraseUser](#) (int)
method to erase users
- void [addUser](#) ()
Method to add new users.

Public Attributes

- bool [valided](#)
bool to return the
- int [nuevoID](#)
- int [nuevoNIF](#)
- int [admin](#)
- bool [isAdmin](#)
- int [actualID](#)
User that is used to login.

Private Attributes

- std::set< [User](#) > [vectorUser](#)
Users storage.

4.7.1 Detailed Description

Definition at line 22 of file DataBase.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 DataBase()

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

Construct a new Data Base object.

Definition at line 25 of file DataBase.cpp.

```
25     {
26         /*User* user1 = new User(1111, 12345678, 0);
27         User* user2 = new User(2222, 23456789, 0);
28         User* user3 = new User(3333, 34567890, 0);
29         User* user4 = new User(1212, 12121212, 1);
30         vectorUser.insert(*user1);
31         vectorUser.insert(*user2);
32         vectorUser.insert(*user3);
33         vectorUser.insert(*user4);*/
34
35     }
```

4.7.3 Member Function Documentation

4.7.3.1 addUser()

```
void DataBase::addUser ( )
```

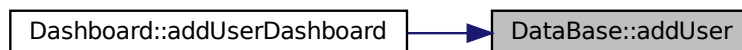
Method to add new users.

Definition at line 73 of file DataBase.cpp.

```
73     {
74
75         for (User u : vectorUser) {
76
77             if (u.getUserID() == nuevoID || u.getUserNIF() == nuevoNIF) {
78
79                 cout << "ID o NIF ya existentes" << endl;
80                 break;
81             } else {
82                 vectorUser.insert(User(nuevoNIF, nuevoID, admin));
83             }
84         }
85     }
```

References admin, nuevoID, and nuevoNIF.

Here is the caller graph for this function:



4.7.3.2 eraseUser()

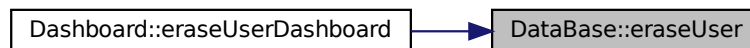
```
void DataBase::eraseUser (
    int posicion )
```

method to erase users

Definition at line 63 of file DataBase.cpp.

```
63     {
64
65     for (User u : vectorUser) {
66         if (posicion == u.getUserID() ) {
67             vectorUser.erase(u);
68             break;
69         }
70     }
71 }
```

Here is the caller graph for this function:



4.7.3.3 readUsers()

```
void DataBase::readUsers ( )
```

Method to read the current users from the file users.dat.

Definition at line 88 of file DataBase.cpp.

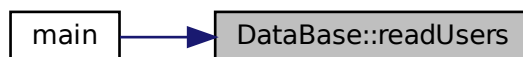
```
88     {
89         ifstream inUsersFile ("./resources/users.dat", ios::in | ios::binary);
90
91
92         if (!inUsersFile) { // file couldn't be opened
93             cerr << "File could not be opened" << endl;
94             exit (1);
95         }
96
97         User user;
98
99         inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
100
101         while (inUsersFile && !inUsersFile.eof()) {
102             if (user.getNumRecord() != 0) {
103                 vectorUser.insert (user);
104             }
105             inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
106         }
107
108 }
```

References User::getNumRecord().

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.3.4 userList()

```
void DataBase::userList ( )
```

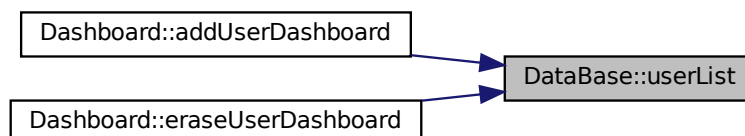
list of the current usersd

Definition at line 139 of file DataBase.cpp.

```

139     {
140         int posicion = 1;
141         for (User u : vectorUser) {
142             cout << "\t" << posicion << " -\t" << u.getUserID() << "\t" << u.getUserNIF() << endl;
143             posicion += 1;
144         }
145     }
  
```

Here is the caller graph for this function:



4.7.3.5 validUser()

```
bool DataBase::validUser (
    int id,
    int nif )
```

Validates the users credentials that are stored in the database.

Returns

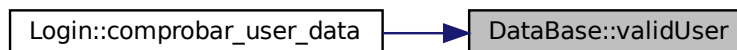
```
true
false
```

Definition at line 39 of file DataBase.cpp.

```
39         {
40
41     /* For each user in the list the atributes ID and NIF are compared whit
42        the ones introduced by the user from the login*/
43     for (User u : vectorUser) {
44
45         if (u.getUserID() == id && u.getUserNIF() == nif) {
46             // when the correct user is found the loop is over
47             validated = true;
48             if (u.getUserADMIN() == 1) {
49                 isAdmin = true;
50             } else {
51                 isAdmin = false;
52             }
53
54
55             break;
56         } else {
57             validated = false;
58         }
59     }
60     return validated;
61 }
```

References validated.

Here is the caller graph for this function:

**4.7.3.6 writeUsers()**

```
void DataBase::writeUsers ( )
```

Methot used for write in the file users.dat the users at the end of the program.

Definition at line 111 of file DataBase.cpp.

```
111     {
112         int numRecord = 1;
```

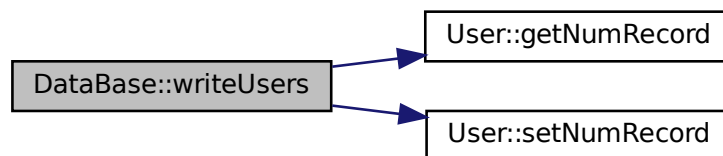
```

113
114     fstream outUsersFile ( "../resources/users.dat", ios::out | ios::binary | ios::trunc);
115
116
117     if (!outUsersFile) { // file couldn't be opened
118         cerr << "File could not be opened" << endl;
119         exit (1);
120     }
121
122
123     for (User u : vectorUser) {
124
125         User user(u.getUserNIF(), u.getUserID(), u.getUserADMIN());
126         user.setNumRecord (numRecord);
127
128         outUsersFile.seekp ((user.getNumRecord() - 1 ) *
129                             sizeof (User));
130
131         outUsersFile.write (reinterpret_cast <const char *> (&user),
132                             sizeof (User));
133         numRecord++;
134     }
135 }
136
137 }

```

References `User::getNumRecord()`, and `User::setNumRecord()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.4 Member Data Documentation

4.7.4.1 actualID

`int DataBase::actualID`

`User` that is used to login.

Definition at line 89 of file `DataBase.h`.

4.7.4.2 admin

```
int DataBase::admin
```

Definition at line 82 of file DataBase.h.

4.7.4.3 isAdmin

```
bool DataBase::isAdmin
```

Definition at line 83 of file DataBase.h.

4.7.4.4 nuevoID

```
int DataBase::nuevoID
```

Definition at line 80 of file DataBase.h.

4.7.4.5 nuevoNIF

```
int DataBase::nuevoNIF
```

Definition at line 81 of file DataBase.h.

4.7.4.6 valided

```
bool DataBase::valided
```

bool to return the

Definition at line 64 of file DataBase.h.

4.7.4.7 vectorUser

```
std::set<User> DataBase::vectorUser [private]
```

Users storage.

Definition at line 96 of file DataBase.h.

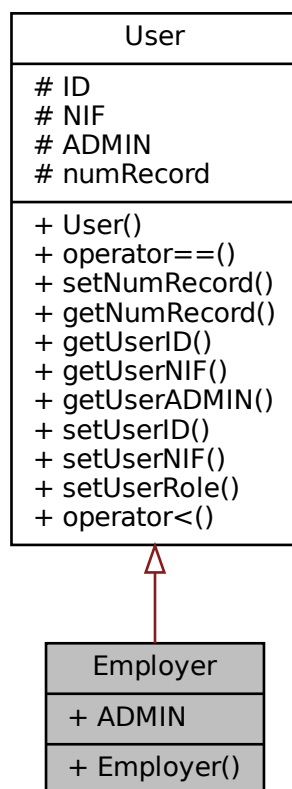
The documentation for this class was generated from the following files:

- include/[DataBase.h](#)
- src/[DataBase.cpp](#)

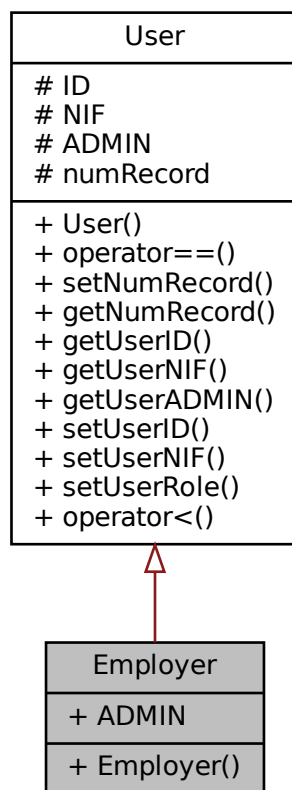
4.8 Employer Class Reference

```
#include <Employer.h>
```

Inheritance diagram for Employer:



Collaboration diagram for Employer:



Public Member Functions

- [Employer](#) ()

Public Attributes

- int [ADMIN](#) = 0

Additional Inherited Members

4.8.1 Detailed Description

Definition at line 20 of file Employer.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Employer()

```
Employer::Employer ( )
```

4.8.3 Member Data Documentation

4.8.3.1 ADMIN

```
int Employer::ADMIN = 0
```

Definition at line 23 of file Employer.h.

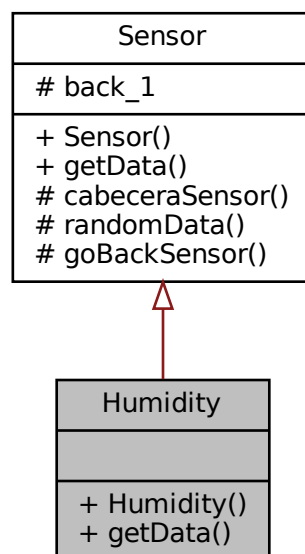
The documentation for this class was generated from the following file:

- [include/Employer.h](#)

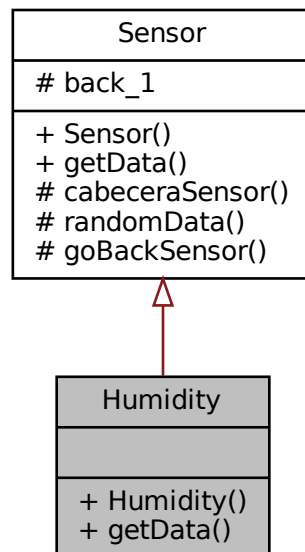
4.9 Humidity Class Reference

```
#include <Humidity.h>
```

Inheritance diagram for Humidity:



Collaboration diagram for Humidity:



Public Member Functions

- [Humidity](#) ()
Construct a new [Humidity](#) object.
- void [getData](#) ()
Get the Data object.

