Julio Veganos e Hijos interfaz

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

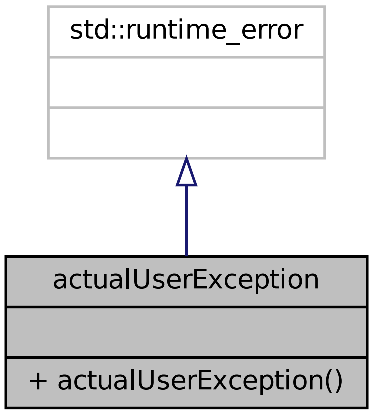
Chapter 4

Class Documentation

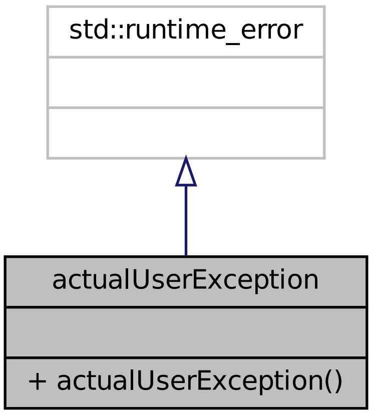
4.1 actualUserException Class Reference

```
#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.

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

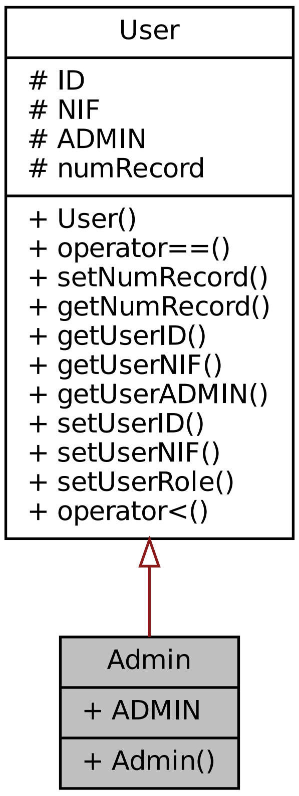
The documentation for this class was generated from the following file:

- include/Exceptions.h

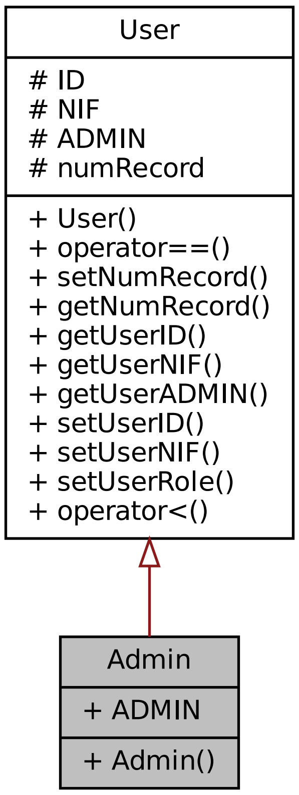
4.2 Admin Class Reference

```
#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.

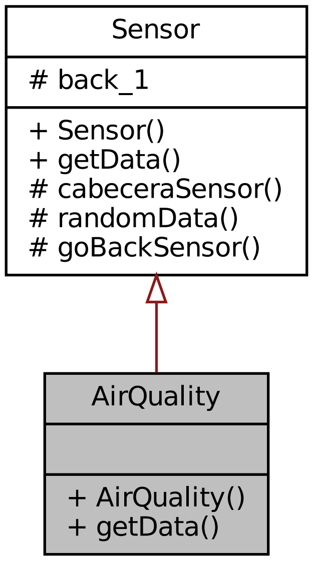
The documentation for this class was generated from the following file:

- include/Admin.h

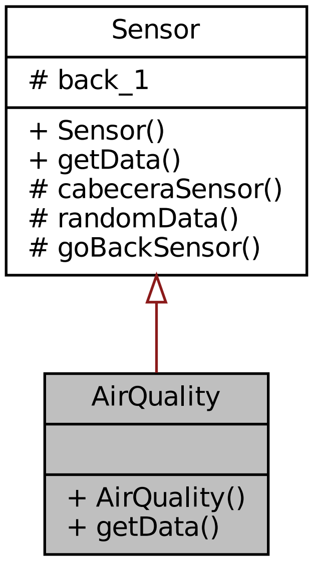
4.3 AirQuality Class Reference

```
#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()

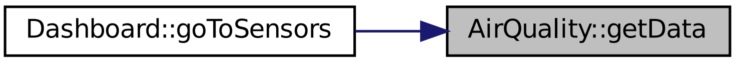
```
void AirQuality::getData ( )
```

Definition at line 11 of file AirQuality.cpp.

```
                        {
    while (true) {
        int random = this->randomData();
        cout « "\n\t\tAir quality:\t\t" « (random + 30) « " ppm\n" « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cin » this->back_1;
        switch (this->back_1) {

        case 0:
            return;
            break;

        default:

            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            this->cabeceraSensor();
            break;
        }
    }
}
```

References back_1.

Here is the caller graph for this function:



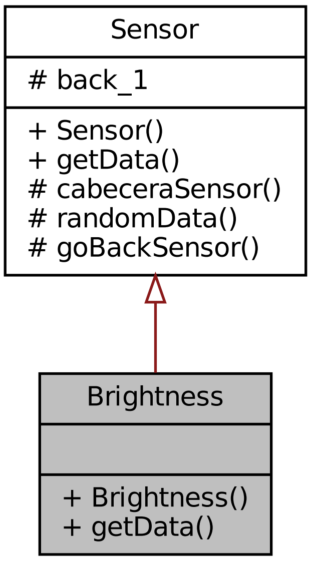
The documentation for this class was generated from the following files:

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

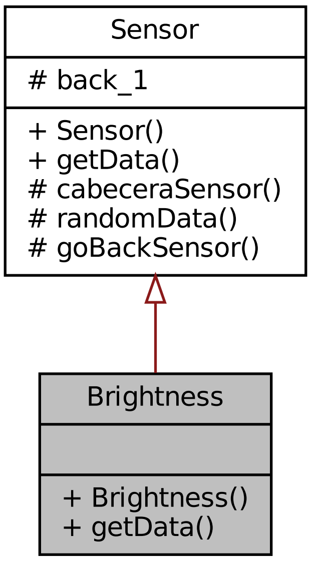
4.4 Brightness Class Reference

```
#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.

```
{}
```

4.4.3 Member Function Documentation

4.4.3.1 getData()

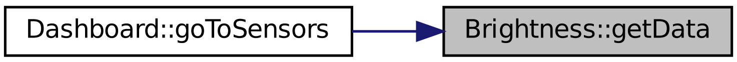
```
void Brightness::getData ( )
```

Definition at line 11 of file Brightness.cpp.

```
                           {
    while (true) {
        int random = this->randomData();
        cout « "\n\t\tBrightness:\t\t" « (random + 10) « " lmen/m²\n" « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cin » this->back_1;
        switch (this->back_1) {

        case 0:
            return;
            break;

        default:

            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            this->cabeceraSensor();
            break;
        }
    }
}
```

References back_1.

Here is the caller graph for this function:



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

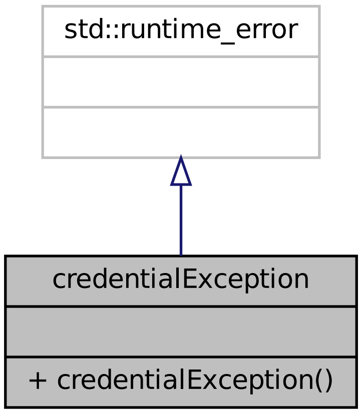
- include/Brightness.h
- src/Brightness.cpp

4.5 credentialException Class Reference

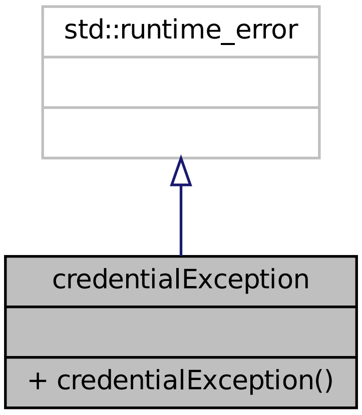
If the ID or NIF are incorrect, the exception will be thrown.

```
#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") {}
```

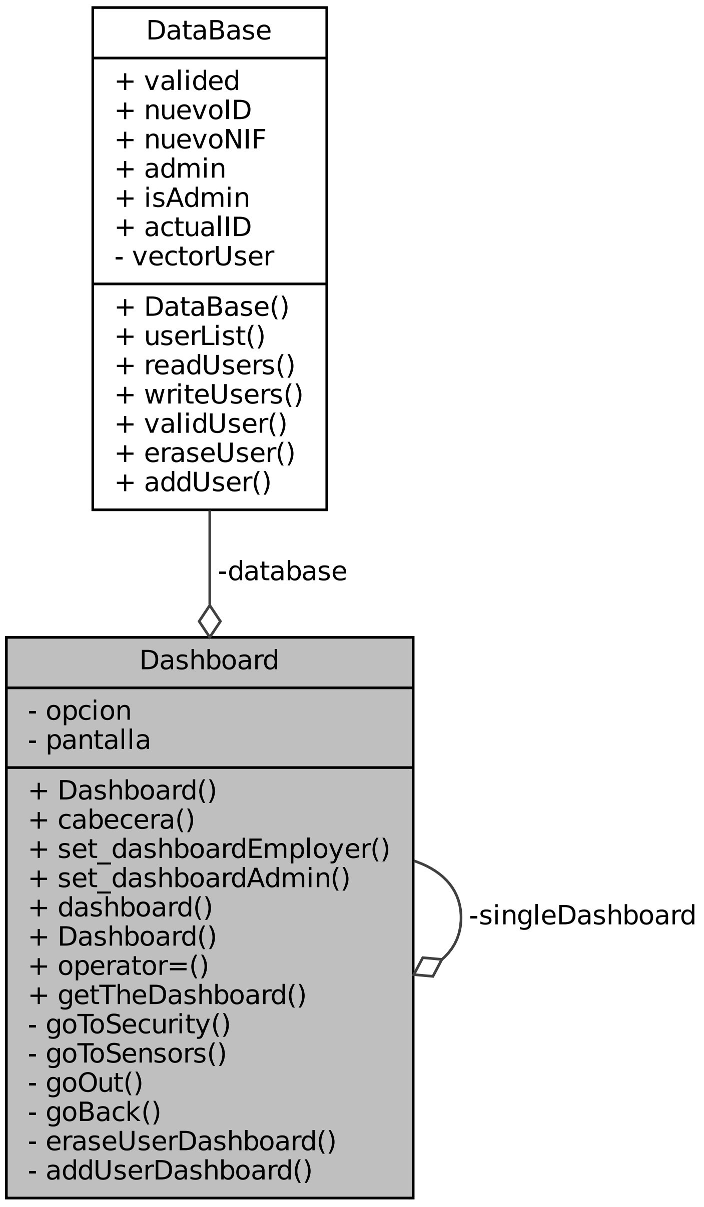
The documentation for this class was generated from the following file:

- 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)
- Dashboard (Dashboard &otherDashboard)=delete
- void operator= (const Dashboard &)=delete

Static Public Member Functions

- static Dashboard ∗ getTheDashboard ()

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

Static Private Attributes

- static Dashboard ∗ singleDashboard = nullptr

4.6.1 Detailed Description

Definition at line 19 of file Dashboard.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Dashboard() [1/2]

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

Definition at line 27 of file Dashboard.cpp.

```
{}
```

4.6.2.2 Dashboard() [2/2]

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

4.6.3 Member Function Documentation

4.6.3.1 addUserDashboard()

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

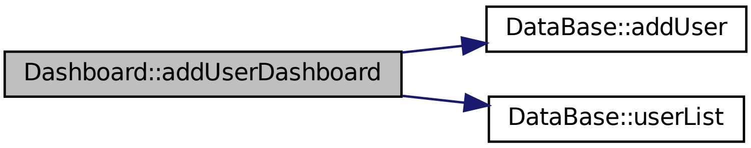
Function that allows the admin to add new users.

Definition at line 263 of file Dashboard.cpp.

```
                                     {
    while (true) {
        cabecera();
        database->userList();
        cout «
       "\n-----------------------------------------------------------------------------------------------------\n"
       « endl;
        cout « "Si desea añadir un usuario introduzca 1, para salir introduzca 0" « endl;
        cout « "»> ";
        cin » opcion;
        try {
            switch (opcion)
            {
            case 1:
                cout « "Introduzca ID del nuevo usuario" « endl;
                cout « "»> ";
                cin » database->nuevoID;
                //cout « database->nuevoID « endl;
                cout « "Introduzca NIF del nuevo usuario" « endl;
                cout « "»> ";
                cin » database->nuevoNIF;
                //cout « database->nuevoNIF « endl;
                cout « "¿El usuario es adiministrador? (introduzaca 0 para NO, 1 para SI" « endl;
                cout « "»> ";
                cin » database->admin;
                database->addUser();
                break;
            case 0:
                return;
            default:
                throw instructionException();
            }
        } catch (instructionException &except) {
            cout « "Exception: " « except.what() « endl;
            system("sleep 2");
        }

    }
}
```

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()

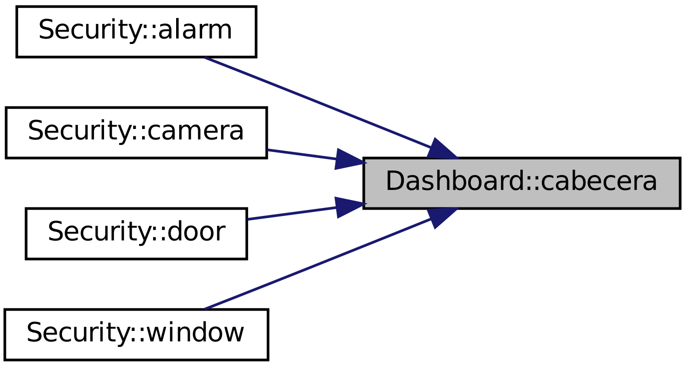
```
void Dashboard::cabecera ( )
```

Principal header with the name of the business.

Definition at line 41 of file Dashboard.cpp.

```
                          {
    system("clear");
    cout « "\x1b[32m
       \n"
"          / /                           ||   / /                                                        \n"
"         / /          //  ( )  ___      ||  / /  ___      ___      ___        __      ___      ___      \n"
"        / / //   / / // / / //    ) )   || / / //___) ) //   ) ) //   ) ) //   ) )  //   ) ) ((   ) )   \n"
"       / / //   / / // / / //   / /     ||/ / //       ((___/ / //   / / //    / / //   / /   \\ \\     \n"
" ((___/ / ((___( ( // / / ((___/ /      |  / ((____     //__   ((___( ( //   / /  ((___/ / //   ) )     \n"
"                                                                                                        \n"
"                         //     / /                                                                     \n"
"            ___         //___  / / ( )     ( )  ___      ___                                            \n"
"          //___) )     / ___   / / /      / / //   ) ) ((  ) )                                          \n"
"         //           //    / / / /      / / //   / /   \\ \\                                           \n"
"        ((____       //    / / / /  ((  / / ((___/ / //   ) )    \x1b[0m                               \n\n"
"------------------------------------------------------------------------------------------------------- \n"
    « endl;

};
```

Here is the caller graph for this function:



4.6.3.3 dashboard()

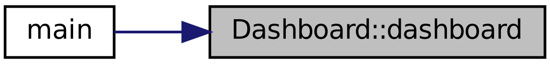
```
void Dashboard::dashboard (
            DataBase * base,
            bool isAdmin )
```

Definition at line 301 of file Dashboard.cpp.

```
                                                        {
    database = base;
    if (isAdmin) {
        set_dashboardAdmin();
    } else {
        set_dashboardEmployer();
    }
}
```

References database.

Here is the caller graph for this function:



4.6.3.4 eraseUserDashboard()

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

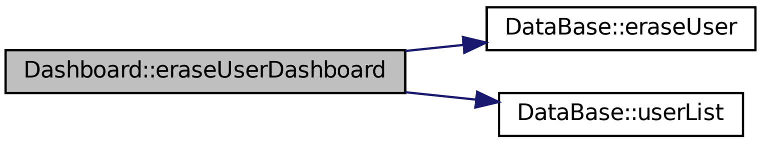
Method that allow the admin to erase the users.

Definition at line 234 of file Dashboard.cpp.

```
                                          {
    while (true) {
        int opcion;
        cabecera();
        database->userList();
        cout «
       "\n---------------------------------------------------------------------------------------------------\n"
       « endl;
        cout « "\nIntroduzca el ID del usuario que desa borrar o introduzca 0 para volver" « endl;
        cout « "\n\t\tUsuario actual: " « database->actualID « endl;
        cout « "»> ";
        cin » opcion;
        try {
            switch (opcion) {
            case 0:
                return;
            default:
                if (database->actualID == opcion) {
                    throw actualUserException();
                    break;
                } else {
                    database->eraseUser(opcion);
                }
            }
        } catch  (actualUserException &except) {
            cout « "Exception: " « except.what() « endl;
            system("sleep 2");
        }
    }
}
```

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

Here is the call graph for this function:



4.6.3.5 getTheDashboard()

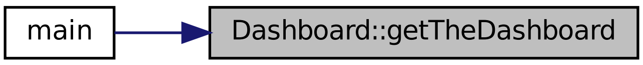
```
Dashboard * Dashboard::getTheDashboard ( )  [static]
```

Definition at line 312 of file Dashboard.cpp.

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

    return singleDashboard;
```

```
}
```

Here is the caller graph for this function:



4.6.3.6 goBack()

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

4.6.3.7 goOut()

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

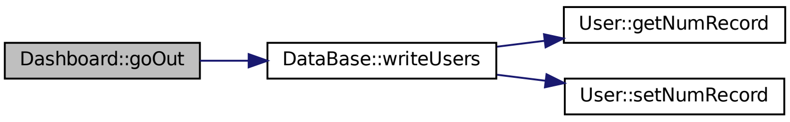
Function to exit the program when the option is selected.

Definition at line 227 of file Dashboard.cpp.

```
                        {
    system("clear");
    cout « "\x1b[41mSaliendo del sistema...\x1b[0m" « endl;
    database->writeUsers();
    exit(0);
}
```

References database, and DataBase::writeUsers().

Here is the call graph for this function:



4.6.3.8 goToSecurity()

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

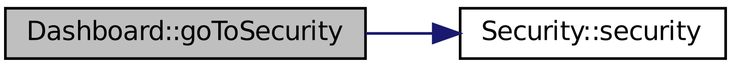
Menu with options for the security.

Definition at line 187 of file Dashboard.cpp.

```
                             {
    while (true) {
        // Heading and options
        cabecera();
        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;
        cout «
       "\n----------------------------------------------------------------------------------------------------\n"
       « endl;
        cout « "Seleccione una opcion escribiendo el número correspondiente.\n" « endl;
        cout « "»> ";
        cin » opcion;
        try {
            switch (opcion) {
            case 0:
                return;
            case 1:
                cabecera();
                s.security("puertas");
                break;
            case 2:
                cabecera();
                s.security("ventanas");
                break;
            case 3:
                cabecera();
                s.security("camaras");
                break;
            case 4:
                cabecera();
                s.security("alarma");
                break;
            default:
                throw instructionException();
            }
        } catch (instructionException &except) {
            cout « "Exception: " « except.what() « endl;
            system("sleep 2");
        }
    }
};
```

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

Here is the call graph for this function:



4.6.3.9 goToSensors()

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

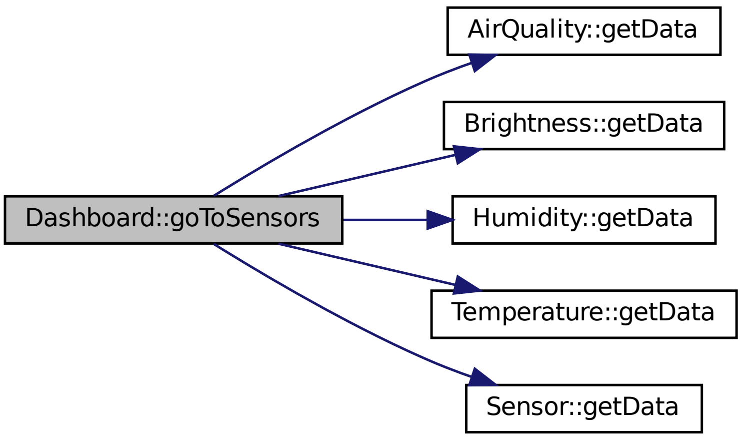
menu that shows the different sensors. For each instruction a sensor is shown

Definition at line 133 of file Dashboard.cpp.