Additional Inherited Members

4.9.1 Detailed Description

Definition at line 23 of file Humidity.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Humidity()

```
Humidity::Humidity ( )
```

Construct a new [Humidity](#) object.

Definition at line 8 of file Humidity.cpp.

```
8 {}
```

4.9.3 Member Function Documentation

4.9.3.1 getData()

```
void Humidity::getData ( )
```

Get the Data object.

Definition at line 11 of file Humidity.cpp.

```
11         {
12     while (true) {
13         int random = this->randomData();
14         cout << "\n\t\tHumidity:\t\t" << (random + 30) << " g/m3\n" << endl;
15         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
16         cin >> this->back_1;
17         switch (this->back_1) {
18
19             case 0:
20                 return;
21                 break;
22
23             default:
24
25                 cout << "Orden incorrecta" << endl;
26                 system("sleep 1");
27                 this->cabeceraSensor();
28                 break;
29         }
30     }
31 }
```

References back_1.

Here is the caller graph for this function:



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

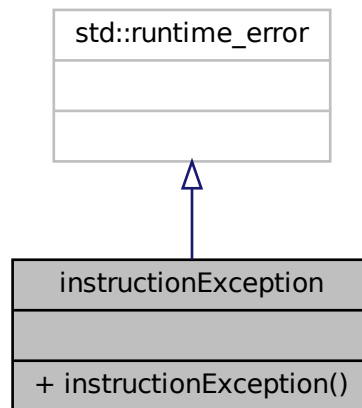
- [include/Humidity.h](#)
- [src/Humidity.cpp](#)

4.10 instructionException Class Reference

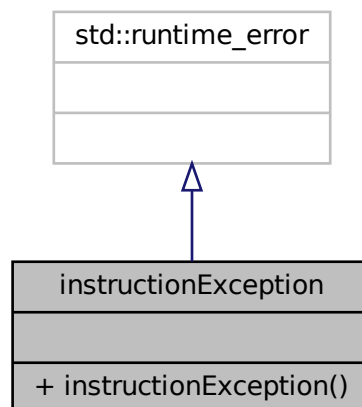
If the instruction of the menu is wrong, the exception appears.

```
#include <Exceptions.h>
```

Inheritance diagram for instructionException:



Collaboration diagram for instructionException:



Public Member Functions

- [instructionException](#) ()

4.10.1 Detailed Description

If the instruction of the menu is wrong, the exception appears.

Definition at line 41 of file Exceptions.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 instructionException()

```
instructionException::instructionException ( ) [inline]
```

Definition at line 43 of file Exceptions.h.

```
44      : std::runtime_error ("Instrucción incorrecta") {}
```

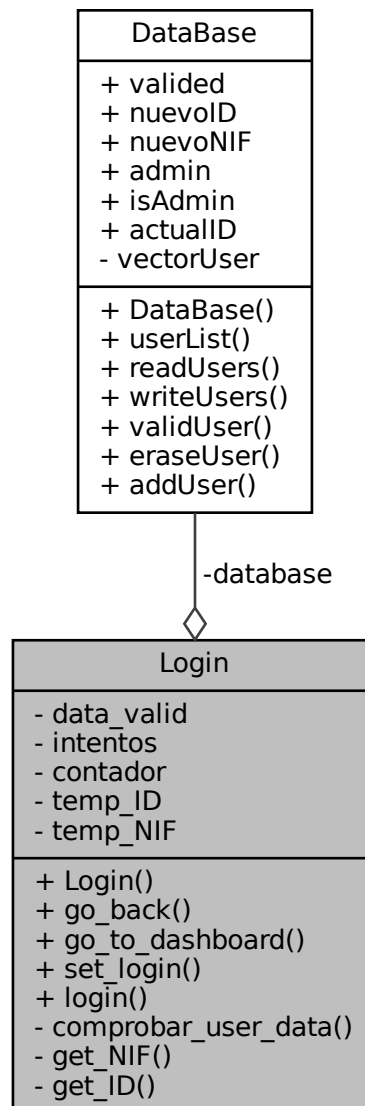
The documentation for this class was generated from the following file:

- [include/Exceptions.h](#)

4.11 Login Class Reference

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

Collaboration diagram for Login:



Public Member Functions

- [Login](#) ()
- void [go_back](#) ()
- bool [go_to_dashboard](#) ()
- void [set_login](#) ()
- void [login](#) ([DataBase](#) *)

Private Member Functions

- void [comprobar_user_data](#) ()

method that calls a database function which verify the credentials If the function return a false, a mesage is printed to show the user that is wrong At the third wrong try the program finishes with a 'access denied' error

- void `get_NIF` ()

Method to get the ID.

- void `get_ID` ()

Method to get the NIF.

Private Attributes

- `DataBase` * `database`

Variable of the data base type wich save a pointer.

- bool `data_valid`
- int `intentos`
- int `contador`
- int `temp_ID`
- int `temp_NIF`

4.11.1 Detailed Description

Definition at line 17 of file Login.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Login()

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

Definition at line 24 of file Login.cpp.

```
24 {} // Login constructor
```

4.11.3 Member Function Documentation

4.11.3.1 comprobar_user_data()

```
void Login::comprobar_user_data ( ) [private]
```

method that calls a database function which verify the credentials If the function return a false, a message is printed to show the user that is wrong At the third wrong try the program finishes with a 'access denied' error

Definition at line 78 of file Login.cpp.

```

78         {
79
80     try {
81         data_valid = database->validUser(temp_ID, temp_NIF);
82         if (!data_valid) {
83             throw credentialException();
84         }
85     }
86     catch (credentialException &except) {
87         cout << "\n\n\tException: " << except.what() << endl;
88         contador++;
89         system("sleep 1.5");
90         if (contador != 3) {
91             system("clear");
92         } else {
93             // at the third try the program finish and denied the access
94             cout << "\u001b[31m\n\t\tAccess denied.\u001b[0m\n" << endl;
95             go_back();
96         }
97     }
98 };

```

References contador, data_valid, database, temp_ID, temp_NIF, and DataBase::validUser().

Here is the call graph for this function:



4.11.3.2 get_ID()

```
void Login::get_ID ( ) [private]
```

Method to get the NIF.

Definition at line 39 of file Login.cpp.

```

39     {
40         string id;
41         cout << "\n\tID: ";
42         cin >> id;
43         try {
44             temp_ID = stoi(id);
45         }
46         catch (std::invalid_argument) {
47             cout << "Exception: chars and string are invalid" << endl;
48         }
49         catch (std::out_of_range) {
50             cout << "Exception: ID lenght out of range" << endl;
51             system("sleep 1");
52         }
53 };

```

References temp_ID.

4.11.3.3 get_NIF()

```
void Login::get_NIF ( ) [private]
```

Method to get the ID.

Definition at line 55 of file Login.cpp.

```
55     {
56         string nif;
57         cout << "\n\tNIF: ";
58         cin >> nif;
59         try {
60             temp_NIF = stoi(nif);
61         }
62         catch (std::invalid_argument) {
63             cout << "Exception: chars and string are invalid" << endl;
64             system("sleep 1");
65         }
66         catch (std::out_of_range) {
67             cout << "Exception: NIF lenght out of range" << endl;
68             system("sleep 1");
69         }
70     }
71 };
```

References temp_NIF.

4.11.3.4 go_back()

```
void Login::go_back ( )
```

Definition at line 109 of file Login.cpp.

```
109     {
110         exit(0);
111     };
```

4.11.3.5 go_to_dashboard()

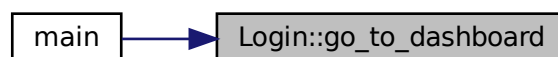
```
bool Login::go_to_dashboard ( )
```

Definition at line 100 of file Login.cpp.

```
100     {
101
102         system("sleep 1");
103         if (database->isAdmin) {
104             return true;
105         }
106         return false;
107     };
```

References database, and DataBase::isAdmin.

Here is the caller graph for this function:



4.11.3.6 login()

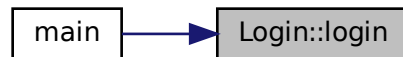
```
void Login::login (
    DataBase * data )
```

Definition at line 141 of file Login.cpp.

```
141
142     database = data;
143     system("clear");
144     set_login();
145     go_to_dashboard();
146
147 }
```

References database.

Here is the caller graph for this function:



4.11.3.7 set_login()

```
void Login::set_login ( )
```

Definition at line 115 of file Login.cpp.

```

115     {
116         while (true) {
117             // Title
118             cout << "\t\t\t-----" << endl;
119             cout << "\t\t\t\x1b[47m\x1b[30mLOGIN DE USUARIO\x1b[0m" << endl;
120             cout << "\t\t\t-----" << endl;
121
122             // The credentials are asked and are validated
123             get_ID();
124             get_NIF();
125
126             comprobar_user_data();
127
128
129             // If the credentials are correct the user goes to the dashboard
130             if (data_valid == true) {
131                 database->actualID = temp_ID;
132                 cout << database->actualID << endl;
133                 cout << "\u001b[32m\n\n\t\t\tBienvenido al sistema\u001b[0m" << endl;
134                 contador = 0;
135                 return;
136             }
137         }
138     };

```

References DataBase::actualID, contador, data_valid, database, and temp_ID.

4.11.4 Member Data Documentation

4.11.4.1 contador

```
int Login::contador [private]
```

Definition at line 53 of file Login.h.

4.11.4.2 data_valid

```
bool Login::data_valid [private]
```

Definition at line 51 of file Login.h.

4.11.4.3 database

```
DataBase* Login::database [private]
```

Variable of the data base type wich save a pointer.

Definition at line 49 of file Login.h.

4.11.4.4 intentos

```
int Login::intentos [private]
```

Definition at line 52 of file Login.h.

4.11.4.5 temp_ID

```
int Login::temp_ID [private]
```

Definition at line 54 of file Login.h.

4.11.4.6 temp_NIF

```
int Login::temp_NIF [private]
```

Definition at line 55 of file Login.h.

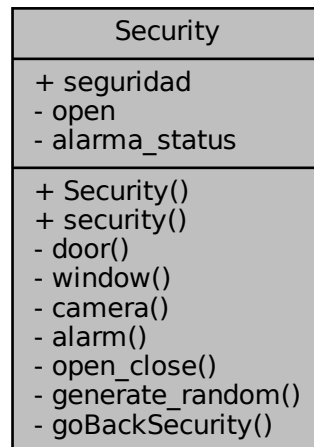
The documentation for this class was generated from the following files:

- [include/Login.h](#)
- [src/Login.cpp](#)

4.12 Security Class Reference

```
#include <Security.h>
```

Collaboration diagram for Security:



Public Member Functions

- [Security](#) ()
- void [security](#) (string)

Depend on the instruction, the security class shows a different menu.

Public Attributes

- string [seguridad](#)

Private Member Functions

- void [door](#) ()
- void [window](#) ()
- void [camera](#) ()
- void [alarm](#) ()
- void [open_close](#) ()
- int [generate_random](#) ()

Generate a random value and depend on it in some methods is used to generate a true or false value.

- void [goBackSecurity](#) ()

Private Attributes

- string [open](#)
- string [alarma_status](#)

4.12.1 Detailed Description

Definition at line 17 of file Security.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Security()

```
Security::Security ( )
```

Definition at line 24 of file Security.cpp.

```
24 {}
```

4.12.3 Member Function Documentation

4.12.3.1 alarm()

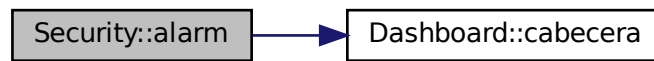
```
void Security::alarm ( ) [private]
```

Definition at line 118 of file Security.cpp.

```
118     {
119         while (true) {
120             cout << "\n\tAlarma: " << alarm\_status << "\n\n" << endl;
121             cout << "Para encender introduzca 1, para apagar introduzca 2, para salir 0\n" << endl;
122             cout << "»> ";
123             cin >> entrada;
124             switch (entrada) {
125
126                 case 0:
127                     return;
128
129                 case 1:
130                     alarm\_status = "encendido";
131                     db.cabecera\(\);
132                     break;
133
134                 case 2:
135                     alarm\_status = "apagado";
136                     db.cabecera\(\);
137                     break;
138                 default:
139                     cout << "Orden incorrecta" << endl;
140                     system("sleep 1");
141                     db.cabecera\(\);
142                     break;
143             }
144         }
145     }
```

References [alarm_status](#), [Dashboard::cabecera\(\)](#), [db](#), and [entrada](#).

Here is the call graph for this function:



4.12.3.2 camera()

```
void Security::camera ( ) [private]
```

Definition at line 100 of file Security.cpp.

```
100         {
101     while (true) {
102         cout << "Imagina que hay una camara" << endl;
103         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
104         cin >> back;
105         switch (back) {
106             case 0:
107                 return;
108                 break;
109             default:
110                 cout << "Orden incorrecta" << endl;
111                 system("sleep 1");
112                 db.cabecera();
113                 break;
114         }
115     }
116 }
```

References back, Dashboard::cabecera(), and db.

Here is the call graph for this function:



4.12.3.3 door()

void Security::door () [private]

Definition at line 42 of file Security.cpp.

```

42     {
43     while (true) {
44         open_close();
45         cout << "Puerta: 01\tLocalización: Huerto\tEstado: " << open << endl;
46         open_close();
47         cout << "Puerta: 02\tLocalización: Almacen\tEstado: " << open << endl;
48         open_close();
49         cout << "Puerta: 03\tLocalización: Despacho\tEstado: " << open << endl;
50         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
51         cout << ">> ";
52         cin >> back;
53     }
54     // If an order different of 0 is inserted, the door menu is restarted.
55     switch (back) {
56     case 0:
57         return;
58         break;
59     default:
60         cout << "Orden incorrecta" << endl;
61         system("sleep 1");
62         db.cabecera();
63         break;
64     }
65 }
66 }
67 }
68 }
69 }
```

References back, Dashboard::cabecera(), db, and open.

Here is the call graph for this function:



4.12.3.4 generate_random()

int Security::generate_random () [private]

Generate a random value and depend on it in some methods is used to generate a true or false value.

Returns

int

Definition at line 148 of file Security.cpp.

```

148     {
149         srand(time(NULL));
150         return rand()%10;
151     }
```

4.12.3.5 goBackSecurity()

```
void Security::goBackSecurity ( ) [private]
```

4.12.3.6 open_close()

```
void Security::open_close ( ) [private]
```

Definition at line 154 of file Security.cpp.

```
154     {
155         int random = generate_random();
156         if (random >= 0 && random <= 4) {
157             open = "close";
158         } else {
159             open = "open";
160         }
161     }
```

References open.

4.12.3.7 security()

```
void Security::security (
    string seguridad )
```

Depend on the instruction, the security class shows a different menu.

Parameters

<i>string</i>	seguridad, depending on it's "value" the class shows a different menu
---------------	---

Definition at line 29 of file Security.cpp.

```
29     {
30         if (seguridad == "puertas") {
31             door();
32         } else if (seguridad == "ventanas") {
33             window();
34         } else if (seguridad == "camaras") {
35             camera();
36         } else if (seguridad == "alarma") {
37             alarm();
38         }
39     }
```

References seguridad.

Here is the caller graph for this function:



4.12.3.8 window()

```
void Security::window ( ) [private]
```

Definition at line 72 of file Security.cpp.

```

72         {
73     while (true) {
74         open_close();
75         cout << "Ventana: 01\tLocalización: Huerto\tEstado: " << open << endl;
76         open_close();
77         cout << "Ventana: 02\tLocalización: Almacen\tEstado: " << open << endl;
78         open_close();
79         cout << "Ventana: 03\tLocalización: Despacho\tEstado: " << open << endl;
80         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
81         cin >> back;
82         // if an order different of 0 the programm restart
83         switch (back) {
84
85             case 0:
86                 return;
87                 break;
88
89             default:
90
91                 cout << "Orden incorrecta" << endl;
92                 system("sleep 1");
93                 db.cabecera();
94                 break;
95         }
96     }
97 }
```

References back, Dashboard::cabecera(), db, and open.

Here is the call graph for this function:



4.12.4 Member Data Documentation

4.12.4.1 alarma_status

```
string Security::alarma_status [private]
```

Definition at line 49 of file Security.h.

4.12.4.2 open

```
string Security::open [private]
```

Definition at line 48 of file Security.h.

4.12.4.3 seguridad

```
string Security::seguridad
```

Definition at line 20 of file Security.h.

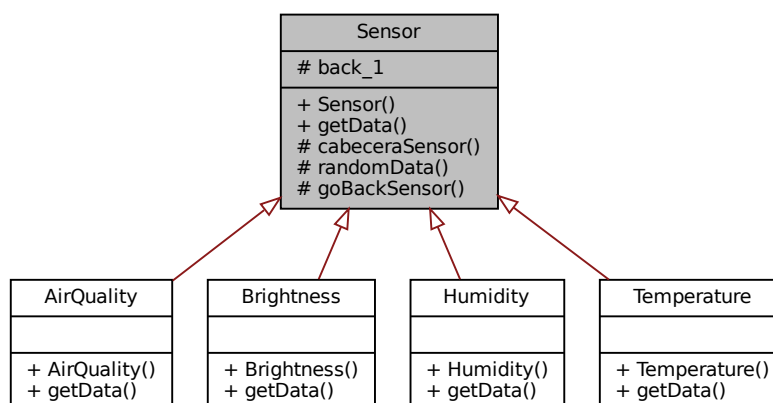
The documentation for this class was generated from the following files:

- [include/Security.h](#)
- [src/Security.cpp](#)

4.13 Sensor Class Reference

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

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:

Sensor
back_1
+ Sensor() + getData() # cabeceraSensor() # randomData() # goBackSensor()

Public Member Functions

- [Sensor](#) ()
- void [getData](#) (int)
Get the Data of the object.

Protected Member Functions

- void [cabeceraSensor](#) ()
Heading used in all the sensors.
- int [randomData](#) ()
Method that generate a random number which is used to show a value for each sensor.
- void [goBackSensor](#) ()

Protected Attributes

- int [back_1](#)

4.13.1 Detailed Description

Definition at line 18 of file Sensor.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Sensor()

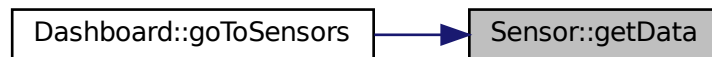
```
Sensor::Sensor ( )
```

Definition at line 15 of file Sensor.cpp.
15 {}


```
56         case 0:
57             return;
58             break;
59
60         default:
61
62             cout << "Orden incorrecta" << endl;
63             system("sleep 1");
64             cabeceraSensor();
65             break;
66     }
67 }
68 }
69 }
```

References back_1, and opcion.

Here is the caller graph for this function:



4.13.3.3 goBackSensor()

```
void Sensor::goBackSensor ( ) [protected]
```

4.13.3.4 randomData()

```
int Sensor::randomData ( ) [protected]
```

Method that generate a random number which is used to show a value for each sensor.

Returns

int

Definition at line 21 of file Sensor.cpp.

```
21     {
22         srand(time(NULL));
23         return rand()%10;
24     }
```

4.13.4 Member Data Documentation

4.13.4.1 back_1

```
int Sensor::back_1 [protected]
```

Definition at line 35 of file Sensor.h.

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

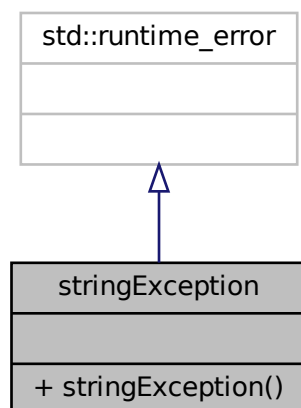
- [include/Sensor.h](#)
- [src/Sensor.cpp](#)

4.14 stringException Class Reference

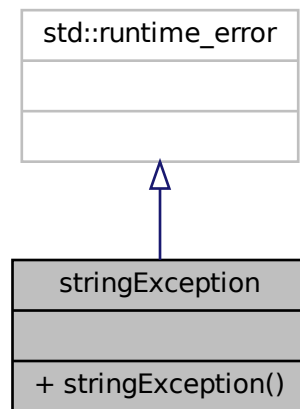
If a string is introduced when an int is needed an exception will be thrown.

```
#include <Exceptions.h>
```

Inheritance diagram for stringException:



Collaboration diagram for `stringException`:



Public Member Functions

- [stringException](#) ()

4.14.1 Detailed Description

If a string is introduced when an int is needed an exception will be thrown.