```
                              {
    while (true) {
        // Heading and options
        cabecera();
        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\t==> 0" « endl;
        cout «
       "\n-----------------------------------------------------------------------------------------------------\n"
       « endl;
        cout « "Seleccione una opcion escribiendo el número correspondiente.\n" « endl;
        cout « "»> ";
        cin » opcion;
        try {
            switch (opcion) {
            case 0:
            return;

            case 1:
                cabecera();
                hum.getData();
                break;

            case 2:
                cabecera();
                brigh.getData();
                break;

            case 3:
                cabecera();
                airq.getData();
                break;

            case 4:
                cabecera();
                temp.getData();
                break;

            case 5:
                cabecera();
                sen.getData(opcion);
                break;

            case 6:
                cabecera();
                sen.getData(opcion);
                break;
            default:
                throw instructionException();
            }
        } catch (instructionException &except) {
            cout « "Exception: " « except.what() « endl;
            system("sleep 2");
        }
    }
};
```

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:



4.6.3.10 operator=()

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

4.6.3.11 set_dashboardAdmin()

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

Set the menu for the Admins.

Definition at line 93 of file Dashboard.cpp.

```
                                  {

      while (true) {
          cabecera();
          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;

          cout «
      "-----------------------------------------------------------------------------------------\n"
      « endl;
           cout « "Seleccione una opcion escribiendo el número correspondiente.\n" « endl;
           cout « "»> ";
           cin » opcion;
           try {
           switch (opcion) {
           case 0:
               goOut();
               break;
           case 1:
               goToSensors();
               break;
           case 2:
               goToSecurity();
               break;
           case 3:
               addUserDashboard();
               break;
           case 4:
               eraseUserDashboard();
               break;
           case 5:
               return;
           default:
                   throw instructionException();
               }
           } catch (instructionException &except) {
               cout « "Exception: " « except.what() « endl;
               system("sleep 2");
           }
       }
   };
```

References opcion.

4.6.3.12 set_dashboardEmployer()

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

Set the menu for the employers.

Definition at line 61 of file Dashboard.cpp.

```
                                      {
    while (true) {
        cabecera();
        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;

        cout «
"----------------------------------------------------------------------------------------------------\n"
« endl;
        cout « "Seleccione una opcion escribiendo el número correspondiente.\n" « endl;
        cout « "»> ";
        cin » opcion;
        try {
        switch (opcion) {
        case 0:
            goOut();
            break;
        case 1:
            goToSensors();
            break;
        case 2:
            goToSecurity();
            break;
        case 3:
            return;
        default:
                throw instructionException();
            }
        } catch (instructionException &except) {
            cout « "Exception: " « except.what() « endl;
            system("sleep 2");
        }
    }
};
```

References opcion.

4.6.4 Member Data Documentation

4.6.4.1 database

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

Definition at line 82 of file Dashboard.h.

4.6.4.2 opcion

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

Definition at line 80 of file Dashboard.h.

4.6.4.3 pantalla

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

Definition at line 81 of file Dashboard.h.

4.6.4.4 singleDashboard

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

Definition at line 84 of file Dashboard.h.

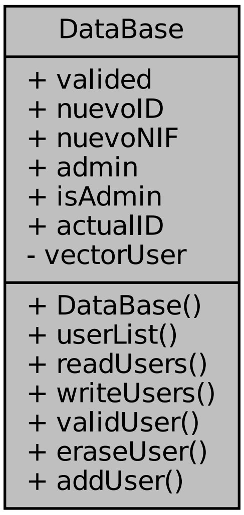
The documentation for this class was generated from the following files:

- 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.

```
                     {
    /*User* user1 = new User(1111, 12345678, 0);
    User* user2 = new User(2222, 23456789, 0);
    User* user3 = new User(3333, 34567890, 0);
    User* user4 = new User(1212, 12121212, 1);
    vectorUser.insert(*user1);
    vectorUser.insert(*user2);
    vectorUser.insert(*user3);
    vectorUser.insert(*user4);*/

}
```

4.7.3 Member Function Documentation

4.7.3.1 addUser()

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

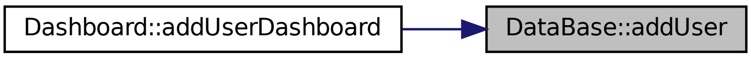
Method to add new users.

Definition at line 73 of file DataBase.cpp.

```
                          {

    for (User u : vectorUser) {

        if (u.getUserID() == nuevoID || u.getUserNIF() == nuevoNIF) {

            cout « "ID o NIF ya existentes" « endl;
            break;
        } else {
            vectorUser.insert(User(nuevoNIF, nuevoID, admin));
        }
    }
}
```

References admin, nuevoID, and nuevoNIF.

Here is the caller graph for this function:



4.7.3.2 eraseUser()

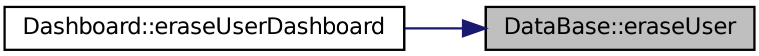
```
void DataBase::eraseUser (
              int posicion )
```

method to erase users

Definition at line 63 of file DataBase.cpp.

```
                                        {

    for (User u : vectorUser) {
        if (posicion == u.getUserID() ) {
            vectorUser.erase(u);
            break;
        }
    }
}
```

Here is the caller graph for this function:



4.7.3.3 readUsers()

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

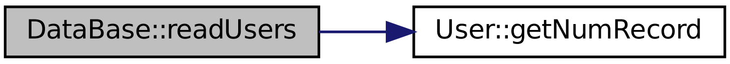
Method to read the current users from the file users.dat.

Definition at line 88 of file DataBase.cpp.

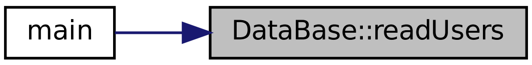
```
                            {
    ifstream inUsersFile ("./resources/users.dat", ios::in | ios::binary);


    if (!inUsersFile) { // file couldn’t be opened
        cerr « "File could not be opened" « endl;
        exit (1);
    }

    User user;

    inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));

     while (inUsersFile && !inUsersFile.eof()) {
         if (user.getNumRecord() != 0) {
             vectorUser.insert (user);
         }
         inUsersFile.read (reinterpret_cast <char *>(&user), sizeof (User));
     }

}
```

References User::getNumRecord().

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.3.4 userList()

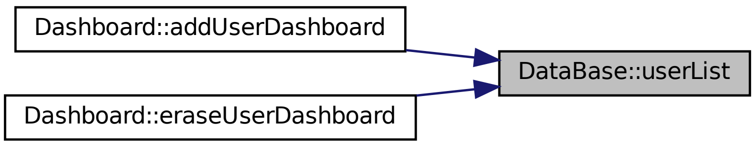
```
void DataBase::userList ( )
```

list of the current usersd

Definition at line 136 of file DataBase.cpp.

```
                           {
    int posicion = 1;
    for (User u : vectorUser) {
        cout « "\t" « posicion « " -\t" « u.getUserID() « "\t" « u.getUserNIF() « endl;
        posicion += 1;
    }
}
```

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

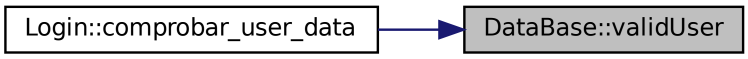
false

Definition at line 39 of file DataBase.cpp.

```
                                           {

    /*  For each user in the list the atributes ID and NIF are compared whit
        the ones introducted by the user from the login*/
    for (User u : vectorUser) {

        if (u.getUserID() == id && u.getUserNIF() == nif) {
            // when the correct user is found the loop is over
            valided = true;
            if (u.getUserADMIN() == 1) {
                isAdmin = true;
            } else {
                isAdmin = false;
            }


            break;
        } else {
            valided = false;
        }
    }
    return valided;
}
```

References valided.

Here is the caller graph for this function:



4.7.3.6 writeUsers()

`void DataBase::writeUsers ( )`

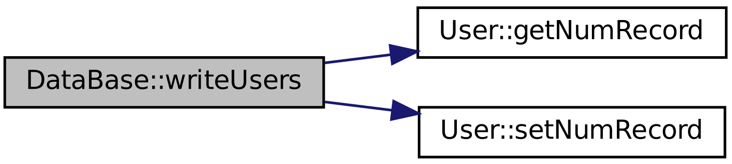
Methot used for write in the file users.dat the users at the end of the program.

Definition at line 111 of file DataBase.cpp.

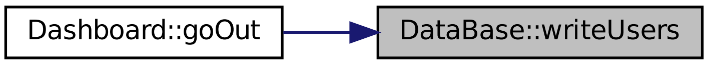
```
                           {
    int numRecord = 1;

    fstream outUsersFile ("./resources/users.dat", ios::out | ios::binary | ios::trunc);


    if (!outUsersFile) { // file couldn't be opened
        cerr « "File could not be opened" « endl;
        exit (1);
    }

    for (User u : vectorUser) {

        User user(u.getUserNIF(), u.getUserID(), u.getUserADMIN());
        user.setNumRecord (numRecord);

        outUsersFile.seekp ((user.getNumRecord() - 1 ) *
        sizeof (User));

        outUsersFile.write (reinterpret_cast <const char *> (&user),
        sizeof (User));
        numRecord++;
    }
}
```

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.

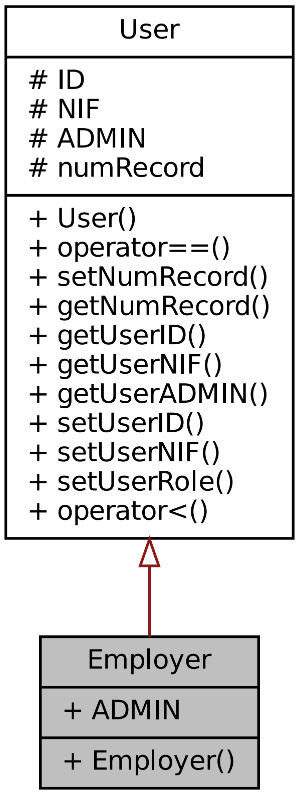
The documentation for this class was generated from the following files:

- include/DataBase.h
- src/DataBase.cpp

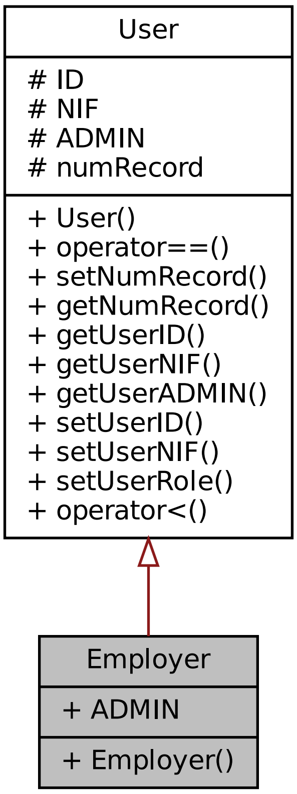
4.8 Employer Class Reference

```
#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.