Definition at line 21 of file `Exceptions.h`.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 stringException()

```
stringException::stringException ( ) [inline]
```

Definition at line 23 of file `Exceptions.h`.

```
24 : std::runtime_error ("No se admiten letras o palabras, introduzca de nuevo") {}
```

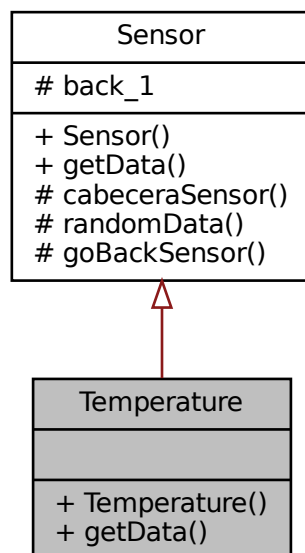
The documentation for this class was generated from the following file:

- [include/Exceptions.h](#)

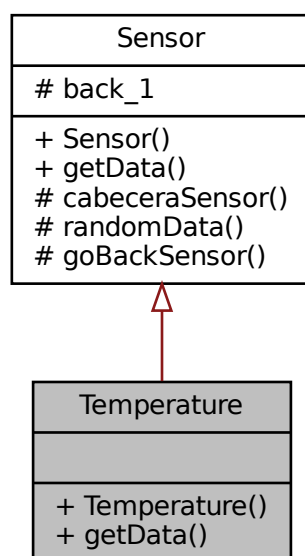
4.15 Temperature Class Reference

```
#include <Temperature.h>
```

Inheritance diagram for Temperature:



Collaboration diagram for Temperature:



Public Member Functions

- [Temperature](#) ()
Construct a new [Temperature](#) object.
- void [getData](#) ()
Get the object data.

Additional Inherited Members

4.15.1 Detailed Description

Definition at line 20 of file `Temperature.h`.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 [Temperature\(\)](#)

```
Temperature::Temperature ( )
```

Construct a new [Temperature](#) object.

Definition at line 8 of file `Temperature.cpp`.
8 {}

4.15.3 Member Function Documentation

4.15.3.1 [getData\(\)](#)

```
void Temperature::getData ( )
```

Get the object data.

Definition at line 11 of file `Temperature.cpp`.

```
11                                     {
12     while (true) {
13         int random = this->randomData();
14         cout << "\n\t\tTemperature:\t\t" << (random + 20) << " °C\n" << endl;
15         cout << "\n\t\tPara volver atrás introduzca 0 + ENTER" << endl;
16         cin >> this->back_1;
17         switch (this->back_1) {
18
19             case 0:
20                 return;
21                 break;
22
23             default:
24
25                 cout << "Orden incorrecta" << endl;
26                 system("sleep 1");
27                 this->cabeceraSensor();
```

```
28         break;  
29     }  
30 }  
31  
32 }
```

References back_1.

Here is the caller graph for this function:



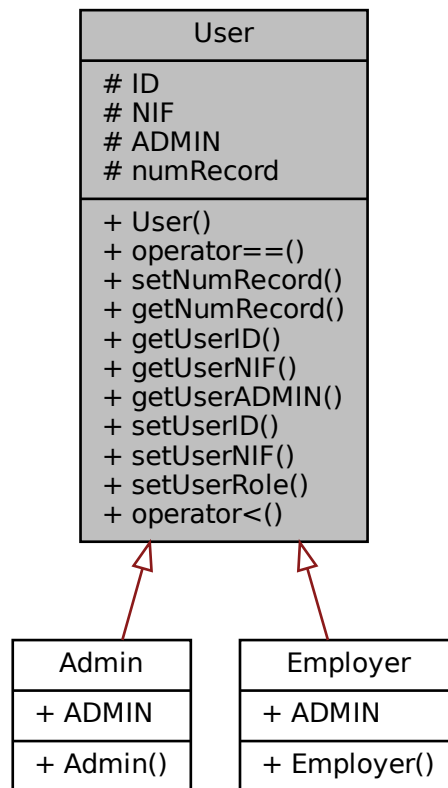
The documentation for this class was generated from the following files:

- [include/Temperature.h](#)
- [src/Temperature.cpp](#)

4.16 User Class Reference

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

Inheritance diagram for User:



Collaboration diagram for User:

User
ID # NIF # ADMIN # numRecord
+ User() + operator==() + setNumRecord() + getNumRecord() + getUserID() + getUserNIF() + getUserADMIN() + setUserID() + setUserNIF() + setUserRole() + operator<()

Public Member Functions

- `User` (int=0, int=0, int=0)
Construct a new `User` object.
- bool `operator==` (const `User` &) const
- void `setNumRecord` (int)
- int `getNumRecord` () const
- int `getUserID` () const
Get the `User` ID.
- int `getUserNIF` () const
Get the `User` NIF.
- int `getUserADMIN` () const
Get the `User` ADMIN.
- void `setUserID` (int)
- void `setUserNIF` (int)
- void `setUserRole` (int)
- bool `operator<` (const `User` &) const
overloading of the operator < to compare users

Protected Attributes

- int `ID`
`User` attributes.
- int `NIF`
- int `ADMIN`
- int `numRecord`

4.16.1 Detailed Description

Definition at line 14 of file User.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 User()

```
User::User (
    int id = 0,
    int nif = 0,
    int admin = 0 )
```

Construct a new [User](#) object.

Parameters

<i>int</i>	the first is the ID
<i>int</i>	the second is the NIF
<i>int</i>	the third is the role of the user

Definition at line 18 of file User.cpp.

```
18                                     { // Constructor
19     setUserID (id);
20     setUserNIF (nif);
21     setUserRole (admin);
22 }
```

References [admin](#).

4.16.3 Member Function Documentation

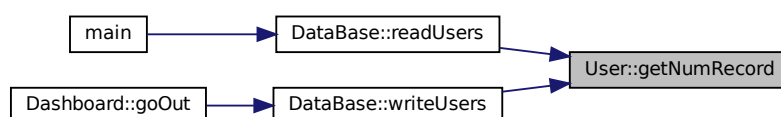
4.16.3.1 getNumRecord()

```
int User::getNumRecord ( ) const
```

Definition at line 62 of file User.cpp.

```
62     {
63         return numRecord;
64     }
```

Here is the caller graph for this function:



4.16.3.2 getUserADMIN()

```
int User::getUserADMIN ( ) const
```

Get the [User](#) ADMIN.

Returns

int

Definition at line 26 of file User.cpp.

```
26 {  
27     return ADMIN;  
28 }
```

References ADMIN.

4.16.3.3 getUserID()

```
int User::getUserID ( ) const
```

Get the [User](#) ID.

Returns

int

Definition at line 30 of file User.cpp.

```
30 {  
31     return ID;  
32 }
```

References ID.

4.16.3.4 getUserNIF()

```
int User::getUserNIF ( ) const
```

Get the [User](#) NIF.

Returns

int

Definition at line 34 of file User.cpp.

```
34 {  
35     return NIF;  
36 }
```

References NIF.

4.16.3.5 operator<()

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

overloading of the operator < to compare users

Returns

true

false

Definition at line 51 of file User.cpp.

```
51 {
52     return ID < user.ID;
53 }
```

References ID, and ID.

4.16.3.6 operator==()

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

Definition at line 55 of file User.cpp.

```
55 {
56     if (ID == user.ID && NIF == user.NIF) {
57         return true;
58     }
59     return false;
60 }
```

References ID, ID, NIF, and NIF.

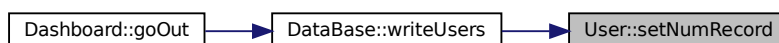
4.16.3.7 setNumRecord()

```
void User::setNumRecord (
    int record )
```

Definition at line 66 of file User.cpp.

```
66 {
67     numRecord = record;
68 }
```

Here is the caller graph for this function:



4.16.3.8 setUserID()

```
void User::setUserID (
    int myId )
```

Definition at line 46 of file User.cpp.

```
46     {
47     NIF = myId;
48 }
```

References NIF.

4.16.3.9 setUserNIF()

```
void User::setUserNIF (
    int myNif )
```

Definition at line 42 of file User.cpp.

```
42     {
43     ID = myNif;
44 }
```

References ID.

4.16.3.10 setUserRole()

```
void User::setUserRole (
    int myRole )
```

Definition at line 38 of file User.cpp.

```
38     {
39     ADMIN = myRole;
40 }
```

References ADMIN.

4.16.4 Member Data Documentation

4.16.4.1 ADMIN

```
int User::ADMIN [protected]
```

Definition at line 76 of file User.h.

4.16.4.2 ID

```
int User::ID [protected]
```

[User](#) attributes.

Definition at line 74 of file User.h.

4.16.4.3 NIF

```
int User::NIF [protected]
```

Definition at line 75 of file User.h.

4.16.4.4 numRecord

```
int User::numRecord [protected]
```

Definition at line 77 of file User.h.

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

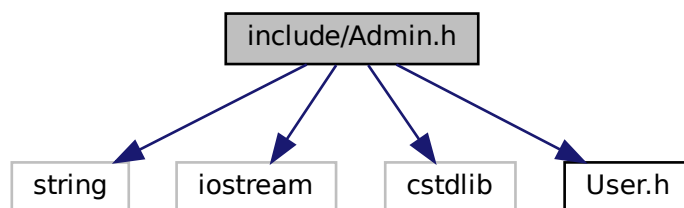
- [include/User.h](#)
- [src/User.cpp](#)

Chapter 5

File Documentation

5.1 include/Admin.h File Reference

```
#include <string>
#include <iostream>
#include <cstdlib>
#include "User.h"
Include dependency graph for Admin.h:
```



Classes

- class [Admin](#)

5.1.1 Detailed Description

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-12-08

Copyright

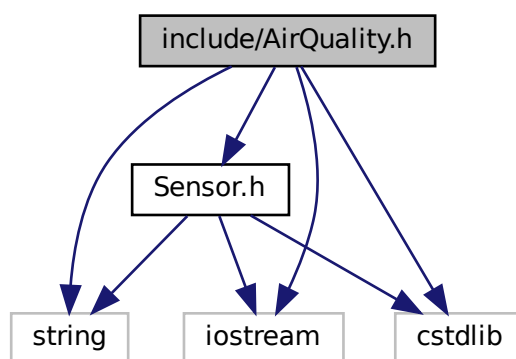
Copyright (c) 2022

5.2 include/AirQuality.h File Reference

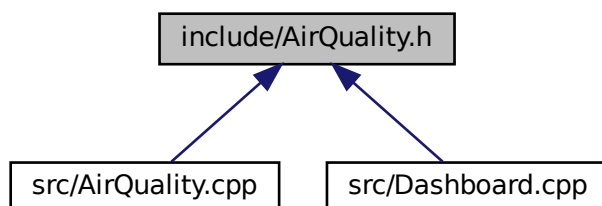
Air Quality sensor.