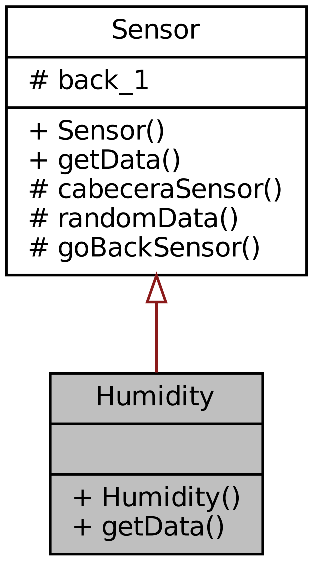
The documentation for this class was generated from the following file:

- include/Employer.h

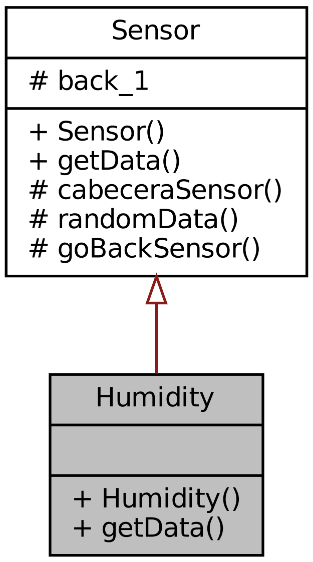
4.9 Humidity Class Reference

```
#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 ( )
```

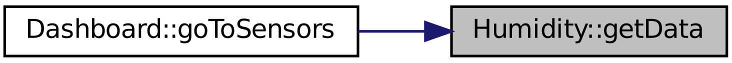
Get the Data object.

Definition at line 11 of file Humidity.cpp.

```
                          {
    while (true) {
        int random = this->randomData();
        cout « "\n\t\tHumidity:\t\t" « (random + 30) « " g/m³\n" « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cin » this->back_1;
        switch (this->back_1) {

        case 0:
            return;
            break;

        default:

            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            this->cabeceraSensor();
            break;
        }
    }
}
```

References back_1.

Here is the caller graph for this function:



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

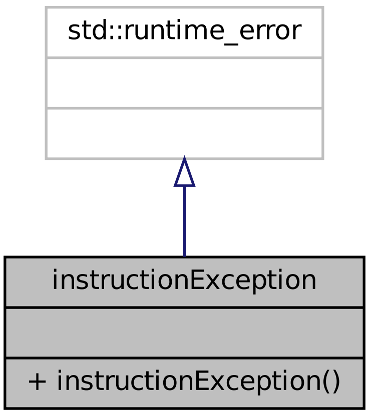
- include/Humidity.h
- src/Humidity.cpp

4.10 instructionException Class Reference

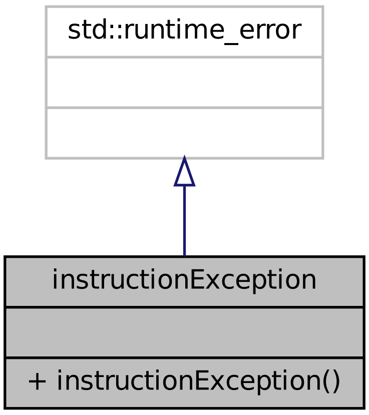
If the instruction of the menu is wrong, the exception appears.

```
#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") {}
```

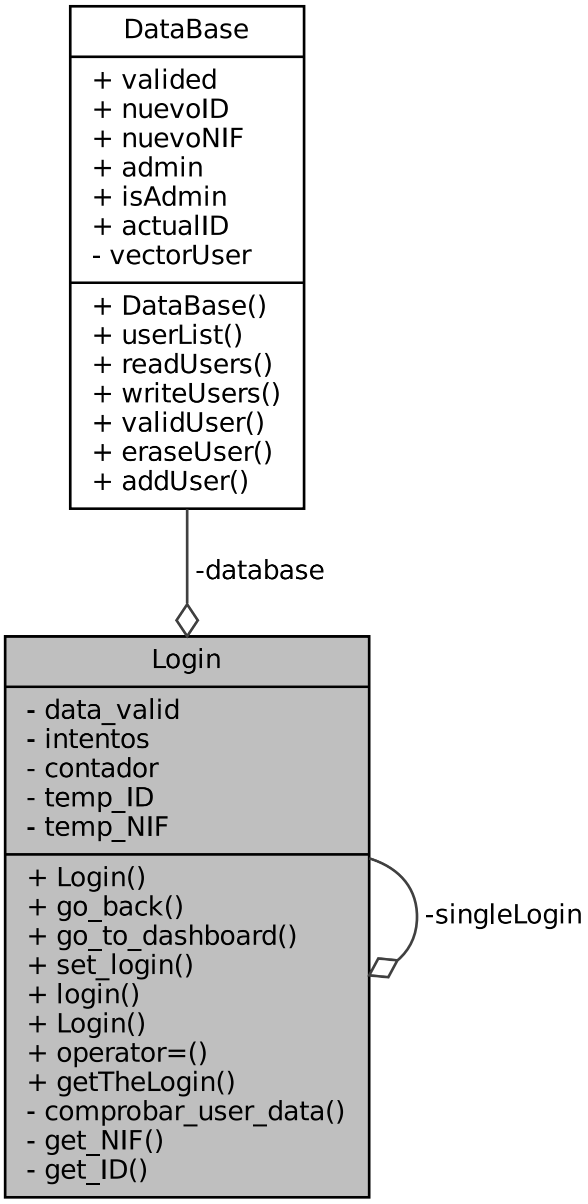
The documentation for this class was generated from the following file:

- 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 ∗)
- Login (Login &otherLogin)=delete
- void operator= (const Login &)=delete

Static Public Member Functions

- static Login ∗ getTheLogin ()

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

Static Private Attributes

- static Login ∗ singleLogin = nullptr

4.11.1 Detailed Description

Definition at line 17 of file Login.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Login() [1/2]

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

Definition at line 24 of file Login.cpp.

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

4.11.2.2 Login() [2/2]

```
Login::Login (
            Login & otherLogin )  [delete]
```

4.11.3 Member Function Documentation

4.11.3.1 comprobar_user_data()

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

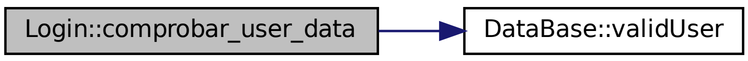
method that calls a database function which verify the credentials If the function return a false, a mesage is printed to show the user that is wrong At the third wrong try the program finishes with a 'access denied' error

Definition at line 78 of file Login.cpp.

```
                                   {

      try {
      data_valid = database->validUser(temp_ID, temp_NIF);
          if (!data_valid) {
              throw credentialException();
          }
      }
      catch (credentialException &except) {
          cout « "\n\n\tException: "« except.what() « endl;
          contador++;
          system("sleep 1.5");
          if (contador != 3) {
              system("clear");
          } else {
              // at the third try the program finish and denied the access
              cout « "\u001b[31m\n\t\tAccess denied.\u001b[0m\n" « endl;
              go_back();
          }
      }
 };
```

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.

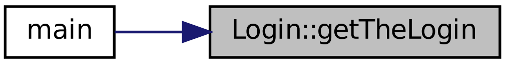
4.11.3.4 getTheLogin()

```
Login * Login::getTheLogin ( )  [static]
```

Definition at line 151 of file Login.cpp.

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

  return singleLogin;
}
```

Here is the caller graph for this function:



4.11.3.5 go_back()

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

Definition at line 109 of file Login.cpp.

```
                    {
    exit(0);
};
```

4.11.3.6 go_to_dashboard()

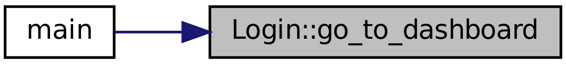
```
bool Login::go_to_dashboard ( )
```

Definition at line 100 of file Login.cpp.

```
                            {

    system("sleep 1");
    if (database->isAdmin) {
        return true;
    }
    return false;
};
```

References database, and DataBase::isAdmin.

Here is the caller graph for this function:



4.11.3.7 login()

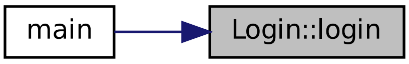
```
void Login::login (
            DataBase * data )
```

Definition at line 141 of file Login.cpp.

```
                                 {
    database = data;
    system("clear");
    set_login();
    go_to_dashboard();

};
```

References database.

Here is the caller graph for this function:



4.11.3.8 operator=()

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

4.11.3.9 set_login()

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

Definition at line 115 of file Login.cpp.

```
                             {
        while (true) {
            // Title
            cout « "\t\t\t----------------" « endl;
            cout « "\t\t\t\x1b[47m\x1b[30mLOGIN DE USUARIO\x1b[0m" « endl;
            cout « "\t\t\t----------------" « endl;

            // The credentials are asked and are validated
            get_ID();
            get_NIF();

            comprobar_user_data();


            // If the credentials are correct the user goes to the dashboard
            if (data_valid == true) {
                database->actualID = temp_ID;
                //cout « database->actualID « endl;
                cout « "\u001b[32m\n\n\t\t\tBienvenido al sistema\u001b[0m" « endl;
                contador = 0;
                return;
            }
        }
};
```

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 57 of file Login.h.

4.11.4.2 data_valid

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

Definition at line 55 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 53 of file Login.h.

4.11.4.4 intentos

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

Definition at line 56 of file Login.h.

4.11.4.5 singleLogin

```
Login * Login::singleLogin = nullptr  [static], [private]
```

Definition at line 61 of file Login.h.

4.11.4.6 temp_ID

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

Definition at line 58 of file Login.h.

4.11.4.7 temp_NIF

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

Definition at line 59 of file Login.h.

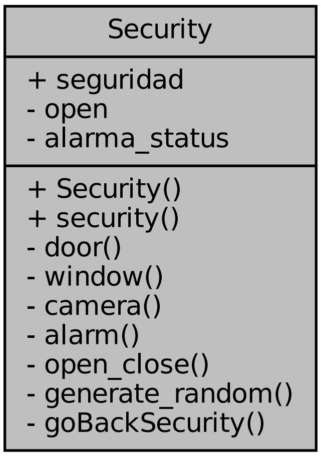
The documentation for this class was generated from the following files:

- 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.