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

Include dependency graph for AirQuality.h:



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



Classes

- class [AirQuality](#)

5.2.1 Detailed Description

Air Quality sensor.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

Copyright

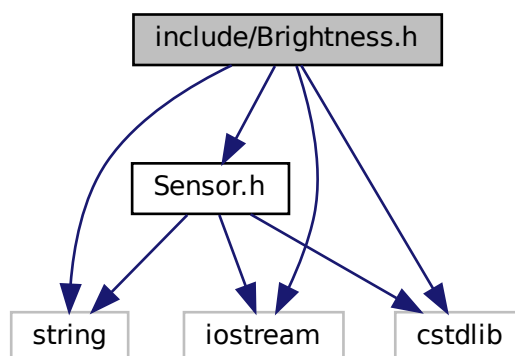
Copyright (c) 2022

5.3 include/Brightness.h File Reference

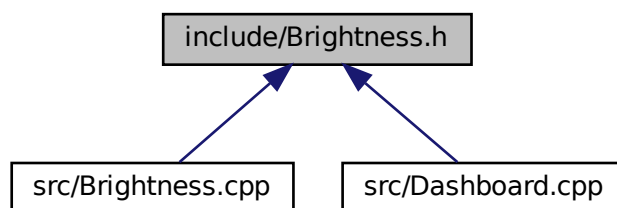
Luminosity sensor.

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

Include dependency graph for Brightness.h:



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



Classes

- class [Brightness](#)

5.3.1 Detailed Description

Luminosity sensor.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

Copyright

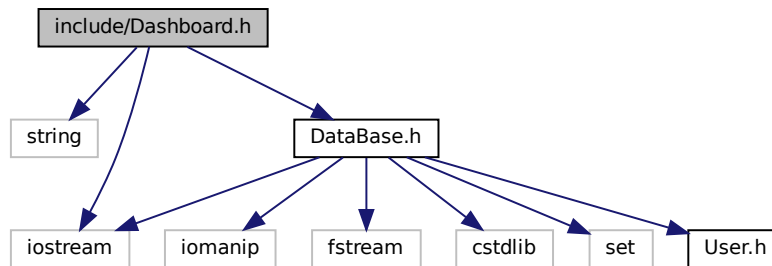
Copyright (c) 2022

5.4 include/Dashboard.h File Reference

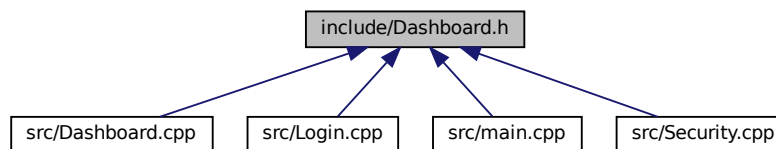
main menu

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

Include dependency graph for Dashboard.h:



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



Classes

- class [Dashboard](#)

5.4.1 Detailed Description

main menu

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

3

Date

2022-11-30

Copyright

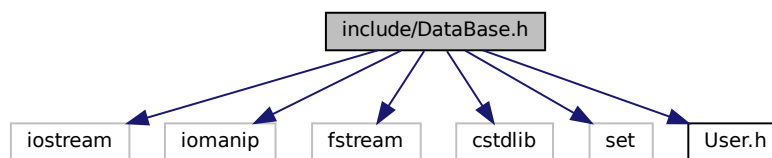
Copyright (c) 2022

5.5 include/DataBase.h File Reference

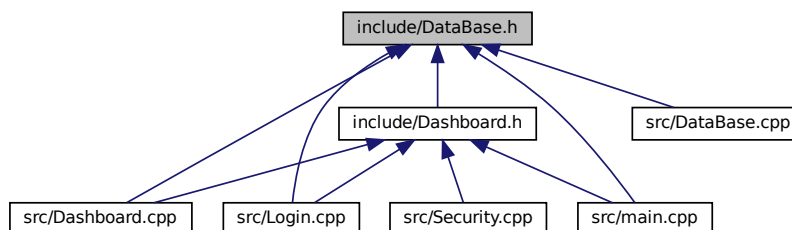
[DataBase](#) implementation and validate users's methods.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <set>
#include "User.h"
```

Include dependency graph for DataBase.h:



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



Classes

- class [DataBase](#)

5.5.1 Detailed Description

[DataBase](#) implementation and validate users's methods.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

3.0

Date

2022-11-30

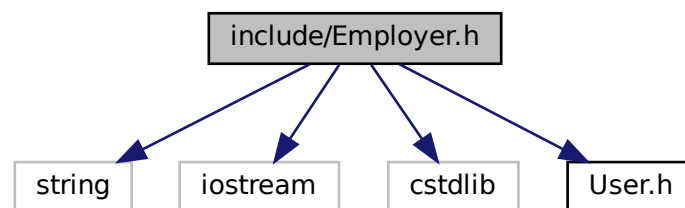
Copyright

Copyright (c) 2022

5.6 include/Employer.h File Reference

Object employer.

```
#include <string>
#include <iostream>
#include <cstdlib>
#include "User.h"
Include dependency graph for Employer.h:
```



Classes

- class [Employer](#)

5.6.1 Detailed Description

Object employer.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-12-08

Copyright

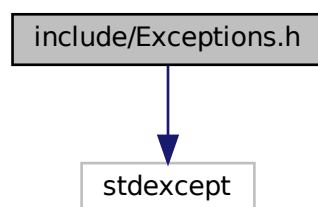
Copyright (c) 2022

5.7 include/Exceptions.h File Reference

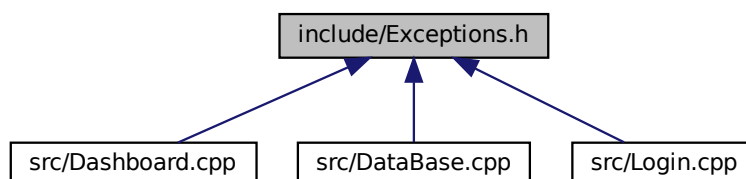
Exception that may occur in the interface.

```
#include <stdexcept>
```

Include dependency graph for Exceptions.h:



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



Classes

- class [stringException](#)
If a string is introduced when an int is needed an exception will be thrown.
- class [credentialException](#)
If the ID or NIF are incorrect, the exception will be thrown.
- class [instructionException](#)
If the instruction of the menu is wrong, the exception appears.
- class [actualUserException](#)

5.7.1 Detailed Description

Exception that may occur in the interface.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-12-18

Copyright

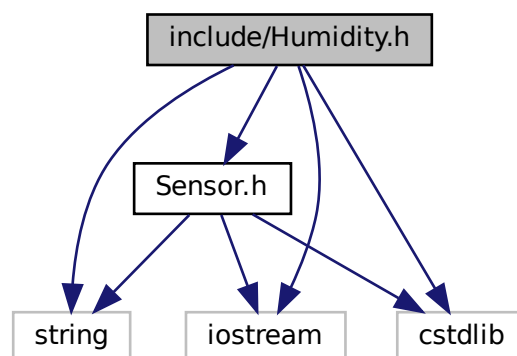
Copyright (c) 2022

5.8 include/Humidity.h File Reference

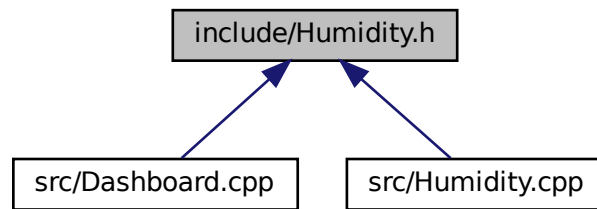
[Humidity](#) sensor.

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

Include dependency graph for Humidity.h:



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



Classes

- class [Humidity](#)

5.8.1 Detailed Description

[Humidity](#) sensor.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

Copyright

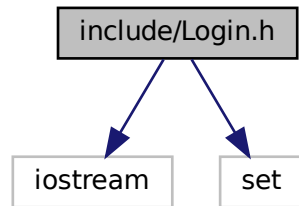
Copyright (c) 2022

5.9 include/Login.h File Reference

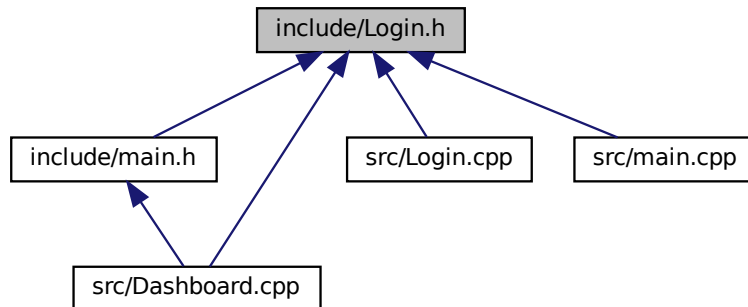
Login interface.