```
{}
```

4.12.3 Member Function Documentation

4.12.3.1 alarm()

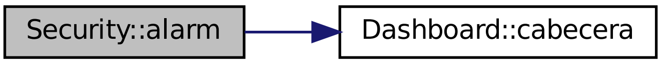
```
void Security::alarm ( ) [private]
```

Definition at line 118 of file Security.cpp.

```
                      {
    while (true) {
        cout « "\n\tAlarma: " « alarm_status « "\n\n" « endl;
        cout « "Para encender introduzca 1, para apagar introduzca 2, para salir 0\n" « endl;
        cout « "»> ";
        cin » entrada;
        switch (entrada) {

        case 0:
            return;

        case 1:
            alarm_status = "encendido";
            db.cabecera();
            break;

        case 2:
            alarm_status = "apagado";
            db.cabecera();
            break;
        default:
            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            db.cabecera();
            break;
        }
    }
}
```

References alarm_status, Dashboard::cabecera(), db, and entrada.

Here is the call graph for this function:



4.12.3.2 camera()

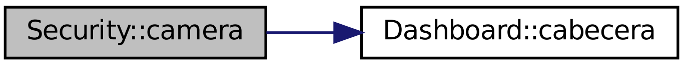
```
void Security::camera ( )  [private]
```

Definition at line 100 of file Security.cpp.

```
                          {
    while (true) {
        cout « "Imagina que hay una camara" « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cin » back;
        switch (back) {
        case 0:
            return;
            break;
        default:
            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            db.cabecera();
            break;
        }
    }
}
```

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

Here is the call graph for this function:



4.12.3.3 door()

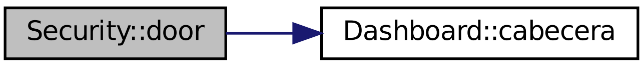
```
void Security::door ( ) [private]
```

Definition at line 42 of file Security.cpp.

```
                {
    while (true) {
        open_close();
        cout « "Puerta: 01\tLocalización: Huerto\tEstado: " « open « endl;
        open_close();
        cout « "Puerta: 02\tLocalización: Almacen\tEstado: " « open « endl;
        open_close();
        cout « "Puerta: 03\tLocalización: Despacho\tEstado: " « open « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cout « "»> ";
        cin » back;

        // If an order different of 0 is inserted, the door menu is restarted.
        switch (back) {

        case 0:
            return;
            break;

        default:

            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            db.cabecera();
            break;
        }
    }
}
```

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.

```
                                {
    srand(time(NULL));
    return rand()%10;
}
```

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.

```
                         {
    int random = generate_random();
    if (random >= 0 && random <= 4) {
        open = "close";
    } else {
        open = "open";
    }
}
```

References open.

4.12.3.7 security()

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

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

Parameters

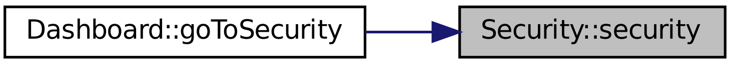
string	seguridad, depending on it's "value" the class shows a different menu

Definition at line 29 of file Security.cpp.

```
                                         {
    if (seguridad == "puertas") {
        door();
    } else if (seguridad == "ventanas") {
        window();
    } else if (seguridad == "camaras") {
        camera();
    } else if (seguridad == "alarma") {
        alarm();
    }
}
```

References seguridad.

Here is the caller graph for this function:



4.12.3.8 window()

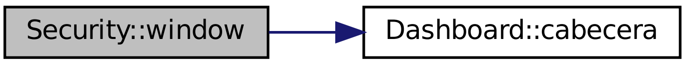
```
void Security::window ( )  [private]
```

Definition at line 72 of file Security.cpp.

```
                         {
    while (true) {
        open_close();
        cout « "Ventana: 01\tLocalización: Huerto\tEstado: " « open « endl;
        open_close();
        cout « "Ventana: 02\tLocalización: Almacen\tEstado: " « open « endl;
        open_close();
        cout « "Ventana: 03\tLocalización: Despacho\tEstado: " « open « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cin » back;
        // if an order different of 0 the programm restart
        switch (back) {

        case 0:
            return;
            break;

        default:

            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            db.cabecera();
            break;
        }
    }
}
```

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.

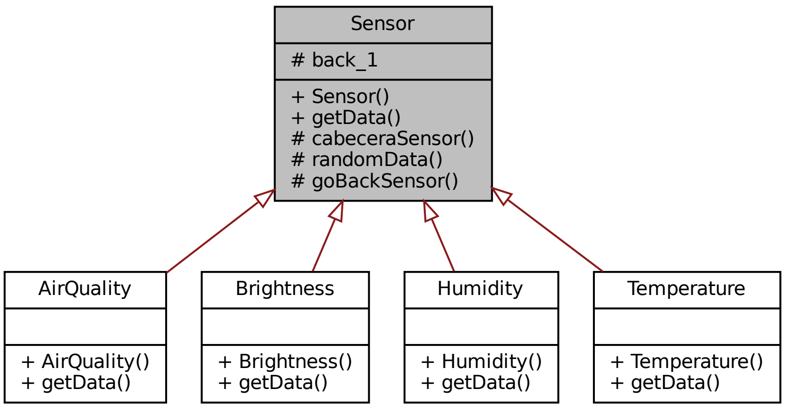
The documentation for this class was generated from the following files:

- include/Security.h
- src/Security.cpp

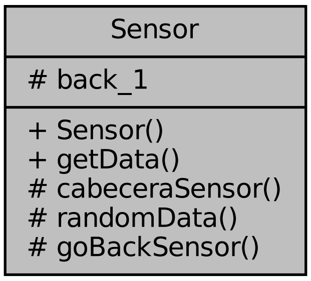
4.13 Sensor Class Reference

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

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



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 {}
```

4.13.3 Member Function Documentation

4.13.3.1 cabeceraSensor()

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

Heading used in all the sensors.

Definition at line 72 of file Sensor.cpp.

```
72                              {
73     system("clear");
74     cout « "\x1b[32m
          \n"
75 "          / /                           ||   / /                                                      \n"
76 "         / /          //  ( )  ___      ||  / /  ___      ___      ___       __      ___      ___     \n"
77 "        / / //   / / // / / //   ) )    || / / //___) ) //   ) ) //   ) ) //   ) ) //   ) ) ((   ) )  \n"
78 "       / / //   / / // / / //   / /     ||/ / //       ((___/ / //   / / //   / / //   / /   \\ \\    \n"
79 " ((___/ / ((___( ( // / / ((___/ /      | / ((____      //__   ((___( ( //   / / ((___/ / //   ) )    \n"
80 "                                                                                                      \n"
81 "                         //    / /                                                                    \n"
82 "            ___         //___ / / ( )     ( )  ___      ___                                           \n"
83 "          //___) )     / ___   / / /     / / //   ) )  ((   ) )                                       \n"

84 "         //           //    / / / /     / / //   / /    \\ \\                                         \n"

85 "        ((____       //    / / / / ((  / /  ((___/ / //   ) )    \x1b[0m                             \n\n"
86 "----------------------------------------------------------------------------------------------------- \n"
       « endl;
87
88 }
```

4.13.3.2 getData()

```
void Sensor::getData (
              int opcion )
```

Get the Data of the object.

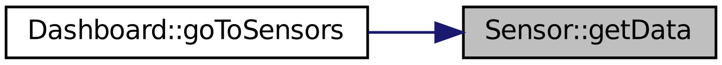
Definition at line 28 of file Sensor.cpp.

```
28                                  {
29
30     if (opcion == 5) {
31         while (true) {
32             cout « "\n\t\tRGB camera:\t\t*imagen*\n" « endl;
33             cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
34             cin » back_1;
35             switch (back_1) {
36
37             case 0:
38                 return;
39                 break;
40
41             default:
42
43                 cout « "Orden incorrecta" « endl;
44                 system("sleep 1");
45                 cabeceraSensor();
46                 break;
47             }
48         }
49     }  else if (opcion == 6) {
50         while (true) {
51             cout « "\n\t\tTermal camera:\t\t*imagen*\n" « endl;
52             cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
53             cin » back_1;
54             switch (back_1) {
55
```

```
            case 0:
                return;
                break;

            default:

                cout « "Orden incorrecta" « endl;
                system("sleep 1");
                cabeceraSensor();
                break;
            }
        }
    }
}
```

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.

```
                        {
    srand(time(NULL));
    return rand()%10;
}
```

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:

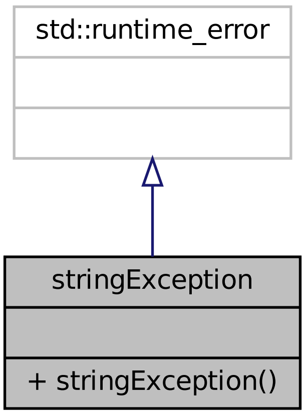
- include/Sensor.h
- src/Sensor.cpp

4.14 stringException Class Reference

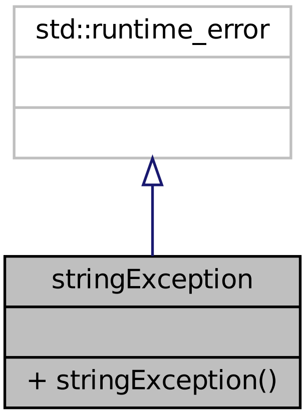
If a string is introduced when an int is needed an exception will be thrown.

```
#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") {}
```

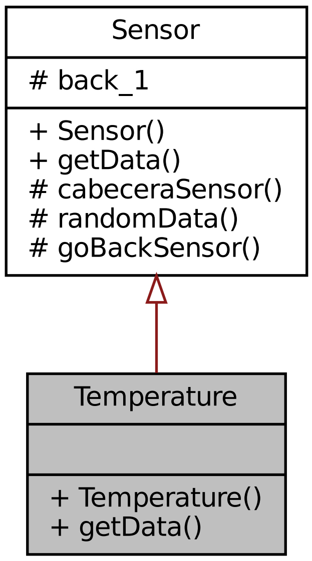
The documentation for this class was generated from the following file:

- include/Exceptions.h

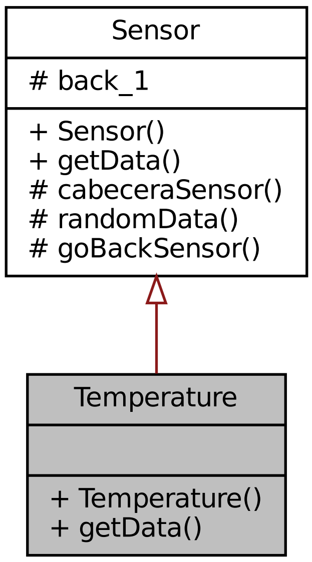
4.15 Temperature Class Reference

```
#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.