```
#include <iostream>
#include <set>
```

Include dependency graph for Login.h:



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



Classes

- class [Login](#)

5.9.1 Detailed Description

Login interface.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

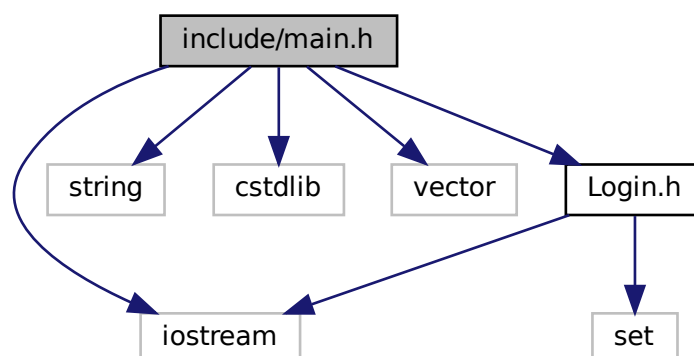
Copyright

Copyright (c) 2022

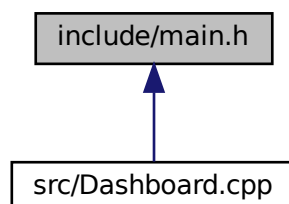
5.10 include/main.h File Reference

Main program.

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <vector>
#include "Login.h"
Include dependency graph for main.h:
```



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



Variables

- [Login l](#)
- [Dashboard d](#)

5.10.1 Detailed Description

Main program.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

2

Date

2022-11-30

Copyright

Copyright (c) 2022

5.10.2 Variable Documentation

5.10.2.1 d

[Dashboard d](#)

Definition at line 22 of file main.h.

5.10.2.2 l

[Login l](#)

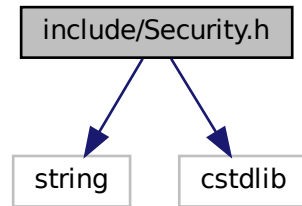
Definition at line 21 of file main.h.

5.11 include/Security.h File Reference

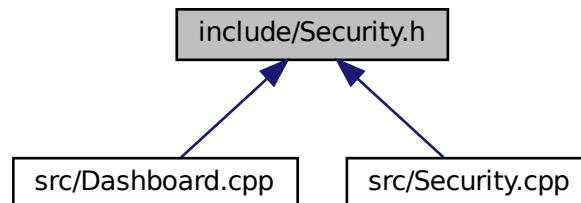
[Security](#) methods.

```
#include <string>
#include <cstdlib>
```

Include dependency graph for Security.h:



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



Classes

- class [Security](#)

5.11.1 Detailed Description

[Security](#) methods.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

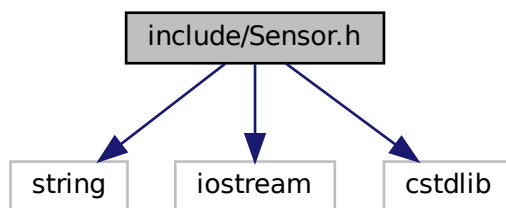
Copyright

Copyright (c) 2022

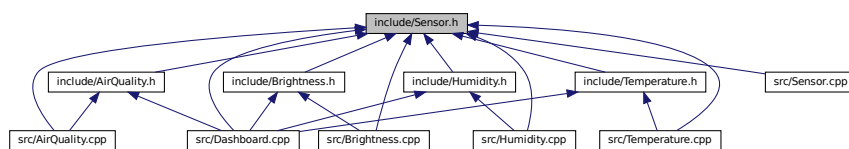
5.12 include/Sensor.h File Reference

```
#include <string>
#include <iostream>
#include <cstdlib>
```

Include dependency graph for Sensor.h:



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



Classes

- class [Sensor](#)

5.12.1 Detailed Description

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

Copyright

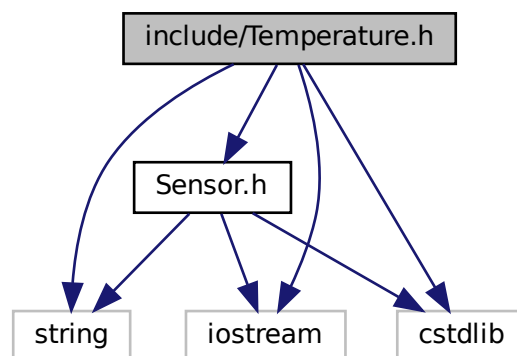
Copyright (c) 2022

5.13 include/Temperature.h File Reference

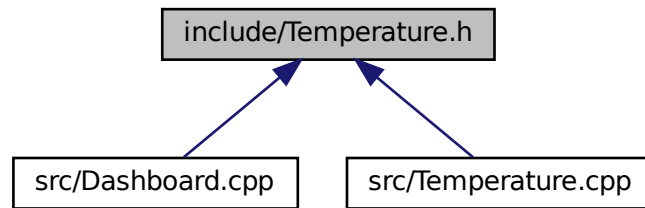
[Temperature](#) sensor.

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

Include dependency graph for Temperature.h:



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



Classes

- class [Temperature](#)

5.13.1 Detailed Description

[Temperature](#) sensor.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

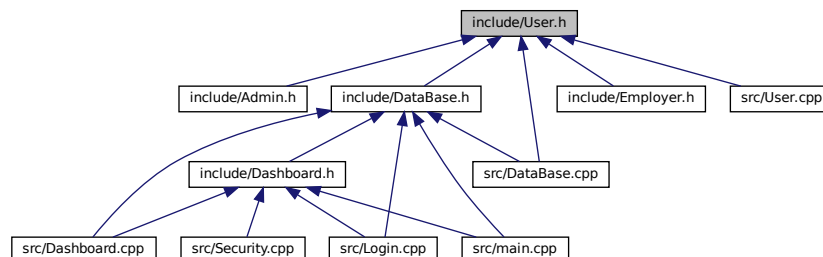
Copyright

Copyright (c) 2022

5.14 include/User.h File Reference

[User](#) object's file.

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



Classes

- class [User](#)

5.14.1 Detailed Description

[User](#) object's file.

Author

Iker Peral del Pino (i.peral.2021@alumnos.urjc.es)

Version

0.1

Date

2022-11-30

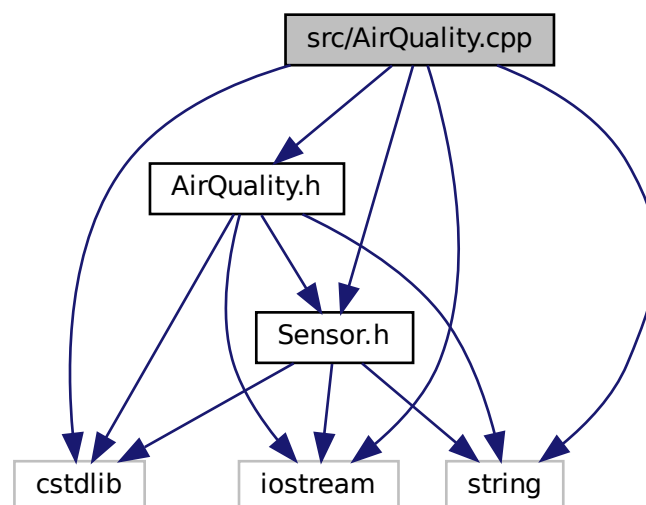
Copyright

Copyright (c) 2022

5.15 src/AirQuality.cpp File Reference

```
#include <iostream>
#include <string>
#include <cstdlib>
#include "Sensor.h"
#include "AirQuality.h"
```

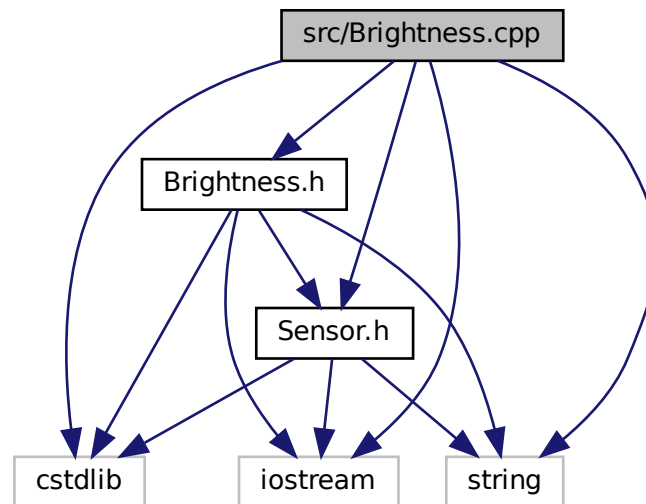
Include dependency graph for AirQuality.cpp:



5.16 src/Brightness.cpp File Reference

```
#include <iostream>
#include <string>
#include <cstdlib>
#include "Sensor.h"
#include "Brightness.h"
```

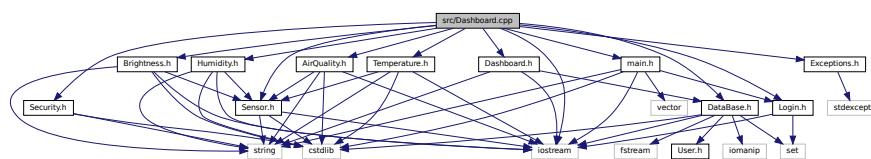
Include dependency graph for Brightness.cpp:



5.17 src/Dashboard.cpp File Reference

```
#include <iostream>
#include "Temperature.h"
#include "Humidity.h"
#include "Brightness.h"
#include "AirQuality.h"
#include "Security.h"
#include "Sensor.h"
#include "DataBase.h"
#include "Dashboard.h"
#include "main.h"
#include "Login.h"
#include "Exceptions.h"
```

Include dependency graph for Dashboard.cpp:



Variables

- `int opcion`
- `Security s`
- `Sensor sen`
- `Temperature temp`
- `Brightness brigh`
- `Humidity hum`
- `AirQuality airq`
- `DataBase * database`

5.17.1 Variable Documentation

5.17.1.1 `airq`

`AirQuality airq`

Definition at line 36 of file Dashboard.cpp.

5.17.1.2 `brigh`

`Brightness brigh`

Definition at line 34 of file Dashboard.cpp.

5.17.1.3 `database`

`DataBase* database`

Definition at line 38 of file Dashboard.cpp.

5.17.1.4 `hum`

`Humidity hum`

Definition at line 35 of file Dashboard.cpp.

5.17.1.5 opcion

`int opcion`

Definition at line 24 of file Dashboard.cpp.

5.17.1.6 s

`Security s`

Definition at line 31 of file Dashboard.cpp.

5.17.1.7 sen

`Sensor sen`

Definition at line 32 of file Dashboard.cpp.