```
{}
```

4.15.3 Member Function Documentation

4.15.3.1 getData()

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

Get the object data.

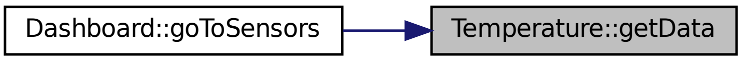
Definition at line 11 of file Temperature.cpp.

```
                            {
    while (true) {
        int random = this->randomData();
        cout « "\n\t\tTemperature:\t\t" « (random + 20) « " °C\n" « endl;
        cout « "\n\t\tPara volver atrás introduzca 0 + ENTER" « endl;
        cin » this->back_1;
        switch (this->back_1) {

        case 0:
            return;
            break;

        default:

            cout « "Orden incorrecta" « endl;
            system("sleep 1");
            this->cabeceraSensor();
```

```
                break;
            }
        }

}
```

References back_1.

Here is the caller graph for this function:



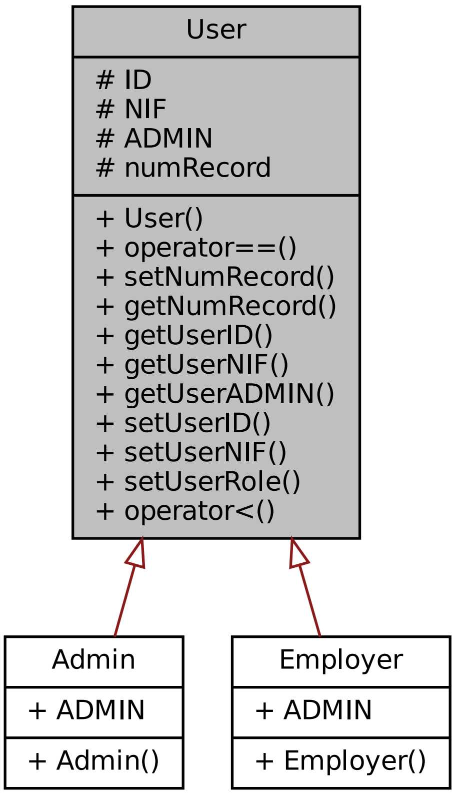
The documentation for this class was generated from the following files:

- include/Temperature.h
- src/Temperature.cpp

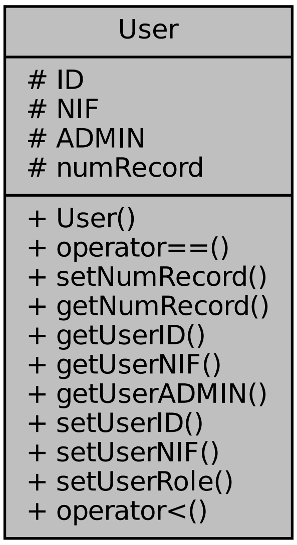
4.16 User Class Reference

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

Inheritance diagram for User:



Collaboration diagram for User:



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 atributes.
- 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

int	the first is the ID
int	the second is the NIF
int	the third is the role of the user

Definition at line 18 of file User.cpp.

```
                                     { // Constructor
      setUserID (id);
      setUserNIF (nif);
      setUserRole (admin);
  }
```

References admin.

4.16.3 Member Function Documentation

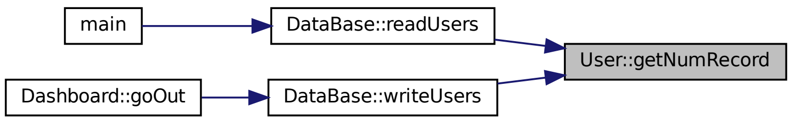
4.16.3.1 getNumRecord()

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

Definition at line 62 of file User.cpp.

```
                              {
          return numRecord;
  }
```

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.

```
                            {
    return ADMIN;
}
```

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.

```
                         {
    return ID;
}
```

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.

```
                          {
    return NIF;
}
```

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.

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

References ID, and ID.

4.16.3.6 operator==()

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

Definition at line 55 of file User.cpp.

```
                                           {
    if (ID == user.ID && NIF == user.NIF) {
        return true;
    }
    return false;
}
```

References ID, ID, NIF, and NIF.

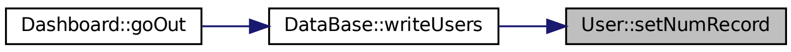
4.16.3.7 setNumRecord()

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

Definition at line 66 of file User.cpp.

```
                                  {
        numRecord = record;
}
```

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.

```
                              {
    NIF = myId;
}
```

References NIF.

4.16.3.9 setUserNIF()

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

Definition at line 42 of file User.cpp.

```
                                {
    ID = myNif;
}
```

References ID.

4.16.3.10 setUserRole()

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

Definition at line 38 of file User.cpp.

```
                                  {
    ADMIN = myRole;
}
```

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 atributes.

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

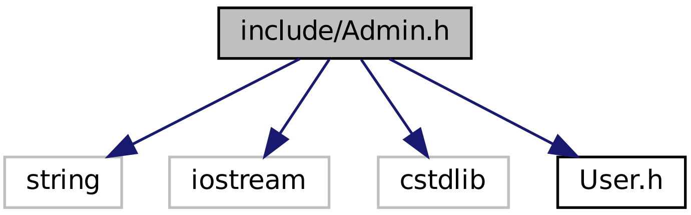
Chapter 5

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

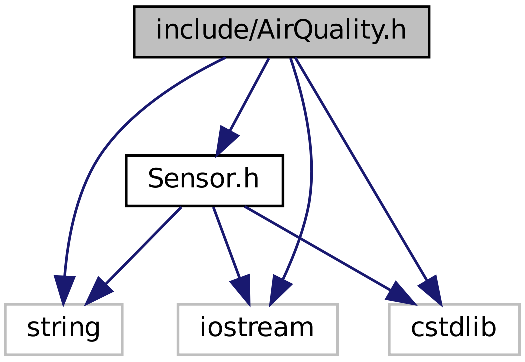
Copyright (c) 2022

5.2 include/AirQuality.h File Reference

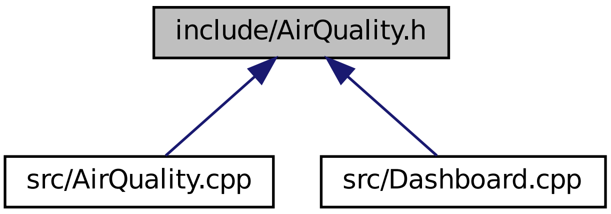
Air Quality sensor.

```
#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

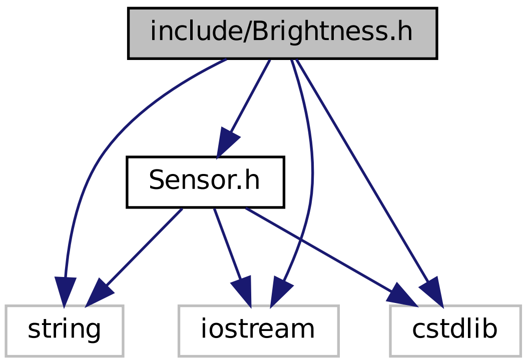
Copyright (c) 2022

5.3 include/Brightness.h File Reference

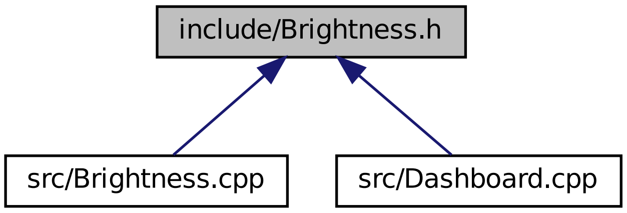
Luminosity sensor.

```
#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

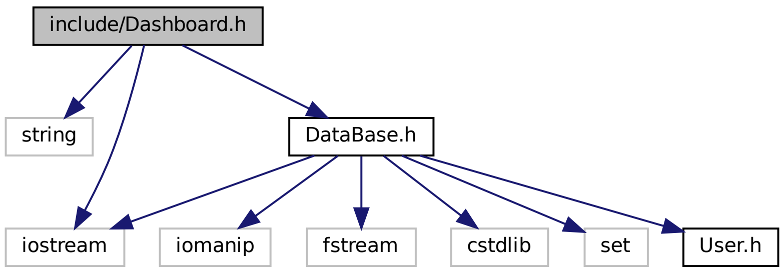
Copyright (c) 2022

5.4 include/Dashboard.h File Reference

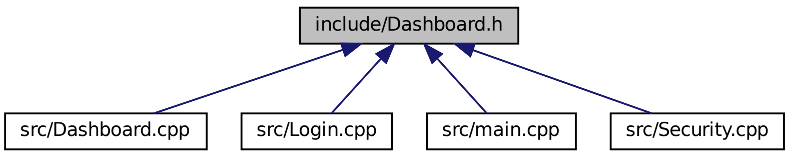
main menu