5.17.1.8 temp

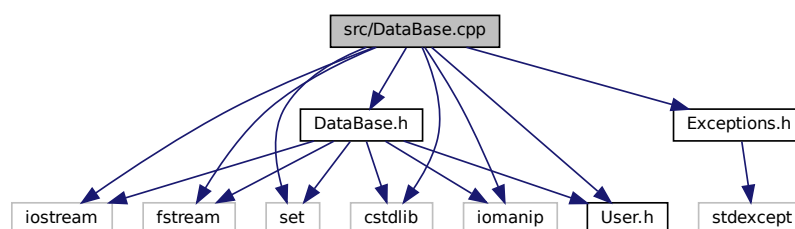
`Temperature temp`

Definition at line 33 of file Dashboard.cpp.

5.18 src/DataBase.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <set>
#include <cstdlib>
#include <iomanip>
#include "DataBase.h"
#include "User.h"
#include "Exceptions.h"
```

Include dependency graph for DataBase.cpp:



Variables

- bool [valided](#)
- int [nuevoID](#)
- int [nuevoNIF](#)
- int [admin](#)

5.18.1 Variable Documentation

5.18.1.1 admin

```
int admin
```

Definition at line 22 of file DataBase.cpp.

5.18.1.2 nuevoID

```
int nuevoID
```

Definition at line 20 of file DataBase.cpp.

5.18.1.3 nuevoNIF

```
int nuevoNIF
```

Definition at line 21 of file DataBase.cpp.

5.18.1.4 valided

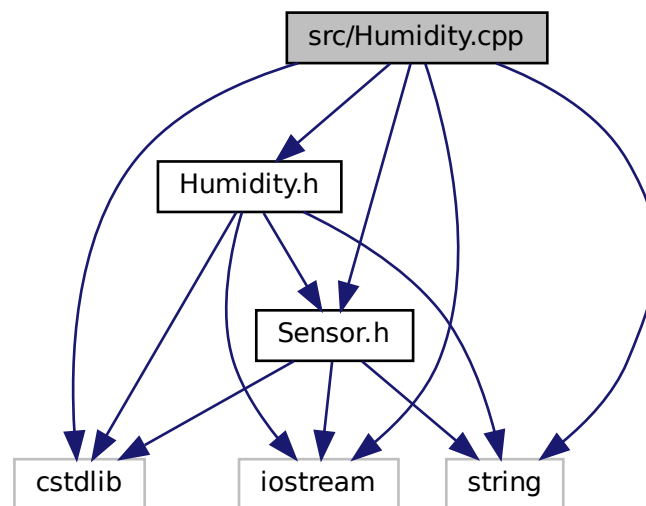
```
bool valided
```

Definition at line 19 of file DataBase.cpp.

5.19 src/Humidity.cpp File Reference

```
#include <iostream>
#include <string>
#include <cstdlib>
#include "Sensor.h"
#include "Humidity.h"
```

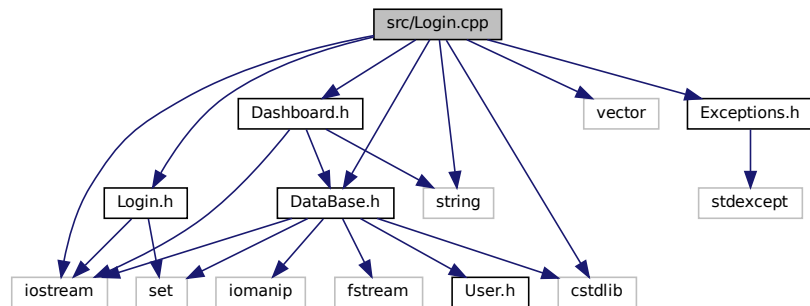
Include dependency graph for Humidity.cpp:



5.20 src/Login.cpp File Reference

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <vector>
#include "Dashboard.h"
#include "DataBase.h"
#include "Login.h"
#include "Exceptions.h"
```

Include dependency graph for Login.cpp:



Variables

- int `temp_ID`
- int `temp_NIF`
- bool `data_valid` = false
- int `contador` = 0
- int `intentos` = 3

5.20.1 Variable Documentation

5.20.1.1 contador

```
int contador = 0
```

Definition at line 33 of file Login.cpp.

5.20.1.2 data_valid

```
bool data_valid = false
```

Definition at line 30 of file Login.cpp.

5.20.1.3 intentos

```
int intentos = 3
```

Definition at line 34 of file Login.cpp.

5.20.1.4 temp_ID

```
int temp_ID
```

Definition at line 27 of file Login.cpp.

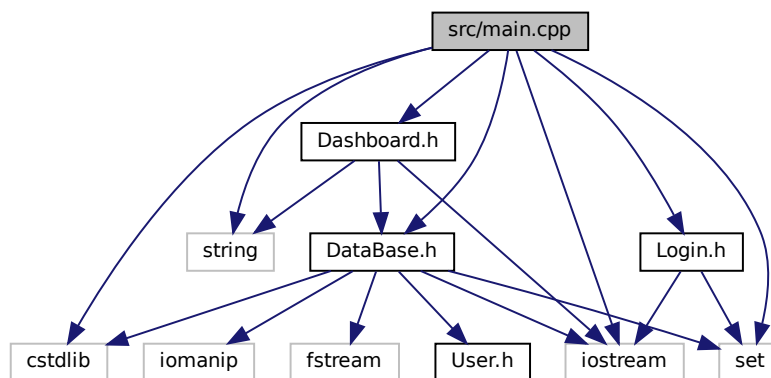
5.20.1.5 temp_NIF

```
int temp_NIF
```

Definition at line 28 of file Login.cpp.

5.21 src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <set>
#include "DataBase.h"
#include "Login.h"
#include "Dashboard.h"
Include dependency graph for main.cpp:
```



Functions

- int [main](#) ()

5.21.1 Function Documentation

5.21.1.1 main()

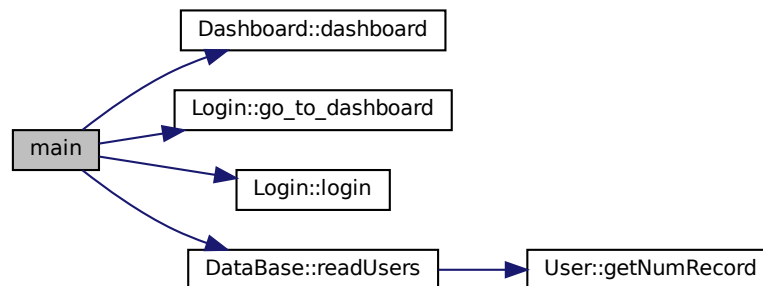
```
int main ( )
```

Definition at line 19 of file main.cpp.

```
19     {
20     DataBase* ptrbase;
21     try {
22         ptrbase = new DataBase();
23     } catch (bad_alloc &except) {
24         cerr << "Exception!: " << except.what() << endl;
25     }
26
27     ptrbase->readUsers();
28     // a login and a dashboard objects are created
29     Login l;
30     Dashboard d;
31     while (true) {
32
33         system("clear"); // used in lots of functions to clear the screen
34         l.login(ptrbase); // initialize the login screen
35         d.dashboard(ptrbase, l.go_to_dashboard());
36     }
37     return 0;
38 }
```

References d, Dashboard::dashboard(), Login::go_to_dashboard(), l, Login::login(), and DataBase::readUsers().

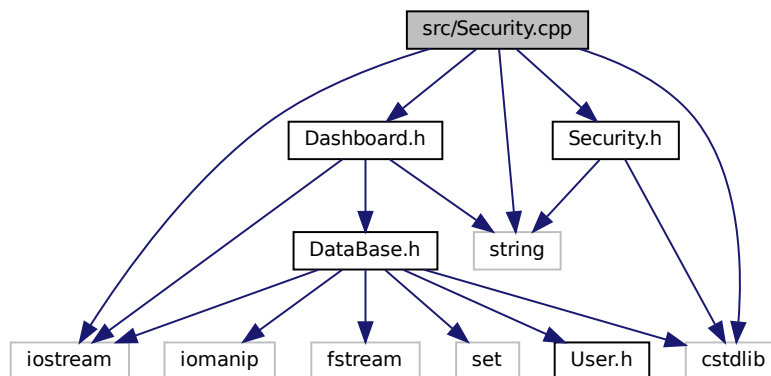
Here is the call graph for this function:



5.22 src/Security.cpp File Reference

```
#include <iostream>
#include <string>
#include <cstdlib>
#include "Security.h"
#include "Dashboard.h"
```


Include dependency graph for Security.cpp:



Variables

- string `seguridad`
- string `open`
- string `alarm_status` = "encendido"
- int `entrada`
- int `back`
- `Dashboard db`

5.22.1 Variable Documentation

5.22.1.1 alarm_status

```
string alarm_status = "encendido"
```

Definition at line 19 of file `Security.cpp`.

5.22.1.2 back

```
int back
```

Definition at line 22 of file `Security.cpp`.

5.22.1.3 db

[Dashboard](#) db

Definition at line 26 of file Security.cpp.

5.22.1.4 entrada

`int entrada`

Definition at line 20 of file Security.cpp.

5.22.1.5 open

`string open`

Definition at line 18 of file Security.cpp.

5.22.1.6 seguridad

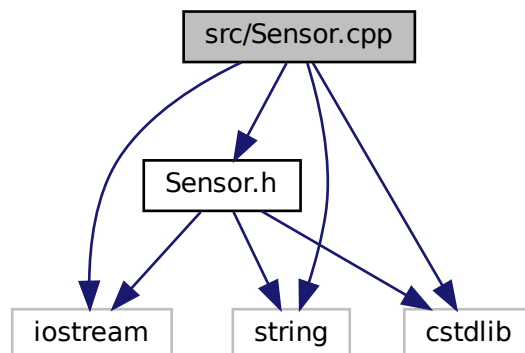
`string seguridad`

Definition at line 17 of file Security.cpp.

5.23 src/Sensor.cpp File Reference

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

Include dependency graph for Sensor.cpp:



Variables

- int [back_1](#)

5.23.1 Variable Documentation

5.23.1.1 back_1

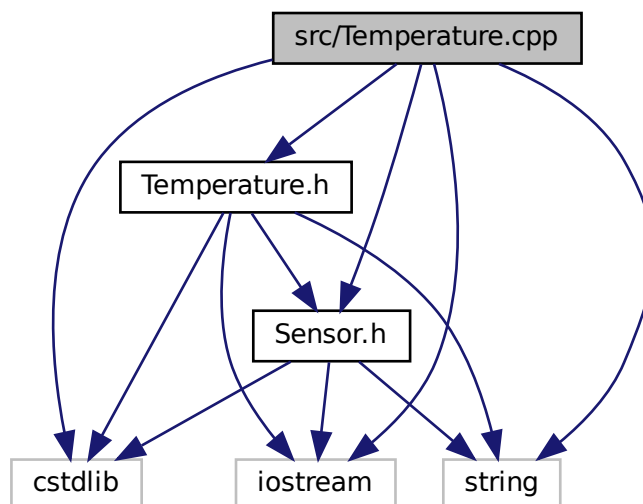
```
int back_1
```

Definition at line 18 of file Sensor.cpp.

5.24 src/Temperature.cpp File Reference

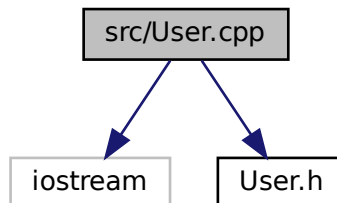
```
#include <iostream>
#include <string>
#include <cstdlib>
#include "Sensor.h"
#include "Temperature.h"
```

Include dependency graph for Temperature.cpp:



5.25 src/User.cpp File Reference