```
#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

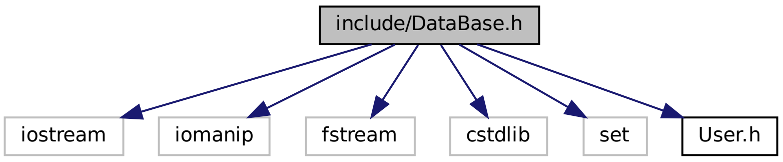
Copyright (c) 2022

5.5 include/DataBase.h File Reference

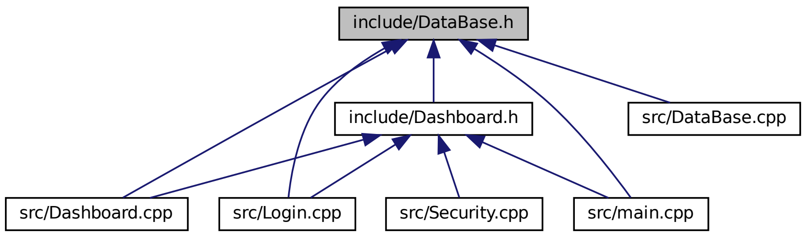
DataBase implementation and validate users's methods.

```
#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

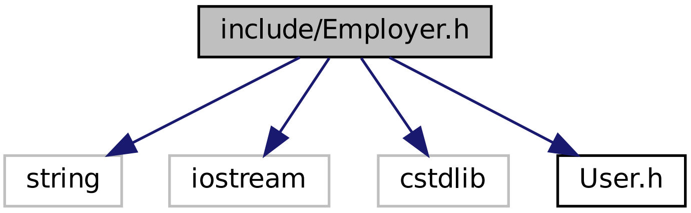
Copyright (c) 2022

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

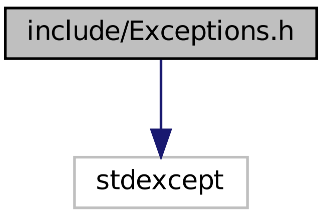
Copyright (c) 2022

5.7 include/Exceptions.h File Reference

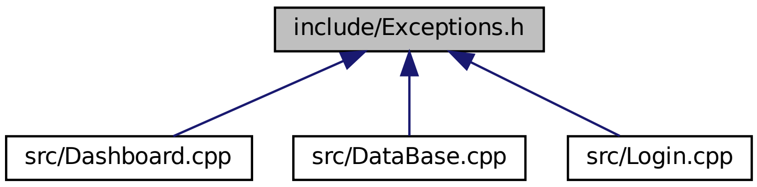
Exception that may ocurr in the interface.

```
#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 ocurr in the interface.

Author

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

Version

0.1

Date

2022-12-18

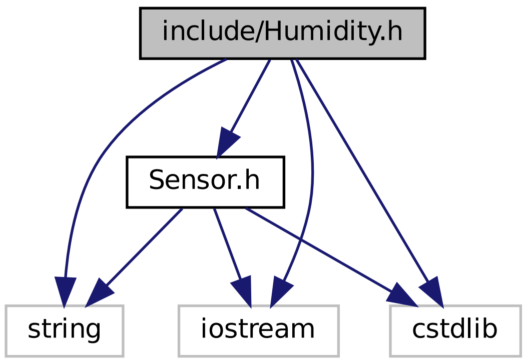
Copyright

Copyright (c) 2022

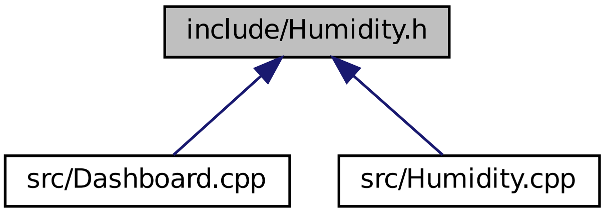
5.8 include/Humidity.h File Reference

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

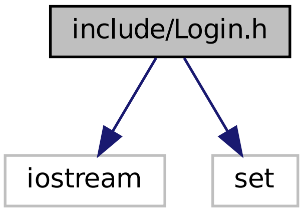
Copyright (c) 2022

5.9 include/Login.h File Reference

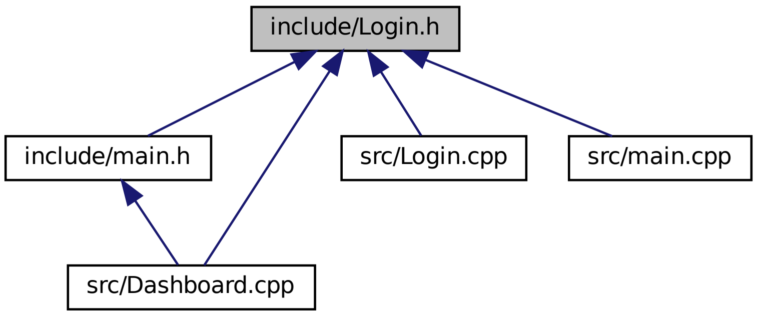
Login interface.

```
#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

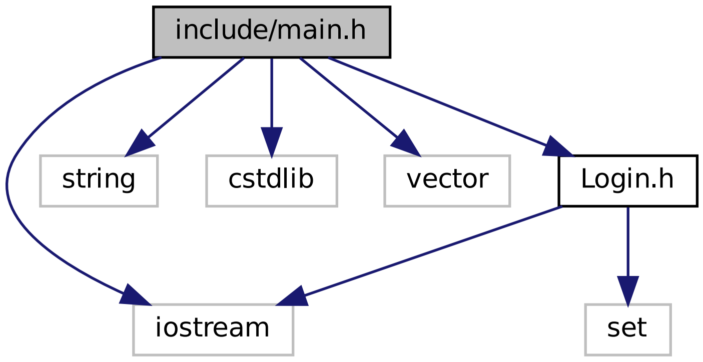
Copyright (c) 2022

5.10 include/main.h File Reference

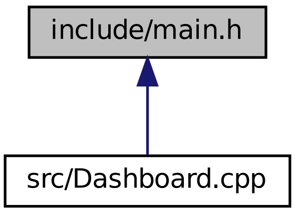
Main program.

```
#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
```

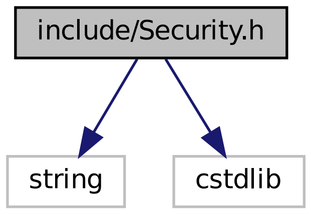
Definition at line 21 of file main.h.

5.11 include/Security.h File Reference

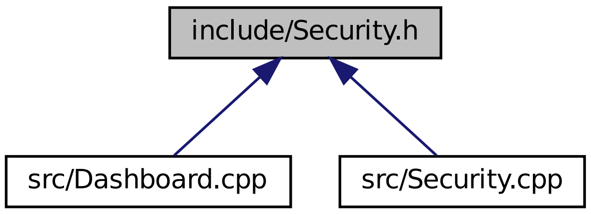
Security methods.

```
#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

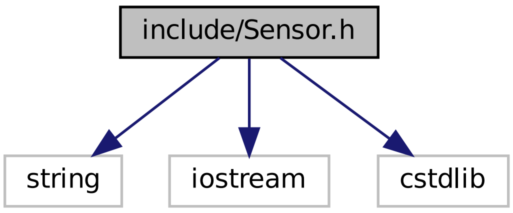
Copyright

Copyright (c) 2022

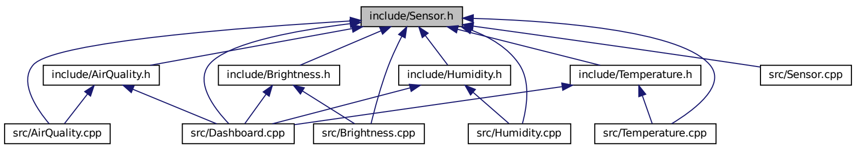
5.12 include/Sensor.h File Reference

```
#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

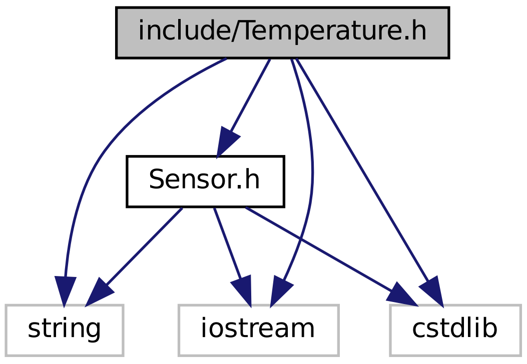
Copyright (c) 2022

5.13 include/Temperature.h File Reference

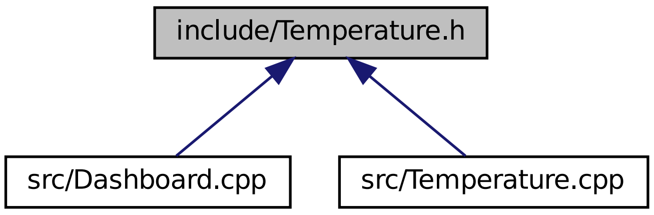
Temperature sensor.

```
#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

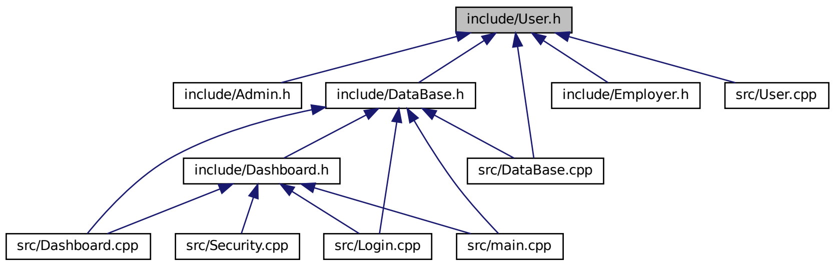
Copyright

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

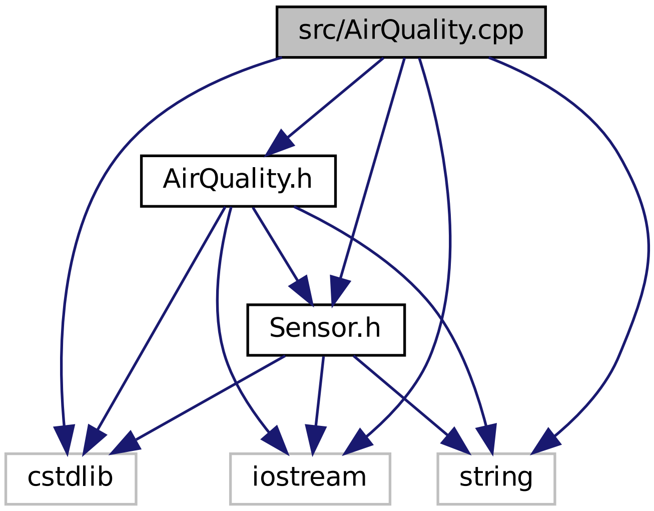
Copyright

Copyright (c) 2022

5.15 src/AirQuality.cpp File Reference

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

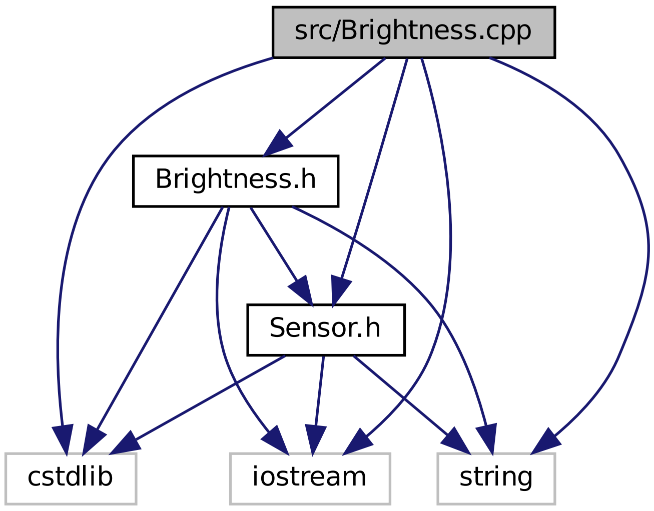
Include dependency graph for AirQuality.cpp:



5.16 src/Brightness.cpp File Reference

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

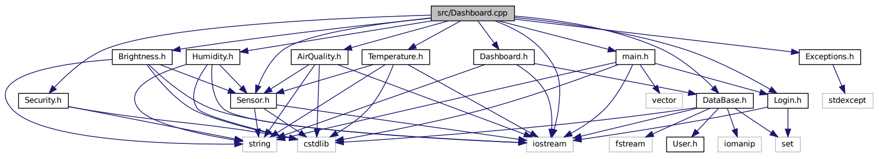
Include dependency graph for Brightness.cpp:



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

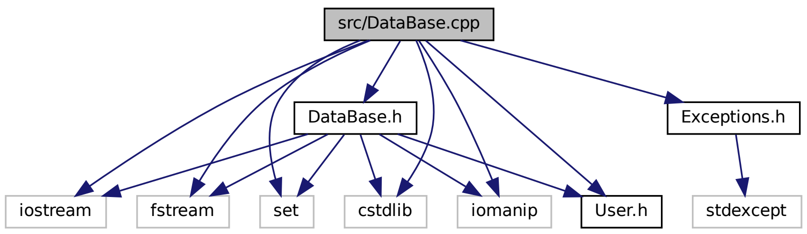
```
Temperature temp
```

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

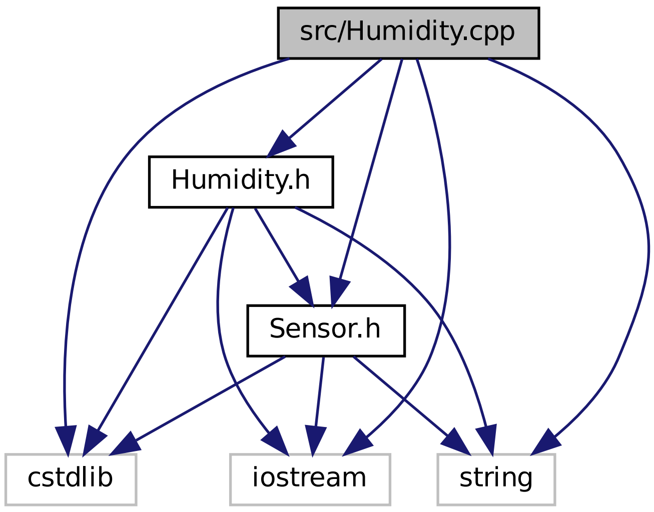
```
bool valided
```

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"
```

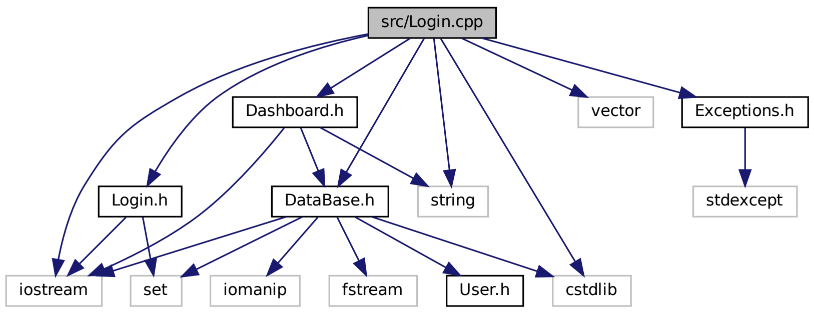
Include dependency graph for Humidity.cpp:



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

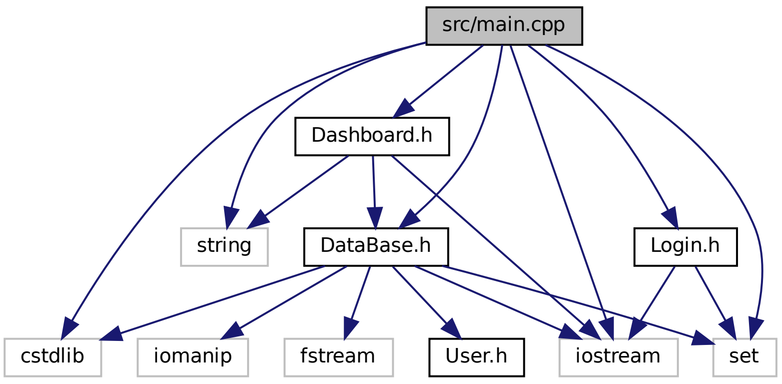
```
int temp_NIF
```

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()

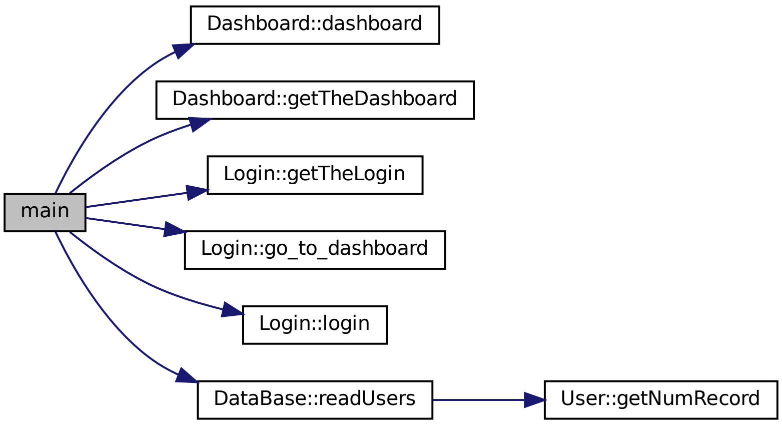
```
int main ( )
```

Definition at line 19 of file main.cpp.

```
        {
    DataBase* ptrbase;
    try {
        ptrbase = new DataBase();
    } catch (bad_alloc &except) {
    cerr « "Exception!: " « except.what() « endl;
    }

    ptrbase->readUsers();
    // a login and a dashboard objects are created
    Login* login = Login::getTheLogin ();

    Dashboard* dashboard = Dashboard::getTheDashboard ();

    while (true) {

        system("clear"); // used in lots of functions to clear the screen

        login->login(ptrbase); // initialize the login screen

        dashboard->dashboard(ptrbase, login->go_to_dashboard());
    }
    return 0;
}
```

References Dashboard::dashboard(), Dashboard::getTheDashboard(), Login::getTheLogin(), Login::go_to_↩ dashboard(), Login::login(), and DataBase::readUsers().

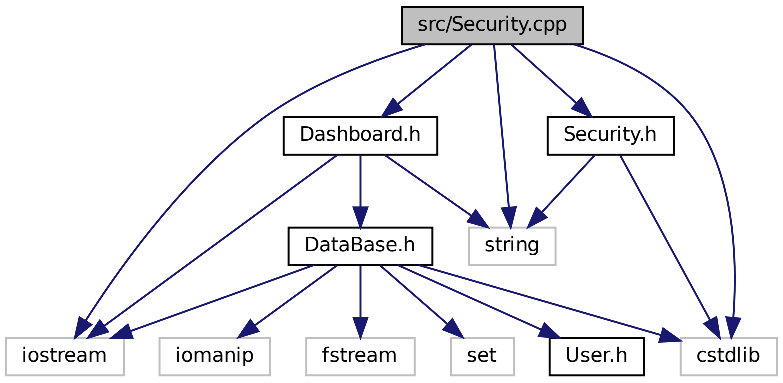
Here is the call graph for this function:



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

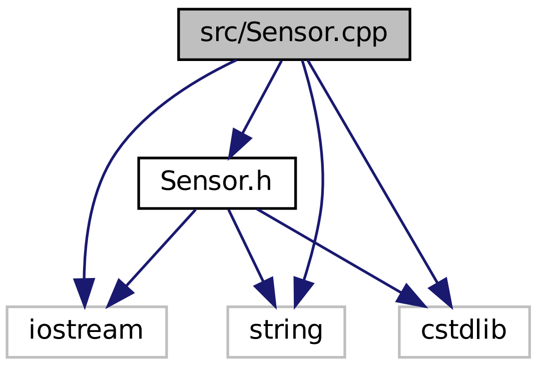
```
string seguridad
```

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

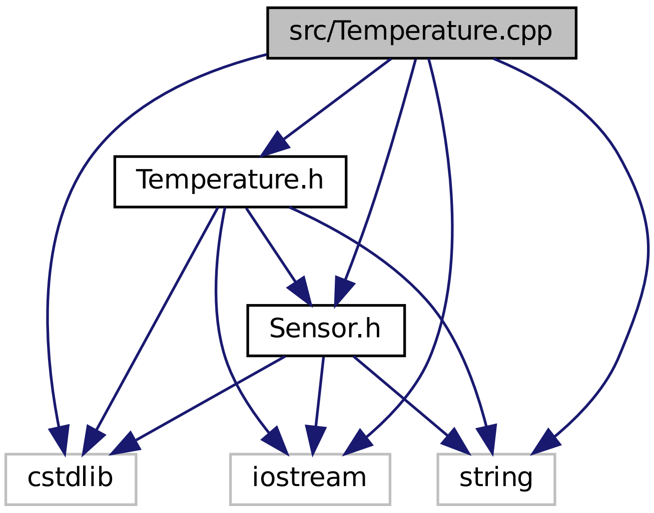
```
int back_1
```

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"
```

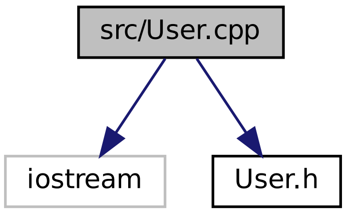
Include dependency graph for Temperature.cpp:



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