```
#include <iostream>
#include "User.h"
Include dependency graph for User.cpp:
```



Variables

- int [ID](#)
- int [NIF](#)
- int [ADMIN](#)

5.25.1 Variable Documentation

5.25.1.1 ADMIN

```
int ADMIN
```

Definition at line 16 of file `User.cpp`.

5.25.1.2 ID

```
int ID
```

Definition at line 14 of file `User.cpp`.

5.25.1.3 NIF

```
int NIF
```

Definition at line 15 of file `User.cpp`.

Index

- actualID
 - DataBase, [34](#)
- actualUserException, [7](#)
 - actualUserException, [8](#)
- addUser
 - DataBase, [30](#)
- addUserDashboard
 - Dashboard, [19](#)
- ADMIN
 - Admin, [11](#)
 - Employer, [38](#)
 - User, [69](#)
 - User.cpp, [100](#)
- Admin, [9](#)
 - ADMIN, [11](#)
 - Admin, [10](#)
- admin
 - DataBase, [34](#)
 - DataBase.cpp, [92](#)
- airq
 - Dashboard.cpp, [90](#)
- AirQuality, [11](#)
 - AirQuality, [12](#)
 - getData, [13](#)
- alarm
 - Security, [50](#)
- alarm_status
 - Security.cpp, [97](#)
- alarma_status
 - Security, [54](#)
- back
 - Security.cpp, [97](#)
- back_1
 - Sensor, [58](#)
 - Sensor.cpp, [99](#)
- brigh
 - Dashboard.cpp, [90](#)
- Brightness, [13](#)
 - Brightness, [15](#)
 - getData, [15](#)
- cabecera
 - Dashboard, [20](#)
- cabeceraSensor
 - Sensor, [57](#)
- camera
 - Security, [51](#)
- comprobar_user_data
 - Login, [44](#)
- contador
 - Login, [47](#)
 - Login.cpp, [94](#)
- credentialException, [16](#)
 - credentialException, [17](#)
- d
 - main.h, [83](#)
- Dashboard, [18](#)
 - addUserDashboard, [19](#)
 - cabecera, [20](#)
 - Dashboard, [19](#)
 - dashboard, [21](#)
 - database, [27](#)
 - eraseUserDashboard, [22](#)
 - goBack, [23](#)
 - goOut, [23](#)
 - goToSecurity, [23](#)
 - goToSensors, [24](#)
 - opcion, [27](#)
 - pantalla, [28](#)
 - set_dashboardAdmin, [26](#)
 - set_dashboardEmployer, [27](#)
- dashboard
 - Dashboard, [21](#)
- Dashboard.cpp
 - airq, [90](#)
 - brigh, [90](#)
 - database, [90](#)
 - hum, [90](#)
 - opcion, [90](#)
 - s, [91](#)
 - sen, [91](#)
 - temp, [91](#)
- data_valid
 - Login, [48](#)
 - Login.cpp, [94](#)
- DataBase, [28](#)
 - actualID, [34](#)
 - addUser, [30](#)
 - admin, [34](#)
 - DataBase, [29](#)
 - eraseUser, [30](#)
 - isAdmin, [35](#)
 - nuevoID, [35](#)
 - nuevoNIF, [35](#)
 - readUsers, [31](#)
 - userList, [32](#)
 - valided, [35](#)
 - validUser, [32](#)

- vectorUser, 35
- writeUsers, 33
- database
 - Dashboard, 27
 - Dashboard.cpp, 90
 - Login, 48
- DataBase.cpp
 - admin, 92
 - nuevoID, 92
 - nuevoNIF, 92
 - valided, 92
- db
 - Security.cpp, 97
- door
 - Security, 51
- Employer, 36
 - ADMIN, 38
 - Employer, 37
- entrada
 - Security.cpp, 98
- eraseUser
 - DataBase, 30
- eraseUserDashboard
 - Dashboard, 22
- generate_random
 - Security, 52
- get_ID
 - Login, 45
- get_NIF
 - Login, 45
- getData
 - AirQuality, 13
 - Brightness, 15
 - Humidity, 40
 - Sensor, 57
 - Temperature, 62
- getNumRecord
 - User, 66
- getUserADMIN
 - User, 67
- getUserID
 - User, 67
- getUserNIF
 - User, 67
- go_back
 - Login, 46
- go_to_dashboard
 - Login, 46
- goBack
 - Dashboard, 23
- goBackSecurity
 - Security, 52
- goBackSensor
 - Sensor, 58
- goOut
 - Dashboard, 23
- goToSecurity
 - Dashboard, 23
- goToSensors
 - Dashboard, 24
- hum
 - Dashboard.cpp, 90
- Humidity, 38
 - getData, 40
 - Humidity, 39
- ID
 - User, 69
 - User.cpp, 100
- include/Admin.h, 71
- include/AirQuality.h, 72
- include/Brightness.h, 73
- include/Dashboard.h, 75
- include/DataBase.h, 76
- include/Employer.h, 77
- include/Exceptions.h, 78
- include/Humidity.h, 79
- include/Login.h, 81
- include/main.h, 82
- include/Security.h, 84
- include/Sensor.h, 85
- include/Temperature.h, 86
- include/User.h, 87
- instructionException, 41
 - instructionException, 42
- intentos
 - Login, 48
 - Login.cpp, 94
- isAdmin
 - DataBase, 35
- I
 - main.h, 83
- Login, 42
 - comprobar_user_data, 44
 - contador, 47
 - data_valid, 48
 - database, 48
 - get_ID, 45
 - get_NIF, 45
 - go_back, 46
 - go_to_dashboard, 46
 - intentos, 48
 - Login, 44
 - login, 46
 - set_login, 47
 - temp_ID, 48
 - temp_NIF, 48
- login
 - Login, 46
- Login.cpp
 - contador, 94
 - data_valid, 94
 - intentos, 94
 - temp_ID, 94

- temp_NIF, 95
- main
 - main.cpp, 95
- main.cpp
 - main, 95
- main.h
 - d, 83
 - l, 83
- NIF
 - User, 70
 - User.cpp, 100
- nuevoID
 - DataBase, 35
 - DataBase.cpp, 92
- nuevoNIF
 - DataBase, 35
 - DataBase.cpp, 92
- numRecord
 - User, 70
- opcion
 - Dashboard, 27
 - Dashboard.cpp, 90
- open
 - Security, 55
 - Security.cpp, 98
- open_close
 - Security, 53
- operator<
 - User, 67
- operator==
 - User, 68
- pantalla
 - Dashboard, 28
- randomData
 - Sensor, 58
- readUsers
 - DataBase, 31
- s
 - Dashboard.cpp, 91
- Security, 49
 - alarm, 50
 - alarma_status, 54
 - camera, 51
 - door, 51
 - generate_random, 52
 - goBackSecurity, 52
 - open, 55
 - open_close, 53
 - Security, 50
 - security, 53
 - seguridad, 55
 - window, 54
- security
 - Security, 53
- Security.cpp
 - alarm_status, 97
 - back, 97
 - db, 97
 - entrada, 98
 - open, 98
 - seguridad, 98
- seguridad
 - Security, 55
 - Security.cpp, 98
- sen
 - Dashboard.cpp, 91
- Sensor, 55
 - back_1, 58
 - cabeceraSensor, 57
 - getData, 57
 - goBackSensor, 58
 - randomData, 58
 - Sensor, 56
- Sensor.cpp
 - back_1, 99
- set_dashboardAdmin
 - Dashboard, 26
- set_dashboardEmployer
 - Dashboard, 27
- set_login
 - Login, 47
- setNumRecord
 - User, 68
- setUserID
 - User, 68
- setUserNIF
 - User, 69
- setUserRole
 - User, 69
- src/AirQuality.cpp, 88
- src/Brightness.cpp, 89
- src/Dashboard.cpp, 89
- src/DataBase.cpp, 91
- src/Humidity.cpp, 93
- src/Login.cpp, 93
- src/main.cpp, 95
- src/Security.cpp, 96
- src/Sensor.cpp, 98
- src/Temperature.cpp, 99
- src/User.cpp, 100
- stringException, 59
 - stringException, 60
- temp
 - Dashboard.cpp, 91
- temp_ID
 - Login, 48
 - Login.cpp, 94
- temp_NIF
 - Login, 48
 - Login.cpp, 95
- Temperature, 61
 - getData, 62

- Temperature, [62](#)
- User, [63](#)
 - ADMIN, [69](#)
 - getNumRecord, [66](#)
 - getUserADMIN, [67](#)
 - getUserID, [67](#)
 - getUserNIF, [67](#)
 - ID, [69](#)
 - NIF, [70](#)
 - numRecord, [70](#)
 - operator<, [67](#)
 - operator==, [68](#)
 - setNumRecord, [68](#)
 - setUserID, [68](#)
 - setUserNIF, [69](#)
 - setUserRole, [69](#)
 - User, [66](#)
- User.cpp
 - ADMIN, [100](#)
 - ID, [100](#)
 - NIF, [100](#)
- userList
 - DataBase, [32](#)
- validated
 - DataBase, [35](#)
 - DataBase.cpp, [92](#)
- validUser
 - DataBase, [32](#)
- vectorUser
 - DataBase, [35](#)
- window
 - Security, [54](#)
- writeUsers
 - DataBase, [33](#)